NORME INTERNATIONALE

INTERNATIONAL STANDARD

CEI
IEC
60822

Première édition
First edition
1988-12

CEI 822 VSB

Bus parallèle de sous-système du bus CEI 821 VMEbus

IEC 822 VSB

Parallel Sub-system Bus of the IEC 821 VMEbus

Numéro de référence
Reference number
CEI/IEC 60822: 1988

Numéros des publications

Depuis le 1er janvier 1997, les publications de la CEI sont numérotées à partir de 60000.

Publications consolidées

Les versions consolidées de certaines publications de la CEI incorporant les amendements sont disponibles. Par exemple, les numéros d'édition 1.0, 1.1 et 1.2 indiquent respectivement la publication de base, la publication de base incorporant l'amendement 1, et la publication de base incorporant les amendements 1 et 2.

Validité de la présente publication

Le contenu technique des publications de la CEI est constamment revu par la CEI afin qu'il reflète l'état actuel de la technique.

Des renseignements relatifs à la date de reconfirmation de la publication sont disponibles dans le Catalogue de la CEI.

Les renseignements relatifs à des questions à l'étude et des travaux en cours entrepris par le comité technique qui a établi cette publication, ainsi que la liste des publications établies, se trouvent dans les documents ci-dessous:

- **«Site web» de la CEI***
- **Catalogue des publications de la CEI**
  Publié annuellement et mis à jour régulièrement
  (Catalogue en ligne)*
- **Bulletin de la CEI**
  Disponible à la fois au «site web» de la CEI* et comme périodique imprimé

Terminologie, symboles graphiques et littéraux

En ce qui concerne la terminologie générale, le lecteur se reportera à la CEI 60050: *Vocabulaire Electrotechnique International* (VEI).

Pour les symboles graphiques, les symboles littéraux et les signes d'usage général approuvés par la CEI, le lecteur consultera la CEI 60027: *Symboles littéraux à utiliser en électrotechnique,* la CEI 60417: *Symboles graphiques utilisables sur le matériel. Index, relevé et compilation des feuilles individuelles,* et la CEI 60617: *Symboles graphiques pour schémas.*

* Voir adresse «site web» sur la page de titre.

Numbering

As from 1 January 1997 all IEC publications are issued with a designation in the 60000 series.

Consolidated publications

Consolidated versions of some IEC publications including amendments are available. For example, edition numbers 1.0, 1.1 and 1.2 refer, respectively, to the base publication, the base publication incorporating amendment 1 and the base publication incorporating amendments 1 and 2.

Validity of this publication

The technical content of IEC publications is kept under constant review by the IEC, thus ensuring that the content reflects current technology.

Information relating to the date of the reconfirmation of the publication is available in the IEC catalogue.

Information on the subjects under consideration and work in progress undertaken by the technical committee which has prepared this publication, as well as the list of publications issued, is to be found at the following IEC sources:

- **IEC web site***
- **Catalogue of IEC publications**
  Published yearly with regular updates
  (On-line catalogue)*
- **IEC Bulletin**
  Available both at the IEC web site* and as a printed periodical

Terminology, graphical and letter symbols

For general terminology, readers are referred to IEC 60050: *International Electrotechnical Vocabulary* (IEV).

For graphical symbols, and letter symbols and signs approved by the IEC for general use, readers are referred to publications IEC 60027: *Letter symbols to be used in electrical technology*, IEC 60417: *Graphical symbols for use on equipment. Index, survey and compilation of the single sheets* and IEC 60617: *Graphical symbols for diagrams.*

* See web site address on title page.

NORME INTERNATIONALE

INTERNATIONAL STANDARD

CEI
IEC
60822

Première édition
First edition
1988-12

CEI 822 VSB

Bus parallèle de sous-système du bus CEI 821 VMEbus

IEC 822 VSB

Parallel Sub-system Bus of the IEC 821 VMEbus

International Electrotechnical Commission
Telefax: +41 22 919 0300
e-mail: inmail@iec.ch
3, rue de Varembé Geneva, Switzerland
IEC web site http: //www.iec.ch

IEC
Commission Electrotechnique Internationale
International Electrotechnical Commission
Международная Электротехническая Комиссия

CODE PRIX
PRICE CODE XG

Pour prix, voir catalogue en vigueur
For price, see current catalogue

SOMMAIRE

Pages

PREAMBULE 14

PREFACE 14

CHAPITRE 0: DOMAINE D'APPLICATION

CHAPITRE 1: INTRODUCTION A LA NORME DU BUS CEI 822 VSB

Sections

1.1 Objectifs de la norme CEI 822 VSB, bus parallèle de sous-système du bus CEI 821 VMEbus (désormais référencé VSB) 18

1.2 Eléments du système VSB 18
1.2.1 Définitions générales 18
1.2.1.1 Définition de la structure physique 18
1.2.1.2 Définition de la structure fonctionnelle 20
1.2.1.3 Types de cycles VSB 24
1.3 Diagrammes de la norme VSB 30
1.4 Terminologie utilisée dans la norme 30
1.4.1 Etats des lignes de signaux 32
1.4.2 Utilisation de l'astérisque (*) 34
1.5 Spécification du protocole 34

CHAPITRE 2: BUS DE TRANSFERT DE DONNEES DU VSB

2.1 Introduction 38
2.2 Lignes du bus de transfert de données 40
2.2.1 Lignes d'adresse 40
2.2.1.1 AD00-AD31 40
2.2.1.2 SPACE0-SPACE1 42
2.2.1.3 SIZE0-SIZE1 42
2.2.1.4 ASACK0*-ASACK1* 42
2.2.1.5 GA0-GA2 44
2.2.2 Lignes de données AD00-AD31 44
2.2.3 Lignes de commande 44
2.2.3.1 PAS* 44
2.2.3.2 AC 46
2.2.3.3 WR* 46
2.2.3.4 LOCK* 46
2.2.3.5 DS* 46
2.2.3.6 WAIT* 46
2.2.3.7 ACK* 48
2.2.3.8 ERR* 48
2.2.3.9 IRQ* 48
2.2.3.10 CACHE* 50
2.3 Modules du DTB - Description générale 50
2.3.1 MAITRE 52
2.3.2 ESCLAVE 54

CONTENTS

Page

FOREWORD 15

PREFACE 15

CHAPTER 0: SCOPE

CHAPTER 1: INTRODUCTION TO THE IEC 822 VSB BUS STANDARD

Section

1.1 Standard objectives of the IEC 822 VSB parallel Subsystem Bus of the IEC 821 VMEbus (Subsystem henceforth referred to as VSB) 19
1.2 VSB system elements 19
1.2.1 Basic definitions 19
1.2.1.1 Physical structure definition 19
1.2.1.2 Functional structure definition 21
1.2.1.3 Types of VSB cycles 25
1.3 VSB standard diagrams 31
1.4 Standard terminology 31
1.4.1 Signal line states 33
1.4.2 Use of the asterisk (*) 35
1.5 Protocol specification 35

CHAPTER 2: VSB DATA TRANSFER BUS

2.1 Introduction 39
2.2 Data Transfer Bus lines 41
2.2.1 Adressing lines 41
2.2.1.1 AD00-AD31 41
2.2.1.2 SPACE0-SPACE1 43
2.2.1.3 SIZE0-SIZE1 43
2.2.1.4 ASACK0*-ASACK1* 43
2.2.1.5 GA0-GA2 45
2.2.2 Data lines AD00-AD31 45
2.2.3 Control lines 45
2.2.3.1 PAS* 45
2.2.3.2 AC 47
2.2.3.3 WR* 47
2.2.3.4 LOCK* 47
2.2.3.5 DS* 47
2.2.3.6 WAIT* 47
2.2.3.7 ACK* 49
2.2.3.8 ERR* 49
2.2.3.9 IRQ* 49
2.2.3.10 CACHE* 51
2.3 DTB modules - Basic description 51
2.3.1 MASTER 53
2.3.2 SLAVE 55

Sections Pages

2.4 Possibilités des MAITRES et des ESCLAVES 56
2.4.1 Possibilités d'adressage 60
2.4.1.1 Possibilités de base pour l'adressage 62
2.4.1.2 Possibilité UNIQUEMENT D'ADRESSAGE 64
2.4.2 Possibilités de transfert de données 66
2.4.2.1 Possibilité de base de transfert de données des MAITRES 66
2.4.2.2 Possibilités de base de transferts de données des ESCLAVES .. 68
2.4.2.3 Dimensionnement dynamique du bus 70
2.4.2.4 Possibilité de TRANSFERT UNIQUE 72
2.4.2.5 Possibilité de TRANSFERT PAR BLOC 74
2.4.2.6 Possibilité de TRANSFERT INDIVISIBLE 78
2.4.3 Possibilités d'interruption 82
2.4.3.1 Possibilités d'interruption de base des MAITRES et des ESCLAVES 82
2.4.3.2 Possibilités de cycle de RECONNAISSANCE D'INTERRUPTION 86
2.5 Interaction entre les MAITRES et les ESCLAVES 90
2.5.1 Interaction entre les MAITRES et les ESCLAVES pendant la phase de diffusion d'adresse 92
2.5.1.1 Déroulement de la phase de diffusion d'adresse 92
2.5.1.2 Evolution des signaux pendant la phase de diffusion d'adresse 98
2.5.2 Interaction entre les MAITRES et les ESCLAVES pendant le transfert de données 104
2.5.2.1 Déroulement d'un transfert de données en écriture 106
2.5.2.2 Déroulement d'un transfert de données en lecture 112
2.5.2.3 Evolution des signaux pendant la phase de transfert de données 116
2.5.3 Interaction entre les MAITRES et les ESCLAVES pendant la fin du cycle 124
2.5.3.1 Déroulement de la fin d'un cycle 124
2.5.4 Interaction entre le MAITRE IHV et les ESCLAVES pendant le cycle de RECONNAISSANCE D'INTERRUPTION 126
2.5.4.1 Organigramme d'un cycle de RECONNAISSANCE D'INTERRUPTION 128
2.5.4.2 Evolution des signaux pendant le cycle de RECONNAISSANCE D'INTERRUPTION 136
2.6 Spécifications de chronologie du bus de transfert de données 138

CHAPITRE 3: ARBITRAGE DU BUS DE TRANSFERT DE DONNEES DU VSB

3.1 Introduction 188
3.1.1 Types d'arbitrage 190
3.2 Lignes d'arbitrage du bus 190
3.2.1 BREQ* 190
3.2.2 BUSY* 190
3.2.3 BGIN*/BGOUT* 192
3.3 Modules d'arbitrage - Description générale 192
3.3.1 ARBITRE 192
3.3.2 DEMANDEUR 194
3.4 Possibilités du DEMANDEUR 198
3.4.1 Arbitrage série 200
3.4.1.1 Interaction entre l'ARBITRE et les DEMANDEURS SER 202
3.4.1.2 Evolution des signaux pendant l'arbitrage série 208

Section		Page
2.4	Capabilities of MASTERS and SLAVES	57
2.4.1	Addressing capabilities	61
2.4.1.1	Basic addressing capabilities	63
2.4.1.2	ADDRESS-ONLY capability	65
2.4.2	Data transfer capabilities	67
2.4.2.1	Basic data transfer capability of MASTERS	67
2.4.2.2	Basic data transfer capabilities of SLAVES	69
2.4.2.3	Dynamic bus sizing	71
2.4.2.4	SINGLE-TRANSFER capability	73
2.4.2.5	BLOCK-TRANSFER capability	75
2.4.2.6	INDIVISIBLE-ACCESS capability	79
2.4.3	Interrupt capability	83
2.4.3.1	Basic interrupt capabilities of MASTERS and SLAVES	83
2.4.3.2	INTERRUPT-ACKNOWLEDGE cycle capabilities	87
2.5	Interaction between MASTERS and SLAVES	91
2.5.1	Interaction between MASTERS and SLAVES during address broadcast phase	93
2.5.1.1	Flow of the address broadcast phase	93
2.5.1.2	Signaling during the address broadcast phase	99
2.5.2	Interaction between MASTERS and SLAVES during the data transfer	105
2.5.2.1	Flow of a write data transfer	107
2.5.2.2	Flow of a read data transfer	113
2.5.2.3	Signaling during the data transfer phase	117
2.5.3	Interaction between MASTERS and SLAVES during cycle termination	125
2.5.3.1	Flow of the termination of a cycle	125
2.5.4	Interaction between the IHV MASTER and SLAVES during the INTERRUPT-ACKNOWLEDGE cycles	127
2.5.4.1	Flow of an INTERRUPT-ACKNOWLEDGE cycle	129
2.5.4.2	Signaling during the INTERRUPT-ACKNOWLEDGE cycle	137
2.6	Data transfer bus timing specifications	139

CHAPTER 3: VSB DATA TRANSFER BUS ARBITRATION

3.1	Introduction	189
3.1.1	Types of Arbitration	191
3.2	Arbitration Bus lines	191
3.2.1	BREQ*	191
3.2.2	BUSY*	191
3.2.3	BGIN*/BGOUT*	193
3.3	Arbitration modules - Basic description	193
3.3.1	ARBITER	193
3.3.2	REQUESTER	195
3.4	Capabilities of the REQUESTER	199
3.4.1	Serial Arbitration	201
3.4.1.1	Interaction between the ARBITER and SER REQUESTERS	203
3.4.1.2	Signaling during Serial Arbitration	209

Sections Pages

3.4.2 Possibilités de l'arbitrage parallèle 212
3.4.2.1 Déroulement d'un cycle d'ARBITRAGE 212
3.4.2.2 Evolution des signaux pendant le cycle d'ARBITRAGE 218
3.4.3 Séquence de mise sous tension 220
3.4.3.1 Déroulement d'une séquence de mise sous tension 220
3.4.3.2 Interaction entre les modules du bus d'arbitrage pendant le démarrage 226
3.5 Interaction entre le MAITRE, son DEMANDEUR associé et/ou son ARBITRE associé 228
3.5.1 Acquisition du DTB 228
3.5.2 Libération du DTB 228
3.5.3 Course critique entre les demandes du MAITRE et les allocations de l'ARBITRE 230
3.6 Spécifications de chronologie du bus d'arbitrage 230

CHAPITRE 4: CARACTERISTIQUES ELECTRIQUES DES CARTES VSB

4.1 Introduction 252
4.1.1 Terminologie 252
4.2 Distribution de l'alimentation 256
4.2.1 Caractéristiques de tension courant continu 256
4.2.2 Caractéristiques électriques du connecteur 256
4.3 Spécifications de commande et de réception du bus 256
4.3.1 Généralités 256
4.3.2 REGLES de commande et de charge pour les lignes trois états (AD00-AD31, DS*, PAS*, LOCK*, SIZE0-SIZE1, SPACE0-SPACE1, WR*) 260
4.3.3 REGLES de commande et de charge pour les lignes à collecteur ouvert (AC, ACK*, AD24-AD31, ASACK0*-ASACK1*, BREQ*, BUSY*, CACHE*, ERR*, IRQ*, WAIT*) 264
4.3.4 REGLES de commande et de charge pour BGIN* et BGOUT* 268
4.3.5 REGLES de réception pour les lignes d'adressage géographique (GA0-GA2) 270
4.3.6 Informations supplémentaires 270
4.4 Interconnexion des lignes de signal - Résumé 272

CHAPITRE 5: SPECIFICATIONS DU FOND DE PANIER DU VSB

5.1 Introduction 276
5.2 Caractéristiques physiques du fond de panier 276
5.3 Distribution du courant d'alimentation 280
5.4 Caractéristiques électriques du fond de panier 280
5.4.1 Impédance caractéristique 280
5.4.2 Réseaux d'adaptation 288
5.5 Interconnexion des lignes de signaux 292
5.5.1 Généralités 292
5.5.2 Chaîne série BGIN*/BGOUT* 294
5.5.3 Adressage géographique 294
5.5.4 Informations supplémentaires 296
5.6 Affectation des broches VSB 296

ANNEXE A 300

Section Page

3.4.2 Parallel Arbitration capability 213
3.4.2.1 Flow of an ARBITRATION cycle 213
3.4.2.2 Signaling during the ARBITRATION cycle 219
3.4.3 Power-up sequence 221
3.4.3.1 Flow of the power-up sequence 221
3.4.3.2 Interaction between arbitration bus modules during power-up . 227

3.5 Interaction between the MASTER, its associated REQUESTER and/or its associated ARBITER 229
3.5.1 Acquisition of the DTB 229
3.5.2 Release of the DTB 229
3.5.3 Race conditions between MASTER requests and ARBITER grants .. 231

3.6 Arbitration bus timing specifications..... 231

CHAPTER 4: ELECTRICAL CHARACTERISTICS OF VSB BOARDS

4.1 Introduction 253
4.1.1 Terminology 253
4.2 Power distribution 257
4.2.1 D.C. voltage characteristics 257
4.2.2 Connector electrical ratings 257
4.3 Bus driving and receiving requirements 257
4.3.1 General 257
4.3.2 Driving and loading RULES for three-state lines (AD00-AD31, DS*, PAS*, LOCK*, SIZE0-SIZE1, SPACE0-SPACE1, WR*) 261
4.3.3 Driving and loading RULES for open-collector lines (AC, ACK*, AD24-AD31, ASACK0*-ASACK1*, BREQ*, BUSY*, CACHE*, ERR*, IRQ*, WAIT*) 265
4.3.4 Driving and loading RULES for BGIN* and BGOUT* 269
4.3.5 Receiving RULES for the geographical addressing lines (GA0-GA2) 271
4.3.6 Additional information 271
4.4 Signal lines interconnection - Summary 273

CHAPTER 5: VSB BACKPLANE SPECIFICATIONS

5.1 Introduction 277
5.2 Backplane physical characteristics 277
5.3 Power distribution 281
5.4 Backplane electrical characteristics 281
5.4.1 Characteristic impedance 281
5.4.2 Termination networks 289
5.5 Signal line interconnection 293
5.5.1 General 293
5.5.2 BGIN*/BGOUT* daisy-chain 295
5.5.3 Geographical addressing 295
5.5.4 Additional information 297
5.6 VSB pin assignment 297

APPENDIX A 301

Figures Pages

1-1 Modules fonctionnels et sous-ensembles de bus définis par la norme VSB 22
1-2 Notations utilisées dans les chronogrammes 36
2-1 Schéma-bloc fonctionnel du bus de transfert de données 38
2-2 Schéma-bloc: MAITRE 52
2-3 Schéma-bloc: ESCLAVE 54
2-4 Organigramme général d'un cycle VSB 58
2-5 Organigramme général d'un cycle UNIQUEMENT D'ADRESSAGE 64
2-6 Organisation des données 66
2-7 Organigramme général d'un cycle de TRANSFERT UNIQUE 72
2-8 Organigramme général d'un cycle de TRANSFERT PAR BLOC 76
2-9 Organigramme général d'un cycle de RECONNAISSANCE D'INTERRUPTION 86
2-10 Organigramme de la phase de diffusion d'adresse 96
2-11 Organigramme d'un transfert de données en écriture 110
2-12 Organigramme d'un transfert de données en lecture 114
2-13 Organigramme de la fin du cycle 126
2-14 Organigramme d'un cycle de RECONNAISSANCE D'INTERRUPTION 132
2-15 Chronologie des signaux LOCK*, WR*, SIZE0-SIZE1 et SPACE0-SPACE1, d'un MAITRE actif, d'un MAITRE IHV actif et d'un DEMANDEUR PAR actif, pour les cycles de TRANSFERT UNIQUE, TRANSFERT PAR BLOC, RECONNAISSANCE D'INTERRUPTION et ARBITRAGE 146
2-16 Chronologie de la diffusion d'adresse du MAITRE actif et des ESCLAVES pour les cycles UNIQUEMENT D'ADRESSAGE, TRANSFERT UNIQUE et TRANSFERT PAR BLOC 148
2-17 Fin de cycle du MAITRE actif et des ESCLAVES pour les cycles UNIQUEMENT D'ADRESSAGE 150
2-18 Chronologie d'un transfert de données en écriture du MAITRE actif et des ESCLAVES pour les cycles de TRANSFERT UNIQUE et TRANSFERT PAR BLOC 152
2-19 Chronologie d'un transfert de données en lecture du MAITRE actif et des ESCLAVES pour les cycles de TRANSFERT UNIQUE, TRANSFERT PAR BLOC et RECONNAISSANCE D'INTERRUPTION 156
2-20 Phase de sélection du MAITRE IHV et des ESCLAVES INTV pour les cycles de RECONNAISSANCE D'INTERRUPTION 160
2-21 Chronologie des MAITRES et des ESCLAVES entre les cycles 162
2-22 Chronologie du transfert de contrôle du DTB 164
2-23 Déphasage entre ASACK0* et ASACK1* 166
2-24 Déphasage entre ACK* et ERR* 166
3-1 Schéma-bloc fonctionnel du bus d'arbitrage 188
3-2 Schéma-bloc: ARBITRE 194
3-3 Schéma-bloc: DEMANDEUR SER 196
3-4 Schéma-bloc: DEMANDEUR PAR 198
3-5 Organigramme de l'arbitrage série: deux DEMANDEURS 204
3-6 Organigramme général d'un cycle d'ARBITRAGE 212
3-7 Organigramme d'un cycle d'ARBITRAGE 216
3-8 Organigramme de la séquence de démarrage 224
3-9 DEMANDEUR PAR actif, DEMANDEUR PAR concurrent et ESCLAVE au repos: cycle d'ARBITRAGE 236
3-10 Chronogramme du démarrage à la mise sous tension 238

Figure Page

1-1 Functional modules and sub-buses defined by the VSB standard ... 23

1-2 Signal timing notation ... 37
2-1 Data Transfer Bus functional block diagram ... 39
2-2 Block diagram: MASTER ... 53
2-3 Block diagram: SLAVE ... 55
2-4 General flow of a VSB cycle ... 59
2-5 General flow of an ADDRESS-ONLY cycle ... 65
2-6 Organization of data ... 67
2-7 General flow of a SINGLE-TRANSFER cycle ... 73
2-8 General flow of a BLOCK-TRANSFER cycle ... 77
2-9 General flow of an INTERRUPT-ACKNOWLEDGE cycle ... 87
2-10 Flow of the address broadcast phase ... 97
2-11 Flow of a write data transfer ... 111
2-12 Flow of a read data transfer ... 115
2-13 Flow of the termination of the cycle ... 127
2-14 Flow of an INTERRUPT-ACKNOWLEDGE cycle ... 133
2-15 Active MASTER, active IHV MASTER and active PAR REQUESTER, LOCK*, WR*, SIZE0-SIZE1 and SPACE0-SPACE1 timing, SINGLE-TRANSFER, BLOCK-TRANSFER, INTERRUPT-ACKNOWLEDGE and ARBITRATION cycles ... 147
2-16 Active MASTER and SLAVES, address broadcast timing, ADDRESS-ONLY, SINGLE-TRANSFER and BLOCK-TRANSFER cycles ... 149
2-17 Active MASTER and SLAVES, cycle termination ADDRESS-ONLY cycles ... 151
2-18 Active MASTER and SLAVES, write data transfer timing, SINGLE-TRANSFER and BLOCK-TRANSFER cycles ... 153
2-19 Active MASTER and SLAVES, read data transfer timing, SINGLE-TRANSFER, BLOCK-TRANSFER and INTERRUPT-ACKNOWLEDGE cycles ... 157
2-20 IHV MASTER and INTV SLAVES, selection phase INTERRUPT-ACKNOWLEDGE cycles ... 161
2-21 MASTERS and SLAVES intercycle timing ... 163
2-22 DTB control transfer timing ... 165
2-23 Skew between ASACK0* and ASACK1* ... 167
2-24 Skew between ACK* and ERR* ... 167
3-1 Arbitration bus functional block diagram ... 189
3-2 Block diagram: ARBITER ... 195
3-3 Block diagram: SER REQUESTER ... 197
3-4 Block diagram: PAR REQUESTER ... 199
3-5 Serial Arbitration flow diagram: two REQUESTERS ... 205
3-6 General flow of an ARBITRATION cycle ... 213
3-7 Flow of an ARBITRATION cycle ... 217
3-8 Flow of the power-up sequence ... 225
3-9 Active PAR REQUESTER, contending PAR REQUESTER and idle SLAVE ARBITRATION cycle ... 237
3-10 Power-up timing ... 239

Figures

		Pages
4-1	Niveaux des signaux VSB	258
5-1	Dimensions du fond de panier VSB	278
5-2	Section transversale du microruban d'une ligne de signal du fond de panier	282
5-3	Z_o en fonction de la largeur de piste	284
5-4	C_o en fonction de la largeur de piste	284
5-5	Adaptation standard du bus	290
5-6	Illustration de la chaîne série BGIN*/BGOUT*	294
5-7	Circuit résistance/capacité des lignes d'adressage géographique	294
A1	Organigramme de la phase de sélection	302
A2	Commande de la phase de sélection; schéma-bloc général	304
A3	Un exemple pour la logique de sélection	306

Tableaux

2-1	REGLES et AUTORISATIONS qui spécifient l'utilisation des lignes pointillées par les différents types de MAITRES	52
2-2	REGLES et AUTORISATIONS qui spécifient l'utilisation des lignes pointillées par les différents types d'ESCLAVES	54
2-3	Mnémoniques qui spécifient les possibilités d'adressage	62
2-4	Mnémonique qui spécifie la possibilité UNIQUEMENT D'ADRESSAGE	64
2-5	Mnémoniques qui spécifient les possibilités de base de transferts de données des ESCLAVES	68
2-6	Mnémonique qui spécifie la possibilité de TRANSFERT PAR BLOC	78
2-7	Mnémoniques qui spécifient les possibilités d'interruption	84
2-8	Mnémoniques qui spécifient les possibilités de transfert de MOT D'ETAT/ID des MAITRES IHV et des ESCLAVES INTV	90
2-9	Utilisation de SPACE0 et SPACE1 pour sélectionner l'espace d'adresse	98
2-10	Codage de SIZE0 et SIZE1 pour une dimension requise du transfert	100
2-11	Utilisation de AD00 et AD01 pour sélectionner l'emplacement de l'octet d'adresse la plus basse à atteindre	100
2-12	Codage de SIZE0, SIZE1, AD00 et AD01 pour définir les emplacements d'octet à atteindre	102
2-13	Codage de ASACK0* et ASACK1* pour définir la dimension de l'ESCLAVE	104
2-14	Positionnement des données valides sur AD00-AD31 par le MAITRE actif pendant les cycles d'écriture	116
2-15	Utilisation de AD00-AD31 par un ESCLAVE D32 pour accéder aux emplacements d'octet	118
2-16	Utilisation de AD16-AD31 par un ESCLAVE D16 pour accéder aux emplacements d'octet	120
2-17	Utilisation de AD24-AD31 par un ESCLAVE D08 pour accéder aux emplacements d'octet	120
2-18	Utilisation de SPACE0, SPACE1 et WR* pour sélectionner un cycle de RECONNAISSANCE D'INTERRUPTION	136
2-19	Utilisation des lignes de données par les ESCLAVES INTV D08, D16 et D32 pendant les cycles de RECONNAISSANCE D'INTERRUPTION	138
2-20	Paramètres de temps d'un MAITRE actif, d'un ESCLAVE répondant, d'un ESCLAVE participant et d'un ESCLAVE au repos	142
2-21	Paramètres de temps d'un MAITRE IHV, d'un ESCLAVE INTV répondant, d'un ESCLAVE INTV concurrent et d'un ESCLAVE au repos	144
2-22	MAITRE, spécifications de la chronologie	168
2-23	ESCLAVE, spécifications de la chronologie	178

Figure

		Page
4-1	VSB signal levels	259
5-1	VSB backplane dimensions	279
5-2	Cross-section of a backplane microstrip signal line	283
5-3	Z_o versus line width	285
5-4	C_o versus line width	285
5-5	Standard bus termination	291
5-6	BGIN*/BGOUT* daisy-chain illustration	295
5-7	Geographical addressing lines resistor/capacitor circuit	295
A1	Flow of the selection phase	303
A2	Selection phase control; a high level block diagram	305
A3	An example for the selection logic	307

Table

2-1	RULES and PERMISSIONS that specify the use of the dotted lines by the various types of MASTERS	53
2-2	RULES and PERMISSIONS that specify the use of the dotted lines by the various types of SLAVES	55
2-3	Mnemonics that specify addressing capabilities	63
2-4	Mnemonic that specifies ADDRESS-ONLY capability	65
2-5	Mnemonics that specify the basic data transfer capabilities of SLAVES	69
2-6	Mnemonic that specifies BLOCK-TRANSFER capability	79
2-7	Mnemonics that specify interrupt capabilities	85
2-8	Mnemonics that specify STATUS/ID transfer capabilities of IHV MASTERS and INTV SLAVES	91
2-9	Use of SPACE0 and SPACE1 to select the address space	99
2-10	Encoding of SIZE0 and SIZE1 for requested size of the transfer	101
2-11	Use of AD00 and AD01 to select the lowest addressed byte location to be accessed	101
2-12	Encoding of SIZE0, SIZE1, AD00 and AD01 to define the byte locations to be accessed	103
2-13	Encoding of ASACK0* and ASACK1* to define the size of the SLAVE	105
2-14	Placement of valid data on AD00-AD31 by the active MASTER during write cycles	117
2-15	Use of AD00-AD31 by a D32 SLAVE to access byte locations	119
2-16	Use of AD16-AD31 by a D16 SLAVE to access byte locations	121
2-17	Use of AD24-AD31 by a D08 SLAVE to access byte locations	121
2-18	Use of SPACE0, SPACE1 and WR* to select an INTERRUPT-ACKNOWLEDGE cycle	137
2-19	Use of the data lines by D08, D16 and D32 INTV SLAVES during INTERRUPT-ACKNOWLEDGE cycles	139
2-20	Active MASTER, responding SLAVE, participating SLAVE and idle SLAVE timing parameters	143
2-21	IHV MASTER, responding INTV SLAVE, contending INTV SLAVE and idle SLAVE timing parameters	145
2-22	MASTER, timing specifications	169
2-23	SLAVE, timing specifications	179

Tableaux Pages

3-1 REGLES et AUTORISATIONS spécifiant l'utilisation des lignes pointillées par les différents types de DEMANDEURS SER 196
3-2 Mnémoniques utilisés pour décrire les DEMANDEURS 200
3-3 Utilisation de SPACE0-SPACE1 et WR* pour sélectionner un cycle d'ARBITRAGE 218
3-4 Paramètres de temps d'un DEMANDEUR PAR actif, d'un DEMANDEUR PAR concurrent et d'un ESCLAVE au repos 232
3-5 Paramètres de temps à la mise sous tension 234
3-6 Spécifications de chronologie du DEMANDEUR actif 240
3-7 Spécifications de chronologie des DEMANDEURS concurrents 244
3-8 Spécifications de chronologie à la mise sous tension 248
4-1 Spécifications de commande et de réception du bus 260
4-2 Interconnexion des lignes de signal - Résumé 274
5-1 Spécification des tensions du bus 280
5-2 Adaptation des lignes de signaux 292
5-3 Affectation des emplacements de l'adressage géographique 296
5-4 Affectation des broches VSB 298

Table Page

3-1 RULES and PERMISSIONS that specify the use of the dotted lines by the various types of SER REQUESTERS 197
3-2 Mnemonics that are used to describe REQUESTERS 201
3-3 Use of SPACE0-SPACE1 and WR* to select an ARBITRATION cycle 219
3-4 Active PAR REQUESTER, contending PAR REQUESTER and idle SLAVE timing parameters 233
3-5 Power-up timing parameters 235
3-6 Active REQUESTER timing specifications 241
3-7 Contending REQUESTER timing specifications 245
3-8 Power-up timing specifications 249
4-1 Bus driving and receiving requirements 261
4-2 Signal line interconnection - Summary 275
5-1 Bus voltage specification 281
5-2 Signal line termination 293
5-3 Geographical addressing slot assignment 297
5-4 VSB pin assignment 299

COMMISSION ELECTROTECHNIQUE INTERNATIONALE

CEI 822 VSB

BUS PARALLELE DE SOUS-SYSTEME DU BUS CEI 821 VMEbus

PREAMBULE

1) Les décisions ou accords officiels de la CEI en ce qui concerne les questions techniques, préparés par des Comités d'Etudes où sont représentés tous les Comités nationaux s'intéressant à ces questions, expriment dans la plus grande mesure possible un accord international sur les sujets examinés.

2) Ces décisions constituent des recommandations internationales et sont agréées comme telles par les Comités nationaux.

3) Dans le but d'encourager l'unification internationale, la CEI exprime le voeu que tous les Comités nationaux adoptent dans leurs règles nationales le texte de la recommandation de la CEI, dans la mesure où les conditions nationales le permettent. Toute divergence entre la recommandation de la CEI et la règle nationale correspondante doit, dans la mesure du possible, être indiquée en termes clairs dans cette dernière.

4) La CEI n'a fixé aucune procédure concernant le marquage comme indication d'approbation et sa responsabilité n'est pas engagée quand il est déclaré qu'un matériel est conforme à l'une de ses recommandations.

PREFACE

La présente norme a été établie par le Sous-Comité 47B: Systèmes à microprocesseurs, du Comité d'Etudes n° 47 de la CEI: Dispositifs à semiconducteurs.

Le texte de cette norme est issu des documents suivants:

Règle des Six Mois	Rapport de vote
47B(BC)22	47B(BC)27

Pour de plus amples renseignements, consulter le rapport de vote mentionné dans le tableau ci-dessus.

Les publications suivantes de la CEI sont citées dans la présente norme:

Publications n^{os} 603-2 (1980): Connecteurs pour fréquences inférieures à 3 MHz pour utilisation avec cartes imprimées, Deuxième partie: Connecteurs pour circuits imprimés en deux parties, pour grille de base de 2,54 mm (0,1 in) avec caractéristiques de montage communes.

821 (1987): BUS CEI 821 - Bus système à microprocesseurs pour données de 1 à 4 octets.

INTERNATIONAL ELECTROTECHNICAL COMMISSION

IEC 822 VSB

PARALLEL SUB-SYSTEM BUS OF THE IEC 821 VMEbus

FOREWORD

1) The formal decisions or agreements of the IEC on technical matters, prepared by Technical Committees on which all the National Committees having a special interest therein are represented, express, as nearly as possible, an international consensus of opinion on the subjects dealt with.

2) They have the form of recommendations for international use and they are accepted by the National Committees in that sense.

3) In order to promote international unification, the IEC expresses the wish that all National Committees should adopt the text of the IEC recommendation for their national rules in so far as national conditions will permit. Any divergence between the IEC recommendation and the corresponding national rules should, as far as possible, be clearly indicated in the latter.

4) The IEC has not laid down any procedure concerning marking as an indication of approval and has no responsibility when an item of equipment is declared to comply with one of its recommendations.

PREFACE

This standard has been prepared by Sub-Committee 47B: Microprocessor Systems, of IEC Technical Committee No. 47: Semiconductor Devices.

The text of this standard is based on the following documents:

Six Months' Rule	Report on Voting
47B(CO)22	47B(CO)27

Further information can be found in the Report on Voting indicated in the table above.

The following IEC publications are quoted in this standard:

Publications Nos. 603-2 (1980): Connectors for frequencies below 3 MHz for use with printed boards, Part 2: Two-part connectors for printed boards, for basic grid of 2.54 mm (0.1 in) with common mounting features.

821 (1987): IEC 821 BUS - Microprocessor system bus for 1 to 4 byte data.

CEI 822 VSB

BUS PARALLELE DE SOUS-SYSTEME DU BUS CEI 821 VMEbus

CHAPITRE 0: DOMAINE D'APPLICATION

L'introduction de microprocesseurs 32 bits de hautes performances, de même que la demande de la communauté des utilisateurs dans le domaine des micro-ordinateurs ont créé un besoin de systèmes multi-processeurs construits à partir d'ensembles de cartes. L'accroissement du nombre de fonctions que de tels systèmes peuvent offrir a nécessité l'introduction d'un bus de sous-système performant. Le VSB (VME Subsystem Bus) a été conçu pour répondre à ces exigences.

Il inclut un bus asynchrone de transfert de données à haute vitesse qui permet à des maîtres de diriger des transferts de données binaires vers ou depuis des esclaves. Le maître initialise les cycles de bus de façon à transférer les données entre lui-même et les esclaves. L'esclave détecte les cycles de bus qui sont déclenchés par le maître actif et, quand il a été selectionné au cours de ces cycles, transfère les données entre lui-même et le maître.

Quatre types de cycles ont été définis: un cycle uniquement d'adressage, un cycle de transfert unique, un cycle de transfert par bloc et un cycle de reconnaissance d'interruption. Pour maximaliser le taux de transfert dans les systèmes multiprocesseurs, la norme VSB définit un mécanisme qui permet au maître de diffuser des données à un nombre quelconque d'esclaves au cours d'un cycle unique. De plus, le mécanisme de transfert de données supporte le dimensionnement dynamique du bus aussi bien que le verrouillage de ressource ou l'utilisation de mémoire cache.

Le bus d'arbitrage est le second des deux sous-ensembles définis dans la norme VSB. Il permet à des modules arbitres et/ou à des modules demandeurs de coordonner l'usage du bus de transfert de données. Deux méthodes d'arbitrage sont définies - une méthode d'arbitrage série (chaîne série) et une méthode parallèle (distribuée). Ces méthodes d'arbitrage fournissent des protocoles pour mettre en place des ensembles de sous-systèmes d'architectures différentes. En utilisant la méthode d'arbitrage série, un concepteur peut définir un sous-système à maître unique incluant une seule carte processeur accédant à un volume important de mémoire. Cette méthode peut être utilisée pour construire un système donnant la priorité à un maître primaire qui, lorsqu'il le peut, accorde le bus à d'autres maîtres secondaires. A l'inverse, un sous-système multiprocesseur peut être défini en utilisant la méthode d'arbitrage parallèle.

IEC 822 VSB

PARALLEL SUB-SYSTEM BUS OF THE IEC 821 VMEbus

CHAPTER 0: SCOPE

The introduction of high performance of 32-bit microprocessors, as well as the demands placed on microcomputers by the user community have created a need for multiprocessor systems built from board level products. The increase in the number of functions that such systems provided necessitated the introduction of a sophisticated subsystem bus. The VSB (VME Subsystem Bus) was designed to respond to these requirements.

It includes a high speed asynchronous data transfer bus which allows masters to direct the transfer of binary data to and from slaves. The master initiates bus cycles in order to transfer data between itself and slaves. The slave detects bus cycles that are initiated by the active master and, when those cycles select it, transfers data between itself and the master.

Four types of cycles are defined: an address-only cycle, a single transfer cycle, a block transfer cycle, and an interrupt acknowledge cycle. To maximize data transfer rates in multiprocessor systems, the VSB standard defines a mechanism that allows the master to broadcast the data to any number of slaves in the course of a single cycle. In addition, the data transfer mechanism supports dynamic bus sizing as well as resource locking and data caching.

The arbitration bus is the second of the two sub-buses defined in the VSB standard. It allows arbiter modules and/or requester modules to coordinate the use of the data transfer bus. Two arbitration methods are defined - a serial arbitration method and a parallel (distributed) arbitration method. These arbitration methods provide protocols to implement an array of subsystem architectures. Using the serial arbitration method, a designer can implement a single master subsystem that includes a single processor board requiring access to large amounts of memory. This method could be used to build a system that gives priority to a primary master that, when it can, grants the bus to other secondary masters. At the other end of the spectrum, a multiprocessing subsystem can be implemented using the parallel arbitration method.

CHAPITRE 1: INTRODUCTION A LA NORME DU BUS CEI 822 VSB

1.1 *Objectifs de la norme CEI 822 VSB, bus parallèle de sous-système du bus CEI 821 VMEbus (désormais référencé VSB)*

Le présent bus VSB est un bus d'extension de sous-système local. Il permet à une carte processeur d'accéder à la mémoire additionnelle et aux entrées/sorties sur un bus local, réduisant ainsi le trafic sur le bus global et améliorant les taux de transfert total du système. Le système a été conçu avec les objectifs suivants:

a) Améliorer les performances des systèmes multiprocesseurs en permettant la réalisation de sous-systèmes locaux.

b) Spécifier les caractéristiques électriques requises pour réaliser des cartes qui transféreront de façon fiable des données sur le bus VSB.

c) Spécifier les exigences mécaniques pour être compatible avec les systèmes VSB.

d) Spécifier les protocoles qui définissent avec précision l'interaction entre le VSB et les unités qui lui sont connectées.

e) Fournir la terminologie et les définitions nécessaires à la description des protocoles VSB.

1.2 *Eléments du système VSB*

1.2.1 *Définitions générales*

La structure du VSB peut être décrite de deux façons différentes: sa structure mécanique et sa structure fonctionnelle.

Le VSB est principalement utilisé en tant que bus secondaire, il n'y a pas de spécifications mécaniques des produits au niveau de la carte VSB et/ou au niveau du châssis. Il est supposé que les produits incluant le VSB ont été conçus pour être conformes aux spécifications mécaniques du bus global du système. C'est pourquoi la norme VSB décrit seulement les dimensions physiques du fond de panier.

Les spécifications fonctionnelles du VSB décrivent le fonctionnement du bus, les modules fonctionnels participant à ses diverses opérations et les règles régissant leur comportement. Ce paragraphe fournit des définitions informelles de termes de base utilisés pour décrire à la fois les structures mécaniques et fonctionnelles du VSB.

1.2.1.1 *Définition de la structure physique*

CARTE

Carte de circuit imprimé, son ensemble de composants électroniques et au moins un connecteur 96 broches.

CHAPTER 1: INTRODUCTION TO THE IEC 822 VSB BUS STANDARD

1.1 *Standard objectives of the IEC 822 VSB parallel Subsystem Bus of the IEC 821 VMEbus (Subsystem henceforth referred to as VSB)*

This VSB bus is a local subsystem extension bus. It allows a processor board to access additional memory and I/O over a local bus, removing traffic from the global bus and improving the total throughput of the system. The system has been conceived with the following objectives:

a) To improve the performance of multiprocessor systems by allowing the design of local subsystems.

b) To specify the electrical characteristics required to design boards that will reliably transfer data over the VSB.

c) To specify the mechanical requirements to be compatible with VSB systems.

d) To specify protocols that precisely define the interaction between the VSB and devices interfaced to it.

e) To provide terminology and definitions that describe VSB protocols.

1.2 *VSB system elements*

1.2.1 *Basic definitions*

The structure of the VSB can be described from two points of view: its mechanical structure and its functional structure.

Because the primary use of the VSB is as a secondary bus, there are no mechanical specifications of VSB board level, and/or box level products. It is assumed that products that include the VSB have been designed to comply with the mechanical specifications of the global system bus. Therefore, the VSB standard only describes the physical dimensions of the backplane.

The functional specifications of the VSB describe how the bus works, what functional modules participate in its various operations, and the rules that govern their behavior. This paragraph provides informal definitions for the basic terms used to describe both the mechanical and functional structure of the VSB.

1.2.1.1 *Physical structure definition*

BOARD

A printed circuit (PC) board, its collection of electronic components, and at least one 96-pin connector.

FOND DE PANIER VSB

Ensemble constitué d'une carte de circuit imprimé et de connecteurs 96 broches. Le fond de panier réunit les 64 broches des deux rangées extérieures de connecteurs VSB, assurant ainsi le cheminement des signaux nécessaire aux opérations VSB.

EMPLACEMENT

Position à laquelle une carte peut être insérée dans un fond de panier. Chaque emplacement VSB est doté au moins d'un connecteur 96 broches.

CHASSIS

Cadre rigide servant de support mécanique aux cartes insérées dans le fond de panier, garantissant un embrochage correct des connecteurs et évitant que les cartes adjacentes ne rentrent en contact les unes avec les autres. Il guide aussi le flux d'air de refroidissement à travers le système et garantit le maintien en cas de vibrations des cartes insérées dans le fond de panier.

1.2.1.2 *Définition de la structure fonctionnelle*

La figure 1-1, page 22, montre un schéma-bloc des modules fonctionnels et des sous-ensembles de bus définis par la norme VSB.

LOGIQUE D'INTERFACE DE FOND DE PANIER

Logique spéciale d'interface qui prend en considération les caractéristiques du fond de panier. La norme VSB prescrit certaines exigences de conception de cette logique, tenant compte de l'impédance de la ligne de signal, des temps de propagation, des valeurs d'adaptation, de la longueur maximale du fond de panier et du nombre d'emplacements autorisés.

MODULE FONCTIONNEL

Ensemble de circuits électroniques localisés sur une carte accomplissant une tâche spécifique. Les modules fonctionnels sont utilisés comme support pour commenter les protocoles de bus et ne doivent pas être considérés comme une contrainte de conception de la logique réelle.

BUS DE TRANSFERT DE DONNEES

Un des deux sous-ensembles de bus définis dans la norme VSB. Il permet aux MAITRES d'assurer le transfert des données binaires vers ou à partir des ESCLAVES. (Le bus de transfert de données VSB est souvent désigné par le mnémonique DTB.) Le DTB comporte 32 lignes d'adresses/données multiplexées et les signaux de contrôle associés qui sont nécessaires à l'exécution des cycles sur le VSB.

MAITRE

Module fonctionnel qui déclenche les cycles du bus dans le but de transférer des données entre lui-même et des modules ESCLAVES du VSB. Le MAITRE qui contrôle le DTB à un moment donné est appelé le MAITRE actif.

ESCLAVE

Module fonctionnel qui détecte les cycles de bus déclenchés par le MAITRE actif et qui, lorsque ces cycles l'ont sélectionné, transfère les données entre lui-même et le MAITRE. La norme VSB définit un mécanisme qui permet à un nombre quelconque d'ESCLAVES de participer à un cycle de bus.

VSB BACKPLANE
An assembly that includes a printed circuit (PC) board and 96-pin connectors. The backplane buses the 64 pins on the two outer rows of the VSB connectors, providing the signal paths needed for VSB operation.

SLOT
A position where a board can be inserted into a backplane. Each VSB slot provides at least one 96-pin connector.

SUBRACK
A rigid framework that provides mechanical support for boards inserted into the backplane, ensuring that the connectors mate properly and that adjacent boards do not contact each other. It also guides the cooling airflow through the system, and ensures that inserted boards are not disengaged from the backplane due to vibration.

1.2.1.2 *Functional structure definition*

Figure 1-1, page 23, shows a block diagram of the functional modules and sub-buses defined by the VSB standard.

BACKPLANE INTERFACE LOGIC
Special interface logic that takes into account the characteristics of the backplane. The VSB standard prescribes certain requirements for the design of this logic, which take into account the signal line impedance, propagation times, termination values, the maximum length of the backplane and the number of slots allowed.

FUNCTIONAL MODULE
A collection of electronic circuitry that resides on one board and works to accomplish a specific task. Functional modules are used as a vehicle for discussing bus protocols, and should not be considered to constrain the design of actual logic.

DATA TRANSFER BUS
One of the two sub-buses defined in the VSB standard. It allows MASTERS to direct the transfer of binary data to and from SLAVES. (The VSB Data Transfer Bus is often abbreviated DTB.) The DTB contains 32 multiplexed address/data lines and the associated control signals that are required to execute cycles on the VSB.

MASTER
A functional module that initiates bus cycles in order to transfer data between itself and VSB SLAVES. The MASTER that is currently in control of the DTB is referred to as the active MASTER.

SLAVE
A functional module that detects bus cycles initiated by the active MASTER and, when those cycles select it, transfers data between itself and the MASTER. The VSB standard defines a mechanism through which any number of SLAVES can participate in a bus cycle.

ESCLAVE SELECTIONNE
Tous les ESCLAVES qui sont sélectionnés par le cycle.

ESCLAVE REPONDANT
ESCLAVE sélectionné qui répond au MAITRE actif en reconnaissant le transfert de données ou le transfert MOT D'ETAT/IDentificateur (MOT D'ETAT/ID).

ESCLAVE PARTICIPANT
ESCLAVE sélectionné qui choisit de participer au cycle par la prise en compte des données transitant sur le bus de données.

ESCLAVE AU REPOS
ESCLAVE qui n'est pas sélectionné par le cycle.

ESCLAVE CONCURRENT
ESCLAVE qui a une demande d'interruption en attente et qui participe à un cycle de RECONNAISSANCE D'INTERRUPTION.

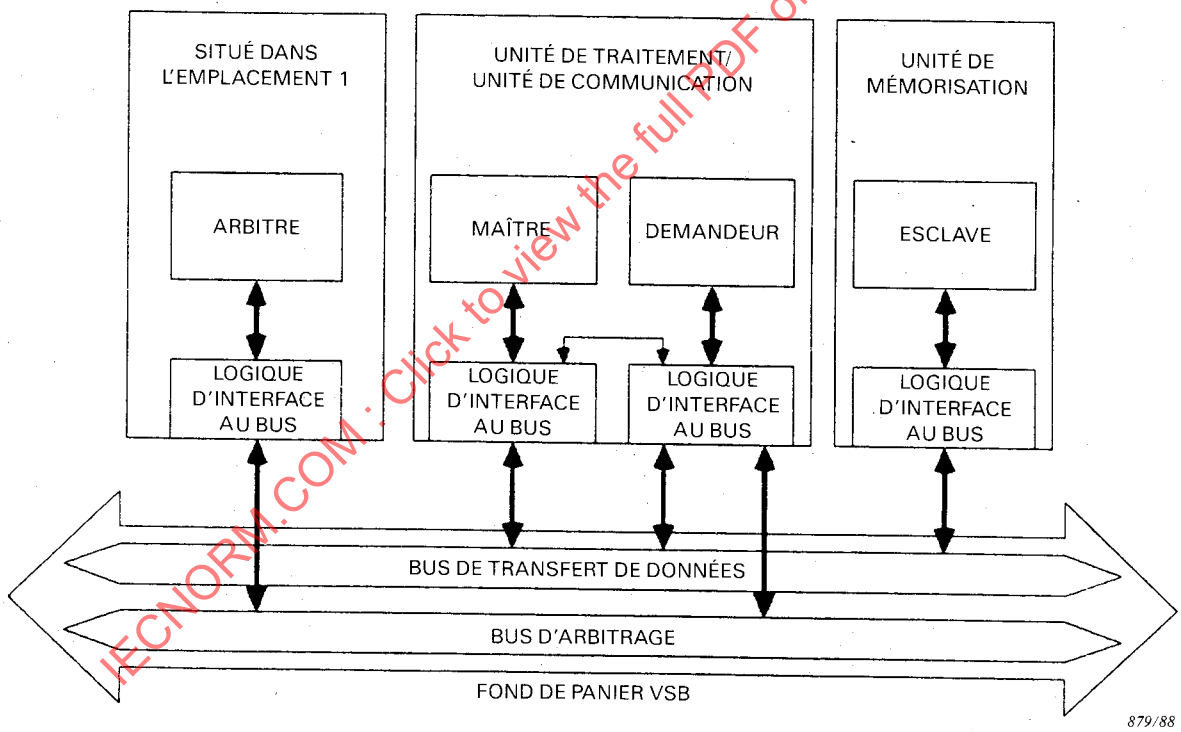


Fig. 1-1. - Modules fonctionnels et sous-ensembles de bus définis par la norme VSB.

BUS D'ARBITRAGE DU VSB
Deuxième sous-ensemble du bus défini dans la norme VSB. Il permet aux modules ARBITRE et/ou DEMANDEUR de coordonner l'utilisation du DTB par les MAITRES situés sur le VSB. La norme VSB définit deux méthodes d'arbitrage: une méthode d'arbitrage série et une méthode d'arbitrage parallèle.

SELECTED SLAVE
All SLAVES that are selected by the cycle.

RESPONDING SLAVE
The one selected SLAVE which responds to the active MASTER by acknowledging the data transfer of the STATUS/ID transfer.

PARTICIPATING SLAVE
Selected SLAVE which chooses to participate in the cycle by capturing the data carried on the data lines.

IDLE SLAVE
SLAVE which is not selected by the cycle.

CONTENDING SLAVE
SLAVE that has an interrupt request pending and that participates in an INTERRUPT-ACKNOWLEDGE cycle.

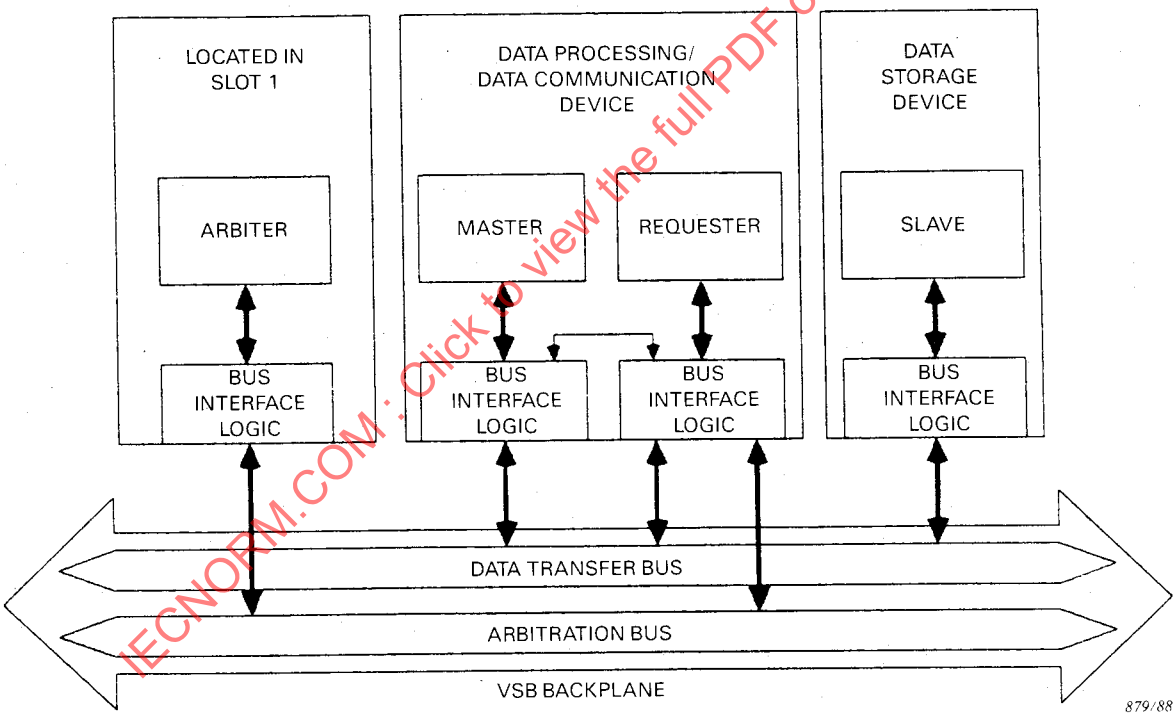


Fig. 1-1. - Functional modules and sub-buses defined by the VSB standard.

VSB ARBITRATION BUS
The second of the two sub-buses defined in the VSB standard. It allows ARBITER modules and/or REQUESTER modules to coordinate the use of the DTB by VSB MASTERS. The VSB defines two arbitration methods - a Serial arbitration method and a Parallel arbitration method.

DEMANDEUR

Module fonctionnel situé sur la carte d'un MAITRE et qui requiert l'usage du DTB à la demande de son MAITRE. Lorsque l'arbitrage série est implanté, après la demande d'utilisation du DTB, le DEMANDEUR attend que l'allocation du bus lui soit accordée par l'ARBITRE.

Dans la méthode d'arbitrage parallèle, le DEMANDEUR associé au MAITRE actif déclenche un cycle d'ARBITRAGE. Ce cycle d'ARBITRAGE est utilisé pour déterminer le MAITRE qui aura l'autorisation d'utiliser le DTB. La norme VSB appelle le DEMANDEUR associé au MAITRE actif le DEMANDEUR actif.

DEMANDEUR CONCURRENT

DEMANDEUR qui a une demande de bus en attente et qui participe à un cycle d'ARBITRAGE.

ARBITRE

Quand la méthode d'arbitrage série est employée, le module ARBITRE accepte les demandes d'allocation du DTB émises par les DEMANDEURS et accorde le contrôle du DTB à un seul DEMANDEUR à un moment donné. Il y a un seul ARBITRE actif dans le mécanisme d'arbitrage série et il est toujours situé dans l'emplacement 1. Dans la méthode d'arbitrage parallèle, il n'y a pas d'ARBITRE.

CHAINE SERIE

Type spécial de ligne de signal qui propage l'allocation du bus de carte à carte, partant de la carte installée dans le premier emplacement et allant jusqu'à celle située dans le dernier emplacement.

ADRESSAGE GEOGRAPHIQUE

Dispositif selon lequel une adresse unique est affectée à chaque emplacement du fond de panier. Cette adresse peut être lue par la carte qui est installée dans l'emplacement. La norme VSB définit l'utilisation de l'adressage géographique pour deux raisons: (1) il forme une partie de l'IDentificateur D'INTERRUPTION (ID INTERRUPTION) utilisé pendant le cycle de RECONNAISSANCE D'INTERRUPTION, et (2) il forme une partie de l'IDentificateur D'ARBITRAGE (ID ARBITRAGE) utilisé pendant un cycle d'ARBITRAGE parallèle. L'adressage géographique peut aussi être utilisé pour définir des variables globales de la carte telles que l'adresse de base d'une carte mémoire.

1.2.1.3 *Types de cycles VSB*

CYCLE DE BUS VSB

C'est une séquence de transitions de niveau sur les lignes de signaux du DTB dont l'objet est le transfert d'une adresse et (dans la plupart des cas) de données entre le MAITRE actif et les ESCLAVES sélectionnés. Les protocoles du VSB sont complètement asynchrones. Le MAITRE actif positionne un signal de validation qui indique qu'un cycle est en cours. L'ESCLAVE répondant acquitte le signal du MAITRE. Cependant, l'ESCLAVE répondant peut retarder son acquittement aussi longtemps qu'il en a besoin. Le cycle DTB est généralement divisé en trois phases: une diffusion d'adresse, zéro ou plusieurs transferts de données, puis une fin de cycle.

REQUESTER

A functional module that resides on the same board as a MASTER and requests use of the DTB whenever its MASTER needs it. When implementing Serial arbitration, after requesting use of the DTB, the REQUESTER waits for the bus to be granted to it by the ARBITER.

In the Parallel arbitration method, the REQUESTER that is associated with the active MASTER initiates an ARBITRATION cycle. This ARBITRATION cycle is used to determine which MASTER will be granted use of the DTB. The VSB standard calls the REQUESTER that is associated with the active MASTER the active REQUESTER.

CONTENDING REQUESTER

REQUESTER that has a bus request pending and that participates in an ARBITRATION cycle.

ARBITER

When implementing the Serial arbitration method, the ARBITER module accepts requests for the DTB from REQUESTERS and grants control of the DTB to one REQUESTER at a time. There is one and only one active ARBITER in the Serial arbitration scheme, and it is always located in slot 1. An ARBITER is not required in the Parallel arbitration method.

DAISY CHAIN

A special type of signal line that is used to propagate bus grants from board to board, starting with the board installed in the first slot and ending with the one installed in the last slot.

GEOGRAPHICAL ADDRESSING

A scheme wherein each slot in the backplane is assigned a unique address. This address can be read by the board that is installed in the slot. The VSB standard defines the use of the geographical address for two purposes: (1) it forms part of the INTERRUPT ID used during an INTERRUPT-ACKNOWLEDGE cycle and, (2) it forms part of the ARBITRATION ID used during a Parallel ARBITRATION cycle. The geographical address can also be used to set global board variables such as the base address of a memory board.

1.2.1.3 *Types of VSB cycles*

VSB BUS CYCLE

A sequence of level transitions on the signal lines of the DTB that results in the transfer of an address and (in most cases) data between the active MASTER and selected SLAVES. The protocols of the VSB are fully asynchronous. The active MASTER asserts a strobe signal indicating that a cycle is in progress. The responding SLAVE acknowledges the MASTER'S signal. However, the responding SLAVE can delay its acknowledgment for as long as it needs. The DTB cycle is generally divided into three phases: an address broadcast, zero or more data transfers, and then cycle termination.

DIFFUSION D'ADRESSE

Phase d'un cycle de bus dans laquelle un ESCLAVE est sélectionné comme l'ESCLAVE répondant et zéro ou plusieurs ESCLAVES comme ESCLAVES participants. Pendant la phase de diffusion d'adresse, le MAITRE actif émet l'information d'adresse, puis positionne un signal de validation d'adresse. Après l'acquittement de la diffusion d'adresse par les ESCLAVES, le MAITRE termine la diffusion d'adresse.

TRANSFERT DE DONNEES

Phase d'un cycle pendant laquelle les données sont transférées entre le MAITRE et les ESCLAVES sélectionnés. Elle commence lorsque le MAITRE actif positionne le signal de validation de données et se termine après que l'ESCLAVE répondant a acquitté le transfert de données et que tous les ESCLAVES participants indiquent qu'ils sont prêts à participer à un nouveau cycle.

FIN DE CYCLE

Phase pendant laquelle le MAITRE termine le cycle et les ESCLAVES acquittent cette fin de cycle en positionnant les signaux du bus dans l'état intercycle.

AJUSTEMENT DYNAMIQUE DU BUS

Possibilité pour certains microprocesseurs d'ajuster le nombre et la dimension des transferts de données à la dimension du champ de données à laquelle la carte répondante peut accéder en un transfert. Pendant la partie de diffusion d'adresse du cycle, l'ESCLAVE informe le MAITRE sur la dimension du champ de données qu'il émet ou reçoit. Cette information est disponible pour la logique de la carte laquelle peut alors ajuster la dimension de la donnée adressée pendant un transfert de donnée à la possibilité de l'ESCLAVE.

DIFFUSION DE DONNEES

Opération de diffusion dans laquelle les ESCLAVES participants acquièrent les données placées sur les lignes de données par le MAITRE actif pendant un cycle d'écriture.

ECOUTE DE DONNEES

Opération d'écoute dans laquelle les ESCLAVES participants acquièrent les données placées sur les lignes de données par l'ESCLAVE répondant pendant un cycle de lecture.

CYCLE DE TRANSFERT UNIQUE EN LECTURE

Cycle utilisé pour transférer 1, 2, 3 ou 4 octets de l'ESCLAVE répondant au MAITRE actif et éventuellement aux ESCLAVES participants. Le cycle commence lorsque le MAITRE actif diffuse l'information d'adresse sur les lignes d'adresse/donnée. Chaque ESCLAVE détermine, après avoir vérifié l'adresse, s'il doit répondre au cycle. Si c'est le cas, il acquitte l'adresse et extrait les données de sa mémoire locale. Lorsque le MAITRE libère les lignes d'adresse/donnée, l'ESCLAVE répondant positionne ses données sur les mêmes lignes et acquitte le transfert. Le MAITRE aussi bien que les ESCLAVES participants acquièrent les données. Lorsque tous les ESCLAVES sélectionnés ont signalé leur acquittement, le MAITRE termine le cycle.

ADDRESS BROADCAST

The phase of a bus cycle which selects one SLAVE as the responding SLAVE and zero or more SLAVES as participating SLAVES. During the address broadcast the active MASTER broadcasts the addressing information and then asserts an address strobe. After the SLAVES acknowledge the address broadcast, the MASTER terminates the address broadcast.

DATA TRANSFER

The phase of a cycle during which data is transferred between the MASTER and the selected SLAVES. It starts when the active MASTER asserts the data strobe and ends after the responding SLAVE acknowledges the transfer and all participating SLAVES indicate that they are ready to participate in a new cycle.

CYCLE TERMINATION

The phase of a cycle during which the MASTER terminates the cycle and SLAVES acknowledge this termination by establishing the inter-cycle state of bus signals.

DYNAMIC BUS SIZING

The ability of some microprocessors to adjust the number and the size of data transfers to the amount of data that the responding board can access in one transfer. During the address broadcast portion of the cycle, the SLAVE informs the MASTER how many data lines it actually drives or receives. This information is made available to on-board logic which can then adjust the amount of data that it accesses during the data transfer to the capabilities of the SLAVE.

DATA BROADCAST

A broadcast operation is one wherein participating SLAVES capture the data that is placed on the data lines by the active MASTER during a write cycle.

DATA BROADCALL

A broadcall operation is one wherein participating SLAVES capture the data that is placed on the data lines by the responding SLAVE during a read cycle.

SINGLE-TRANSFER READ CYCLE

A cycle that is used to transfer 1, 2, 3, or 4 bytes from the responding SLAVE to the active MASTER, and possibly to participating SLAVES. The cycle begins when the active MASTER broadcasts the addressing information on the address/data lines. Each SLAVE checks the address to see if it is to respond to the cycle. If so, it acknowledges the address and retrieves the data from its internal storage. When the MASTER releases the address/data lines, the responding SLAVE places its data on them and acknowledges the transfer. The MASTER as well as participating SLAVES capture the data. After all selected SLAVES signal their agreement the MASTER terminates the cycle.

CYCLE DE TRANSFERT UNIQUE EN ECRITURE

Cycle utilisé pour transférer 1, 2, 3 ou 4 octets du MAITRE actif aux ESCLAVES sélectionnés. Le cycle commence quand le MAITRE diffuse l'information d'adresse sur les lignes d'adresse/donnée. Chaque ESCLAVE détermine, après avoir vérifié l'adresse, s'il doit participer au cycle. L'ESCLAVE répondant acquitte la diffusion d'adresse. Le MAITRE commute alors les lignes d'adresse/donnée pour envoyer les données et place ses données sur le bus. Les ESCLAVES sélectionnés peuvent alors acquérir les données. L'ESCLAVE répondant acquitte le transfert. Lorsque tous les ESCLAVES sélectionnés ont signalé leur acquittement, le MAITRE termine le cycle.

CYCLE DE TRANSFERT PAR BLOC EN LECTURE

Cycle DTB utilisé pour transférer un bloc d'octets de l'ESCLAVE répondant vers le MAITRE actif et éventuellement vers les ESCLAVES participants. Ce transfert effectue une série de transferts de données de 1, 2 ou 4 octets. Il diffère d'une série de cycles de TRANSFERT UNIQUE en lecture en ce sens que le MAITRE diffuse l'adresse une seule fois en début de cycle. La responsabilité de gérer l'adresse de chacun des transferts suivants est laissée aux ESCLAVES sélectionnés.

CYCLE DE TRANSFERT PAR BLOC EN ECRITURE

Cycle DTB utilisé pour transférer un bloc d'octets du MAITRE actif aux ESCLAVES sélectionnés. Ce transfert effectue une série de transferts de données de 1, 2 ou 4 octets. Il diffère d'une série de cycles de TRANSFERT UNIQUE en écriture en ce sens que le MAITRE diffuse l'adresse une seule fois au début du cycle. La responsabilité de gérer l'adresse de chacun des transferts suivants est laissée aux ESCLAVES sélectionnés.

CYCLE D'ACCES INDIVISIBLE

Cycle DTB utilisé pour accéder de façon indivisible aux emplacements d'un ESCLAVE en interdisant à tout autre MAITRE d'accéder aux mêmes emplacements tant que l'opération n'est pas terminée.

CYCLE UNIQUEMENT D'ADRESSAGE

Cycle DTB qui consiste en une diffusion d'adresse sans transfert de données. Le MAITRE actif termine le cycle lorsque les ESCLAVES ont acquitté la diffusion d'adresse.

CYCLE DE RECONNAISSANCE D'INTERRUPTION

Cycle DTB déclenché par un MAITRE en réponse à une demande d'interruption d'un ESCLAVE. Un cycle de RECONNAISSANCE D'INTERRUPTION met en jeu deux types d'ESCLAVES. Pendant le cycle de RECONNAISSANCE D'INTERRUPTION, tous les ESCLAVES concurrents commandent un ID INTERRUPTION sur le bus. Cet identificateur est une combinaison de l'adresse géographique de la carte, fournie par l'emplacement du fond de panier, et d'un code de priorité fourni par la logique sur la carte définie par l'utilisateur. L'ID INTERRUPTION est utilisé pour déterminer lequel des ESCLAVES concurrents répondra au cycle.

SINGLE-TRANSFER WRITE CYCLE

A cycle that is used to transfer 1, 2, 3, or 4 bytes from the active MASTER to the selected SLAVES. The cycle begins when the MASTER broadcasts the addressing information on the address/data lines. Each SLAVE checks the address to see if it is to participate in the cycle. The responding SLAVE acknowledges the address broadcast. The MASTER then switches the address/data lines to carry data, and places its data on the bus. The selected SLAVES can then store the data. The responding SLAVE acknowledges the transfer. After all selected SLAVES signal their agreement the MASTER terminates the cycle.

BLOCK-TRANSFER READ CYCLE

A DTB cycle that is used to transfer a block of bytes from the responding SLAVE to the active MASTER, and possibly to participating SLAVES. This transfer is done using a number of 1, 2, or 4-byte data transfers. It differs from a series of SINGLE-TRANSFER read cycles in that the MASTER broadcasts the address only once, at the beginning of the cycle. It is the responsibility of the selected SLAVES to control the address for each subsequent data transfer.

BLOCK-TRANSFER WRITE CYCLE

A DTB cycle that is used to transfer a block of bytes from the active MASTER to the selected SLAVES. This transfer is done using a series of 1, 2, or 4-byte data transfers. It differs from a series of SINGLE-TRANSFER write cycles in that the MASTER broadcasts the address only once, at the beginning of the cycle. It is the responsibility of the selected SLAVES to control the address for each subsequent data transfer.

INDIVISIBLE-ACCESS CYCLE

A DTB cycle that is used to access SLAVE locations indivisibly and without permitting any other MASTER to access these locations until the operation is complete.

ADDRESS-ONLY CYCLE

A DTB cycle that consists of an address broadcast, but no data transfer. The active MASTER terminates the cycle after the SLAVES acknowledge the address broadcast.

INTERRUPT-ACKNOWLEDGE CYCLE

A DTB cycle that is initiated by a MASTER in response to an interrupt request from a SLAVE. An INTERRUPT-ACKNOWLEDGE cycle involves two types of SLAVES. During the INTERRUPT-ACKNOWLEDGE cycle, all contending SLAVES drive an INTERRUPT ID on the bus. This ID is a combination of the geographical address of the board that is supplied by the backplane slot, and a priority code that is supplied by user defined on-board logic. The INTERRUPT ID is used to determine which of the contending SLAVES will respond to the cycle.

CYCLE D'ARBITRAGE

Cycle déclenché par le DEMANDEUR actif, en réponse à une demande de bus, lorsque son MAITRE actif associé n'a plus besoin du bus. Ce cycle sélectionne le MAITRE qui recevra l'allocation d'utilisation du DTB. Si le DEMANDEUR actif détecte une demande du bus et si son MAITRE associé n'a plus besoin du bus, il déclenche alors un cycle d'ARBITRAGE. Pendant le cycle d'ARBITRAGE, tous les DEMANDEURS concurrents placent un ID ARBITRAGE sur le bus. Cet identificateur est une combinaison de l'adresse géographique de la carte, fournie par l'emplacement du fond de panier, et d'un code de priorité fourni par la logique sur la carte définie par l'utilisateur. A la fin du cycle d'ARBITRAGE, un des DEMANDEURS concurrents devient le DEMANDEUR actif.

1.3 *Diagrammes de la norme VSB*

Trois types de diagrammes sont utilisés pour aider à définir et à décrire le fonctionnement du VSB.

a) Les *schémas-blocs* montrent les exigences d'interconnexion des lignes de signaux des modules fonctionnels définis par la norme VSB.

b) Les *organigrammes* montrent le déroulement des événements tels qu'ils apparaissent pendant une opération VSB. Les événements sont énoncés textuellement et décrivent séquentiellement l'interaction entre deux ou plusieurs modules fonctionnels. La norme VSB décrit en détail le comportement des divers modules fonctionnels. Elle commente la façon dont un module répond à un signal sans indiquer d'où vient le signal. Pour cette raison, une spécification de protocole ne donne pas au lecteur une image complète de ce qui se passe sur le bus. Les organigrammes sont utilisés pour aider le lecteur à surmonter cette difficulté.

c) Les *chronogrammes* montrent les relations de temps entre les transitions des signaux. Les paramètres de temps ont des limites minimales et/ou maximales qui leur sont associées. Certains de ces paramètres spécifient le comportement de la logique d'interface du fond de panier, alors que d'autres spécifient le comportement des modules fonctionnels.

1.4 *Terminologie utilisée dans la norme*

Pour éviter toute confusion et pour clarifier les exigences de conformité, de nombreux paragraphes de cette norme sont référencés de mots clés numérotés séquentiellement indiquant le type d'information qu'ils contiennent. Les mots clés sont:

REGLE
RECOMMANDATION
SUGGESTION
AUTORISATION
OBSERVATION

ARBITRATION CYCLE

A cycle that is initiated by the active REQUESTER in response to a bus request, after its associated active MASTER no longer needs the bus. This cycle is used to select the MASTER that will be granted use of the DTB. If the active REQUESTER detects a request for the bus, and if its associated MASTER no longer needs the bus, it initiates an ARBITRATION cycle. During the ARBITRATION cycle, all contending REQUESTERS drive an ARBITRATION ID on the bus. This ID is a combination of the geographical address of the board that is supplied by the backplane slot, and a priority code that is supplied by user defined on-board logic. At the end of the ARBITRATION cycle one of the contending REQUESTERS becomes the active REQUESTER.

1.3 *VSB standard diagrams*

Three types of diagrams are used to help define and describe the operation of the VSB:

a) *Block diagrams* show the signal line interconnect requirements of the functional modules defined by the VSB standard.

b) *Flow diagrams* show the stream of events as they would occur during a VSB operation. The events are stated in words and sequentially describe the interaction between two or more functional modules. The VSB standard describes in detail the behavior of the various functional modules. It discusses how a module responds to a signal without saying where the signal came from. Because of this, a protocol specification does not give the reader a complete picture of what happens over the bus. Flow diagrams are used to help the reader overcome this difficulty.

c) *Timing diagrams* show the timing relationships between signal transitions. The timing parameters have minimum and/or maximum limits associated with them. Some of these timing parameters specify the behavior of the backplane interface logic, while others specify the behavior of the functional modules.

1.4 *Standard terminology*

To avoid confusion, and to make very clear what the requirements for compliance are, many of the paragraphs in this standard are labeled with sequentially numbered keywords that indicate the type of information they contain. The keywords are:

RULE
RECOMMENDATION
SUGGESTION
PERMISSION
OBSERVATION

REGLE chapitre.numéro:

Les règles forment la structure de base de la norme VSB. Elles sont parfois exprimées sous forme textuelle et parfois sous forme de figures ou de tableaux. Les règles sont caractérisées par un style impératif. Les mots en lettres majuscules DOIT et NE DOIT PAS sont réservés pour établir les règles de cette norme et ne sont pas utilisés pour d'autres usages.

RECOMMANDATION chapitre.numéro:

Partout où une recommandation apparaît, les concepteurs auront la prudence de suivre les conseils donnés. Faire autrement pourrait conduire à des problèmes graves ou à une mauvaise performance. Bien que le VSB ait été conçu pour des systèmes de haute performance, il est possible de concevoir un système VSB conforme à toutes les règles, mais ayant des performances médiocres. Dans de nombreux cas, un concepteur a besoin d'un certain niveau d'expérience avec le VSB de façon à réaliser des cartes qui fournissent la performance optimale. Les recommandations contenues dans cette norme sont fondées sur ce type d'expérience et sont fournies aux concepteurs pour accélérer leur apprentissage.

SUGGESTION chapitre.numéro:

Dans la norme VSB, une suggestion est un conseil utile mais non vital. Le lecteur est encouragé à ne pas l'écarter sans y réfléchir. Certains choix de conception des cartes VSB sont difficiles tant que l'on n'a pas acquis suffisamment d'expérience. Les suggestions sont faites pour aider le concepteur encore inexpérimenté. Plusieurs suggestions se rapportent soit à la conception de cartes facilement reconfigurables pour opérer avec d'autres cartes, soit à la conception de cartes qui sont plus faciles à dépanner.

AUTORISATION chapitre.numéro:

Dans certains cas une règle VSB n'interdit pas spécifiquement une certaine méthode de conception, mais, le lecteur peut se demander si celle-ci transgresse la règle ou si elle risque de conduire à un ennui. Les autorisations confirment au lecteur qu'une certaine méthode de conception est acceptable et n'entraînera pas de difficultés. Le mot PEUT, en lettres majuscules, est réservé pour énoncer les autorisations de cette norme et n'est pas utilisé pour d'autres usages.

OBSERVATION chapitre.numéro:

Les observations ne donnent pas de conseils spécifiques. Elles découlent généralement de la discussion. Elles soulignent les implications de certaines exigences du VSB et attirent l'attention sur certains points qui autrement pourraient être négligés. Elles donnent aussi la raison de certaines exigences afin que le lecteur comprenne la nécessité de les suivre.

1.4.1 *Etats des lignes de signaux*

Les protocoles du VSB sont décrits en terme de niveaux et de transitions sur les lignes de bus. Une ligne de signal est toujours présumée être dans l'un des deux niveaux, haut ou bas, ou en transition entre ces deux niveaux.

RULE chapter.number:
Rules form the basic framework of the VSB standard. They are sometimes expressed in text form and sometimes in the form of figures or tables. Rules are characterized by an imperative style. The upper case words MUST and MUST NOT are reserved for stating rules in this standard and are not used for any other purpose.

RECOMMENDATION chapter.number:
Wherever a recommendation appears, designers would be wise to take the advice given. Doing otherwise might result in some awkward problem or poor performance. While the VSB has been designed to support high performance systems, it is possible to design a VSB system that complies with all the rules, but has poor performance. In many cases, a designer needs a certain level of experience with VSB in order to design boards that deliver top performance. Recommendations found in this standard are based on this kind of experience, and are provided to designers to speed their traversal of the learning curve.

SUGGESTION chapter.number:
A suggestion in the VSB standard contains advice which is helpful but not vital. The reader is encouraged to consider the advice before discarding it. Some decisions made while designing VSB boards are difficult until experience has been gained with the VSB. Suggestions are included to help a designer who has not yet gained this experience. Some suggestions have to do with designing boards that can be easily reconfigured for operation with other boards, or with designing boards that are easier to debug.

PERMISSION chapter.number:
In some cases a VSB rule does not specifically prohibit a certain design approach. However, the reader might be left wondering whether that approach violates the spirit of the rule, or whether it might lead to some subtle problem. Permissions reassure the reader that a certain approach is acceptable, and will cause no problems. The upper case word MAY is reserved for stating permissions in this standard and is not used for any other purpose.

OBSERVATION chapter.number:
Observations do not offer any specific advice. They usually follow naturally from what has just been discussed. They spell out the implications of certain VSB requirements and bring attention to things that might otherwise be overlooked. They also give the rationale behind certain requirements, so that the reader understands why they are needed.

1.4.1 *Signal line states*

The protocols of the VSB are described in terms of levels and transitions on bus lines. A signal line is always assumed to be in one of two levels, high or low, or in transition between these two levels.

Pour les lignes de signaux significatifs sur niveaux, le niveau de tension TTL représente une information significative. Les lignes de signaux qui sont décrites comme significatives sur niveau peuvent être soit au niveau haut, soit au niveau bas. Chaque fois que le terme haut est employé, il se réfère au niveau haut de tension TTL. Le terme bas se réfère à un niveau bas de tension TTL.

Les lignes de signaux significatives sur front marquent un événement lorsqu'elles font une transition entre les deux niveaux de tension TTL. Il y a deux transitions possibles qui peuvent apparaître sur une ligne de signal. Un front montant est le temps durant lequel un signal effectue sa transition du niveau bas au niveau haut. Le front descendant est le temps durant lequel un signal effectue sa transition de niveau haut au niveau bas.

Les temps de montée et de descente des émetteurs de bus sont la résultante d'un jeu complexe d'interactions mettant en jeu l'impédance des lignes de signaux du fond de panier, les adaptations et la longueur des lignes de signaux, l'impédance de source des émetteurs et la charge capacitive des lignes de signaux. La norme VSB ne spécifie pas les temps de montée et de descente. A la place, elle spécifie les caractéristiques électriques des émetteurs et des récepteurs. Elle spécifie aussi toutes les contraintes temporelles des signaux, prenant en compte le cas le plus défavorable de charge du bus et l'effet provoqué sur les temps de propagation de ces émetteurs. Si les concepteurs de VSB suivent ces contraintes temporelles, alors leurs cartes auront un fonctionnement fiable avec d'autres cartes compatibles VSB dans les conditions les plus défavorables.

1.4.2 *Utilisation de l'astérisque (*)*

Les noms de certains signaux ont un suffixe astérisque (*) pour aider à définir leur utilisation. L'astérisque a les significations suivantes:

a) Un astérisque (*) suivant le nom d'un signal qui est significatif sur un niveau indique un signal actif ou valide au niveau bas.

b) Un astérisque (*) suivant le nom d'un signal qui est significatif sur un front indique que le front descendant a une plus grande importance dans le protocole que le front montant.

1.5 *Spécification du protocole*

Le protocole principal utilisé dans le système VSB est un protocole en boucle fermée sur des lignes de bus interverrouillées. Chaque signal interverrouillé a un module source et un ou plusieurs modules destination. Un module spécifique émet et est acquitté par le ou les modules récepteurs. Une relation d'interverrouillage existe entre les modules émetteurs et récepteurs jusqu'à ce que le signal soit correctement acquitté. Par exemple, un MAITRE positionne le signal de validation de donnée qui sera reconnu plus tard par un signal d'acquittement (aucune limite de temps n'est prescrite dans la norme VSB). Le MAITRE ne doit pas enlever le signal de validation de donnée tant que tous les ESCLAVES n'ont pas acquitté le transfert de données.

On level significant signal lines, the TTL voltage level represents meaningful information. Signal lines that are described as level significant can be either high or low. Whenever the term high is used, it refers to a high TTL voltage level. The term low refers to a low TTL voltage level.

Edge significant signal lines mark an event when they make a transition between the two TTL voltage levels. There are two possible transitions which can appear on a signal line. A rising edge is the time during which a signal makes its transition from a low level to a high level. The falling edge is the time during which a signal makes its transition from a high level to a low level.

The rise and fall times of bus drivers are the result of a complex set of interactions involving the impedance of the backplane's signal lines, the terminations and length of the signal lines, the source impedance of the drivers, and the capacitive loading of the signal lines. The VSB standard does not specify rise and fall times. Instead, it specifies the electrical characteristics for drivers and receivers. It also specifies all signal timing requirements, taking into account the worst case bus loading and the effect it has on the propagation delay times of these drivers. If VSB designers follow these timing requirements, then their boards will operate reliably with other VSB compatible boards under worst case conditions.

1.4.2 *Use of the asterisk (*)*

Some signal names have an asterisk suffix (*) to help define their usage. The meaning of the asterisk is as follows:

a) An asterisk (*) following the name of a signal which is level significant denotes that the signal is true or valid when the signal is low.

b) An asterisk (*) following the name of a signal which is edge significant denotes that the falling edge is of greater significance in the protocol than is the rising edge.

1.5 *Protocol specification*

The primary protocol used in the VSB system is a closed loop protocol on interlocked bus lines. Each interlocked signal has a source module and one or more destination modules. It is sent from a specific module and is acknowledged by the receiving module(s). An interlocked relationship exists between the sending and the receiving modules until the signal is properly acknowledged. For example, a MASTER asserts the data strobe which is handshaked later with an acknowledge signal (no time limit is prescribed by the VSB standard). The MASTER does not remove the data strobe until all SLAVES have acknowledged the data transfer.

Les lignes de signaux utilisées sur le VSB peuvent être commandées par différents modules à des instants différents. Elles sont commandées par des émetteurs qui peuvent être commutés en/hors service sur chaque carte. Il est très important que leurs temps de commutation soient vérifiés avec soin pour éviter que deux émetteurs ne tentent de commander le même signal à des niveaux différents. Une notation spéciale est utilisée dans les chronogrammes pour spécifier leurs temps de commutation. Cela est montré dans la figure 1-2.

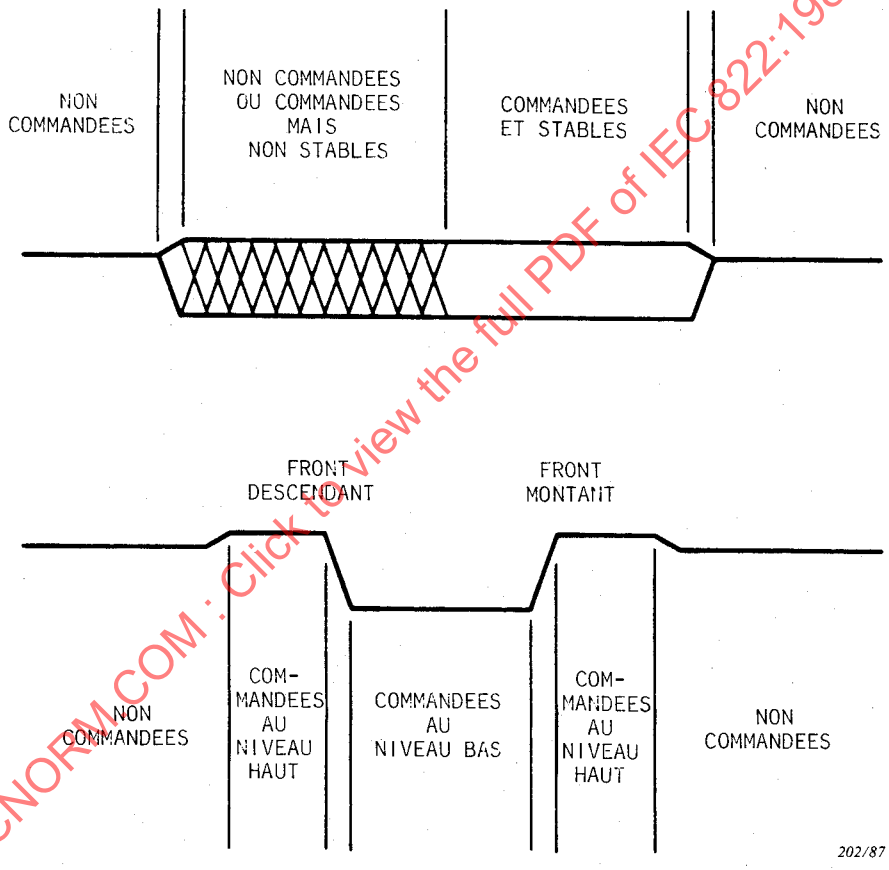


Fig. 1-2. - Notations utilisées dans les chronogrammes.

The signal lines used on the VSB may be driven by different modules at different times. They are driven with drivers that can be turned on and off at each board. It is very important that their turn-on and turn-off times be carefully controlled to prevent two drivers from attempting to drive the same signal line to different levels. A special notation in the timing diagrams is used to specify their turn-on and turn-off times. It is shown in Figure 1-2.

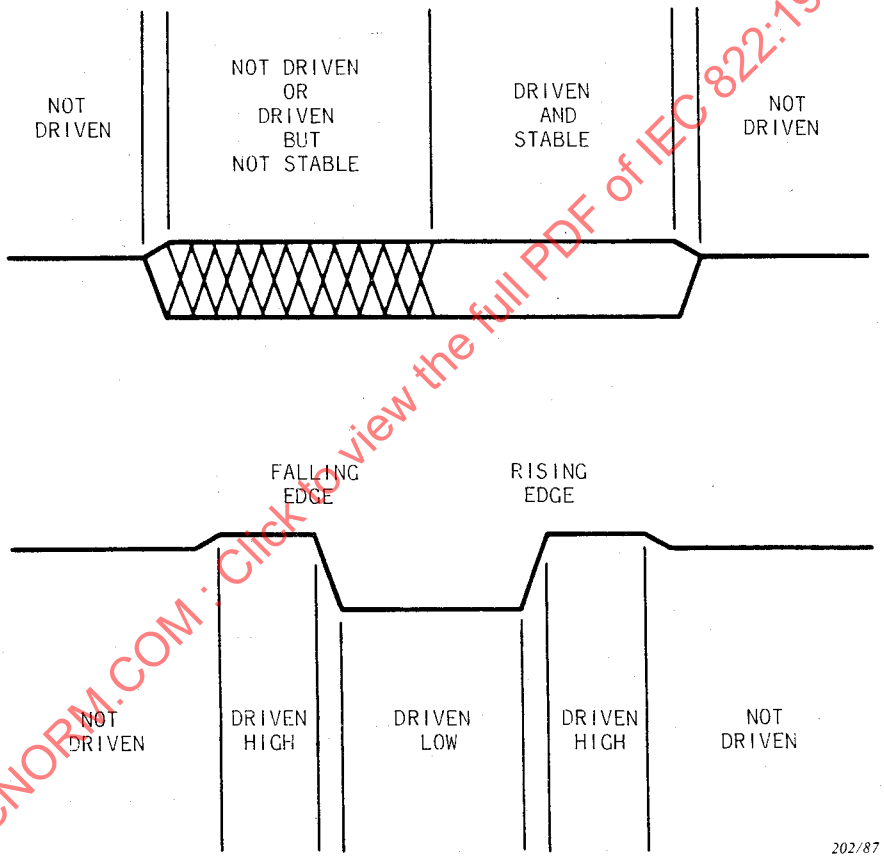


Fig. 1-2. - Signal timing notation.

CHAPITRE 2: BUS DE TRANSFERT DE DONNEES DU VSB

2.1 *Introduction*

Le VSB comprend un bus de transfert de données (DTB) à haute vitesse, asynchrone multiplexe. La figure 2-1 montre un système VSB typique incluant tous les modules fonctionnels du DTB. Les MAITRES utilisent le DTB pour sélectionner les emplacements d'octet des ESCLAVES et pour transférer les données vers ou à partir de ces emplacements. Certains MAITRES et ESCLAVES utilisent toutes les lignes du DTB, tandis que d'autres en utilisent seulement une partie.

Après avoir déclenché un cycle de transfert de données, le MAITRE attend que l'ESCLAVE répondant acquitte le transfert avant de terminer le cycle. Les protocoles de transfert asynchrones du VSB permettent à un ESCLAVE de prendre le temps nécessaire pour répondre.

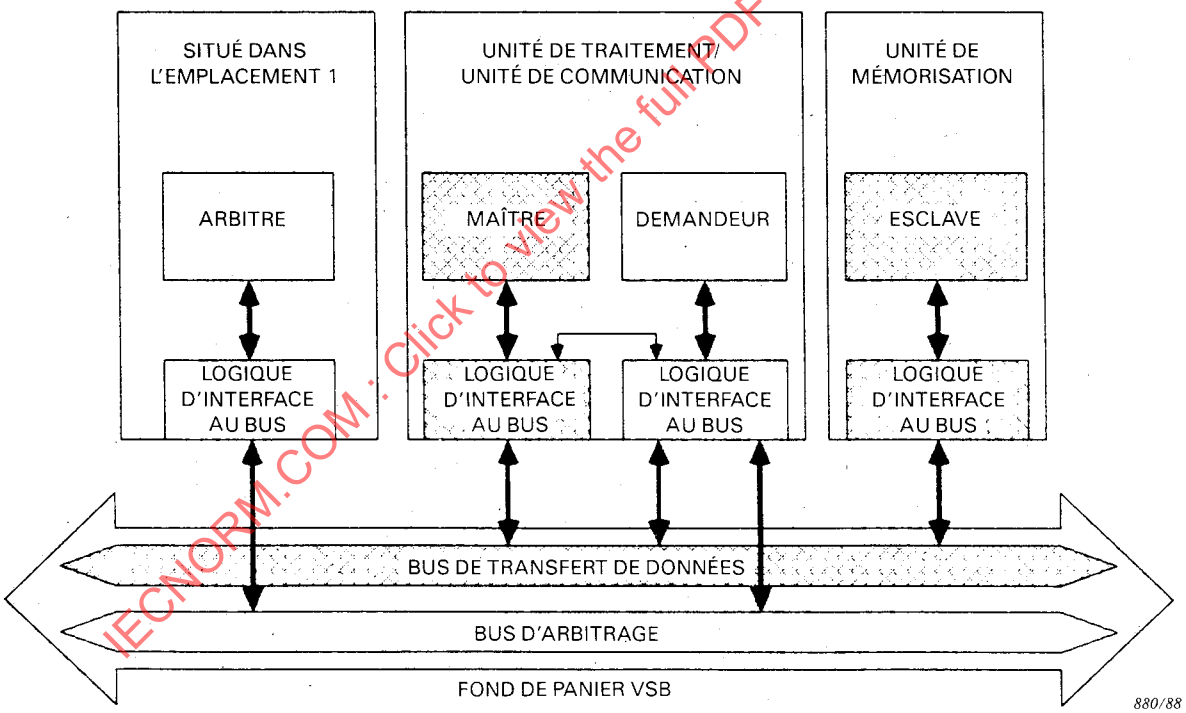


Fig. 2-1. - Schéma-bloc fonctionnel du bus de transfert de données.

CHAPTER 2: VSB DATA TRANSFER BUS

2.1 *Introduction*

The VSB includes a high speed asynchronous multiplexed Data Transfer Bus (DTB). Figure 2-1 shows a typical VSB system including all the functional modules of the DTB. MASTERS use the DTB to select byte locations provided by SLAVES and to transfer data to or from those locations. Some MASTERS and SLAVES use all of the DTB lines, while others use only a subset.

After a MASTER initiates a data transfer cycle, it waits for the responding SLAVE to acknowledge the transfer before finishing the cycle. The asynchronous transfer protocols of the VSB allow a SLAVE to take as long as it needs to respond.

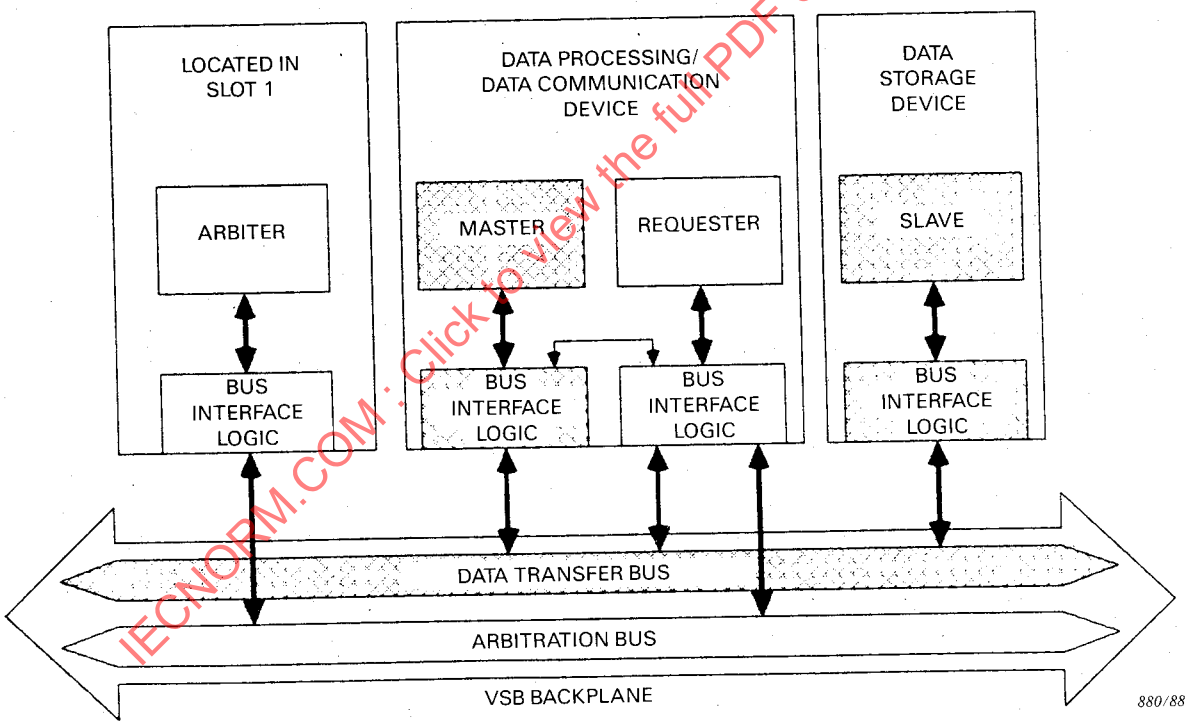


Fig. 2-1. - Data Transfer Bus functional block diagram.

2.2 *Lignes du bus de transfert de données*

Les lignes du bus de transfert de données VSB peuvent être classées en trois catégories:

a) Lignes d'adresse:	AD00-AD31	- Adresse/Donnée
	SPACE0	- Sélectionne espace 0
	SPACE1	- Sélectionne espace 1
	SIZE0	- Demande dimension 0
	SIZE1	- Demande dimension 1
	ASACK0*	- Acquittement d'adresse/dimension 0
	ASACK1*	- Acquittement d'adresse/dimension 1
	GA0-GA2	- Adressage géographique
b) Lignes de données:	AD00-AD31	- Adresse/Donnée
c) Lignes de commande:	PAS*	- Validation d'adresse physique
	AC	- Décodage d'adresse complet
	WR*	- Ecriture
	LOCK*	- Verrouillage
	WAIT*	- Attendre
	ASACK0*	- Acquittement d'adresse/dimension 0
	ASACK1*	- Acquittement d'adresse/dimension 1
	DS*	- Validation de donnée
	ACK*	- Acquittement de donnée
	ERR*	- Donnée erronée
	IRQ*	- Demande d'interruption
	CACHE*	- Peut être copié en "cache"

Cette section fournit une description générale des lignes du bus de transfert de données pour familiariser le lecteur avec leur fonction. La fonction de certaines de ces lignes peut varier durant les cycles de RECONNAISSANCE D'INTERRUPTION décrits au paragraphe 2.5.4. De plus, certaines lignes du DTB sont utilisées dans la méthode d'arbitrage parallèle décrite au chapitre 3.

2.2.1 *Lignes d'adresse*

Le MAITRE actif commande les lignes d'adresse pendant la phase de diffusion d'adresse du cycle pour sélectionner l'ESCLAVE qui répondra à la phase de transfert de données ainsi que les ESCLAVES qui y participeront.

2.2.1.1 *AD00-AD31*

Pendant la phase de diffusion d'adresse d'un cycle, le MAITRE actif place sur les lignes AD00-AD31 l'adresse des emplacements d'octet auxquels on accède pendant la phase de transfert de données du cycle.

L'adresse d'un groupe de 4 octets est déterminée par le niveau des lignes d'adresse AD02-AD31. L'adresse des emplacements d'octet dans un groupe de 4 octets dépend seulement du niveau des deux bits les moins significatifs des lignes d'adresse, c'est-à-dire les lignes d'adresse AD00 et AD01. Les emplacements d'octet sont répertoriés selon OCTET(0), OCTET(1), OCTET(2) et OCTET(3) en fonction de l'état de AD00 et AD01, comme il est défini au paragraphe 2.5.1.2, tableau 2-11.

2.2 *Data Transfer Bus lines*

The VSB Data Transfer Bus lines can be grouped into three categories:

a) Addressing lines:	AD00-AD31	- Address/Data
	SPACE0	- SPACE select 0
	SPACE1	- SPACE select 1
	SIZE0	- SIZE request 0
	SIZE1	- SIZE request 1
	ASACK0*	- Address/Size ACKnowledge 0
	ASACK1*	- Address/Size ACKnowledge 1
	GA0-GA2	- Geographical Addressing
b) Data lines:	AD00-AD31	- Address/Data
c) Control lines:	PAS*	- Physical Address Strobe
	AC	- Address decode Complete
	WR*	- WRite
	LOCK*	- LOCK
	WAIT*	- WAIT
	ASACK0*	- Address/Size ACKnowledge 0
	ASACK1*	- Address/Size ACKnowledge 1
	DS*	- Data Strobe
	ACK*	- data ACKnowledge
	ERR*	- data ERRor
	IRQ*	- Interrupt ReQuest
	CACHE*	- CACHEable

This section provides a general description of the Data Transfer Bus lines to familiarize the reader with their function. The function of some of the lines might vary during INTERRUPT-ACKNOWLEDGE cycles, described in Paragraph 2.5.4. In addition, some of the DTB lines are used in the parallel arbitration method, as described in Chapter 3.

2.2.1 *Addressing lines*

The active MASTER drives the addressing lines during the address broadcast phase of the cycle to select the SLAVE that will respond to, as well as SLAVES that will participate in, the data transfer phase.

2.2.1.1 *AD00-AD31*

During the address broadcast phase of a cycle, the active MASTER drives address lines AD00-AD31 with the address of the byte location(s) that are to be accessed during the data transfer phase of the cycle.

The address of a 4-byte group is determined by the level of address lines AD02-AD31. The address of byte locations within a 4-byte group differs only by the level of the two least significant bits of the address lines, i.e., address lines AD00 and AD01. Byte Locations are categorized as BYTE(0), BYTE(1), BYTE(2) and BYTE(3) according to the state of AD00 and AD01 in their address, as defined in Paragraph 2.5.1.2, Table 2-11.

OBSERVATION 2.1:
Lorsque l'accès à plus d'un emplacement d'octet est demandé dans un cycle, l'adresse sur les lignes AD00-AD31 est la plus basse parmi les adresses des emplacements d'octet.

2.2.1.2 *SPACE0-SPACE1*

Le MAITRE actif commande SPACE0 et SPACE1 avec un code d'espace. Il est utilisé pour sélectionner l'un des trois espaces d'adresse, ou pour déclencher un cycle de RECONNAISSANCE D'INTERRUPTION ou un cycle d'ARBITRAGE. Le VSB définit trois espaces d'adresse séparés de 4 Goctets chacun. Ces trois espaces d'adresse sont l'espace d'adresse système, l'espace d'adresse des entrées/sorties et l'espace d'adresse complémentaire.

2.2.1.3 *SIZE0-SIZE1*

Pendant la phase de diffusion d'adresse, le MAITRE actif place sur SIZE0-SIZE1 un code de dimension. Ce code de dimension indique le nombre d'emplacement(s) d'octet auquel le MAITRE désire accéder durant la phase de transfert de données. La norme VSB définit quatre dimensions de transfert de données: octet unique, double octet, triple octet et quadruple octet.

Pendant les *transferts octet unique*, le MAITRE demande l'accès à un emplacement d'octet.

Pendant les *transferts double octet*, le MAITRE demande l'accès à deux emplacements d'octet consécutifs.

Pendant les *transferts triple octet*, le MAITRE demande l'accès à trois emplacements d'octet consécutifs.

Pendant les *transferts quadruple octet*, le MAITRE demande l'accès à quatre emplacements d'octet consécutifs.

2.2.1.4 *ASACK0*-ASACK1**

ASACK0* et ASACK1* remplissent une double fonction:

a) L'ESCLAVE répondant place son code de dimension sur ces lignes. Le code de dimension de l'ESCLAVE indique au MAITRE actif le nombre maximal d'emplacements d'octet auquel l'ESCLAVE répondant peut accéder en un seul cycle. La norme VSB définit trois dimensions d'ESCLAVE: les ESCLAVES octet unique, double octet et quadruple octet.

Un *ESCLAVE octet unique* répond aux demandes de transfert de données en accédant à un seul emplacement d'octet à chaque transfert de données.

Un *ESCLAVE double octet* répond aux demandes de transfert de données en accédant à un ou deux emplacements d'octet à chaque transfert de données.

Un *ESCLAVE quadruple octet* répond aux demandes de transfert de données en accédant à un, deux, trois ou quatre emplacements d'octet à chaque transfert de données.

OBSERVATION 2.1:
When access to more than one byte location is requested in a cycle, the address on address lines AD00-AD31 is the lowest among the addresses of the byte locations.

2.2.1.2 *SPACE0-SPACE1*

The active MASTER drives SPACE0 and SPACE1 with a space code. It is used to select one of three address spaces, or to initiate an INTERRUPT-ACKNOWLEDGE or an ARBITRATION cycle. The VSB defines three separate address spaces of 4 Gbyte each. These three address spaces are the System Address Space, the I/O Address Space and the Alternate Address Space.

2.2.1.3 *SIZE0-SIZE1*

During the address broadcast phase, the active MASTER drives SIZE0-SIZE1 with a size code. This size code indicates the number of byte location(s) that the MASTER wishes to access during the data transfer phase. The VSB standard defines four sizes of data transfer: Single-Byte, Double-Byte, Triple-Byte and Quad-Byte.

During *Single-Byte transfers*, the MASTER requests access to one byte location.

During *Double-Byte transfers*, the MASTER requests access to two consecutive byte locations.

During *Triple-Byte transfers*, the MASTER requests access to three consecutive byte locations.

During *Quad-Byte transfers*, the MASTER requests access to four consecutive byte locations.

2.2.1.4 *ASACK0*-ASACK1**

ASACK0* and ASACK1* serve a dual function:

a) The responding SLAVE drives its size code on these lines. This SLAVE size code informs the active MASTER what is the maximum number of byte locations that the responding SLAVE can access in a single transfer. The VSB standard defines three SLAVE sizes: Single-Byte, Double-Byte and Quad-Byte SLAVES.

A *Single-Byte SLAVE* responds to data transfer requests by accessing only one byte location during each data transfer.

A *Double-Byte SLAVE* responds to data transfer requests by accessing either one or two byte locations during each data transfer.

A *Quad-Byte SLAVE* responds to data transfer requests by accessing either one, two, three or four byte locations during each data transfer.

b) Le premier front descendant de ASACK0* ou de ASACK1* indique au MAITRE actif que l'ESCLAVE répondant a terminé son décodage d'adresse et qu'il est prêt à commencer la phase de transfert de données.

2.2.1.5 *GA0-GA2*

Dans un système VSB, une adresse unique est affectée à chaque emplacement du fond de panier. Cette adresse est donnée par les niveaux de tension sur GA0-GA2. Lorsqu'une carte est installée dans un emplacement du fond de panier, elle utilise les niveaux de GA0-GA2 pour positionner une partie de l'ID INTERRUPTION qu'elle utilise pendant les cycles de RECONNAISSANCE D'INTERRUPTION. (De plus, comme il est décrit au chapitre 3, GA0-GA2 donnent une partie de l'ID ARBITRAGE utilisé pendant les cycles d'ARBITRAGE parallèle.) Le chapitre 5, paragraphe 5.5.3, décrit comment sont définis les niveaux de tension des bits d'adressage géographique de chaque emplacement.

2.2.2 *Lignes de données AD00-AD31*

Les lignes de données AD00-AD31 sont utilisées pour transférer les données entre le MAITRE actif et les ESCLAVES VSB. Pendant les cycles de lecture, l'ESCLAVE répondant extrait les données de sa mémoire interne et les envoie sur les lignes de données et le MAITRE actif prend les données sur les lignes de données AD00-AD31. Pendant les cycles d'écriture, le MAITRE actif fournit les données valides sur les lignes de données AD00-AD31. L'ESCLAVE répondant prend ces données et les mémorise dans sa mémoire interne. Les ESCLAVES participants peuvent aussi prendre les données transférées pendant les cycles de lecture ou d'écriture.

2.2.3 *Lignes de commande*

Les lignes de commande coordonnent le transfert de l'information d'adresse du MAITRE actif vers les ESCLAVES VSB et des données entre le MAITRE actif et les ESCLAVES sélectionnés.

2.2.3.1 *PAS**

Le MAITRE actif positionne la ligne PAS* au niveau bas pour commencer la phase de diffusion d'adresse des cycles VSB et la maintient au niveau bas pendant la durée du cycle. Les ESCLAVES VSB commencent la séquence de décodage d'adresse lorsqu'ils détectent un front descendant sur la ligne PAS*. L'ESCLAVE répondant commande ASACK0* et/ou ASACK1* au niveau bas. Une relation d'interverrouillage existe entre le MAITRE actif et l'ESCLAVE répondant aussi longtemps que le MAITRE commande la ligne PAS* au niveau bas et que l'ESCLAVE commande ASACK0* et/ou ASACK1* au niveau bas. Cette relation d'interverrouillage cesse lorsque le MAITRE actif permet à la ligne PAS* de remonter au niveau haut et après que l'ESCLAVE répondant a libéré ASACK0*-ASACK1* au niveau haut.

b) The first falling edge of either ASACK0* or ASACK1* informs the active MASTER that the responding SLAVE has finished decoding the address and that it is ready to start the data transfer phase.

2.2.1.5 *GA0-GA2*

In the VSB system, each slot in the backplane is assigned a unique address. This address is set by the voltage levels on GA0-GA2. When the board is installed in the backplane slot, it uses the levels of GA0-GA2 to set part of the INTERRUPT ID that it uses during INTERRUPT-ACKNOWLEDGE cycles. (In addition, as described in Chapter 3, GA0-GA2 set part of the ARBITRATION ID used during parallel ARBITRATION cycles.) Chapter 5, Paragraph 5.5.3, describes how the voltage levels of the geographical addressing bits in each slot are set.

2.2.2 *Data lines AD00-AD31*

Data lines AD00-AD31 are used to carry the data between the active MASTER and VSB SLAVES. During read cycles, the responding SLAVE retrieves data from its internal storage and drives it on the data lines, and the active MASTER captures the data from data lines AD00-AD31. During write cycles, the active MASTER drives data lines AD00-AD31 with valid data. The responding SLAVE captures this data and stores it in its internal storage. Participating SLAVES might also capture the data transferred during read or write cycles.

2.2.3 *Control lines*

The control lines coordinate the transfer of the addressing information from the active MASTER to VSB SLAVES and of the data between the active MASTER and the selected SLAVES.

2.2.3.1 PAS*

The active MASTER drives PAS* low to start the address broadcast phase of VSB cycles and maintains it low for the duration of the cycle. VSB SLAVES start the address decode sequence when they detect a falling edge on PAS*. The responding SLAVE drives ASACK0* and/or ASACK1* low. An interlocked relationship exists between the active MASTER and the responding SLAVE for as long as the MASTER drives PAS* low and the SLAVE drives ASACK0* and/or ASACK1* low. This interlocked relationship ends when the active MASTER allows PAS* to go high, and after the responding SLAVE releases ASACK0*-ASACK1* to high.

2.2.3.2 *AC*

La ligne AC remplit une double fonction:

a) Combinée avec WAIT*, elle fournit aux ESCLAVES VSB un moyen pour prolonger la phase de diffusion d'adresse afin de s'adapter à leurs spécifications de décodage d'adresse. Lorsque le MAITRE actif détecte un niveau bas sur WAIT*, il ne lui est pas permis de terminer la phase de diffusion d'adresse et de commencer la phase de transfert de données avant d'avoir détecté un niveau haut sur la ligne AC.

b) Tous les ESCLAVES libèrent AC au niveau haut en réponse à la phase de diffusion d'adresse. L'ESCLAVE répondant commande la ligne AC au niveau bas après avoir accédé au dernier emplacement d'octet auquel il peut accéder sans avoir à demander une nouvelle adresse de la part du MAITRE actif. Lorsque le MAITRE actif détecte un niveau bas sur AC avant de recevoir ACK* au niveau bas, il termine le cycle.

2.2.3.3 *WR**

Le MAITRE actif commande la ligne WR* pendant la phase de diffusion d'adresse pour informer les ESCLAVES sur la direction du transfert de données. Le MAITRE actif positionne WR* au niveau haut lorsqu'il acquiert des données des emplacements d'octet fournies par l'ESCLAVE répondant. Le MAITRE actif commande WR* au niveau bas pour mémoriser des données dans les emplacements d'octet de l'ESCLAVE répondant.

2.2.3.4 *LOCK**

Les MAITRES commandent la ligne LOCK* au niveau bas pour déclencher les cycles d'ACCES INDIVISIBLE. Certains ESCLAVES VSB permettent l'accès à des emplacements d'octet qu'ils fournissent sur plus d'un port; par exemple, leur mémoire peut être accessible depuis le bus global du système. De tels ESCLAVES verrouilleront en général l'accès à partir des autres ports aux emplacements d'octet sélectionnés pendant les cycles d'ACCES INDIVISIBLE. Cela assure l'indivisibilité du cycle VSB.

2.2.3.5 *DS**

Le MAITRE actif commande la ligne DS* au niveau bas pour indiquer aux ESCLAVES sélectionnés que la phase de diffusion d'adresse est terminée et que la phase de transfert de données est en cours.

2.2.3.6 *WAIT**

WAIT* remplit une double fonction:

a) Combinée avec AC, la ligne WAIT* fournit aux ESCLAVES VSB un moyen de prolonger la phase de diffusion d'adresse pour s'adapter à leurs spécifications de décodage d'adresse. Quand le MAITRE actif détecte un niveau bas sur WAIT*, il ne lui est pas permis de terminer la phase de diffusion d'adresse et de commencer la phase de transfert de données avant d'avoir détecté un niveau haut sur la ligne AC.

2.2.3.2 *AC*

The AC line serves a dual function:

a) In conjunction with WAIT*, it provides VSB SLAVES with a mechanism to prolong the address broadcast phase to fit their address decoding requirements. When the active MASTER detects a low level on WAIT*, it is not allowed to complete the address broadcast phase and start the data transfer phase until it detects a high level on AC.

b) All SLAVES release AC to high in response to the address broadcast phase. The responding SLAVE drives AC low after it has accessed the last byte location it can access without requiring a new address from the active MASTER. When the active MASTER detects a low level on AC before it receives ACK* low, it terminates the cycle.

2.2.3.3 *WR**

The active MASTER drives the WR* line during the address broadcast phase to inform SLAVES of the direction of the data transfer. The active MASTER drives WR* high when it retrieves data from byte locations provided by the responding SLAVE. The active MASTER drives WR* low to store data in byte locations provided by the responding SLAVE.

2.2.3.4 *LOCK**

MASTERS drive the LOCK* line low to initiate INDIVISIBLE-ACCESS cycles. Some VSB SLAVES allow access to the byte locations they provide from more than one port; for example, their memory might be accessible over the global system bus. Such SLAVES will typically lock out access from other ports to the byte locations that they provide for the duration of the INDIVISIBLE-ACCESS cycle. This ensures the indivisibility of the VSB cycle.

2.2.3.5 *DS**

The active MASTER drives DS* low to indicate to the selected SLAVES that the address broadcast phase is complete and that the data transfer phase is in progress.

2.2.3.6 *WAIT**

WAIT* serves a dual function:

a) In conjunction with AC, it provides VSB SLAVES with a mechanism to prolong the address broadcast phase to fit their address decoding requirements. When the active MASTER detects a low level on WAIT*, it is not allowed to complete the address broadcast phase and start the data transfer phase until it detects a high level on AC.

b) Les ESCLAVES participants libèrent la ligne WAIT* au niveau haut pour indiquer qu'ils ont terminé le transfert de données courant et sont prêts à participer à un autre transfert. Ce mécanisme permet au MAITRE actif de procéder aux opérations de diffusion et d'écoute de données. Pendant la diffusion de données, les données que le MAITRE actif place sur les lignes de données sont prélevées par les ESCLAVES participants. Pendant une opération d'écoute de données, les données que l'ESCLAVE répondant place sur les lignes de données sont prélevées par les ESCLAVES participants.

Par exemple, après avoir effectué une opération de cache sur un élément de donnée, le système contient deux copies, l'une en mémoire globale du VSB et l'autre en mémoire cache privée. Pour s'assurer que les deux copies sont identiques, le contrôleur de cache peut surveiller tous les cycles VSB et soit invalider la copie en cache soit la remettre à jour lorsqu'il détecte un cycle d'écriture vers la copie globale de la donnée. Cependant, cette opération doit être terminée avant que le transfert qui l'a provoquée soit terminé, de telle sorte que le contrôleur de cache soit prêt à en surveiller une autre. Cette exigence est assurée par la ligne WAIT* qui peut être maintenue au niveau bas par un tel contrôleur de cache, pour empêcher le MAITRE actif de terminer le transfert en cours avant que le cache ait été remis à jour.

2.2.3.7 *ACK**

Pendant les cycles d'écriture, l'ESCLAVE répondant commande la ligne ACK* au niveau bas pour informer le MAITRE actif qu'il a prélevé les données sur les lignes AD00-AD31. Pendant les cycles de lecture, l'ESCLAVE répondant commande la ligne ACK* au niveau bas pour informer le MAITRE que des données valides ont été placées sur tout ou partie des lignes AD00-AD31. Le MAITRE actif interprète le front descendant de ACK* comme un signal de l'ESCLAVE répondant, signifiant qu'il a terminé sa participation à la phase de transfert des données du cycle.

2.2.3.8 *ERR**

L'ESCLAVE répondant commande ERR* au niveau bas pour informer le MAITRE actif qu'une erreur (par exemple une erreur de parité) a été détectée pendant le cycle de transfert de données. Le MAITRE actif interprète le front descendant de ERR* comme un signal de l'ESCLAVE répondant, signifiant qu'il a terminé sa participation à la phase de transfert de données du cycle.

2.2.3.9 *IRQ**

La ligne IRQ* est utilisée par les ESCLAVES pour demander une interruption à un MAITRE. Le MAITRE répond à une demande d'interruption de deux manières:

a) Il peut déclencher un cycle de RECONNAISSANCE D'INTERRUPTION. Les ESCLAVES qui ont une demande d'interruption en attente participent au cycle de RECONNAISSANCE D'INTERRUPTION pour déterminer quel ESCLAVE aura sa demande d'interruption servie. Le MAITRE actif lit alors l'information de MOT D'ETAT/ID de cet ESCLAVE.

b) Participating SLAVES release WAIT* to high to indicate that they are done with the current data transfer and are ready to participate in a new one. This allows the active MASTER to perform broadcast and broadcall operations. During a broadcast operation, the data that the active MASTER drives on the data lines is captured by participating SLAVES. During a broadcall operation, the data that the responding SLAVE drives on the data lines is captured by participating SLAVES.

For example, after caching a data item, the system contains two copies of it, one in global VSB memory and one in private cache memory. To ensure that these two copies are identical, the cache controller can monitor all VSB cycles, and either invalidate the cached copy or update it when it detects a write cycle to the global copy of the data. However, this operation should be completed before the transfer that caused it is terminated, so that the cache controller is ready to monitor a new one. This requirement is handled by the WAIT* line, which can be maintained low by such a cache controller, to prevent the active MASTER from terminating the current transfer until the cache has been updated.

2.2.3.7 *ACK**

During write cycles, the responding SLAVE drives the ACK* line low to inform the active MASTER that it has captured the data on AD00-AD31. During read cycles the responding SLAVE drives ACK* low to inform the MASTER that valid data has been placed on some or all of AD00-AD31. The active MASTER interprets the falling edge of ACK* as a signal that the responding SLAVE has completed its participation in the data transfer portion of the cycle.

2.2.3.8 *ERR**

The responding SLAVE drives ERR* low to inform the active MASTER that an error (e.g. a parity error) has been encountered during the data transfer cycle. The active MASTER interprets the falling edge of ERR* as a signal that the responding SLAVE has completed its participation in the data transfer portion of the cycle.

2.2.3.9 *IRQ**

The IRQ* line is used by SLAVES to request an interrupt from a MASTER. The MASTER responds to an interrupt request in one of two ways:

a) It might initiate an INTERRUPT-ACKNOWLEDGE cycle. SLAVES that have an interrupt request pending participate in the INTERRUPT-ACKNOWLEDGE cycle to determine the one SLAVE whose interrupt request will be serviced. The active MASTER then reads STATUS/ID information from this SLAVE.

b) Il peut déclencher un cycle de lecture VSB, interrogeant tour à tour les unités pour obtenir l'information de MOT D'ETAT/ID jusqu'à ce qu'une unité qui a une demande d'interruption en attente soit trouvée.

2.2.3.10 *CACHE**

L'ESCLAVE répondant commande CACHE* pour indiquer si les données transférées peuvent être copiées dans un cache. Par exemple, une carte mémoire peut commander la ligne CACHE* au niveau bas lorsqu'elle répond au cycle. Le MAITRE actif et les ESCLAVES participants peuvent alors copier les données dans le cache. Par ailleurs, un contrôleur d'entrées/sorties ne commande pas CACHE* au niveau bas pendant le cycle empêchant ainsi un MAITRE de copier dans le cache les données lues à partir d'un registre d'état.

2.3 *Modules du DTB - Description générale*

Les figures 2-2 et 2-3, pages 52 et 54, présentent les schémas-blocs pour les deux types de modules fonctionnels DTB: MAITRE et ESCLAVE.

REGLE 2.1:
Les lignes de signaux de sortie présentées en lignes continues dans les figures 2-2 et 2-3 DOIVENT être commandées par le module, à moins qu'il les commande toujours au niveau haut.

OBSERVATION 2.2:
SI une ligne de sortie n'est pas commandée,
ALORS les impédances d'adaptation assurent qu'elle est au niveau haut.

REGLE 2.2:
Les lignes de signaux d'entrée présentées en lignes continues dans les figures 2-2 et 2-3 DOIVENT être surveillées et recevoir des réponses de façon appropriée.

OBSERVATION 2.3:
Les REGLES et AUTORISATIONS pour commander et surveiller des lignes de signaux présentées en pointillé dans les figures 2-2 et 2-3 sont données dans les tableaux 2-1 et 2-2.

b) It might initiate a VSB read cycle, polling the various devices for STATUS/ID information until a device that has an interrupt request pending is found.

2.2.3.10 *CACHE**

The responding SLAVE drives CACHE* to indicate whether the data transferred is cacheable. For example, a memory board might drive the CACHE* line low when it responds to a cycle. The active MASTER, as well as participating SLAVES can then cache the data. On the other hand, an I/O controller does not drive CACHE* low during the cycle, preventing the MASTER from caching data that is read from a status register.

2.3 *DTB modules - Basic description*

Figures 2-2 and 2-3, pages 53 and 55, provide block diagrams for the two types of DTB functional modules: MASTER and SLAVE.

RULE 2.1:
Output signal lines shown with solid lines in Figures 2-2 and 2-3 MUST be driven by the module, unless it would always drive them high.

OBSERVATION 2.2:
IF an output signal line is not driven,
THEN terminators ensure that it is high.

RULE 2.2:
Input signal lines shown with solid lines in Figures 2-2 and 2-3 MUST be monitored and responded to in the appropriate fashion.

OBSERVATION 2.3:
RULES and PERMISSIONS for driving and monitoring signal lines shown with dotted lines in Figures 2-2 and 2-3 are given in Tables 2-1 and 2-2.

2.3.1 *MAITRE*

Le schéma-bloc du MAITRE est présenté dans la figure 2-2. Les lignes pointillées montrent les signaux dont l'utilisation varie selon les différents types de MAITRES. Le tableau 2-1 montre comment les différents types de MAITRES utilisent ces lignes.

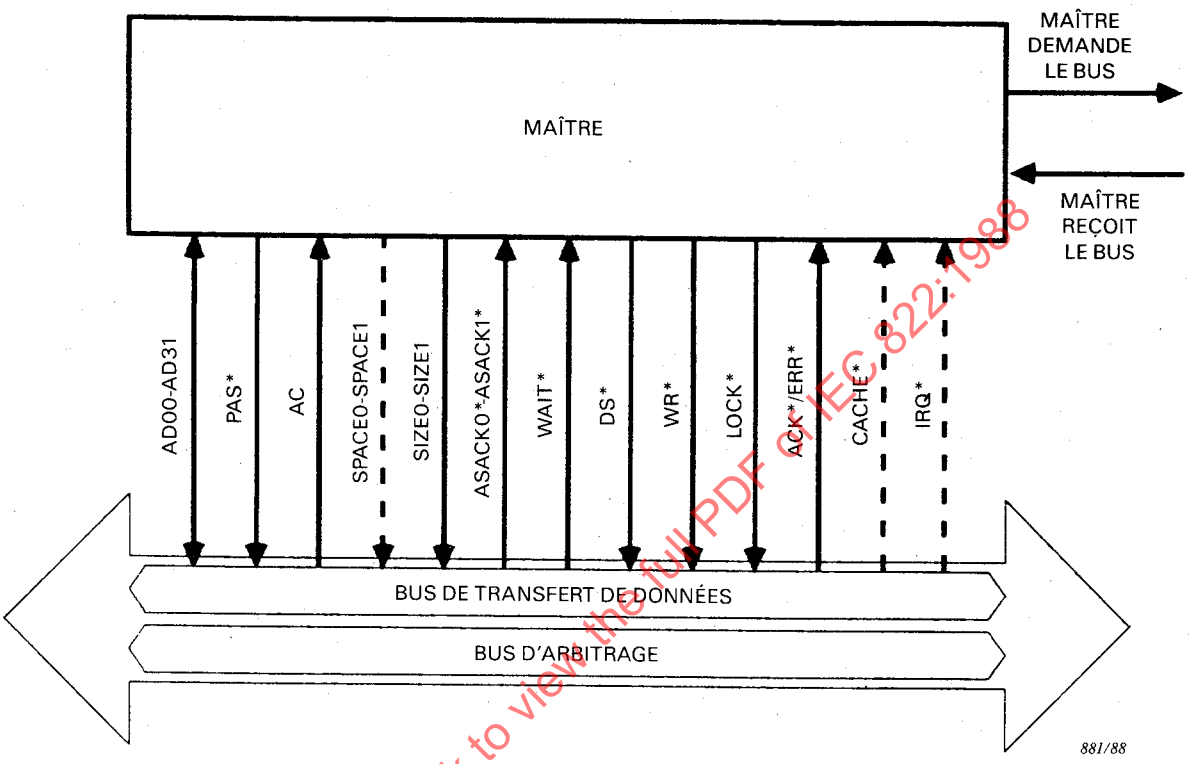


Fig. 2-2. - Schéma-bloc: MAITRE.

Tableau 2-1

REGLES et AUTORISATIONS qui spécifient l'utilisation des lignes pointillées par les différents types de MAITRES

Type de MAITRE	Utilisation des lignes pointillées
IHP	DOIT surveiller IRQ*
IHV	DOIT surveiller IRQ*
TOUS LES MAITRES	PEUVENT ou NON surveiller CACHE*

Note:

Les mnémoniques IHP et IHV sont définis au paragraphe 2.4.3.1, tableau 2-7.

2.3.1 *MASTER*

The block diagram of the MASTER is shown in Figure 2-2. The dotted lines in the diagram show signals whose use varies among the various types of MASTERS. Table 2-1 specifies how the various types of MASTERS use these lines.

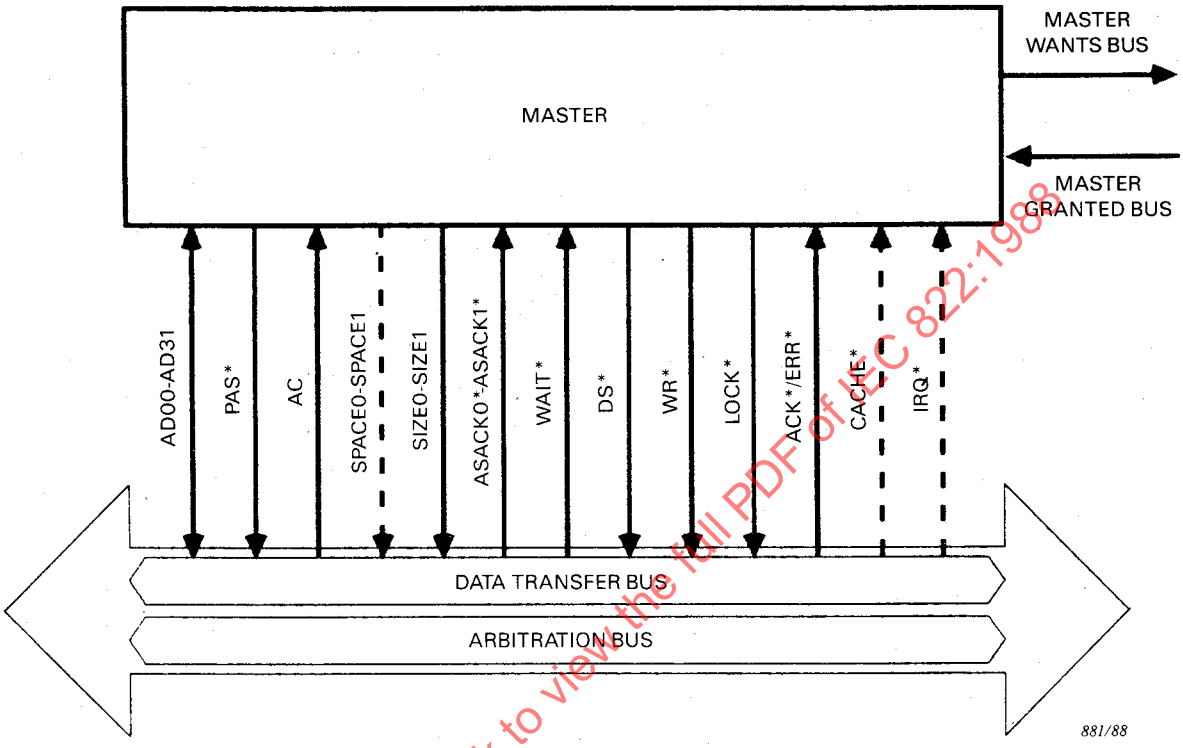


Fig. 2-2. - Block diagram: MASTER.

Table 2-1

RULES and PERMISSIONS that specify the use of the dotted lines by the various types of MASTERS

Type of MASTER	Use of dotted lines
IHP	MUST monitor IRQ*
IHV	MUST monitor IRQ*
ALL MASTERS	MAY or MAY NOT monitor CACHE*

Note:
The mnemonics IHP and IHV are defined in Paragraph 2.4.3.1, Table 2-7.

2.3.2 *ESCLAVE*

Le schéma-bloc de l'ESCLAVE est présenté dans la figure 2-3. Les lignes pointillées dans le schéma montrent les signaux dont l'utilisation varie selon les différents types d'ESCLAVES. Le tableau 2-2 montre comment les différents types d'ESCLAVES utilisent ces lignes.

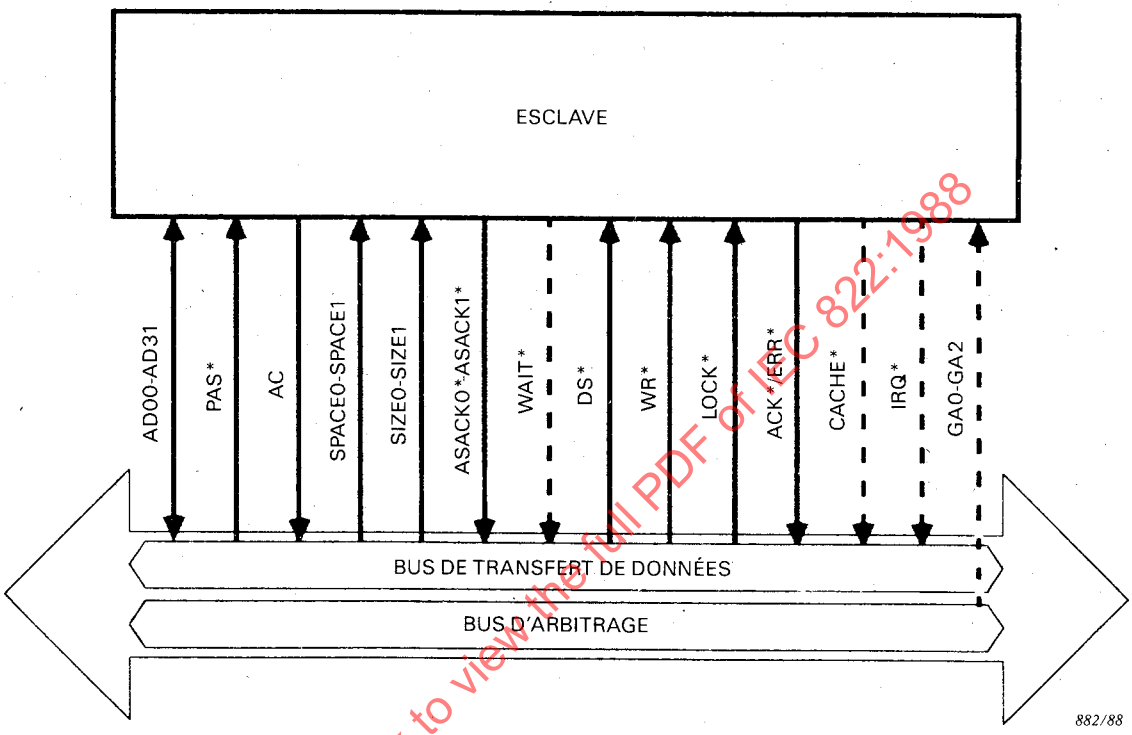


Fig. 2-3. - Schéma-bloc: ESCLAVE.

Tableau 2-2

REGLES et AUTORISATIONS qui spécifient l'utilisation des lignes pointillées par les différents types d'ESCLAVES

Type d'ESCLAVE	Utilisation des lignes pointillées
D08	DOIT surveiller et commander AD24-AD31 PEUT ou NON surveiller ou commander AD00-AD23 PEUT ou NON commander WAIT*
D16	DOIT surveiller et commander AD16-AD31 PEUT ou NON surveiller ou commander AD00-AD15 PEUT ou NON commander WAIT*
D32	DOIT surveiller et commander AD00-AD31 PEUT ou NON commander WAIT*
INTP	DOIT commander IRQ* PEUT ou NON commander WAIT* PEUT ou NON surveiller GA0-GA2

(Suite à la page 56)

2.3.2 *SLAVE*

The block diagram of the SLAVE is shown in Figure 2-3. The dotted lines in the diagram show signals whose use varies among the various types of SLAVES. Table 2-2 shows how the various types of SLAVES use these lines.

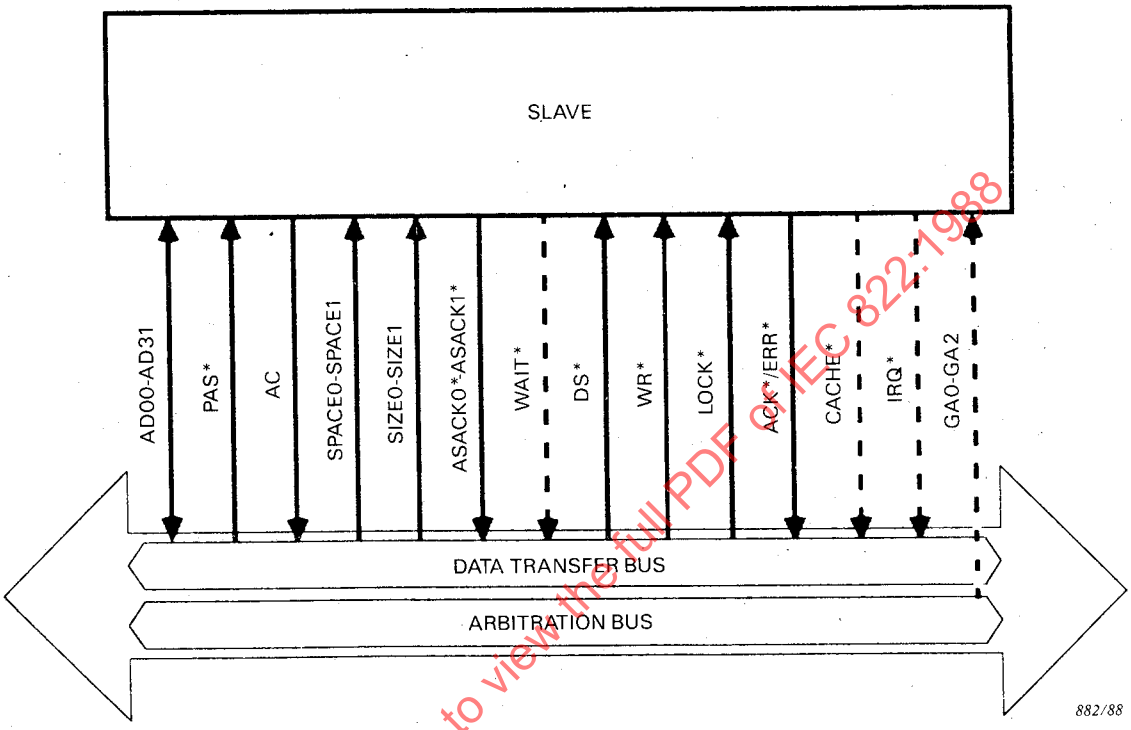


Fig. 2-3. - Block diagram: SLAVE.

Table 2-2

RULES and PERMISSIONS that specify the use of the dotted lines by the various types of SLAVES

Type of SLAVE	Use of dotted lines
D08	MUST monitor and drive AD24-AD31 MAY or MAY NOT monitor or drive AD00-AD23 MAY or MAY NOT drive WAIT*
D16	MUST monitor and drive AD16-AD31 MAY or MAY NOT monitor or drive AD00-AD15 MAY or MAY NOT drive WAIT*
D32	MUST monitor and drive AD00-AD31 MAY or MAY NOT drive WAIT*
INTP	MUST drive IRQ* MAY or MAY NOT drive WAIT* MAY or MAY NOT monitor GA0-GA2

(Continued on page 57)

Tableau 2-2 (suite)

Type d'ESCLAVE	Utilisation des lignes pointillées
INTV	DOIT commander IRQ* DOIT commander WAIT* DOIT surveiller GA0-GA2
TOUS LES ESCLAVES	PEUVENT ou NON commander ERR* PEUVENT ou NON commander CACHE*

Notes:

1.- Les mnémoniques D08, D16 et D32 sont définis au paragraphe 2.4.2.2, tableau 2-5.

2.- Les mnémoniques INTP et INTV sont définis au paragraphe 2.4.3.1, tableau 2-7.

OBSERVATION 2.4:

Tous les ESCLAVES VSB surveillent chaque diffusion d'adresse. Certains ESCLAVES décodent l'adresse, déterminent qu'ils ne sont pas concernés et libèrent AC et WAIT* au niveau haut immédiatement après avoir décodé l'adresse. Ces ESCLAVES sont appelés ESCLAVES au repos.

OBSERVATION 2.5:

Certains ESCLAVES décodent l'adresse et décident qu'ils doivent réagir. Ces ESCLAVES libèrent AC au niveau haut après le décodage d'adresse. Cependant, ils maintiennent WAIT* au niveau bas, le libérant au niveau haut seulement après avoir accompli la phase de transfert de données et sont prêts à participer à une nouvelle phase de transfert de données ou à un nouveau cycle. Ces ESCLAVES sont appelés ESCLAVES participants.

OBSERVATION 2.6:

Il y a un seul ESCLAVE qui répond au cycle. Il répond en acquittant la phase de diffusion d'adresse avec les signaux ASACK0* et/ou ASACK1*, et il répond à la phase de transfert de données avec le signal ACK* et/ou le signal ERR*. Cet ESCLAVE est appelé l'ESCLAVE répondant.

2.4 *Possibilités des MAITRES et des ESCLAVES*

Le protocole VSB définit deux types de cycle de base qui sont utilisés pour le transfert de données: le cycle de TRANSFERT UNIQUE et le cycle de TRANSFERT PAR BLOC. Un autre type de cycle, le cycle UNIQUEMENT D'ADRESSAGE est utilisé pour étendre les possibilités des MAITRES pendant la phase de diffusion d'adresses. Chacun de ces types de cycle peut être utilisé pour accéder aux emplacements d'octet dans n'importe lequel de trois espaces d'adresse: l'espace d'adresse système, l'espace d'adresse complémentaire et l'espace d'adresse des entrées/sorties. De plus, chacun de ces cycles peut être rendu indivisible en commandant au niveau bas la ligne LOCK*. Un type de cycle supplémentaire est le cycle de RECONNAISSANCE D'INTERRUPTION, utilisé par les MAITRES et ESCLAVES pour coordonner le service des interruptions. Cette section décrit comment les MAITRES et des ESCLAVES peuvent exécuter les cycles cités ci-dessus et établit les mnémoniques qui définissent ces possibilités.

Table 2-2 (continued)

Type of SLAVE	Use of dotted lines
INTV	MUST drive IRQ* MUST drive WAIT* MUST monitor GA0-GA2
ALL SLAVES	MAY or MAY NOT drive ERR* MAY or MAY NOT drive CACHE*

Notes:

1.- The mnemonics D08, D16 and D32 are defined in Paragraph 2.4.2.2, Table 2-5.

2.- The mnemonics INTP and INTV are defined in Paragraph 2.4.3.1, Table 2-7.

OBSERVATION 2.4:

All VSB SLAVES monitor each address broadcast. Some SLAVES decode the address, determine that they do not have to act upon it, and release AC and WAIT* to high immediately after they finish decoding the address. These SLAVES are called idle SLAVES.

OBSERVATION 2.5:

Some SLAVES decode the address and determine that they need to act upon it. These SLAVES release AC to high after they finish decoding the address. However, they maintain WAIT* low, releasing it to high only after they have acted upon the data transfer phase and are ready to participate in a new data transfer phase or a new cycle. These SLAVES are called participating SLAVES.

OBSERVATION 2.6:

No more than one SLAVE responds to the cycle. It does so by acknowledging the address broadcast phase on ASACK0* and/or ASACK1*, and the data transfer phase on ACK* and/or ERR*. This SLAVE is called the responding SLAVE.

2.4 *Capabilities of MASTERS and SLAVES*

The protocols of the VSB define two basic types of cycles that are used to transfer data: the SINGLE-TRANSFER cycle and the BLOCK-TRANSFER cycle. Another type of cycle, the ADDRESS-ONLY cycle, is used to extend the capabilities of MASTERS during the address broadcast phase. Each of these types of cycles can be used to access byte locations in any of the three address spaces: System Address Space, Alternate Address Space and I/O Address Space. In addition, each of these cycles can be made indivisible by driving the LOCK* line low. An additional type of cycle is the INTERRUPT-ACKNOWLEDGE cycle which is used by MASTERS and SLAVES to coordinate the service of interrupts. This section describes the capabilities of MASTERS and SLAVES to execute the above cycles and establishes mnemonics that define those capabilities.

Une relation d'interverrouillage existe entre les MAITRES et les ESCLAVES au cours des cycles VSB. Tel qu'il est présenté dans la figure 2-4, le cycle commence après que le MAITRE a obtenu l'usage exclusif du bus de transfert de données (comme décrit au chapitre 3). Le MAITRE actif déclenche le cycle en diffusant l'information d'adresse aux ESCLAVES VSB qui acquittent et répondent à la diffusion d'adresse. Le MAITRE actif maintient la diffusion d'adresse jusqu'à ce qu'il détecte cet acquittement et termine alors la diffusion d'adresse.

Le MAITRE actif peut alors déclencher un ou plusieurs transferts de données ou il peut terminer le cycle sans effectuer aucun transfert de données. Lorsque les ESCLAVES détectent un transfert de données, ils l'acquittent. Le MAITRE maintient le transfert de données jusqu'à ce qu'il détecte cet acquittement. Après que le MAITRE a terminé le transfert de données, il peut exécuter un autre transfert de données ou peut terminer le cycle. Lorsque les ESCLAVES détectent une fin de cycle, ils l'acquittent. Cela termine le cycle.

Les possibilités des MAITRES et des ESCLAVES qui ont trait à la phase de diffusion d'adresse sont décrites au paragraphe 2.4.1.

Les possibilités des MAITRES et des ESCLAVES qui ont trait à la phase de transfert de données sont décrites au paragraphe 2.4.2.

Les possibilités des MAITRES et des ESCLAVES qui ont trait aux cycles de RECONNAISSANCE D'INTERRUPTION sont décrites au paragraphe 2.4.3.

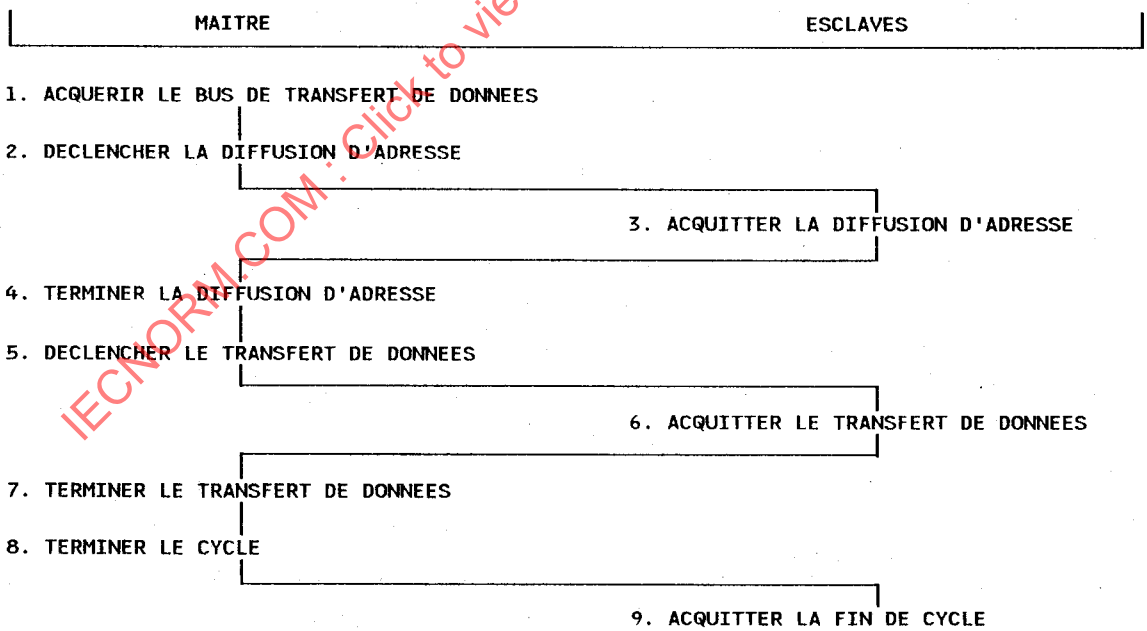


Fig. 2-4. - Organigramme général d'un cycle VSB.

An interlocked relationship exists between MASTERS and SLAVES during the course of VSB cycles. As shown in Figure 2-4, the cycle begins after the MASTER has been granted exclusive use of the Data Transfer Bus (as described in Chapter 3). The active MASTER initiates the cycle by broadcasting addressing information to VSB SLAVES, which acknowledge and respond to the address broadcast. The active MASTER maintains the address broadcast until it detects this acknowledgment and then terminates the address broadcast.

The active MASTER might then initiate one or more data transfers, or it might terminate the cycle without performing any data transfers. When SLAVES detect a data transfer they acknowledge it. The MASTER maintains the data transfer until it detects this acknowledgment. After the MASTER terminates the data transfer it might execute another data transfer, or it might terminate the cycle. When SLAVES detect a cycle termination they acknowledge it. This concludes the cycle.

The capabilities of MASTERS and SLAVES that pertain to the address broadcast phase are described in Paragraph 2.4.1.

The capabilities of MASTERS and SLAVES that pertain to the data transfer phase are described in Paragraph 2.4.2.

The capabilities of MASTERS and SLAVES that pertain to INTERRUPT-ACKNOWLEDGE cycles are described in Paragraph 2.4.3.

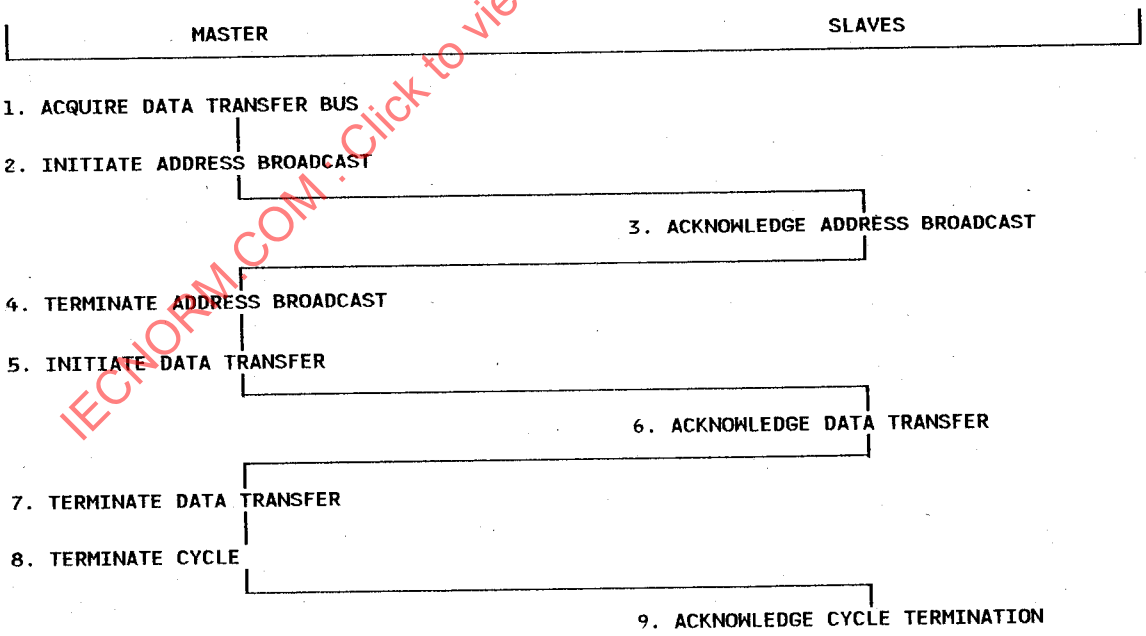


Fig. 2-4. - General flow of a VSB cycle.

2.4.1 *Possibilités d'adressage*

La plus petite unité d'adressage qui peut être fournie par des ESCLAVES est l'emplacement d'octet. Un groupe de quatre emplacements d'octet accessibles simultanément dans un seul transfert de données est appelé un groupe de 4 octets. Durant le déroulement des cycles VSB, un MAITRE peut demander l'accès à un, deux, trois ou quatre emplacements d'octet fournis par les ESCLAVES.

La notation *cycle OCTET(x)* est utilisée pour décrire un cycle qui demande l'accès à l'emplacement x, où x peut prendre les valeurs 0, 1, 2 ou 3. Par exemple, un cycle OCTET(0) demande l'accès à l'emplacement d'octet(0) d'un groupe de 4 octets.

La notation *cycle OCTET(x-y)* est utilisée pour décrire un cycle qui demande l'accès à un ensemble d'emplacements d'octet consécutifs dans le même groupe de 4 octets où x-y définit une dimension entre 0 et 3. Par exemple, un cycle OCTET(1-2) demande l'accès aux emplacements OCTET(1) et OCTET(2) du groupe de 4 octets.

Une notation spéciale *cycle OCTET(x-y)(z)* est utilisée pour décrire des cycles qui spécifient l'accès à des emplacements d'octet dans deux groupes consécutifs de 4 octets. Par exemple, un cycle OCTET(1-3)(0) demande l'accès aux emplacements OCTET(1-3) du groupe n de 4 octets et un accès à l'emplacement OCTET(0) du groupe de 4 octets n + 1. Des cycles multiples sont toujours nécessaires pour accomplir ce type d'accès. Si dans l'exemple ci-dessus, un ESCLAVE D32 est sélectionné, alors on accède aux emplacements OCTET(1-3) du groupe de 4 octets n durant le premier cycle. On accédera à l'emplacement OCTET(0) du groupe de 4 octets n + 1 dans un cycle suivant, déclenché par la logique de la carte qui supporte le dimensionnement dynamique du bus. (Les mnémoniques D08, D16 et D32 sont définis dans le paragraphe 2.4.2.2, tableau 2-5.)

Quatre types de cycles sont définis:

Lors des *cycles quadruple octet*, le MAITRE actif demande l'accès à quatre emplacements d'octet consécutifs. Il y a quatre types de cycles quadruple octet: OCTET(0-3), OCTET(1-3)(0), OCTET(2-3)(0-1) et OCTET(3)(0-2). Le seul cycle quadruple OCTET qui peut être accompli en un seul transfert de données est le cycle OCTET(0-3) vers un ESCLAVE D32.

Lors des *cycles triple octet*, le MAITRE actif demande l'accès à trois emplacements d'octet consécutifs. Les cycles triple octet qui peuvent être accomplis en un seul transfert de données vers un ESCLAVE D32 sont les cycles: OCTET(0-2) et OCTET(1-3). Les cycles triple octet OCTET(2-3)(0) et OCTET(3)(0-1) de même que tous les cycles triple octet qui s'adressent à des ESCLAVES du type D16 ou D08 requièrent des transferts de données multiples.

Lors des *cycles double octet*, le MAITRE actif demande l'accès à deux emplacements d'octet consécutifs. Les quatre types de cycles double octet sont: OCTET(0-1), OCTET(1-2), OCTET(2-3) et OCTET(3)(0). Les cycles OCTET(0-1) et OCTET(2-3) peuvent être accomplis en un seul transfert de données vers des ESCLAVES du type D16 et D32. De plus, le cycle OCTET(1-2) peut être exécuté en un seul transfert de données vers un ESCLAVE du type D32. Tous les autres types de cycles double octet demandent de multiples transferts de données.

2.4.1 *Addressing capabilities*

The smallest addressable unit of storage that can be provided by SLAVES is the byte location. A group of four byte locations which can be accessed simultaneously in a single data transfer is called a 4-byte group. During the course of VSB cycles a MASTER might request access to either one, two, three or four byte locations provided by SLAVES.

The notation *BYTE(x) cycle* is used to describe a cycle that requests access to byte location x, where x can assume the value 0, 1, 2 or 3. For example, a BYTE(0) cycle requests access to byte location BYTE(0) of the 4-byte group.

The notation *BYTE(x-y) cycle* is used to describe a cycle that requests access to a set of consecutive byte locations in the same 4-byte group, where x-y defines a range between 0 and 3. For example, a BYTE(1-2) cycle requests access to byte locations BYTE(1) and BYTE(2) of the 4-byte group.

A special notation *BYTE(x-y)(z) cycle* is used to describe cycles that specify access to byte locations in two consecutive 4-byte groups. For example, a BYTE(1-3)(0) cycle requests access to byte locations BYTE(1-3) of 4-byte group n and byte location BYTE(0) of 4-byte group n + 1. Multiple cycles are always required to complete this type of data access. If this example selects a D32 SLAVE, then byte locations BYTE(1-3) of 4-byte group n will be accessed during the first cycle. Byte location BYTE(0) of 4-byte group n + 1 will be accessed in a subsequent cycle initiated by on-board logic that supports dynamic bus sizing. (The mnemonics D08, D16 and D32 are defined in Paragraph 2.4.2.2, Table 2-5.)

Four types of cycles are defined:

During *Quad-Byte cycles*, the active MASTER requests access to four consecutive byte locations. There are four types of Quad-Byte cycles: BYTE(0-3), BYTE(1-3)(0), BYTE(2-3)(0-1) and BYTE(3)(0-2). The only Quad-Byte cycle that can be completed in a single data transfer is a BYTE(0-3) cycle to a D32 SLAVE.

During *Triple-Byte cycles*, the active MASTER requests access to three consecutive byte locations. Triple-Byte cycles that can be completed in a single data transfer to a D32 SLAVE are: BYTE(0-2) and BYTE(1-3). BYTE(2-3)(0) and BYTE(3)(0-1) Triple-Byte cycles, as well as any Triple-Byte cycle to D16 or D08 SLAVES, require multiple data transfers.

During *Double-Byte cycles*, the active MASTER requests access to two consecutive byte locations. The four types of Double-Byte cycles are: BYTE(0-1), BYTE(1-2), BYTE(2-3) and BYTE(3)(0). BYTE(0-1) and BYTE(2-3) cycles can be completed in a single data transfer to both D16 and D32 SLAVES. In addition, a BYTE(1-2) cycle can be completed in a single data transfer to a D32 SLAVE. All other types of Double-Byte cycles require multiple data transfers.

Lors des *cycles octet unique*, le MAITRE actif accède à un seul emplacement d'octet. Les quatre types de transfert octet unique sont: OCTET(0), OCTET(1), OCTET(2) et OCTET(3). Tous les types de cycle octet unique s'effectuent en un seul transfert de données.

2.4.1.1 *Possibilités de base pour l'adressage*

Les MAITRES diffusent une adresse sur le VSB au début de chaque cycle. Cette adresse peut sélectionner des emplacements d'octet dans l'espace d'adresse système, l'espace d'adresse complémentaire ou l'espace d'adresse des entrées/sorties. Le tableau 2-3 montre les mnémoniques qui sont utilisés pour décrire les possibilités d'adressage des MAITRES et ESCLAVES.

Tableau 2-3

Mnémoniques qui spécifient les possibilités d'adressage

Le mnémonique suivant	Lorsqu'il est appliqué à un	Signifie qu'il
SAS	MAITRE	Peut générer des cycles vers l'espace d'adresse système.
	ESCLAVE	Peut accepter des cycles de l'espace d'adresse système
ALTAS	MAITRE	Peut générer des cycles vers l'espace d'adresse complémentaire.
	ESCLAVE	Peut accepter des cycles de l'espace d'adresse complémentaire
IOAS	MAITRE	Peut générer des cycles vers l'espace d'adresse d'entrée/sortie.
	ESCLAVE	Peut accepter des cycles de l'espace d'adresse d'entrée/sortie

Le MAITRE actif commande un code d'espace sur les lignes SPACE0-SPACE1 pendant la phase de diffusion d'adresse. Ce code d'espace informe les ESCLAVES si l'adresse diffusée sélectionne des emplacements d'octet dans l'espace d'adresse système, dans l'espace d'adresse complémentaire ou dans l'espace d'adresse d'entrée/sortie comme défini au paragraphe 2.5.1.2, tableau 2-9.

OBSERVATION 2.7:

En plus des trois modes d'adressage décrits ici, il y a un quatrième mode qui est utilisé pour sélectionner les cycles de RECONNAISSANCE D'INTERRUPTION et les cycles d'ARBITRAGE. Ces cycles peuvent être discernés des autres cycles VSB par la présence de niveaux différents sur les lignes SPACE0-SPACE1, comme décrit au paragraphe 2.5.4.2, tableau 2-18.

During *Single-Byte cycles*, the active MASTER accesses one byte location. The four types of Single-Byte cycles are: BYTE(0), BYTE(1), BYTE(2) and BYTE(3). All types of Single-Byte cycles complete in a single data transfer.

2.4.1.1 *Basic addressing capabilities*

MASTERS broadcast an address over the VSB at the beginning of each cycle. This address might select byte locations in the System Address Space, in the Alternate Address Space or in the I/O Address Space. Table 2-3 shows the mnemonics that are used to describe the addressing capabilities of MASTERS and SLAVES.

Table 2-3

Mnemonics that specify addressing capabilities

The following mnemonic	When applied to a	Means that it can
SAS	MASTER	Generate cycles to the System Address Space.
	SLAVE	Accept cycles to the System Address Space
ALTAS	MASTER	Generate cycles to the Alternate Address Space.
	SLAVE	Accept cycles to the Alternate Address Space
IOAS	MASTER	Generate cycles to the I/O Address Space.
	SLAVE	Accept cycles to the I/O Address Space

The active MASTER drives a space code on SPACE0-SPACE1 during the address broadcast phase. This space code informs the SLAVES whether the broadcasted address selects byte locations in the System Address Space, in the Alternate Address Space or in the I/O Address Space, as defined in Paragraph 2.5.1.2, Table 2-9.

OBSERVATION 2.7:
In addition to the three addressing modes described here, there is a fourth mode which is used to select INTERRUPT-ACKNOWLEDGE and ARBITRATION cycles. They can be distinguished from other VSB cycles by the levels of SPACE0-SPACE1, as described in Paragraph 2.5.4.2, Table 2-18.

2.4.1.2 *Possibilité UNIQUEMENT D'ADRESSAGE*

Le cycle UNIQUEMENT D'ADRESSAGE est le seul cycle sur le bus de transfert de données qui n'est pas utilisé pour transférer des données. Le MAITRE actif commence le cycle UNIQUEMENT D'ADRESSAGE en exécutant la phase de diffusion d'adresse. Après que sa diffusion d'adresse a été acquittée correctement, le MAITRE termine le cycle sans exécuter aucun transfert de données (voir figure 2-5).

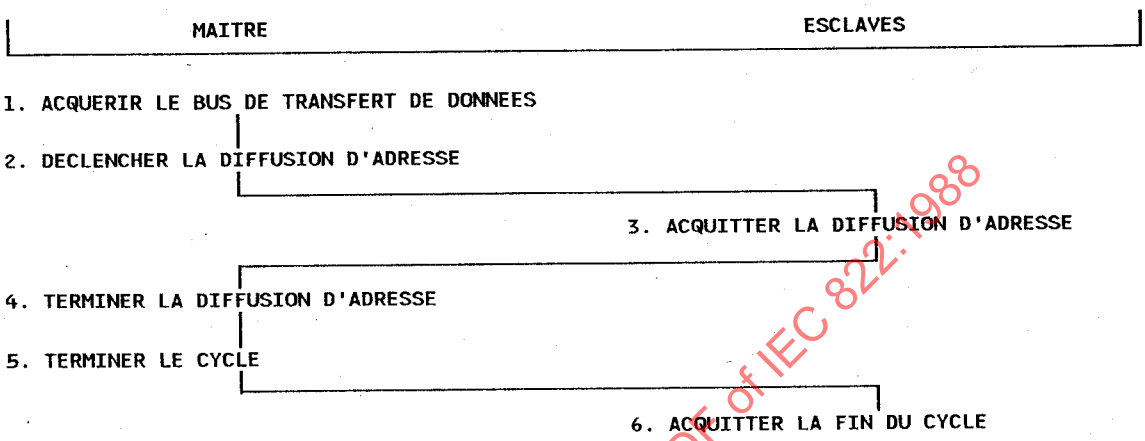


Fig. 2-5. - Organigramme général d'un cycle UNIQUEMENT D'ADRESSAGE.

Le tableau 2-4 montre comment le mnémonique AD0 correspondant à la possibilité UNIQUEMENT D'ADRESSAGE est utilisé pour décrire les MAITRES.

Tableau 2-4

Mnémonique qui spécifie la possibilité UNIQUEMENT D'ADRESSAGE

Le mnémonique suivant	Lorsqu'il est appliqué à un	Signifie qu'il
AD0	MAITRE	Peut générer des cycles UNIQUEMENT D'ADRESSAGE

OBSERVATION 2.8:
Les cycles UNIQUEMENT D'ADRESSAGE peuvent être utilisés pour accroître les performances en permettant à une carte CPU de diffuser une adresse avant qu'elle ait déterminé si cette adresse sélectionne un dispositif sur la carte ou un dispositif sur le bus VSB. Cette méthode permet à la carte CPU et aux ESCLAVES VSB de décoder l'adresse en même temps.

REGLE 2.3:
Les ESCLAVES VSB DOIVENT pouvoir accepter des cycles UNIQUEMENT D'ADRESSAGE sans perte de données ou mauvais fonctionnement.

2.4.1.2 *ADDRESS-ONLY capability*

The ADDRESS-ONLY cycle is the only cycle on the Data Transfer Bus that is not used to transfer data. The active MASTER starts the ADDRESS-ONLY cycle by executing the address broadcast phase. After its address broadcast is acknowledged in the proper manner, the MASTER terminates the cycle without executing any data transfers (see Figure 2-5).

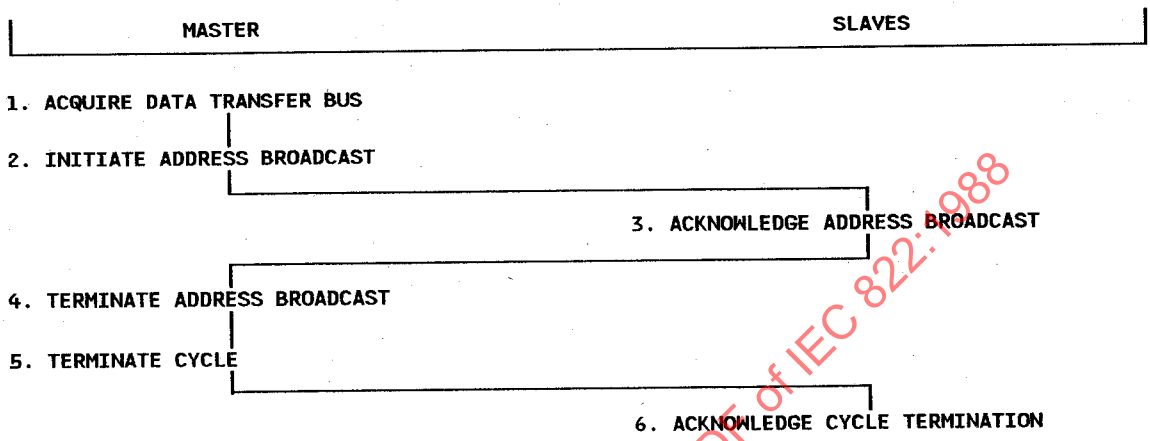


Fig. 2-5. - General flow of an ADDRESS-ONLY cycle.

Table 2-4 shows how the ADDRESS-ONLY mnemonic ADO is used to describe MASTERS.

Table 2-4

Mnemonic that specifies ADDRESS-ONLY capability

The following mnemonic	When applied to a	Means that it
ADO	MASTER	Can generate ADDRESS-ONLY cycles

OBSERVATION 2.8:
ADDRESS-ONLY cycles can be used to enhance performance by allowing a CPU board to broadcast an address before it has determined whether that address selects an on-board device or a VSB device. Broadcasting the address in this fashion allows the CPU board and VSB SLAVES to decode the address at the same time.

RULE 2.3:
VSB SLAVES MUST be able to tolerate ADDRESS-ONLY cycles without loss of data or incorrect operation.

2.4.2 *Possibilités de transfert de données*

Pendant les cycles de transfert de données VSB, le MAITRE actif dirige le transfert de données de ou vers les emplacements d'octet fournis par les ESCLAVES VSB.

Une notation spéciale est utilisée pour décrire les données émises par la logique de la carte (par exemple un microprocesseur). Ces données peuvent avoir une largeur de 8, 16, 24 ou 32 bits. Chaque groupe de 8 bits de donnée est référencé comme un octet de donnée.

Une donnée quadruple octet est une entité d'information qui contient quatre octets de donnée appelés DONNEE(0), DONNEE(1), DONNEE(2) et DONNEE(3).

Une donnée triple octet est une entité d'information qui contient trois octets de donnée appelés DONNEE(1), DONNEE(2) et DONNEE(3).

Une donnée double octet est une entité d'information qui contient deux octets de donnée appelés DONNEE(2) et DONNEE(3).

Une donnée octet unique est une entité d'information qui contient un seul octet de donnée appelé DONNEE(3).

L'organisation des quatre dimensions d'entités d'information est montrée dans la figure 2-6.

Donnée quadruple octet	DONNEE(0)	DONNEE(1)	DONNEE(2)	DONNEE(3)
Donnée triple octet		DONNEE(1)	DONNEE(2)	DONNEE(3)
Donnée double octet			DONNEE(2)	DONNEE(3)
Donnée octet unique				DONNEE(3)

Fig. 2-6. - Organisation des données.

OBSERVATION 2.9:
La relation d'information entre les octets d'une entité de donnée dépasse le cadre de cette norme. Divers processeurs peuvent organiser leurs données de façons différentes. Lorsque des systèmes sont constitués de processeurs différents, la responsabilité d'un échange correct de données entre ces processeurs devra être à la charge du logiciel du système.

2.4.2.1 *Possibilité de base de transfert de données des MAITRES*

La norme VSB définit une seule possibilité de base de transfert de données pour les MAITRES.

REGLE 2.4:
Tous les MAITRES VSB DOIVENT commander et surveiller les lignes de données AD00-AD31.

2.4.2 *Data transfer capabilities*

During VSB data transfer cycles, the active MASTER directs the transfer of data to and from byte locations provided by VSB SLAVES.

A special notation is used to describe the data that on-board logic (e.g. a microprocessor) outputs. This data might be either 8, 16, 24, or 32 bits wide. Each 8-bits of the data is referred to as a data byte.

Quad-Byte data is a data item that contains four bytes of data called DATA(0), DATA(1), DATA(2) and DATA(3).

Triple-Byte data is a data item that contains three bytes of data called DATA(1), DATA(2) and DATA(3).

Double-Byte data is a data item that contains two bytes of data called DATA(2) and DATA(3).

Single-Byte data is a data item that contains only one byte of data called DATA(3).

The organization of the four sizes of data items are shown in Figure 2-6.

Quad-Byte data	DATA(0)	DATA(1)	DATA(2)	DATA(3)
Triple-Byte data		DATA(1)	DATA(2)	DATA(3)
Double-Byte data			DATA(2)	DATA(3)
Single-Byte data				DATA(3)

Fig. 2-6. - Organization of data.

OBSERVATION 2.9:
The relationship of the information among the bytes of a data item is beyond the scope of this standard. Different processors might organize information in different ways. When systems are assembled using a mixture of processors, the responsibility for correct data interchange among these processors should be borne by system software.

2.4.2.1 *Basic data transfer capability of MASTERS*

The VSB standard defines only one basic data transfer capability for MASTERS.

RULE 2.4:
All VSB MASTERS MUST drive and monitor data lines AD00-AD31.

OBSERVATION 2.10:
L'exigence de commander toutes les lignes de données permet aux MAITRES VSB de générer les types de cycles suivants:

- Cycle octet unique aux emplacements d'octet: OCTET(0), OCTET(1), OCTET(2) et OCTET(3);
- Cycle double octet aux emplacements d'octet: OCTET(0-1), OCTET(1-2), OCTET(2-3) et OCTET(3)(0);
- Cycle triple octet aux emplacements d'octet: OCTET(0-2), OCTET(1-3), OCTET(2-3)(0) et OCTET(3)(0-1);
- Cycle quadruple octet aux emplacements d'octet: OCTET(0-3), OCTET(1-3)(0), OCTET(2-3)(0-1) et OCTET(3)(0-2).

2.4.2.2 *Possibilités de base de transferts de données des ESCLAVES*

La norme VSB définit trois possibilités de base de transferts de données pour les ESCLAVES. Le tableau 2-5 montre comment les mnémoniques D08, D16 et D32 sont utilisés pour décrire les possibilités de base de transferts de données des ESCLAVES.

Tableau 2-5

Mnémoniques qui spécifient les possibilités de base de transferts de données des ESCLAVES

Le mnémonique suivant	Lorsqu'il est appliqué à un	Signifie qu'il peut accomplir les cycles suivants dans un transfert unique de données
D08	ESCLAVE	Transferts de données octet unique à: OCTET(0) OCTET(1) OCTET(2) OCTET(3)
D16	ESCLAVE	Transferts de données octet unique à: OCTET(0) OCTET(1) OCTET(2) OCTET(3) et transferts de données double octet à: OCTET(0-1) OCTET(2-3)
D32	ESCLAVE	Transferts de données octet unique à: OCTET(0) OCTET(1) OCTET(2) OCTET(3) et transferts de données double octet à: OCTET(0-1) OCTET(1-2) OCTET(2-3) et transferts de données triple octet à: OCTET(0-2) OCTET(1-3) et transferts de données quadruple octet à: OCTET(0-3)

OBSERVATION 2.10:
The requirement to drive all the data lines allows VSB MASTERS to generate the following types of cycles:

- Single-Byte cycles to byte locations: BYTE(0), BYTE(1), BYTE(2) and BYTE(3);
- Double-Byte cycles to byte locations: BYTE(0-1), BYTE(1-2), BYTE(2-3) and BYTE(3)(0);
- Triple-Byte cycles to byte locations: BYTE(0-2), BYTE(1-3), BYTE(2-3)(0) and BYTE(3)(0-1);
- Quad-Byte cycles to byte locations: BYTE(0-3), BYTE(1-3)(0), BYTE(2-3)(0-1) and BYTE(3)(0-2).

2.4.2.2 *Basic data transfer capabilities of SLAVES*

The VSB standard defines three basic data transfer capabilities for SLAVES. Table 2-5 shows how the mnemonics D08, D16 and D32 are used to describe the basic data transfer capabilities of SLAVES.

Table 2-5

Mnemonics that specify the basic data transfer capabilities of SLAVES

The following mnemonic	When applied to a	Means that it can accommodate the following cycles in a single data transfer
D08	SLAVE	Single-Byte data transfers to: BYTE(0) BYTE(1) BYTE(2) BYTE(3)
D16	SLAVE	Single-Byte data transfers to: BYTE(0) BYTE(1) BYTE(2) BYTE(3) and Double-Byte data transfers to: BYTE(0-1) BYTE(2-3)
D32	SLAVE	Single-Byte data transfers to: BYTE(0) BYTE(1) BYTE(2) BYTE(3) and Double-Byte data transfers to: BYTE(0-1) BYTE(1-2) BYTE(2-3) and Triple-Byte data transfers to: BYTE(0-2) BYTE(1-3) and Quad-Byte data transfers to: BYTE(0-3)

2.4.2.3 *Dimensionnement dynamique du bus*

Le dimensionnement dynamique est une des caractéristiques principales du bus de transfert de données du VSB. Le dimensionnement dynamique du bus est la possibilité d'une logique locale d'ajuster automatiquement la taille du transfert de données et le nombre de cycles qui sont requis pour le transfert d'une entité d'information aux possibilités de base de transferts de données de la carte qui répond. La possibilité d'effectuer le dimensionnement dynamique permet aux programmeurs de ranger des données sans avoir à se préoccuper de l'organisation de la mémoire du système sur lequel le logiciel opère. Le système matériel lui-même a la charge d'ajuster la taille du transfert aux contraintes physiques du système. Le protocole VSB supporte le dimensionnement dynamique du bus de la façon suivante:

L'ESCLAVE répondant acquitte la phase de diffusion d'adresse en s'identifiant comme un ESCLAVE du type D08, D16 ou D32 sur les lignes ASACK0*-ASACK1* (voir paragraphe 2.5.1.2, tableau 2-13). Cela permet au MAITRE actif de déterminer quelles sont les lignes de données commandées avec des données valides pendant la phase du transfert de données des cycles de lecture (voir paragraphe 2.5.2.3, tableaux 2-15, 2-16 et 2-17). En plus, l'information de dimension de l'ESCLAVE est rendue disponible à la logique locale, lui permettant ainsi d'effectuer des cycles supplémentaires jusqu'à ce qu'on ait accédé à tous les emplacements d'octet requis.

Pendant certains cycles d'écriture, le MAITRE actif positionne plusieurs lignes de données avec les mêmes octets pour s'adapter aux trois types de dimension d'ESCLAVE permis. Cette opération est parfois désignée duplication de données. Lorsqu'un MAITRE actif déclenche un cycle d'écriture, il ne peut pas savoir si l'ESCLAVE qui va répondre au cycle est un ESCLAVE de type D08, D16 ou D32. Dans ce cas, le MAITRE actif place les mêmes octets sur plusieurs lignes de données pour être compatible avec les différents types d'ESCLAVES (voir paragraphe 2.5.2.3, tableau 2-14). Cela assure que des données seront transférées pendant le premier transfert même si plusieurs transferts sont nécessaires pour accomplir le cycle.

Par exemple, on peut avoir réalisé une carte processeur de telle sorte que les données double octet pour un cycle d'écriture OCTET(1-2) soient placées uniquement sur les lignes AD16-AD23. Cependant, un ESCLAVE D08 surveille seulement les lignes AD24-AD31 ce qui l'empêche de prélever toute donnée pendant le premier transfert de données. Par conséquent, pendant l'exécution d'un cycle d'écriture OCTET(1-2), l'information DONNEE(2) est placée sur AD24-AD31 de même que sur AD16-AD23, DONNEE(3) est placée uniquement sur AD08-AD15. Donc, si un ESCLAVE D32 répond, l'accès à tous les emplacements d'octet se fera en un transfert unique de données. Que ce soit un ESCLAVE D08 ou D16 qui réponde, la duplication de DONNEE(2) assure que celle-ci pourra être lue pendant le premier transfert de données. Cela permet aux microprocesseurs qui supportent le dimensionnement dynamique du bus d'accomplir toujours le transfert de données de DONNEE(3) en un seul cycle supplémentaire.

2.4.2.3 *Dynamic bus sizing*

Central to the data transfer capabilities of the VSB is the support of dynamic bus sizing. Dynamic bus sizing describes the ability of on-board logic to automatically adjust the size of the data transfer and the number of cycles that are required to transfer a data item, to the basic data transfer capabilities of the responding board. The ability to perform dynamic bus sizing allows programmers to store information without regard for the memory organization of the system on which the software is run. The system hardware itself is then given the task of adjusting the size of the transfer to the physical constraints of the system. The VSB protocols support dynamic bus sizing as follows:

The responding SLAVE acknowledges the address broadcast phase by identifying itself as a D08, D16 or a D32 SLAVE on ASACK0*-ASACK1* (see Paragraph 2.5.1.2, Table 2-13). This allows the active MASTER to determine which data lines are driven with valid data during the data transfer phase of read cycles (see Paragraph 2.5.2.3, Tables 2-15, 2-16 and 2-17). In addition, the SLAVE size information is made available to on-board logic, allowing it to perform additional cycles until all the required byte locations have been accessed.

During some write cycles, the active MASTER drives multiple data lines with the same data bytes to accommodate the three allowed SLAVE sizes. This operation is sometimes referred to as data replication. When the active MASTER initiates a write cycle, it has no way of knowing whether the SLAVE that will respond to the cycle is a D08, D16 or a D32 SLAVE. When appropriate, the active MASTER places the same data bytes on multiple data lines to accommodate the various types of SLAVES (see Paragraph 2.5.2.3, Table 2-14). This ensures that some data will be transferred during the first data transfer even if multiple transfers are required to complete the cycle.

For example, one could have designed a processor board so that Double-Byte data for a BYTE(1-2) write cycle were only placed on AD16-AD23. However, a D08 SLAVE only monitors data lines AD24-AD31, preventing it from capturing any data during the first data transfer. Therefore, when executing a BYTE(1-2) write cycle, DATA(2) is placed on AD24-AD31 as well as on AD16-AD23. DATA(3) is placed on AD08-AD15 only. Therefore, if a D32 SLAVE is responding, all the required byte locations will be accessed in the course of a single data transfer. The multiple placement of DATA(2) ensures that if either a D08 or a D16 SLAVE is responding, they can both read DATA(2) during the first data transfer. This allows microprocessors that support dynamic bus sizing to always complete the data transfer of DATA(3) in only one additional cycle.

2.4.2.4 *Possibilité de TRANSFERT UNIQUE*

Un cycle de TRANSFERT UNIQUE consiste en une diffusion d'adresse, suivie par un seul transfert de données. Il y a deux types de cycles de TRANSFERT UNIQUE:

a) Les cycles de TRANSFERT UNIQUE en écriture pendant lesquels le MAITRE range les données dans les emplacements d'octet des ESCLAVES.

b) Les cycles de TRANSFERT UNIQUE en lecture durant lesquels le MAITRE acquiert l'information à partir des emplacements d'octet des ESCLAVES.

REGLE 2.5:
Tous les ESCLAVES VSB DOIVENT inclure la possibilité de TRANSFERT UNIQUE.

La figure 2-7 montre l'organigramme général d'un cycle de TRANSFERT UNIQUE.

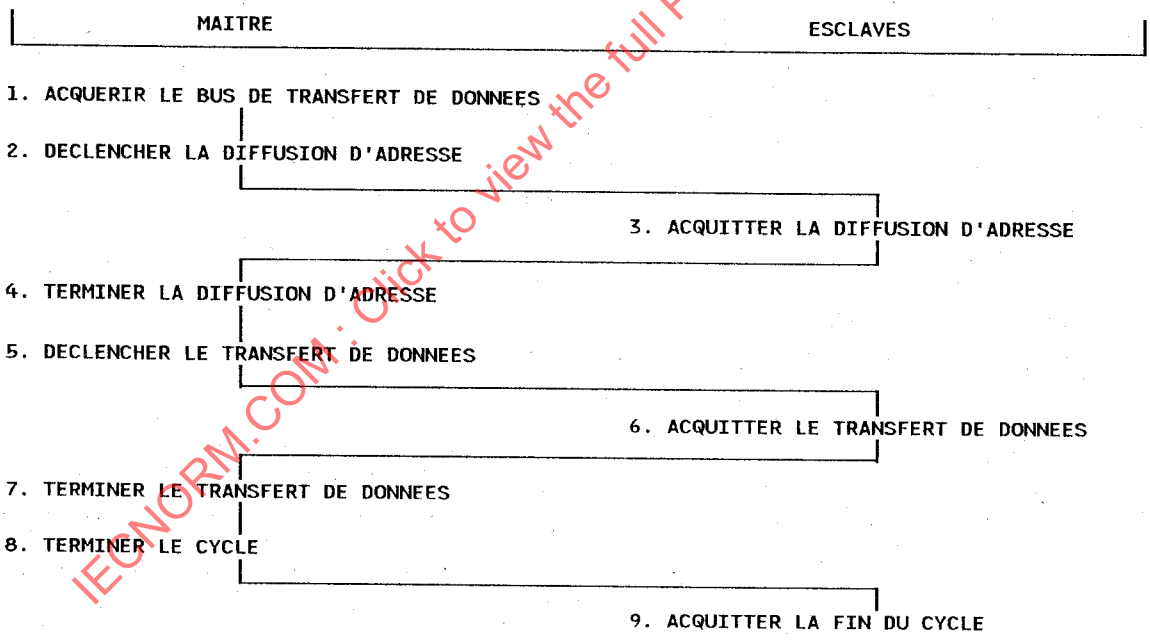


Fig. 2-7. - Organigramme général d'un cycle de TRANSFERT UNIQUE.

2.4.2.4 *SINGLE-TRANSFER capability*

A SINGLE-TRANSFER cycle consists of an address broadcast followed by one data transfer. There are two types of SINGLE-TRANSFER cycles:

a) SINGLE-TRANSFER Write Cycles, during which the MASTER stores data in byte locations provided by SLAVES.

b) SINGLE-TRANSFER Read Cycles, during which the MASTER retrieves data from byte locations provided by SLAVES.

RULE 2.5:
All VSB SLAVES MUST include SINGLE-TRANSFER capability.

Figure 2-7 shows the general flow of a SINGLE-TRANSFER cycle.

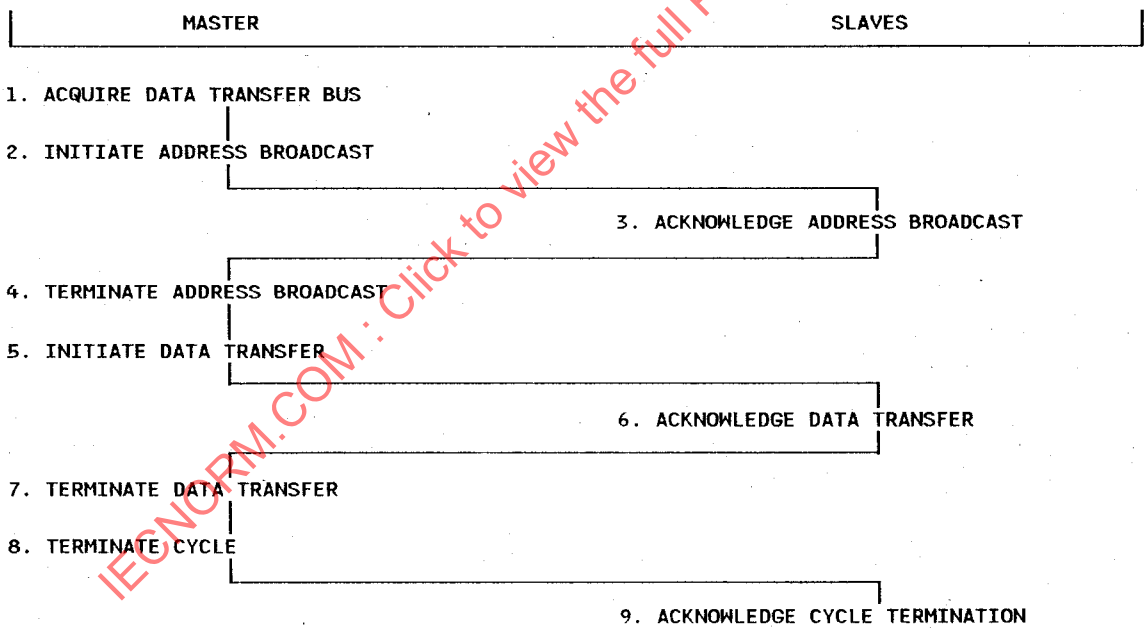


Fig. 2-7. - General flow of a SINGLE-TRANSFER cycle.

2.4.2.5 *Possibilité de TRANSFERT PAR BLOC*

Les MAITRES transfèrent souvent des blocs de données. C'est pourquoi les cycles de TRANSFERT PAR BLOC s'avèrent très utiles. Ils permettent au MAITRE d'émettre une seule adresse et de transférer un bloc de données sans avoir à émettre des adresses supplémentaires.

Il y a deux types principaux de cycles de TRANSFERT PAR BLOC:

a) Cycles de TRANSFERT PAR BLOC en écriture au cours desquels le MAITRE range des données dans des emplacement d'octet des ESCLAVES.

b) Cycles de TRANSFERT PAR BLOC en lecture au cours desquels le MAITRE acquiert les données mémorisées dans des emplacements d'octet des ESCLAVES.

Le cycle de TRANSFERT PAR BLOC en lecture est similaire à une suite de cycles de TRANSFERT UNIQUE en lecture. De même, le cycle de TRANSFERT PAR BLOC en écriture est similaire à une suite de cycles de TRANSFERT UNIQUE en écriture. La différence réside dans le fait que le MAITRE diffuse l'adresse initiale une seule fois au commencement du cycle et procède ensuite à l'exécution d'autant de transferts de données qu'il est nécessaire.

OBSERVATION 2.11:
Le contrôle du DTB ne peut être transféré pendant les cycles de TRANSFERT PAR BLOC parce que le signal de validation d'adresse est maintenu au niveau bas pendant tous les transferts de données et que le contrôle du DTB peut être transféré seulement lorsque le signal de validation d'adresse est au niveau haut.

Chacun des cycles en question peut être l'un des suivants:

a) Cycle de TRANSFERT PAR BLOC quadruple octet, dans lequel chaque transfert de données accède à quatre octets de données dans les emplacements d'octet des ESCLAVES D32.

b) Cycle de TRANSFERT PAR BLOC double octet, dans lequel chaque transfert de données accède à deux emplacements de données des ESCLAVES D16 ou D32.

c) Cycle de TRANSFERT PAR BLOC octet unique, dans lequel chaque transfert de données accède à un emplacement de données des ESCLAVES D08, D16 ou D32.

Quand un MAITRE déclenche un cycle de TRANSFERT PAR BLOC, l'ESCLAVE répondant mémorise l'adresse dans un compteur d'adresses local (au moins la partie moins significative de l'adresse) et acquitte la diffusion d'adresse. Le MAITRE commence alors le transfert de données. Lorsque l'ESCLAVE répondant acquitte le transfert de données, le MAITRE le termine (c'est-à-dire qu'il commande la ligne de validation de données au niveau haut). Cependant, le MAITRE ne permet pas à la ligne de validation d'adresse de remonter au niveau haut. Par contre, il commande de façon répétitive la ligne de validation de données au niveau bas en interverrouillage avec les acquittements de transferts de données de l'ESCLAVE répondant. L'ESCLAVE répondant utilise son compteur d'adresses local pour générer l'adresse de chaque transfert de données. La figure 2-8, page 76, montre l'organigramme général d'un cycle de TRANSFERT PAR BLOC.

2.4.2.5 *BLOCK-TRANSFER capability*

MASTERS often transfer blocks of data. When this is the case, BLOCK-TRANSFER cycles are very useful. They allow the MASTER to provide a single address, and then transfer a block of data without providing additional addresses.

There are two primary types of BLOCK-TRANSFER cycles:

a) BLOCK-TRANSFER Write Cycles, during which the MASTER stores data in byte locations provided by SLAVES.

b) BLOCK-TRANSFER Read Cycles, during which the MASTER retrieves data that was stored in byte locations provided by SLAVES.

The BLOCK-TRANSFER read cycle is similar to a string of SINGLE-TRANSFER read cycles. Likewise, the BLOCK-TRANSFER write cycle is similar to a string of SINGLE-TRANSFER write cycles. The difference is that the MASTER only broadcasts the initial address once, at the beginning of the cycle, and then proceeds to execute as many data transfers as are required.

OBSERVATION 2.11:
Control of the DTB cannot be transferred during BLOCK-TRANSFER cycles because the address strobe is held low through all of the data transfers, and control of the DTB can only be transferred while the address strobe is high.

Each of these cycles can be one of the following:

a) Quad-byte BLOCK-TRANSFER cycles, in which each data transfer accesses four bytes of data in byte locations provided by D32 SLAVES.

b) Double-byte BLOCK-TRANSFER cycles, in which each data transfer accesses two bytes of data in byte locations provided by D16 or D32 SLAVES.

c) Single-Byte BLOCK-TRANSFER cycles, in which each data transfer accesses one byte of data in byte locations provided by D08, D16 or D32 SLAVES.

When a MASTER initiates a BLOCK-TRANSFER cycle, the responding SLAVE latches (at least the low order part of) the address into an on-board address counter and acknowledges the address broadcast. The MASTER then starts the data transfer. After the responding SLAVE acknowledges the data transfer the MASTER terminates it (i.e. it drives the data strobe high). However, the MASTER does not allow the address strobe to go high. Instead, it repeatedly drives the data strobe low, interlocked with data transfer acknowledgments from the responding SLAVE. The responding SLAVE uses its on-board address counter to generate the address for each data transfer. Figure 2-8, page 77, shows the general flow of a BLOCK-TRANSFER cycle.

Le VSB ne limite pas le nombre d'octets qu'un MAITRE peut transférer pendant un cycle de TRANSFERT PAR BLOC. Cependant, pour simplifier la réalisation des ESCLAVES effectuant des TRANSFERTS PAR BLOC et pour éviter des pertes de données, un cycle de TRANSFERT PAR BLOC ne peut dépasser les frontières d'une carte. Par conséquent, les ESCLAVES effectuant des TRANSFERTS PAR BLOC sont tenus de signaler au MAITRE actif lorsqu'ils ne peuvent plus accomplir de transfert sans recevoir une autre adresse. Par exemple, un ESCLAVE effectuant des TRANSFERTS PAR BLOC et mémorisant seulement les huit lignes d'adresse les moins significatives (demandant seulement un compteur 8 bits) devra signaler la limite de son propre domaine d'adressage lorsqu'une frontière de 256 octets est atteinte. Par contre, un compteur d'adresses 32 bits d'un ESCLAVE effectuant des TRANSFERTS PAR BLOC n'impose pas de limites à la dimension des transferts. Cependant, un tel ESCLAVE devra encore signaler la fin de son propre domaine d'adressage après avoir atteint le dernier emplacement d'octet de la carte.

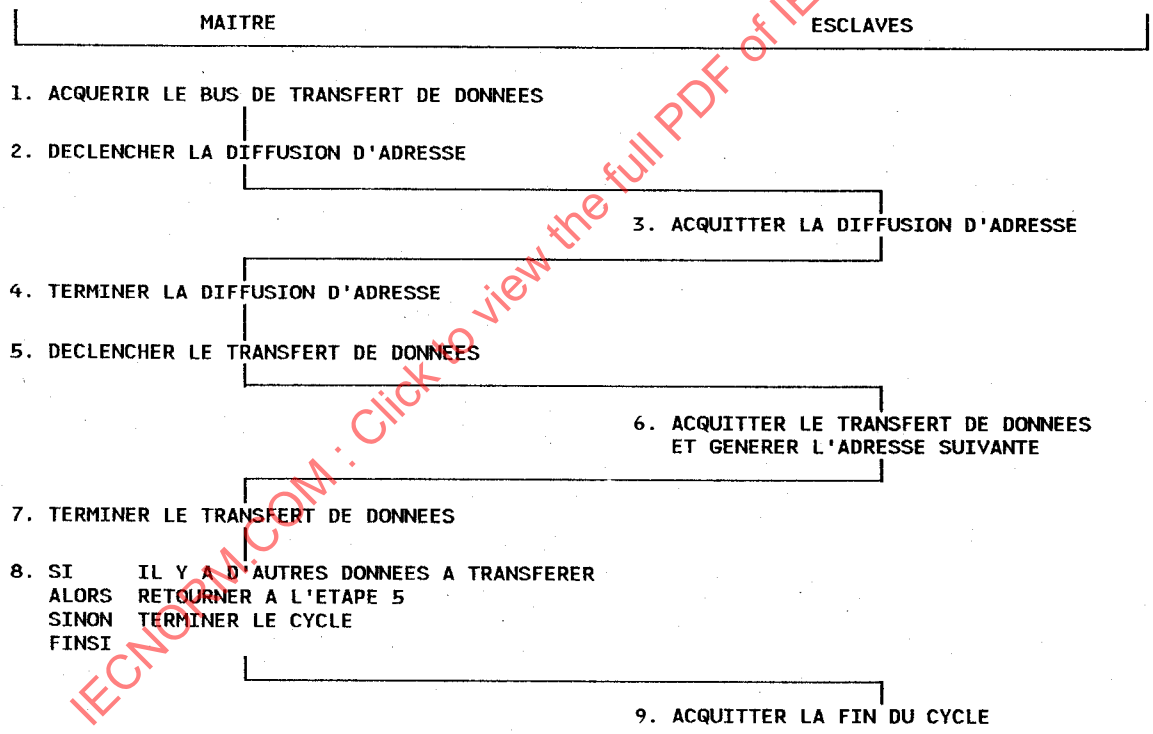


Fig. 2-8. - Organigramme général d'un cycle de TRANSFERT PAR BLOC.

The VSB does not limit the number of bytes that a MASTER can transfer during a BLOCK-TRANSFER cycle. However, to simplify the design of BLOCK-TRANSFER SLAVES and to protect against loss of data, a BLOCK-TRANSFER cycle cannot cross a board boundary. Therefore, BLOCK-TRANSFER SLAVES are required to signal the active MASTER when they can no longer perform data transfers without receiving another address. For example, a BLOCK-TRANSFER SLAVE that only latches the lower eight address lines (requiring only an 8-bit counter) will signal the end of its self addressing range when a 256-byte boundary is crossed. On the other hand, a 32-bit address counter on a BLOCK-TRANSFER SLAVE imposes no limit on the size of the transfer. However, such a SLAVE is still required to signal the end of its self addressing range after it has accessed the last byte location it provides on-board.

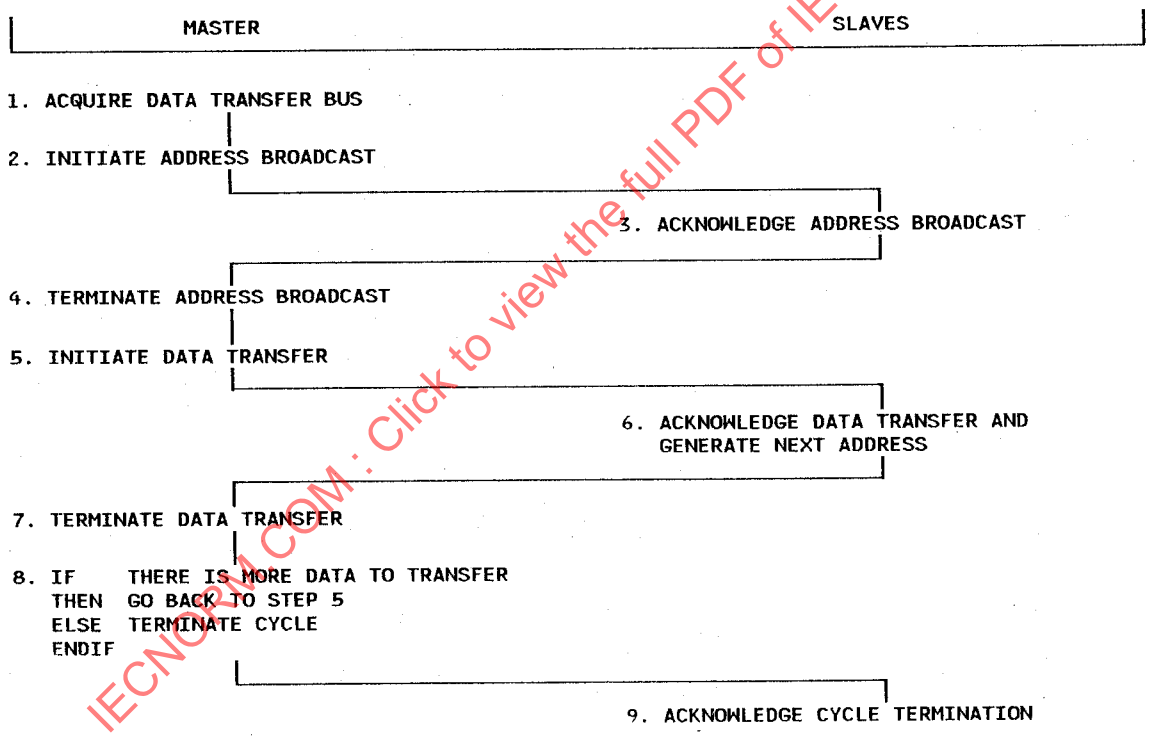


Fig. 2-8. - General flow of a BLOCK-TRANSFER cycle.

Le tableau 2-6 montre le mnémonique utilisé pour décrire la possibilité de TRANSFERT PAR BLOC des MAITRES et des ESCLAVES.

Tableau 2-6

Mnémonique qui spécifie la possibilité de TRANSFERT PAR BLOC

Le mnémonique suivant	Lorsqu'il s'applique à un	Signifie qu'il
BLT	MAITRE	Peut générer des cycles de TRANSFERT PAR BLOC d'octets uniques, et des cycles de TRANSFERT PAR BLOC de doubles octets, et des cycles de TRANSFERT PAR BLOC de quadruples octets
	ESCLAVE D08	Répond à tous les cycles de TRANSFERT PAR BLOC en exécutant des transferts de données par octet unique
	ESCLAVE D16	Répond aux cycles de TRANSFERT PAR BLOC d'octets uniques en exécutant des transferts de données par octet unique. Répond aux cycles de TRANSFERT PAR BLOC de doubles octets et de quadruples octets en exécutant des transferts de données par double octet
	ESCLAVE D32	Répond aux cycles de TRANSFERT PAR BLOC d'octets uniques en exécutant des transferts de données par octet unique. Répond aux cycles de TRANSFERT PAR BLOC de doubles octets en exécutant des transferts de données par double octet. Répond aux cycles de TRANSFERT PAR BLOC de quadruples octets en exécutant des transferts de données par quadruple octet

2.4.2.6 *Possibilité de TRANSFERT INDIVISIBLE*

De nombreux processeurs modernes ont la possibilité d'effectuer des séquences indivisibles, c'est-à-dire des séquences d'accès mémoire consécutifs, sans permettre à un autre processeur ou un autre MAITRE d'avoir accès aux mêmes emplacements d'octet avant que la séquence ne soit achevée. De tels processeurs différencient les séquences indivisibles des autres cycles normaux du bus, de telle sorte que la logique externe puisse garantir l'indivisibilité exigée.

La ligne LOCK* du VSB peut être utilisée pour signaler une telle indivisibilité, mais elle doit l'être avec précaution pour éviter les situations de blocage du système. Ce paragraphe fournit une description de deux types d'accès indivisibles et fournit des suggestions pour éviter les blocages.

Table 2-6 lists the mnemonic that is used to describe the BLOCK-TRANSFER capability of MASTERS and SLAVES.

Table 2-6

Mnemonic that specifies BLOCK-TRANSFER capability

The following mnemonic	When applied to a	Means that it
BLT	MASTER	Can generate Single-Byte BLOCK-TRANSFER cycles, and Double-Byte BLOCK-TRANSFER cycles, and Quad-Byte BLOCK TRANSFER cycles
	D08 SLAVE	Responds to all BLOCK-TRANSFER cycles by executing Single-Byte data transfers
	D16 SLAVE	Responds to Single-Byte BLOCK-TRANSFER cycles by executing Single-Byte data transfers. Responds to Double-Byte and Quad-Byte BLOCK-TRANSFER cycles by executing Double-Byte data transfers
	D32 SLAVE	Responds to Single-Byte BLOCK-TRANSFER cycles by executing Single-Byte data transfers. Responds to Double-Byte BLOCK-TRANSFER cycles by executing Double-Byte data transfers. Responds to Quad-Byte BLOCK-TRANSFER cycles by executing Quad-Byte data transfers

2.4.2.6 *INDIVISIBLE-ACCESS capability*

Many modern processors can perform indivisible sequences, that is, sequences of memory accesses that have to proceed one after the other, without allowing any other processor or MASTER to access any of the same byte locations before the sequence is complete. Such processors signal indivisible sequences differently from normal bus cycles, so that external hardware can guarantee the needed indivisibility.

The LOCK* line in the VSB can be used to signal such indivisibility, but it has to be used with caution to avoid system deadlocks. This paragraph provides a description of two types of indivisible accesses and offers suggestions as to how to avoid system deadlocks.

Il y a deux classes de séquences indivisibles:

a) Une *séquence indivisible intradomaine* (INTRASEQ) est une séquence dans laquelle l'accès à tous les emplacements d'octet impliqués dans la séquence est contrôlé par le même mécanisme d'arbitrage, c'est-à-dire que l'accès à tous ces emplacements est alloué simultanément. Un exemple d'opération INTRASEQ est l'instruction "Test and Set" au cours de laquelle on accède à un emplacement d'octet unique deux fois en séquence. Un autre exemple est celui où tous les emplacements d'octet auxquels on a accédé lors de la séquence sont localisés sur la même carte.

b) Une *séquence indivisible interdomaine* (INTERSEQ) est une séquence dans laquelle l'accès aux emplacements d'octet impliqués dans la séquence est contrôlé par deux ou plusieurs mécanismes d'arbitrage. Dans ce cas, deux ou plusieurs processeurs peuvent avoir chacun l'accord pour accéder à certains des emplacements d'octet impliqués mais pas à tous. Un exemple d'opération INTERSEQ est une séquence qui met en jeu l'accès à un emplacement d'octet sur le bus VSB ainsi qu'à un emplacement d'octet sur le bus global du système.

L'utilisation de la ligne LOCK* pour garantir l'indivisibilité est suffisante dans les opérations INTRASEQ, mais non pour les opérations INTERSEQ. Cela est bien démontré par un exemple:

Deux cartes CPU, CPU A et CPU B ont chacune une mémoire double accès autorisant l'accès local et par le bus VSB. L'accès à la mémoire du CPU A se fait à partir de l'adresse 100000 alors que celui à la mémoire du CPU B se fait à partir de l'adresse 200000.

Le CPU A commence une opération INTERSEQ qui met en jeu sa mémoire locale à l'adresse 100000 (à laquelle il accède d'abord) et la mémoire globale à l'adresse 200000 (à laquelle il accède ensuite).

Au même instant, le CPU B commence une opération INTERSEQ qui met en jeu sa mémoire locale à l'adresse 200000 (à laquelle il accède d'abord) puis la mémoire globale à l'adresse 100000 (à laquelle il accède ensuite).

Dans cet exemple, sans un mécanisme de contrôle pour les opérations INTERSEQ, chaque CPU accèdera d'abord aux emplacements d'octet de sa propre mémoire locale puis demandera le contrôle du bus VSB pour accéder au deuxième emplacement d'octet en mémoire globale. On suppose que le CPU A a obtenu l'allocation du bus VSB d'abord. Il prend le contrôle du bus VSB et cherche à accéder à l'emplacement 200000 dans la mémoire du CPU B. Cependant, cet accès ne peut pas lui être accordé puisque le CPU B est au milieu de son opération INTERSEQ; celle-ci ne peut pas être achevée tant qu'il n'a pas l'allocation du bus VSB. Cette allocation ne pourra pas lui être accordée tant que le CPU A n'aura pas terminé son opération INTERSEQ. Le système est dans une situation de blocage.

Une façon d'éviter une telle situation de blocage est d'interdire au logiciel du système d'exécuter des opérations du type INTERSEQ. Cela peut se faire en plaçant en mémoire globale toutes les variables auxquelles on peut accéder de façon indivisible, c'est-à-dire, une mémoire à laquelle on accède seulement à partir du bus global du système. Imposer une telle contrainte signifie que tout le logiciel y adhère.

There are two classes of indivisible sequences:

a) An *Intradomain Indivisible Sequence* (INTRASEQ) is one in which access to all the byte locations involved in the sequence is controlled by the same arbitration mechanism - i.e. access to all of them is granted simultaneously. An example of an INTRASEQ is a Test and Set instruction, in which a single byte location is accessed twice in succession. Another example is when all of the byte locations which are accessed during the sequence are on the same board.

b) An *Interdomain Indivisible Sequence* (INTERSEQ) is one in which access to the byte locations involved in the sequence is controlled by two or more arbitration mechanisms. In such a case, two or more processors might each be granted access to some of the byte locations involved but not all of them. An example of an INTERSEQ is a sequence which involves access to a byte location on the VSB and also a byte location on the global system bus.

Use of the LOCK* line is sufficient to guarantee indivisibility for INTRASEQS but not for INTERSEQS. This is best demonstrated by an example:

Two CPU boards, CPU A and CPU B, each have dual-ported memory which allows access both locally and from the VSB. CPU A's memory is accessed starting at address 100000, while CPU B's memory is accessed starting at address 200000.

CPU A starts an INTERSEQ which involves its local memory at address 100000 (accessed first) and global memory at address 200000 (accessed second).

At the same time CPU B starts an INTERSEQ which involves its local memory at address 200000 (accessed first) and global memory at address 100000 (accessed second).

In this example, without a control mechanism for INTERSEQS, each CPU will first access the byte location in its respective local memory, and then request control of the VSB to access the second, global byte location. Suppose that CPU A is granted use of the VSB first. It assumes control of the VSB and attempts to access byte location 200000 in CPU B's memory. However, this access can not be granted since CPU B is in the middle of its own INTERSEQ, which it cannot complete until it is granted use of the VSB, which cannot be granted until CPU A completes its INTERSEQ. The system is in a deadlock.

One way to prevent deadlock in such a case is to constrain the system software from executing INTERSEQS. This can be achieved by placing all the variables that might ever be accessed in an indivisible manner in global memory, that is, memory that can only be accessed over the global system bus. Imposing such a constraint requires that all software adhere to it.

Cette stratégie est bonne pour une information à laquelle tous les processeurs d'un système accèdent plus ou moins également. Mais pour une information utilisée souvent par un processeur et occasionnellement par d'autres, placer cette information en mémoire globale accroît le trafic sur le bus et peut réduire les performances du système, comparé à la solution qui consiste à la placer dans la mémoire locale du processeur prédominant.

Une manière plus générale de résoudre les situations potentielles de blocage système est de définir un protocole d'exécution des opérations INTERSEQ. Les cartes CPU capables de générer des opérations INTERSEQ doivent demander et obtenir le contrôle d'une ressource centrale AVANT de commencer TOUTE PARTIE d'une séquence indivisible. Peu importe de quelle ressource centrale il s'agit, l'essentiel est que tous les processeurs respectent la même règle. Il est judicieux de choisir comme telle ressource le bus global du système. Ce choix est meilleur que celui du bus sous-système parce que les cartes processeurs sur différents bus sous-système peuvent avoir à exécuter des opérations INTERSEQ qui mettent en jeu des emplacements d'octet qui sont locaux sur une autre carte processeur.

2.4.3 *Possibilités d'interruption*

Le VSB définit un protocole qui permet aux ESCLAVES de demander un service d'interruption à un MAITRE qui répondra à ces requêtes. Tout système qui a des possibilités d'interruption comprend des logiciels appelés routines de service d'interruption. Le protocole d'interruption du VSB définit comment les ESCLAVES transfèrent l'information MOT D'ETAT/ID au MAITRE. Cette information est utilisée par la logique locale pour appeler une routine de service d'interruption.

OBSERVATION 2.12:
Le protocole VSB ne limite pas la quantité d'information du MOT D'ETAT/ID qui peut être transférée au cours d'un cycle unique.

2.4.3.1 *Possibilités d'interruption de base des MAITRES et des ESCLAVES*

Le VSB définit deux possibilités d'interruption de base pour les MAITRES et les ESCLAVES:

a) En réponse à une demande d'interruption, le MAITRE déclenche et les ESCLAVES participent à un cycle de RECONNAISSANCE D'INTERRUPTION. A chaque ESCLAVE est affecté un ID INTERRUPTION unique qui est utilisé pour régler les priorités entre plusieurs interruptions. Cet ID INTERRUPTION est un identificateur de 7 bits qui est une combinaison de l'adresse sur 3 bits de l'emplacement dans le châssis et d'un code de priorité sur 4 bits, défini par l'utilisateur. Il est utilisé durant la phase de sélection d'un cycle de RECONNAISSANCE D'INTERRUPTION pour sélectionner l'ESCLAVE dont la demande d'interruption sera servie par la suite. L'ESCLAVE transmet le MOT D'ETAT/ID au MAITRE et libère au niveau haut sa participation à la ligne IRQ*. Lorsqu'un cycle de RECONNAISSANCE D'INTERRUPTION est effectué, cette opération de lecture est appelée transfert de MOT D'ETAT/ID.

This strategy is fine for information that is accessed more or less equally by all the processors in a system. But, for information that is accessed frequently by one processor and only occasionally by others, placing the information in global memory increases bus traffic and can reduce system performance, compared to placing it in memory local to the predominant processor.

A more general solution to potential system deadlocks is to establish a protocol for executing INTERSEQS. CPU boards that are capable of generating INTERSEQS have to request and be granted control of an agreed upon central resource BEFORE executing ANY PART of an indivisible sequence. It does not much matter what this central resource is, just that all processor boards "play by the same rule". An obvious choice for such a central resource is the global system bus. It is a better choice than the sub-system bus because processor boards on different sub-system buses might need to perform INTERSEQS that involve byte locations that are local to another processor board.

2.4.3 *Interrupt capability*

The VSB defines a protocol that allows SLAVES to request interrupt service from a MASTER which, in turn, responds to these requests. Any system that has interrupt capability includes software routines that are called interrupt service routines. The interrupt protocol of the VSB defines how the SLAVE transfers STATUS/ID information to the MASTER. This information is used by on-board logic to invoke an interrupt service routine.

OBSERVATION 2.12:
The VSB protocols do not limit the amount of the STATUS/ID information that can be transferred in the course of a single cycle.

2.4.3.1 *Basic interrupt capabilities of MASTERS and SLAVES*

The VSB defines two basic interrupt capabilities for MASTERS and SLAVES:

a) In response to an interrupt request, the MASTER initiates and SLAVES participate in an INTERRUPT-ACKNOWLEDGE cycle. Each SLAVE is assigned a unique INTERRUPT ID which is used to prioritize multiple interrupts. This INTERRUPT ID is a 7-bit ID which is a combination of the 3-bit slot address and a 4-bit user defined priority code. It is used during the selection phase of an INTERRUPT-ACKNOWLEDGE cycle to select the SLAVE whose interrupt request will be serviced next. This SLAVE transfers the STATUS/ID information to the MASTER, and releases its contribution to the IRQ* line to high. When executing an INTERRUPT-ACKNOWLEDGE cycle, this read operation is referred to as a STATUS/ID transfer.

b) En réponse à une demande d'interruption, le MAITRE déclenche des cycles de lecture à des adresses prédéfinies pour déterminer quel est l'ESCLAVE qui a demandé une interruption. Ces ESCLAVES ont un registre MOT D'ETAT/ID et libèrent leur contribution à la ligne IRQ* au niveau haut pendant le cycle de lecture du registre MOT D'ETAT/ID. Le MAITRE peut alors procéder à des cycles de lecture additionnels pour lire des informations MOT D'ETAT/ID complémentaires de l'ESCLAVE. L'information MOT D'ETAT/ID que le MAITRE acquiert de l'ESCLAVE est transférée au cours d'un transfert de données en lecture, comme défini au paragraphe 2.5.2.2. Lorsqu'il répond à une demande d'interruption par interrogation, le MAITRE exécute soit un cycle de TRANSFERT UNIQUE en lecture soit un cycle de TRANSFERT PAR BLOC en lecture, selon ses possibilités et les possibilités de l'ESCLAVE répondant.

Le tableau 2-7 montre les mnémoniques utilisés pour décrire les possibilités d'interruption de base des MAITRES et des ESCLAVES.

Tableau 2-7

Mnémoniques qui spécifient les possibilités d'interruption

Le mnémonique suivant	Lorsqu'il s'applique à un	Signifie qu'il
IHV	MAITRE	Répond aux demandes d'interruption en lançant un cycle de RECONNAISSANCE D'INTERRUPTION
IHP	MAITRE	Répond aux demandes d'interruption en lançant un cycle de lecture pour déterminer leur origine
INTV	ESCLAVE	Peut générer des demandes d'interruption et participer aux cycles de RECONNAISSANCE D'INTERRUPTION
INTP	ESCLAVE	Peut générer des demandes d'interruption et se faire reconnaître lors d'un cycle de lecture

REGLE 2.6:
Les ESCLAVES INTV DOIVENT inclure la possibilité INTP.

b) In response to an interrupt request, the MASTER initiates read cycles to predefined addresses to determine which SLAVE has requested an interrupt. These SLAVES include a STATUS/ID register, and release their contribution to the IRQ* line to high during the read cycle that accesses their STATUS/ID register. The MASTER might then execute additional cycles to read additional STATUS/ID information from the SLAVE. The STATUS/ID information that the MASTER reads from the SLAVE is transferred in the course of a read data transfer, as defined in Paragraph 2.5.2.2. When responding to an interrupt request by polling, a MASTER executes either a SINGLE-TRANSFER or a BLOCK-TRANSFER read cycle, depending on its capabilities and on the capabilities of the responding SLAVE.

Table 2-7 shows the mnemonics that are used to describe the basic interrupt capabilities of MASTERS and SLAVES.

Table 2-7

Mnemonics that specify interrupt capabilities

The following mnemonic	When applied to a	Means that it
IHV	MASTER	Responds to interrupt requests by initiating an INTERRUPT-ACKNOWLEDGE cycle
IHP	MASTER	Responds to interrupt requests by initiating a read cycle to determine their cause
INTV	SLAVE	Can generate interrupt requests and participate in INTERRUPT-ACKNOWLEDGE cycles
INTP	SLAVE	Can generate interrupt requests and identify itself in the course of a read cycle

RULE 2.6:
INTV SLAVES MUST include INTP capability.

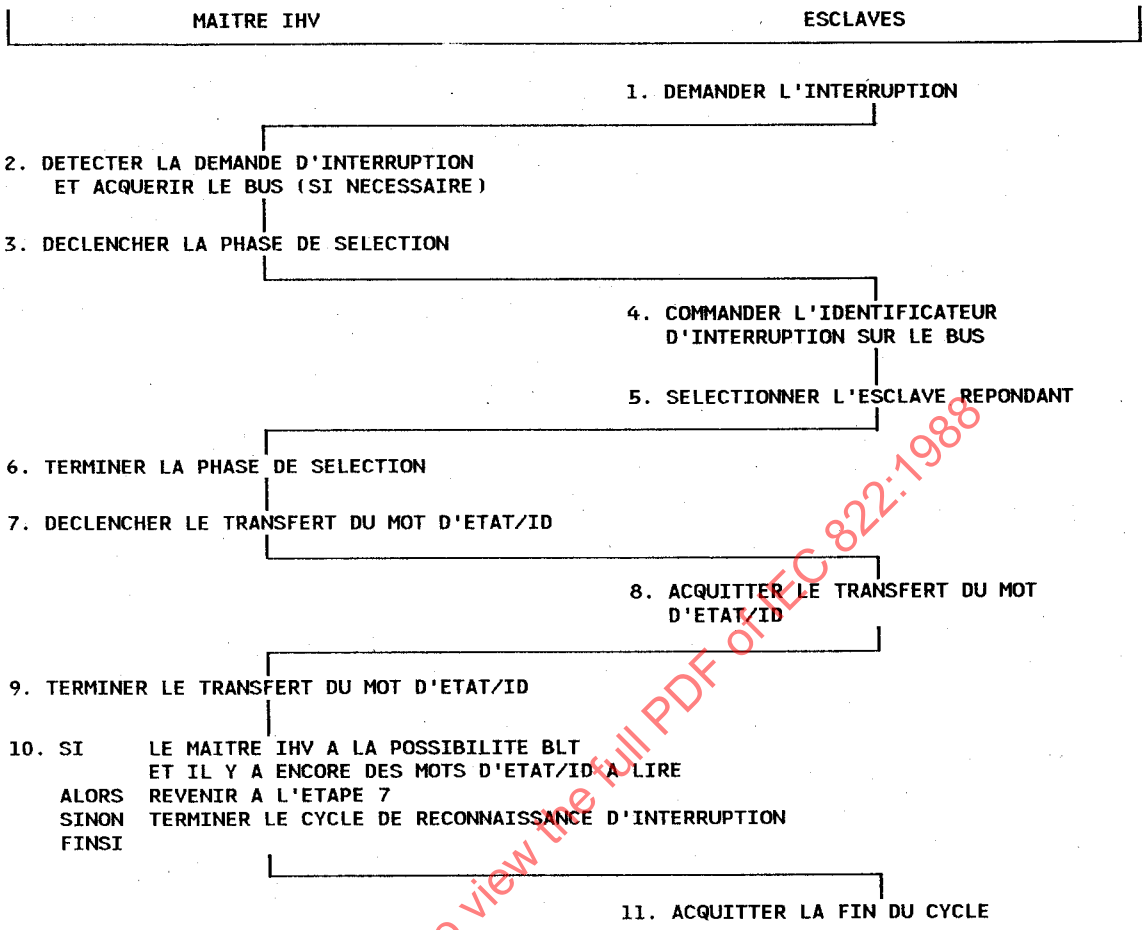


Fig. 2-9. - Organigramme général d'un cycle de RECONNAISSANCE D'INTERRUPTION.

2.4.3.2 *Possibilités de cycle de RECONNAISSANCE D'INTERRUPTION.*

Quand un MAITRE répond à une demande d'interruption, il lit l'information du MOT D'ETAT/ID de l'ESCLAVE. Comme décrit précédemment, le MAITRE lit ce MOT D'ETAT/ID au cours d'un cycle de RECONNAISSANCE D'INTERRUPTION ou par interrogation successive des ESCLAVES. L'organigramme général d'un cycle de RECONNAISSANCE D'INTERRUPTION est montré dans la figure 2-9. Le cycle commence lorsqu'un ESCLAVE demande un service d'interruption. Le MAITRE IHV qui gère les interruptions obtient le contrôle du bus et déclenche un cycle de RECONNAISSANCE D'INTERRUPTION constitué de trois phases distinctes:

a) La première phase est la phase de sélection. Elle commence lorsque le MAITRE IHV déclenche un cycle de RECONNAISSANCE D'INTERRUPTION et se termine après avoir déterminé quel est l'ESCLAVE INTV répondant (étapes 3-6 de la figure 2-9). L'interaction entre le MAITRE INTV et les ESCLAVES INTV pendant cette phase est décrite au paragraphe 2.5.4.1.

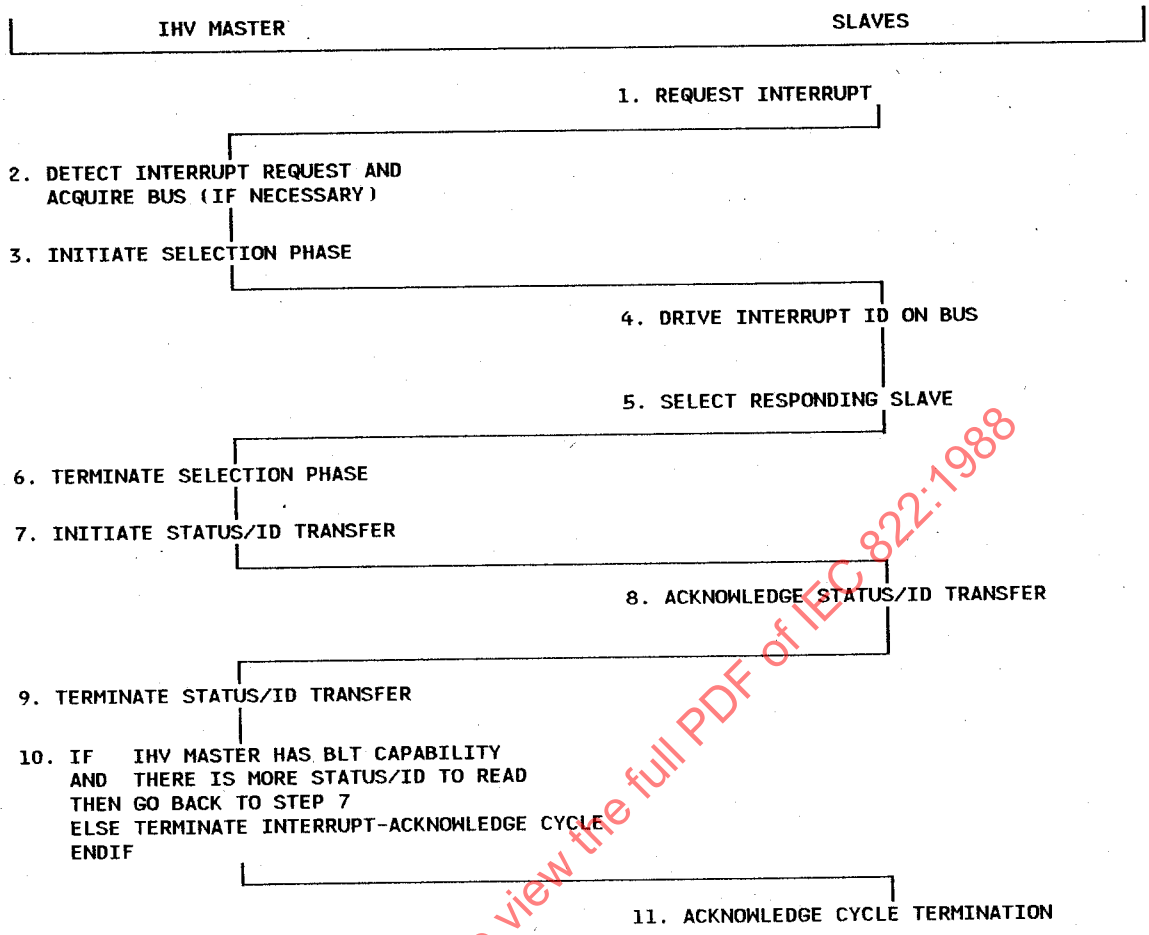


Fig. 2-9. - General flow of an INTERRUPT-ACKNOWLEDGE cycle.

2.4.3.2 *INTERRUPT-ACKNOWLEDGE cycle capabilities*

When a MASTER responds to an interrupt request, it reads STATUS/ID information from the SLAVE. As described above, the MASTER reads this STATUS/ID in the course of an INTERRUPT-ACKNOWLEDGE cycle or by polling the SLAVES. The general flow of the INTERRUPT-ACKNOWLEDGE cycle is shown in Figure 2-9. The cycle starts when a SLAVE requests an interrupt service. The IHV MASTER that services interrupts acquires the bus and initiates an INTERRUPT-ACKNOWLEDGE cycle which is comprised of three distinct phases:

a) The first phase is the selection phase. It starts when the IHV MASTER initiates an INTERRUPT-ACKNOWLEDGE cycle and ends after the responding INTV SLAVE is determined (steps 3-6 in Figure 2-9). The interaction between the INTV MASTER and INTV SLAVES during this phase is described in Paragraph 2.5.4.1.

b) La deuxième phase est la phase de transfert du MOT D'ETAT/ ID. Elle commence après que l'ESCLAVE INTV répondant a été sélectionné et se termine après que toute l'information du MOT D'ETAT/ID a été transférée (étapes 7-9 de la figure 2-9). L'interaction entre le MAITRE IHV et l'ESCLAVE INTV répondant pendant cette phase est décrite dans le paragraphe 2.5.4.1.

c) La troisième phase est la phase de fin du cycle de RECONNAISSANCE D'INTERRUPTION (étapes 9-11 de la figure 2-9), décrite au paragraphe 2.5.3.

Il y a trois types de cycles de RECONNAISSANCE D'INTERRUPTION:

a) Les cycles de RECONNAISSANCE D'INTERRUPTION quadruple octet dans lesquels chaque transfert du MOT D'ETAT/ID accède à quatre octets consécutifs de l'information du MOT D'ETAT/ID disponible.

b) Les cycles de RECONNAISSANCE D'INTERRUPTION double octet, dans lesquels chaque transfert de MOT D'ETAT/ID accède à deux octets consécutifs de l'information du MOT D'ETAT/ID disponible.

c) Les cycles de RECONNAISSANCE D'INTERRUPTION octet unique, dans lesquels chaque transfert du MOT D'ETAT/ID accède à un octet de l'information du MOT D'ETAT/ID disponible.

Le tableau 2-8 montre comment les mnémoniques D08, D16, D32 et BLT sont utilisés pour décrire les possibilités de transfert de l'information du MOT D'ETAT/ID des MAITRES IHV et des ESCLAVES INTV.

b) The second phase is the STATUS/ID transfer phase. It starts after the responding INTV SLAVE is selected, and ends after all of the STATUS/ID information is transferred (steps 7-9 in Figure 2-9). The interaction between the IHV MASTER and the responding INTV SLAVE during this phase is described in Paragraph 2.5.4.1.

c) The third phase is the termination phase of the INTERRUPT-ACKNOWLEDGE cycle (steps 9-11 in Figure 2-9), described in Paragraph 2.5.3.

There are three types of INTERRUPT-ACKNOWLEDGE cycles:

a) Quad-byte INTERRUPT-ACKNOWLEDGE cycles, in which each STATUS/ID transfer accesses four consecutive bytes of the available STATUS/ID information.

b) Double-byte INTERRUPT-ACKNOWLEDGE cycles, in which each STATUS/ID transfer accesses two consecutive bytes of the available STATUS/ID information.

c) Single-byte INTERRUPT-ACKNOWLEDGE cycles, in which each STATUS/ID transfer accesses one byte of the available STATUS/ID information.

Table 2-8 shows how the mnemonics D08, D16, D32 and BLT are used to describe the STATUS/ID transfer capabilities of IHV MASTERS and INTV SLAVES.

Tableau 2-8

Mnémoniques qui spécifient les possibilités de transfert de MOT D'ETAT/ID des MAITRES IHV et des ESCLAVES INTV

Le mnémonique suivant	Lorsqu'il s'applique à un	Signifie qu'il
D08	ESCLAVE INTV	Répond à toutes les demandes de MOT D'ETAT/ID en exécutant un transfert de MOT D'ETAT/ID octet unique
D16	ESCLAVE INTV	Répond aux demandes de transfert de MOT D'ETAT/ID octet unique en exécutant un transfert de MOT D'ETAT/ID octet unique. Répond aux demandes de transfert de MOT D'ETAT/ID double et quadruple octets en exécutant un transfert de MOT D'ETAT/ID double octet
D32	ESCLAVE INTV	Répond aux demandes de transfert de MOT D'ETAT/ID octet unique en exécutant un transfert de MOT D'ETAT/ID octet unique. Répond aux demandes de transfert de MOT D'ETAT/ID double octet en exécutant un transfert de MOT D'ETAT/ID double octet. Répond aux demandes de transfert de MOT D'ETAT/ID quadruple octet en exécutant un transfert de MOT D'ETAT/ID quadruple octet
BLT	MAITRE INTV	Peut demander des transferts de MOT D'ETAT/ID multiples au cours d'un cycle de RECONNAISSANCE D'INTERRUPTION
BLT	ESCLAVE INTV	Peut répondre correctement aux transferts de MOT D'ETAT/ID multiples au cours d'un cycle de RECONNAISSANCE D'INTERRUPTION

2.5 *Interaction entre les MAITRES et les ESCLAVES*

L'interaction entre les MAITRES et les ESCLAVES est décrite à l'aide d'organigrammes. Le paragraphe 2.5.1 décrit l'interaction entre les MAITRES et les ESCLAVES pendant la phase de diffusion d'adresse.

Le paragraphe 2.5.2 décrit l'interaction entre les MAITRES et les ESCLAVES pendant la phase de transfert de données.

Le paragraphe 2.5.3 décrit l'interaction entre les MAITRES et les ESCLAVES pendant la fin de cycle.

L'interaction entre les MAITRES et les ESCLAVES pendant les cycles de RECONNAISSANCE D'INTERRUPTION est décrite au paragraphe 2.5.4.

Table 2-8

Mnemonics that specify STATUS/ID transfer capabilities of IHV MASTERS and INTV SLAVES

The following mnemonic	When applied to a	Means that it
D08	INTV SLAVE	Responds to all STATUS/ID requests by executing a Single-Byte STATUS/ID transfer
D16	INTV SLAVE	Responds to Single-Byte Status/ID transfer requests by executing a Single-Byte STATUS/ID transfer. Responds to Double-Byte and Quad-Byte STATUS/ID transfer requests by executing a Double-Byte STATUS/ID transfer
D32	INTV SLAVE	Responds to Single-Byte Status/ID transfer requests by executing a Single-Byte STATUS/ID transfer. Responds to Double-Byte STATUS/ID transfers request by executing a Single-Byte STATUS/ID transfer. Responds to Quad-Byte Status/ID transfer requests by executing a Quad-Byte STATUS/ID transfer
BLT	INTV MASTER	Can request multiple STATUS/ID transfers in the course of an INTERRUPT-ACKNOWLEDGE cycle
BLT	INTV SLAVE	Can properly respond to multiple STATUS/ID transfers in the course of an INTERRUPT-ACKNOWLEDGE cycle

2.5 *Interaction between MASTERS and SLAVES*

The interaction between MASTERS and SLAVES is described with the aid of flow diagrams. Paragraph 2.5.1 describes the interaction between MASTERS and SLAVES during the address broadcast phase.

Paragraph 2.5.2 describes the interaction between MASTERS and SLAVES during the data transfer phase.

Paragraph 2.5.3 describes the interaction between MASTERS and SLAVES during cycle termination.

The interaction between MASTERS and SLAVES during INTERRUPT-ACKNOWLEDGE cycles is described in Paragraph 2.5.4.

2.5.1 *Interaction entre les MAITRES et les ESCLAVES pendant la phase de diffusion d'adresse*

Une relation d'interverrouillage existe entre les MAITRES et les ESCLAVES pendant la phase de diffusion d'adresse du cycle VSB. Une fois que le MAITRE actif a commencé la phase de diffusion d'adresse, il doit maintenir tous les signaux concernés valides jusqu'à ce que sa diffusion d'adresse soit complètement acquittée. Les ESCLAVES VSB peuvent acquitter la diffusion d'adresse de trois façons différentes, devenant ainsi des ESCLAVES au repos, participants ou répondants.

Le paragraphe 2.5.1.1 et la figure 2-10, page 96, montrent le déroulement de la diffusion d'adresse et l'interaction des MAITRES et ESCLAVES pendant cette phase. Ce paragraphe donne seulement une description informelle du protocole de façon à familiariser le lecteur avec le déroulement de la diffusion d'adresse. Le paragraphe 2.5.1.2 constitue une spécification formelle de ce protocole, alors que la section 2.6 spécifie les contraintes temporelles.

2.5.1.1 *Déroulement de la phase de diffusion d'adresse*

La figure 2-10, page 96, montre l'interaction entre les MAITRES et les ESCLAVES pendant la phase de diffusion d'adresse. L'étape 1 décrit l'état intercycle des ESCLAVES VSB. Après avoir détecté un niveau haut sur la ligne PAS*, les ESCLAVES libèrent ASACK0*-ASACK1* au niveau haut et commandent la ligne AC au niveau bas. Certains ESCLAVES peuvent, en outre, commander la ligne WAIT* au niveau bas.

La diffusion d'adresse commence à l'étape 2 après que le MAITRE a obtenu le contrôle du DTB, devenant ainsi le MAITRE actif. (Le chapitre 3 décrit comment le MAITRE obtient l'autorisation d'utiliser le DTB.) Après avoir détecté PAS* au niveau haut, ce qui signifie que le MAITRE précédent ne commande plus les lignes du bus, le MAITRE actif commande les lignes SPACE0-SPACE1 pour sélectionner un espace d'adresse et les lignes SIZE0 et SIZE1 pour définir le nombre d'emplacements d'octet auquel il souhaite accéder. Il commande WR* soit au niveau haut pour un cycle de lecture, soit au niveau bas pour un cycle d'écriture. Si le cycle doit être indivisible, il commande la ligne LOCK* au niveau bas. Après avoir détecté ASACK0* et ASACK1* au niveau haut, ce qui signifie que l'ESCLAVE répondant du cycle de lecture précédent ne commande plus les lignes de données, le MAITRE actif commande les lignes AD02-AD31 pour sélectionner un groupe de 4 octets et AD00-AD01 pour sélectionner la position d'octet d'adresse la plus basse dans le groupe de 4 octets. Il commande alors PAS* au niveau bas pour signaler le transfert de l'information d'adresse aux ESCLAVES.

A l'étape 3, les ESCLAVES acquittent la diffusion d'adresse. Après avoir reçu un niveau bas sur PAS*, ils décodent l'adresse, le code d'espace et le code de dimension pour déterminer s'ils ont à répondre ou à participer au transfert de données. Les ESCLAVES capables de faire des TRANSFERTS PAR BLOC mémorisent les lignes d'adresse concernées dans un compteur d'adresses local afin de générer l'adresse pour les transferts de données qui suivent.

2.5.1 *Interaction between MASTERS and SLAVES during the address broadcast phase*

An interlocked relationship exists between MASTERS and SLAVES during the address broadcast phase of the VSB cycle. Once the active MASTER has started the address broadcast it is required to maintain all the relevant signal lines valid unt,il its address broadcast is fully acknowledged. VSB SLAVES might acknowledge the address broadcast in one of three different ways, becoming either idle, participating or responding SLAVES.

Paragraph 2.5.1.1 and Figure 2-10, page 97, show the flow of the address broadcast, and how MASTERS and SLAVES interact during this phase. This paragraph only provides an informal description of the protocol so as to familiarize the reader with the flow of the address broadcast. Paragraph 2.5.1.2 constitutes a formal specification of this protocol, while Section 2.6 specifies the timing requirements.

2.5.1.1 *Flow of the address broadcast phase*

Figure 2-10, page 97, shows the interaction between MASTERS and SLAVES during the address broadcast phase. Step 1 describes the intercycle state of VSB SLAVES. After they detect a high level on PAS*, SLAVES release ASACK0*-ASACK1* to high, and drive AC low. Some SLAVES might in addition drive WAIT* to low.

The address broadcast begins in step 2 after the MASTER has been granted use of the DTB, becoming the active MASTER. (Chapter 3 describes how the MASTER is granted use of the DTB.) After receiving PAS* high, signifying that the previous MASTER stopped driving bus lines, the active MASTER drives SPACE0-SPACE1 to select an address space and SIZE0-SIZE1 to define the number of byte locations that it wishes to access. It drives WR* either high for a read cycle, or low for a write cycle. If the cycle is to be locked, it drives LOCK* low. After detecting ASACK0*-ASACK1* high, signifying that the SLAVE that responded to the previous read cycle has stopped driving the data lines, the active MASTER drives AD02-AD31 to select a 4-byte group and AD00-AD01 to select the lowest addressed byte location to be accessed within the 4-byte group. It then drives PAS* low to signal the transfer of the addressing information to the SLAVES.

In step 3, SLAVES acknowledge the address broadcast. After they receive a low level on PAS*, they decode the address, the space code, and the size code to determine if they are to respond to or participate in the data transfer. SLAVES that have BLOCK-TRANSFER capability latch the required address lines into an on-board address counter, using it to generate the address for subsequent data transfers.

L'ESCLAVE répondant au transfert de données commande les lignes ASACK0*-ASACK1* pour indiquer sa dimension, s'assure que la ligne CACHE* est valide et libère AC et WAIT* au niveau haut. Tous les ESCLAVES au repos libèrent AC et WAIT* au niveau haut. Les ESCLAVES qui souhaitent participer au transfert de données ne doivent pas libérer WAIT* au niveau haut mais le maintenir au niveau bas jusqu'à ce qu'ils soient prêts à participer à un nouveau transfert de données.

A l'étape 4, le MAITRE termine la diffusion d'adresse. Il le fait après que la diffusion d'adresse a été acquittée par les ESCLAVES. Si le MAITRE actif détecte un niveau bas sur WAIT*, il lui faut alors maintenir la diffusion d'adresse valide jusqu'à ce qu'il détecte un niveau haut sur la ligne AC. D'autre part, si la ligne WAIT* n'est pas commandée au niveau bas, alors le MAITRE actif est autorisé à terminer la diffusion d'adresse après avoir détecté un front descendant soit sur ASACK0*, soit sur ASACK1*, ou un front montant sur AC, quelque soit l'ordre d'arrivée de ces signaux.

La façon selon laquelle les ESCLAVES VSB acquittent la phase de diffusion d'adresse peut caractériser une des conditions suivantes:

a) Un ESCLAVE répondant a été sélectionné et tous les ESCLAVES sont prêts à participer au transfert de données. Cette condition est caractérisée par un niveau bas sur ASACK0* et/ou ASACK1*, et un niveau haut sur AC.

b) Au moins un ESCLAVE participant est prêt à participer au transfert de données mais aucun ESCLAVE n'a été configuré pour répondre au cycle. Cette condition est caractérisée par un niveau haut sur ASACK0*, ASACK1* et AC, et un niveau bas sur WAIT*.

c) Aucun ESCLAVE n'a été configuré ni pour répondre ni pour participer au cycle. Cette condition est caractérisée par un niveau haut sur ASACK0*, ASACK1*, AC et WAIT*.

Dans tous les cas, le MAITRE actif termine la diffusion d'adresse. Il le fait en invalidant ses émetteurs d'adresse. Cela termine l'interaction entre MAITRES et ESCLAVES pendant la partie de diffusion d'adresse du cycle.

Selon la conception du MAITRE, relative au type d'acquittement d'adresse qu'il reçoit, et au type de cycle qu'il exécute, le MAITRE peut terminer le cycle, tel que décrit au paragraphe 2.5.3 ou procéder à un ou plusieurs transferts de données tel que décrit au paragraphe 2.5.2.

The SLAVE that will be responding to the data transfer drives ASACK0*-ASACK1* to indicate its size, ensures that the CACHE* line is valid, and releases AC and WAIT* to high. All idle SLAVES release AC and WAIT* to high. SLAVES that wish to participate in the data transfer do not release WAIT* to high, maintaining it low until after they are ready to participate in a new data transfer.

In step 4, the MASTER terminates the address broadcast. It does so after the address broadcast is acknowledged by the SLAVES. If the active MASTER detects a low level on WAIT*, then it is required to maintain the address broadcast valid until after it detects a high level on AC. On the other hand, if WAIT* is not driven low, then the active MASTER is allowed to terminate the address broadcast after it detects a falling edge on either ASACK0* or on ASACK1*, or a rising edge on AC, whichever comes first.

The way VSB SLAVES acknowledge the address broadcast phase might specify one of the following conditions:

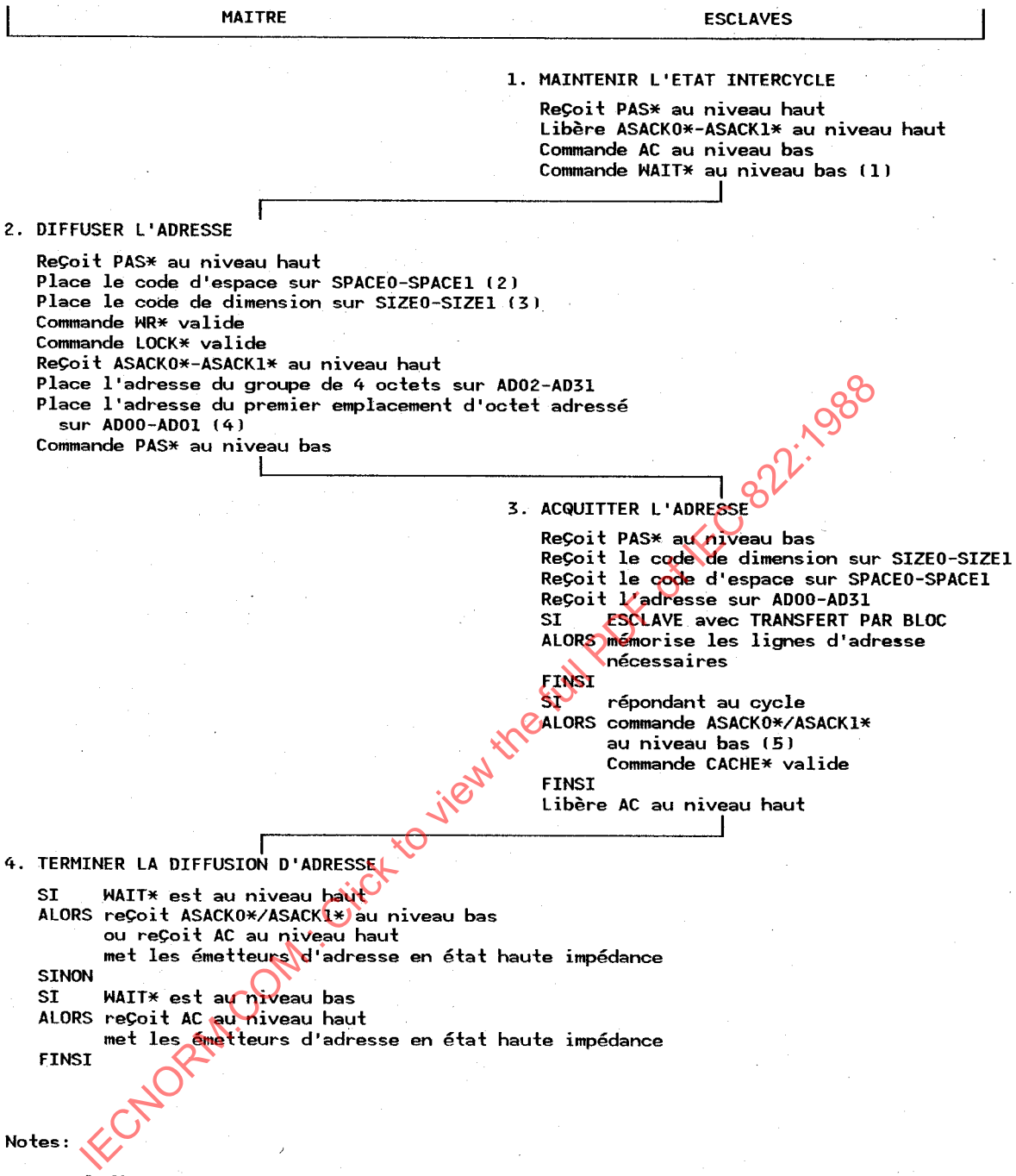
a) A responding SLAVE has been selected and all SLAVES are ready to participate in the data transfer. This condition is determined by a low level on ASACK0* and/or ASACK1*, and a high level on AC.

b) At least one participating SLAVE is ready to participate in a data transfer but no SLAVE has been configured to respond to the cycle. This condition is determined by a high level on ASACK0*, ASACK1* and AC, and a low level on WAIT*.

c) No SLAVES have been configured to either respond to or participate in the cycle. This condition is determined by a high level on ASACK0*, ASACK1*, AC and WAIT*.

In any case the active MASTER terminates the address broadcast. It does so by turning its address buffers off. This terminates the interaction between MASTERS and SLAVES during the address broadcast portion of the cycle.

Depending on the MASTER'S design, on the type of address acknowledge it receives, and on the type of cycle it executes, the MASTER might then either terminate the cycle, as described in Paragraph 2.5.3, or proceed to execute one or more data transfers, as described in Paragraph 2.5.2.



Notes:

1.- A l'étape 1, seuls les ESCLAVES qui veulent étendre la phase de diffusion d'adresse ou la phase de transfert de données commandent la ligne WAIT* au niveau bas.

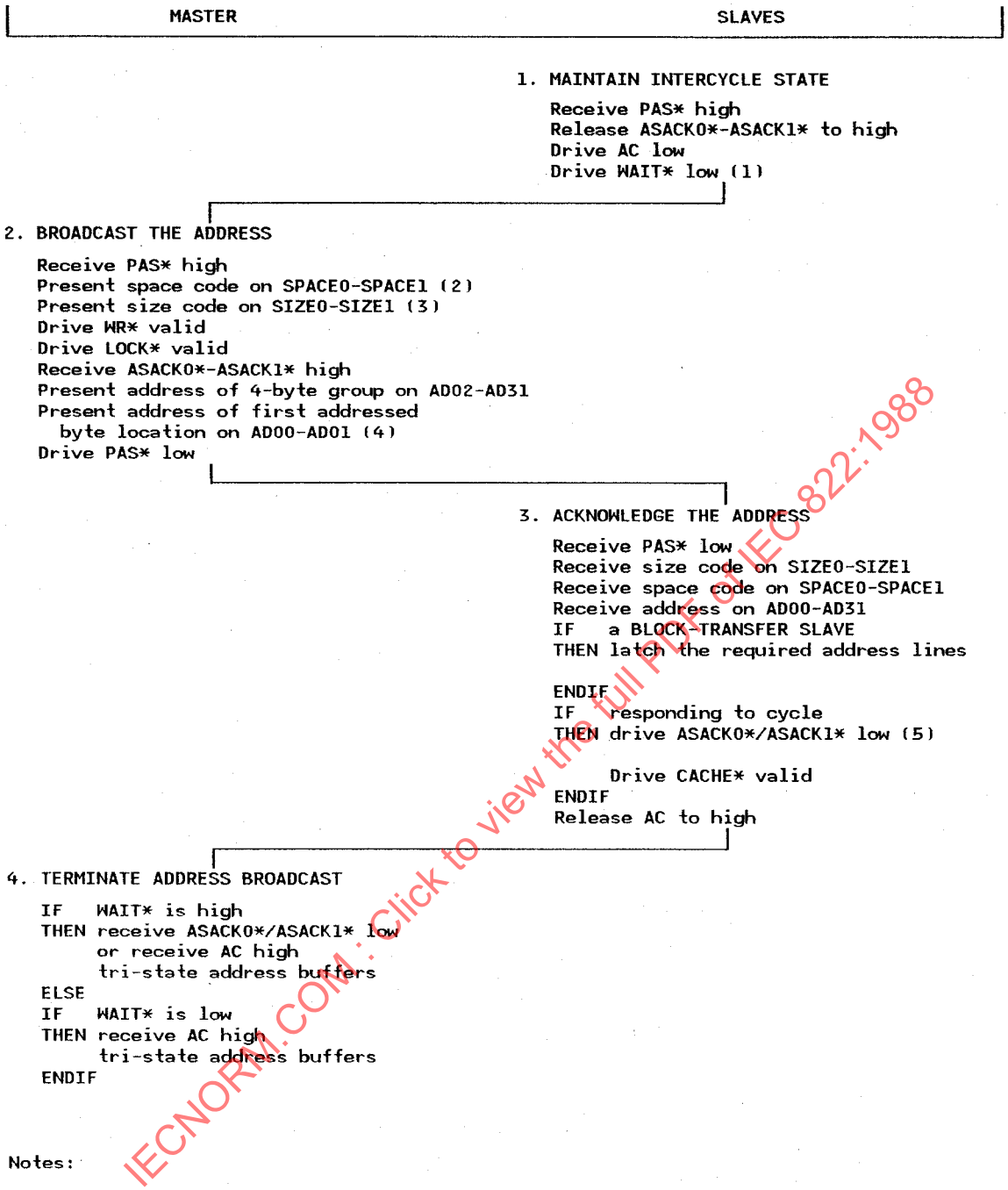
2.- A l'étape 2, le MAITRE actif commande SPACE0-SPACE1 pour sélectionner un des trois espaces d'adresse, comme défini au paragraphe 2.5.1.2, tableau 2-9.

3.- A l'étape 2, le MAITRE actif commande SIZE0-SIZE1 pour sélectionner la dimension du transfert de données, comme défini au paragraphe 2.5.1.2, tableau 2-10.

4.- A l'étape 2, le MAITRE actif commande AD00-AD01 pour sélectionner l'emplacement d'octet d'adresse la plus basse, comme défini au paragraphe 2.5.1.2, tableau 2-11.

5.- A l'étape 3, l'ESCLAVE répondant commande ASACK0*-ASACK1* pour informer le MAITRE de sa dimension, comme défini au paragraphe 2.5.1.2, tableau 2-13.

Fig. 2-10. - Organigramme de la phase de diffusion d'adresse.



Notes:

1.- In step 1, only those SLAVES that wish to extend either the address broadcast phase or the data transfer phase drive WAIT* to low.

2.- In step 2, the active MASTER drives SPACE0-SPACE1 to select one of the three address spaces, as defined in Paragraph 2.5.1.2, Table 2-9.

3.- In step 2, the active MASTER drives SIZE0-SIZE1 to select the size of the data transfer, as defined in Paragraph 2.5.1.2, Table 2-10.

4.- In step 2, the active MASTER drives AD00-AD01 to select the lowest addressed byte location, as defined in Paragraph 2.5.1.2, Table 2-11.

5.- In step 3, the responding SLAVE drives ASACK0*-ASACK1* to inform the MASTER of its size, as defined in Paragraph 2.5.1.2, Table 2-13.

Fig. 2-10. - Flow of the address broadcast phase.

2.5.1.2 *Evolution des signaux pendant la phase de diffusion d'adresse*

Ce paragraphe donne les spécifications formelles pour commander et libérer les lignes VSB pendant la phase de diffusion d'adresse des cycles VSB. Les spécifications des signaux pendant la phase de sélection d'un cycle de RECONNAISSANCE D'INTERRUPTION sont données au paragraphe 2.5.4.

REGLE 2.7:
Tous les ESCLAVES VSB DOIVENT commander AC au niveau bas chaque fois qu'ils détectent un niveau haut sur PAS*.

REGLE 2.8:
Tous les ESCLAVES VSB capables de commander WAIT* au niveau bas DOIVENT le faire chaque fois qu'ils détectent un niveau haut sur PAS*.

REGLE 2.9:
Avant de commander l'un des signaux AD00-AD31, SPACE0-SPACE1, SIZE0-SIZE1, WR*, LOCK* ou PAS*, le MAITRE actif DOIT avoir reçu l'allocation du DTB comme spécifié dans le chapitre 3.

REGLE 2.10:
Le MAITRE actif DOIT fournir le code d'espace sur SPACE0-SPACE1 pour sélectionner un espace d'adresse, comme indiqué dans le tableau 2-9.

REGLE 2.11:
Les ESCLAVES VSB DOIVENT décoder le code d'espace sur SPACE0-SPACE1 pour sélectionner l'espace d'adresse, comme indiqué dans le tableau 2-9.

Tableau 2-9

Utilisation de SPACE0 et SPACE1
pour sélectionner l'espace d'adresse

Espace d'adresse	SPACE1	SPACE0
Espace d'adresse alterné	bas	haut
Espace d'adresse E/S	haut	bas
Espace d'adresse système	haut	haut

REGLE 2.12:
Pendant la phase de diffusion d'adresse, le MAITRE actif DOIT placer le code de dimension sur SIZE0-SIZE1, conformément au tableau 2-10, pour demander la dimension du transfert de données.

REGLE 2.13:
Les ESCLAVES VSB DOIVENT décoder le code de dimension sur SIZE0-SIZE1 conformément au tableau 2-10, pour connaître la dimension voulue pour le transfert de données.

2.5.1.2 *Signaling during the address broadcast phase*

This paragraph provides the formal specifications for driving and releasing VSB signal lines during the address broadcast phase of VSB cycles. The signaling specifications during the selection phase of an INTERRUPT-ACKNOWLEDGE cycle are covered in Paragraph 2.5.4.

RULE 2.7:
All VSB SLAVES MUST drive AC low whenever they detect a high level on PAS*.

RULE 2.8:
VSB SLAVES that are capable of driving WAIT* low MUST do so whenever they detect a high level on PAS*.

RULE 2.9:
Before driving any of AD00-AD31, SPACE0-SPACE1, SIZE0-SIZE1, WR*, LOCK* or PAS*, the active MASTER MUST be granted use of the Data Transfer Bus as specified in Chapter 3.

RULE 2.10:
The active MASTER MUST provide the space code on SPACE0-SPACE1 to select an address space, as shown in Table 2-9.

RULE 2.11:
VSB SLAVES MUST decode the space code on SPACE0-SPACE1 to select an address space, as shown in Table 2-9.

Table 2-9

Use of SPACE0 and SPACE1
to select the address space

Address space	SPACE1	SPACE0
Alternate address space	low	high
I/O address space	high	low
System address space	high	high

RULE 2.12:
During the address broadcast phase, the active MASTER MUST place the size code on SIZE0-SIZE1 to request the size of the data transfer, as shown in Table 2-10.

RULE 2.13:
VSB SLAVES MUST decode the size code on SIZE0-SIZE1 to determine the requested size of the data transfer, as shown in Table 2-10.

Tableau 2-10

Codage de SIZE0 et SIZE1 pour une dimension requise du transfert

Dimension du transfert de données	SIZE1	SIZE0
Transfert de données quadruple octet	bas	bas
Transfert de données octet unique	bas	haut
Transfert de données double octet	haut	bas
Transfert de données triple octet	haut	haut

REGLE 2.14:
Pendant la phase de diffusion d'adresse, le MAITRE actif DOIT commander les lignes d'adresse AD00-AD01, conformément au tableau 2-11, pour sélectionner le déplacement qui, dans un groupe de 4 octets, donne l'emplacement de l'octet d'adresse la plus basse.

REGLE 2.15:
Pendant la phase de diffusion d'adresse des cycles VSB, les ESCLAVES DOIVENT décoder les lignes d'adresse AD00-AD01, conformément au tableau 2-11, pour sélectionner le déplacement qui, dans un groupe de 4 octets, donne l'emplacement de l'octet d'adresse la plus basse.

Tableau 2-11

Utilisation de AD00 et AD01 pour sélectionner l'emplacement de l'octet d'adresse la plus basse à atteindre

Octet d'adresse la plus basse à atteindre	AD01	AD00
OCTET(0)	bas	bas
OCTET(1)	bas	haut
OCTET(2)	haut	bas
OCTET(3)	haut	haut

REGLE 2.16:
Pendant la phase de diffusion d'adresse, les MAITRES actifs DOIVENT coder SIZE0, SIZE1, AD00 et AD01, conformément au tableau 2-12, pour définir les emplacements d'octet à atteindre.

REGLE 2.17:
Les ESCLAVES DOIVENT décoder SIZE0, SIZE1, AD00 et AD01, conformément au tableau 2-12, pour déterminer quels sont les emplacements d'octet à atteindre.

REGLE 2.18:
SI le MAITRE actif veut exécuter un cycle d'écriture, c'est-à-dire ranger une donnée dans des emplacements d'octet fournis par des ESCLAVES,
ALORS il DOIT commander la ligne WR* au niveau bas.

Table 2-10

Encoding of SIZE0 and SIZE1 for requested size of the transfer

Requested size of data transfer	SIZE1	SIZE0
Quad-Byte data transfer	low	low
Single-Byte data transfer	low	high
Double-Byte data transfer	high	low
Triple-Byte data transfer	high	high

RULE 2.14:
During the address broadcast phase, the active MASTER MUST drive address lines AD00-AD01 to select the offset of the lowest addressed byte location that is accessed within a 4-byte group, as shown in Table 2-11.

RULE 2.15:
During the address broadcast portion of VSB cycles, SLAVES MUST decode address lines AD00-AD01 to select the offset of the lowest addressed byte location that is accessed within a 4-byte group, as shown in Table 2-11.

Table 2-11

Use of AD00 and AD01 to select the lowest addressed byte location to be accessed

Lowest address byte accessed	AD01	AD00
BYTE(0)	low	low
BYTE(1)	low	high
BYTE(2)	high	low
BYTE(3)	high	high

RULE 2.16:
During the address broadcast phase, the active MASTERS MUST encode SIZE0, SIZE1, AD00 and AD01 to define the byte locations that are to be accessed, as shown in Table 2-12.

RULE 2.17:
SLAVES MUST decode SIZE0, SIZE1, AD00 and AD01 to determine which byte locations are to be accessed, as shown in Table 2-12.

RULE 2.18:
IF the active MASTER wants to execute a write cycle, i.e. store data in byte locations provided by SLAVES,

THEN it MUST drive the WR* line to a low level.

REGLE 2.19:
SI le MAITRE actif veut exécuter un cycle de lecture, c'est-à-dire extraire des données d'emplacements d'octet fournis par des ESCLAVES,
ALORS il DOIT garantir que la ligne WR* est au niveau haut.

REGLE 2.20:
SI le MAITRE actif veut exécuter un cycle d'ACCES INDIVISIBLE,
ALORS il DOIT commander la ligne LOCK* au niveau bas.

REGLE 2.21:
Tous les ESCLAVES DOIVENT décoder AD00-AD31 et SPACE0-SPACE1 et NE DOIVENT commander ni ASACK0* ni ASACK1* au niveau bas s'ils ne possèdent pas les emplacements d'octet demandés dans l'espace d'adresse requis.

Tableau 2-12

Codage de SIZE0, SIZE1, AD00 et AD01 pour définir les emplacements d'octet à atteindre

Emplacement d'octet à atteindre	SIZE1	SIZE0	AD01	AD00
OCTET(0)	bas	haut	bas	bas
OCTET(1)	bas	haut	bas	haut
OCTET(2)	bas	haut	haut	bas
OCTET(3)	bas	haut	haut	haut
OCTET(0-1)	haut	bas	bas	bas
OCTETS(1-2)	haut	bas	bas	haut
OCTETS(2-3)	haut	bas	haut	bas
OCTETS(3)(0)	haut	bas	haut	haut
OCTETS(0-2)	haut	haut	bas	bas
OCTETS(1-3)	haut	haut	bas	haut
OCTETS(2-3)(0)	haut	haut	haut	bas
OCTETS(3)(0-1)	haut	haut	haut	haut
OCTETS(0-3)	bas	bas	bas	bas
OCTETS(1-3)(0)	bas	bas	bas	haut
OCTETS(2-3)(0-1)	bas	bas	haut	bas
OCTETS(3)(0-2)	bas	bas	haut	haut

REGLE 2.22:
Pendant la phase de diffusion d'adresse, l'ESCLAVE répondant DOIT commander et le MAITRE actif DOIT décoder le code sur ASACK0*-ASACK1*, conformément au tableau 2-13, pour définir la dimension de l'ESCLAVE.

RULE 2.19:
IF the active MASTER wants to execute a read cycle, i.e. retrieve data from byte locations provided by SLAVES,

THEN it MUST ensure that the WR* line is high.

RULE 2.20:
IF the active MASTER wants to execute an INDIVISIBLE-ACCESS cycle,
THEN it MUST drive the LOCK* line to a low level.

RULE 2.21:
All SLAVES MUST decode AD00-AD31 and SPACE0-SPACE1, and MUST NOT drive either ASACK0* or ASACK1* to low if they do not provide the requested byte locations in the requested address space.

Table 2-12

Encoding of SIZE0, SIZE1, AD00 and AD01 to define the byte locations to be accessed

Byte locations to be accessed	SIZE1	SIZE0	AD01	AD00
BYTE(0)	low	high	low	low
BYTE(1)	low	high	low	high
BYTE(2)	low	high	high	low
BYTE(3)	low	high	high	high
BYTE(0-1)	high	low	low	low
BYTE(1-2)	high	low	low	high
BYTE(2-3)	high	low	high	low
BYTE(3)(0)	high	low	high	high
BYTE(0-2)	high	high	low	low
BYTE(1-3)	high	high	low	high
BYTE(2-3)(0)	high	high	high	low
BYTE(3)(0-1)	high	high	high	high
BYTE(0-3)	low	low	low	low
BYTE(1-3)(0)	low	low	low	high
BYTE(2-3)(0-1)	low	low	high	low
BYTE(3)(0-2)	low	low	high	high

RULE 2.22:
During the address broadcast phase, the responding SLAVE MUST drive and the active MASTER MUST decode the code on ASACK0*-ASACK1* to define the size of the SLAVE, as shown in Table 2-13.

Tableau 2-13

Codage de ASACK0* et ASACK1* pour définir la dimension de l'ESCLAVE

Dimension de l'ESCLAVE	ASACK1*	ASACK0*
ESCLAVE D32 répondant	bas	bas
ESCLAVE D16 répondant	bas	haut
ESCLAVE D08 répondant	haut	bas
Pas d'ESCLAVE répondant	haut	haut

AUTORISATION 2.1:
L'ESCLAVE répondant PEUT commander la ligne CACHE* à un niveau valide pendant le cycle.

SUGGESTION 2.1:
Pour une performance optimale, concevoir les ESCLAVES de telle sorte qu'ils libèrent AC au niveau haut dès que possible.

OBSERVATION 2.13:
Les ESCLAVES participants maintiennent WAIT* au niveau bas pendant la phase de diffusion d'adresse.

REGLE 2.23:
SI un ESCLAVE participant ne libère pas la ligne WAIT* au niveau haut pendant la phase de diffusion d'adresse,
ALORS il DOIT la libérer au niveau haut pendant la phase de transfert de données du même cycle.

OBSERVATION 2.14:
Un module de dépassement de temps n'est pas nécessaire dans le système VSB puisque:

a) Tous les ESCLAVES VSB doivent libérer AC au niveau haut après avoir terminé le décodage d'adresse.

b) Les ESCLAVES participants maintiennent un niveau bas sur WAIT* jusqu'à ce qu'ils aient effectué le transfert de données.

c) L'ESCLAVE répondant doit commander ASACK0* et/ou ASACK1* au niveau bas lorsqu'il finit le décodage d'adresse.

Par conséquent, si le MAITRE actif détecte un niveau haut sur ASACK0*-ASACK1*, AC et WAIT* après avoir commandé PAS* au niveau bas, alors il est sûr qu'aucun ESCLAVE VSB n'est configuré pour répondre ou participer à un accès à l'adresse requise. A ce moment-là, le MAITRE actif peut terminer le cycle.

2.5.2 *Interaction entre les MAITRES et les ESCLAVES pendant le transfert de données*

Une relation d'interverrouillage existe entre les MAITRES et les ESCLAVES pendant la phase de transfert de données des cycles VSB. Une fois que le MAITRE actif a déclenché un transfert de données, il lui faut maintenir toutes les lignes de signaux concernés valides jusqu'à ce qu'il ait été acquitté correctement. Les ESCLAVES VSB acquittent le transfert de données de deux façons différentes selon qu'ils participent ou qu'ils répondent au cycle en cours.

Table 2-13

Encoding of ASACK0* and ASACK1* to define the size of the SLAVE

Size of SLAVE	ASACK1*	ASACK0*
D32 SLAVE responding	low	low
D16 SLAVE responding	low	high
D08 SLAVE responding	high	low
No SLAVE responding	high	high

PERMISSION 2.1:
The responding SLAVE MAY drive CACHE* to a valid level during the cycle.

SUGGESTION 2.1:
For optimum performance, design SLAVES so that they release AC to high as soon as possible.

OBSERVATION 2.13:
Participating SLAVES maintain WAIT* low during the address broadcast phase.

RULE 2.23:
IF a participating SLAVE does not release WAIT* to high during the address broadcast phase,
THEN it MUST release it to high during the data transfer phase of the same cycle.

OBSERVATION 2.14:
A bus timer module is not required in the VSB system since:

a) All VSB SLAVES are required to release AC to high after they finish decoding the address.

b) Participating SLAVES maintain a low level on WAIT* until after they have finished the data transfer.

c) The responding SLAVE is required to drive ASACK0* and/or ASACK1* low when it finishes decoding the address.

Therefore, if the active MASTER detects a high level on ASACK0*-ASACK1*, AC and WAIT* after it has driven PAS* low, then it is assured that no VSB SLAVES are configured to either respond to or participate in an access to the requested address. At such a time, the active MASTER can terminate the cycle.

2.5.2 *Interaction between MASTERS and SLAVES during the data transfer*

An interlocked relationship exists between MASTERS and SLAVES during the data transfer phase of VSB cycles. Once the active MASTER has started a data transfer, it is required to maintain all the relevant signal lines valid until it is acknowledged in the proper manner. VSB SLAVES acknowledge the data transfer in two different ways, depending whether they are participating in, or responding to the cycle.

Le paragraphe 2.5.2.1 et la figure 2-11, page 110, décrivent le déroulement et l'interaction entre le MAITRE actif et les ESCLAVES sélectionnés pendant un transfert de données en écriture. Le paragraphe 2.5.2.2 et la figure 2-12, page 114, décrivent l'organigramme et l'interaction entre le MAITRE actif et les ESCLAVES sélectionnés pendant un transfert de données en lecture. Ces paragraphes fournissent simplement une description informelle du protocole pour familiariser le lecteur avec le déroulement de la phase de transfert de données. Le paragraphe 2.5.2.3 constitue une spécification formelle de ce protocole, alors que la section 2.6 spécifie les contraintes temporelles.

2.5.2.1 *Déroulement d'un transfert de données en écriture*

La figure 2-11, présente l'organigramme d'un transfert de données en écriture, c'est-à-dire un transfert de données que le MAITRE actif utilise pour ranger des données dans des emplacements d'octet des ESCLAVES. Le transfert commence après que le MAITRE actif a terminé l'étape 1 de la phase de diffusion d'adresse. Il déclenche alors le transfert de données en exécutant l'étape 2, consistant à placer ses données sur les lignes AD00-AD31. Lorsque le MAITRE actif détecte un niveau haut à la fois sur ACK* et ERR*, il commande DS* au niveau bas, ce qui signale aux ESCLAVES participants et à l'ESCLAVE répondant que la phase de transfert de données est en cours.

Tous les ESCLAVES sélectionnés acquittent le transfert de données à l'étape 3.

L'ESCLAVE répondant détecte DS* et WR* commandés au niveau bas et acquitte le transfert de données de l'une des trois façons possibles:

a) Il prélève les données sur AD00-AD31, les range dans les emplacements d'octet sélectionnés pendant la phase de diffusion d'adresse, puis commande ACK* au niveau bas.

b) Il détermine qu'il n'est pas capable d'effectuer l'opération d'écriture requise (par exemple écriture dans une mémoire ROM) et commande la ligne ERR* au niveau bas.

c) Il commande ACK* au niveau bas puis dans un délai maximal spécifié, il commande aussi la ligne ERR* au niveau bas.

De plus, si l'ESCLAVE répondant a atteint la limite de son intervalle d'adressage (c'est-à-dire qu'il répond pour le dernier emplacement d'octet auquel il peut accéder sans une nouvelle diffusion d'adresse par le MAITRE actif), alors il commande AC au niveau bas avant de commander ou ACK* ou ERR* au niveau bas.

Tous les ESCLAVES participants prélèvent les données sur AD00-AD31 et libèrent WAIT* au niveau haut. De plus, si un ESCLAVE participant a atteint la limite de son intervalle d'adressage, il commande AC au niveau bas avant de libérer WAIT* au niveau haut.

A l'étape 4, quand le MAITRE actif reçoit ACK* et/ou ERR* au niveau bas et après avoir détecté un niveau haut sur WAIT*, il termine le transfert de données en permettant à DS* de passer au niveau haut.

Paragraph 2.5.2.1 and Figure 2-11, page 111, describe the flow and the interaction between the active MASTER and the selected SLAVES during a write data transfer. Paragraph 2.5.2.2 and Figure 2-12, page 115, describe the flow and the interaction between the active MASTER and the selected SLAVES during a read data transfer. These paragraphs only provide an informal description of the protocol to familiarize the reader with the flow of the data transfer phase. Paragraph 2.5.2.3 constitutes a formal specification of this protocol, while Section 2.6 specifies the timing requirements.

2.5.2.1 *Flow of a write data transfer*

Figure 2-11, shows the flow diagram for a write data transfer, i.e. a data transfer that the active MASTER uses to store data in byte locations provided by SLAVES. The transfer begins after the active MASTER has terminated the address broadcast phase step 1. It then initiates the data transfer by executing step 2, placing its data on AD00-AD31. When the active MASTER detects a high level on both ACK* and ERR*, it drives DS* low, signaling to the participating SLAVES and to the responding SLAVE that the data transfer phase is in progress.

All selected SLAVES acknowledge the data transfer in step 3.

The responding SLAVE detects DS* driven low and WR* driven low and acknowledges the transfer in one of three possible ways:

a) It captures the data from AD00-AD31, stores it into the byte locations that were selected during the address broadcast, and then drives ACK* low.

b) It determines that it is not capable of performing the requested write operation (e.g. write to ROM) and drives ERR* low.

c) It drives ACK* low and then, within a specified maximum time, drives ERR* low as well.

In addition, if the responding SLAVE has reached the end of its self addressing range (i.e. it is responding to the last byte location it can access without a new address broadcast from the active MASTER), then it drives AC low before driving either ACK* or ERR* low.

All participating SLAVES capture the data from AD00-AD31 and release WAIT* to high. In addition, if a participating SLAVE has reached the end of its self addressing range, then it drives AC low before releasing WAIT* to high.

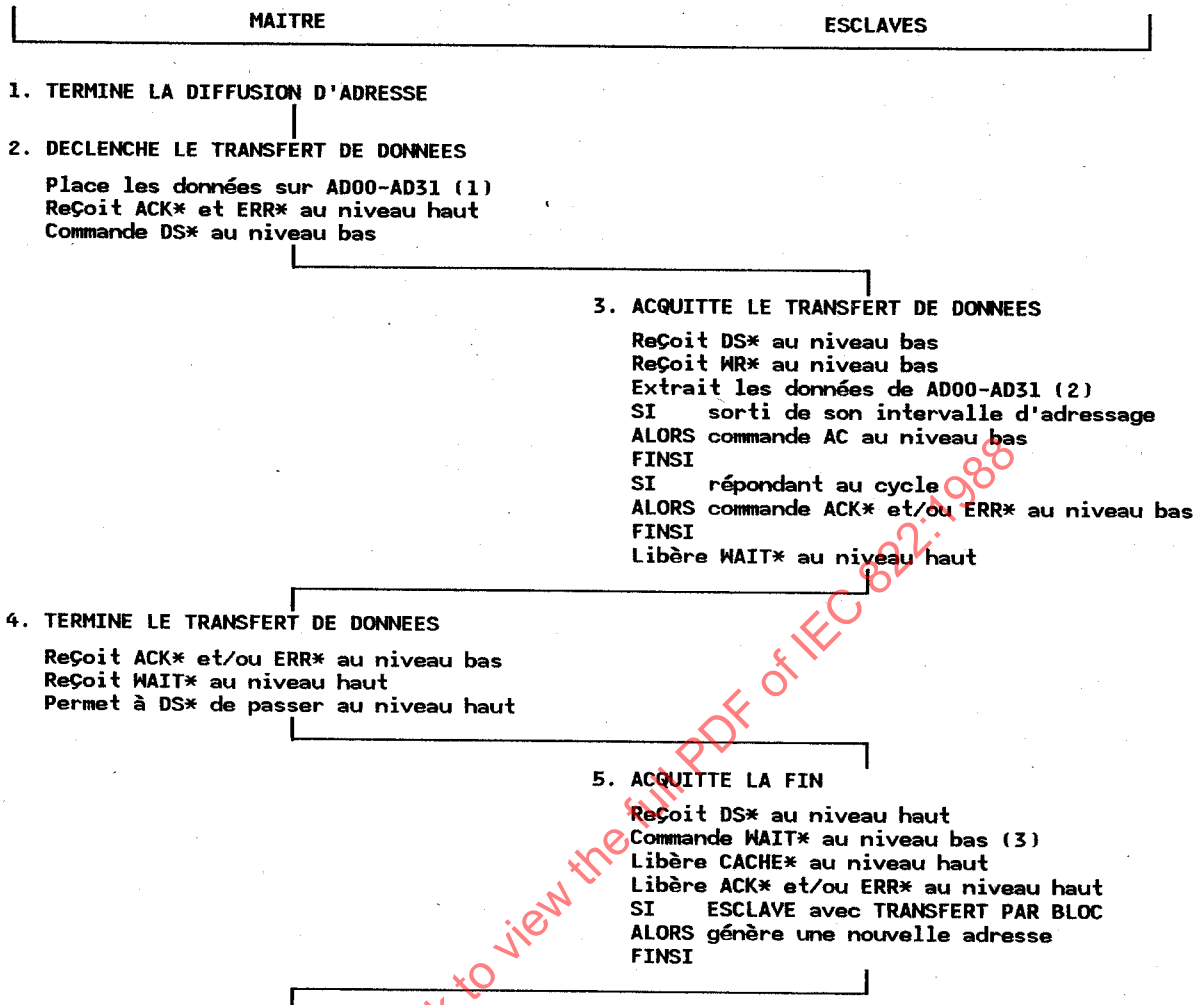
In step 4, when the active MASTER receives ACK* and/or ERR* driven low, and after it detects a high level on WAIT*, it terminates the data transfer by allowing DS* to go high.

A l'étape 5, après avoir reçu DS* au niveau haut, les ESCLAVES qui souhaitent participer au prochain cycle ou à la prochaine phase de transfert de donnée commandent la ligne WAIT* au niveau bas. L'ESCLAVE répondant libère CACHE*, ACK* et ERR* au niveau haut. Les ESCLAVES qui n'ont pas atteint la limite de leur intervalle d'adressage génèrent l'adresse pour le prochain transfert de données.

Cela termine le transfert de données en écriture. A l'étape 6, le MAITRE actif peut déclencher un autre transfert de données en écriture ou peut terminer le cycle comme décrit au paragraphe 2.5.3. Il déclenche un autre transfert de données s'il a d'autres données à transférer, mais seulement si ERR* n'était pas au niveau bas à l'étape 4 et si AC est encore au niveau haut. Si une quelconque de ces conditions n'est pas remplie, c'est-à-dire AC est au niveau bas ou ERR* était au niveau bas à l'étape 4, ou si toutes les données ont été transférées, alors il termine le cycle.

In step 5, after receiving DS* high, the SLAVES that wish to participate in the next cycle or the next data transfer phase drive WAIT* low. The responding SLAVE releases CACHE*, ACK* and ERR* to high. SLAVES that have not reached the end of their self addressing range generate the address for the next data transfer.

This terminates the write data transfer. In step 6 the active MASTER might initiate another write data transfer, or it might terminate the cycle as described in Paragraph 2.5.3. It initiates another data transfer if it has more data to transfer, but only if ERR* was not low in step 4, and if AC is still high. If any of these conditions is not met, i.e. AC is low, or ERR* was low in step 4, or if it has transferred all the data it needed to, then it terminates the cycle.



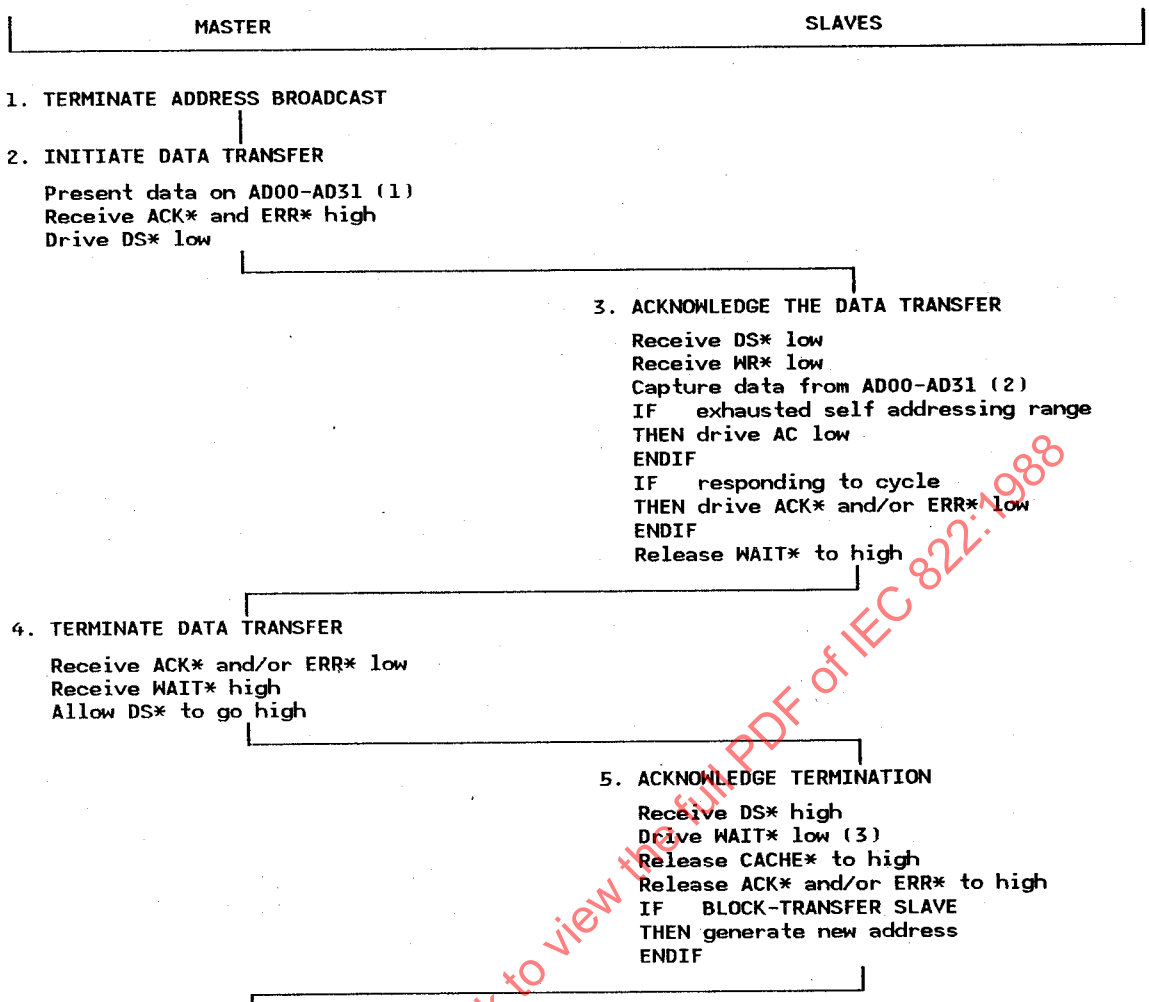
Notes:

1.- A l'étape 2, les octets de données placés par le MAITRE sur les lignes de données dépendent du type de cycle qu'il exécute comme défini au paragraphe 2.5.2.3, tableau 2-14.

2.- A l'étape 3, l'ESCLAVE prélève les données sur les lignes AD00-AD31, comme défini au paragraphe 2.5.2.3, tableaux 2-15, 2-16 et 2-17.

3.- A l'étape 5, seuls les ESCLAVES qui désirent participer au transfert de données et en sont capables commandent la ligne WAIT* au niveau bas.

Fig. 2-11. - Organigramme d'un transfert de données en écriture.



Notes:

1.- In step 2, the data bytes that the MASTER places on the data lines depend on the type of cycle it is executing, as defined in Paragraph 2.5.2.3, Table 2-14.

2.- In step 3, the SLAVE captures the data off AD00-AD31, as defined in Paragraph 2.5.2.3, Tables 2-15, 2-16 and 2-17.

3.- In step 5, only those SLAVES that are capable of, and wish to participate in the data transfer, drive WAIT* low.

Fig. 2-11. - Flow of a write data transfer.

2.5.2.2 *Déroulement d'un transfert de données en lecture*

La figure 2-12, page 114, montre l'organigramme d'un transfert de données en lecture, c'est-à-dire un transfert utilisé par le MAITRE actif pour récupérer des données à des emplacements d'octet fournis par des ESCLAVES. Le transfert de données commence après que le MAITRE actif a terminé l'étape 1 de la phase de diffusion d'adresse. Il déclenche alors le transfert de données exécutant l'étape 2. Quand le MAITRE actif détecte un niveau haut à la fois sur ACK* et ERR*, il commande DS* au niveau bas, signalant ainsi aux ESCLAVES sélectionnés que la phase de transfert de données est en cours.

Tous les ESCLAVES sélectionnés acquittent le transfert de données à l'étape 3. L'ESCLAVE répondant détecte DS* au niveau bas et un niveau haut sur WR* et acquitte alors le transfert de l'une des trois façons possibles:

a) Il place les données extraites de sa mémoire interne sur les lignes de données appropriées et positionne la ligne ACK* au niveau bas.

b) Il détermine qu'il n'est pas capable d'effectuer l'opération de lecture demandée et commande la ligne ERR* au niveau bas.

c) Il commande ACK* au niveau bas puis dans un délai maximal spécifié, commande la ligne ERR* au niveau bas si nécessaire (cette méthode convient au cas où l'ESCLAVE détecte une erreur de parité).

Si l'ESCLAVE répondant commande sa ligne IRQ* au niveau bas et que le transfert de données en lecture accède à son registre MOT D'ETAT/ID, alors il libère, en plus, sa ligne IRQ* au niveau haut. S'il a atteint la limite de son intervalle d'adressage (c'est-à-dire qu'il est en train d'accéder à la dernière position d'octet à laquelle il peut accéder sans une nouvelle diffusion d'adresse de la part du MAITRE actif), alors il commande la ligne AC au niveau bas avant de commander soit ACK*, soit ERR* au niveau bas. Les ESCLAVES participants prélèvent les données et repositionnent la ligne WAIT* au niveau haut. S'ils ont atteint la limite de leur domaine d'adressage, alors ils commandent AC au niveau bas, avant de libérer WAIT* au niveau haut.

A l'étape 4, lorsque le MAITRE actif reçoit ACK* et/ou ERR* au niveau bas et après avoir détecté un niveau haut sur WAIT*, il termine le transfert de données en permettant à DS* de passer au niveau haut.

A l'étape 5, après réception de DS* au niveau haut, les ESCLAVES qui désirent participer au prochain cycle ou au transfert de données commandent WAIT* au niveau bas. L'ESCLAVE répondant libère CACHE*, ACK* et ERR* au niveau haut. Les ESCLAVES qui n'ont pas atteint la limite de leur intervalle d'adressage génèrent l'adresse pour le prochain transfert de données.

Cela termine le transfert de données en lecture. A l'étape 6, le MAITRE actif peut soit déclencher un autre transfert de données en lecture soit terminer le cycle comme décrit au paragraphe 2.5.3. Il déclenche un autre transfert de données s'il a d'autres données à transférer, mais seulement si ERR* ou AC n'étaient pas au niveau bas à l'étape 4. Si l'une de ces conditions n'est pas remplie, c'est-à-dire que soit ERR*, soit AC* sont au niveau bas à l'étape 4, ou que toutes les données voulues ont été transférées, alors il termine le cycle.

2.5.2.2 *Flow of a read data transfer*

Figure 2-12, page 115, shows the flow diagram for a read data transfer, i.e. a data transfer that the active MASTER uses to retrieve data from byte locations provided by SLAVES. The data transfer begins after the active MASTER has completed the address broadcast phase in step 1. It then initiates the data transfer by executing step 2. When the active MASTER detects a high level on both ACK* and ERR*, it drives DS* low, signaling to the selected SLAVES that the data transfer phase is in progress.

All selected SLAVES acknowledge the data transfer in step 3. The responding SLAVE detects DS* driven low and a high level on WR* and acknowledges the transfer in one of three possible ways:

a) It places the data it has retrieved from its internal storage on the appropriate data lines and then drives ACK* low.

b) It determines that it is not capable of performing the requested read operation and drives ERR* low.

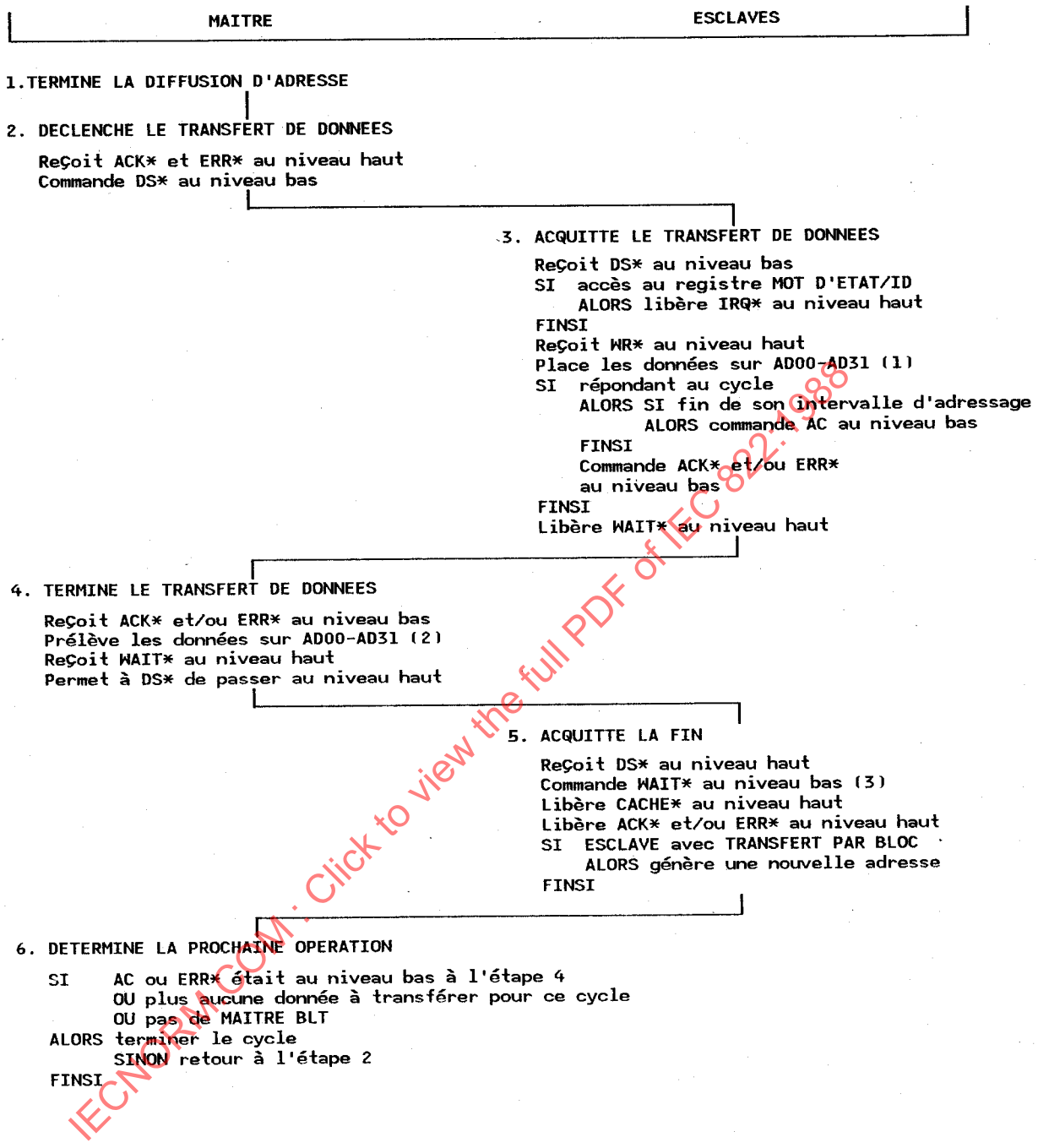
c) It drives ACK* low and then, within a specified maximum time, drives ERR* low as well. (This method is appropriate if the SLAVE detects a parity error.)

If the responding SLAVE is driving its IRQ* line low and the read data transfer is accessing its STATUS/ID register, then, in addition, it releases its IRQ* line to high. If it has reached the end of its self addressing range, (i.e. it is responding to the last byte location it can access without a new address broadcast from the active MASTER), then it drives AC low prior to driving either ACK* or ERR* low. Participating SLAVES capture the data and release WAIT* to high. If they reached the end of its self addressing range, then it drives AC low before releasing WAIT* to high.

In step 4, when the active MASTER receives ACK* and/or ERR* driven low, and after it detects a high level on WAIT*, it terminates the data transfer by allowing DS* to go high.

In step 5, after receiving DS* high, SLAVES that wish to participate in the next cycle or data transfer drive WAIT* low. The responding SLAVE releases CACHE*, ACK* and ERR* to high. SLAVES that have not reached the end of their self addressing range generate the address for the next data transfer.

This terminates the read data transfer. In step 6 the active MASTER, might initiate another read data transfer, or it might terminate the cycle as described in Paragraph 2.5.3. It initiates another data transfer if it has more data to transfer, but only if ERR* or AC were not low in step 4. If any of these conditions is not met, i.e. either ERR* or AC were low in step 4, or if it has transferred all the data it needed to, then it terminates the cycle.



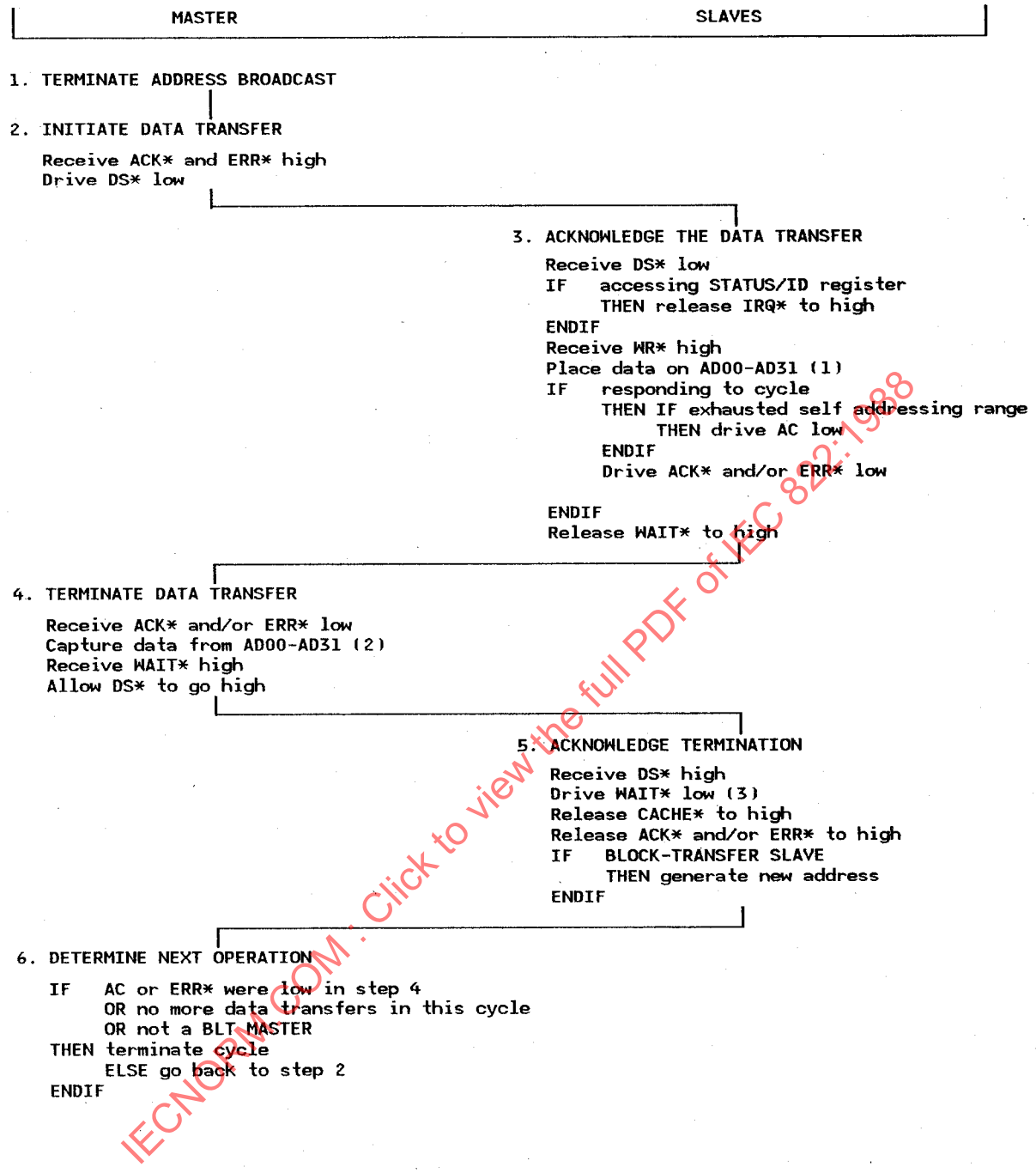
Notes:

1.- A l'étape 3, les lignes de données commandées par l'ESCLAVE répondant sont fonction de sa dimension et du type de cycle auquel il répond, comme défini au paragraphe 2.5.2.3, tableaux 2-15, 2-16 et 2-17.

2.- A l'étape 4, les lignes de données lues par le MAITRE sont fonction de la dimension de l'ESCLAVE répondant et du type de cycle que le MAITRE exécute, comme défini au paragraphe 2.5.2.3, tableaux 2-15, 2-16 et 2-17.

3.- A l'étape 5, seuls les ESCLAVES qui désirent participer au transfert de données et en sont capables commandent la ligne WAIT* au niveau bas.

Fig. 2-12. - Organigramme d'un transfert de données en lecture.



Notes:

1.- In step 3, the data lines that the responding SLAVE drives valid are determined by its size and by the type of cycle it is responding to, as defined in Paragraph 2.5.2.3, Tables 2-15, 2-16 and 2-17.

2.- In step 4, the data lines that the MASTER captures the data from are determined by the size of the responding SLAVE and by the type of cycle that the MASTER is executing, as defined in Paragraph 2.5.2.3, Tables 2-15, 2-16 and 2-17.

3.- In step 5, only those SLAVES that are capable of, and wish to participate in the data transfer, drive WAIT* low.

Fig. 2-12. - Flow of a read data transfer.

2.5.2.3 *Evolution des signaux pendant la phase de transfert de données*

REGLE 2.24:
Pour permettre le dimensionnement dynamique du bus pendant les transferts en écriture, le MAITRE actif DOIT valider les données sur AD00-AD31 comme indiqué au tableau 2-14.

Tableau 2-14

Positionnement des données valides sur AD00-AD31
par le MAITRE actif pendant les cycles d'écriture

Type de cycle	AD31-AD24	AD23-AD16	AD15-AD08	AD07-AD00
OCTET(0) OCTET(1) OCTET(2) OCTET(3)	DONNEE(3) DONNEE(3) DONNEE(3) DONNEE(3)	DONNEE(3) DONNEE(3)	DONNEE(3)	DONNEE(3)
OCTET(0-1) OCTET(1-2) OCTET(2-3) OCTET(3)(0)	DONNEE(2) DONNEE(2) DONNEE(2) DONNEE(2)	DONNEE(3) DONNEE(2) DONNEE(3) DONNEE(2)	DONNEE(3) DONNEE(2)	DONNEE(3) DONNEE(2)
OCTET(0-2) OCTET(1-3) OCTET(2-3)(0) OCTET(3)(0-1)	DONNEE(1) DONNEE(1) DONNEE(1) DONNEE(1)	DONNEE(2) DONNEE(1) DONNEE(2) DONNEE(1)	DONNEE(3) DONNEE(2) DONNEE(1)	DONNEE(3) DONNEE(2) DONNEE(1)
OCTET(0-3) OCTET(1-3)(0) OCTET(2-3)(0-1) OCTET(3)(0-2)	DONNEE(0) DONNEE(0) DONNEE(0) DONNEE(0)	DONNEE(1) DONNEE(0) DONNEE(1) DONNEE(0)	DONNEE(2) DONNEE(1) DONNEE(0)	DONNEE(3) DONNEE(2) DONNEE(1) DONNEE(0)

Note:
Un blanc dans le tableau signifie que le niveau des lignes de données est indéfini pendant le cycle indiqué.

SUGGESTION 2.2:
Pour optimiser la performance, concevoir les MAITRES tels que DS* puisse revenir au niveau haut aussitôt que ACK* et/ou ERR* sont passés au niveau bas, pourvu que WAIT* soit au niveau haut.

OBSERVATION 2.15:
Le MAITRE actif commande WR* valide pendant la phase de diffusion d'adresse.

REGLE 2.25:
Pendant les cycles de lecture, un ESCLAVE D32 répondant DOIT prélever les données des emplacements d'octet comme indiqué au tableau 2-15, et DOIT les placer sur AD00-AD31.

REGLE 2.26:
Pendant les cycles d'écriture, les ESCLAVES D32 DOIVENT prélever les données des lignes AD00-AD31 comme indiqué au tableau 2-15, et DOIVENT les ranger dans les emplacements d'octet adressés.

2.5.2.3 *Signaling during the data transfer phase*

RULE 2.24:
To support dynamic bus sizing during write data transfers, the active MASTER MUST drive valid data on AD00-AD31 as shown in Table 2-14.

Table 2-14

Placement of valid data on AD00-AD31 by the active MASTER during write cycles

Type of cycle	AD31-AD24	AD23-AD16	AD15-AD08	AD07-AD00
BYTE(0) BYTE(1) BYTE(2) BYTE(3)	DATA(3) DATA(3) DATA(3) DATA(3)	DATA(3) DATA(3)	DATA(3)	DATA(3)
BYTE(0-1) BYTE(1-2) BYTE(2-3) BYTE(3)(0)	DATA(2) DATA(2) DATA(2) DATA(2)	DATA(3) DATA(2) DATA(3) DATA(2)	DATA(3) DATA(2)	DATA(3) DATA(2)
BYTE(0-2) BYTE(1-3) BYTE(2-3)(0) BYTE(3)(0-1)	DATA(1) DATA(1) DATA(1) DATA(1)	DATA(2) DATA(1) DATA(2) DATA(1)	DATA(3) DATA(2) DATA(1)	DATA(3) DATA(2) DATA(1)
BYTE(0-3) BYTE(1-3)(0) BYTE(2-3)(0-1) BYTE(3)(0-2)	DATA(0) DATA(0) DATA(0) DATA(0)	DATA(1) DATA(0) DATA(1) DATA(0)	DATA(2) DATA(1) DATA(0)	DATA(3) DATA(2) DATA(1) DATA(0)

Note:
A blank entry in the table means that the state of the indicated data lines is undefined during the indicated cycle.

SUGGESTION 2.2:
For optimum performance, design MASTERS to allow DS* to go high as soon as possible after ACK* and/or ERR* go low, but provided that WAIT* is high.

OBSERVATION 2.15:
The active MASTER drives WR* valid during the address broadcast phase.

RULE 2.25:
During read cycles, a responding D32 SLAVE MUST retrieve data from byte locations as shown in the Table 2-15, and MUST drive them on AD00-AD31.

RULE 2.26:
During write cycles, D32 SLAVES MUST capture data from AD00-AD31 as shown in the Table 2-15, and MUST store it in the addressed byte locations.

REGLE 2.27:

Pendant les transferts de données en lecture dans lesquels le MAITRE actif est interverrouillé avec un ESCLAVE D32, le MAITRE actif DOIT prélever les données valides de AD00-AD31 comme indiqué au tableau 2-15.

Tableau 2-15

Utilisation de AD00-AD31 par un ESCLAVE D32 pour accéder aux emplacements d'octet

Type de cycle	Emplacements d'octet accédés	Lignes de données utilisées
OCTET(0) OCTET(1) OCTET(2) OCTET(3)	OCTET(0) OCTET(1) OCTET(2) OCTET(3)	AD31-AD24 AD23-AD16 AD15-AD08 AD07-AD00
OCTET(0-1) OCTET(1-2) OCTET(2-3) OCTET(3)(0)	OCTET(0-1) OCTET(1-2) OCTET(2-3) OCTET(3)	AD31-AD16 AD23-AD08 AD15-AD00 AD07-AD00
OCTET(0-2) OCTET(1-3) OCTET(2-3)(0) OCTET(3)(0-1)	OCTET(0-2) OCTET(1-3) OCTET(2-3) OCTET(3)	AD31-AD08 AD23-AD00 AD15-AD00 AD07-AD00
OCTET(0-3) OCTET(1-3)(0) OCTET(2-3)(0-1) OCTET(3)(0-2)	OCTET(0-3) OCTET(1-3) OCTET(2-3) OCTET(3)	AD31-AD00 AD23-AD00 AD15-AD00 AD07-AD00

REGLE 2.28:

Pendant les cycles de lecture, un ESCLAVE D16 répondant DOIT prélever les données des emplacements d'octet comme indiqué au tableau 2-16, et DOIT les placer sur AD16-AD31.

REGLE 2.29:

Pendant les cycles d'écriture, les ESCLAVES D16 DOIVENT prélever les données des lignes AD16-AD31 comme indiqué au tableau 2-16, et DOIVENT les ranger dans les emplacements d'octet adressés.

REGLE 2.30:

Pendant un transfert de données en lecture dans lequel le MAITRE actif est interverrouillé avec un ESCLAVE D16, le MAITRE actif DOIT prélever les données valides de AD16-AD31 comme indiqué au tableau 2-16.

RULE 2.27:
During read data transfers where the active MASTER is interlocked with a D32 SLAVE, the active MASTER MUST capture the valid data from AD00-AD31 as shown in Table 2-15.

Table 2-15

Use of AD00-AD31 by a D32 SLAVE
to access byte locations

Type of cycle	Byte locations accessed	Data lines used
BYTE(0) BYTE(1) BYTE(2) BYTE(3)	BYTE(0) BYTE(1) BYTE(2) BYTE(3)	AD31-AD24 AD23-AD16 AD15-AD08 AD07-AD00
BYTE(0-1) BYTE(1-2) BYTE(2-3) BYTE(3)(0)	BYTE(0-1) BYTE(1-2) BYTE(2-3) BYTE(3)	AD31-AD16 AD23-AD08 AD15-AD00 AD07-AD00
BYTE(0-2) BYTE(1-3) BYTE(2-3)(0) BYTE(3)(0-1)	BYTE(0-2) BYTE(1-3) BYTE(2-3) BYTE(3)	AD31-AD08 AD23-AD00 AD15-AD00 AD07-AD00
BYTE(0-3) BYTE(1-3)(0) BYTE(2-3)(0-1) BYTE(3)(0-2)	BYTE(0-3) BYTE(1-3) BYTE(2-3) BYTE(3)	AD31-AD00 AD23-AD00 AD15-AD00 AD07-AD00

RULE 2.28:
During read cycles, a responding D16 SLAVE MUST retrieve data from byte locations as shown in Table 2-16, and MUST drive it on AD16-AD31.

RULE 2.29:
During write cycles, D16 SLAVES MUST capture data from AD16-AD31 as shown in Table 2-16, and MUST store it in the addressed byte locations.

RULE 2.30:
During read data transfer where the active MASTER is interlocked with a D16 SLAVE, the active MASTER MUST capture the valid data from AD16-AD31 as shown in Table 2-16.

Tableau 2-16

Utilisation de AD16-AD31 par un ESCLAVE D16
pour accéder aux emplacements d'octet

Type de cycle	Emplacements d'octet accédés	Lignes de données utilisées
OCTET(0) OCTET(1) OCTET(2) OCTET(3)	OCTET(0) OCTET(1) OCTET(2) OCTET(3)	AD31-AD24 AD23-AD16 AD31-AD24 AD23-AD16
OCTET(0-1) OCTET(1-2) OCTET(2-3) OCTET(3)(0)	OCTET(0-1) OCTET(1) OCTET(2-3) OCTET(3)	AD31-AD16 AD23-AD16 AD31-AD16 AD23-AD16
OCTET(0-2) OCTET(1-3) OCTET(2-3)(0) OCTET(3)(0-1)	OCTET(0-1) OCTET(1) OCTET(2-3) OCTET(3)	AD31-AD16 AD23-AD16 AD31-AD16 AD23-AD16
OCTET(0-3) OCTET(1-3)(0) OCTET(2-3)(0-1) OCTET(3)(0-2)	OCTET(0-1) OCTET(1) OCTET(2-3) OCTET(3)	AD31-AD16 AD23-AD16 AD31-AD16 AD23-AD16

REGLE 2.31:
Pendant les cycles de lecture, un ESCLAVE D08 répondant DOIT prélever les données des emplacements d'octet comme indiqué au tableau 2-17, et DOIT les placer sur AD24-AD31.

REGLE 2.32:
Pendant les cycles d'écriture, les ESCLAVES D08 DOIVENT prélever les données des lignes AD24-AD31 comme indiqué au tableau 2-17, et DOIVENT les ranger dans les emplacements d'octet adressés.

REGLE 2.33:
Pendant un transfert de données en lecture dans lequel le MAITRE actif est interverrouillé avec un ESCLAVE D08, le MAITRE actif DOIT prélever les données valides de AD24-AD31 comme indiqué au tableau 2-17.

Tableau 2-17

Utilisation de AD24-AD31 par un ESCLAVE D08
pour accéder aux emplacements d'octet

Type de cycle	Emplacements d'octet accédés	Lignes de données utilisées
OCTET(0) OCTET(1) OCTET(2) OCTET(3)	OCTET(0) OCTET(1) OCTET(2) OCTET(3)	AD31-AD24 AD31-AD24 AD31-AD24 AD31-AD24
OCTET(0-1) OCTET(1-2) OCTET(2-3) OCTET(3)(0)	OCTET(0) OCTET(1) OCTET(2) OCTET(3)	AD31-AD24 AD31-AD24 AD31-AD24 AD31-AD24

(Suite à la page 122)

Table 2-16

Use of AD16-AD31 by a D16 SLAVE
to access byte locations

Type of cycle	Byte locations accessed	Data lines used
BYTE(0) BYTE(1) BYTE(2) BYTE(3)	BYTE(0) BYTE(1) BYTE(2) BYTE(3)	AD31-AD24 AD23-AD16 AD31-AD24 AD23-AD16
BYTE(0-1) BYTE(1-2) BYTE(2-3) BYTE(3)(0)	BYTE(0-1) BYTE(1) BYTE(2-3) BYTE(3)	AD31-AD16 AD23-AD16 AD31-AD16 AD23-AD16
BYTE(0-2) BYTE(1-3) BYTE(2-3)(0) BYTE(3)(0-1)	BYTE(0-1) BYTE(1) BYTE(2-3) BYTE(3)	AD31-AD16 AD23-AD16 AD31-AD16 AD23-AD16
BYTE(0-3) BYTE(1-3)(0) BYTE(2-3)(0-1) BYTE(3)(0-2)	BYTE(0-1) BYTE(1) BYTE(2-3) BYTE(3)	AD31-AD16 AD23-AD16 AD31-AD16 AD23-AD16

RULE 2.31:
During read cycles, a responding D08 SLAVE MUST retrieve data from byte locations as shown in Table 2-17, and MUST drive it on AD24-AD31.

RULE 2.32:
During write cycles, D08 SLAVES MUST capture data from AD24-AD31 as shown in Table 2-17, and MUST store it in the addressed byte locations.

RULE 2.33:
During read data transfer where the active MASTER is interlocked with a D08 SLAVE, the active MASTER MUST capture the valid data from AD24-AD31 as shown in Table 2-17.

Table 2-17

Use of AD24-AD31 by a D08 SLAVE
to access byte locations

Type of cycle	Byte locations accessed	Data lines used
BYTE(0) BYTE(1) BYTE(2) BYTE(3)	BYTE(0) BYTE(1) BYTE(2) BYTE(3)	AD31-AD24 AD31-AD24 AD31-AD24 AD31-AD24
BYTE(0-1) BYTE(1-2) BYTE(2-3) BYTE(3)(0)	BYTE(0) BYTE(1) BYTE(2) BYTE(3)	AD31-AD24 AD31-AD24 AD31-AD24 AD31-AD24

(Continued on page 123)

Tableau 2-17 (suite)

Type de cycle	Emplacements d'octet accédés	Lignes de données utilisées
OCTET(0-2) OCTET(1-3) OCTET(2-3)(0) OCTET(3)(0-1)	OCTET(0) OCTET(1) OCTET(2) OCTET(3)	AD31-AD24 AD31-AD24 AD31-AD24 AD31-AD24
OCTET(0-3) OCTET(1-3)(0) OCTET(2-3)(0-1) OCTET(3)(0-2)	OCTET(0) OCTET(1) OCTET(2) OCTET(3)	AD31-AD24 AD31-AD24 AD31-AD24 AD31-AD24

REGLE 2.34:
SI aucune erreur n'a été détectée pendant le cycle,
ALORS l'ESCLAVE répondant doit commander la ligne ACK* au niveau bas pendant la phase de transfert de données.

SUGGESTION 2.3:
Pour optimiser les performances, concevoir les ESCLAVES libérant ACK* et ERR* au niveau haut aussitôt que possible après avoir détecté DS* au niveau haut.

OBSERVATION 2.16:
Les ESCLAVES participants sont assurés que la phase de transfert de données reste valide jusqu'à ce qu'ils libèrent WAIT* au niveau haut.

SUGGESTION 2.4:
Pour optimiser les performances, concevoir les ESCLAVES participants libérant WAIT* au niveau haut aussitôt que possible.

AUTORISATION 2.2:
Les ESCLAVES ne pouvant pas participer aux cycles VSB PEUVENT être conçus pour ne pas commander la ligne WAIT* au niveau bas.

REGLE 2.35:
L'ESCLAVE répondant DOIT commander AC au niveau bas avant de commander ACK* et/ou ERR* au niveau bas lorsqu'il acquitte le transfert du dernier emplacement d'octet auquel il peut accéder sans avoir besoin d'une autre adresse du MAITRE.

REGLE 2.36:
Les ESCLAVES participants DOIVENT commander AC au niveau bas avant de libérer WAIT* au niveau haut pendant le transfert de données au dernier emplacement d'octet auquel ils peuvent accéder sans avoir besoin d'une autre adresse du MAITRE.

REGLE 2.37:
SI pendant la phase de transfert de données, le MAITRE actif détecte un niveau bas sur AC avant de détecter ACK* et/ou ERR* au niveau bas et WAIT* au niveau haut,
ALORS il DOIT finir le cycle.

Table 2-17 (continued)

Type of cycle	Byte locations accessed	Data lines used
BYTE(0-2) BYTE(1-3) BYTE(2-3)(0) BYTE(3)(0-1)	BYTE(0) BYTE(1) BYTE(2) BYTE(3)	AD31-AD24 AD31-AD24 AD31-AD24 AD31-AD24
BYTE(0-3) BYTE(1-3)(0) BYTE(2-3)(0-1) BYTE(3)(0-2)	BYTE(0) BYTE(1) BYTE(2) BYTE(3)	AD31-AD24 AD31-AD24 AD31-AD24 AD31-AD24

RULE 2.34:
IF no errors have been encountered during the cycle,
THEN the responding SLAVE MUST drive ACK* low during the data transfer phase.

SUGGESTION 2.3:
For optimum performance, design SLAVES to release ACK* and ERR* to high as soon as possible after detecting DS* high.

OBSERVATION 2.16:
Participating SLAVES are guaranteed that the data transfer phase remains valid until they release WAIT* to high.

SUGGESTION 2.4:
For optimum performance, design participating SLAVES to release WAIT* to high as soon as possible.

PERMISSION 2.2:
SLAVES that are not capable of participating in VSB cycles MAY be designed to not drive WAIT* low.

RULE 2.35:
The responding SLAVE MUST drive AC low before it drives ACK* and/or ERR* low when it acknowledges the transfer to the last byte location it can access without requiring another address from the MASTER.

RULE 2.36:
Participating SLAVES MUST drive AC low before they release WAIT* to high during the data transfer to the last byte location they can access without requiring another address from the MASTER.

RULE 2.37:
IF the active MASTER detects a low level on AC before it detects ACK* and/or ERR* low and WAIT* high during the data transfer phase,
THEN it MUST terminate the cycle.

OBSERVATION 2.17:
L'ESCLAVE A TRANSFERT UNIQUE répondant se comporte comme un ESCLAVE A TRANSFERT PAR BLOC qui ne pourrait effectuer qu'un seul transfert de données au cours du cycle. Il commande AC au niveau bas avant de répondre au premier transfert de données (c'est-à-dire il commande ACK* et/ou ERR* au niveau bas au début du cycle). Cela garantit la compatibilité entre les MAITRES qui possèdent la possibilité BLT et les ESCLAVES qui ne la possèdent pas.

2.5.3 *Interaction entre les MAITRES et les ESCLAVES pendant la fin du cycle*

Il existe une relation d'interverrouillage entre MAITRES et ESCLAVES pendant la fin d'un cycle VSB. Lorsque le MAITRE actif a terminé le cycle, tous les ESCLAVES VSB acquittent cette fin.

Le paragraphe 2.5.3.1 et la figure 2-13, page 126, décrivent le déroulement et l'interaction entre le MAITRE actif et les ESCLAVES VSB pendant la fin de cycle. Ce paragraphe fournit une description informelle du protocole afin de familiariser le lecteur avec le déroulement des opérations pendant la fin de cycle. La section 2.6 spécifie les contraintes temporelles.

2.5.3.1 *Déroulement de la fin d'un cycle*

La figure 2-13 décrit l'organigramme de la fin d'un cycle. Le MAITRE actif peut terminer un cycle VSB après avoir terminé:

a) la diffusion d'adresse, lors d'un cycle UNIQUEMENT D'ADRESSAGE, ou

b) un transfert de données au cours d'un cycle de TRANSFERT UNIQUE ou TRANSFERT PAR BLOC, ou

c) une phase de sélection au cours d'un cycle de RECONNAISSANCE D'INTERRUPTION, ou

d) un transfert de MOT D'ETAT/ID au cours d'un cycle de RECONNAISSANCE D'INTERRUPTION.

Il termine le cycle à l'étape 2 dans laquelle il autorise les lignes PAS*, SIZE0-SIZE1, SPACE0-SPACE1, WR* et LOCK* à passer au niveau haut. Si le MAITRE termine le cycle après l'exécution d'un transfert de données en écriture, alors il remet ses émetteurs de données à l'état haute impédance, permettant ainsi aux lignes AD00-AD31 de remonter au niveau haut.

A l'étape 3, l'ESCLAVE répondant détecte un niveau haut sur la ligne PAS* et libère ASACK0*-ASACK1* au niveau haut. S'il commande la ligne CACHE*, alors il libère la ligne CACHE* au niveau haut. Si le cycle se termine après l'exécution d'un transfert de données en lecture ou après l'exécution d'un transfert de MOT D'ETAT/ID, l'ESCLAVE répondant remet ses émetteurs de données à l'état haute impédance, permettant ainsi aux lignes AD00-AD31 de passer au niveau haut. De plus, tous les ESCLAVES doivent commander la ligne AC au niveau bas, tandis que les ESCLAVES conçus pour le faire commandent la ligne WAIT* au niveau bas. Cela termine le cycle.

OBSERVATION 2.17:
The responding SINGLE-TRANSFER SLAVE behaves like a BLOCK-TRANSFER SLAVE that can accommodate only one data transfer in the course of the cycle. It drives AC low before responding to the first data transfer (i.e. drives ACK* and/or ERR* low for the first time in the cycle). This guarantees the compatibility among MASTERS that include, and SLAVES that do not include, BLT capability.

2.5.3 *Interaction between MASTERS and SLAVES during cycle termination*

An interlocked relationship exists between MASTERS and SLAVES during the termination of a VSB cycle. Once the active MASTER has terminated the cycle, all VSB SLAVES acknowledge this termination.

Paragraph 2.5.3.1 and Figure 2-13, page 127, describe the flow and the interaction between the active MASTER and VSB SLAVES during cycle termination. This paragraph provides an informal description of the protocol to familiarize the reader with the flow during cycle termination. Section 2.6 specifies the timing requirements.

2.5.3.1 *Flow of the termination of a cycle*

Figure 2-13 describes the flow of the termination of a cycle. The active MASTER might terminate a VSB cycle after it has terminated:

a) the address broadcast, performing an ADDRESS-ONLY cycle, or

b) a data transfer in the course of either a SINGLE-TRANSFER or a BLOCK-TRANSFER cycle, or

c) a selection phase in the course of an INTERRUPT-ACKNOWLEDGE cycle, or

d) a STATUS/ID transfer in the course of an INTERRUPT-ACKNOWLEDGE cycle.

It terminates the cycle in step 2 where it allows PAS*, SIZE0-SIZE1, SPACE0-SPACE1, WR* and LOCK* to go high. If the MASTER terminates the cycle after executing a write data transfer, then it tri-states the data buffers, allowing AD00-AD31 to go high as well.

In step 3 the responding SLAVE detects a high level on PAS* and releases ASACK0*-ASACK1* to high. If it is driving CACHE*, then it releases CACHE* to high. If the cycle is terminated after executing a read data transfer, or after executing a STATUS/ID transfer, then the responding SLAVE tri-states its data buffers, allowing AD00-AD31 to go high as well. In addition, all SLAVES are required to drive AC low, while those SLAVES that are designed to do so, drive WAIT* low. This terminates the cycle.

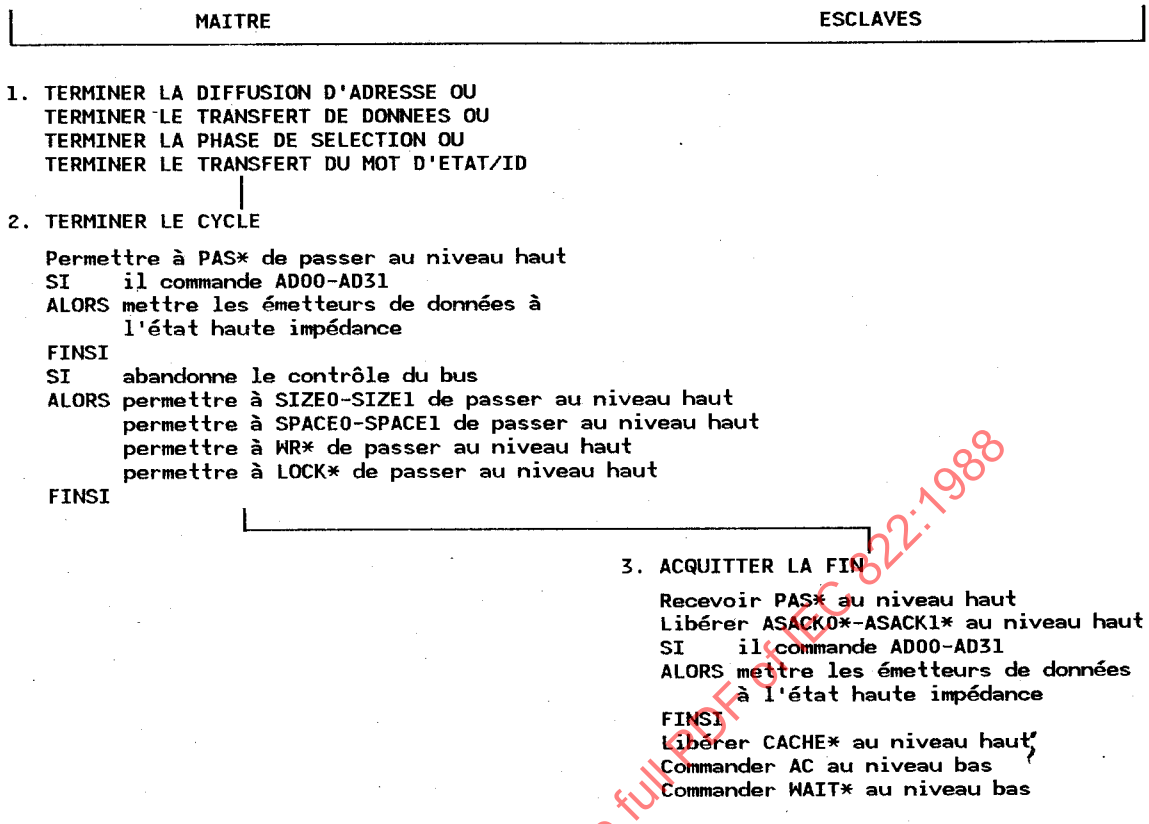


Fig. 2-13. - Organigramme de la fin du cycle.

2.5.4 *Interaction entre le MAITRE IHV et les ESCLAVES pendant le cycle de RECONNAISSANCE D'INTERRUPTION*

Le cycle de RECONNAISSANCE D'INTERRUPTION comporte trois phases. La première phase est la phase de sélection. Pendant cette phase, tous les ESCLAVES concurrents, c'est-à-dire ceux qui ont une demande d'interruption en attente, se disputent le bus pour déterminer lequel d'entre eux répondra au cycle. La deuxième phase, phase de transfert du MOT D'ETAT/ID, est déclenchée lorsque la phase de sélection est terminée. Pendant cette phase, l'ESCLAVE INTV répondant transfère au MAITRE IHV son information MOT D'ETAT/ID. Dans la troisième phase, le MAITRE IHV termine le cycle de RECONNAISSANCE D'INTERRUPTION.

Il existe une relation d'interverrouillage entre le MAITRE IHV et les ESCLAVES INTV pendant le cycle de RECONNAISSANCE D'INTERRUPTION. Sitôt que le MAITRE actif IHV a déclenché la phase de sélection, il lui faut maintenir tous les signaux utiles valides jusqu'à ce que tous les ESCLAVES concurrents signalent la fin de cette phase. Le MAITRE IHV commence alors la phase de transfert de l'information MOT D'ETAT/ID qu'il termine lorsque l'ESCLAVE répondant acquittera le transfert.

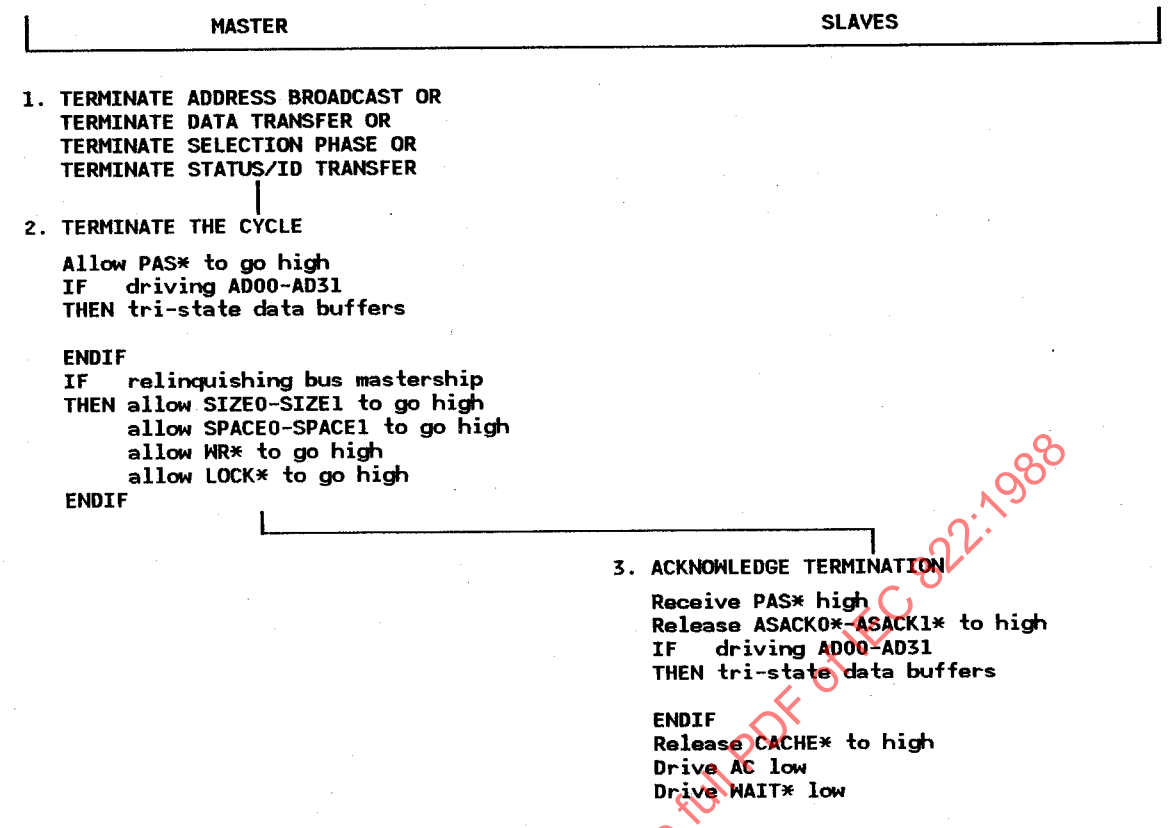


Fig. 2-13. - Flow of the termination of the cycle.

2.5.4 *Interaction between the IHV MASTER and SLAVES during the INTERRUPT-ACKNOWLEDGE cycle*

The INTERRUPT-ACKNOWLEDGE cycle is comprised of three phases. The first phase is the selection phase. During this phase all contending SLAVES, i.e. those SLAVES that have an interrupt pending, contend for the bus to determine which one will respond to the cycle. The second phase, the STATUS/ID transfer phase, is started after the selection phase is completed. During this phase, the responding INTV SLAVE transfers to the IHV MASTER its STATUS/ID information. In the third part, the IHV MASTER terminates the INTERRUPT-ACKNOWLEDGE cycle.

An interlocked relationship exists between the IHV MASTER and INTV SLAVES during the INTERRUPT-ACKNOWLEDGE cycle. Once the active IHV MASTER has started the selection phase, it is required to maintain all the relevant signal lines valid until all contending SLAVES signal its completion. The IHV MASTER then start the STATUS/ID transfer phase, terminating it after the responding SLAVE acknowledges the transfer.

La figure 2-14, page 132, du paragraphe 2.5.4.1 montre l'organigramme d'un cycle de RECONNAISSANCE D'INTERRUPTION et l'interaction du MAITRE IHV et des ESCLAVES pendant le cycle. Ce paragraphe fournit une description informelle du protocole afin de familiariser le lecteur avec le déroulement du cycle de RECONNAISSANCE D'INTERRUPTION. Le paragraphe 2.5.4.2 constitue la spécification formelle de ce protocole, tandis que la section 2.6 spécifie les contraintes temporelles.

2.5.4.1 *Organigramme d'un cycle de RECONNAISSANCE D'INTERRUPTION*

L'organigramme d'un cycle de RECONNAISSANCE D'INTERRUPTION est montré dans la figure 2-14. La séquence d'interruption débute à l'étape 1 lorsqu'un ESCLAVE INTV commande la ligne IRQ* au niveau bas.

A l'étape 2, le MAITRE IHV reçoit IRQ* au niveau bas et s'il n'est pas déjà MAITRE actif, il acquiert le contrôle du bus comme défini au chapitre 3.

A l'étape 3, après avoir obtenu l'autorisation d'utiliser le bus, le MAITRE actif IHV déclenche la phase de sélection. Il s'assure d'abord que le MAITRE précédent a cessé de commander le bus en vérifiant que PAS* est au niveau haut. Puis, il commande SIZE0-SIZE1 pour demander la dimension du transfert de MOT D'ETAT/ID. Il commande ensemble les deux lignes SPACE0 et SPACE1 au niveau bas et la ligne WR* au niveau haut pour indiquer aux ESCLAVES qu'un cycle de RECONNAISSANCE D'INTERRUPTION est en cours. Après avoir détecté que ASACK0*-ASACK1* est au niveau haut ce qui signifie que l'ESCLAVE qui répondait au cycle précédent n'est plus sur le bus, le MAITRE actif IHV commande PAS* au niveau bas.

A l'étape 4, après la détection d'un front descendant sur la ligne PAS*, les ESCLAVES reçoivent SPACE0-SPACE1 au niveau bas et WR* au niveau haut et déduisent qu'un cycle de RECONNAISSANCE D'INTERRUPTION est en cours. A ce moment-là, les ESCLAVES qui n'ont pas les possibilités INTV ou ne commandent pas IRQ* au niveau bas libèrent à la fois AC et WAIT* au niveau haut. Tous les ESCLAVES concurrents, c'est-à-dire, les ESCLAVES INTV qui ont commandé IRQ* au niveau bas commandent leur ID INTERRUPTION sur les lignes AD24-AD30 puis libèrent la ligne WAIT* au niveau haut.

A l'étape 5, les ESCLAVES INTV concurrents négocient pour sélectionner celui qui répondra au cycle en fournissant son information MOT D'ETAT/ID. Après avoir permis à leur logique de sélection de s'établir comme décrit dans l'annexe A, les ESCLAVES concurrents placent leur code de dimension sur les lignes ASACK0*-ASACK1* et retirent leur contribution à la ligne AC pour lui permettre de passer au niveau haut. Alors, l'ID INTERRUPTION placé sur les lignes AD24-AD30 (désigné par BUS-INT-ID) sera égal à l'ID INTERRUPTION de l'ESCLAVE INTV dont la priorité est la plus élevée.

A l'étape 6, après réception d'un niveau haut sur AC, les ESCLAVES concurrents comparent l'ID INTERRUPTION présent sur le bus à leur propre ID INTERRUPTION. L'ESCLAVE dont l'identificateur correspond aux niveaux sur AD24-AD30 devient l'ESCLAVE répondant. De plus, à ce moment, le code de dimension présent sur les lignes ASACK0*-ASACK1* est celui de l'ESCLAVE répondant. Tous les ESCLAVES

Paragraph 2.5.4.1, Figure 2-14, page 133, shows the flow of the INTERRUPT-ACKNOWLEDGE cycle, and how the IHV MASTER and SLAVES interact during the cycle. This paragraph provides an informal description of the protocol, to familiarize the reader with the flow of the INTERRUPT-ACKNOWLEDGE cycle. Paragraph 2.5.4.2 constitutes a formal specification of this protocol, while Section 2.6 specifies the timing requirements.

2.5.4.1 *Flow of an INTERRUPT-ACKNOWLEDGE cycle*

The flow of an INTERRUPT-ACKNOWLEDGE cycle is shown in Figure 2-14. The interrupt sequence begins in step 1 when an INTV SLAVE drives IRQ* low.

In step 2, the IHV MASTER receives IRQ* driven low and, if it is not already the active MASTER, acquires bus mastership as defined in Chapter 3.

In step 3, after it has been granted the bus, the active IHV MASTER initiates the selection phase. It first ensures that the previous MASTER has stopped driving the bus by verifying that PAS* is high. It then drives SIZE0-SIZE1 to request the size of the STATUS/ID transfer. It drives both SPACE0 and SPACE1 to low and WR* to high to broadcast to the SLAVES that an INTERRUPT-ACKNOWLEDGE cycle is in progress. After detecting ASACK0*-ASACK1* high, signifying that the SLAVE that responded to the previous cycle is no longer on the bus, the active IHV MASTER drives PAS* to low.

In step 4, after detecting a falling edge on PAS*, the SLAVES receive SPACE0-SPACE1 low and WR* high and determine that an INTERRUPT-ACKNOWLEDGE cycle is in progress. At this time, SLAVES that either do not include INTV capability, or are not driving IRQ* low, release both AC and WAIT* to high. All contending SLAVES, i.e. those INTV SLAVES that are driving IRQ* low, drive their INTERRUPT ID on AD24-AD30, and then release WAIT* to high.

In step 5, the contending INTV SLAVES negotiate to select the one which will respond to the cycle by supplying a STATUS/ID. After allowing their selection logic to settle, as described in Appendix A, contending SLAVES drive their size code on ASACK0*-ASACK1* and release their contribution to the AC line in order to let it to high. By then, the INTERRUPT-ID carried on AD24-AD30 (referred to as BUS-INT-ID) will be equal to the INTERRUPT-ID of the INTV SLAVE whose interrupt priority is the highest.

In step 6, after receiving a high level on AC, the contending SLAVES compare the INTERRUPT-ID carried on the bus to their own INTERRUPT-ID. The one SLAVE whose ID matches the level of AD24-AD30 becomes the responding SLAVE. In addition, at this time the size code carried on ASACK0*-ASACK1* is that of the responding SLAVE.

concurrents doivent alors libérer les lignes AD24-AD30 au niveau haut avant un délai maximal prescrit après avoir retiré leur contribution au maintien de la ligne AC pour lui permettre de passer au niveau haut.

A l'étape 7, le MAITRE IHV déclenche la phase de transfert du MOT D'ETAT/ID. Cependant, si le MAITRE IHV détermine que l'interruption était demandée par des ESCLAVES INTP, il termine le cycle de RECONNAISSANCE D'INTERRUPTION. Cette condition est déterminée par un niveau haut sur AC et sur ASACK0* et ASACK1*. Dans ce cas, le MAITRE déclenche des cycles de lecture des registres MOT D'ETAT/ID des ESCLAVES du système. Un niveau haut sur AC et un niveau bas sur ASACK0* ou sur ASACK1* indiquent que la phase de sélection du générateur d'interruption est terminée et qu'un ESCLAVE répondant INTV a été sélectionné. Dans ce cas, après avoir reçu AC au niveau haut, le MAITRE IHV attend un minimum de temps prescrit puis commande DS* au niveau bas pour déclencher la phase de transfert du MOT D'ETAT/ID du cycle de RECONNAISSANCE D'INTERRUPTION.

A l'étape 8, l'ESCLAVE INTV répondant transfère son information de MOT D'ETAT/ID au MAITRE IHV. Après réception de DS* au niveau bas, il place le MOT D'ETAT/ID sur une partie ou toutes les lignes AD00-AD31. Les lignes de données qu'il commande dépendent de ses possibilités de transfert de MOT D'ETAT/ID et de la dimension de MOT D'ETAT/ID requise par le MAITRE IHV. En plus, l'ESCLAVE INTV répondant retire sa contribution au maintien de la ligne IRQ* pour lui permettre de passer au niveau haut. Si l'ESCLAVE répondant a transféré tous ses MOTS D'ETAT/ID disponibles, il commande AC au niveau bas. Au contraire, s'il a encore de l'information de MOT D'ETAT/ID disponible, il maintient un niveau haut sur AC. L'ESCLAVE commande alors ACK* au niveau bas.

Par exemple, le MAITRE IHV demande un MOT D'ETAT/ID octet unique (c'est-à-dire avec SIZE0 au niveau bas et SIZE1 au niveau haut), mais l'ESCLAVE INTV répondant a quatre octets de MOT D'ETAT/ID. Dans ce cas, l'ESCLAVE maintiendra un niveau haut sur AC, signalant ainsi au MAITRE qu'il a encore des informations de MOT D'ETAT/ID disponibles.

A l'étape 9, après avoir détecté ACK* au niveau bas, le MAITRE IHV prélève le MOT D'ETAT/ID sur les lignes de données. Il termine alors la phase de transfert du MOT D'ETAT/ID en permettant à DS* de passer au niveau haut.

A l'étape 10, l'ESCLAVE INTV répondant reçoit DS* au niveau haut et relâche ACK* au niveau haut. Cela termine la phase de transfert du MOT D'ETAT/ID.

A l'étape 11, le MAITRE IHV peut déclencher un autre transfert de MOT D'ETAT/ID ou il peut terminer le cycle de RECONNAISSANCE D'INTERRUPTION comme décrit au paragraphe 2.5.3. Si le MAITRE IHV inclut des possibilités de transfert par bloc et si la ligne AC n'est pas commandée au niveau bas par l'ESCLAVE INTV répondant, le MAITRE déclenche un autre transfert de MOT D'ETAT/ID. Il procède ainsi en exécutant l'étape 7. Si le MAITRE IHV n'inclut pas de possibilités du type BLT ou s'il détecte la ligne AC commandée au niveau bas par l'ESCLAVE INTV répondant, il termine le cycle de RECONNAISSANCE D'INTERRUPTION.

All contending SLAVES are then required to release AD24-AD30 to high within a maximum prescribed time after releasing their contribution to the AC line to high.

In step 7, the IHV MASTER starts the STATUS/ID transfer phase. However, if the IHV MASTER determines that the interrupt was requested by INTP SLAVES, it terminates the INTERRUPT-ACKNOWLEDGE cycle. This condition is determined by a high level on AC as well as on ASACK0* and ASACK1*. When this is the case, the MASTER initiates read cycles to the STATUS/ID registers of the system's SLAVES. A high level on AC and a low level on either ASACK0* or on ASACK1* signify that the interrupter selection phase is complete, and that a responding INTV SLAVE has been selected. In this case, after receiving AC high, the IHV MASTER waits a minimum prescribed time and then drives DS* to low to initiate the STATUS/ID transfer phase of the INTERRUPT-ACKNOWLEDGE cycle.

In step 8, the responding INTV SLAVE transfers its STATUS/ID information to the IHV MASTER. After it receives DS* low it drives the STATUS/ID on some or all of AD00-AD31. The data lines that it drives depend on its STATUS/ID transfer capabilities, and on the STATUS/ID size that was requested by the IHV MASTER. In addition, the responding INTV SLAVE releases its contribution to the IRQ* line in order to let it to high. If the responding SLAVE has transferred all of its available STATUS/ID, then it drives AC to low. On the other hand, if it has more STATUS/ID information available, then it maintains a high level on AC. The SLAVE then drives ACK* to low.

For example, the IHV MASTER requests a Single-Byte STATUS/ID (i.e. drives SIZE0 to low and SIZE1 to high), but the responding INTV SLAVE has four bytes of STATUS/ID. In such a case, the SLAVE will maintain a high level on AC, signaling to the MASTER that more STATUS/ID information is available.

In step 9, after detecting a low on ACK*, the IHV MASTER captures the STATUS/ID from the data lines. It then terminates the STATUS/ID transfer phase by allowing DS* to go high.

In step 10, the responding INTV SLAVE receives DS* high and releases ACK* to high. This terminates the STATUS/ID transfer phase.

In step 11, the IHV MASTER might initiate another STATUS/ID transfer, or it might terminate the INTERRUPT-ACKNOWLEDGE cycle as described in Paragraph 2.5.3. If the IHV MASTER includes block transfer capabilities, and if AC is not driven low by the responding INTV SLAVE, then the MASTER initiates another STATUS/ID transfer. It does so by executing step 7. If the IHV MASTER does not include BLT capabilities, or if it detects AC driven low by the responding INTV SLAVE, then it terminates the INTERRUPT-ACKNOWLEDGE cycle.

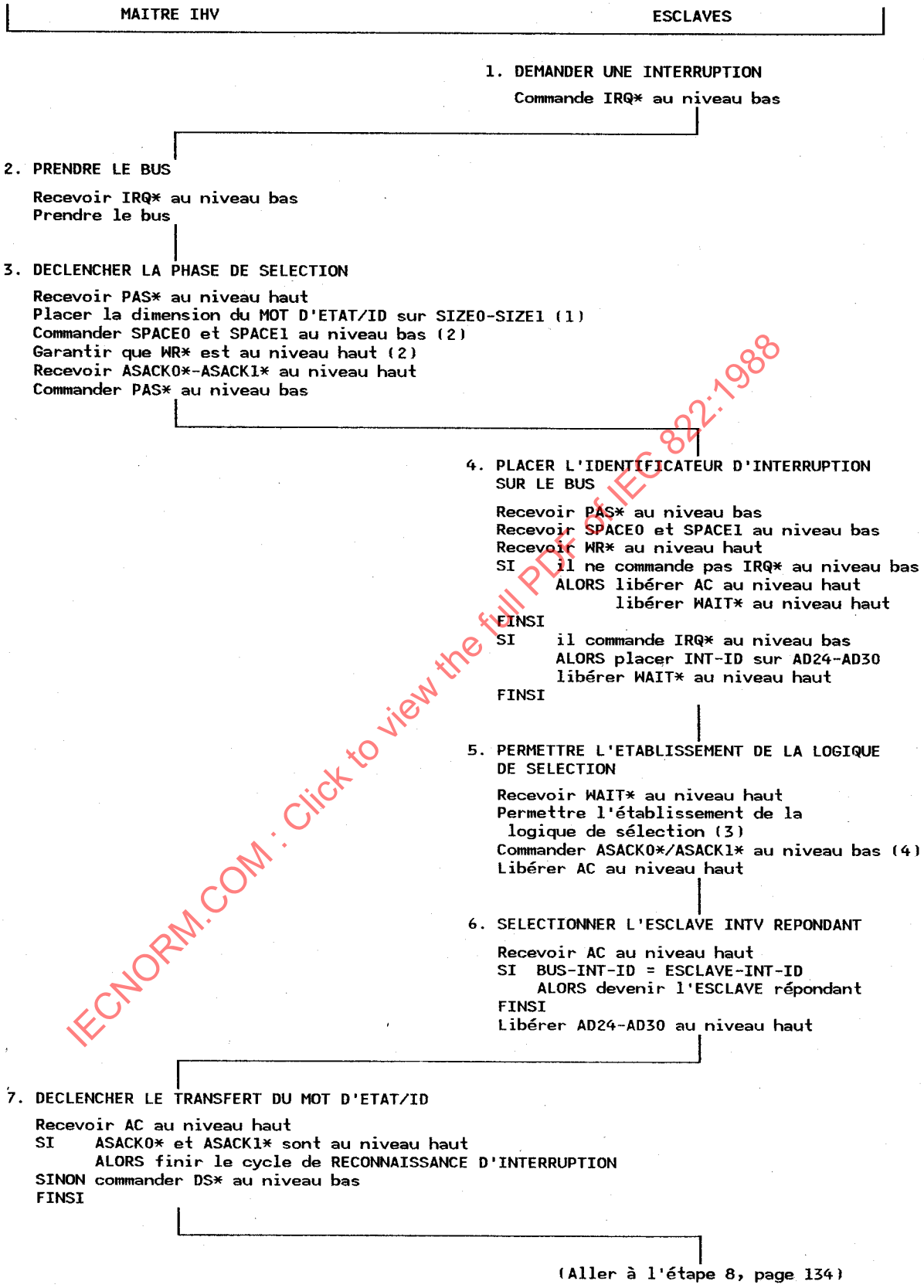


Fig. 2-14. - Organigramme d'un cycle de RECONNAISSANCE D'INTERRUPTION.

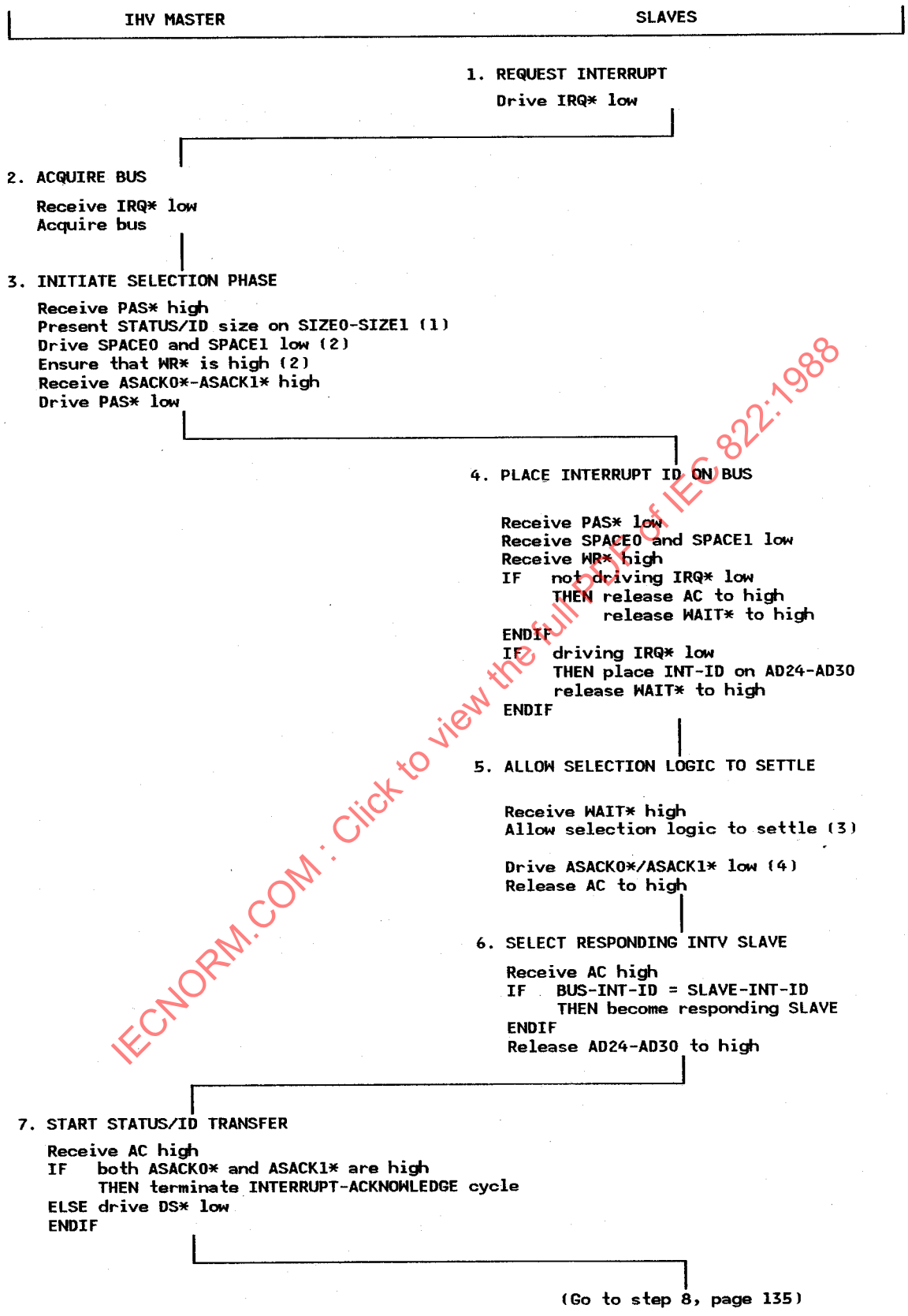
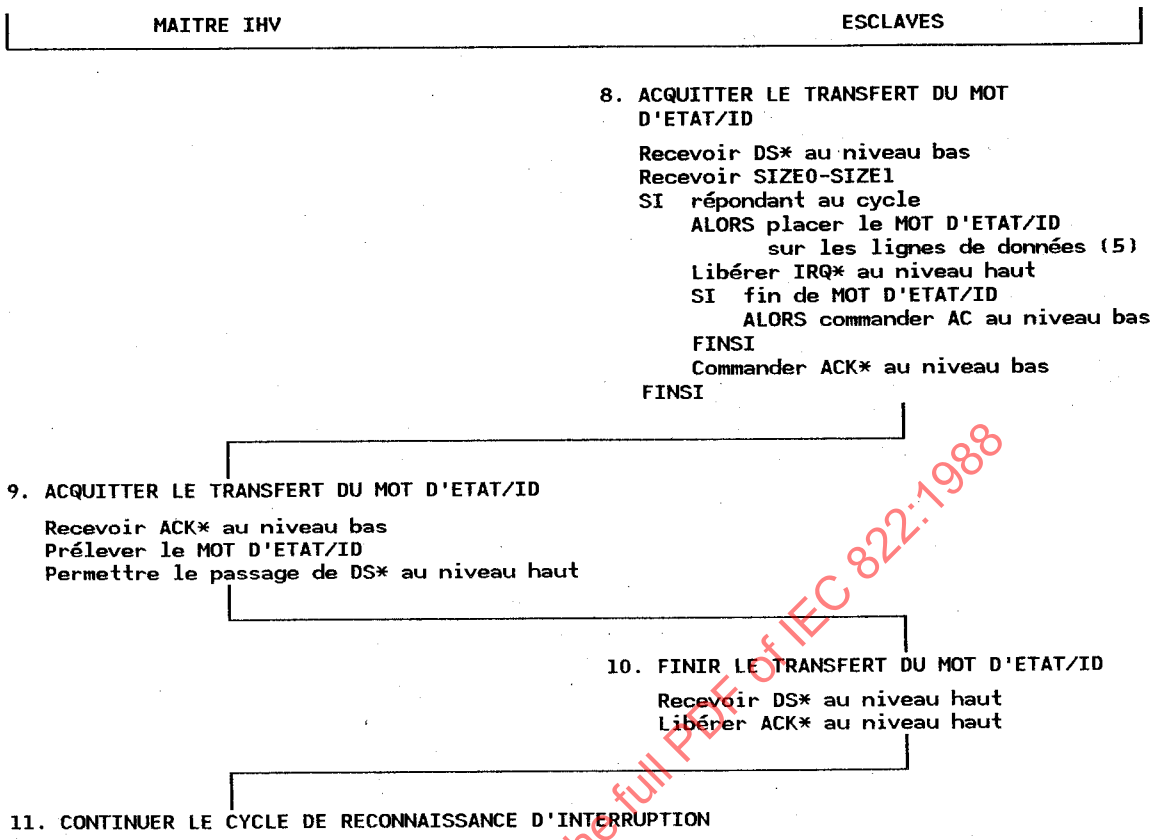


Fig. 2-14. - Flow of an INTERRUPT-ACKNOWLEDGE cycle.



Notes:

1.- A l'étape 3, le MAITRE IHV commande SIZE0-SIZE1 pour demander un transfert de MOT D'ETAT/ID octet unique, double ou quadruple octet, comme défini au paragraphe 2.5.1.2, tableau 2-10.

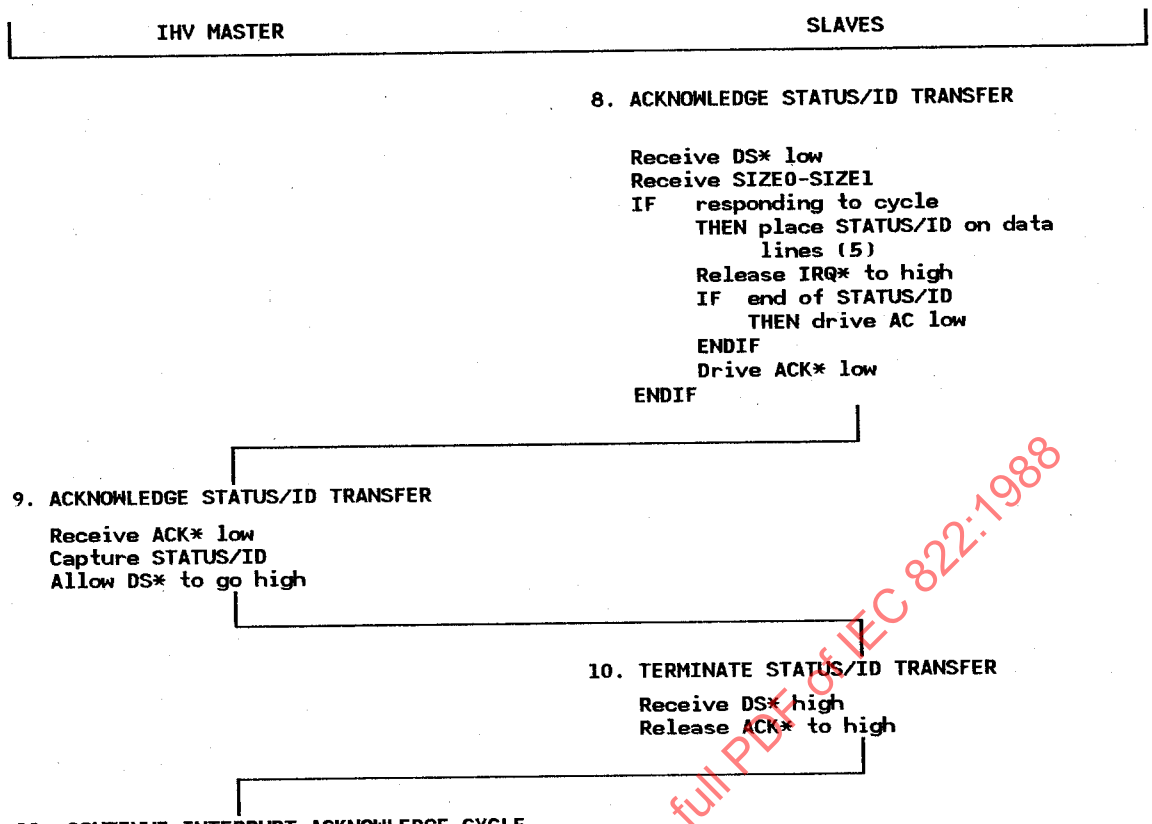
2.- A l'étape 3, le MAITRE IHV commande les deux lignes SPACE0 et SPACE1 au niveau bas et WR* au niveau haut pour informer les ESCLAVES qu'un cycle de RECONNAISSANCE D'INTERRUPTION est en cours, comme défini au paragraphe 2.5.4.2, tableau 2-18.

3.- A l'étape 5, les ESCLAVES concurrents permettent à la logique de sélection d'accomplir son processus. Les caractéristiques de cette logique de sélection sont décrits dans l'annexe A.

4.- A l'étape 5, les ESCLAVES INTV commandent ASACK0*-ASACK1* pour indiquer leur dimension au MAITRE IHV, comme défini au paragraphe 2.5.1.2, tableau 2-13.

5.- A l'étape 8, les lignes de données commandées par l'ESCLAVE répondant pour émettre son MOT D'ETAT/ID dépendent de sa dimension et du type de cycle de RECONNAISSANCE D'INTERRUPTION auquel il répond, comme indiqué au paragraphe 2.5.4.2, tableau 2-19.

FIGURE 2-14 (fin).



Notes:

1.- In step 3, the IHV MASTER drives SIZE0-SIZE1 to request a Single-Byte, Double-Byte, or Quad-Byte STATUS/ID transfer, as defined in Paragraph 2.5.1.2, Table 2-10.

2.- In step 3, the IHV MASTER drives both SPACE0 and SPACE1 low and WR* to high to inform SLAVES that an INTERRUPT-ACKNOWLEDGE cycle is in progress, as defined in Paragraph 2.5.4.2, Table 2-18.

3.- In step 5, the contending SLAVES allow time for the selection process to be completed by their selection logic. The attributes of this selection logic are described in Appendix A.

4.- In step 5, INTV SLAVES drive ASACK0*-ASACK1* to inform the IHV MASTER of their size, as defined in Paragraph 2.5.1.2, Table 2-13.

5.- In step 8, the data lines that the responding SLAVE drives with its STATUS/ID depend on its size and on the type of INTERRUPT-ACKNOWLEDGE cycle it responds to, as shown in Paragraph 2.5.4.2, Table 2-19.

FIGURE 2-14 (concluded).

2.5.4.2 *Evolution des signaux pendant le cycle de RECONNAISSANCE D'INTERRUPTION*

AUTORISATION 2.3:
Les ESCLAVES VSB PEUVENT commander IRQ* au niveau bas à n'importe quel moment après la séquence de remise à zéro par le système.

REGLE 2.38:
Le MAITRE IHV actif DOIT placer le code d'espace sur les lignes SPACE0-SPACE1 et commander la ligne WR* pour sélectionner un cycle de RECONNAISSANCE D'INTERRUPTION comme indique le tableau 2-18.

REGLE 2.39:
Les ESCLAVES VSB DOIVENT décoder le code d'espace sur SPACE0-SPACE1 et la ligne WR* pour déterminer qu'un cycle de RECONNAISSANCE D'INTERRUPTION est en cours comme indique le tableau 2-18.

Tableau 2-18

Utilisation de SPACE0, SPACE1 et WR* pour sélectionner un cycle de RECONNAISSANCE D'INTERRUPTION

Type de cycle	SPACE1	SPACE0	WR*
Cycle de RECONNAISSANCE D'INTERRUPTION	bas	bas	haut

REGLE 2.40:
SI un ESCLAVE INTV commande la ligne IRQ* au niveau bas et détecte un cycle de RECONNAISSANCE D'INTERRUPTION,
ALORS il DOIT placer un ID INTERRUPTION sur les lignes AD24-AD30 et son code de dimension sur ASACK0*-ASACK1*.

REGLE 2.41:
L'ID INTERRUPTION placé par un ESCLAVE concurrent sur AD24-AD30 DOIT inclure les bits d'adressage géographique comme suit: GA0 sur AD24, GA1 sur AD25 et GA2 sur AD26.

OBSERVATION 2.18:
Les bits d'adressage géographique (décrits au chapitre 5) assurent que chaque carte a un code ID INTERRUPTION unique qui conduira à une sélection correcte et sans ambiguïté d'un seul des ESCLAVES INTV concurrents pour répondre au cycle de RECONNAISSANCE D'INTERRUPTION.

AUTORISATION 2.4:
L'ID INTERRUPTION placé par un ESCLAVE concurrent PEUT inclure des bits définis par l'utilisateur sur les lignes AD27-AD30.

2.5.4.2 *Signaling during the INTERRUPT-ACKNOWLEDGE cycle*

PERMISSION 2.3:
VSB SLAVES MAY drive IRQ* low any time after the system has completed its reset sequence.

RULE 2.38:
The active IHV MASTER MUST drive the space code on SPACE0-SPACE1 and the WR* line to select an INTERRUPT-ACKNOWLEDGE cycle as shown in Table 2-18.

RULE 2.39:
VSB SLAVES MUST decode the space code on SPACE0-SPACE1 and the WR* line to determine that an INTERRUPT-ACKNOWLEDGE cycle is in progress, as shown in Table 2-18.

Table 2-18

Use of SPACE0, SPACE1 and WR* to select an INTERRUPT-ACKNOWLEDGE cycle

Type of cycle	SPACE1	SPACE0	WR*
INTERRUPT-ACKNOWLEDGE cycle	low	low	high

RULE 2.40:
IF an INTV SLAVE is driving the IRQ* line low and detects an INTERRUPT-ACKNOWLEDGE cycle,
THEN it MUST drive an INTERRUPT ID on AD24-AD30, and its size code on ASACK0*-ASACK1*.

RULE 2.41:
The INTERRUPT ID that a contending SLAVE drives on AD24-AD30 MUST include the geographical addressing bits as follows: GA0 on AD24, GA1 on AD25 and GA2 on AD26.

OBSERVATION 2.18:
The geographical addressing bits (described in Chapter 5) ensure that each board has a unique INTERRUPT ID code that will result in a positive and unambiguous selection of only one of the contending INTV SLAVES to respond to the INTERRUPT-ACKNOWLEDGE cycle.

PERMISSION 2.4:
The INTERRUPT ID that a contending SLAVE drives MAY include user defined and supplied bits on AD27-AD30.

REGLE 2.42:
SI l'ID INTERRUPTION d'un ESCLAVE concurrent ne correspond pas aux niveaux reçus sur les lignes AD24-AD30 du bus,
ALORS il NE DOIT PAS répondre à la phase de transfert de MOT D'ETAT/ID du cycle.

REGLE 2.43:
Pendant la phase de transfert du MOT D'ETAT/ID, les ESCLAVES INTV répondants du type D08, D16 et D32 DOIVENT placer leur MOT D'ETAT/ID sur les lignes de données, comme défini au tableau 2-19.

Tableau 2-19

Utilisation des lignes de données
par les ESCLAVES INTV D08, D16 et D32
pendant les cycles de RECONNAISSANCE D'INTERRUPTION

Type d'ESCLAVE INTV répondant	Type de cycle de RECONNAISSANCE D'INTERRUPTION	Lignes de données commandées avec le MOT D'ETAT/ID
ESCLAVE INTV D08	Tous	AD31-AD24
ESCLAVE INTV D16	Octet unique Double octet Quadruple octet	AD23-AD16 AD31-AD16 AD31-AD16
ESCLAVE INTV D32	Octet unique Double octet Quadruple octet	AD07-AD00 AD15-AD00 AD31-AD00

2.6 *Spécifications de chronologie du bus de transfert de données*

Cette section énonce les REGLES et OBSERVATIONS de chronologie régissant le comportement des MAITRES et des ESCLAVES. Cette information de chronologie se présente sous la forme de figures et tableaux.

En vue de respecter les REGLES de chronologie spécifiées, les concepteurs de cartes auront à prendre en compte les délais de propagation des émetteurs et récepteurs de bus utilisés sur leurs cartes VSB et ce pour le cas le plus défavorable. Le délai de propagation des émetteurs dépend de leur charge en sortie, mais les spécifications des fabricants ne contiennent pas toujours suffisamment d'informations pour calculer les délais de propagation sous diverses charges. Des suggestions sont proposées au chapitre 4 pour aider le concepteur de carte VSB.

Les OBSERVATIONS spécifient la chronologie des transitions de signaux sur les lignes. Ces temps peuvent être considérés comme sûrs tant que les REGLES de charge du fond de panier du chapitre 4 ne sont pas violées. Les REGLES pour les adaptateurs de bus, chapitre 4, garantissent que le protocole et les paramètres de temps continuent à être respectés après la libération des lignes de signaux.

RULE 2.42
IF the INTERRUPT ID of a contending SLAVE does not match the levels of AD24-AD30 as received from the bus,
THEN it MUST NOT respond to the STATUS/ID transfer phase of the cycle.

RULE 2.43:
During the STATUS/ID transfer phase, responding D08, D16 and D32 INTV SLAVES MUST drive their STATUS/ID on the data lines, as defined in Table 2-19.

Table 2-19

Use of the data lines by D08 D16 and D32 INTV SLAVES during INTERRUPT-ACKNOWLEDGE cycles

Type of responding INTV SLAVE	Type of INTERRUPT-ACKNOWLEDGE cycle	Data lines driven with STATUS/ID
D08 INTV SLAVE	All	AD31-AD24
D16 INTV SLAVE	Single-Byte Double-Byte Quad-Byte	AD23-AD16 AD31-AD16 AD31-AD16
D32 INTV SLAVE	Single-Byte Double-Byte Quad-Byte	AD07-AD00 AD15-AD00 AD31-AD00

2.6 *Data transfer bus timing specifications*

This section describes the timing RULES and OBSERVATIONS that govern the behavior of MASTERS and SLAVES. This timing information is in the form of figures and tables.

In order to meet the specified timing RULES, board designers need to take into account the worst case propagation delays of the bus drivers and receivers used on their VSB boards. The propagation delay of the drivers depends on their output loads, and manufacturers specifications do not always give enough information to calculate the propagation delays under various loads. To help the VSB board designer, some suggestions are offered in Chapter 4.

The OBSERVATIONS specify the timing of incoming signal line transitions. These times can be relied upon as long as the backplane loading RULES in Chapter 4 are not violated. The RULES for the bus terminators in Chapter 4 guarantee that the protocol and timing parameters continue to be met after signal lines are released.

Habituellement, on trouve pour chaque REGLE de chronologie une OBSERVATION correspondante. Cependant, le temps qui est garanti dans l'OBSERVATION peut différer du temps spécifié dans la REGLE. Par exemple, une inspection minutieuse des chronogrammes montre que le MAITRE doit garantir un temps d'établissement de données et d'adresse de 10 ns alors que l'ESCLAVE est assuré seulement de 0 ns. Cela est dû aux émetteurs de bus de données et d'adresse qui ne sont pas toujours capables de commander les lignes de signaux du fond de panier à travers la région de seuil, du niveau bas au niveau haut avant que la transition ne se propage à l'extrémité du fond de panier et ne se réfléchisse. Le front descendant des signaux de validation d'adresse et de données franchit cependant normalement le seuil de 0,8 V, sans attendre la réflexion du signal. Le temps d'établissement garanti qui en résulte pour l'ESCLAVE est le temps d'établissement du MAITRE moins deux fois le temps de propagation du bus.

Typically, for each timing RULE there is a corresponding OBSERVATION. However, the time that is guaranteed in the OBSERVATION might differ from the time specified by the RULE. For example, a careful inspection of the timing diagrams shows that the MASTER is required to provide 10 ns of address and data set-up time, but the SLAVE is only guaranteed 0 ns. This is because the address and data bus drivers are not always able to drive the backplane's signal lines completely through the threshold region from low to high until the transition propagates to the end of the backplane and is reflected back. The falling edge of the address and data strobes, however, typically crosses the 0.8-V threshold without waiting for a reflection. The resulting guaranteed set-up time at the SLAVE is the MASTER'S set-up time less two bus propagation times.

Tableau 2-20

Paramètres de temps d'un MAITRE actif, d'un ESCLAVE répondant, d'un ESCLAVE participant et d'un ESCLAVE au repos

NUMERO DE PARAMETRE	MAITRE		ESCLAVE REPONDANT		ESCLAVE PARTICIPANT		ESCLAVE AU REPOS	
	MIN.	MAX.	MIN.	MAX.	MIN.	MAX.	MIN.	MAX.
1	0							
2	10		0		0		0	
3	30		20		20		20	
4	0		0					
5		20		10				
6	0		0					
7	0		0		0		0	
8	10		20					
9	0		0		0		0	
10	0		0					
11	0		0		0		0	
12	0		10					
13	0		0					
14	10		0		0			
15	10		0		0			
16	0		0					
17	0		0					
18A	0		10		0			
18B	0		10		0			
21	0		0					
22		30		20		30		
23	0		10		0			
24	0		0					
25	0		0		0		0	
26	20		10		10		10	
27	0		0		0		0	
28	10		0		0			
29		20						
30	10		0		0		0	
31	0		0		0		0	
32		30		20		20		20
33	0		10		0			
34		30		20		20		20
35		30		20		20		
36	0		0					
37		30		20		20		
38	10		0		0			
39	0		10		0			
40	30		20		20			
41	0		0					
42	0		0					
43	0		0		0			
44	30		20		20		20	
45	30		30					
46	40		40					
47		20						
48		20						
49	30							

Notes:

1.- Les spécifications des paramètres de temps sont données aux tableaux 2-22 et 2-23. Utiliser les numéros de paramètres de temps pour localiser en séquence la spécification correspondante.

2.- Tous les temps sont en nanosecondes.

Table 2-20

Active MASTER, responding SLAVE, participating SLAVE and idle SLAVE timing parameters

PARAMETER NUMBER	MASTER		RESPONDING SLAVE		PARTICIPATING SLAVE		IDLE SLAVE	
	MIN.	MAX.	MIN.	MAX.	MIN.	MAX.	MIN.	MAX.
1	0							
2	10		0		0		0	
3	30		20		20		20	
4	0		0					
5		20		10				
6	0		0					
7	0		0		0		0	
8	10		20					
9	0		0		0		0	
10	0		0					
11	0		0		0		0	
12	0		10					
13	0		0					
14	10		0		0			
15	10		0		0			
16	0		0					
17	0		0					
18A	0		10		0			
18B	0		10		0			
21	0		0					
22		30		20		30		
23	0		10		0			
24	0		0					
25	0		0		0		0	
26	20		10		10		10	
27	0		0		0		0	
28	10		0		0			
29		20						
30	10		0		0		0	
31	0		0		0		0	
32		30		20		20		20
33	0		10		0			
34		30		20		20		20
35		30		20		20		
36	0		0					
37		30		20		20		
38	10		0		0			
39	0		10		0			
40	30		20		20			
41	0		0					
42	0		0					
43	0		0		0			
44	30		20		20		20	
45	30		30					
46	40		40					
47		20						
48		20						
49	30							

Notes:

1.- The specifications for the timing parameters are given in Tables 2-22 and 2-23. Use the timing parameter number to sequentially locate the associated specification.

2.- All times are in nanoseconds.

Tableau 2-21

Paramètres de temps d'un MAITRE IHV, d'un ESCLAVE INTV répondant, d'un ESCLAVE INTV concurrent et d'un ESCLAVE au repos

NUMERO DE PARAMETRE	MAITRE IHV		ESCLAVE INTV REPONDANT		ESCLAVE INTV CONCURRENT		ESCLAVE AU REPOS	
	MIN.	MAX.	MIN.	MAX.	MIN.	MAX.	MIN.	MAX.
1	0							
4	0		0					
5		20		10		10		
7	0		0		0		0	
8	10		20		20			
16	0		0					
17	0		0					
18A	0		10					
19	0		0					
20	0		10					
21	0		0					
23	0		10		0			
24	0		0					
26	20		10		10		10	
27	0		0		0		0	
30	10		0		0		0	
31	0		0		0		0	
32		30		20		20		20
33	0		10		0			
34		30		20		20		20
35		30		20				
36	0		0					
37		30		20				
39	0		10					
40	30		20		20			
41	0		0					
43	0		0		0			
44	30		20		20		20	
45	30		30					
46	40		40					
47		20						
48		20						
49	30							
50	10				0			
51					0			
52					10			
53					0			
54	30				40			
55	0				10			
56		40		40		30		
57	40		40					

Notes:

1.- Les spécifications des paramètres de temps sont données aux tableaux 2-22 et 2-23. Utiliser les numéros de paramètres de temps pour localiser en séquence la spécification correspondante.

2.- Tous les temps sont en nanosecondes.

Table 2-21

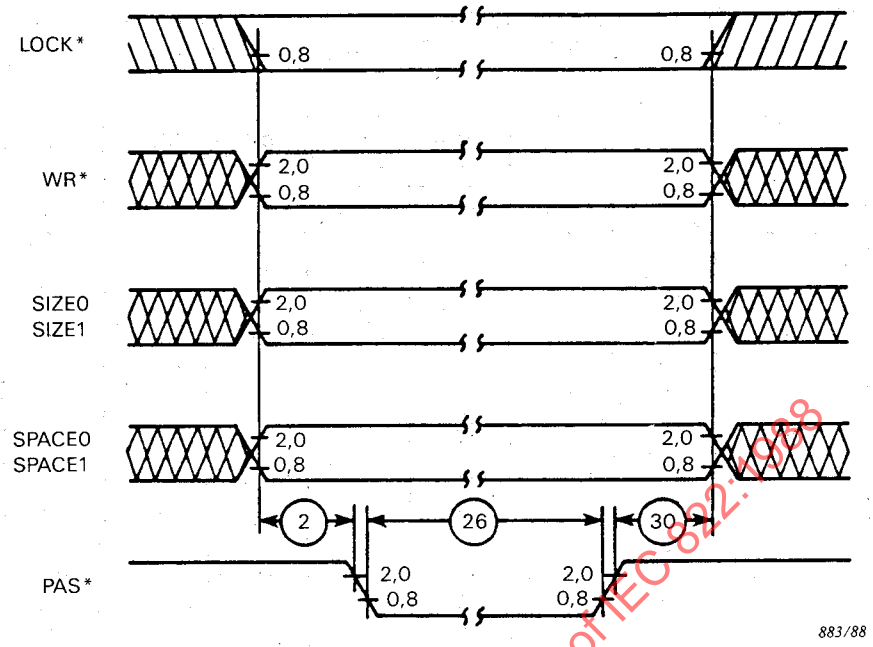
IHV MASTER, responding INTV SLAVE, contending INTV SLAVE and idle SLAVE timing parameters

PARAMETER NUMBER	IHV MASTER		RESPONDING INTV SLAVE		CONTENDING INTV SLAVE		IDLE SLAVE	
	MIN.	MAX.	MIN.	MAX.	MIN.	MAX.	MIN.	MAX.
1	0							
4	0		0					
5		20		10		10		
7	0		0		0		0	
8	10		20		20			
16	0		0					
17	0		0					
18A	0		10					
19	0		0					
20	0		10					
21	0		0					
23	0		10		0			
24	0		0					
26	20		10		10		10	
27	0		0		0		0	
30	10		0		0		0	
31	0		0		0		0	
32		30		20		20		20
33	0		10		0			
34		30		20		20		20
35		30		20				
36	0		0					
37		30		20				
39	0		10					
40	30		20		20			
41	0		0					
43	0		0		0			
44	30		20		20		20	
45	30		30					
46	40		40					
47		20						
48		20						
49	30							
50	10				0			
51					0			
52					10			
53					0			
54	30				40			
55	0				10			
56		40		40		30		
57	40		40					

Notes:

1.- The specifications for the timing parameters are given in Tables 2-22 and 2-23. Use the timing parameter number to sequentially locate the associated specification.

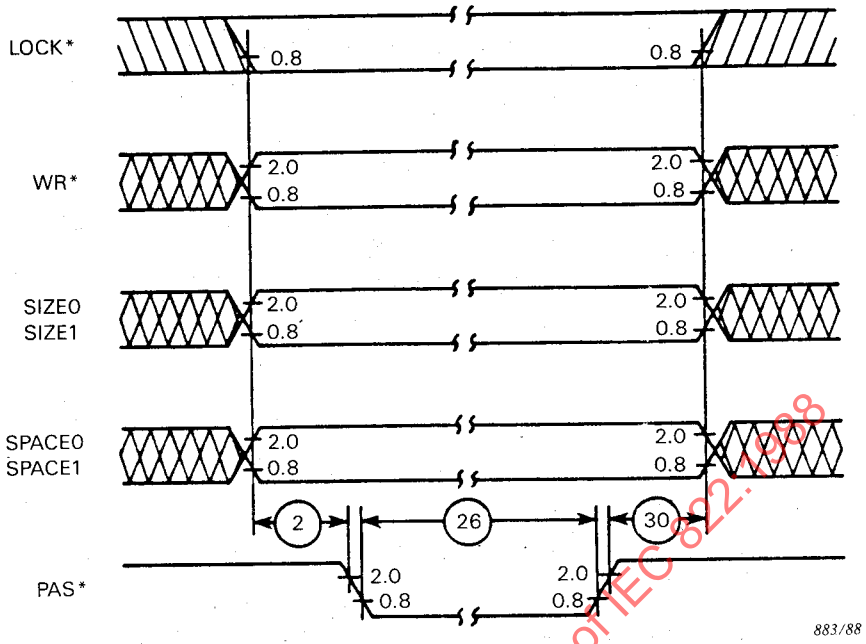
2.- All times are in nanoseconds.



Note:

Pendant tous les cycles le MAITRE actif maintient les niveaux des lignes LOCK*, WR*, SIZE0-SIZE1 et SPACE0-SPACE1 tout au long du cycle, comme spécifié au paragraphe 2.5.1.2.

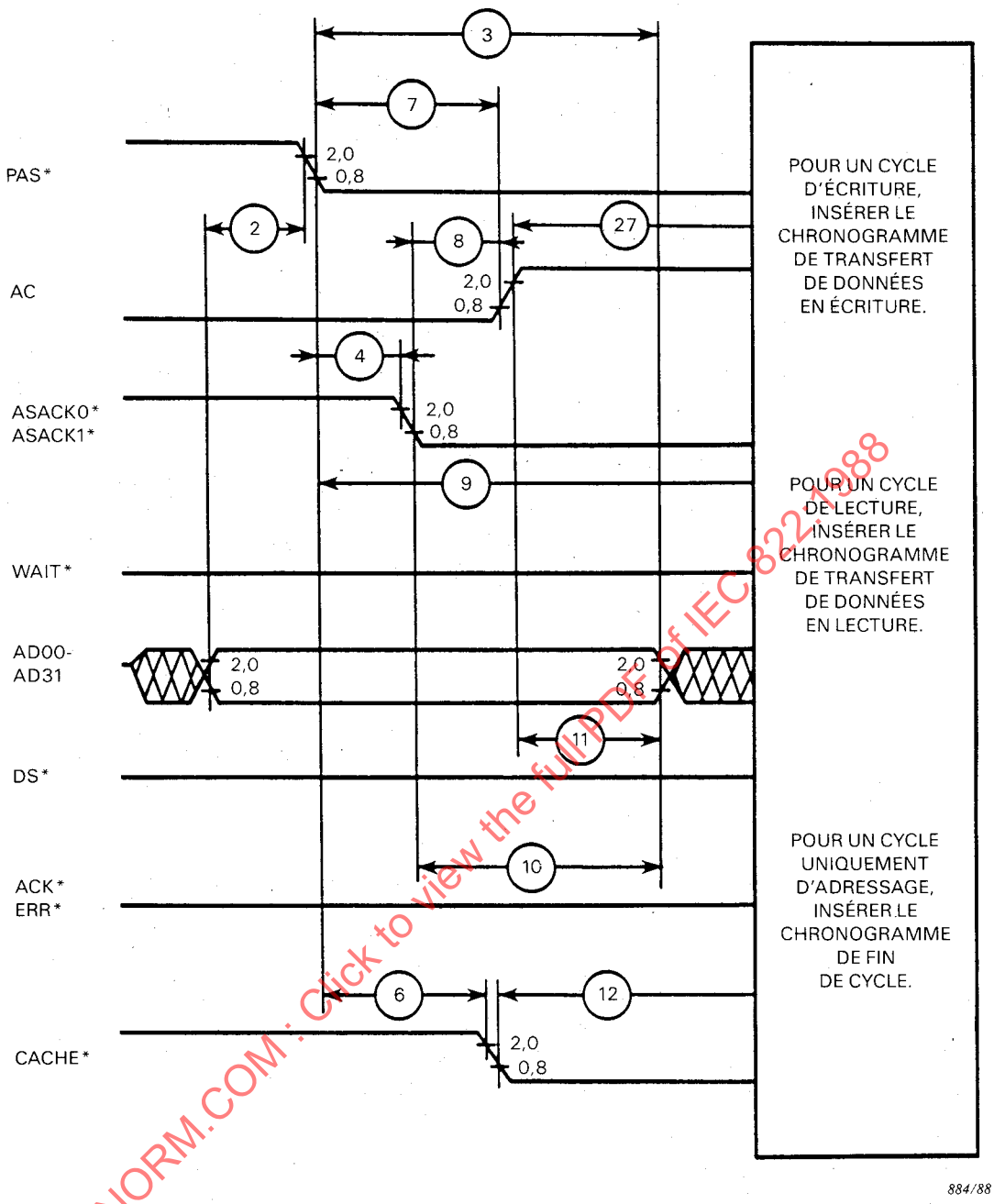
Fig. 2-15. - Chronologie des signaux LOCK*, WR*, SIZE0-SIZE1 et SPACE0-SPACE1 d'un MAITRE actif, d'un MAITRE IHV actif et d'un DEMANDEUR PAR actif pour les cycles de TRANSFERT UNIQUE, TRANSFERT PAR BLOC, RECONNAISSANCE D'INTERRUPTION et ARBITRAGE.



Note:

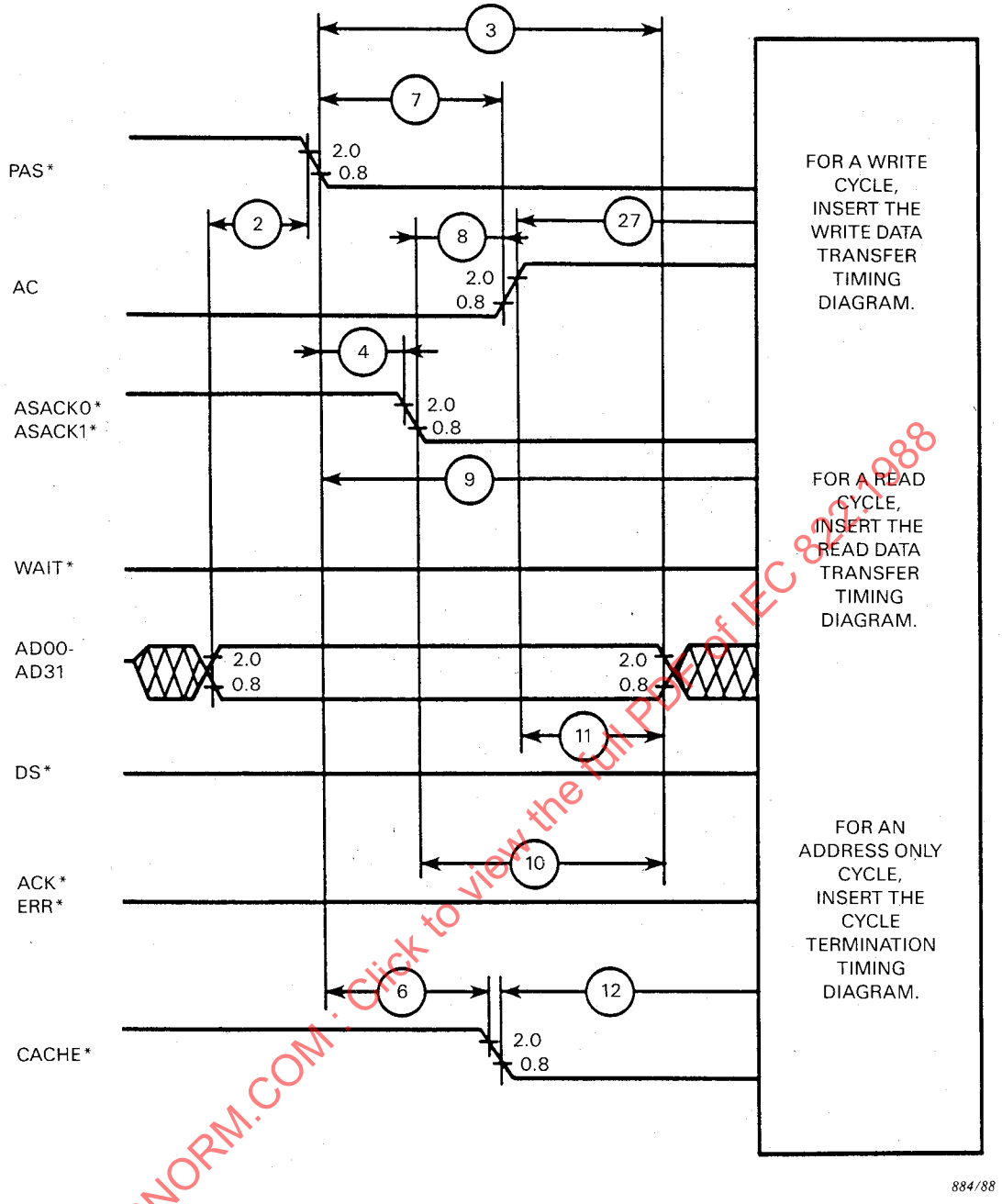
During all cycles, the active MASTER maintains the levels of LOCK*, WR*, SIZE0-SIZE1 and SPACE0-SPACE1 throughout the cycle, as specified in Paragraph 2.5.1.2.

Fig. 2-15. - Active MASTER, active IHV MASTER and active PAR REQUESTER, LOCK*, WR*, SIZE0-SIZE1 and SPACE0-SPACE1 timing, SINGLE-TRANSFER, BLOCK-TRANSFER, INTERRUPT-ACKNOWLEDGE and ARBITRATION cycles.



Note:
Pendant la phase de diffusion d'adresse, le MAITRE actif commande les lignes supplémentaires comme indiqué dans la figure 2-15.

Fig. 2-16. - Chronologie de la diffusion d'adresse du MAITRE actif et des ESCLAVES pour les cycles UNIQUEMENT D'ADRESSAGE, TRANSFERT UNIQUE et TRANSFERT PAR BLOC.



Note:

During the address broadcast phase, the active MASTER drives additional lines valid as shown in Figure 2-15.

Fig. 2-16. - Active MASTER and SLAVES, address broadcast timing, ADDRESS-ONLY, SINGLE-TRANSFER and BLOCK-TRANSFER cycles.

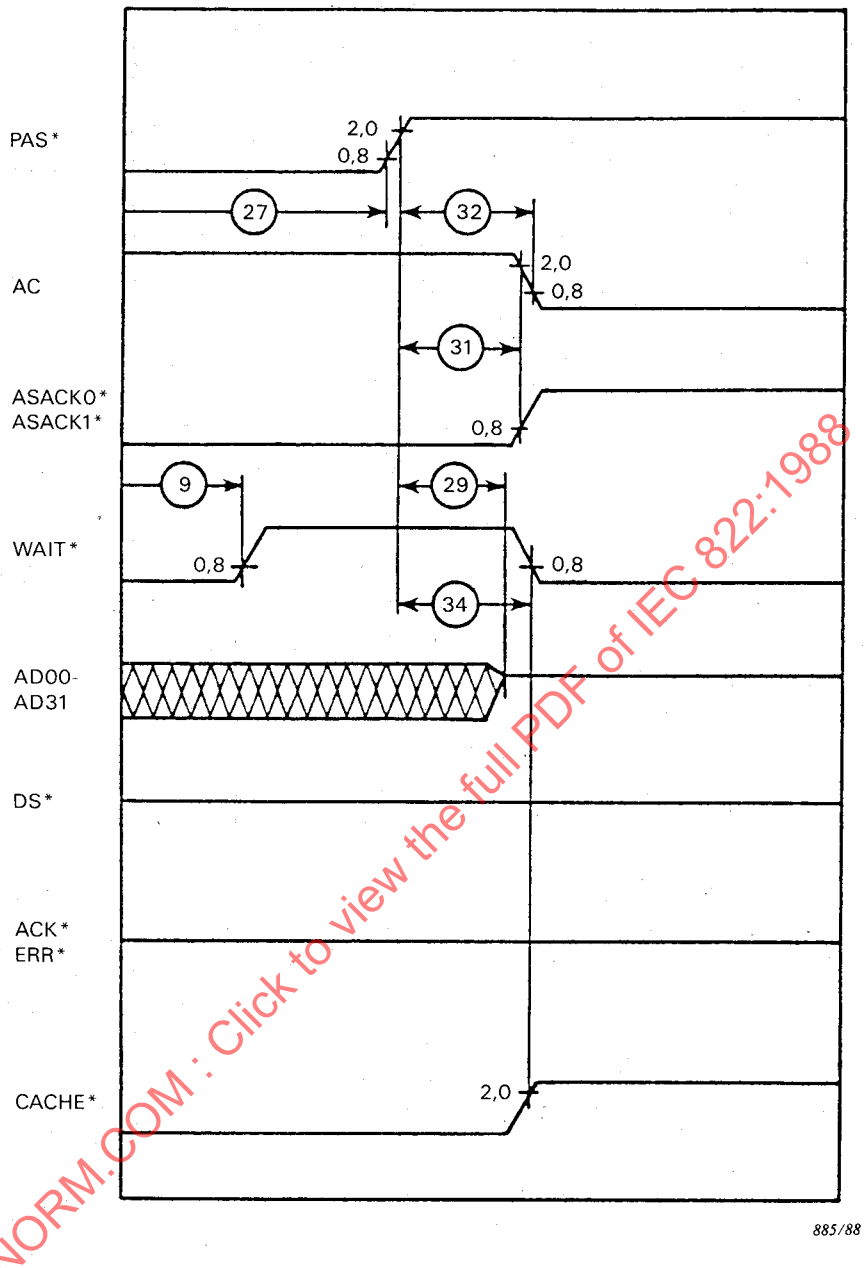


Fig. 2-17. - Fin de cycle du MAITRE actif et des ESCLAVES pour les cycles UNIQUEMENT D'ADRESSAGE.

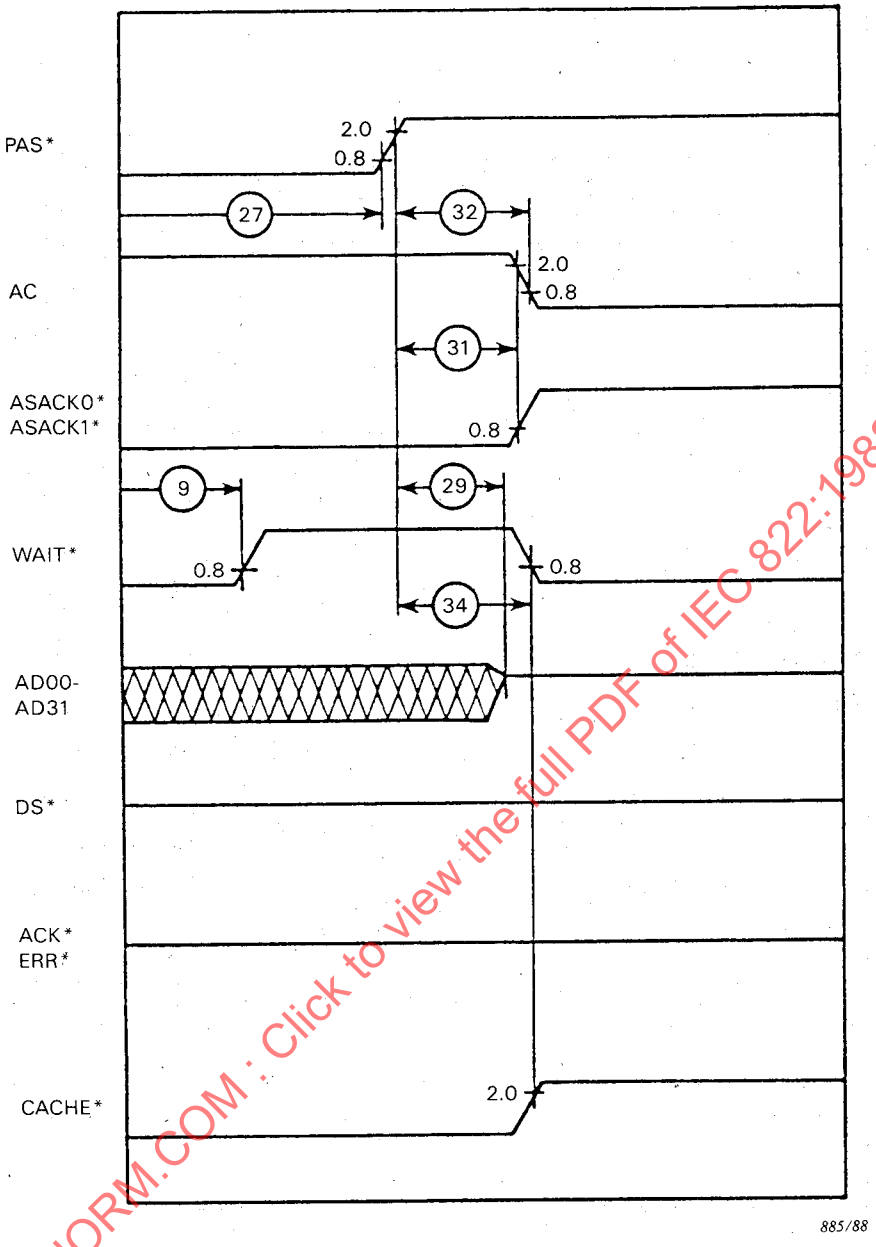
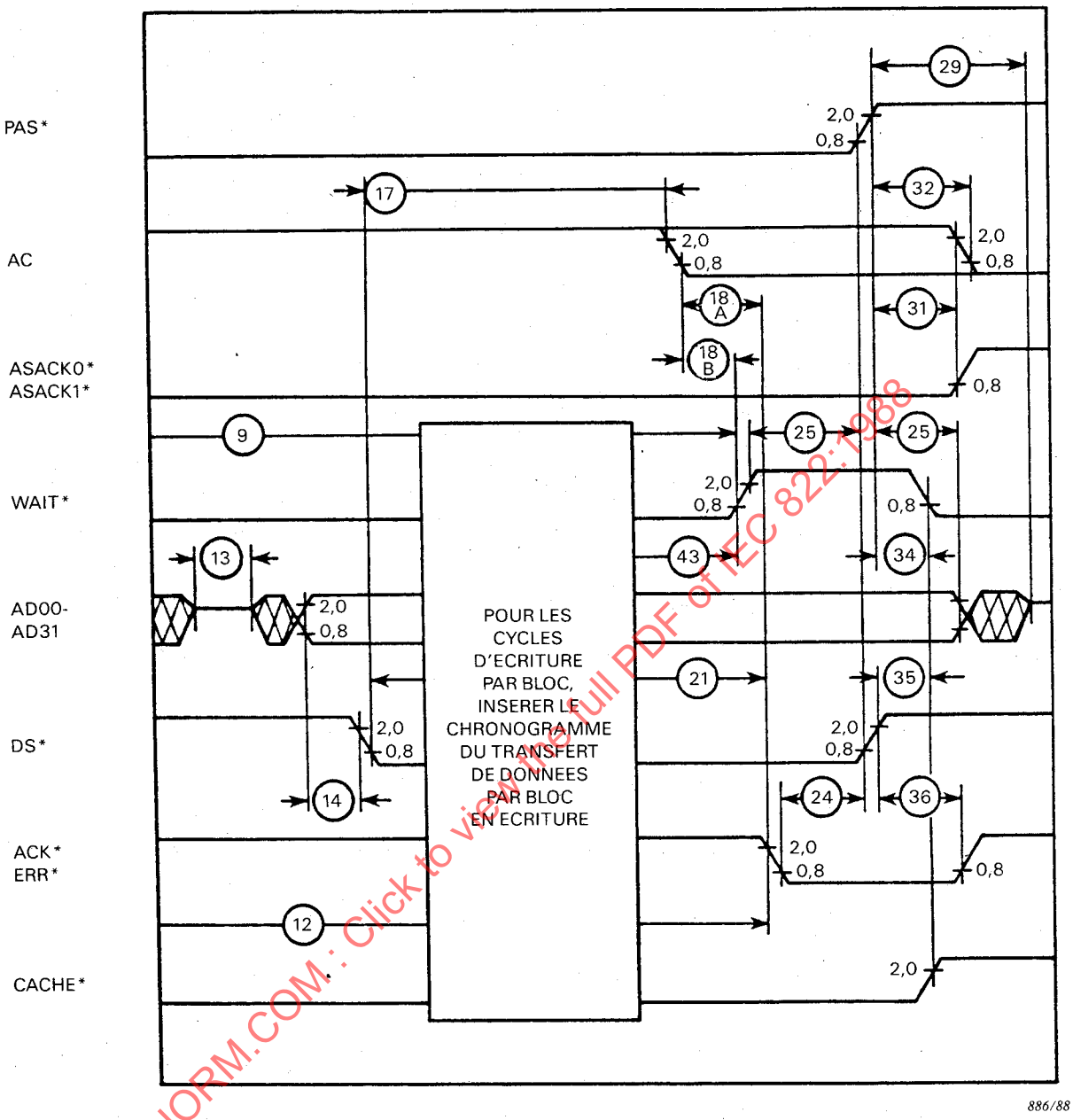


Fig. 2-17. - Active MASTER and SLAVES, cycle termination ADDRESS-ONLY cycles.



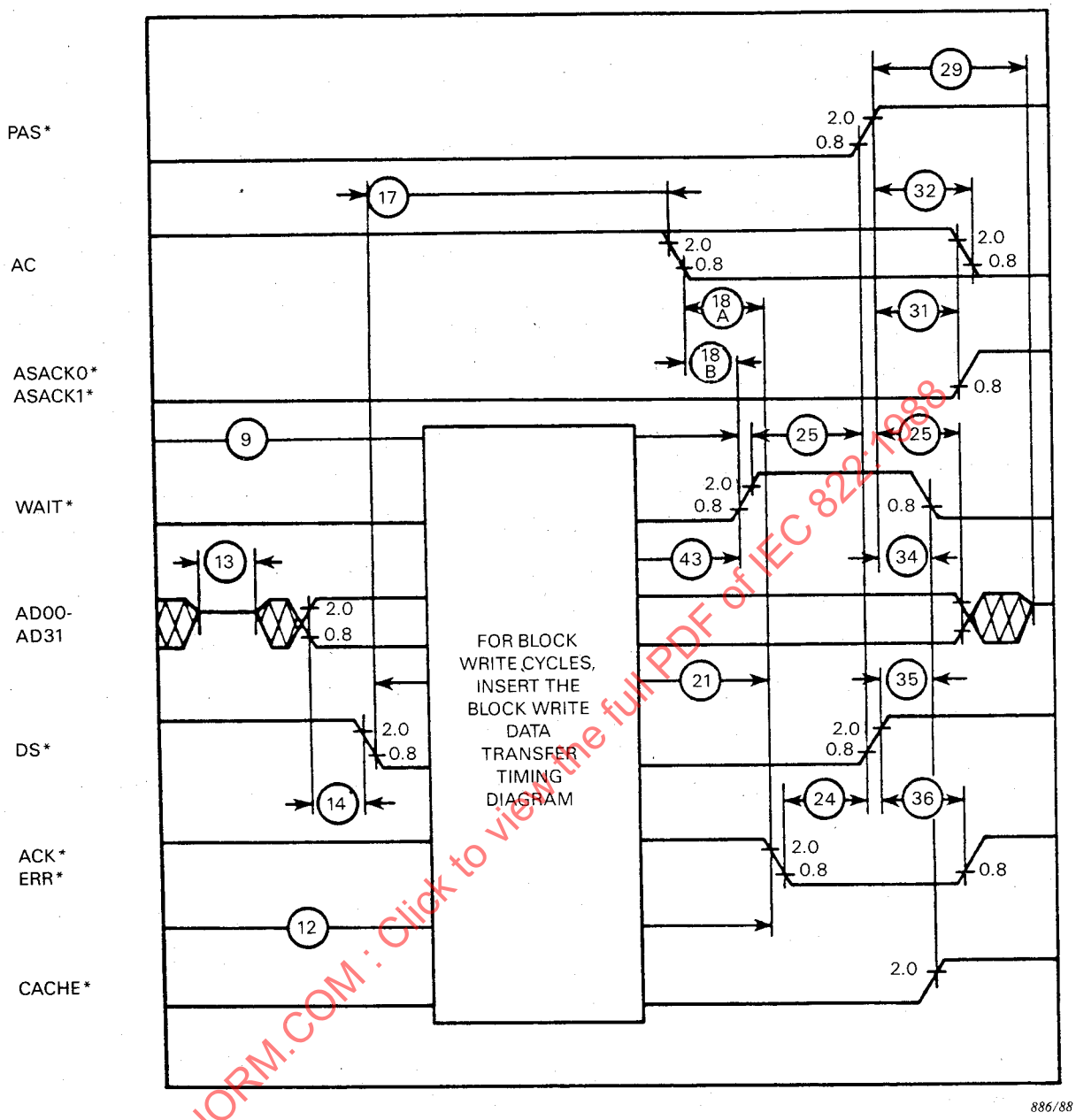
Notes:

1.- Pendant tous les cycles d'écriture, le MAITRE actif maintient un niveau bas sur WR* et le niveau approprié sur LOCK*, SIZE0-SIZE1 et SPACE0-SPACE1, tout au long du cycle (voir figure 2-15, page 146).

2.- Le paragraphe 2.5.1.2 spécifie l'utilisation de ces lignes.

Fig. 2-18. - Chronologie d'un transfert de données en écriture du MAITRE actif et des ESCLAVES pour les cycles de TRANSFERT UNIQUE et TRANSFERT PAR BLOC.

(Suite page 154.)



Notes:

1.- During all write cycles, the active MASTER maintains a low on WR* and the proper level on LOCK*, SIZE0-SIZE1 and SPACE0-SPACE1 throughout the cycle (see Figure 2-15, page 147).

2.- Paragraph 2.5.1.2 specifies the use of these lines.

Fig. 2-18. - Active MASTER and SLAVES, write data transfer timing, SINGLE-TRANSFER and BLOCK-TRANSFER cycles.
(Continued on page 155.)

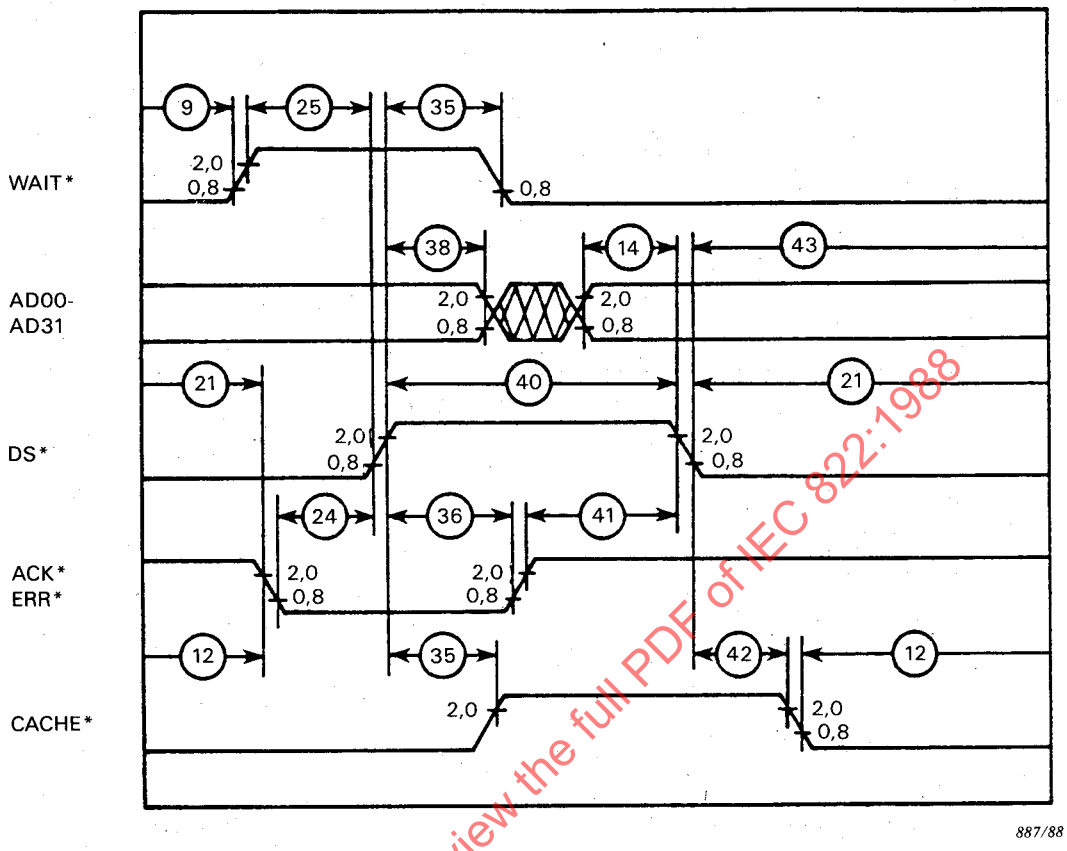


FIGURE 2-18 (fin).

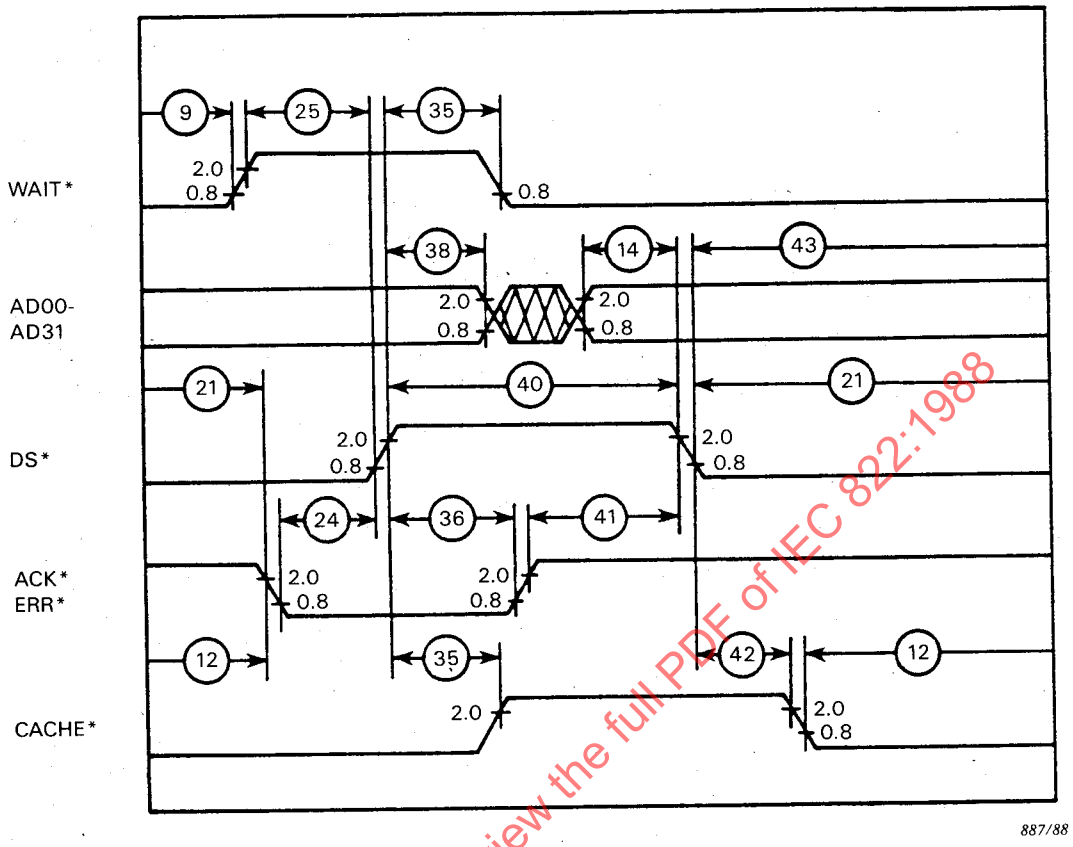
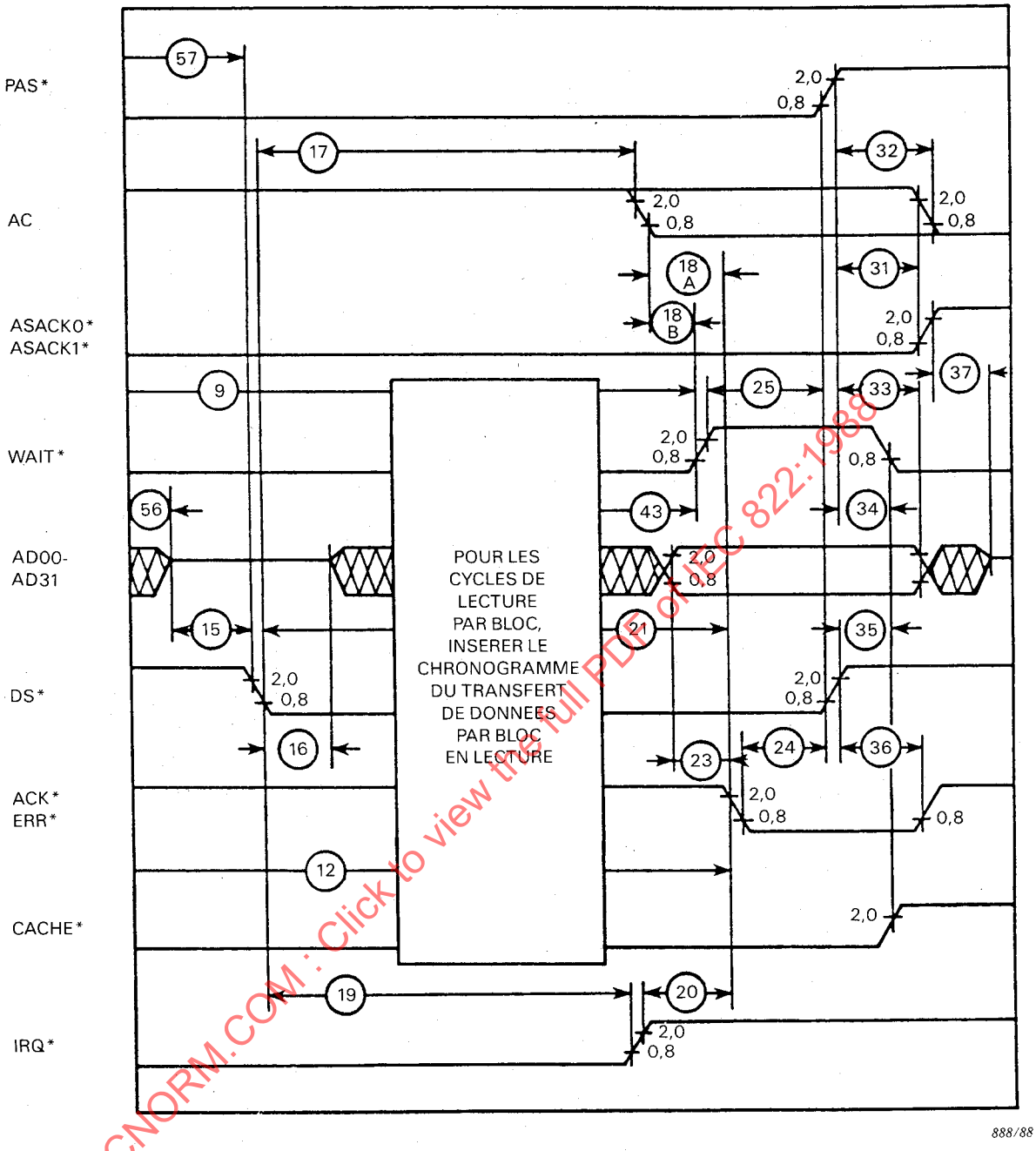


FIGURE 2-18 (concluded).



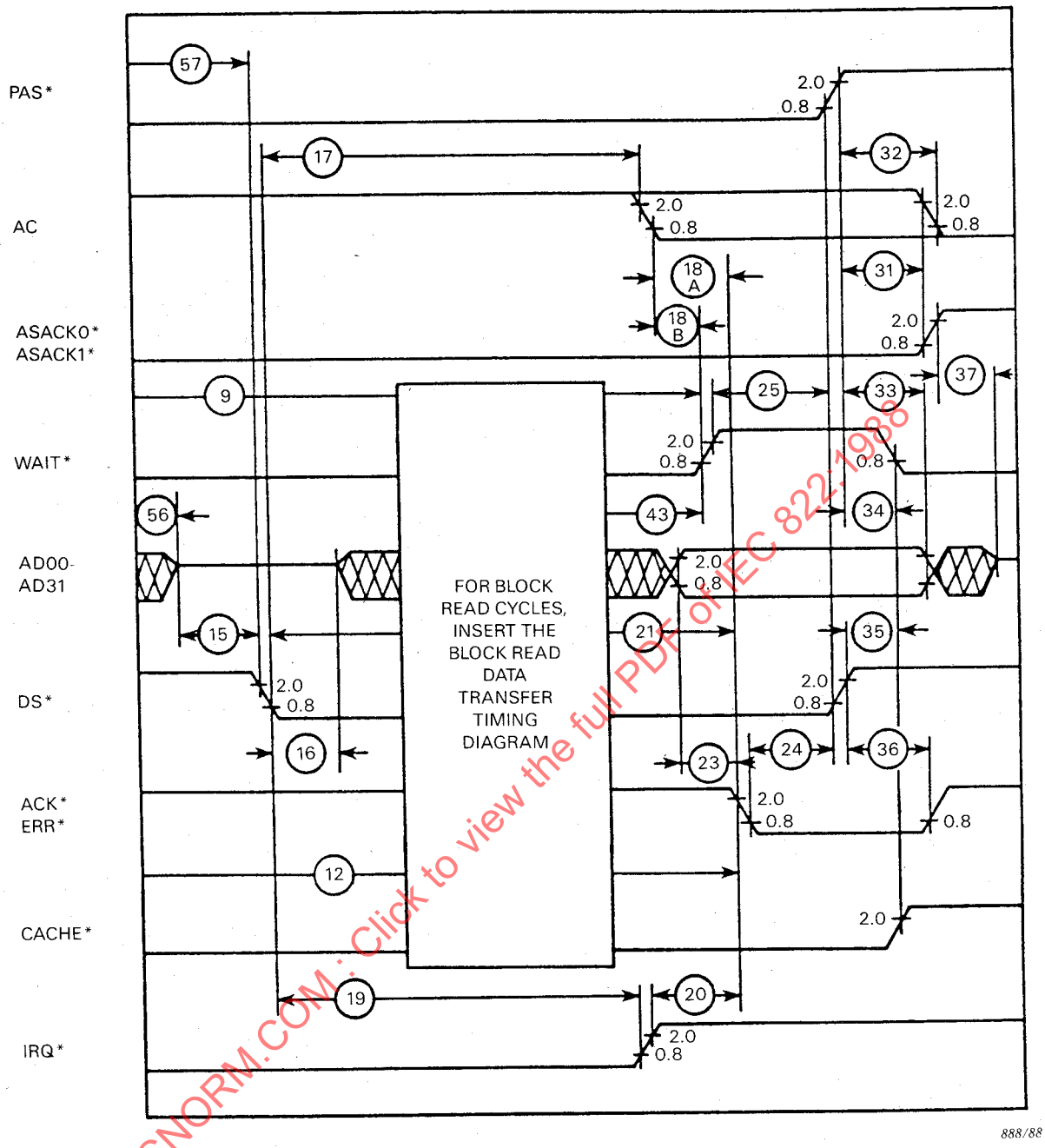
Notes:

1.- Pendant tous les cycles de lecture et cycles de RECONNAISSANCE D'INTERRUPTION, le MAITRE actif maintient un niveau haut sur WR* et le niveau approprié sur LOCK*, SIZE0-SIZE1 et SPACE0-SPACE1 tout au long du cycle (voir figure 2-15, page 146).

2.- Le paragraphe 2.5.1.2 spécifie l'utilisation de ces lignes pendant les cycles de lecture, alors que le paragraphe 2.5.4.2 spécifie leur utilisation pendant les cycles de RECONNAISSANCE D'INTERRUPTION.

Fig. 2-19. - Chronologie d'un transfert de données en lecture du MAITRE actif et des ESCLAVES pour les cycles de TRANSFERT UNIQUE, TRANSFERT PAR BLOC et RECONNAISSANCE D'INTERRUPTION.

(Suite page 158.)



Notes:

1.- During all read cycles and INTERRUPT-ACKNOWLEDGE cycles, the active MASTER maintains a high on WR* and the proper level on LOCK*, SIZE0-SIZE1 and SPACE0-SPACE1 throughout the cycle (see Figure 2-15, page 147).

2.- Paragraph 2.5.1.2 specifies the use of these lines during read cycles, while Paragraph 2.5.4.2 specifies their use during INTERRUPT-ACKNOWLEDGE cycles.

Fig. 2-19. - Active MASTER and SLAVES, read data transfer timing, SINGLE-TRANSFER, BLOCK-TRANSFER and INTERRUPT-ACKNOWLEDGE cycles.

(Continued on page 159.)

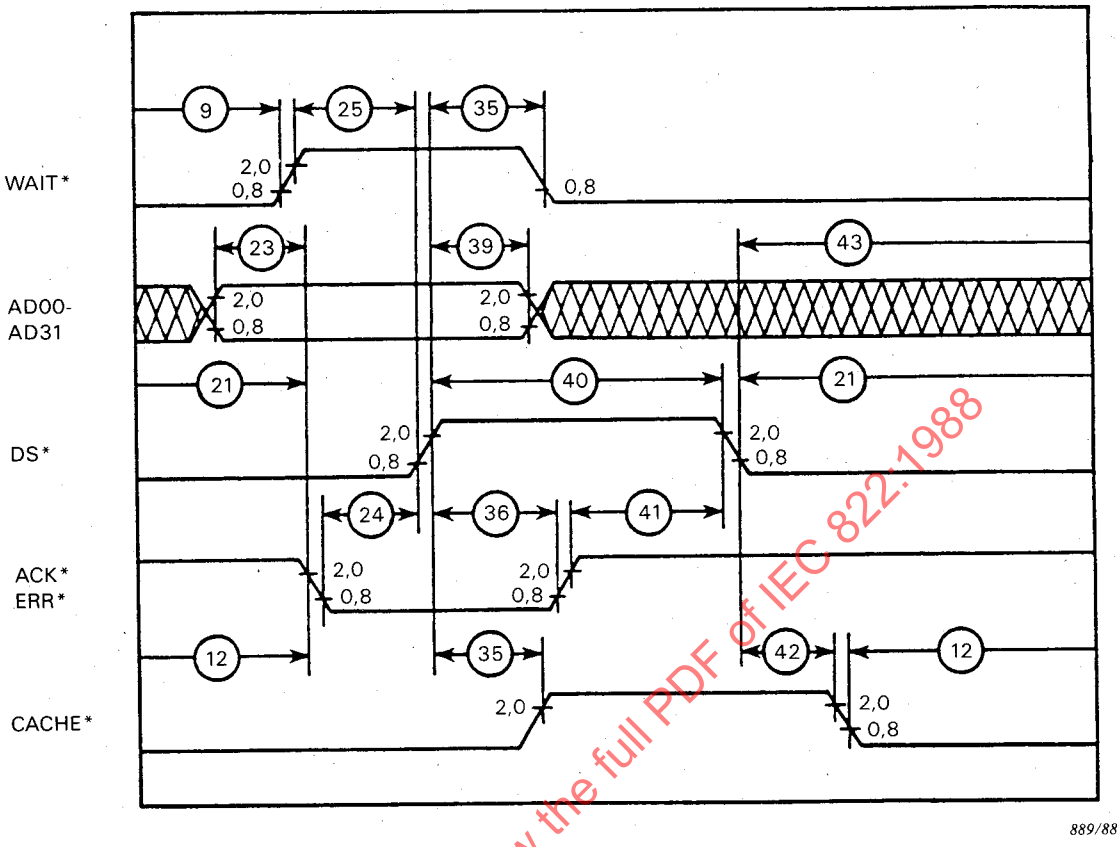


FIGURE 2-19 (fin).

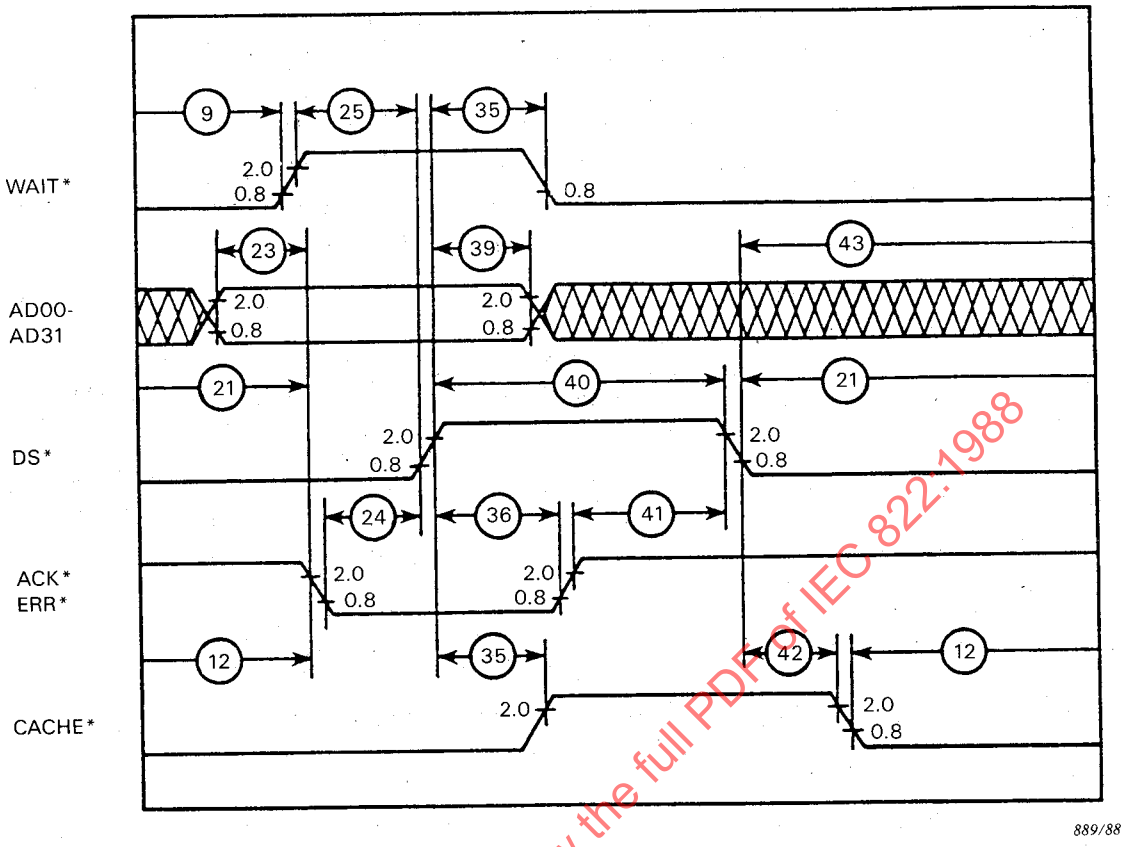
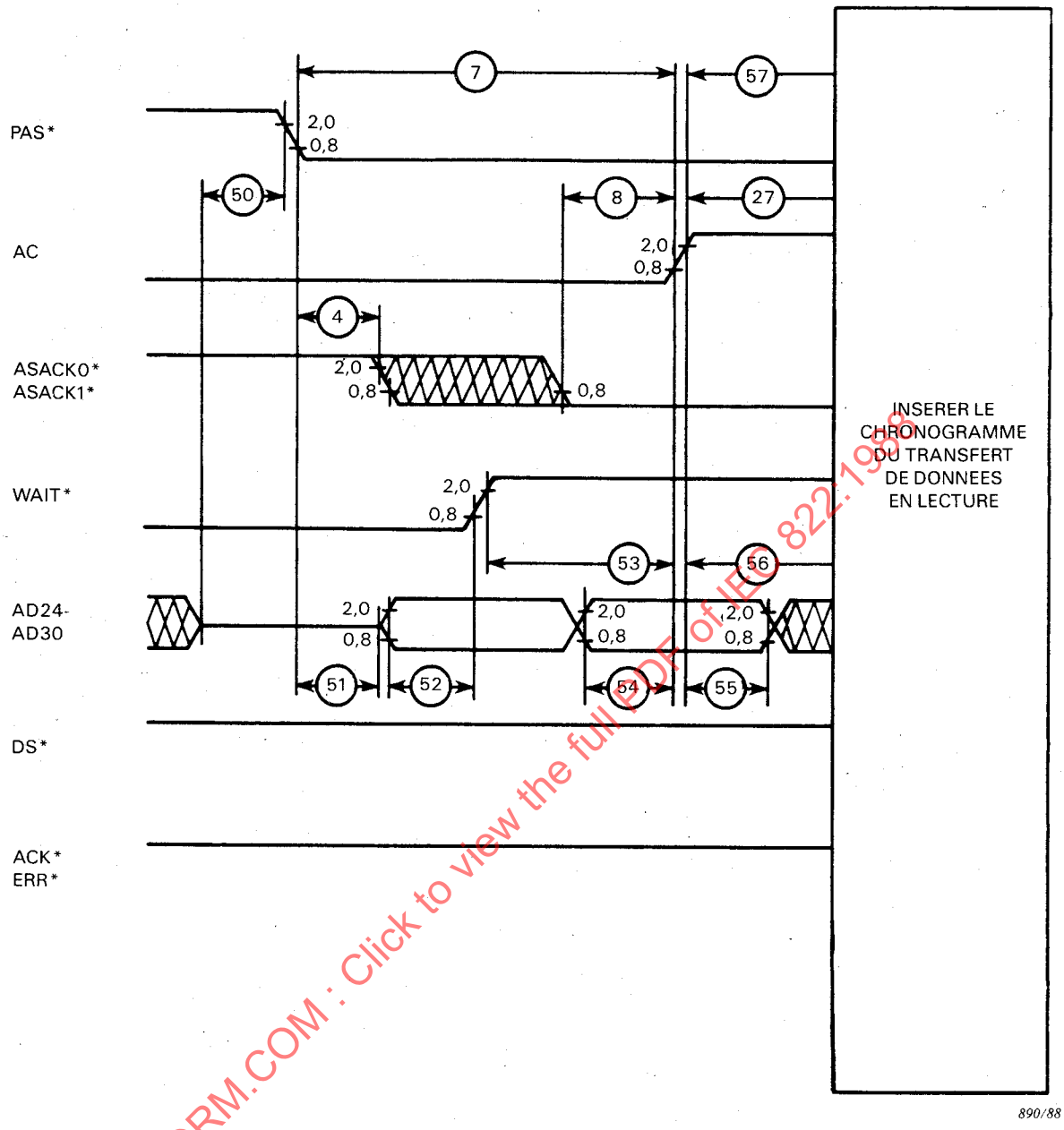


FIGURE 2-19 (concluded).

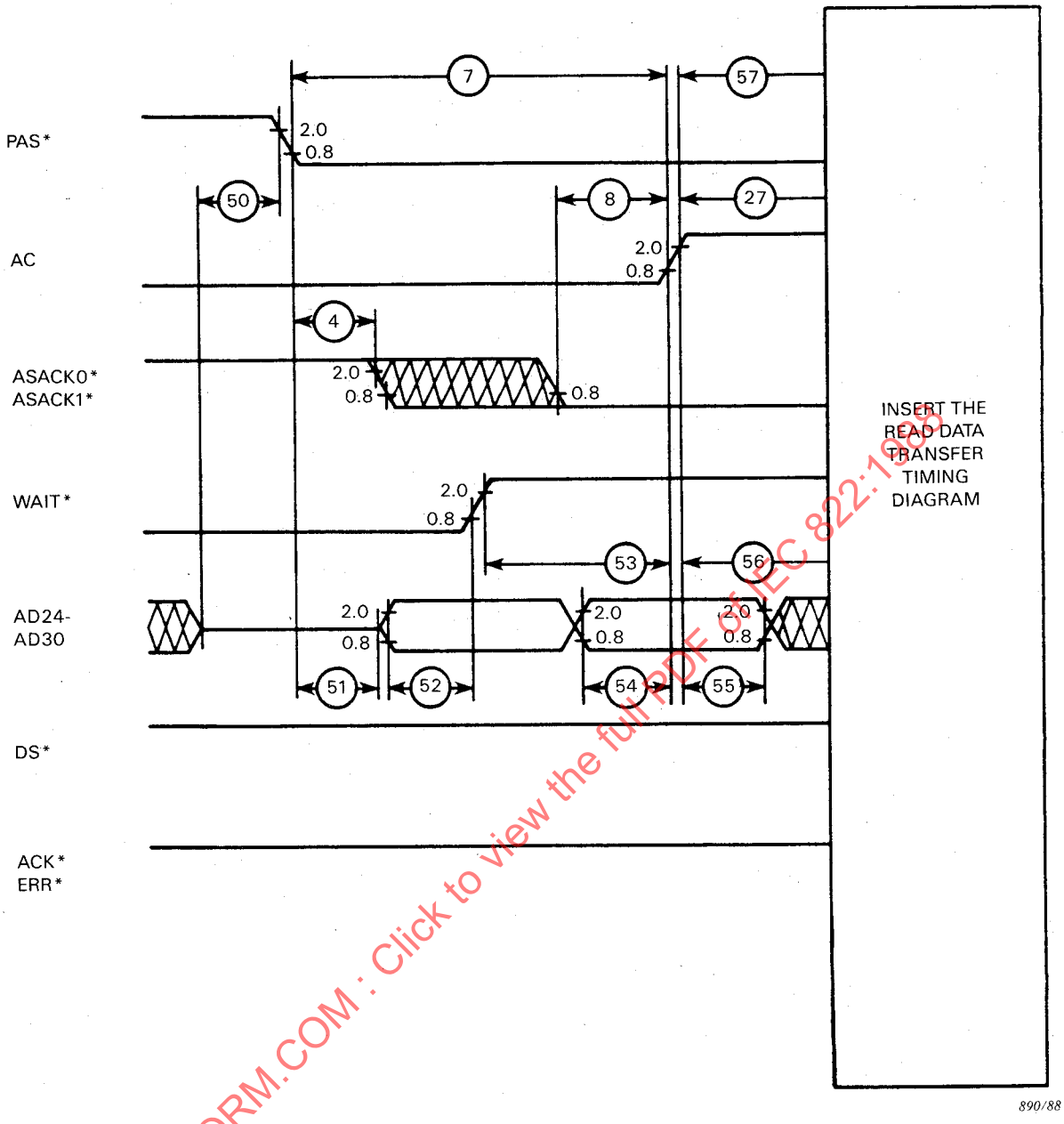


Notes:

1.- Pendant les cycles de RECONNAISSANCE D'INTERRUPTION, le MAITRE actif IHV maintient un niveau haut sur WR*, un niveau bas sur SPACE0-SPACE1 et le niveau approprié sur SIZE0-SIZE1 tout au long du cycle (voir figure 2-15, page 146).

2.- Le paragraphe 2.5.4.2 spécifie leur utilisation pendant les cycles de RECONNAISSANCE D'INTERRUPTION.

Fig. 2-20. - Phase de sélection du MAITRE IHV et des ESCLAVES INTV pour les cycles de RECONNAISSANCE D'INTERRUPTION.



Notes:

1.- During INTERRUPT-ACKNOWLEDGE cycles, the active IHV MASTER maintains a high on WR*, low on both SPACE0-SPACE1 and the proper level on SIZE0-SIZE1 throughout the cycle (see Figure 2-15, page 147).

2.- Paragraph 2.5.4.2 specifies their use during INTERRUPT-ACKNOWLEDGE cycles.

Fig. 2-20. - IHV MASTER and INTV SLAVES, selection phase INTERRUPT-ACKNOWLEDGE cycles.

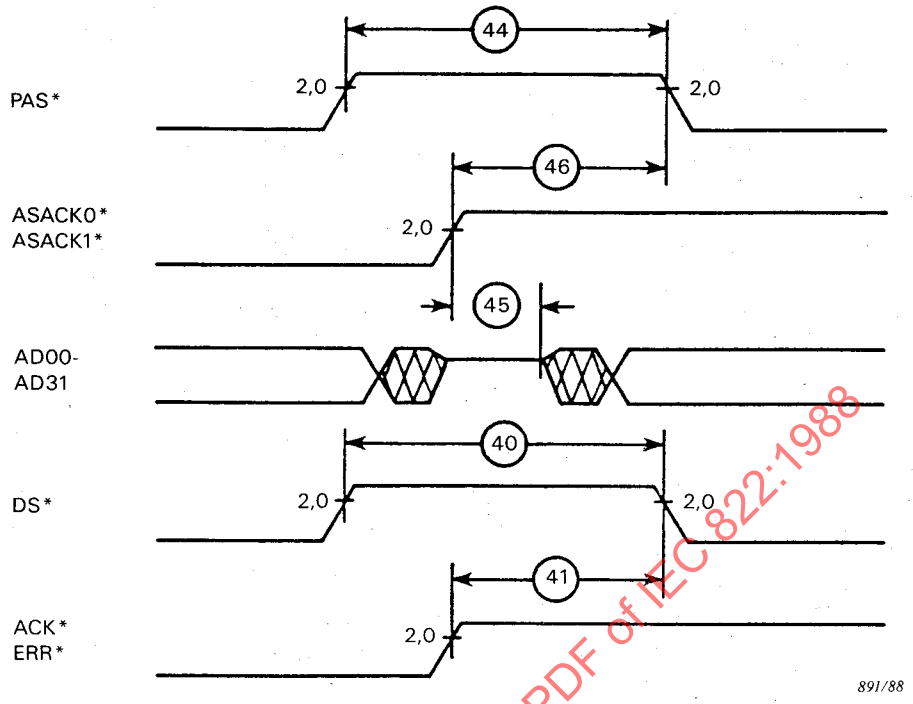


Fig. 2-21. - Chronologie des MAITRES et des ESCLAVES entre les cycles.

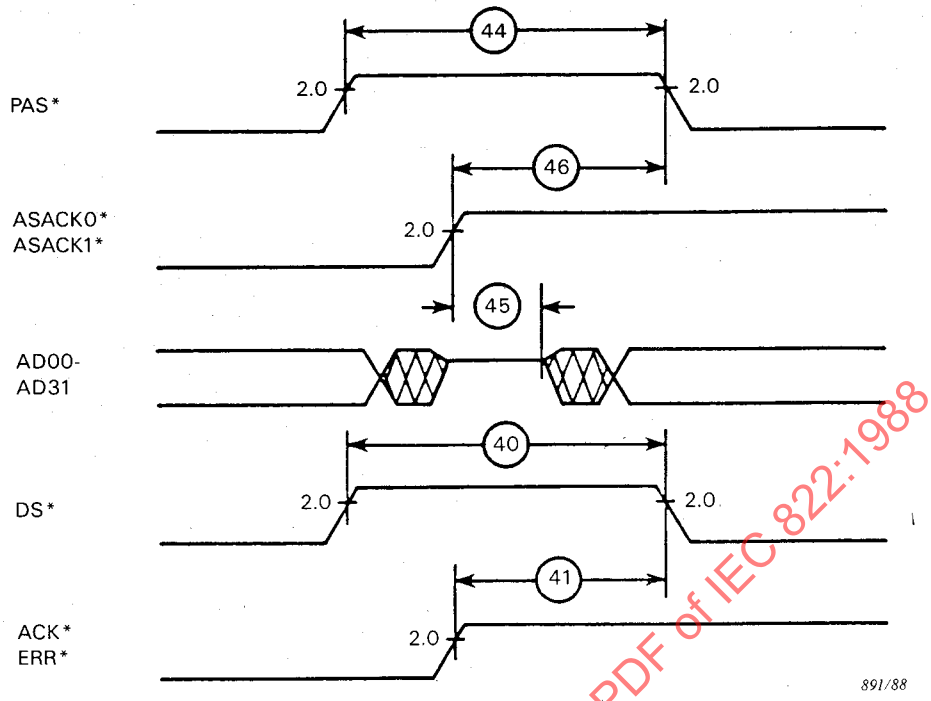
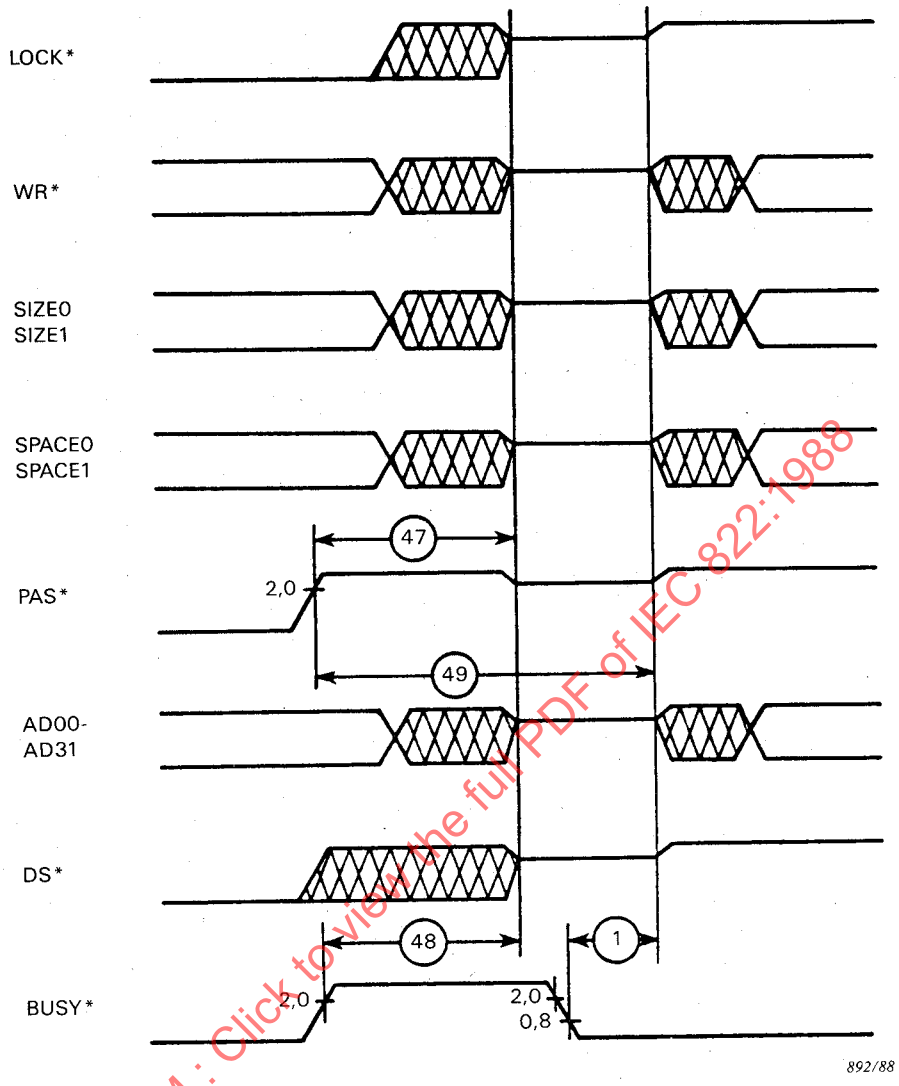


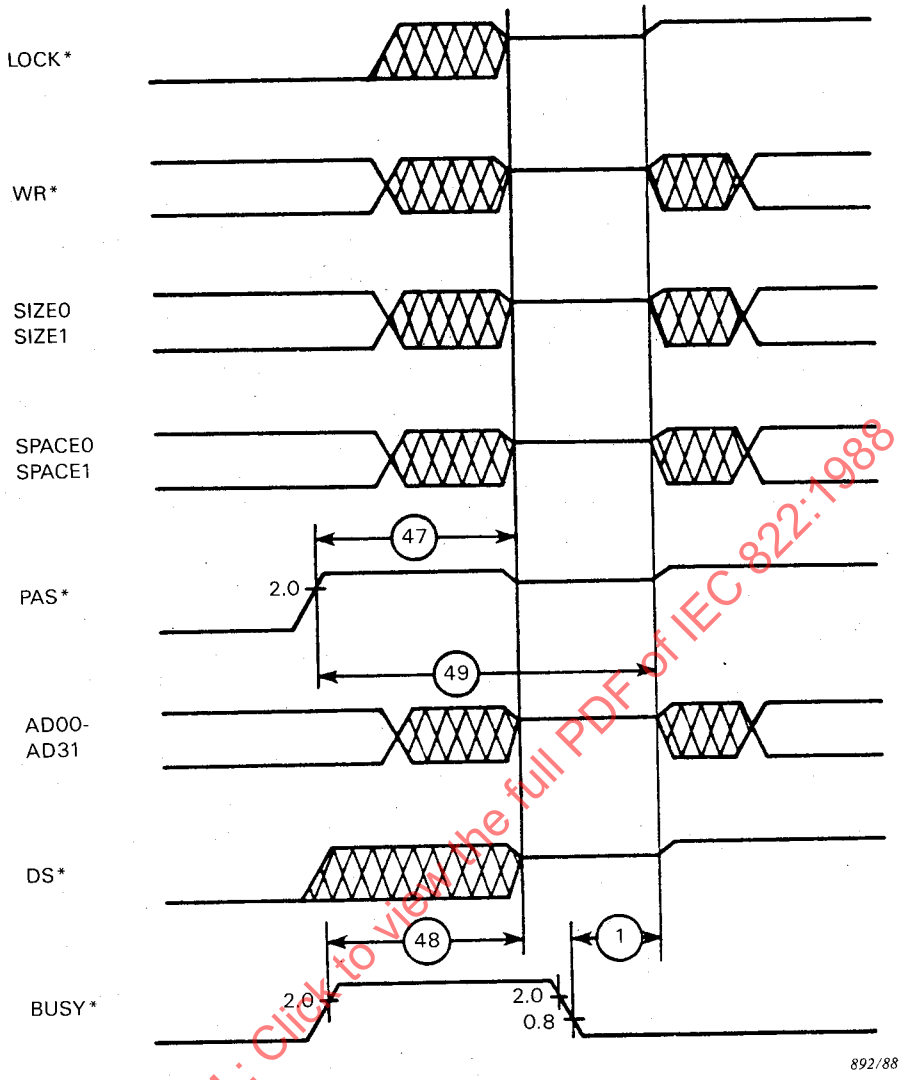
Fig. 2-21. - MASTERS and SLAVES intercycle timing.



Note:

BUSY* est un signal commandé par le DEMANDEUR associé au MAITRE lorsqu'il reçoit l'allocation du bus. Les spécifications de commande et de libération de la ligne BUSY* sont données au chapitre 3.

Fig.- 2-22. - Chronologie du transfert de contrôle du DTB.



Note:

BUSY* is a signal that the REQUESTER that is associated with the MASTER drives when it is granted the bus. The specifications for driving and releasing the BUSY* line are given in Chapter 3.

Fig 2-22. - DTB control transfer timing.

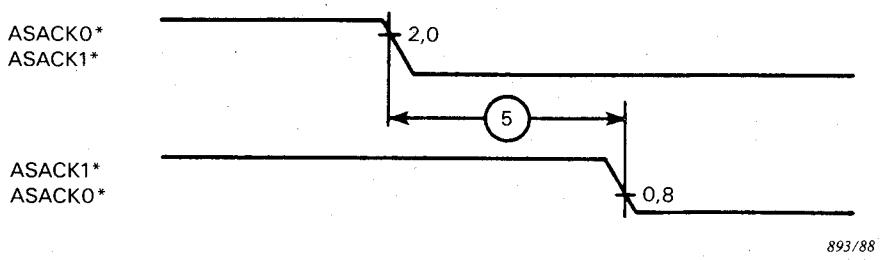


Fig. 2-23. - Déphasage entre ASACK0* et ASACK1*.

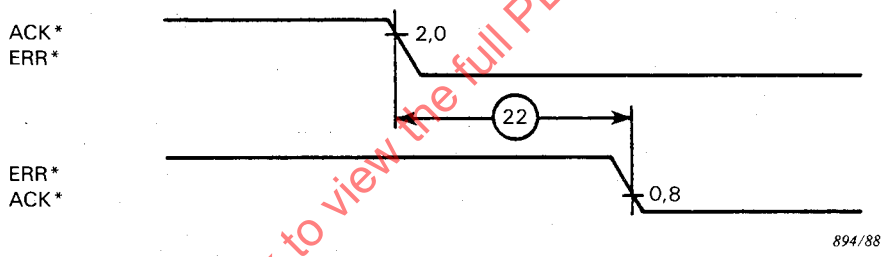


Fig. 2-24. - Déphasage entre ACK* et ERR*.

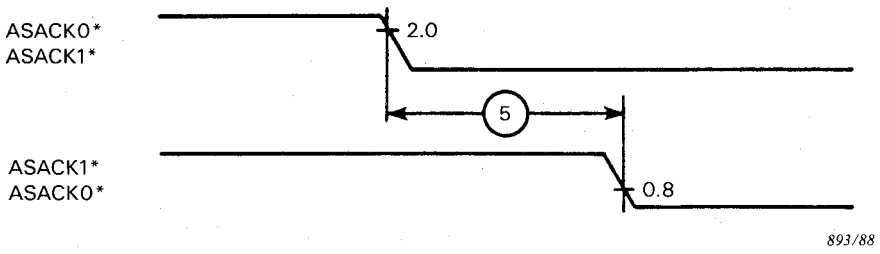


Fig. 2-23. - Skew between ASACK0* and ASACK1*.

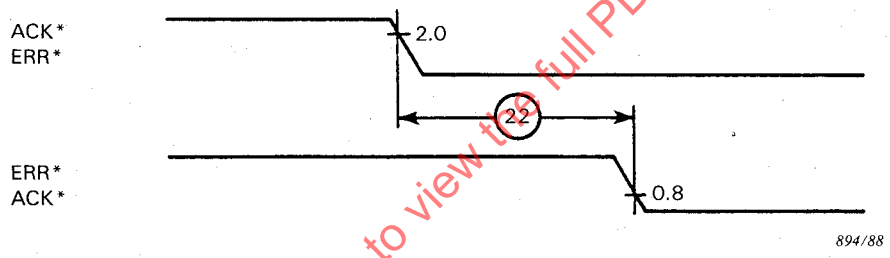


Fig. 2-24. - Skew between ACK* and ERR*.

Tableau 2-22
MAITRE
Spécifications de la chronologie

Note: La numérotation correspond aux paramètres de temps spécifiés dans les tableaux 2-20 et 2-21.

1. REGLE 2.44:
   Lorsqu'il prend le contrôle du VSB, le MAITRE actif NE DOIT commander aucune des lignes LOCK*, WR*, SIZE0-SIZE1, SPACE0-SPACE1, PAS*, AD00-AD31 ou DS* jusqu'à ce que son DEMANDEUR ait reçu l'allocation du bus.

   OBSERVATION 2.19:
   Le MAITRE du DEMANDEUR prend le contrôle du DTB en commandant BUSY* au niveau bas, comme décrit au chapitre 3.

2. REGLE 2.45:
   Le MAITRE actif NE DOIT PAS commander PAS* au niveau bas avant que LOCK*, WR*, SIZE0-SIZE1, SPACE0-SPACE1 et AD00-AD31 aient été valides au minimum pendant ce temps.

3. REGLE 2.46:
   Le MAITRE actif DOIT maintenir une adresse valide sur AD00-AD31 au minimum pendant ce temps après avoir commandé PAS* au niveau bas.

4. OBSERVATION 2.20:
   Le MAITRE actif est assuré que l'ESCLAVE répondant ne commandera au niveau bas ni ASACK0* ni ASACK1* avant ce délai après que le MAITRE actif a commandé PAS* au niveau bas.

5. OBSERVATION 2.21:
   Le MAITRE actif est assuré que si l'ESCLAVE répondant commande à la fois ASACK0* et ASACK1* au niveau bas, alors il commandera le second au niveau bas dans ce délai maximal après avoir commandé le premier au niveau bas.

6. OBSERVATION 2.22:
   Le MAITRE actif est assuré que l'ESCLAVE répondant ne commandera pas CACHE* au niveau bas avant ce délai après que le MAITRE actif a commandé PAS* au niveau bas.

7. OBSERVATION 2.23:
   Le MAITRE actif est assuré que les ESCLAVES ne libéreront pas AC au niveau haut avant ce délai après que le MAITRE actif a commandé PAS* au niveau bas.

8. OBSERVATION 2.24:
   Le MAITRE actif est assuré que l'ESCLAVE répondant ne libérera pas AC au niveau haut avant ce délai après avoir commandé ASACK0* et/ou ASACK1* au niveau bas.

9. OBSERVATION 2.25:
   SI un ESCLAVE commande WAIT* au niveau bas,
   ALORS le MAITRE actif est assuré qu'il ne le libérera pas au niveau haut avant ce délai après que le MAITRE actif a commandé PAS* au niveau bas.

10. REGLE 2.47:
    Le MAITRE actif DOIT maintenir les adresses sur AD00-AD31 pendant ce temps après le passage de ASACK0* et/ou ASACK1* au niveau bas.

Table 2-22
MASTER
Timing specifications

Note: The numbers correspond to the timing parameters specified in Tables 2-20 and 2-21.

1. RULE 2.44:
When taking control of the VSB, the active MASTER MUST NOT drive any of LOCK*, WR*, SIZE0-SIZE1, SPACE0-SPACE1, PAS*, AD00-AD31 or DS* until after its REQUESTER is granted use of the bus.

OBSERVATION 2.19:
The MASTER'S REQUESTER assumes control of the DTB by driving BUSY* low, as described in Chapter 3.

2. RULE 2.45:
The active MASTER MUST NOT drive PAS* low until LOCK*, WR*, SIZE0-SIZE1, SPACE0-SPACE1 and AD00-AD31 have been valid for this minimum time.

3. RULE 2.46:
The active MASTER MUST maintain a valid address on AD00-AD31 for this minimum time after driving PAS* to low.

4. OBSERVATION 2.20:
The active MASTER is guaranteed that the responding SLAVE will not drive either ASACK0* and/or ASACK1* to low until this time after the active MASTER has driven PAS* low.

5. OBSERVATION 2.21:
The active MASTER is guaranteed that if the responding SLAVE drives both ASACK0* and ASACK1* low, then it will drive the second one low within this maximum time after it has driven the first one low.

6. OBSERVATION 2.22:
The active MASTER is guaranteed that the responding SLAVE will not drive CACHE* low until this time after the active MASTER has driven PAS* low.

7. OBSERVATION 2.23:
The active MASTER is guaranteed that SLAVES will not release AC to high until this time after the active MASTER has driven PAS* low.

8. OBSERVATION 2.24:
The active MASTER is guaranteed that the responding SLAVE will not release AC to high until this time after it has driven ASACK0* and/or ASACK1* to low.

9. OBSERVATION 2.25:
IF a SLAVE drives WAIT* low,
THEN the active MASTER is guaranteed that it will not release it to high until this time after the active MASTER has driven PAS* low.

10. RULE 2.47:
The active MASTER MUST maintain the address on AD00-AD31 for this time after ASACK0* and/or ASACK1* are low.

11. REGLE 2.48:
SI le MAITRE actif détecte un niveau bas sur WAIT*,
ALORS il NE DOIT libérer aucun des signaux AD00-AD31 avant ce délai après que AC est passé au niveau haut.

12. OBSERVATION 2.26:
Le MAITRE actif est assuré que si l'ESCLAVE répondant commande CACHE* au niveau bas pendant ce cycle, alors il le fera au moins pendant ce temps avant de commander soit ACK*, soit ERR* au niveau bas.

13. REGLE 2.49:
Pendant les cycles d'écriture, le MAITRE actif NE DOIT PAS placer des données sur AD00-AD31 avant ce délai après la commutation à l'état inactif des émetteurs d'adresse.

14. REGLE 2.50:
Pendant les transferts de données en écriture, le MAITRE actif NE DOIT PAS commander DS* au niveau bas avant que les données n'aient été valides sur AD00-AD31 au minimum pendant ce temps.

15. REGLE 2.51:
Pendant les cycles de lecture et les cycles de RECONNAISSANCE D'INTERRUPTION, le MAITRE NE DOIT PAS commander DS* au niveau bas avant d'avoir libéré toutes les lignes AD00-AD31.

16. OBSERVATION 2.27:
Pendant les cycles de lecture et les cycles de RECONNAISSANCE D'INTERRUPTION, le MAITRE actif est assuré que l'ESCLAVE répondant ne commandera aucune des lignes AD00-AD31 avant ce délai après que le MAITRE a commandé DS* au niveau bas.

17. OBSERVATION 2.28:
Le MAITRE actif est assuré que les ESCLAVES ne commanderont pas AC au niveau bas pour indiquer qu'ils ont atteint les limites de leur intervalle d'adressage avant ce délai après que le MAITRE a commandé DS* au niveau bas.

18A. OBSERVATION 2.29:
Le MAITRE actif est assuré que si l'ESCLAVE répondant commande AC au niveau bas pour indiquer qu'il a atteint les limites de son intervalle d'adressage, alors il le fera au minimum pendant ce temps avant de commander ACK* et/ou ERR* au niveau bas.

18B. OBSERVATION 2.30:
Le MAITRE actif est assuré que si un ESCLAVE participant commande AC au niveau bas pour indiquer qu'il a atteint les limites de son intervalle d'adressage, alors il le fera au minimum pendant ce temps avant de libérer WAIT* au niveau haut.

21. OBSERVATION 2.31:
Le MAITRE actif est assuré que l'ESCLAVE répondant ne commandera ni ACK* ni ERR* au niveau bas avant ce délai après que le MAITRE a commandé DS* au niveau bas.

22. OBSERVATION 2.32:
Le MAITRE actif est assuré que si l'ESCLAVE répondant commande à la fois ACK* et ERR* au niveau bas, alors il commandera le second au niveau bas avant ce délai maximal après avoir commandé le premier au niveau bas.

11. RULE 2.48:
IF the active MASTER detects a low level on WAIT*,
THEN it MUST NOT release any of AD00-AD31 until this time after AC is high.

12. OBSERVATION 2.26:
The active MASTER is guaranteed that if the responding SLAVE drives CACHE* low during the cycle, then it will do so at least this time before driving either ACK* or ERR* low.

13. RULE 2.49:
During write cycles, the active MASTER MUST NOT drive data on AD00-AD31 until this time after turning its address drivers off.

14. RULE 2.50:
During write data transfers, the active MASTER MUST NOT drive DS* low until data has been valid on AD00-AD31 for this minimum time.

15. RULE 2.51:
During read cycles and INTERRUPT-ACKNOWLEDGE cycles, the MASTER MUST NOT drive DS* low until it has released all of AD00-AD31.

16. OBSERVATION 2.27:
During read cycles and INTERRUPT-ACKNOWLEDGE cycles, the active MASTER is guaranteed that the responding SLAVE will not drive any of AD00-AD31 until this time after the MASTER has driven DS* low.

17. OBSERVATION 2.28:
The active MASTER is guaranteed that SLAVES will not drive AC low to indicate that they have reached the end of their self addressing range until this time after the MASTER has driven DS* low.

18A. OBSERVATION 2.29:
The active MASTER is guaranteed that if the responding SLAVE drives AC low to indicate that it has reached the end of its self addressing range, then it will do so at least this time before driving ACK* and/or ERR* low.

18B. OBSERVATION 2.30:
The active MASTER is guaranteed that if a participating SLAVE drives AC low to indicate that it has reached the end of its self addressing range, then it will do so at least this time before releasing WAIT* to high.

21. OBSERVATION 2.31:
The active MASTER is guaranteed that the responding SLAVE will not drive either ACK* or ERR* low until this time after the MASTER has driven DS* low.

22. OBSERVATION 2.32:
The active MASTER is guaranteed that if the responding SLAVE drives both ACK* and ERR* low, then it will drive the second one low within this maximum time after it has driven the first one low.

23. OBSERVATION 2.33:
Pendant les transferts de données en lecture et les transferts de MOT D'ETAT/ID, le MAITRE actif est assuré que l'ESCLAVE répondant présentera des données valides sur AD00-AD31 au minimum pendant ce temps avant de commander ACK* et/ou ERR* au niveau bas.

24. REGLE 2.52:
Le MAITRE actif NE DOIT PAS permettre à PAS* ou DS* de passer au niveau haut avant ce délai après le passage de ACK* et/ou ERR* au niveau bas.

25. REGLE 2.53:
Pendant les cycles de TRANSFERT UNIQUE et les cycles de TRANSFERT PAR BLOC, le MAITRE actif NE DOIT PAS permettre à PAS* ou DS* de passer au niveau haut avant ce délai après avoir détecté WAIT* au niveau haut.

26. REGLE 2.54:
Le MAITRE actif DOIT maintenir PAS* au niveau bas au minimum pendant ce temps.

27. REGLE 2.55:
Pendant les cycles UNIQUEMENT D'ADRESSAGE et les cycles de RECONNAISSANCE D'INTERRUPTION, le MAITRE actif NE DOIT PAS permettre à PAS* de passer au niveau haut avant ce délai après détection d'un niveau haut sur AC.

28. REGLE 2.56:
Pendant les cycles d'écriture, le MAITRE actif NE DOIT PAS modifier AD00-AD31 avant ce délai après avoir permis à PAS* de passer au niveau haut.

29. REGLE 2.57:
Pendant les cycles UNIQUEMENT D'ADRESSAGE et les cycles d'écriture, le MAITRE actif DOIT cesser de commander AD00-AD31 dans ce délai après avoir permis à PAS* de passer au niveau haut.

30. REGLE 2.58:
Le MAITRE NE DOIT PAS modifier les niveaux sur l'une des lignes LOCK*, WR*, SIZE0-SIZE1 ou SPACE0-SPACE1 avant ce délai après avoir permis à PAS* de passer au niveau haut.

31. OBSERVATION 2.34:
Le MAITRE actif est assuré que les ESCLAVES ne libéreront pas ASACK0*-ASACK1* au niveau haut, ou, s'ils n'ont pas atteint les limites de leur intervalle d'adressage, qu'ils ne commanderont pas AC au niveau bas avant ce délai après que le MAITRE a permis à PAS* de passer au niveau haut.

32. OBSERVATION 2.35:
Les MAITRES sont assurés que les ESCLAVES qui n'ont pas atteint les limites de leur intervalle d'adressage commanderont AC au niveau bas dans ce délai après le passage de PAS* au niveau haut.

33. OBSERVATION 2.36:
Pendant les cycles de lecture et les cycles de RECONNAISSANCE D'INTERRUPTION, le MAITRE actif est assuré que l'ESCLAVE répondant ne modifiera pas AD00-AD31 avant ce délai après que le MAITRE a permis à PAS* de passer au niveau haut.

23. OBSERVATION 2.33:
During read data transfers and STATUS/ID transfers, the active MASTER is guaranteed that the responding SLAVE will drive AD00-AD31 with valid data at least this time before driving ACK* and/or ERR* low.

24. RULE 2.52:
The active MASTER MUST NOT allow either PAS* or DS* to go high until this time after ACK* and/or ERR* are low.

25. RULE 2.53:
During SINGLE-TRANSFER cycles and BLOCK-TRANSFER cycles, the active MASTER MUST NOT allow either PAS* or DS* to go high until this time after it detects WAIT* high.

26. RULE 2.54:
The active MASTER MUST maintain PAS* low for this minimum time.

27. RULE 2.55:
During ADDRESS-ONLY cycles and INTERRUPT-ACKNOWLEDGE cycles, the active MASTER MUST NOT allow PAS* to go high until this time after detecting AC high.

28. RULE 2.56:
During write cycles, the active MASTER MUST NOT change AD00-AD31 until this time after allowing PAS* to go high.

29. RULE 2.57:
During ADDRESS-ONLY cycles and write cycles, the active MASTER MUST stop driving AD00-AD31 within this time after allowing PAS* to go high.

30. RULE 2.58:
The MASTER MUST NOT change the levels of any of LOCK*, WR*, SIZE0-SIZE1 or SPACE0-SPACE1 until this time after allowing PAS* to go high.

31. OBSERVATION 2.34:
The active MASTER is guaranteed that SLAVES will not release ASACK0*-ASACK1* to high, or, if they have not reached the end of their self addressing range, will not drive AC to low until this time after the MASTER allows PAS* to go high.

32. OBSERVATION 2.35:
MASTERS are guaranteed that SLAVES who have not reached their self addressing range will drive AC low within this time after PAS* is high.

33. OBSERVATION 2.36:
During read cycles and INTERRUPT-ACKNOWLEDGE cycles, the active MASTER is guaranteed that the responding SLAVE will not change AD00-AD31 until this time after the MASTER allows PAS* to go high.

34. OBSERVATION 2.37:
Les MAITRES sont assurés que l'ESCLAVE répondant libérera CACHE* au niveau haut et que les ESCLAVES participants commanderont WAIT* au niveau bas dans ce délai après que le MAITRE actif a permis à PAS* de passer au niveau haut.

35. OBSERVATION 2.38:
Les MAITRES sont assurés que l'ESCLAVE répondant libérera CACHE* au niveau haut et que les ESCLAVES participants commanderont WAIT* au niveau bas dans ce délai après que le MAITRE actif a permis à DS* de passer au niveau haut.

36. OBSERVATION 2.39:
Le MAITRE actif est assuré que l'ESCLAVE répondant ne libérera ni ACK* ni ERR* au niveau haut avant ce délai après que le MAITRE a permis à DS* de passer au niveau haut.

37. OBSERVATION 2.40:
Après les transferts de données en lecture, les MAITRES sont assurés que l'ESCLAVE répondant cessera de commander AD00-AD31 dans ce délai après avoir libéré soit ASACK0*, soit ASACK1* au niveau haut.

38. REGLE 2.59:
Pendant les cycles de TRANSFERT PAR BLOC en écriture, le MAITRE actif NE DOIT PAS modifier AD00-AD31 avant ce délai après avoir permis à DS* de passer au niveau haut.

39. OBSERVATION 2.41:
Pendant les cycles de TRANSFERT PAR BLOC en lecture et les cycles de RECONNAISSANCE D'INTERRUPTION qui comprennent des transferts de MOT D'ETAT/ID multiples, le MAITRE actif et les ESCLAVES participants sont assurés que l'ESCLAVE répondant ne modifiera pas AD00-AD31 avant ce délai après que le MAITRE actif a permis à DS* de passer au niveau haut.

40. REGLE 2.60:
Un MAITRE NE DOIT PAS commander DS* au niveau bas s'il n'est pas au niveau haut au minimum depuis ce temps.

41. REGLE 2.61:
Le MAITRE actif NE DOIT PAS commander DS* au niveau bas avant ce délai après que ACK* et ERR* sont passés tous deux au niveau haut.

42. OBSERVATION 2.42:
Pendant les cycles de TRANSFERT PAR BLOC, le MAITRE actif est assuré que l'ESCLAVE répondant ne commandera pas CACHE* au niveau bas avant ce délai après que le MAITRE a commandé DS* au niveau bas.

43. OBSERVATION 2.43:
Pendant les cycles de TRANSFERT PAR BLOC, le MAITRE actif est assuré que les ESCLAVES participants ne libéreront pas WAIT* au niveau haut avant ce délai après que le MAITRE a commandé DS* au niveau bas.

44. REGLE 2.62:
Un MAITRE NE DOIT PAS commander PAS* au niveau bas s'il n'est pas au niveau haut au minimum depuis ce temps.

34. OBSERVATION 2.37:
MASTERS are guaranteed that the responding SLAVE will release CACHE* to high, and that participating SLAVES will drive WAIT* low, within this time after the active MASTER allows PAS* to go high.

35. OBSERVATION 2.38:
MASTERS are guaranteed that the responding SLAVE will release CACHE* to high, and that participating SLAVES will drive WAIT* low, within this time after the active MASTER allows DS* to go high.

36. OBSERVATION 2.39:
The active MASTER is guaranteed that the responding SLAVE will not release either ACK* or ERR* to high until this time after the MASTER allows DS* to go high.

37. OBSERVATION 2.40:
After read data transfers, MASTERS are guaranteed that the responding SLAVE will stop driving AD00-AD31 within this time after it releases either ASACK0* or ASACK1* to high.

38. RULE 2.59:
During BLOCK-TRANSFER write cycles, the active MASTER MUST NOT change any of AD00-AD31 until this time after allowing DS* to go high.

39. OBSERVATION 2.41:
During BLOCK-TRANSFER read cycles, and INTERRUPT-ACKNOWLEDGE cycles that include multiple STATUS/ID transfers, the active MASTER, and participating SLAVES are guaranteed that the responding SLAVE will not change any of AD00-AD31 until this time after the active MASTER allows DS* to go high.

40. RULE 2.60:
A MASTER MUST NOT drive DS* low until it has been high for this minimum time.

41. RULE 2.61:
The active MASTER MUST NOT drive DS* low until this time after both ACK* and ERR* are high.

42. OBSERVATION 2.42:
During BLOCK-TRANSFER cycles, the active MASTER is guaranteed that the responding SLAVE will not drive CACHE* low until this time after the MASTER has driven DS* low.

43. OBSERVATION 2.43:
During BLOCK-TRANSFER cycles, the active MASTER is guaranteed that participating SLAVES will not release WAIT* to high until this time after the MASTER has driven DS* low.

44. RULE 2.62:
A MASTER MUST NOT drive PAS* low until it has been high for this minimum time.

45. REGLE 2.63:
Le MAITRE actif NE DOIT PAS commander AD00-AD31 pendant ce temps après que ASACK0* et ASACK1* sont passés tous deux au niveau haut.

46. REGLE 2.64:
Un MAITRE NE DOIT PAS commander PAS* au niveau bas si ASACK0* et ASACK1* ne sont pas tous deux au niveau haut au minimum depuis ce temps.

47. REGLE 2.65:
SI le MAITRE actif permet à PAS* de passer au niveau haut après que son DEMANDEUR a libéré BUSY* au niveau haut,
ALORS il DOIT cesser de commander LOCK*, WR*, SIZE0-SIZE1, SPACE0-SPACE1, PAS*, AD00-AD31 et DS* dans ce délai après avoir permis à PAS* de passer au niveau haut.

48. REGLE 2.66:
SI le MAITRE actif permet à PAS* de passer au niveau haut avant que son DEMANDEUR libère BUSY* au niveau haut,
ALORS il DOIT cesser de commander LOCK*, WR*, SIZE0-SIZE1, SPACE0-SPACE1, PAS*, AD00-AD31 et DS* dans ce délai avant de permettre à son DEMANDEUR de libérer BUSY* au niveau haut.

49. REGLE 2.67:
Quand il prend le contrôle du VSB, le MAITRE actif NE DOIT PAS commander LOCK*, WR*, SIZE0-SIZE1, SPACE0-SPACE1, PAS*, AD00-AD31 ou DS* avant ce délai après que le MAITRE précédent a permis à PAS* de passer au niveau haut.

50. REGLE 2.68:
Pendant les cycles de RECONNAISSANCE D'INTERRUPTION, le MAITRE IHV actif DOIT cesser de commander AD24-AD30 au minimum pendant ce temps avant de commander PAS* au niveau bas.

54. OBSERVATION 2.44:
Pendant la phase de sélection d'un cycle de RECONNAISSANCE D'INTERRUPTION, le MAITRE IHV actif est assuré que AD24-AD30 ont fourni un ID INTERRUPTION valide au minimum pendant ce temps quand il détecte AC au niveau haut.

55. OBSERVATION 2.45:
Pendant la phase de sélection d'un cycle de RECONNAISSANCE D'INTERRUPTION, le MAITRE IHV actif est assuré que l'ID INTERRUPTION de l'ESCLAVE répondant reste valide sur AD24-AD30 au minimum pendant ce temps après que le MAITRE actif a reçu AC au niveau haut.

56. OBSERVATION 2.46:
Pendant les cycles de RECONNAISSANCE D'INTERRUPTION, le MAITRE IHV actif est assuré que AD24-AD30 sera libéré dans ce délai après qu'il a reçu AC au niveau haut.

57. REGLE 2.69:
Pendant les cycles de RECONNAISSANCE D'INTERRUPTION, le MAITRE IHV actif NE DOIT PAS commander DS* au niveau bas avant ce délai après qu'il a reçu AC au niveau haut.

45. RULE 2.63:
The active MASTER MUST NOT drive any of AD00-AD31 for this time after both ASACK0* and ASACK1* are high.

46. RULE 2.64:
A MASTER MUST NOT drive PAS* low until after both ASACK0* and ASACK1* are high for this minimum time.

47. RULE 2.65:
IF the active MASTER allows PAS* to go high after its REQUESTER releases BUSY* to high,
THEN it MUST stop driving all of LOCK*, WR*, SIZE0-SIZE1, SPACE0-SPACE1, PAS*, AD00-AD31 and DS* within this time after allowing PAS* to go high.

48. RULE 2.66:
IF the active MASTER allows PAS* to go high before its REQUESTER releases BUSY* to high,
THEN it MUST stop driving all of LOCK*, WR*, SIZE0-SIZE1, SPACE0-SPACE1, PAS*, AD00-AD31 and DS* within this time before allowing its REQUESTER to release BUSY* to high.

49. RULE 2.67:
When taking control of the VSB, the active MASTER MUST NOT drive any of LOCK*, WR*, SIZE0-SIZE1, SPACE0-SPACE1, PAS*, AD00-AD31 or DS* until this time after the previous MASTER allows PAS* to go high.

50. RULE 2.68:
During INTERRUPT-ACKNOWLEDGE cycles, the active IHV MASTER MUST stop driving AD24-AD30 this minimum time before driving PAS* low.

54. OBSERVATION 2.44:
During the selection phase of an INTERRUPT-ACKNOWLEDGE cycle, the active IHV MASTER is guaranteed that AD24-AD30 have carried a valid INTERRUPT ID for this minimum time when it detects AC high.

55. OBSERVATION 2.45:
During the selection phase of an INTERRUPT-ACKNOWLEDGE cycle, the active IHV MASTER is guaranteed that the INTERRUPT ID of the responding SLAVE remains valid on AD24-AD30 for this minimum time after the active MASTER receives AC high.

56. OBSERVATION 2.46:
During INTERRUPT-ACKNOWLEDGE cycles, the active IHV MASTER is guaranteed that AD24-AD30 will be released within this time after it receives AC high.

57. RULE 2.69:
During INTERRUPT-ACKNOWLEDGE cycles, the active IHV MASTER MUST NOT drive DS* low until this time after it receives AC high.

Tableau 2-23
ESCLAVE
Spécifications de la chronologie

Note: La numérotation correspond aux paramètres de temps spécifiés dans les tableaux 2-20 et 2-21.

2. OBSERVATION 2.47:
Les ESCLAVES sont assurés que le MAITRE actif ne commandera pas PAS* au niveau bas si LOCK*, WR*, SIZE0-SIZE1, SPACE0-SPACE1 et AD00-AD31 ne sont pas valides au minimum depuis ce temps.

3. OBSERVATION 2.48:
Les ESCLAVES sont assurés que le MAITRE actif maintiendra une adresse valide sur AD00-AD31 au minimum pendant ce temps après avoir commandé PAS* au niveau bas.

4. REGLE 2.70:
Les ESCLAVES NE DOIVENT PAS commander ASACK0* et/ou ASACK1* au niveau bas avant ce délai après que PAS* a été au niveau bas.

5. REGLE 2.71:
Pendant les cycles dans lesquels l'ESCLAVE répondant commande à la fois ASACK0* et ASACK1* au niveau bas, il DOIT commander le second au niveau bas dans ce délai maximal après avoir commandé le premier au niveau bas.

6. REGLE 2.72:
L'ESCLAVE répondant NE DOIT PAS commander CACHE* au niveau bas avant ce délai après passage de PAS* au niveau bas.

7. REGLE 2.73:
L'ESCLAVE répondant NE DOIT PAS libérer AC au niveau haut avant ce délai après passage de PAS* au niveau bas.

8. REGLE 2.74:
Lorsqu'il acquitte la phase de diffusion d'adresse, l'ESCLAVE répondant DOIT libérer AC au niveau haut après avoir commandé ASACK0* et/ou ASACK1* au niveau bas, mais il NE DOIT PAS libérer AC au niveau haut avant ce délai après avoir commandé ASACK0* et/ou ASACK1* au niveau bas.

9. REGLE 2.75:
SI un ESCLAVE commande WAIT* au niveau bas,
ALORS il NE DOIT PAS le libérer au niveau haut avant ce délai après que PAS* a été au niveau bas.

10. OBSERVATION 2.49:
L'ESCLAVE répondant est assuré que le MAITRE actif maintiendra une adresse valide sur AD00-AD31 pendant ce temps après que l'ESCLAVE a commandé ASACK0* et/ou ASACK1* au niveau bas.

11. OBSERVATION 2.50:
SI un ESCLAVE commande WAIT* au niveau bas,
ALORS il est assuré que le MAITRE actif maintiendra une adresse valide sur AD00-AD31 pendant ce temps après que AC a été au niveau haut.

Table 2-23
SLAVE
Timing specifications

Note: The numbers correspond to the timing parameters specified in Tables 2-20 and 2-21.

2. OBSERVATION 2.47:
SLAVES are guaranteed that the active MASTER will not drive PAS* low until LOCK*, WR*, SIZE0-SIZE1, SPACE0-SPACE1 and AD00-AD31 have been valid for this minimum time.

3. OBSERVATION 2.48:
SLAVES are guaranteed that the active MASTER will maintain a valid address on AD00-AD31 for this minimum time after it has driven PAS* to low.

4. RULE 2.70:
SLAVES MUST NOT drive ASACK0* and/or ASACK1* to low until this time after PAS* is low.

5. RULE 2.71:
During cycles where the responding SLAVE drives both ASACK0* and ASACK1* low, it MUST drive the second one low within this maximum time after driving the first one low.

6. RULE 2.72:
The responding SLAVE MUST NOT drive CACHE* low until this time after PAS* is low.

7. RULE 2.73:
The responding SLAVE MUST NOT release AC to high until this time after PAS* is low.

8. RULE 2.74:
When acknowledging the address broadcast phase, the responding SLAVE MUST release AC to high after driving ASACK0* and/or ASACK1* low, but it MUST NOT release AC to high until this time after it has driven ASACK0* and/or ASACK1* low.

9. RULE 2.75:
IF a SLAVE drives WAIT* low,
THEN it MUST NOT release it to high until this time after PAS* is low.

10. OBSERVATION 2.49:
The responding SLAVE is guaranteed that the active MASTER will maintain a valid address on AD00-AD31 for this time after the SLAVE has driven ASACK0* and/or ASACK1* to low.

11. OBSERVATION 2.50:
IF a SLAVE drives WAIT* low,
THEN it is guaranteed that the active MASTER will maintain a valid address on AD00-AD31 until this time after AC is high.

12. REGLE 2.76:
SI L'ESCLAVE répondant commande CACHE* au niveau bas pendant le cycle,
ALORS il DOIT le faire au moins pendant ce temps avant de commander soit ACK* soit ERR* au niveau bas.

14. OBSERVATION 2.51:
Les ESCLAVES sont assurés que pendant les cycles de transfert en écriture, le MAITRE actif placera une donnée valide sur AD00-AD31 au moins pendant ce temps avant de commander DS* au niveau bas.

15. OBSERVATION 2.52:
Pendant tous les cycles de lecture et tous les cycles de RECONNAISSANCE D'INTERRUPTION, l'ESCLAVE répondant est assuré que le MAITRE actif ne commandera pas DS* au niveau bas avant ce délai après avoir libéré AD00-AD31.

16. REGLE 2.77:
Pendant tous les cycles de lecture et tous les cycles de RECONNAISSANCE D'INTERRUPTION, l'ESCLAVE répondant NE DOIT PAS commander AD00-AD31 avant ce délai après passage de DS* au niveau bas.

17. REGLE 2.78:
L'ESCLAVE répondant et les ESCLAVES participants NE DOIVENT PAS commander AC au niveau bas pour indiquer qu'ils ont atteint les limites de leur intervalle d'adressage avant ce délai après passage de DS* au niveau bas.

18A. REGLE 2.79:
SI l'ESCLAVE répondant commande AC au niveau bas pour indiquer qu'il a atteint les limites de son intervalle d'adressage,
ALORS il DOIT le faire au moins pendant ce temps avant de commander ACK* et/ou ERR* au niveau bas.

18B. REGLE 2.80:
SI un ESCLAVE participant commande AC au niveau bas pour indiquer qu'il a atteint les limites de son intervalle d'adressage,
ALORS il DOIT le faire au moins pendant ce temps avant de libérer WAIT* au niveau haut.

19. REGLE 2.81:
SI l'ESCLAVE répondant commande IRQ* au niveau bas pendant un transfert de données en lecture qui accède à son registre de MOT D'ETAT/ID ou pendant un cycle de RECONNAISSANCE D'INTERRUPTION,
ALORS il NE DOIT PAS libérer au niveau haut sa contribution au maintien de IRQ* avant ce délai après passage de DS* au niveau bas.

20. REGLE 2.82:
SI l'ESCLAVE répondant commande IRQ* au niveau bas pendant un transfert de données en lecture qui accède à son registre de MOT D'ETAT/ID ou pendant un cycle de RECONNAISSANCE D'INTERRUPTION,
ALORS il DOIT libérer au niveau haut sa contribution au maintien de IRQ* au moins pendant ce temps avant de commander soit ACK*, soit ERR* au niveau bas.

12. RULE 2.76:
IF the responding SLAVE drives CACHE* low during the cycle,

THEN it MUST do so at least this time before driving either ACK* or ERR* low.

14. OBSERVATION 2.51:
SLAVES are guaranteed that during write data transfers, the active MASTER will drive valid data on AD00-AD31 at least this time before driving DS* low.

15. OBSERVATION 2.52:
During all read cycles and INTERRUPT-ACKNOWLEDGE cycles, the responding SLAVE is guaranteed that the active MASTER will not drive DS* low until this time after releasing AD00-AD31.

16. RULE 2.77:
During all read cycles and INTERRUPT-ACKNOWLEDGE cycles, the responding SLAVE MUST NOT drive any of AD00-AD31 until this time after DS* is low.

17. RULE 2.78:
The responding SLAVE and participating SLAVES MUST NOT drive AC low to indicate that they have reached the end of their self addressing range until this time after DS* is low.

18A. RULE 2.79:
IF the responding SLAVE drives AC low to indicate that it has reached the end of its self addressing range,
THEN it MUST do so at least this time before driving ACK* and/or ERR* low.

18B. RULE 2.80:
IF a participating SLAVE drives AC low to indicate that it has reached the end of its self addressing range,

THEN it MUST do so at least this time before releasing WAIT* to high.

19. RULE 2.81:
IF the responding SLAVE is driving IRQ* low during the read data transfer that accesses its STATUS/ID register, or during an INTERRUPT-ACKNOWLEDGE cycle,

THEN it MUST NOT release its contribution to the IRQ* line to high until this time after DS* is low.

20. RULE 2.82:
IF the responding SLAVE is driving IRQ* low during the read data transfer that accesses its STATUS/ID register, or during an INTERRUPT-ACKNOWLEDGE cycle,

THEN it MUST release its contribution to the IRQ* line to high at least this time before driving either ACK* or ERR* to low.

21. REGLE 2.83:
L'ESCLAVE répondant NE DOIT PAS commander ACK* ou ERR* au niveau bas avant ce délai après passage de DS* au niveau bas.

22. REGLE 2.84:
Pendant les cycles dans lesquels l'ESCLAVE répondant commande à la fois ACK* et ERR* au niveau bas, il DOIT commander le second au niveau bas dans ce délai maximal après avoir commandé le premier au niveau bas.

23. REGLE 2.85:
Pendant les transferts de données en lecture et de MOT D'ETAT/ID, l'ESCLAVE répondant DOIT placer des données valides sur AD00-AD31 au moins pendant ce temps avant de commander ACK* ou ERR* au niveau bas.

24. OBSERVATION 2.53:
L'ESCLAVE répondant est assuré que le MAITRE actif ne permettra pas à PAS* ou DS* de passer au niveau haut tant que l'ESCLAVE commande ACK* et/ou ERR* au niveau bas.

25. OBSERVATION 2.54:
Pendant les cycles de TRANSFERT UNIQUE ou TRANSFERT PAR BLOC, les ESCLAVES sont assurés que le MAITRE actif ne permettra pas à DS* ou PAS* de passer au niveau haut avant ce délai après passage de WAIT* au niveau haut.

26. OBSERVATION 2.55:
Les ESCLAVES sont assurés que le MAITRE actif maintiendra PAS* au niveau bas au minimum pendant ce temps.

27. OBSERVATION 2.56:
Pendant les cycles UNIQUEMENT D'ADRESSAGE, les ESCLAVES sont assurés que le MAITRE actif ne permettra pas à PAS* de passer au niveau haut avant ce délai après avoir détecté AC au niveau haut.

28. OBSERVATION 2.57:
Pendant un cycle d'écriture, les ESCLAVES sont assurés que le MAITRE actif ne modifiera pas AD00-AD31 avant ce délai après avoir permis à PAS* de passer au niveau haut.

30. OBSERVATION 2.58:
L'ESCLAVE répondant est assuré que le MAITRE actif ne modifiera pas les niveaux de LOCK*, WR*, SIZE0-SIZE1 et SPACE0-SPACE1 avant ce délai après avoir permis à PAS* de passer au niveau haut.

31. REGLE 2.86:
L'ESCLAVE répondant NE DOIT PAS libérer ASACK0* et/ou ASACK1* au niveau haut ou, s'il maintient encore AC au niveau haut, il NE DOIT PAS commander AC au niveau bas avant ce délai après passage de PAS* au niveau haut.

32. REGLE 2.87:
SI les ESCLAVES maintiennent un niveau haut sur AC quand ils détectent PAS* au niveau haut,
ALORS ils DOIVENT le commander au niveau bas avant ce délai après passage de PAS* au niveau haut.

21. RULE 2.83:
The responding SLAVE MUST NOT drive either ACK* or ERR* low until this time after DS* is low.

22. RULE 2.84:
During cycles where the responding SLAVE drives both ACK* and ERR* low, it MUST drive the second one low within this maximum time after driving the first one low.

23. RULE 2.85:
During read data transfers and STATUS/ID transfers, the responding SLAVE MUST drive AD00-AD31 with valid data at least this time before driving either ACK* or ERR* low.

24. OBSERVATION 2.53:
The responding SLAVE is guaranteed that the active MASTER will not allow either PAS* or DS* to go high until the SLAVE drives ACK* and/or ERR* low.

25. OBSERVATION 2.54:
During SINGLE-TRANSFER and BLOCK-TRANSFER cycles, SLAVES are guaranteed that the active MASTER will not allow either DS* or PAS* to go high until this time after WAIT* is high.

26. OBSERVATION 2.55:
SLAVES are guaranteed that the active MASTER will maintain PAS* low for this minimum time.

27. OBSERVATION 2.56:
During ADDRESS-ONLY cycles, SLAVES are guaranteed that the active MASTER will not allow PAS* to go high until this time after it detects AC high.

28. OBSERVATION 2.57:
During write cycle, SLAVES are guaranteed that the active MASTER will not change AD00-AD31 until this time after it allows PAS* to go high.

30. OBSERVATION 2.58:
The responding SLAVE is guaranteed that the active MASTER will not change the levels of any of LOCK*, WR*, SIZE0-SIZE1 and SPACE0-SPACE1 until this time after it allows PAS* to go high.

31. RULE 2.86:
The responding SLAVE MUST NOT release ASACK0* and/or ASACK1* to high or, if it is still maintaining AC high, it MUST NOT drive AC to low until this time after PAS* is high.

32. RULE 2.87:
IF SLAVES maintain a high level on AC when they detect PAS* high,
THEN they MUST drive it to low within this time after PAS* is high.

33. REGLE 2.88:
Pendant les cycles de lecture et les cycles de RECONNAISSANCE D'INTERRUPTION, l'ESCLAVE répondant NE DOIT PAS modifier AD00-AD31 avant ce délai après passage de PAS* au niveau haut.

34. REGLE 2.89:
L'ESCLAVE répondant DOIT libérer CACHE* au niveau haut et les ESCLAVES participants DOIVENT commander WAIT* au niveau bas avant ce délai après passage de PAS* au niveau haut.

35. REGLE 2.90:
L'ESCLAVE répondant DOIT libérer CACHE* au niveau haut et les ESCLAVES participants DOIVENT commander WAIT* au niveau bas avant ce délai après passage de DS* au niveau haut.

36. REGLE 2.91:
L'ESCLAVE répondant NE DOIT PAS libérer ACK* ou ERR* au niveau haut avant ce délai après passage de DS* au niveau haut.

37. REGLE 2.92:
Pendant les transferts de données en lecture, l'ESCLAVE répondant DOIT cesser de commander AD00-AD31 dans ce délai après avoir libéré ASACK0* ou ASACK1* au niveau haut.

38. OBSERVATION 2.59:
Pendant les cycles de TRANSFERT PAR BLOC en écriture, les ESCLAVES sont assurés que le MAITRE actif ne modifiera pas AD00-AD31 avant ce délai après avoir permis à DS* de passer au niveau haut.

39. REGLE 2.93:
Pendant les cycles de TRANSFERT PAR BLOC en lecture et les cycles de RECONNAISSANCE D'INTERRUPTION qui comprennent des transferts de MOT D'ETAT/ID multiple, l'ESCLAVE répondant NE DOIT PAS modifier AD00-AD31 avant ce délai après passage de DS* au niveau haut.

40. OBSERVATION 2.60:
Les ESCLAVES sont assurés qu'aucun MAITRE ne commandera DS* au niveau bas s'il n'a pas été au niveau haut au minimum pendant ce temps.

41. OBSERVATION 2.61:
L'ESCLAVE répondant est assuré que le MAITRE actif ne commandera pas DS* au niveau bas avant ce délai après que l'ESCLAVE a libéré ACK* et ERR* au niveau haut.

42. REGLE 2.94:
Pendant les cycles de TRANSFERT PAR BLOC, l'ESCLAVE répondant NE DOIT PAS commander CACHE* au niveau bas avant ce délai après passage de DS* au niveau bas.

43. REGLE 2.95:
Pendant les cycles de TRANSFERT PAR BLOC, les ESCLAVES participants NE DOIVENT PAS libérer WAIT* au niveau haut avant ce délai après passage de DS* au niveau bas.

44. OBSERVATION 2.62:
Les ESCLAVES sont assurés qu'aucun MAITRE ne commandera PAS* au niveau bas s'il n'a pas été au niveau haut au minimum pendant ce temps.

33. RULE 2.88:
During read cycles and INTERRUPT ACKNOWLEDGE cycles, the responding SLAVE MUST NOT change AD00-AD31 until this time after PAS* is high.

34. RULE 2.89:
The responding SLAVE MUST release CACHE* to high, and participating SLAVES MUST drive WAIT* low, within this time after PAS* is high.

35. RULE 2.90:
The responding SLAVE MUST release CACHE* to high, and participating SLAVES MUST drive WAIT* low, within this time after DS* is high.

36. RULE 2.91:
The responding SLAVE MUST NOT release either ACK* or ERR* to high until this time after DS* is high.

37. RULE 2.92:
During read data transfers, the responding SLAVE MUST stop driving AD00-AD31 within this time after releasing either ASACK0* or ASACK1* to high.

38. OBSERVATION 2.59:
During BLOCK-TRANSFER write cycles, SLAVES are guaranteed that the active MASTER will not change any of AD00-AD31 until this time after allowing DS* to go high.

39. RULE 2.93:
During BLOCK-TRANSFER read cycles, and INTERRUPT-ACKNOWLEDGE cycles that include multiple STATUS/ID transfers, the responding SLAVE MUST NOT change any of AD00-AD31 until this time after DS* is high.

40. OBSERVATION 2.60:
SLAVES are guaranteed that no MASTER will drive DS* low until it has been high for this minimum time.

41. OBSERVATION 2.61:
The responding SLAVE is guaranteed that the active MASTER will not drive DS* low until this time after the SLAVE releases both ACK* and ERR* to high.

42. RULE 2.94:
During BLOCK-TRANSFER cycles, the responding SLAVE MUST NOT drive CACHE* low until this time after DS* is low.

43. RULE 2.95:
During BLOCK-TRANSFER cycles, participating SLAVES MUST NOT release WAIT* to high until this time after DS* is low.

44. OBSERVATION 2.62:
SLAVES are guaranteed that no MASTER will drive PAS* low until it has been high for this minimum time.

45. OBSERVATION 2.63:
Les ESCLAVES sont assurés que le MAITRE actif ne commandera pas AD00-AD31 avant ce délai après le passage de ASACK0* et ASACK1* au niveau haut.

46. OBSERVATION 2.64:
Les ESCLAVES sont assurés qu'aucun MAITRE ne commandera PAS* au niveau bas au minimum pendant ce temps après le passage de ASACK0* et ASACK1* au niveau haut.

50. OBSERVATION 2.65:
Pendant les cycles de RECONNAISSANCE D'INTERRUPTION, les ESCLAVES concurrents sont assurés que le MAITRE IHV actif libérera AD24-AD30 dans ce délai minimal avant de commander PAS* au niveau bas.

51. REGLE 2.96:
Pendant la phase de sélection d'un cycle de RECONNAISSANCE D'INTERRUPTION, les ESCLAVES concurrents NE DOIVENT PAS commander AD24-AD30 avant ce délai après passage de PAS* au niveau bas.

52. REGLE 2.97:
Pendant la phase de sélection d'un cycle de RECONNAISSANCE D'INTERRUPTION, les ESCLAVES concurrents NE DOIVENT PAS libérer WAIT* au niveau haut avant ce délai après avoir placé leur ID INTERRUPTION sur AD24-AD30.

OBSERVATION 2.66:
Lorsqu'ils détectent WAIT* au niveau haut, les ESCLAVES concurrents sont assurés que l'ID INTERRUPTION est valide sur AD24-AD30.

53. REGLE 2.98:
Les ESCLAVES concurrents NE DOIVENT PAS libérer AC au niveau haut au minimum pendant ce temps après avoir libéré WAIT* au niveau haut.

54. REGLE 2.99:
Pendant la phase de sélection d'un cycle de RECONNAISSANCE D'INTERRUPTION, les ESCLAVES concurrents DOIVENT placer un ID INTERRUPTION valide sur AD24-AD30 au minimum pendant ce temps avant de libérer AC au niveau haut.

55. REGLE 2.100:
Les ESCLAVES concurrents DOIVENT maintenir leur contribution à l'ID INTERRUPTION sur AD24-AD30 au minimum pendant ce temps après avoir libéré AC au niveau haut.

56. REGLE 2.101:
Pendant les cycles de RECONNAISSANCE D'INTERRUPTION, les ESCLAVES concurrents DOIVENT libérer AD24-AD30 avant ce délai après avoir libéré AC au niveau haut.

57. OBSERVATION 2.67:
Pendant les cycles de RECONNAISSANCE D'INTERRUPTION, l'ESCLAVE répondant est assuré que le MAITRE IHV actif ne commandera pas DS* au niveau bas avant ce délai après le passage de AC au niveau haut.

45. OBSERVATION 2.63:
SLAVES are guaranteed that the active MASTER will not drive any of AD00-AD31 for this time after both ASACK0* and ASACK1* are high.

46. OBSERVATION 2.64:
SLAVES are guaranteed that no MASTER will drive PAS* low for this minimum time after both ASACK0* and ASACK1* are high.

50. OBSERVATION 2.65:
During INTERRUPT-ACKNOWLEDGE cycles, contending SLAVES are guaranteed that the active IHV MASTER will release AD24-AD30 this minimum time before it drives PAS* low.

51. RULE 2.96:
During the selection phase of an INTERRUPT-ACKNOWLEDGE cycle, contending SLAVES MUST NOT drive any of AD24-AD30 until this time after PAS* is low.

52. RULE 2.97:
During the selection phase of an INTERRUPT-ACKNOWLEDGE cycle, contending SLAVES MUST NOT release WAIT* to high until this time after they drive their INTERRUPT ID on AD24-AD30.

OBSERVATION 2.66:
Contending SLAVES are guaranteed that the INTERRUPT ID is valid on AD24-AD30 when they detect WAIT* high.

53. RULE 2.98:
Contending SLAVES MUST NOT release AC to high for this minimum time after releasing WAIT* to high.

54. RULE 2.99:
During the selection phase of an INTERRUPT-ACKNOWLEDGE cycle, contending SLAVES MUST drive a valid INTERRUPT ID on AD24-AD30 for this minimum time before releasing AC to high.

55. RULE 2.100:
Contending SLAVES MUST maintain their contribution to the INTERRUPT ID on AD24-AD30 for this minimum time after releasing AC to high.

56. RULE 2.101:
During INTERRUPT-ACKNOWLEDGE cycles, contending SLAVES MUST release AD24-AD30 within this time after releasing AC to high.

57. OBSERVATION 2.67:
During INTERRUPT-ACKNOWLEDGE cycles, the responding SLAVE is guaranteed that the active IHV MASTER will not drive DS* low until this time after AC is high.

CHAPITRE 3: ARBITRAGE DU BUS DE TRANSFERT DE DONNEES DU VSB

3.1 *Introduction*

Parmi les ressources globales d'un système multiprocesseur, la principale est le bus de transfert de données, à travers lequel on accède à toutes les autres ressources globales. Par conséquent, tout système qui permet un fonctionnement multiprocesseur doit procurer une méthode d'allocation efficace pour le bus de transfert de données. Le bus VSB satisfait à cette exigence grâce à son bus d'arbitrage (voir figure 3-1). Les protocoles du bus d'arbitrage VSB assurent l'ordonnancement des demandes d'accès au bus de transfert de données (DTB) émanant de plusieurs MAITRES et affectent le bus à un seul MAITRE à la fois.

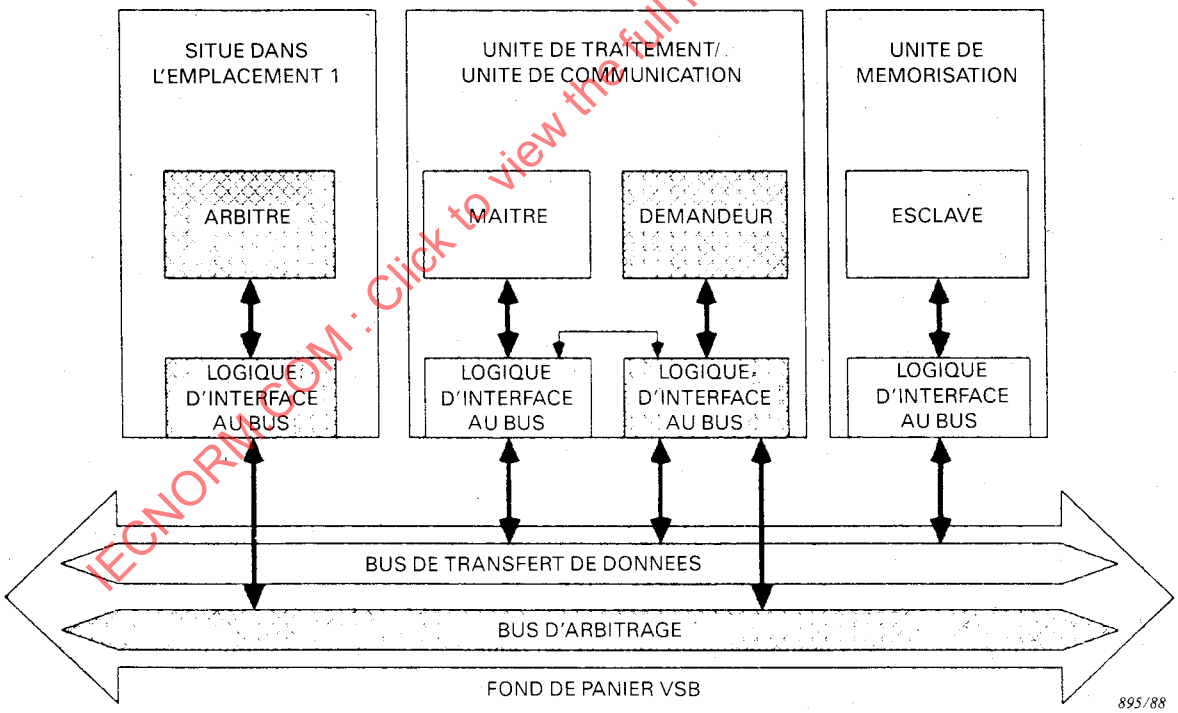


Fig. 3-1. - Schéma-bloc fonctionnel du bus d'arbitrage.

CHAPTER 3: VSB DATA TRANSFER BUS ARBITRATION

3.1 *Introduction*

The most fundamental of the global resources in a multiprocessing system is the Data Transfer Bus through which all other global resources are accessed. Therefore, any system that supports multiprocessing needs to provide an efficient allocation method for the Data Transfer Bus. The VSB meets this need with its Arbitration Bus (see Figure 3-1). The protocols of the VSB Arbitration Bus provide for the scheduling of requests for the DTB from multiple MASTERS and its assignment to only one MASTER at a time.

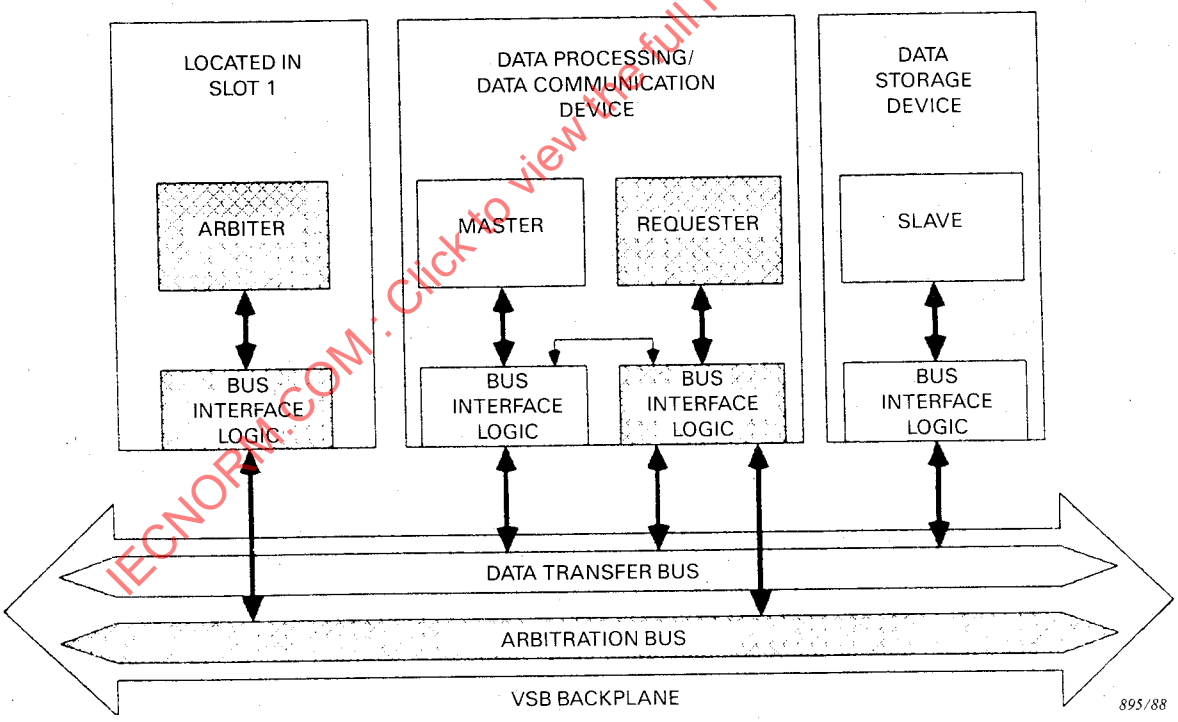


Fig. 3-1. - Arbitration Bus functional block diagram.

3.1.1 *Types d'arbitrage*

Le VSB définit deux méthodes d'arbitrage - une méthode d'arbitrage série et une méthode d'arbitrage parallèle.

Dans la méthode d'arbitrage série, le module DEMANDEUR requiert le bus pour le compte du MAITRE auquel il est associé. Il attend alors que sa demande soit acquittée et que le bus lui soit alloué par l'ARBITRE du système. Le paragraphe 3.4.1 décrit le mécanisme d'arbitrage série.

Dans la méthode d'arbitrage parallèle, le DEMANDEUR associé au MAITRE actif déclenche un cycle d'ARBITRAGE. Tous les DEMANDEURS qui ont une demande en attente participent à ce cycle. Pendant le cycle d'ARBITRAGE, les DEMANDEURS déterminent le MAITRE qui aura l'autorisation d'utiliser le DTB. Dans la norme VSB, le DEMANDEUR associé au MAITRE actif est appelé DEMANDEUR actif. Tous les DEMANDEURS qui ont une demande de bus en attente et qui participent à un cycle d'ARBITRAGE sont appelés les DEMANDEURS concurrents. La méthode d'arbitrage parallèle ne requiert pas d'ARBITRE. Le paragraphe 3.4.2 décrit le mécanisme d'arbitrage parallèle.

La méthode d'arbitrage utilisée est sélectionnée dynamiquement pendant la séquence de mise sous tension du sous-système VSB compte tenu des possibilités des DEMANDEURS du système. Le paragraphe 3.4.3 décrit cette séquence de démarrage.

3.2 *Lignes d'arbitrage du bus*

Les lignes du bus d'arbitrage coordonnent le transfert de la prise de contrôle du bus entre les DEMANDEURS du système. Le bus d'arbitrage comprend les lignes de commande suivantes:

BREQ* - demande du bus
BUSY* - occupation du bus
BGIN* - entrée d'allocation du bus
BGOUT* - sortie d'allocation du bus

OBSERVATION 3.1:
En plus des lignes de commande ci-dessus, le mécanisme d'arbitrage parallèle utilise certaines lignes du bus de transfert de données ainsi que les lignes d'adressage géographique.

3.2.1 *BREQ**

La ligne BREQ* est utilisée par les DEMANDEURS pour demander l'utilisation du DTB. Ils commandent BREQ* au niveau bas en réponse à une requête du DTB de la part de leur MAITRE associé.

3.2.2 *BUSY**

La ligne BUSY* est commandée au niveau bas par le DEMANDEUR actif pour informer l'ARBITRE, ainsi que les autres DEMANDEURS, que le contrôle du DTB a été pris par le MAITRE actif.

3.1.1 *Types of Arbitration*

The VSB defines two arbitration methods - a Serial Arbitration method and a Parallel Arbitration method.

In the Serial Arbitration method, a REQUESTER module requests the bus in behalf of its associated MASTER. It then waits for the request to be acknowledged and the bus to be granted by the system ARBITER. Paragraph 3.4.1 describes the Serial Arbitration mechanism.

In the Parallel Arbitration method, the REQUESTER that is associated with the active MASTER initiates an ARBITRATION cycle. All the REQUESTERS that have a request pending participate in the cycle. During the ARBITRATION cycle, the REQUESTERS determine which MASTER will be granted use of the DTB. The VSB standard calls the REQUESTER that is associated with the active MASTER the active REQUESTER. All REQUESTERS that have a bus request pending and who participate in an ARBITRATION cycle are called contending REQUESTERS. An ARBITER is not required in the Parallel Arbitration method. Paragraph 3.4.2 describes the Parallel Arbitration mechanism.

The arbitration method that is used is dynamically selected during the power-up sequence of the VSB subsystem, depending on the capabilities of the system's REQUESTERS. Paragraph 3.4.3 describes this power-up sequence.

3.2 *Arbitration Bus lines*

The lines of the Arbitration Bus coordinate the transfer of bus mastership between the system's REQUESTERS. The Arbitration Bus includes the following control lines:

BREQ* - Bus REQuest
BUSY* - Bus BUSY
BGIN* - Bus Grant IN
BGOUT* - Bus Grant OUT

OBSERVATION 3.1:
In addition to the above control lines, the Parallel Arbitration mechanism uses some of the Data Transfer Bus lines, as well as the geographical addressing lines.

3.2.1 *BREQ**

The BREQ* line is used by REQUESTERS to request use of the DTB. They drive BREQ* low in response to their associated MASTERS' need for the DTB.

3.2.2 *BUSY**

The BUSY* line is driven low by the active REQUESTER to inform the ARBITER, as well as other REQUESTERS, that control of the DTB has been assumed by the active MASTER.

3.2.3 *BGIN*/BGOUT**

Le processus d'arbitrage série requiert une ligne de signal formant une chaîne série. Le signal entrant dans chaque carte est appelé "Bus Grant IN" - BGIN*, alors que le signal sortant de chaque carte est appelé "Bus Grant OUT" - BGOUT*. Voir chapitre 5.

L'ARBITRE génère un front descendant sur la chaîne série d'allocation du bus pour autoriser l'utilisation du DTB. Ce niveau bas se propage sur la chaîne série. Si un DEMANDEUR a une demande de bus en attente, il ne transfère pas ce niveau bas de son entrée BGIN* vers sa sortie BGOUT*. Sinon, il propage le niveau bas en aval sur la chaîne série.

3.3 *Modules d'arbitrage - Description générale*

Les paragraphes 3.3.1 et 3.3.2 fournissent les schémas-blocs des deux modules définis dans l'arbitrage de bus: ARBITRE et DEMANDEUR.

REGLE 3.1:
Les lignes de signaux de sortie présentées en lignes continues dans les figures 3-2, 3-3 et 3-4, pages 194, 196 et 198, DOIVENT être commandées par le module, à moins qu'il les laisse toujours au niveau haut.

OBSERVATION 3.2:
SI - une ligne de sortie n'est pas commandée,
ALORS les impédances d'adaptation assurent qu'elle est au niveau haut.

REGLE 3.2:
Les lignes de signaux d'entrée présentées en lignes continues dans les figures 3-2, 3-3 et 3-4 DOIVENT être surveillées et acquittées de façon adéquate.

OBSERVATION 3.3:
Les REGLES et AUTORISATIONS pour commander et surveiller la ligne de signal présentée en pointillé dans la figure 3-3 sont données dans le tableau 3-1.

3.3.1 *ARBITRE*

Si l'arbitrage série est implanté, l'ARBITRE coordonne le transfert de la prise de contrôle du DTB. Un ARBITRE n'est pas nécessaire dans le mécanisme d'arbitrage parallèle.

OBSERVATION 3.4:
Le MAITRE qui réside sur la même carte que l'ARBITRE et qui demande le bus en positionnant le signal local LE MAITRE DEMANDE LE BUS est appelé le MAITRE associé à l'ARBITRE.

La figure 3-2 montre le schéma-bloc de l'ARBITRE.

3.2.3 *BGIN*/BGOUT**

The Serial arbitration scheme requires a daisy-chained signal line. The signal entering each board is called the "Bus Grant IN" - BGIN*, while the signal leaving each board is called the "Bus Grant OUT" - BGOUT*. See Chapter 5.

The ARBITER generates a falling edge on the bus grant daisy-chain to grant use of the DTB. This low level propagates down the daisy-chain. If a REQUESTER has a bus request pending, it does not pass this low level from its BGIN* input to its BGOUT* output. Otherwise, it passes the low level down the daisy-chain.

3.3 *Arbitration modules - Basic description*

Paragraphs 3.3.1 and 3.3.2 provide block diagrams for the two modules defined in the Arbitration Bus: ARBITER and REQUESTER.

RULE 3.1:
Output signal lines shown with solid lines in Figures 3-2, 3-3 and 3-4, pages 195, 197 and 199, MUST be driven by the module, unless it would always drive them high.

OBSERVATION 3.2:
IF an output signal line is not driven,
THEN terminators ensure that it is high.

RULE 3.2:
Input signal lines shown with solid lines in Figures 3-2, 3-3 and 3-4 MUST be monitored and responded to in the appropriate fashion.

OBSERVATION 3.3:
RULES and PERMISSIONS for driving and monitoring the signal line shown with dotted lines in Figure 3-3 are given in Table 3-1.

3.3.1 *ARBITER*

When implementing Serial Arbitration, the ARBITER coordinates the transfer of the mastership of the DTB. An ARBITER is not required in the Parallel Arbitration mechanism.

OBSERVATION 3.4:
The MASTER that resides on the same board with the ARBITER and requests the bus by asserting the on-board signal MASTER WANTS BUS is called the ARBITER'S associated MASTER.

Figure 3-2 shows the block diagram of the ARBITER.

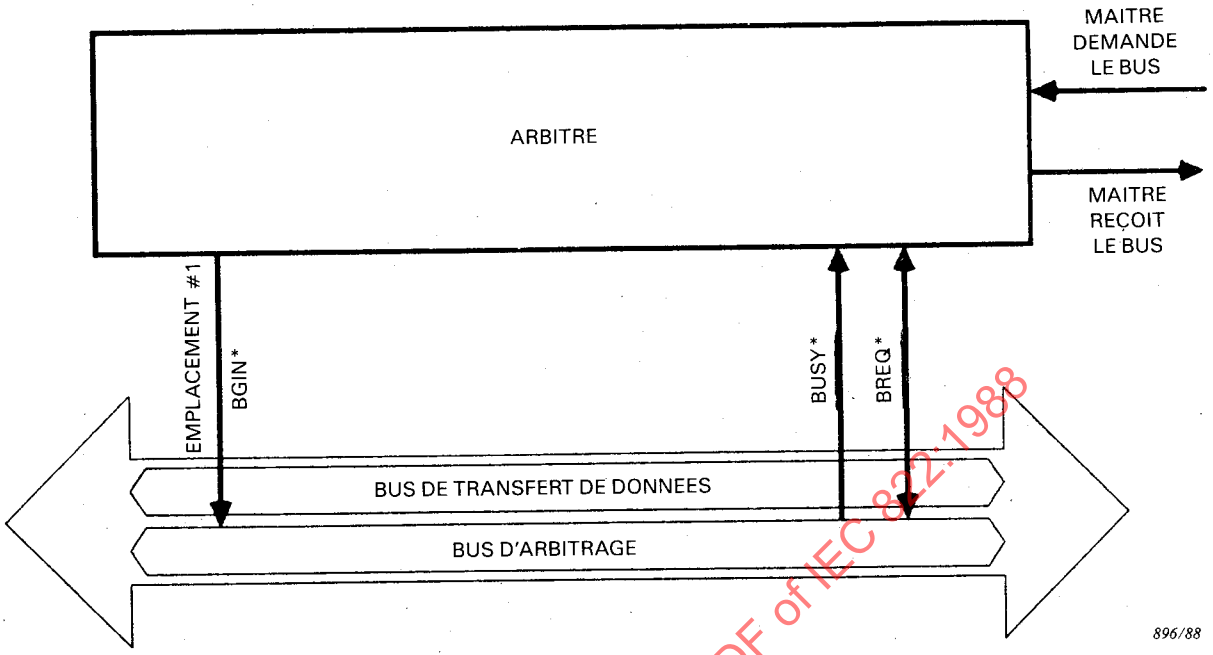


Fig. 3-2. - Schéma-bloc: ARBITRE.

3.3.2 *DEMANDEUR*

Le DEMANDEUR requiert l'utilisation exclusive du DTB pour le compte de son MAITRE associé. Quand la demande est acceptée, le DEMANDEUR prend le contrôle du DTB et signale à son MAITRE associé qu'il a obtenu l'usage du DTB. Après que son MAITRE associé a fini d'utiliser le DTB, le DEMANDEUR rend le DTB disponible pour les autres DEMANDEURS.

Le VSB définit deux types de DEMANDEURS: un DEMANDEUR série (SER) et un DEMANDEUR parallèle (PAR). Le schéma-bloc du DEMANDEUR SER est montré dans la figure 3-3, page 196. Le tableau 3-1 spécifie les contraintes de commande et de surveillance des lignes pointillées par les différents types de DEMANDEURS SER. La figure 3-4, page 198, montre le schéma-bloc du DEMANDEUR PAR.

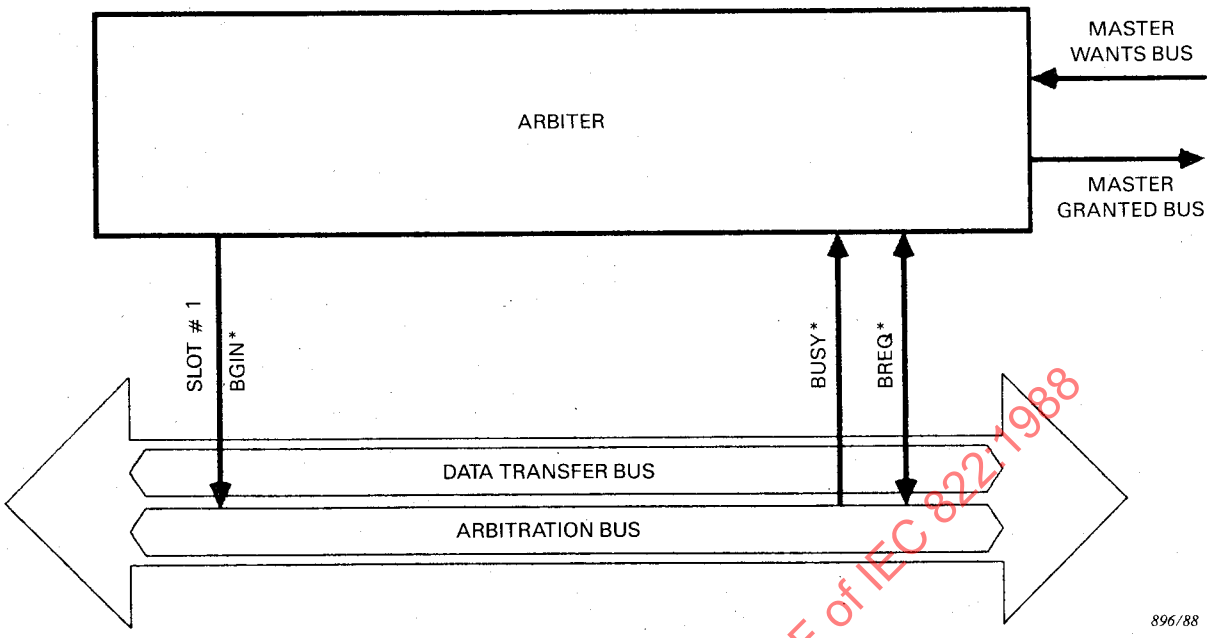


Fig. 3-2. - Block diagram: ARBITER.

3.3.2 *REQUESTER*

The REQUESTER requests exclusive use of the DTB on behalf of its associated MASTER. When this request is granted, the REQUESTER takes control of the DTB and signals its associated MASTER that it has been granted the DTB. After its associated MASTER finishes using the DTB, the REQUESTER makes the DTB available to other REQUESTERS.

The VSB defines two types of REQUESTERS: a serial (SER) REQUESTER and a parallel (PAR) REQUESTER. The block diagram of the SER REQUESTER is shown in Figure 3-3, page 197. Table 3-1 specifies the requirements for driving and monitoring the dotted signal lines by the various types of SER REQUESTERS. Figure 3-4, page 199, shows the block diagram of the PAR REQUESTER.

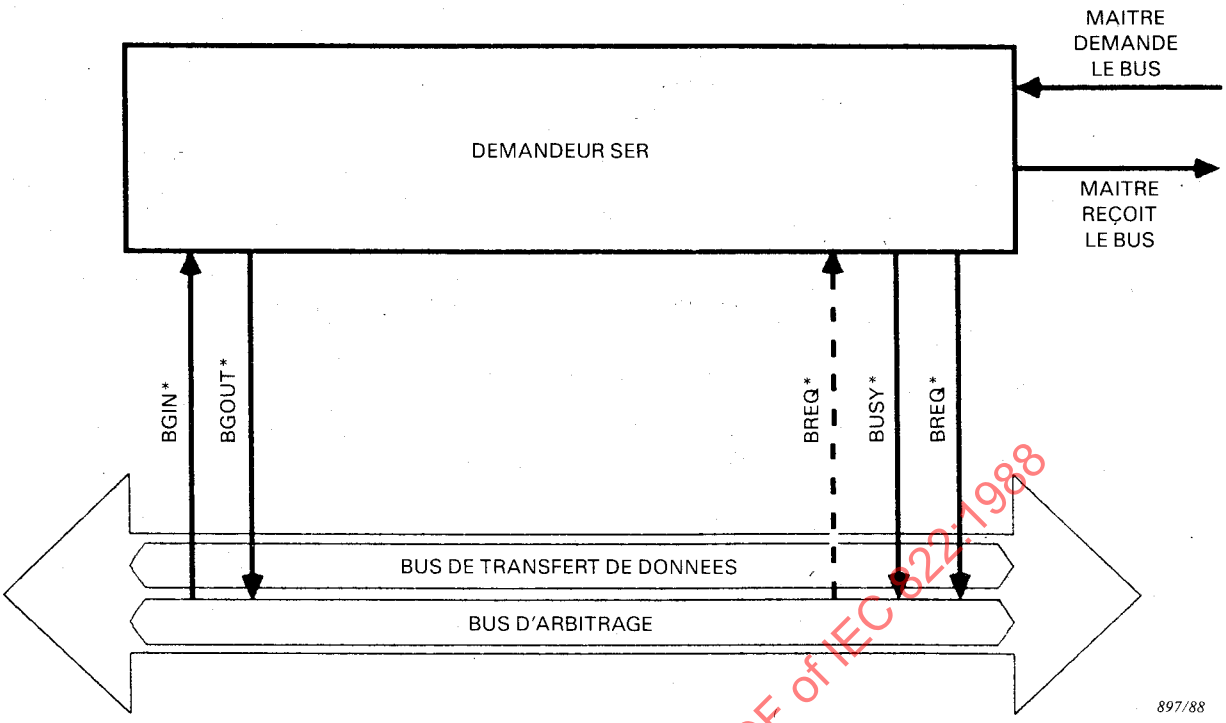


Fig. 3-3. - Schéma-bloc: DEMANDEUR SER.

Tableau 3-1

REGLES et AUTORISATIONS spécifiant l'utilisation des lignes pointillées par les différents types de DEMANDEURS SER

Type de DEMANDEUR	Utilisation de la ligne pointillée
RWD	PEUT ou NE PEUT PAS surveiller BREQ*
ROR	DOIT surveiller BREQ*

Note:
Les mnémoniques SER, RWD et ROR sont définis dans la section 3.4, tableau 3-2.

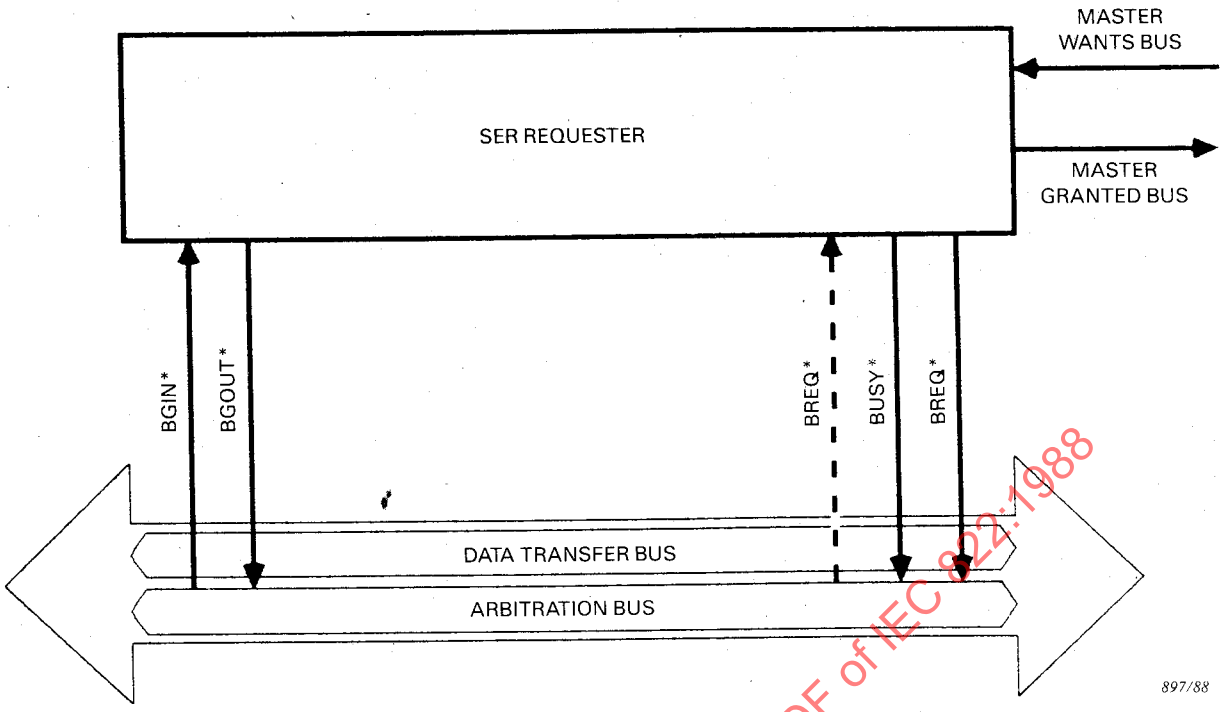


Fig. 3-3. - Block diagram: SER REQUESTER.

Table 3-1

RULES and PERMISSIONS that specify the use of the dotted lines by the various types of SER REQUESTERS

Type of REQUESTER	Use of dotted line
RWD	MAY or MAY NOT monitor BREQ*
ROR	MUST monitor BREQ*

Note:
The mnemonics SER, RWD and ROR are defined in Section 3.4, Table 3-2.

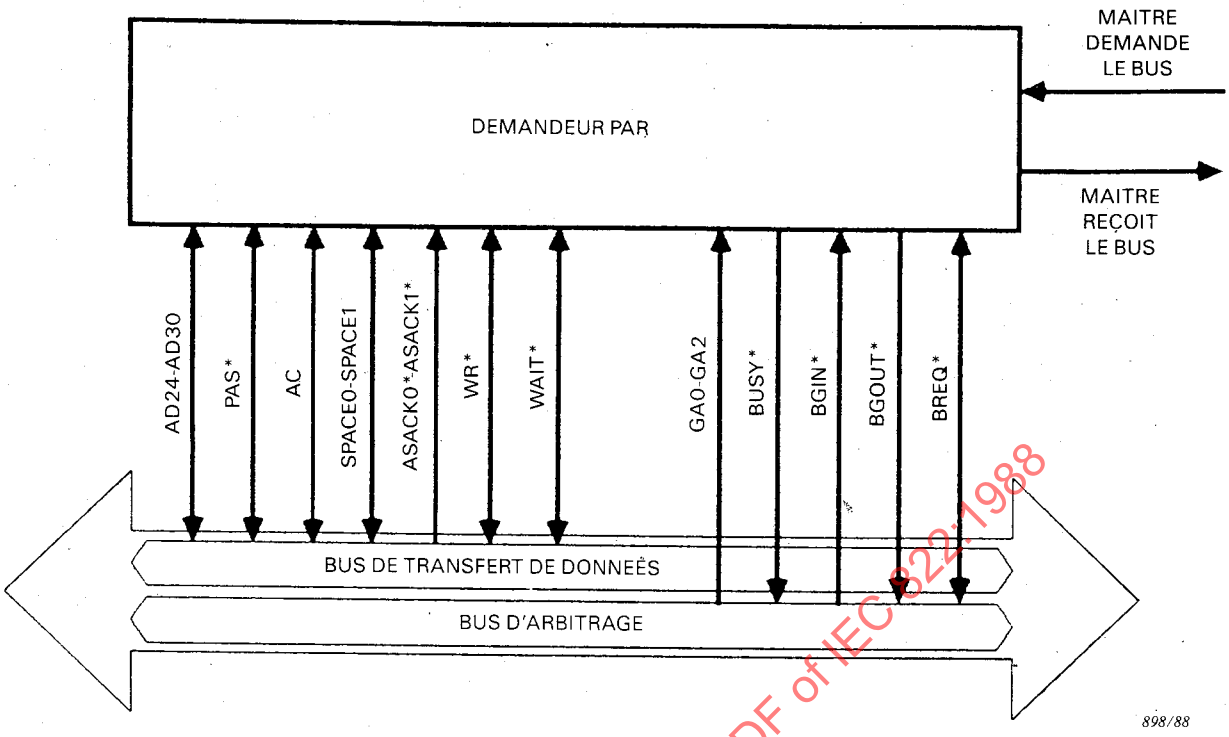


Fig. 3-4. - Schéma-bloc: DEMANDEUR PAR.

3.4 *Possibilités du DEMANDEUR*

Deux types de DEMANDEURS sont décrits dans cette norme: un DEMANDEUR série (SER) et un DEMANDEUR parallèle (PAR).

Les DEMANDEURS SER interagissent avec un ARBITRE pendant la séquence d'ARBITRAGE série pour coordonner l'utilisation du DTB. Deux types de DEMANDEURS SER sont définis: un DEMANDEUR de type "libération après exécution" (RWD) et un DEMANDEUR de type "libération sur demande" (ROR).

Le DEMANDEUR RWD libère BUSY* quand son MAITRE associé commande le signal local MAITRE DEMANDE LE BUS à l'état faux.

Par contre, le DEMANDEUR ROR ne libère pas BUSY* quand le signal local MAITRE DEMANDE LE BUS passe à l'état faux, à moins qu'un autre DEMANDEUR sur le bus ne commande BREQ* au niveau bas. Il surveille la ligne BREQ* et libère BUSY* au niveau haut seulement si une autre demande de bus est en attente. L'utilisation de DEMANDEURS ROR réduit le nombre d'arbitrages susceptibles d'être provoqués par un MAITRE ayant un pourcentage d'occupation du bus élevé.

Le DEMANDEUR PAR interagit avec les autres DEMANDEURS PAR du système pour coordonner l'utilisation du DTB. Cette interaction a lieu pendant le cycle d'ARBITRAGE qui est déclenché par le DEMANDEUR actif.

Le tableau 3-2 montre les mnémoniques utilisés pour décrire les DEMANDEURS.

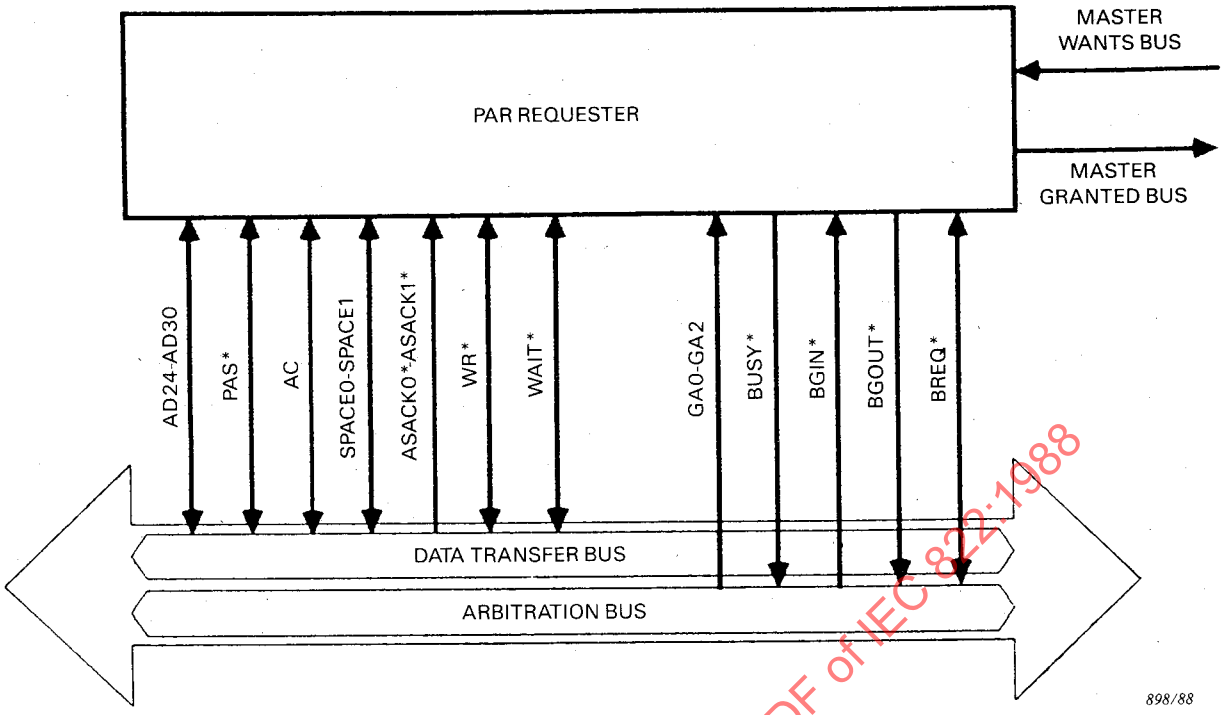


Fig. 3-4. - Block diagram: PAR REQUESTER.

3.4 *Capabilities of the REQUESTER*

Two types of REQUESTERS are described in this standard: a Serial (SER) REQUESTER and a Parallel (PAR) REQUESTER.

SER REQUESTERS interact with an ARBITER during the Serial ARBITRATION sequence to coordinate the use of the DTB. Two types of SER REQUESTERS are defined: a Release When Done (RWD) REQUESTER and a Release On Request (ROR) REQUESTER.

The RWD REQUESTER releases BUSY* when its associated MASTER drives the on-board signal MASTER WANTS BUS false.

The ROR REQUESTER does not release BUSY* when its on-board signal MASTER WANTS BUS goes false, unless some other REQUESTER on the bus drives BREQ* low. It monitors the BREQ* line, and releases BUSY* to high only if another bus request is pending. ROR REQUESTERS reduce the number of arbitrations initiated by a MASTER which is generating a large percentage of the bus traffic.

The PAR REQUESTER interacts with other PAR REQUESTERS in the system to coordinate use of the DTB. This interaction takes place during the ARBITRATION cycle which is initiated by the active REQUESTER.

Table 3-2 shows the mnemonics that are used to describe REQUESTERS.

Tableau 3-2

Mnémoniques utilisés pour décrire les DEMANDEURS

Le mnémonique suivant	Lorsqu'il est appliqué au	Signifie qu'il
SER	DEMANDEUR	Demande le bus en commandant BREQ* au niveau bas et attend un niveau bas sur BGIN* avant de commander BUSY* au niveau bas
PAR	DEMANDEUR	Demande le bus en commandant BREQ* au niveau bas, participe au cycle d'ARBITRAGE et, s'il obtient le bus, commande BUSY* au niveau bas. Il déclenche alors un cycle d'ARBITRAGE quand il détecte un niveau bas sur BREQ*, et après que son MAITRE n'a plus besoin du bus
ROR	DEMANDEUR SER	Maintient le contrôle du bus après que son MAITRE n'a plus besoin du bus, et le libère seulement si BREQ* est commandé au niveau bas
RWD	DEMANDEUR SER	Libère le bus lorsque son MAITRE associé n'a plus besoin du bus

REGLE 3.3:
Tous les DEMANDEURS PAR DOIVENT être dotés de la possibilité SER.

OBSERVATION 3.5:
L'exigence pour tous les DEMANDEURS PAR de pouvoir fonctionner comme des DEMANDEURS SER (c'est-à-dire: après avoir demandé le bus, attendre un niveau bas sur BGIN* avant de prendre le contrôle du bus) a été instituée pour assurer qu'un système VSB comprenant à la fois des DEMANDEURS SER et PAR fonctionne correctement. C'est la responsabilité de l'utilisateur de vérifier qu'un module ARBITRE est installé dans un système qui doit fonctionner en mode d'arbitrage série.

OBSERVATION 3.6:
Le paragraphe 3.4.3 décrit comment, pendant la séquence de mise sous tension, le système VSB détermine s'il doit opérer dans le mode d'arbitrage série ou parallèle.

3.4.1 *Arbitrage série*

Dans la méthode d'arbitrage série quand le DEMANDEUR SER détecte son signal local LE MAITRE DEMANDE LE BUS à l'état vrai, il commande BREQ* au niveau bas. Quand l'ARBITRE détecte ce niveau bas sur BREQ* et si ni son MAITRE associé, ni aucun autre DEMANDEUR n'a obtenu l'allocation du bus, il répond à cette demande par une allocation. Quand le DEMANDEUR reçoit cette allocation, il exécute trois actions:

Table 3-2

Mnemonics that are used to describe REQUESTERS

The following mnemonic	When appplied to a	Means that it
SER	REQUESTER	Requests the bus by driving BREQ* low and waits for a low level on BGIN* before driving BUSY* low
PAR	REQUESTER	Requests the bus by driving BREQ* low, participates in an ARBITRATION cycle and, if it wins, drives BUSY* low. It then initiates an ARBITRATION cycle when it detects a low on BREQ*, and after its MASTER no longer needs the bus
ROR	SER REQUESTER	Maintains bus mastership after its MASTER no longer needs the bus, and releases it only if BREQ* is driven low
RWD	SER REQUESTER	Releases the bus after its associated MASTER no longer needs the bus

RULE 3.3:
All PAR REQUESTERS MUST include SER capability.

OBSERVATION 3.5:
The requirement that all PAR REQUESTERS be capable to operate as SER REQUESTERS (i.e. after requesting the bus, wait for a low on BGIN* before assuming bus mastership) is instituted to ensure that a VSB system that includes a mixture of both SER and PAR REQUESTERS will operate properly. It is the user's responsibility to verify that an ARBITER module is installed in a system that is to operate in the Serial Arbitration mode

OBSERVATION 3.6:
Paragraph 3.4.3 describes how, during the power-up sequence, the VSB system determines whether to operate in the Serial or in the Parallel Arbitration mode.

3.4.1 *Serial Arbitration*

In the Serial Arbitration method, when the SER REQUESTER detects its on-board signal MASTER WANTS BUS true, it drives its BREQ* line low. When the ARBITER detects this low on BREQ*, and provided that neither its associated MASTER nor any other REQUESTER has been granted the bus, it grants this request. When the REQUESTER receives this grant, it does 3 things:

a) Il commande BUSY* au niveau bas.

b) Il supprime sa participation au maintien de la ligne BREQ* en la libérant au niveau haut.

c) Il commande son signal local LE DISPOSITIF RECOIT LE BUS à l'état vrai, permettant ainsi à son MAITRE associé de déclencher des cycles de bus.

3.4.1.1 *Interaction entre l'ARBITRE et les DEMANDEURS SER*

La figure 3-5, page 204, illustre la séquence d'événements qui a lieu lorsqu'un DEMANDEUR ROR et un DEMANDEUR RWD envoient des demandes simultanées à l'ARBITRE. Dans cet exemple, l'ARBITRE et le DEMANDEUR RWD sont sur la même carte dans l'emplacement 1, alors que le demandeur ROR se trouve dans l'emplacement 2.

La séquence commence lorsque les deux DEMANDEURS commandent simultanément BREQ* au niveau bas. L'ARBITRE commande alors BGIN* au niveau bas sur son propre emplacement (emplacement 1). Ce signal BGIN* est surveillé par le DEMANDEUR A (aussi à l'emplacement 1). Quand le DEMANDEUR A détecte BGIN* au niveau bas, il répond en commandant BUSY* au niveau bas. Le DEMANDEUR A libère alors BREQ* et informe le MAITRE A qu'il peut utiliser le DTB. Le DEMANDEUR B continue à commander BREQ* au niveau bas.

Après avoir détecté BUSY* au niveau bas, l'ARBITRE commande BGIN* au niveau haut. Lorsque le MAITRE A a terminé son transfert de données, il commande le signal MAITRE DEMANDE LE BUS à l'état faux. Lorsque le DEMANDEUR A détecte ce signal, il relâche BUSY* au niveau haut.

L'ARBITRE interprète la libération de BUSY* comme un signal d'arbitrage des demandes de bus en cours. Puisque la ligne BREQ* est encore au niveau bas, l'ARBITRE commande BGIN* à nouveau au niveau bas. Quand le DEMANDEUR A détecte BGIN* au niveau bas, il commande son signal BGOUT* au niveau bas puisque son MAITRE associé ne demande pas le DTB. Le DEMANDEUR B détecte alors le niveau bas sur son signal BGIN* et répond en commandant BUSY* au niveau bas, il supprime ainsi sa participation au maintien de la ligne BREQ* en la libérant au niveau haut et informant son MAITRE associé que le DTB est disponible. Lorsque l'ARBITRE détecte le niveau bas sur BUSY*, il commande BGIN* au niveau haut, ce qui conduit le DEMANDEUR A à commander un signal BGOUT* au niveau haut.

Quelque temps après, lorsque le MAITRE B a fini son transfert de données, il commande le signal MAITRE DEMANDE LE BUS à l'état faux, indiquant ainsi qu'il a fini d'utiliser le DTB.

Puisque le DEMANDEUR B est de type ROR, il ne libère pas BUSY* mais le maintient au niveau bas. Dans l'éventualité où le MAITRE B désire utiliser à nouveau le DTB, aucun autre arbitrage n'est nécessaire. Dans cet exemple, cependant, le DEMANDEUR A positionne BREQ* au niveau bas pour indiquer une demande d'utilisation de DTB et le DEMANDEUR B (qui surveille BREQ*) libère alors BUSY* au niveau haut. L'ARBITRE alloue alors le DTB au DEMANDEUR A.

a) It drives BUSY* to low.

b) It releases its contribution to the BREQ* line to high.

c) It drives its on-board signal DEVICE GRANTED BUS true, allowing its associated MASTER to initiate bus cycles.

3.4.1.1 *Interaction between the ARBITER and SER REQUESTERS*

Figure 3-5, page 205, illustrates the sequence of events that takes place when an ROR REQUESTER and an RWD REQUESTER send simultaneous requests to the ARBITER. In this example, the ARBITER and RWD REQUESTER are located on the board in slot 1, while the ROR REQUESTER is located in slot 2.

The sequence begins when both REQUESTERS drive BREQ* low simultaneously. The ARBITER then drives BGIN* low to its own slot (slot 1). That BGIN* signal is monitored by REQUESTER A (also in slot 1). When REQUESTER A detects BGIN* low, it responds by driving BUSY* low. REQUESTER A then releases BREQ* and informs MASTER A that it can use the DTB. REQUESTER B continues to drive BREQ* low.

After detecting BUSY* low, the ARBITER drives BGIN* high. When MASTER A has completed its data transfers, it drives MASTER WANTS BUS false. When REQUESTER A detects this it releases BUSY* to high.

The ARBITER interprets the release of BUSY* as a signal to arbitrate any current bus requests. Since the BREQ* line is still low, the ARBITER drives BGIN* low again. When REQUESTER A detects BGIN* low, it drives its BGOUT* low because its associated MASTER does not need the DTB. REQUESTER B then detects the low on its BGIN* and responds by driving BUSY* low, releasing its contribution to the BREQ* line to high, and informing its associated MASTER that the DTB is available. When the ARBITER detects the low on BUSY*, it drives BGIN* high, which causes REQUESTER A to drive its BGOUT* high.

Some time later, when MASTER B has finished its data transfers, it drives MASTER WANTS BUS false, indicating that it has finished using the DTB.

Since REQUESTER B is an ROR REQUESTER, it does not release BUSY*, but keeps it driven low. In the event that MASTER B needs to use the DTB again, no arbitration will be required. In this example, however, REQUESTER A drives BREQ* low, indicating a need to use the DTB, and REQUESTER B (which is monitoring BREQ*) releases BUSY* to high. The ARBITER then grants the DTB to REQUESTER A.

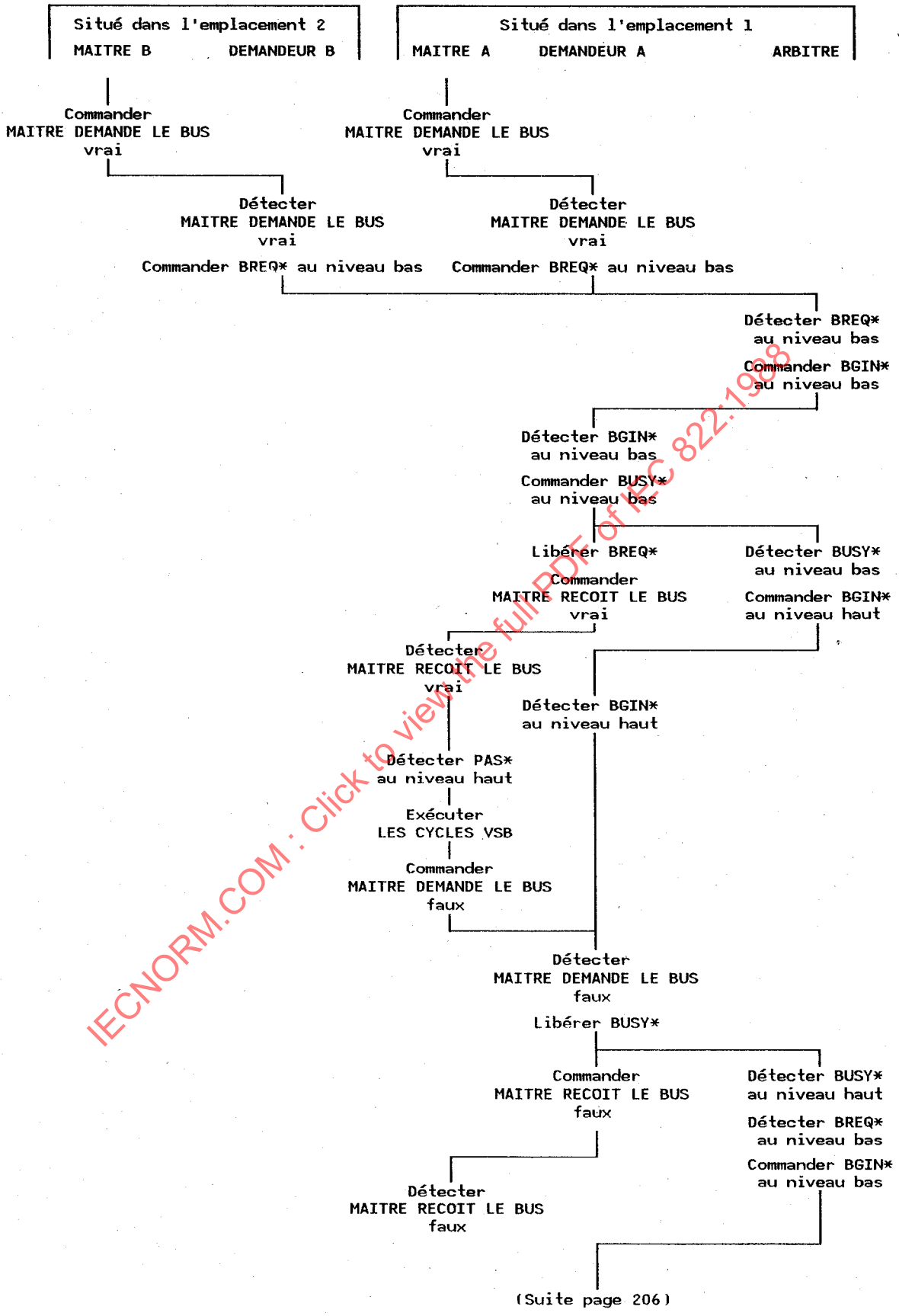


Fig. 3-5. - Organigramme de l'arbitrage série: deux DEMANDEURS.

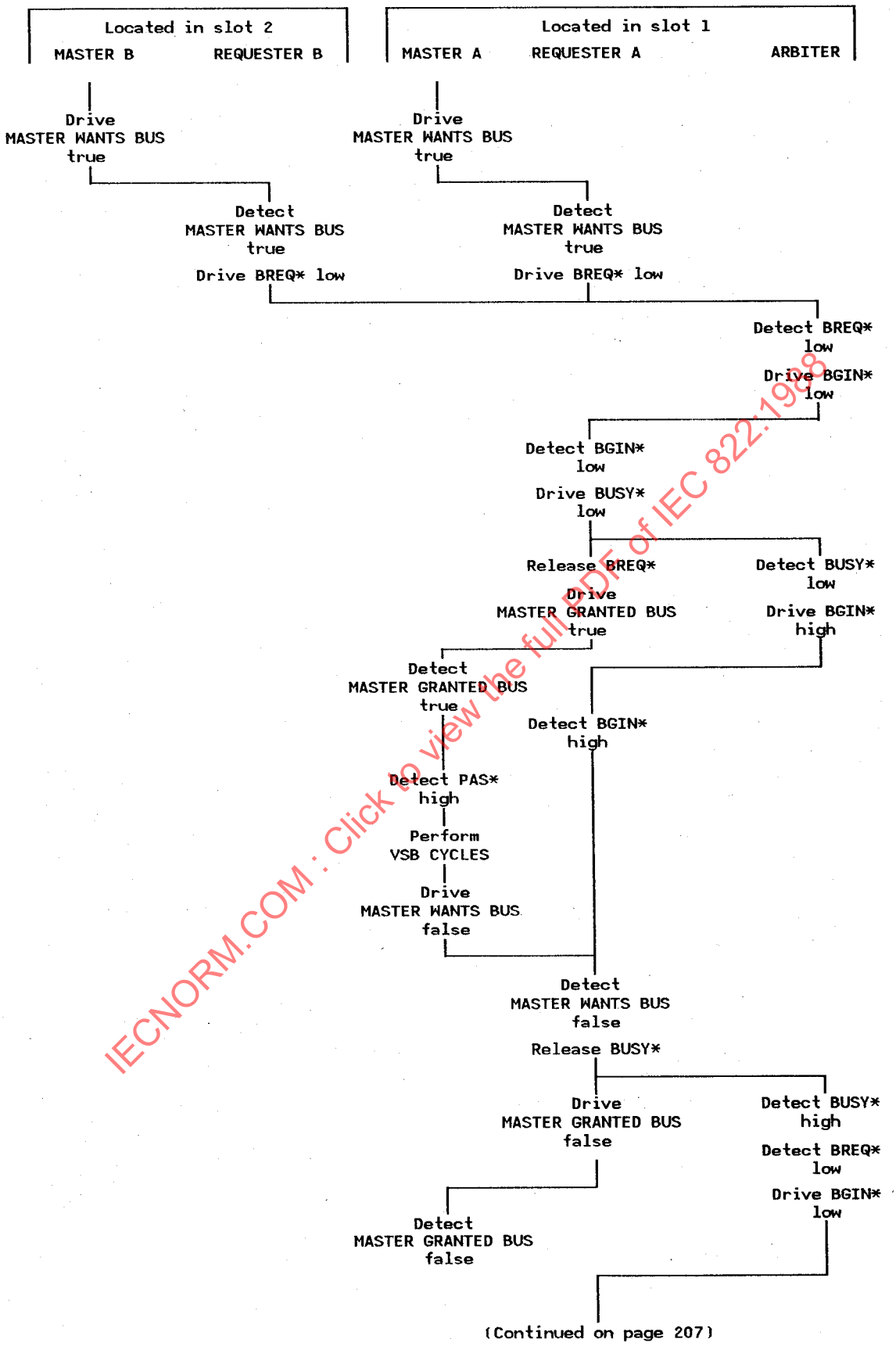


Fig. 3-5. - Serial Arbitration flow diagram: two REQUESTERS.

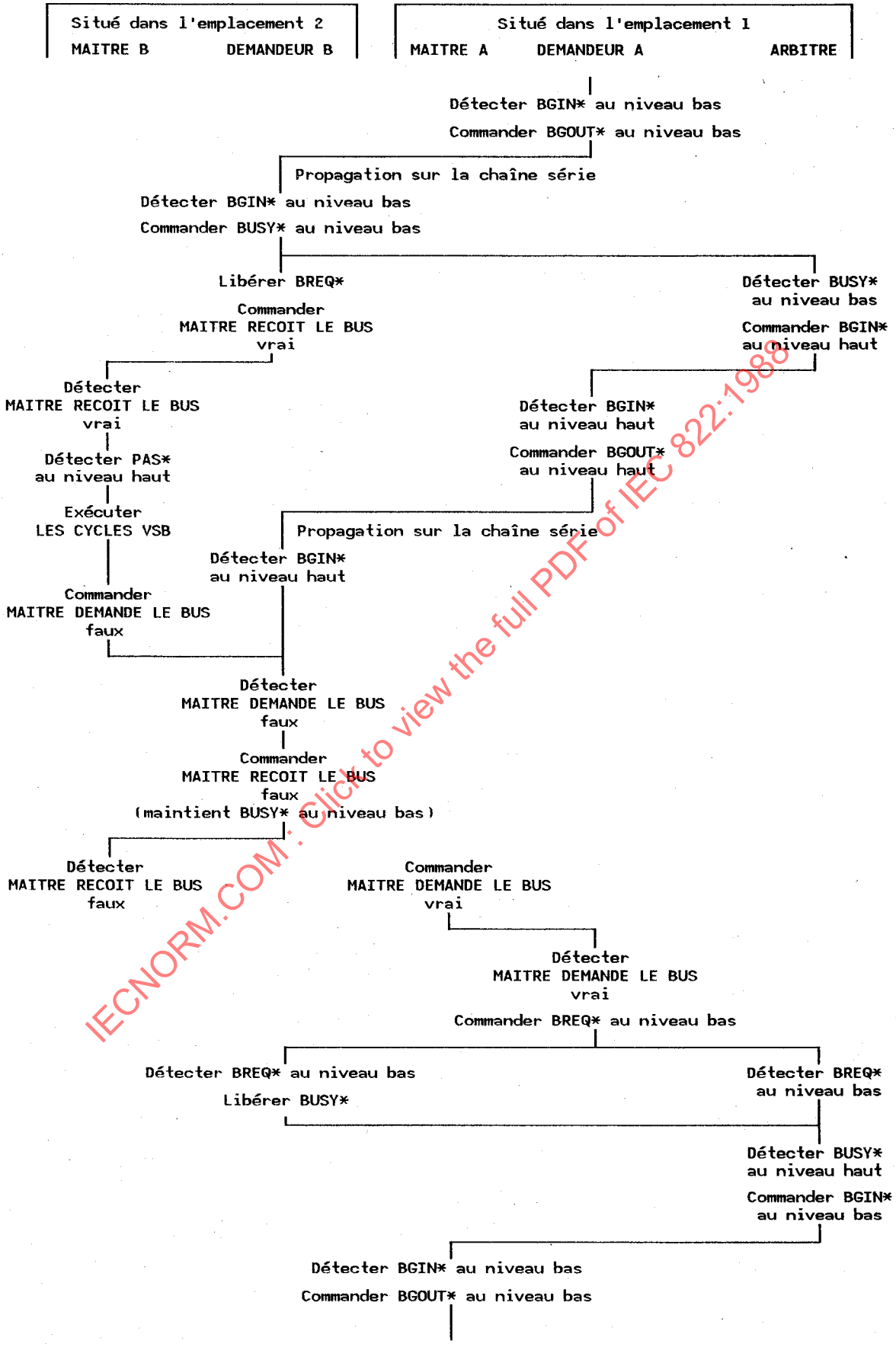


FIGURE 3-5 (fin).

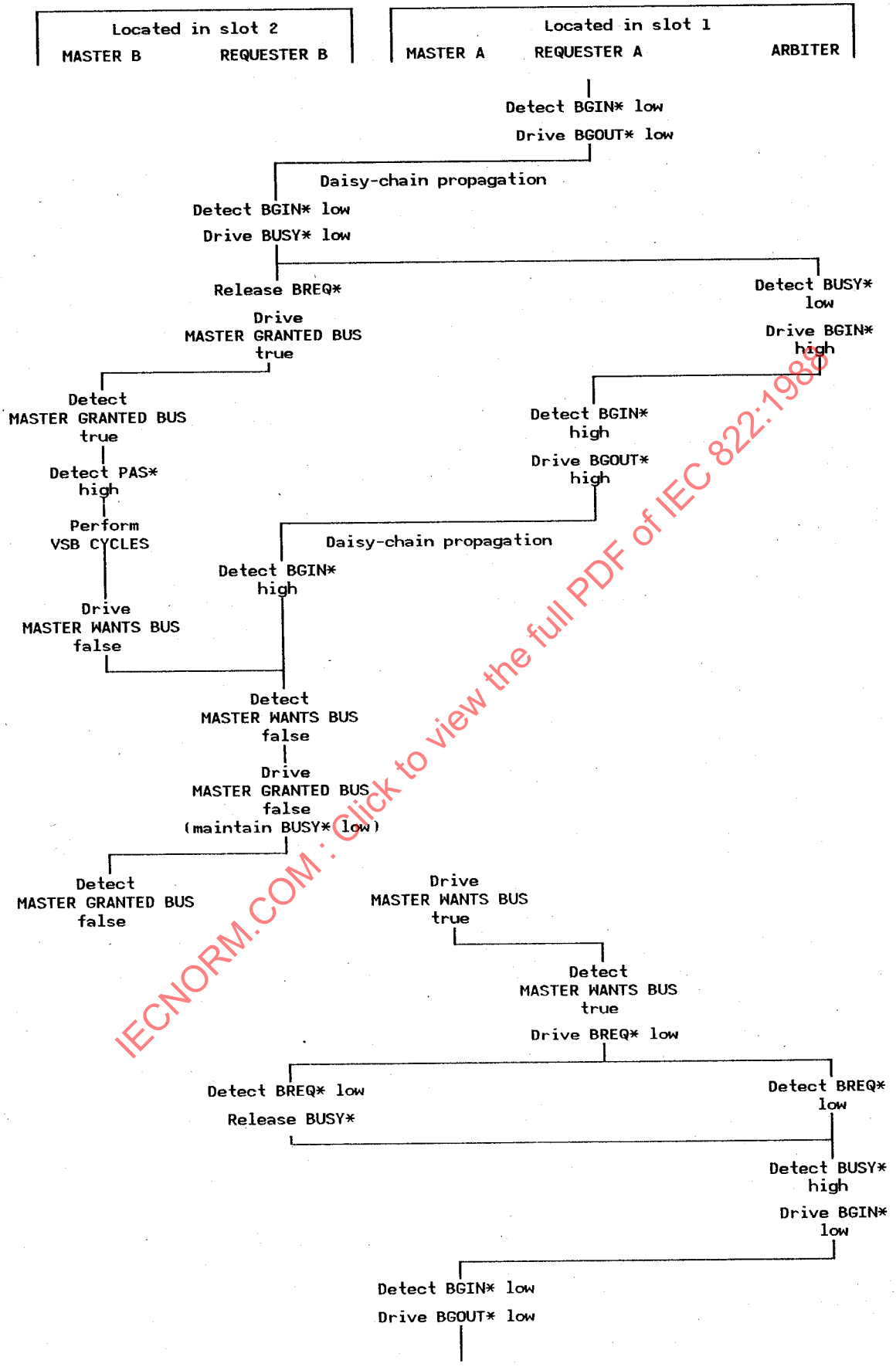


FIGURE 3-5 (concluded).

3.4.1.2 *Evolution des signaux pendant l'arbitrage série*

REGLE 3.4:
L'ARBITRE DOIT être situé dans l'emplacement 1.

REGLE 3.5:
SI la carte située dans l'emplacement 1 inclut un ARBITRE mais n'inclut pas de DEMANDEUR,
ALORS elle DOIT connecter la sortie BGIN* de l'ARBITRE à la sortie BGOUT* de la carte.

REGLE 3.6:
Chaque DEMANDEUR DOIT surveiller son signal local MAITRE DEMANDE LE BUS et DOIT commander BREQ* au niveau bas lorsqu'il détecte MAITRE DEMANDE LE BUS à l'état vrai.

REGLE 3.7:
Quand la requête du DEMANDEUR pour le bus est satisfaite, il DOIT commander son signal local MAITRE RECOIT LE BUS à l'état vrai pour indiquer à son MAITRE associé que le DTB est disponible.

AUTORISATION 3.1:
Le MAITRE qui réside sur la même carte que l'ARBITRE PEUT demander le bus en commandant le signal local MAITRE DEMANDE LE BUS sur la carte contenant l'ARBITRE.

REGLE 3.8:
SI l'ARBITRE détecte son signal local MAITRE DEMANDE LE BUS vrai,
ALORS il DOIT commander BREQ* au niveau bas et, après avoir détecté un niveau haut sur BUSY*, accorder le bus à son MAITRE associé en commandant son signal local MAITRE RECOIT LE BUS à l'état vrai, puis libérer sa contribution au maintien de BREQ* en lui permettant de passer au niveau haut.

REGLE 3.9:
SI l'ARBITRE détecte un niveau bas sur BREQ*,
ALORS il DOIT commander sa sortie BGIN* au niveau bas seulement après avoir détecté BUSY* au niveau haut, ainsi que son signal local MAITRE DEMANDE LE BUS à l'état faux.

REGLE 3.10:
SI un DEMANDEUR détecte BGIN* au niveau bas et ne demande pas le bus,
ALORS il DOIT propager ce niveau bas de son entrée BGIN* vers sa sortie BGOUT*.

REGLE 3.11:
Une carte VSB qui ne surveille pas BGIN* et/ou ne commande pas BGOUT* DOIT garantir la continuité de la chaîne série en assurant la connection sur la carte de ces deux signaux.

REGLE 3.12:
SI un emplacement de fond de panier n'est pas occupé par une carte et s'il y a d'autres cartes en aval dans la chaîne série,
ALORS un cavalier DOIT être installé dans l'emplacement vide pour propager le signal de la chaîne série.

3.4.1.2 *Signaling during Serial Arbitration*

RULE 3.4:
The ARBITER MUST be located in slot 1.

RULE 3.5:
IF the board that is located in slot 1 includes an ARBITER but does not include a REQUESTER,
THEN it MUST connect the ARBITER'S BGIN* output to the board's BGOUT* output.

RULE 3.6:
Each REQUESTER MUST monitor its on-board signal MASTER WANTS BUS and MUST drive BREQ* low when it detects MASTER WANTS BUS true.

RULE 3.7:
When the REQUESTER'S request for the bus is granted, it MUST drive its on-board signal MASTER GRANTED BUS true to indicate to its associated MASTER that the DTB is available.

PERMISSION 3.1:
The MASTER that resides on the same board with the ARBITER MAY request the bus by driving the ARBITER'S on-board signal MASTER WANTS BUS.

RULE 3.8:
IF the ARBITER detects its on-board signal MASTER WANTS BUS driven true,
THEN it MUST drive BREQ* low and, after detecting a high on BUSY*, grant the bus to its associated MASTER by driving its on-board signal MASTER GRANTED BUS true, and then release its contribution to BREQ* to high.

RULE 3.9:
IF the ARBITER detects a low level on BREQ*,
THEN it MUST drive its BGIN* output low, but only after it detects BUSY* driven high, as well as its on-board signal MASTER WANTS BUS driven false.

RULE 3.10:
IF a REQUESTER detects BGIN* low when it is not requesting the bus,
THEN it MUST propagate this low level from its BGIN* input to its BGOUT* output.

RULE 3.11:
A VSB board that does not monitor BGIN* and/or does not drive BGOUT* MUST continue the daisy-chain by providing an on-board connection of these two signals.

RULE 3.12:
IF a backplane slot is not occupied by a board, and if there are boards farther down the daisy-chain,
THEN a jumper MUST be installed at the empty slot to propagate the daisy-chain signal.

OBSERVATION 3.7:
La spécification mécanique du fond de panier du chapitre 5 décrit une possibilité d'installation d'un cavalier à chaque emplacement.

REGLE 3.13:
Dès qu'un DEMANDEUR SER a reçu le contrôle de bus de transfert des données via la chaîne série d'allocation, il DOIT commander BUSY* au niveau bas.

REGLE 3.14:
Dès qu'un DEMANDEUR SER a reçu l'allocation du bus et commande BUSY* au niveau bas, il DOIT supprimer sa participation au maintien de la ligne BREQ* en la libérant au niveau haut.

REGLE 3.15:
Les DEMANDEURS SER doivent maintenir BUSY* au niveau bas au moins 20 ns après avoir libéré BREQ* au niveau haut.

OBSERVATION 3.8:
Le délai de 20 ns entre le front montant sur BREQ* et le front montant sur BUSY* assure que l'ARBITRE ne peut interpréter par erreur la précédente demande de bus comme une nouvelle demande et émettre une autre allocation de bus.

REGLE 3.16:
Dès qu'un DEMANDEUR SER a obtenu l'allocation du bus et commande BUSY* au niveau bas, il DOIT le maintenir au niveau bas jusqu'à ce que son entrée BGIN* soit remontée au niveau haut.

OBSERVATION 3.9:
Cette REGLE assure que la transition au niveau bas de BUSY* a été détectée par l'ARBITRE et que tous les segments de la chaîne série d'allocation de bus sont remontés au niveau haut en vue de l'arbitrage suivant.

SUGGESTION 3.1:
Concevoir les DEMANDEURS SER de façon qu'ils transmettent sur la chaîne série le niveau bas entrant sur BGIN* vers leur sortie BGOUT* aussi rapidement que possible. Cela améliorera la performance du système.

OBSERVATION 3.10:
En raison de la chaîne série, les DEMANDEURS SER ont une priorité dépendant de leur emplacement. Le DEMANDEUR SER le plus près de l'emplacement 1 a la plus haute priorité.

RECOMMANDATION 3.1:
Afin d'éviter les verrouillages du système, concevoir l'ARBITRE avec un dispositif de dépassement de temps qui le conduira à retirer une allocation du bus s'il n'a pas reçu dans un temps prescrit un niveau bas sur la ligne BUSY*.

OBSERVATION 3.11:
La période de dépassement de temps utilisée par l'ARBITRE devra être plus longue que le temps le plus long de propagation possible de la chaîne série d'allocation, ajouté au temps de génération du signal BUSY* par le DEMANDEUR SER le plus lent.

REGLE 3.17:
Dès qu'un ARBITRE a accordé le bus et détecté un niveau bas sur BUSY*, il NE DOIT PAS générer un nouveau signal d'allocation tant que BUSY* n'est pas revenu au niveau haut, exception faite d'une situation de dépassement de temps où aucun DEMANDEUR SER ne répond.

OBSERVATION 3.7:
The backplane mechanical specification in Chapter 5 describes a provision for the installation of a jumper at each slot.

RULE 3.13:
Once a SER REQUESTER has been granted control of the Data Transfer Bus via the bus grant daisy-chain, it MUST drive BUSY* low.

RULE 3.14:
Once a SER REQUESTER has been granted the bus and drives BUSY* low, it MUST release its contribution to the BREQ* line to high.

RULE 3.15:
SER REQUESTERS MUST maintain BUSY* low for at least 20 ns after releasing BREQ* to high.

OBSERVATION 3.8:
The 20 ns delay between the rising edge on BREQ* and the rising edge on BUSY* ensures that the ARBITER does not mistakenly interpret the old bus request as a new one and issue another grant.

RULE 3.16:
Once a SER REQUESTER is granted the bus and drives BUSY* low, it MUST hold it low until its BGIN* input goes high.

OBSERVATION 3.9:
This RULE ensures that the transition of BUSY* to low has been detected by the ARBITER, and that all segments of the bus grant daisy-chain have returned to high in preparation for the next arbitration.

SUGGESTION 3.1:
Design SER REQUESTERS to pass the incoming low level on BGIN* to their BGOUT* daisy-chain output as fast as possible. This will improve system performance.

OBSERVATION 3.10:
Because of the daisy-chain, SER REQUESTERS are prioritized by slot position. The SER REQUESTER closest to slot 1 has the highest priority.

RECOMMENDATION 3.1:
To prevent lock-ups, design the ARBITER with a built-in time-out feature that causes it to withdraw a bus grant if it does not receive a low level on BUSY* within a prescribed time.

OBSERVATION 3.11:
The time-out period used by the ARBITER should be longer than the longest possible bus grant daisy-chain propagation delay time, plus the time the slowest SER REQUESTER takes to generate BUSY*.

RULE 3.17:
Except for a time-out situation where no SER REQUESTER responds, once an ARBITER grants the bus and detects a low level on BUSY*, it MUST NOT generate a new bus grant until after BUSY* goes high.

3.4.2 *Possibilités de l'arbitrage parallèle*

Au cours d'un cycle d'ARBITRAGE parallèle, les DEMANDEURS PAR interagissent les uns avec les autres pour déterminer lequel d'entre eux recevra l'allocation du DTB. A chaque DEMANDEUR PAR est assigné un ID ARBITRAGE unique qui est utilisé pour déterminer la priorité des demandes multiples de bus. L'ID ARBITRAGE est un code sur 7 bits constitué des 3 bits d'adresse géographique d'emplacement et de 4 bits de code définis et fournis par l'utilisateur. L'ID ARBITRAGE est utilisé pendant un cycle d'ARBITRAGE pour sélectionner le MAITRE qui aura l'utilisation du VSB par la suite.

La figure 3-6 montre l'organigramme général d'un cycle d'ARBITRAGE. Le cycle d'ARBITRAGE est très semblable à la phase de sélection d'un cycle de RECONNAISSANCE D'INTERRUPTION. Le cycle démarre quand le DEMANDEUR PAR actif détecte une demande sur le bus. Lorsque son MAITRE associé commande le signal MAITRE DEMANDE LE BUS à l'état faux, le DEMANDEUR PAR déclenche un cycle d'ARBITRAGE. Tous les DEMANDEURS PAR qui ont une demande en attente participent au cycle, à la fin duquel un des DEMANDEURS PAR est sélectionné comme le DEMANDEUR actif suivant. Il informe alors son MAITRE associé qu'il a obtenu l'allocation du bus et maintient le contrôle du bus (c'est-à-dire il commande BUSY* au niveau bas) jusqu'au moment où il déclenche un nouveau cycle d'ARBITRAGE. Cela termine le cycle d'ARBITRAGE.

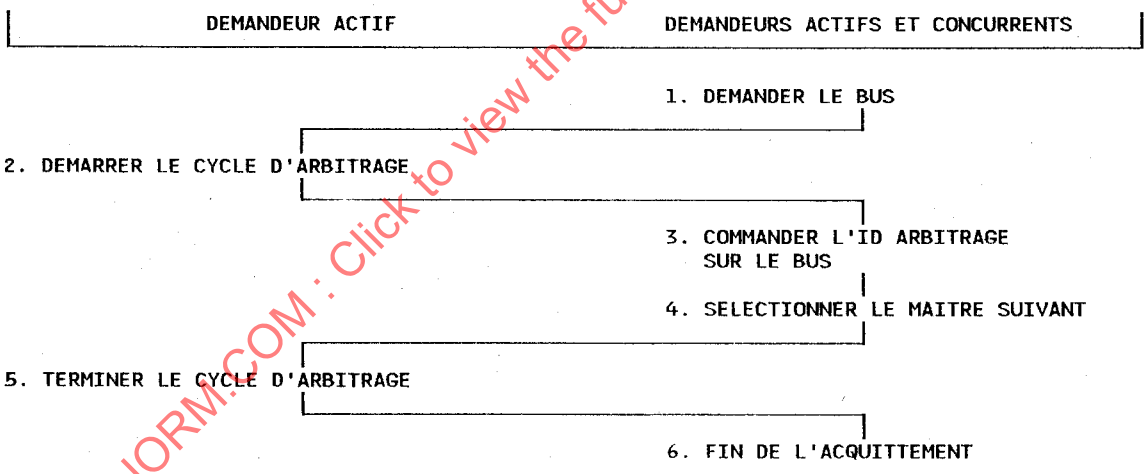


Fig. 3-6. - Organigramme général d'un cycle d'ARBITRAGE.

Le paragraphe 3.4.2.1 et la figure 3-7, page 216, montrent comment le DEMANDEUR PAR actif et les DEMANDEURS PAR concurrents interagissent pendant le cycle d'ARBITRAGE. Ce paragraphe fournit une description informelle du protocole pour familiariser le lecteur avec le déroulement du cycle d'ARBITRAGE. Le paragraphe 3.4.2.2 constitue la spécification formelle de ce protocole, alors que la section 3.6 spécifie les contraintes de chronologie.

3.4.2.1 *Déroulement d'un cycle d'ARBITRAGE*

Le déroulement d'un cycle d'ARBITRAGE est décrit dans la figure 3-7, page 216. La séquence d'ARBITRAGE commence à l'étape 1 quand un DEMANDEUR PAR commande BREQ* au niveau bas.

3.4.2 *Parallel Arbitration capability*

In the course of a Parallel ARBITRATION cycle, PAR REQUESTERS interact with each other to determine which one will be granted the DTB. Each PAR REQUESTER is assigned a unique ARBITRATION ID which is used to prioritize multiple bus requests. This ARBITRATION ID is a 7-bit code which is comprised of the 3-bit geographical slot address and a 4-bit user defined and supplied code. The ARBITRATION ID is used during an ARBITRATION cycle to select the MASTER which will use the VSB thereafter.

Figure 3-6 shows the general flow of an ARBITRATION cycle. The ARBITRATION cycle is very similar to the selection phase of an INTERRUPT-ACKNOWLEDGE cycle. The cycle starts when the active PAR REQUESTER detects a request on the bus. Provided that its associated MASTER drives MASTER WANTS BUS false, the PAR REQUESTER initiates an ARBITRATION cycle. All PAR REQUESTERS that have a bus request pending participate in the cycle, at the end of which one of the PAR REQUESTERS is selected as the next active REQUESTER. It then informs its associated MASTER that it is granted the bus, and then maintains control of the bus (i.e. drives BUSY* low) until it is time to initiate a new ARBITRATION cycle. This terminates the ARBITRATION cycle.

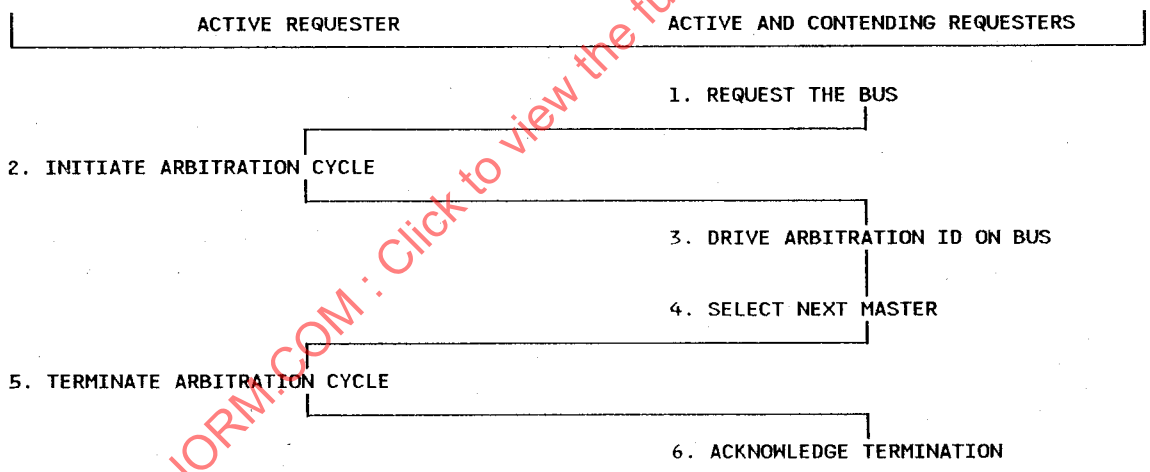


Fig. 3-6. - General flow of an ARBITRATION cycle.

Paragraph 3.4.2.1 and Figure 3-7, page 217, show how the active PAR REQUESTER and contending PAR REQUESTERS interact during the ARBITRATION cycle. This paragraph provides an informal description of the protocol to familiarize the reader with the flow of the ARBITRATION cycle. Paragraph 3.4.2.2 constitutes a formal specification of this protocol, while Section 3.6 specifies the timing requirements.

3.4.2.1 *Flow of an ARBITRATION cycle*

The flow of an ARBITRATION cycle is shown in Figure 3-7, page 217. The ARBITRATION sequence begins in step 1 when a PAR REQUESTER drives BREQ* low.

A l'étape 2, le DEMANDEUR PAR associé au MAITRE qui contrôle le bus (désigné comme le DEMANDEUR actif) reçoit BREQ* au niveau bas. Dès que le MAITRE associé n'utilise plus le bus et a positionné LE MAITRE DEMANDE LE BUS à l'état faux, le DEMANDEUR PAR actif déclenche un cycle d'ARBITRAGE. Il vérifie d'abord que son MAITRE associé ne commande plus les lignes de bus en détectant un niveau haut sur PAS*. Le DEMANDEUR PAR actif positionne alors SPACE0-SPACE1 et WR* au niveau bas pour indiquer qu'un cycle d'ARBITRAGE est en cours. Après avoir détecté un niveau haut sur ASACK0*-ASACK1*, signifiant que l'ESCLAVE qui a répondu au cycle précédent n'est plus sur le bus, le DEMANDEUR PAR actif commande PAS* au niveau bas. Le DEMANDEUR PAR actif libère alors BUSY* au niveau haut.

A l'étape 3, après avoir détecté un front descendant sur PAS*, tous les DEMANDEURS PAR ainsi que les ESCLAVES reçoivent SPACE0-SPACE1 et WR* au niveau bas et déterminent qu'un cycle d'ARBITRAGE est en cours. A cet instant, les DEMANDEURS PAR qui ne commandent pas BREQ* au niveau bas, ainsi que tous les ESCLAVES libèrent WAIT* et AC au niveau haut. Les DEMANDEURS PAR concurrents qui commandent BREQ* au niveau bas, placent leur ID ARBITRAGE sur AD24-AD30 et relâchent ensuite leur contribution à la ligne WAIT* en la libérant au niveau haut.

A l'étape 4, les DEMANDEURS PAR concurrents négocient pour déterminer quel MAITRE aura la prochaine allocation de bus. Après avoir détecté un niveau haut sur WAIT*, indiquant que tous les DEMANDEURS concurrents ont placé leur ID ARBITRAGE sur AD24-AD30, ils attendent le temps minimal prescrit pour permettre la stabilisation de leur logique de sélection (voir annexe A) et relâchent alors leur contribution à la ligne AC en la libérant au niveau haut. Lorsque cela a été fait par tous les DEMANDEURS concurrents, l'ID ARBITRAGE sur AD24-AD30 (désigné BUS-ARB-ID) est égal à l'ID ARBITRAGE (ARB-ID) du DEMANDEUR dont la priorité d'arbitrage est la plus haute.

A l'étape 5, le nouveau DEMANDEUR actif est sélectionné. Après avoir reçu AC au niveau haut, indiquant la stabilisation de la logique de sélection de tous les DEMANDEURS concurrents, ils comparent le BUS-ARB-ID à leur propre identificateur. Le DEMANDEUR PAR dont l'identificateur correspond au niveau de priorité présent sur AD24-AD30 devient le nouveau DEMANDEUR actif. Pour ce faire, il relâche au niveau haut sa contribution à la ligne BREQ* et commande ensuite BUSY* au niveau bas. Tous les DEMANDEURS concurrents doivent libérer AD24-AD30 dans un temps maximal prescrit après que AC a été remonté au niveau haut.

A l'étape 6, le DEMANDEUR PAR qui a déclenché le cycle d'ARBITRAGE détecte AC au niveau haut et termine le cycle. Il le fait en libérant PAS* au niveau haut et permet à WR* et SPACE0-SPACE1 de passer au niveau haut.

A l'étape 7, tous les DEMANDEURS PAR ainsi que tous les ESCLAVES acquittent la fin du cycle d'ARBITRAGE. Quand ils détectent PAS* au niveau haut, ils positionnent AC au niveau bas. Les ESCLAVES du VSB capables de commander WAIT* au niveau bas et tous les DEMANDEURS PAR du VSB commandent WAIT* au niveau bas. Cela termine le cycle d'ARBITRAGE.

In step 2, the PAR REQUESTER associated with the MASTER that is in control of the bus (referred to as the active REQUESTER), receives BREQ* driven low. After its associated MASTER no longer needs the bus and has made MASTER WANTS BUS false, the active PAR REQUESTER initiates an ARBITRATION cycle. It first verifies that its associated MASTER has stopped driving bus lines by detecting a high level on PAS*. The active PAR REQUESTER then drives SPACE0-SPACE1 and WR* to low to indicate that an ARBITRATION cycle is in progress. After detecting a high level on ASACK0*-ASACK1*, signifying that the SLAVE that responded to the previous cycle is off the bus, the active PAR REQUESTER drives PAS* to low. The active PAR REQUESTER then releases BUSY* to high.

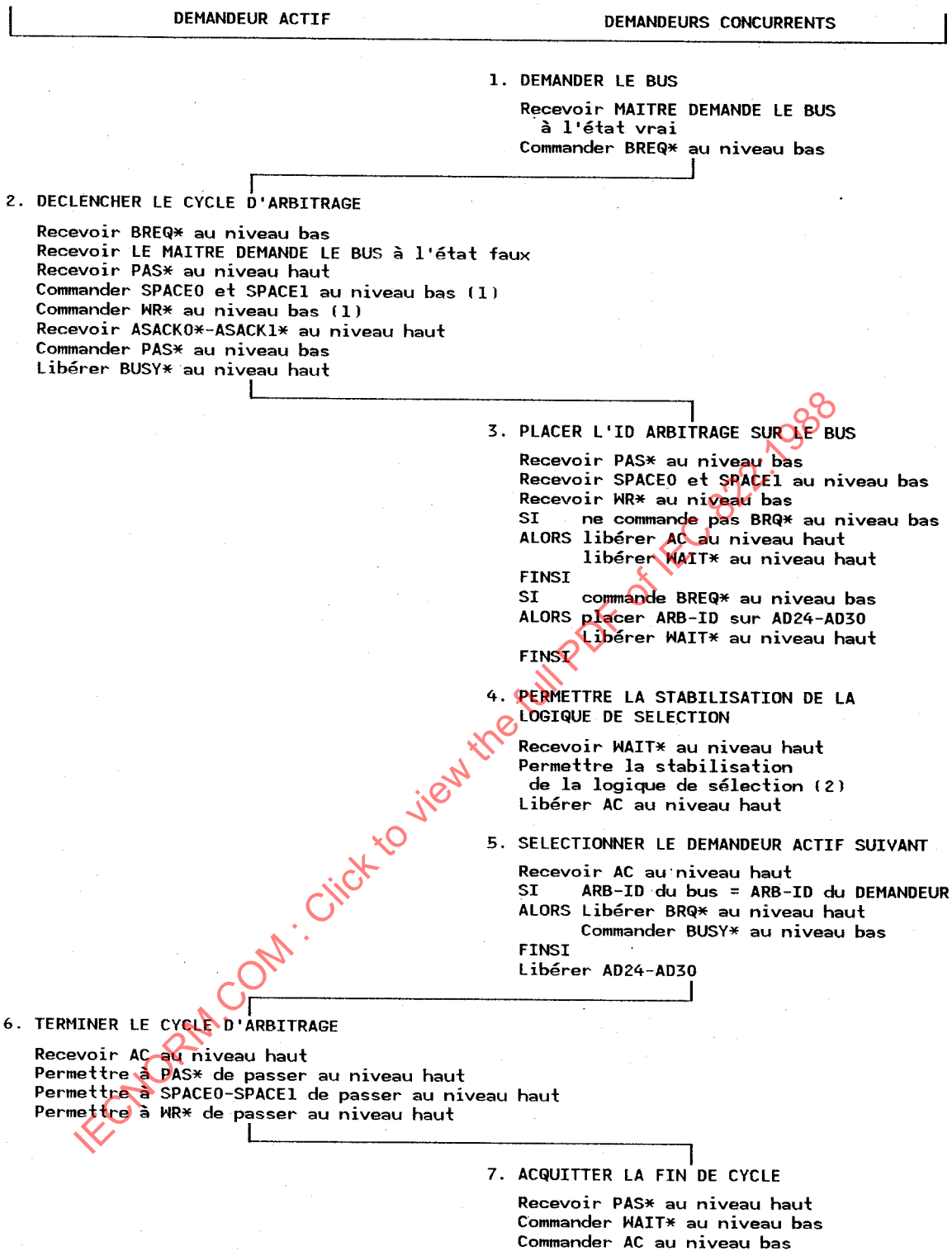
In step 3, after detecting a falling edge on PAS*, all PAR REQUESTERS as well as SLAVES receive SPACE0-SPACE1 and WR* low and determine that an ARBITRATION cycle is in progress. At this time PAR REQUESTERS that are not driving BREQ* low, as well as all SLAVES release both WAIT* and AC to high. Contending PAR REQUESTERS, i.e. those PAR REQUESTERS that are driving BREQ* low, drive their ARBITRATION ID on AD24-AD30, and then release to high their contribution to WAIT*.

In step 4, the contending PAR REQUESTERS negotiate to determine which MASTER will be granted the bus next. After detecting a high level on WAIT*, indicating that all contending REQUESTERS have driven their ARBITRATION ID on AD24-AD30, they wait a minimum prescribed time to allow their selection logic to settle (see Appendix A), and then release their contribution to the AC line to high. By the time all contending REQUESTERS have done so, the ARBITRATION ID carried on AD24-AD30 (referred to as BUS-ARB-ID) is equal to the ARBITRATION ID (ARB-ID) of the REQUESTER whose arbitration priority is the highest.

In step 5 the new active REQUESTER is selected. After receiving AC high, indicating that the selection circuitry of all contending REQUESTERS has settled, they compare the BUS-ARB-ID to their own. The PAR REQUESTER whose ID matches the priority level on AD24-AD30 becomes the next active REQUESTER. To do so it releases its contribution to the BREQ* line to high, and then drives BUSY* low. All contending REQUESTERS are required to release AD24-AD30 within a maximum prescribed time after AC goes high.

In step 6, the PAR REQUESTER that initiated the ARBITRATION cycle detects AC high and terminates it. It does so by releasing PAS* to high, and then allowing WR* and SPACE0-SPACE1 to go high.

In step 7, all PAR REQUESTERS as well as all SLAVES acknowledge the termination of the ARBITRATION cycle. When they detect the high level on PAS* they drive AC to low. Those VSB SLAVES that are capable of driving WAIT* low and all VSB PAR REQUESTERS drive WAIT* low. This terminates the ARBITRATION cycle.

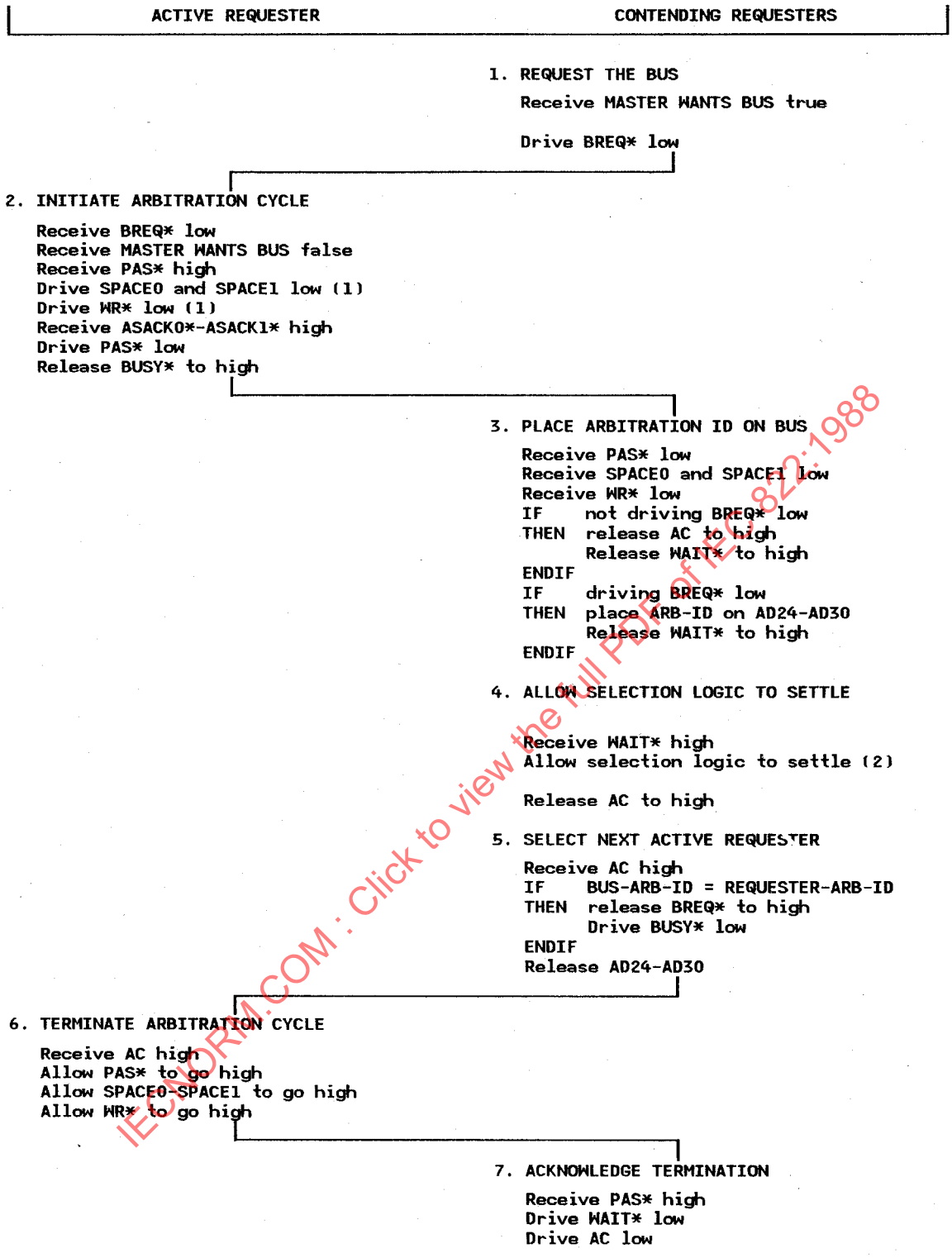


Notes:

1.- A l'étape 2, le DEMANDEUR PAR actif commande SPACE0-SPACE1 et WR* au niveau bas pour informer les ESCLAVES et les DEMANDEURS PAR qu'un cycle d'ARBITRAGE est en cours, comme défini au paragraphe 3.4.2.2, tableau 3-3.

2.- A l'étape 4, les DEMANDEURS PAR concurrents laissent le temps à la logique de sélection d'accomplir le processus de sélection. Les fonctions de cette logique de sélection et les délais de temps sont décrits dans l'annexe A.

Fig. 3-7. - Organigramme d'un cycle d'ARBITRAGE.



Notes:

1.- In step 2, the active PAR REQUESTER drives SPACE0-SPACE1 and WR* low to inform SLAVES and PAR REQUESTERS that an ARBITRATION cycle is in progress, as defined in Paragraph 3.4.2.2, Table 3-3.

2.- In step 4, contending PAR REQUESTERS allow time for the selection process to be completed by their selection logic. The attributes of this selection logic, and the timing of the delay are described in Appendix A.

Fig. 3-7. - Flow of an ARBITRATION cycle.

3.4.2.2 *Evolution des signaux pendant le cycle d'ARBITRAGE*

OBSERVATION 3.12:
Pour que le processus d'arbitrage parallèle fonctionne correctement, il faut qu'il y ait toujours un DEMANDEUR PAR actif. Le paragraphe 3.4.3 décrit comment le DEMANDEUR PAR actif initial est sélectionné pendant la séquence de mise sous tension.

REGLE 3.18:
Les DEMANDEURS PAR NE DOIVENT PAS commander BREQ* à moins qu'ils ne détectent leur signal sur la carte LE MAITRE DEMANDE LE BUS à l'état vrai.

REGLE 3.19:
Le DEMANDEUR PAR actif NE DOIT PAS commencer un cycle d'ARBITRAGE à moins que son signal sur la carte LE MAITRE DEMANDE LE BUS ne soit à l'état faux.

REGLE 3.20:
Durant le cycle d'ARBITRAGE, le DEMANDEUR PAR actif DOIT commander SPACE0-SPACE1 et WR* pour sélectionner un cycle d'ARBITRAGE, comme indiqué dans le tableau 3-3.

REGLE 3.21:
Les ESCLAVES et les DEMANDEURS PAR du bus VSB DOIVENT décoder SPACE0-SPACE1 et WR* pour déterminer qu'un cycle d'ARBITRAGE est en cours, comme indiqué dans le tableau 3-3.

Tableau 3-3

Utilisation de SPACE0-SPACE1 et WR* pour sélectionner un cycle d'ARBITRAGE

Type de cycle	SPACE1	SPACE0	WR*
Cycle d'ARBITRAGE	bas	bas	bas

OBSERVATION 3.13
Des REGLES additionnelles de commande de SPACE0-SPACE1 et WR* sont données au chapitre 2, paragraphe 2.5.1.2, tableau 2-9, et au paragraphe 2.5.4.2, tableau 2-18.

REGLE 3.22:
Les ESCLAVES NE DOIVENT PAS répondre au cycle d'ARBITRAGE.

REGLE 3.23:
SI un DEMANDEUR PAR ne commande pas BREQ* au niveau bas au moment où il détecte un cycle d'ARBITRAGE,
ALORS il doit libérer WAIT* et AC au niveau haut.

REGLE 3.24:
SI un DEMANDEUR PAR commande la ligne BREQ* au niveau bas au moment où il détecte un cycle d'ARBITRAGE,
ALORS il DOIT placer un ID ARBITRAGE sur AD24-AD30 et participer au cycle d'ARBITRAGE.

3.4.2.2 *Signaling during the ARBITRATION cycle*

OBSERVATION 3.12:
For the Parallel Arbitration scheme to function properly, it is required that there always be an active PAR REQUESTER. Paragraph 3.4.3 describes how the initial active PAR REQUESTER is selected during the power-up sequence.

RULE 3.18:
PAR REQUESTERS MUST NOT drive BREQ* unless they detect their on-board signal MASTER WANTS BUS true.

RULE 3.19:
The active PAR REQUESTER MUST NOT start an ARBITRATION cycle unless its on-board signal MASTER WANTS BUS is false.

RULE 3.20:
During ARBITRATION cycles, the active PAR REQUESTER MUST drive SPACE0-SPACE1 and WR* to select an ARBITRATION cycle, as shown in Table 3-3.

RULE 3.21:
VSB SLAVES and PAR REQUESTERS MUST decode SPACE0-SPACE1 and WR* to determine that an ARBITRATION cycle is in progress, as shown in Table 3-3.

Table 3-3

Use of SPACE0-SPACE1 and WR* to select an ARBITRATION cycle

Type of cycle	SPACE1	SPACE0	WR*
ARBITRATION cycle	low	low	low

OBSERVATION 3.13
Additional RULES for driving SPACE0-SPACE1 and WR* are given in Chapter 2, Paragraph 2.5.1.2, Table 2-9, and Paragraph 2.5.4.2, Table 2-18.

RULE 3.22:
SLAVES MUST NOT respond to an ARBITRATION cycle.

RULE 3.23:
IF a PAR REQUESTER is not driving BREQ* low at the time it detects an ARBITRATION cycle,
THEN it MUST release WAIT* and AC to high.

RULE 3.24:
IF a PAR REQUESTER is driving the BREQ* line low at the time it detects an ARBITRATION cycle,
THEN it MUST drive an ARBITRATION ID on AD24-AD30, and participate in the ARBITRATION cycle.

REGLE 3.25:
L'ID ARBITRAGE qu'un DEMANDEUR concurrent place sur AD24-AD30 DOIT inclure les bits d'adressage géographique comme suit: GA0 sur AD24, GA1 sur AD25 et GA2 sur AD26.

OBSERVATION 3.14:
Cette REGLE assure que chaque carte a un code unique d'ID ARBITRAGE qui conduira à sélectionner correctement et sans ambiguïté un seul des DEMANDEURS PAR concurrents comme DEMANDEUR actif suivant. Il en résulte qu'un seul MAITRE utilise le bus à un moment donné.

AUTORISATION 3.2:
L'ID ARBITRAGE commandé par un DEMANDEUR concurrent PEUT inclure des bits définis par l'utilisateur et placés sur les lignes AD27-AD30.

REGLE 3.26:
Un DEMANDEUR concurrent NE DOIT PAS libérer sa contribution à AC avant de recevoir WAIT* au niveau haut et d'avoir attendu un temps suffisant pour la stabilisation de sa logique de sélection.

REGLE 3.27:
Les DEMANDEURS PAR concurrents NE DOIVENT PAS échantillonner ou utiliser le résultat de la logique de sélection avant d'avoir détecté un niveau haut sur AC.

OBSERVATION 3.15:
Des REGLES additionnelles pour les transmissions sur PAS*, SPACE0-SPACE1, WR*, WAIT*, ASACK0*-ASACK1* et AC sont données au chapitre 2, paragraphes 2.4.1, 2.4.2 et 2.4.3.

3.4.3 *Séquence de mise sous tension*

Un même système VSB peut être configuré à la fois avec des DEMANDEURS SER et PAR. Le mode d'opération du système avec arbitrage de type série ou parallèle est déterminé pendant la séquence de mise sous tension.

De plus, lorsque le mode d'arbitrage parallèle est sélectionné, le DEMANDEUR PAR actif initial est sélectionné au cours de la séquence de mise sous tension. Cela est nécessaire puisque le mécanisme d'arbitrage parallèle impose qu'il y ait toujours un DEMANDEUR PAR actif. D'autre part, le contrôle du bus ne peut être transféré qu'au cours d'un cycle d'ARBITRAGE qui ne peut être déclenché que par un DEMANDEUR actif.

Le paragraphe 3.4.3.1 et la figure 3-8, page 224, décrivent l'interaction entre l'ARBITRE, les DEMANDEURS SER et les DEMANDEURS PAR durant la séquence de mise sous tension. Ce paragraphe fournit une description informelle du protocole pour familiariser le lecteur avec la séquence de mise sous tension. Le paragraphe 3.4.3.2 constitue la spécification formelle du protocole de mise sous tension, alors que la section 3.6 spécifie les contraintes temporelles supplémentaires.

3.4.3.1 *Déroulement d'une séquence de mise sous tension*

L'interaction entre l'ARBITRE, les DEMANDEURS SER et les DEMANDEURS PAR pendant la séquence de mise sous tension est montrée dans la figure 3-8. A l'étape 1, les modules du bus d'arbitrage attendent que la tension ait atteint le niveau spécifié, comme défini au chapitre 5, section 5.3.

RULE 3.25:
The ARBITRATION ID that a contending REQUESTER drives on AD24-AD30 MUST include the geographical addressing bits as follows: GA0 on AD24, GA1 on AD25 and GA2 on AD26.

OBSERVATION 3.14:
This RULE ensures that each board has a unique ARBITRATION ID code that will result in a positive and unambiguous selection of only one of the contending PAR REQUESTERS as the next active REQUESTER. This will result in only one MASTER using the bus at a time.

PERMISSION 3.2:
The ARBITRATION ID that a contending REQUESTER drives MAY include user defined and supplied bits on AD27-AD30.

RULE 3.26:
A contending REQUESTER MUST NOT release its contribution to AC before it receives WAIT* high and it has waited a sufficient time thereafter for its selection logic to settle.

RULE 3.27:
Contending PAR REQUESTERS MUST NOT sample or use the result of their selection logic before they detect a high level on AC.

OBSERVATION 3.15:
Additional RULES for signaling on PAS*, SPACE0-SPACE1, WR*, WAIT*, ASACK0*-ASACK1* and AC are given in Chapter 2, Paragraphs 2.4.1, 2.4.2 and 2.4.3.

3.4.3 *Power-up sequence*

A VSB system might be configured with a mixture of SER REQUESTERS and PAR REQUESTERS. The decision whether the system will operate in the Serial or in the Parallel Arbitration mode is determined during the power-up sequence.

In addition, when the Parallel Arbitration mode is selected, the initial active PAR REQUESTER is selected in the course of the power-up sequence. This is necessary since the Parallel Arbitration mechanism requires that there always be an active PAR REQUESTER. This is because bus mastership can only be transferred in the course of an ARBITRATION cycle, which can only be initiated by an active REQUESTER.

Paragraph 3.4.3.1 and Figure 3-8, page 225, describe the interaction between the ARBITER, SER REQUESTERS and PAR REQUESTERS during the power-up sequence. This paragraph provides an informal description of the protocol to familiarize the reader with the power-up sequence. Paragraph 3.4.3.2 constitutes a formal specification of the power-up protocol, while Section 3.6 specifies additional timing requirements.

3.4.3.1 *Flow of the power-up sequence*

The interaction between the ARBITER, SER REQUESTERS and PAR REQUESTERS during the power-up sequence is shown in Figure 3-8. In step 1, the Arbitration Bus modules wait for power to reach its specified range, as defined in Chapter 5, Section 5.3.

A l'étape 2a, les DEMANDEURS SER informent le DEMANDEUR PAR éventuellement installé dans l'emplacement suivant qu'un DEMANDEUR SER au moins est installé dans le système. Pour ce faire, ils assurent que leur ligne BGOUT* est au niveau haut dans les 5 ms après que la tension a été dans l'intervalle spécifié. De plus, les DEMANDEUR SER s'abstiennent de commander BREQ* au niveau bas pendant un minimum de 200 ms après que la tension a atteint son niveau spécifié.

A l'étape 2b, l'ARBITRE commande BGIN* de l'emplacement 1 au niveau haut dans les 5 ms après que la tension a atteint son niveau spécifié. De plus, l'ARBITRE s'abstient de surveiller son entrée BREQ* pendant un minimum de 200 ms après que la tension a atteint son niveau spécifié.

A l'étape 2c, le DEMANDEUR PAR informe le DEMANDEUR PAR éventuellement installé dans l'emplacement suivant, s'il a ou n'a pas détecté l'existence d'un module série (un DEMANDEUR SER ou un ARBITRE). Il déduit qu'il existe un module série dans l'emplacement précédent s'il détecte un niveau haut sur son entrée BGIN*. Il s'assure alors que BGOUT*, BREQ* et BUSY* sont commandés au niveau haut dans les 5 ms qui suivent l'établissement de la tension au niveau spécifié.

D'autre part, si le DEMANDEUR PAR détecte un niveau bas sur son entrée BGIN*, signifiant qu'un DEMANDEUR PAR est installé dans l'emplacement précédent, alors il met sa sortie BGOUT* à l'état haute impédance.

Cette manière de faire assure que l'entrée BGIN* du DEMANDEUR PAR dans l'emplacement suivant sera maintenue à son niveau bas initial. Ce niveau bas est établi grâce à la résistance reliant l'entrée BGIN* des DEMANDEURS PAR à la masse, comme spécifié au chapitre 4, paragraphe 4.3.4. De plus, le DEMANDEUR PAR commande BREQ* et BUSY* au niveau bas pour préparer une éventuelle séquence de sélection de DEMANDEUR potentiel en phase de mise sous tension.

A l'étape 3, en fonction du niveau sur leur entrée BGIN*, les DEMANDEURS PAR se préparent à fonctionner soit en DEMANDEUR SER, soit en DEMANDEUR PAR. Cependant, le niveau que le DEMANDEUR détecte sur son entrée BGIN* peut changer tant que tous les maillons de la chaîne série n'ont pas été stabilisés. Par conséquent, les DEMANDEURS PAR contrôlent leur entrée BGIN* 200 ms après que la tension a atteint le niveau spécifié. Si à ce moment leur entrée BGIN* est au niveau haut, ils déduisent que le système opérera dans le mode série et terminent la phase de mise sous tension.

En revanche, si après 200 ms, leur entrée BGIN* est encore au niveau bas, ils déduisent que le système opérera dans le mode parallèle et exécutent les étapes 4-6 pour sélectionner le DEMANDEUR PAR actif initial.

A l'étape 4, tous les DEMANDEURS PAR placent leur ID ARBITRAGE sur les lignes AD24-AD30 et libèrent ensuite, au niveau haut, leur contribution à la commande de la ligne BREQ*.

A l'étape 5, tous les DEMANDEURS PAR négocient pour déterminer lequel sera DEMANDEUR actif initial. Après avoir détecté un niveau haut sur BREQ* indiquant que tous les DEMANDEURS PAR ont placé leur ID ARBITRAGE sur les lignes AD24-AD30, ils attendent le minimum de temps prescrit pour permettre la stabilisation de leur logique de sélection (voir annexe A) et libèrent ensuite, au niveau haut, leur contribution à la commande de la ligne BUSY*. Cela étant fait par tous

In step 2a, SER REQUESTERS inform a PAR REQUESTER that might be installed in the next slot that at least one SER REQUESTER is installed in the system. They do so by ensuring that their BGOUT* line is high within 5 ms after power is within its specified range. In addition, SER REQUESTERS refrain from driving BREQ* low for a minimum of 200 ms after power is within its specified range.

In step 2b, the ARBITER drives slot 1 BGIN* high within 5 ms after power is within its specified range. In addition, the ARBITER refrains from monitoring its BREQ* input for a minimum of 200 ms after power is within its specified range.

In step 2c, the PAR REQUESTER informs the PAR REQUESTER that might be installed in the next slot whether it detected the existence of a serial module (a SER REQUESTER or an ARBITER). It determines that a serial module is installed in the previous slot by detecting a high level on its BGIN* input. It then ensures that BGOUT*, BREQ* and BUSY* are driven high within 5 ms of power reaching its specified range.

On the other hand, if the PAR REQUESTER detects a low level on its BGIN* input, signifying that a PAR REQUESTER is installed in the previous slot, then it tri-states its BGOUT* output.

Doing so ensures that the BGIN* input of the PAR REQUESTER in the next slot will be maintained in its initial low state. This low level is established by the resistor that connects the BGIN* input of PAR REQUESTERS to ground, as specified in Chapter 4, Paragraph 4.3.4. In addition, the PAR REQUESTER drives BREQ* and BUSY* low in preparation for a potential power-up REQUESTER selection sequence.

In step 3, depending on the level of their BGIN* input, PAR REQUESTERS prepare to operate either as a SER REQUESTER or as a PAR REQUESTER. However, the level that the REQUESTER detects on its BGIN* input might change until all segments of the daisy-chain have settled. And therefore, PAR REQUESTERS monitor their BGIN* input for 200 ms after power is within its specified range. If at that time their BGIN* input is high, they determine that the system will operate in the Serial mode, and conclude the power-up sequence.

On the other hand, if after 200 ms their BGIN* input is still low, they determine that the system will operate in the Parallel mode, and execute step 4-6 to select the initial active PAR REQUESTER.

In step 4, all PAR REQUESTERS drive their ARBITRATION ID on AD24-AD30, and then release their contribution to BREQ* to high.

In step 5, all PAR REQUESTERS negotiate to determine which will be the initial active REQUESTER. After detecting a high level on BREQ*, indicating that all PAR REQUESTERS have driven their ARBITRATION ID on AD24-AD30, they wait a minimum prescribed time to allow their selection logic to settle (see Appendix A), and then release their contribution to the BUSY* line to high. By the time all contending

les DEMANDEURS concurrents, l'ID ARBITRAGE résultant sur AD24-AD30 (désigné par "BUS-ARB-ID") sera égal à l'ID ARBITRAGE (ARB-ID) du DEMANDEUR dont la priorité d'arbitrage est la plus élevée.

A l'étape 6, le DEMANDEUR actif initial est sélectionné. Après réception de BUSY* au niveau haut, indiquant la stabilisation de la logique de sélection de tous les DEMANDEURS PAR, ils comparent le BUS-ARB-ID à leur propre identificateur. Le DEMANDEUR PAR dont l'identificateur correspond au niveau de priorité sur les lignes AD24-AD30 devient le DEMANDEUR actif initial. Pour ce faire, il commande BUSY* au niveau bas. Tous les DEMANDEURS PAR doivent libérer AD24-AD30 dans un temps maximal prescrit, après avoir détecté BUSY* au niveau haut. Cela termine la séquence de mise sous tension.

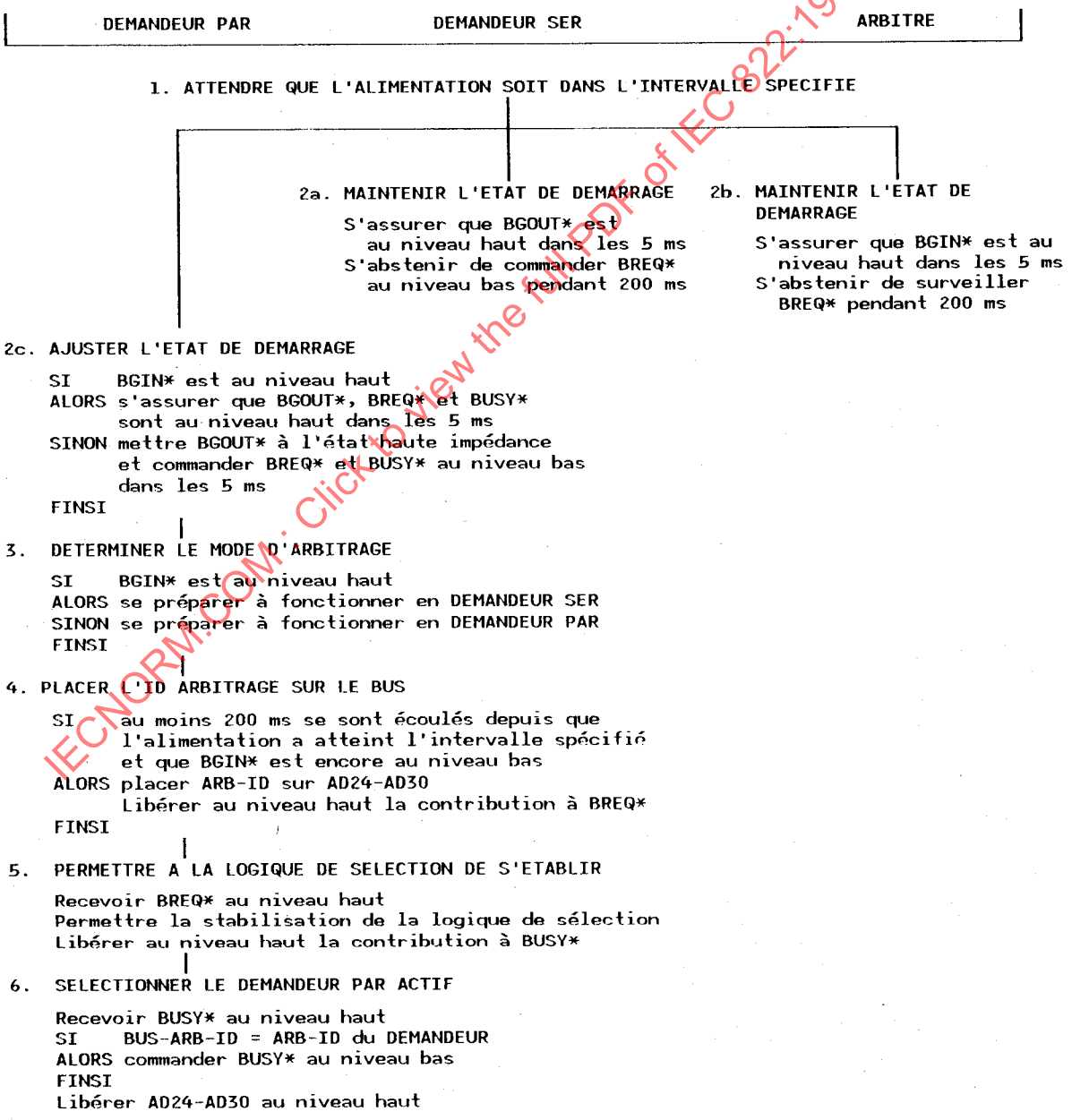


Fig. 3-8. - Organigramme de la séquence de démarrage.

REQUESTERS have done so, the ARBITRATION ID carried on AD24-AD30 (referred to as BUS-ARB-ID) will be equal to the ARBITRATION ID (ARB-ID) of the REQUESTER whose arbitration priority is the highest.

In step 6 the initial active REQUESTER is selected. After receiving BUSY* high, indicating that the selection circuitry of all PAR REQUESTERS has settled, they compare the BUS-ARB-ID to their own. The PAR REQUESTER whose ID matches the priority level on AD24-AD30 becomes the initial active REQUESTER. To do so it drives BUSY* low. All PAR REQUESTERS are required to release AD24-AD30 within a maximum prescribed time after detecting BUSY* high. This terminates the power-up sequence.

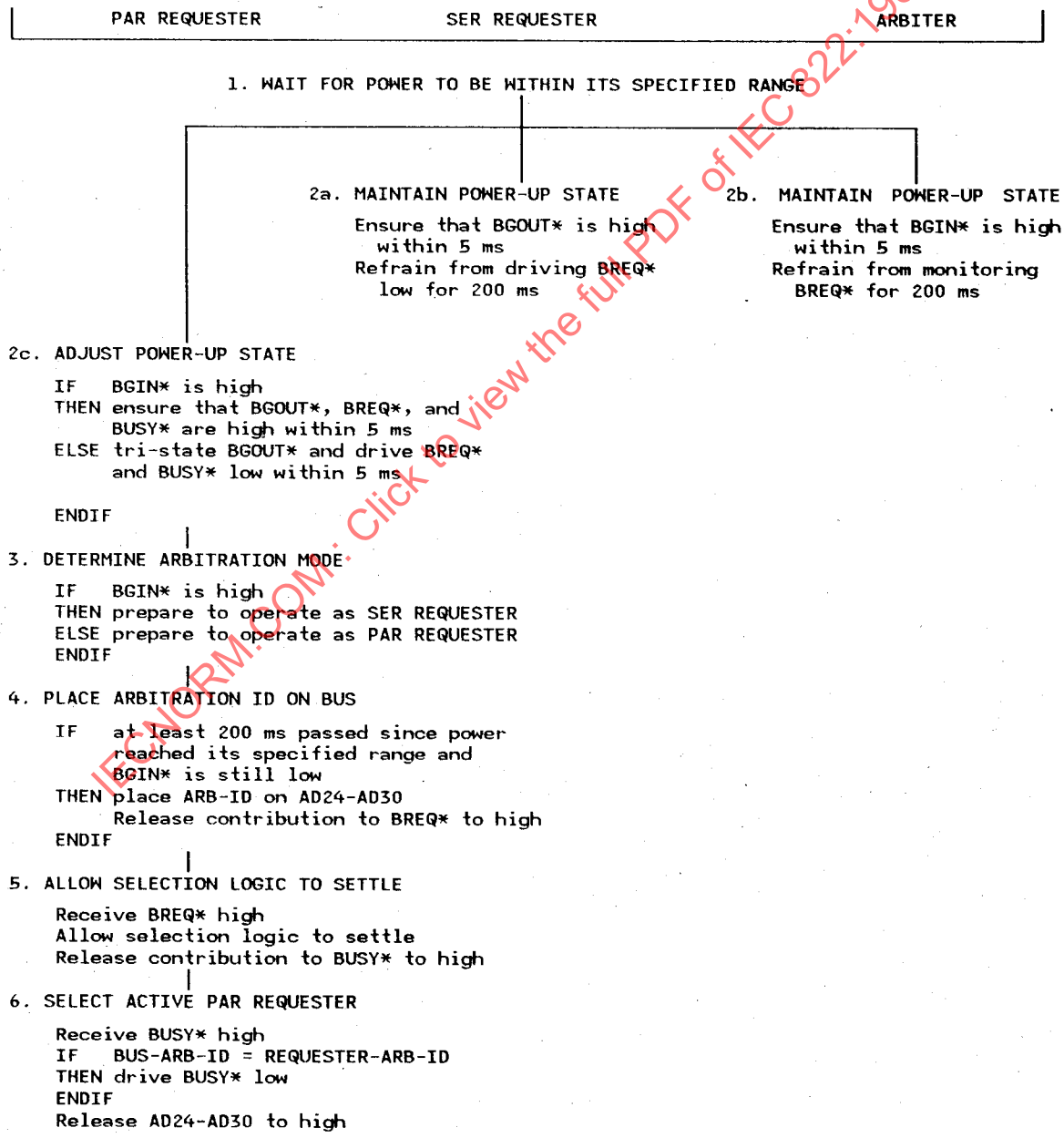


Fig. 3-8. - Flow of the power-up sequence.

3.4.3.2 *Interaction entre les modules du bus d'arbitrage pendant le démarrage*

OBSERVATION 3.16:
Le chapitre 4 décrit la méthode utilisée par l'ARBITRE et par les DEMANDEURS SER pour établir le niveau haut sur la ligne BGIN*.

REGLE 3.28:
L'ARBITRE DOIT garantir que sa sortie BGIN* est au niveau haut dans les 5 ms après que la tension d'alimentation a atteint son niveau spécifié, comme défini au chapitre 5, section 5.3.

REGLE 3.29:
L'ARBITRE NE DOIT PAS commander sa sortie BGIN* au niveau bas pendant au moins 200 ms après que la tension d'alimentation a atteint son niveau spécifié.

REGLE 3.30:
Les DEMANDEURS SER DOIVENT garantir que leur sortie BGOUT* est au niveau haut dans les 5 ms après que la tension d'alimentation a atteint son niveau spécifié, comme défini au chapitre 5, section 5.3.

REGLE 3.31:
Les DEMANDEURS SER NE DOIVENT PAS commander BGOUT* au niveau bas pendant au moins 200 ms après que la tension d'alimentation a atteint son niveau spécifié.

REGLE 3.32:
Les DEMANDEURS SER NE DOIVENT PAS commander BREQ* au niveau bas pendant au moins 200 ms après que la tension d'alimentation a atteint son niveau spécifié.

REGLE 3.33:
L'ARBITRE NE DOIT PAS gérer la ligne BREQ* pendant les 200 ms après que la tension d'alimentation a atteint son niveau spécifié.

OBSERVATION 3.17:
Le chapitre 4 spécifie la méthode utilisée par les DEMANDEURS PAR pour établir le niveau bas initial sur leur entrée BGIN*.

REGLE 3.34:
SI un DEMANDEUR PAR détecte un niveau haut sur son entrée BGIN*,
ALORS il DOIT s'assurer que les signaux BGOUT*, BREQ* et BUSY* sont au niveau haut dans les 5 ms après avoir détecté ce niveau haut.

REGLE 3.35:
SI un DEMANDEUR PAR détecte un niveau bas sur son entrée BGIN*,
ALORS il DOIT mettre son émetteur de signal BGOUT* dans l'état haute impédance et commander BREQ* et BUSY* au niveau bas dans les 5 ms après avoir détecté ce niveau bas.

REGLE 3.36:
Après avoir libéré BREQ* et BUSY* au niveau haut au cours d'une séquence de sélection dans la phase de démarrage, les DEMANDEURS PAR NE DOIVENT PAS commander BREQ* au niveau bas tant qu'ils n'ont pas détecté BUSY* au niveau bas à la fin de la séquence de sélection de la mise sous tension.

3.4.3.2 *Interaction between arbitration bus modules during power-up*

OBSERVATION 3.16:
Chapter 4 specifies the method used by the ARBITER and by SER REQUESTERS to establish the high level on the BGIN* line.

RULE 3.28:
The ARBITER MUST ensure that its BGIN* output is high within 5 ms after system power is within its specified range, as defined in Chapter 5, Section 5.3.

RULE 3.29:
The ARBITER MUST NOT drive its BGIN* output low for at least 200 ms after power is within its specified range.

RULE 3.30:
SER REQUESTERS MUST ensure that their BGOUT* output is high within 5 ms after system power is within its specified range, as defined in Chapter 5, Section 5.3.

RULE 3.31:
SER REQUESTERS MUST NOT drive BGOUT* low for at least 200 ms after power is within its specified range.

RULE 3.32:
SER REQUESTERS MUST NOT drive BREQ* low for at least 200 ms after power is within its specified range.

RULE 3.33:
The ARBITER MUST NOT monitor the BREQ* line for 200 ms after power is within its specified range.

OBSERVATION 3.17:
Chapter 4 specifies the method used by PAR REQUESTERS to establish the initial low level on their BGIN* input.

RULE 3.34:
IF a PAR REQUESTER detects a high level on its BGIN* input,

THEN it MUST ensure that BGOUT*, BREQ* and BUSY* are high within 5 ms after detecting this high level.

RULE 3.35:
IF a PAR REQUESTER detects a low level on its BGIN* input,

THEN it MUST tri-state its BGOUT* driver, and drive BREQ* and BUSY* to low within 5 ms after detecting this low level.

RULE 3.36:
After releasing BREQ* and BUSY* to high in the course of a power-up selection sequence, PAR REQUESTERS MUST NOT drive BREQ* low until after they detect BUSY* driven low at the completion of the power-up selection sequence.

3.5 *Interaction entre le MAITRE, son DEMANDEUR associé et/ou son ARBITRE associé*

3.5.1 Acquisition du DTB

Etant donné les caractéristiques de la chaîne série, les DEMANDEURS SER qui sont installés dans les emplacements de rang peu élevé du fond de panier reçoivent l'acquittement du bus avant ceux qui sont installés dans les emplacements de rang plus élevé. Si une carte, installée dans un emplacement de rang peu élevé, génère un taux élevé de trafic sur le bus, elle peut, dans certains cas, empêcher les cartes installées dans des emplacements de rang plus élevé d'obtenir l'autorisation d'utilisation du bus.

Par exemple, considérons un système qui inclut trois cartes processeurs: CPU A dans l'emplacement 1, CPU B en 2, et CPU C en 3. Dans certains cas, les trois demandent le bus au même instant. Quand l'ARBITRE autorise l'utilisation du bus, la carte CPU A détecte un niveau bas sur son signal BGIN* et commence à utiliser le bus. Après que la carte CPU A a libéré le bus et que l'ARBITRE a émis une nouvelle allocation, la carte CPU A la transmet sur la chaîne série. La carte CPU B détecte cette autorisation et commence à utiliser le bus. Alors que la carte CPU B utilise le bus, la carte CPU A émet une nouvelle demande. Après que la carte CPU B a libéré le bus et que l'ARBITRE a alloué le bus à nouveau, la carte CPU A, qui voit la première l'allocation, prend le contrôle du bus. Dans certains cas extrêmes, la carte CPU C n'aura jamais l'allocation du bus.

RECOMMANDATION 3.2:
Pour promouvoir l'équité dans les accès au bus, réaliser les DEMANDEURS SER de telle sorte que lorsqu'ils ont libéré le bus (c'est-à-dire relâché BUSY* au niveau haut), ils s'abstiennent de demander le bus (c'est-à-dire de piloter BREQ* au niveau bas), tant qu'ils n'auront pas détecté un niveau haut sur BREQ*. Cela garantit que les cartes qui sont situées plus en aval sur la chaîne série auront un accès équitable au bus.

3.5.2 *Libération du DTB*

Le protocole d'arbitrage du bus détermine quand et comment le DTB est alloué aux différents MAITRES du système. Cependant, il ne contrôle pas l'instant où les MAITRES libèrent le bus (DTB).

Les MAITRES utilisent plusieurs critères de décision pour libérer le DTB. Les MAITRES IHV libèrent le bus après avoir accompli un cycle de RECONNAISSANCE D'INTERRUPTION, alors que des MAITRES libèrent le bus lorsqu'ils ont fini leur transfert de données.

OBSERVATION 3.18:
Quels que soient les critères utilisés pour décider de la libération du bus, un arbitrage a lieu avant qu'un autre MAITRE commence à l'utiliser. Cet arbitrage a lieu ou bien PENDANT le dernier transfert de données ou APRES ce transfert, en fonction de la consigne que le MAITRE actif donne à son DEMANDEUR SER sur la carte.

PERMISSION 3.3:
Les MAITRES PEUVENT libérer le DTB soit durant soit après leur dernier transfert de données.

3.5 *Interaction between the MASTER, its associated REQUESTER and/or its associated ARBITER*

3.5.1 *Acquisition of the DTB*

Due to the characteristics of the daisy-chain, SER REQUESTERS that are installed in the lower numbered slots of the backplane are granted the bus before those that are installed in the higher numbered slots. If a board that is installed in low numbered slot generates a substantial amount of bus traffic, it might, in some cases, prevent boards that are installed in higher numbered slots from ever being granted the bus.

For example, consider a system that includes three processor boards: CPU A in slot 1, CPU B in slot 2, and CPU C in slot 3. In some cases, all three request the bus at the same time. When the ARBITER grants use of the bus, CPU A detects a low level on its BGIN* signal and starts using the bus. After CPU A releases the bus and the ARBITER issues a new bus grant, CPU A passes it down the daisy chain. CPU B detects this grant and starts using the bus. While CPU B uses the bus, CPU A drives its request low again. After CPU B releases the bus and the ARBITER grants the bus again, CPU A, being first to see the grant, assumes control of the bus. In some extreme cases, CPU C will never be granted use of the bus.

RECOMMENDATION 3.2:
To promote fairness in accessing the bus, design SER REQUESTERS that have just released the bus, i.e. released BUSY* to high, to refrain from requesting the bus, i.e. drive BREQ* low, until after they detect a high level on BREQ*. This will assure that boards that reside further down the daisy-chain have equal access to the bus.

3.5.2 *Release of the DTB*

The bus arbitration protocol determines how and when the DTB is granted to the various MASTERS in the system. It does not, however, control when MASTERS release the DTB.

MASTERS use several criteria in deciding when to release the DTB. IHV MASTERS give up the bus after they finish an INTERRUPT-ACKNOWLEDGE cycle, while MASTERS give up the bus when they finish their data transfer.

OBSERVATION 3.18:
Whatever criteria are used to decide when to release the DTB, arbitration is done before another MASTER begins using it. This arbitration takes place either DURING the last data transfer or AFTER that transfer, depending on how much notice the active MASTER gives to its on-board SER REQUESTER.

PERMISSION 3.3:
MASTERS MAY release the DTB either during or after their last data transfer.

Par exemple, si durant le dernier transfert de données le MAITRE notifie à son DEMANDEUR SER sur la carte qu'il n'utilisera plus le bus, le DEMANDEUR SER libère BUSY* et un ARBITRAGE a lieu pendant le dernier transfert. Mais si le MAITRE attend que le dernier transfert soit terminé avant de le signaler à son DEMANDEUR SER sur la carte, le DTB reste au repos lorsque l'arbitrage est effectué.

3.5.3 *Course critique entre les demandes du MAITRE et les allocations de l'ARBITRE*

Supposons deux DEMANDEURS SER: un DEMANDEUR SER A et un DEMANDEUR SER B. Le DEMANDEUR SER B qui est le plus en aval sur la chaîne série demande le bus et l'ARBITRE commande la ligne correspondante d'allocation de bus au niveau bas. L'accord du bus parvient au DEMANDEUR SER A juste au moment où le MAITRE A signale qu'il désire le bus. Si le DEMANDEUR A est mal conçu, cette situation peut l'amener à commander momentanément sa ligne BGOUT* au niveau bas pendant un court instant puis au niveau haut à nouveau, d'où il résulte un niveau bas transitoire.

REGLE 3.37:

Les DEMANDEURS SER doivent être conçus pour garantir qu'aucun niveau bas transitoire ne sera généré sur leur ligne BGOUT*.

OBSERVATION 3.19:

Si le DEMANDEUR SER est conçu de telle manière qu'il mémorise l'état de la ligne locale LE MAITRE DEMANDE LE BUS sur le front descendant de sa ligne BGIN* et si ce signal est en transition quand le front descendant arrive, la sortie de la bascule pourra soit osciller, soit rester dans l'intervalle entre le niveau bas et le niveau haut pendant un court moment. C'est pourquoi la norme VSB n'établit pas un temps limite pour que le DEMANDEUR SER propage l'accord de bus. Elle interdit seulement que le DEMANDEUR SER puisse générer un état bas transitoire sur sa ligne BGOUT* qui pourrait être interprété comme une allocation de bus par un DEMANDEUR SER placé en aval sur la chaîne série.

PERMISSION 3.4:

Si un DEMANDEUR SER détecte que son MAITRE associé demande le bus entre l'instant où il reçoit une allocation de bus destiné à un autre DEMANDEUR SER et l'instant où il propagerait ce signal, il PEUT traiter l'allocation de bus comme lui étant destinée. Dans ce cas, l'autre DEMANDEUR SER maintient sa requête de bus jusqu'à ce qu'une autre allocation de bus soit émise.

3.6 *Spécifications de chronologie du bus d'arbitrage*

Cette section décrit les spécifications temporelles qui gouvernent le comportement des DEMANDEURS PAR. La chronologie est représentée sous forme de figures et de tableaux.

Afin de respecter les REGLES de temps spécifiées, les concepteurs de cartes auront à prendre en considération les cas les plus défavorables de temps de propagation des émetteurs et récepteurs de bus utilisés sur leurs cartes VSB. Les délais de propagation des émetteurs dépendent de leur charge en sortie. Cependant, les spécifications des fabricants ne donnent pas toujours suffisamment d'informations pour calculer les temps de propagation sous diverses charges. Pour aider le concepteur de cartes VSB, quelques suggestions sont fournies au chapitre 4.

For example, if the MASTER notifies its on-board SER REQUESTER during the last data transfer that it no longer needs the bus, the SER REQUESTER releases BUSY*, and arbitration takes place during the last transfer. But if the MASTER waits until the last transfer has completed before signaling its on-board SER REQUESTER, the DTB remains idle while the arbitration is done.

3.5.3 *Race conditions between MASTER requests and ARBITER grants*

Suppose that there are two SER REQUESTERS: SER REQUESTER A and SER REQUESTER B. SER REQUESTER B, which is farther down the daisy-chain, requests the bus and the ARBITER drives the corresponding bus grant line low. This bus grant arrives at SER REQUESTER A just as MASTER A signals that it wants the bus. If SER REQUESTER A is improperly designed, this situation might cause it to momentarily drive its BGOUT* line low and then high again resulting in a low-going transient.

RULE 3.37:
SER REQUESTERS MUST be designed to ensure that no momentary low-going transients are generated on their BGOUT* line.

OBSERVATION 3.19:
If the SER REQUESTER is designed such that it latches the state of the on-board MASTER WANTS BUS line upon the falling edge of its BGIN* line, and if that signal is in transition when the falling edge occurs, the outputs of the latch will sometimes oscillate, or remain in the threshold region between the high and low levels, for a short time. Because of this, the VSB standard does not set a time limit for the SER REQUESTER to pass along the bus grant. It only prohibits the SER REQUESTER from generating a low-going transient on its BGOUT* line which might be interpreted as a bus grant by a SER REQUESTER further down the daisy-chain.

PERMISSION 3.4:
If a SER REQUESTER detects that its associated MASTER needs the bus between the time that it receives a bus grant intended for another SER REQUESTER, and the time it would pass that bus grant on, it MAY treat the bus grant as its own. In this case the other SER REQUESTER maintains its bus request until another bus grant is issued.

3.6 *Arbitration bus timing specifications*

This section describes the timing specifications that govern the behavior of PAR REQUESTERS. This timing information is in the form of figures and tables.

In order to meet the specified timing RULES, board designers need to take into account the worst case propagation delays of the bus drivers and receivers used on their VSB boards. The propagation delay of the drivers depends on their output loads. However, manufacturers' specifications do not always give enough information to calculate the propagation delays under various loads. To help the VSB board designer, some suggestions are offered in Chapter 4.

Les OBSERVATIONS spécifient les temps de transition des signaux entrants. Ces temps sont fiables tant que les REGLES de charges de fond de panier, données au chapitre 4, sont observées. Les REGLES relatives aux impédances d'adaptation, données au chapitre 4, garantissent que les protocoles et les paramètres de temps continuent à être respectés après que les lignes de signaux ont été libérées.

Tableau 3-4

Paramètres de temps d'un DEMANDEUR PAR actif, d'un DEMANDEUR PAR concurrent et d'un ESCLAVE au repos

NUMERO DE PARAMETRE	DEMANDEUR ACTIF		DEMANDEUR CONCURRENT		ESCLAVE AU REPOS	
	MIN.	MAX.	MIN.	MAX.	MIN.	MAX.
2	10		0		0	
7	0		0		0	
26	20		10		10	
27	0		0		0	
30	10		0		0	
31	0		0		0	
32		30		20		20
34		30		20		20
44	30		20		20	
47		20				
49	30					
50	10		0			
51			0			
52			10			
53			0			
54	30		40			
55	0		10			
58	0		0			
59	0		0			
60	0		0			
61	0		0			
62	20		20			
63		20		30		

Notes:

1.- Tous les paramètres de temps sont en nanosecondes.

2.- Les spécifications pour les paramètres de temps sont données dans les tableaux 3-6 et 3-7. Utiliser le numéro de paramètre pour localiser séquentiellement la spécification qui lui est associée.

3.- Les diagrammes correspondant aux paramètres 2, 26, 30, 44, 47 et 49 se trouvent au chapitre 2.

The OBSERVATIONS specify the timing of incoming signal line transitions. These times can be relied upon as long as the backplane loading RULES in Chapter 4 are not violated. The RULES for the bus terminators in Chapter 4 guarantee that the protocol and timing parameters continue to be met after signal lines are released.

Table 3-4

Active PAR REQUESTER, contending PAR REQUESTER and idle SLAVE timing parameters

PARAMETER NUMBER	ACTIVE REQUESTER		CONTENDING REQUESTER		IDLE SLAVE	
	MIN.	MAX.	MIN.	MAX.	MIN.	MAX.
2	10		0		0	
7	0		0		0	
26	20		10		10	
27	0		0		0	
30	10		0		0	
31	0		0		0	
32		30		20		20
34		30		20		20
44	30		20		20	
47		20				
49	30					
50	10		0			
51			0			
52			10			
53			0			
54	30		40			
55	0		10			
58	0		0			
59	0		0			
60	0		0			
61	0		0			
62	20		20			
63		20		30		

Notes:

1.- All timing parameters are in nanoseconds.

2.- The specifications for the timing parameters are given in Tables 3-6 and 3-7. Use the timing parameter number to sequentially locate its associated specification.

3.- The timing diagrams for parameters 2, 26, 30, 44, 47 and 49 can be found in Chapter 2.

Tableau 3-5

Paramètres de temps à la mise sous tension

NUMERO DE PARAMETRE	DEMANDEUR PAR ACTIF		TOUS LES DEMANDEURS PAR	
	MIN.	MAX.	MIN.	MAX.
64				5
65				5
66			200	
67			10	
68			0	
69			30	
70			20	
71	40			
72			30	

Notes:

1.- Les paramètres de temps 64-66 sont en millisecondes. Tous les autres paramètres de temps sont en nanosecondes.

2.- Les spécifications pour les paramètres de temps sont données au tableau 3-8. Utiliser le numéro de paramètre pour localiser séquentiellement la spécification qui lui est associée.

Table 3-5

Power-up timing parameters

PARAMETER NUMBER	ACTIVE PAR REQUESTER		ALL PAR REQUESTERS	
	MIN.	MAX.	MIN.	MAX.
64				5
65				5
66			200	
67			10	
68			0	
69			30	
70			20	
71	40			
72			30	

Notes:

1.- Timing parameters 64-66 are in milliseconds. All other timing parameters are in nanoseconds.

2.- The specifications for the timing parameters are given in Table 3-8. Use the timing parameter number to sequentially locate its associated specification.

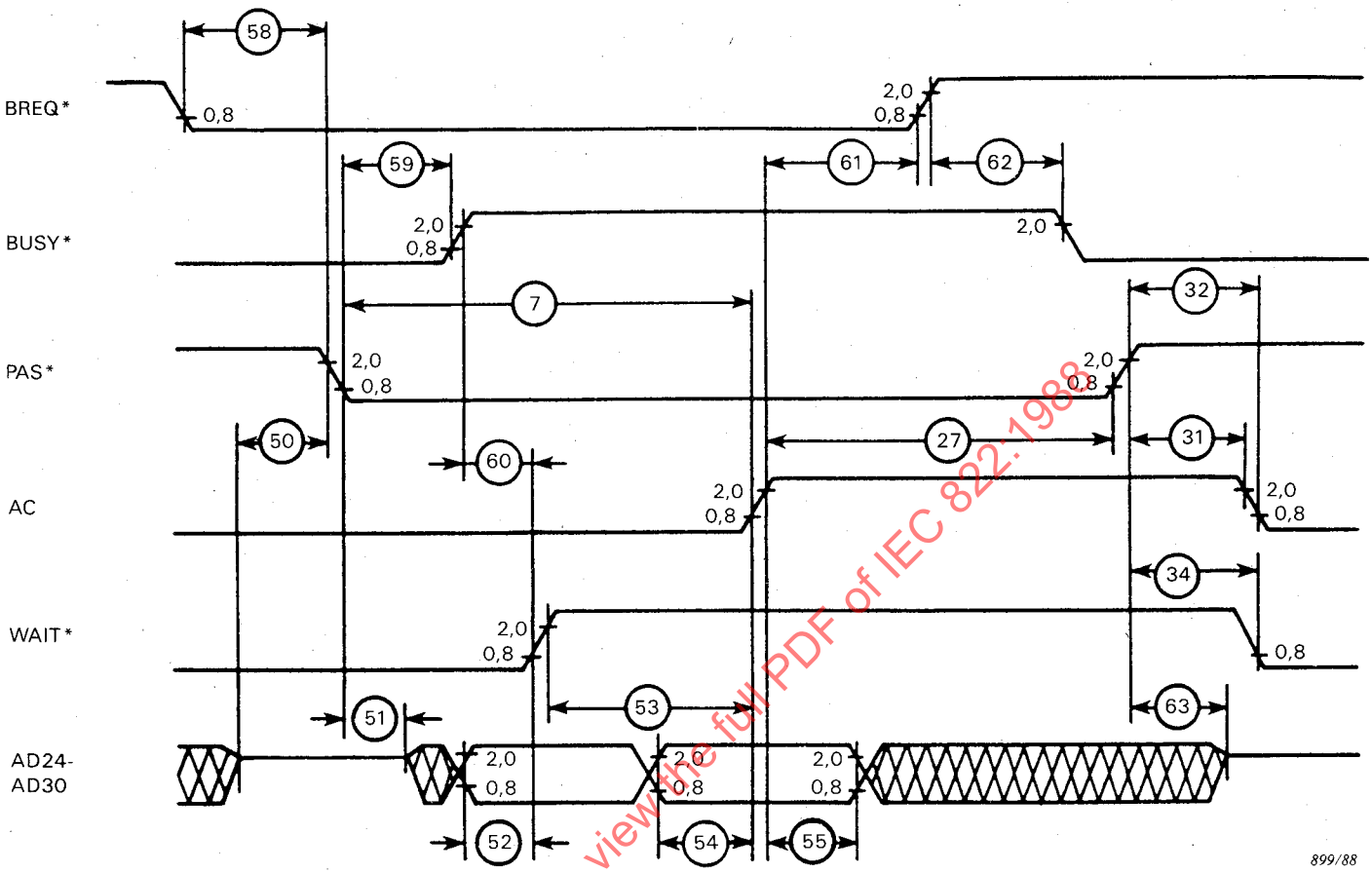


Fig. 3-9. - DEMANDEUR PAR actif, DEMANDEUR PAR concurrent et ESCLAVE au repos: cycle d'ARBITRAGE.

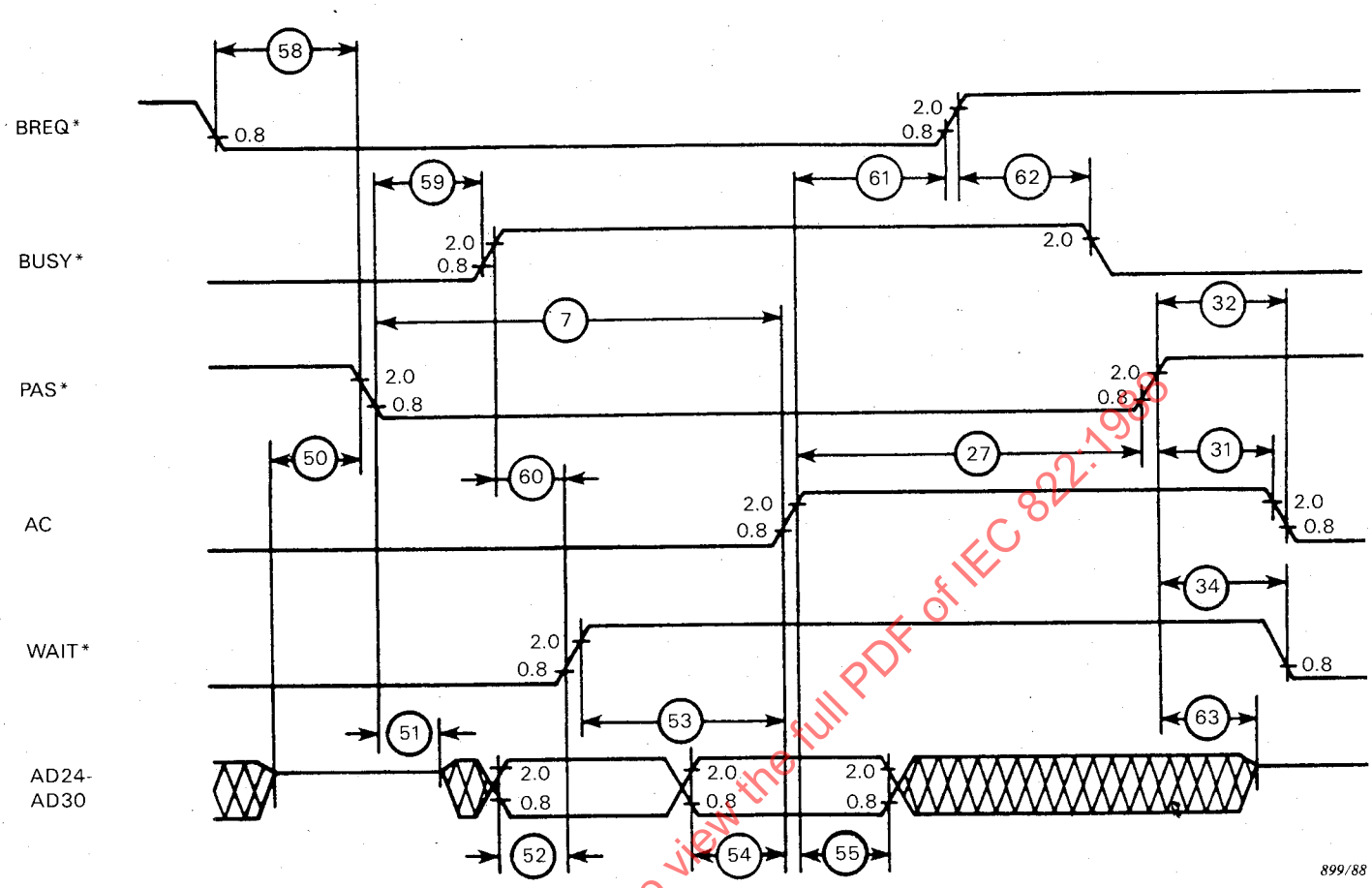
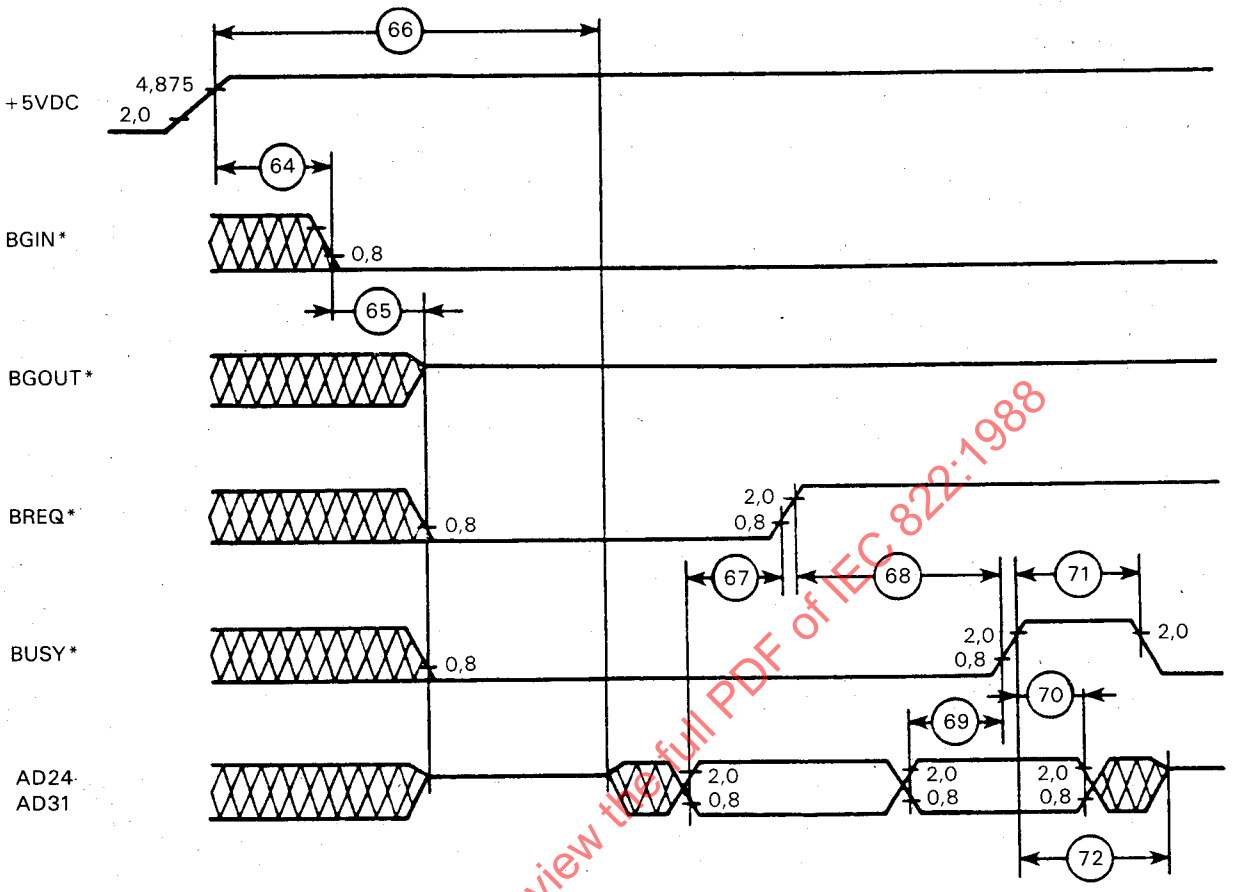


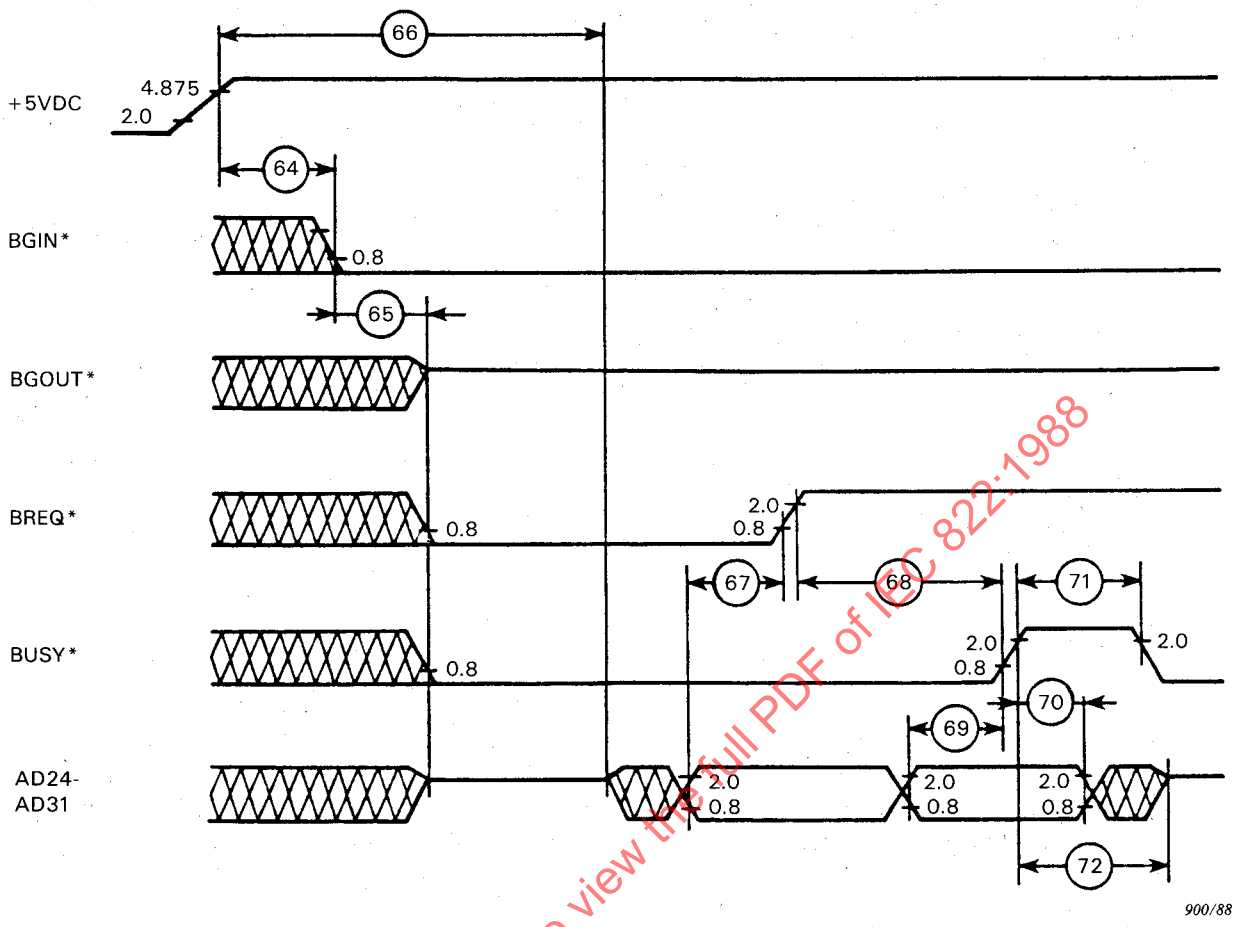
Fig. 3-9. - Active PAR REQUESTER, contending PAR REQUESTER and idle SLAVE:
ARBITRATION cycle.



Note:

Ce chronogramme ne s'applique que si aucun DEMANDEUR SER ni aucun ARBITRE ne sont installés dans le fond de panier.

Fig. 3-10.- Chronogramme du démarrage à la mise sous tension.



Note:

This timing diagram only applies when no SER REQUESTERS or ARBITER are installed in the backplane.

Fig. 3-10. - Power-up timing.

Tableau 3-6

Spécifications de chronologie du DEMANDEUR actif

Note: La numérotation correspond aux paramètres de temps spécifiés dans le tableau 3-4.

2. REGLE 3.38:
Le DEMANDEUR actif NE DOIT PAS commander PAS* au niveau bas tant que WR* et SPACE0-SPACE1 n'ont pas été au niveau bas au minimum pendant ce temps.

7. OBSERVATION 3.20:
Le DEMANDEUR actif est assuré que les DEMANDEURS concurrents ne libéreront pas AC au niveau haut avant ce délai après que le DEMANDEUR actif a commandé PAS* au niveau bas.

26. REGLE 3.39:
Le DEMANDEUR actif DOIT maintenir PAS* au niveau bas au minimum pendant ce temps.

27. REGLE 3.40:
Durant les cycles d'ARBITRAGE, le DEMANDEUR ACTIF NE DOIT PAS libérer PAS* au niveau haut avant ce délai après la détection de AC au niveau haut.

30. REGLE 3.41:
Le DEMANDEUR actif NE DOIT PAS changer les niveaux de WR* ou SPACE0-SPACE1 avant ce délai après avoir libéré PAS* au niveau haut.

31. OBSERVATION 3.21:
Le DEMANDEUR actif a la garantie que les DEMANDEURS concurrents ne commanderont pas AC au niveau bas ni ne libéreront ASACK0*-ASACK1* au niveau haut avant ce délai après que le MAITRE actif a libéré PAS* au niveau haut.

32. OBSERVATION 3.22:
Le MAITRE actif est assuré que les DEMANDEURS concurrents et les ESCLAVES au repos commanderont AC au niveau bas pendant ce délai après que PAS* est passé au niveau haut.

34. OBSERVATION 3.23:
Le DEMANDEUR actif est assuré que les DEMANDEURS concurrents et que les ESCLAVES qui le peuvent commanderont WAIT* au niveau bas pendant ce délai après que PAS* a été au niveau haut.

44. REGLE 3.42:
Le DEMANDEUR actif NE DOIT PAS commander PAS* au niveau bas tant qu'il n'est pas resté au niveau haut au minimum pendant ce temps.

47. REGLE 3.43:
Lorsqu'il termine un cycle d'ARBITRAGE, le DEMANDEUR actif DOIT libérer PAS*, WR* et SPACE0-SPACE1 dans ce délai après avoir permis à PAS* de passer au niveau haut.

Table 3-6

Active REQUESTER timing specifications

Note: The numbers correspond to the timing parameters specified in Table 3-4.

2. RULE 3.38:
The active REQUESTER MUST NOT drive PAS* low until WR* and SPACE0-SPACE1 have been low for this minimum time.

7. OBSERVATION 3.20:
The active REQUESTER is guaranteed that the contending REQUESTERS will not release AC to high until this time after the active REQUESTER has driven PAS* low.

26. RULE 3.39:
The active REQUESTER MUST maintain PAS* low for this minimum time.

27. RULE 3.40:
During ARBITRATION cycles, the active REQUESTER MUST NOT allow PAS* to go high until this time after detecting AC high.

30. RULE 3.41:
The active REQUESTER MUST NOT change the levels of WR* or SPACE0-SPACE1 until this time after it allows PAS* to go high.

31. OBSERVATION 3.21:
The active REQUESTER is guaranteed that contending REQUESTERS will neither drive AC to low nor release ASACK0*-ASACK1* to high until this time after the active REQUESTER allows PAS* to go high.

32. OBSERVATION 3.22:
The active REQUESTER is guaranteed that contending REQUESTERS and idle SLAVES will drive AC low within this time after PAS* is high.

34. OBSERVATION 3.23:
The active REQUESTER is guaranteed that contending REQUESTERS, and those SLAVES that can, will drive WAIT* low, within this time after PAS* is high.

44. RULE 3.42:
The active REQUESTER MUST NOT drive PAS* low until it has been high for this minimum time.

47. RULE 3.43:
When terminating an ARBITRATION cycle, the active REQUESTER MUST release PAS*, WR* and SPACE0-SPACE1 within this time after allowing PAS* to go high.

49. REGLE 3.44:
Quand il déclenche un cycle d'ARBITRAGE, le DEMANDEUR actif NE DOIT PAS commander WR*, SPACE0-SPACE1 ou PAS* avant ce délai après que son MAITRE a permis à PAS* de passer au niveau haut.

50. REGLE 3.45:
Pendant les cycles d'ARBITRAGE, le DEMANDEUR actif DOIT libérer AD24-AD30 au minimum pendant ce temps avant de commander PAS* au niveau bas.

54. OBSERVATION 3.24:
Pendant un cycle d'ARBITRAGE, le DEMANDEUR actif est assuré que AD24-AD30 véhiculent un ID ARBITRAGE valide au minimum pendant ce temps quand il détecte AC au niveau haut.

55. OBSERVATION 3.25:
Le DEMANDEUR actif est assuré que l'ID ARBITRAGE du nouveau DEMANDEUR actif sélectionné reste validé sur AD24-AD30 au minimum pendant ce temps après avoir détecté AC au niveau haut.

58. REGLE 3.46:
Le DEMANDEUR actif NE DOIT PAS commander PAS* au niveau bas avant ce délai après que BREQ* est passé au niveau bas.

59. REGLE 3.47:
Le DEMANDEUR actif NE DOIT PAS libérer BUSY* au niveau haut avant ce délai après avoir commandé PAS* au niveau bas.

60. REGLE 3.48:
Le DEMANDEUR actif NE DOIT PAS libérer au niveau haut sa contribution à la ligne WAIT* avant ce délai après avoir libéré BUSY* au niveau haut.

61. REGLE 3.49:
Le nouveau DEMANDEUR actif NE DOIT PAS libérer au niveau haut sa contribution à la ligne BREQ* avant ce délai après que AC est passé au niveau haut.

62. REGLE 3.50:
Le nouveau DEMANDEUR actif NE DOIT PAS commander BUSY* au niveau bas avant ce délai après qu'il a libéré au niveau haut sa contribution à la ligne BREQ*.

63. OBSERVATION 3.26:
Le DEMANDEUR actif est assuré que les DEMANDEURS concurrents libéreront AD24-AD30 dans ce délai après que PAS* est passé au niveau haut.

49. RULE 3.44:
When initiating an ARBITRATION cycle, the active REQUESTER MUST NOT drive WR*, SPACE0-SPACE1, or PAS* until this time after its MASTER allows PAS* to go high.

50. RULE 3.45:
During ARBITRATION cycles, the active REQUESTER MUST release AD24-AD30 this minimum time before driving PAS* low.

54. OBSERVATION 3.24:
During an ARBITRATION cycle, the active REQUESTER is guaranteed that AD24-AD30 carried a valid ARBITRATION ID for this minimum time when it detects AC high.

55. OBSERVATION 3.25:
The active REQUESTER is guaranteed that the ARBITRATION ID of the newly selected active REQUESTER remains valid on AD24-AD30 for this minimum time after it detects AC high.

58. RULE 3.46:
The active REQUESTER MUST NOT drive PAS* low until this time after BREQ* is low.

59. RULE 3.47:
The active REQUESTER MUST NOT release BUSY* to high until this time after driving PAS* low.

60. RULE 3.48:
The active REQUESTER MUST NOT release its contribution to the WAIT* line to high until this time after releasing BUSY* to high.

61. RULE 3.49:
The new active REQUESTER MUST NOT release its contribution to the BREQ* line to high until this time after AC is high.

62. RULE 3.50:
The new active REQUESTER MUST NOT drive BUSY* low until this time after it released its contribution to the BREQ* line to high.

63. OBSERVATION 3.26:
The active REQUESTER is guaranteed that contending REQUESTERS will release AD24-AD30 within this time after PAS* is high.

Tableau 3-7

Spécifications de chronologie des DEMANDEURS concurrents

Note: La numérotation correspond aux paramètres de temps spécifiés dans le tableau 3-4.

2. OBSERVATION 3.27:
Les DEMANDEURS concurrents et les ESCLAVES au repos sont assurés que le DEMANDEUR actif ne commandera pas PAS* au niveau bas tant que WR* et SPACE0-SPACE1 n'auront pas été au niveau bas au minimum pendant ce temps.

7. REGLE 3.51:
Les DEMANDEURS concurrents et les ESCLAVES au repos NE DOIVENT PAS libérer AC au niveau haut avant ce délai après le passage au niveau bas de PAS*.

26. OBSERVATION 3.28:
Les DEMANDEURS concurrents et les ESCLAVES au repos sont assurés que le DEMANDEUR actif maintiendra PAS* au niveau bas au minimum pendant ce temps.

27. OBSERVATION 3.29:
Les DEMANDEURS concurrents et les ESCLAVES au repos sont assurés que le DEMANDEUR actif ne libérera pas PAS* au niveau haut avant ce délai après avoir détecté AC au niveau haut.

30. OBSERVATION 3.30:
Les DEMANDEURS concurrents sont assurés que le DEMANDEUR actif ne changera pas les niveaux de WR* et SPACE0-SPACE1 avant ce délai après avoir libéré PAS* au niveau haut.

31. REGLE 3.52:
Les DEMANDEURS PAR et les ESCLAVES au repos NE DOIVENT PAS commander AC au niveau bas avant ce délai après la remontée de PAS* au niveau haut.

32. REGLE 3.53:
Les DEMANDEURS concurrents doivent commander AC au niveau bas dans ce délai après le passage au niveau haut de PAS*.

34. REGLE 3.54:
Les DEMANDEURS concurrents et les ESCLAVES qui le peuvent DOIVENT commander WAIT* au niveau bas dans ce délai après le passage au niveau haut de PAS*.

44. OBSERVATION 3.31:
Les DEMANDEURS concurrents et les ESCLAVES au repos sont assurés que lorsque PAS* remonte au niveau haut, il le restera au minimum pendant ce temps.

50. OBSERVATION 3.32:
Durant les cycles d'ARBITRAGE, les DEMANDEURS concurrents sont assurés que le MAITRE DU DEMANDEUR actif libérera AD24-AD30 au minimum pendant ce temps avant que le DEMANDEUR actif commande PAS* au niveau bas.

Table 3-7

Contending REQUESTER timing specifications

Note: The numbers correspond to the timing parameters specified in Table 3-4.

2. OBSERVATION 3.27:
Contending REQUESTERS and idle SLAVES are guaranteed that the active REQUESTER will not drive PAS* low until WR* and SPACE0-SPACE1 have been low for this minimum time.

7. RULE 3.51:
Contending REQUESTERS and idle SLAVES MUST NOT release AC to high until this time after PAS* is low.

26. OBSERVATION 3.28:
Contending REQUESTERS and idle SLAVES are guaranteed that the active REQUESTER will maintain PAS* low for this minimum time.

27. OBSERVATION 3.29:
Contending REQUESTERS and idle SLAVES are guaranteed that the active REQUESTER will not allow PAS* to go high until this time after it detects AC high.

30. OBSERVATION 3.30:
Contending REQUESTERS are guaranteed that the active REQUESTER will not change the levels of WR* and SPACE0-SPACE1 until this time after it allows PAS* to go high.

31. RULE 3.52:
PAR REQUESTERS and idle SLAVES MUST NOT drive AC low until this time after PAS* is high.

32. RULE 3.53:
Contending REQUESTERS MUST drive AC to low within this time after PAS* is high.

34. RULE 3.54:
Contending REQUESTERS and those SLAVES that can, MUST drive WAIT* low, within this time after PAS* is high.

44. OBSERVATION 3.31:
Contending REQUESTERS and idle SLAVES are guaranteed that once PAS* is allowed to go high, it will remain high for this minimum time.

50. OBSERVATION 3.32:
During ARBITRATION cycles, contending REQUESTERS are guaranteed that the active REQUESTER'S MASTER will release AD24-AD30 this minimum time before the active REQUESTER drives PAS* low.

51. REGLE 3.55:
Durant le cycle d'ARBITRAGE, les DEMANDEURS concurrents NE DOIVENT commander aucune des lignes AD24-AD30 avant ce délai après le passage de PAS* au niveau bas.

OBSERVATION 3.33:
Les DEMANDEURS concurrents sont assurés qu'aucun DEMANDEUR concurrent ne pilotera aucune des lignes AD24-AD30 avant ce délai après le passage de PAS* au niveau bas.

52. REGLE 3.56:
Durant le cycle d'ARBITRAGE, les DEMANDEURS concurrents NE DOIVENT PAS libérer WAIT* au niveau haut avant ce délai après positionnement de leur ID ARBITRAGE sur AD24-AD30.

OBSERVATION 3.34:
Les DEMANDEURS concurrents sont assurés que l'ID ARBITRAGE sur AD24-AD30 est valide lorsqu'ils détectent WAIT* au niveau haut.

53. REGLE 3.57:
Les DEMANDEURS concurrents NE DOIVENT PAS libérer AC au niveau haut au minimum pendant ce temps après libération de WAIT* au niveau haut.

OBSERVATION 3.35:
Les DEMANDEURS concurrents sont assurés qu'aucun DEMANDEUR concurrent ne libérera AC au niveau haut avant ce délai après libération de WAIT* au niveau haut.

54. REGLE 3.58:
Durant le cycle d'arbitrage, les DEMANDEURS concurrents DOIVENT placer un ID ARBITRAGE valide sur les lignes AD24-AD30 au minimum pendant ce temps avant de libérer au niveau haut leur contribution à la ligne AC.

OBSERVATION 3.36:
Les DEMANDEURS concurrents sont assurés que AD24-AD30 véhiculent un ID ARBITRAGE valide au minimum pendant ce temps lorsqu'il détecte AC au niveau haut.

55. REGLE 3.59:
Les DEMANDEURS concurrents DOIVENT maintenir leur contribution à l'ID ARBITRAGE sur AD24-AD30 au minimum pendant ce temps après la libération de AC au niveau haut.

OBSERVATION 3.37:
Les DEMANDEURS concurrents sont assurés que l'ID ARBITRAGE du demandeur actif nouvellement sélectionné reste valide sur AD24-AD30 au minimum pendant ce temps après qu'ils ont détecté AC au niveau haut.

58. OBSERVATION 3.38:
Les DEMANDEURS concurrents sont assurés que le DEMANDEUR actif ne commandera pas PAS* au niveau bas avant ce délai après positionnement de BREQ* au niveau bas.

51. RULE 3.55:
During the ARBITRATION cycle, contending REQUESTERS MUST NOT drive any of AD24-AD30 until this time after PAS* is low.

OBSERVATION 3.33:
Contending REQUESTERS are guaranteed that no contending REQUESTER will drive any of AD24-AD30 until this time after PAS* is low.

52. RULE 3.56:
During the ARBITRATION cycle, contending REQUESTERS MUST NOT release WAIT* to high until this time after they drive their ARBITRATION ID on AD24-AD30.

OBSERVATION 3.34:
Contending REQUESTERS are guaranteed that the ARBITRATION ID on AD24-AD30 is valid when they detect WAIT* high.

53. RULE 3.57:
Contending REQUESTERS MUST NOT release AC to high for this minimum time after releasing WAIT* to high.

OBSERVATION 3.35:
Contending REQUESTERS are guaranteed that no contending REQUESTER will release AC to high until this time after releasing WAIT* to high.

54. RULE 3.58:
During the ARBITRATION cycle, contending REQUESTERS MUST drive a valid ARBITRATION ID on AD24-AD30 for this minimum time before releasing their contribution to the AC line to high.

OBSERVATION 3.36:
Contending REQUESTERS are guaranteed that AD24-AD30 carried a valid ARBITRATION ID for this minimum time when they detect AC high.

55. RULE 3.59:
Contending REQUESTERS MUST maintain their contribution to the ARBITRATION ID on AD24-AD30 for this minimum time after releasing AC to high.

OBSERVATION 3.37:
Contending REQUESTERS are guaranteed that the ARBITRATION ID of the newly selected active REQUESTER remains valid on AD24-AD30 for this minimum time after they detect AC high.

58. OBSERVATION 3.38:
Contending REQUESTERS are guaranteed that the active REQUESTER will not drive PAS* low until this time after BREQ* is low.

59. OBSERVATION 3.39:
Les DEMANDEURS concurrents sont assurés que le DEMANDEUR actif ne libérera pas BUSY* au niveau haut avant ce délai après le positionnement de PAS* au niveau bas.

60. OBSERVATION 3.40:
Les DEMANDEURS concurrents sont assurés que le DEMANDEUR actif ne libérera pas au niveau haut la ligne WAIT* avant ce délai après la libération de BUSY* au niveau haut.

61. OBSERVATION 3.41:
Les DEMANDEURS concurrents sont assurés que le nouveau DEMANDEUR actif ne libérera pas au niveau haut la ligne BREQ* avant ce délai après le passage au niveau haut de AC.

62. OBSERVATION 3.42:
Les DEMANDEURS concurrents sont assurés que le nouveau DEMANDEUR actif ne positionnera pas BUSY* au niveau bas avant ce délai, après qu'il a libéré au niveau haut la ligne BREQ*.

63. REGLE 3.60:
Les DEMANDEURS concurrents DOIVENT libérer AD24-AD30 avant ce délai après le passage de PAS* au niveau haut.

Tableau 3-8

Spécifications de chronologie à la mise sous tension

Note: La numérotation correspond aux paramètres de temps donnés dans le tableau 3-5.

OBSERVATION 3.43:
Les REGLES additionnelles qui gouvernent le comportement de l'ARBITRE et des DEMANDEURS SER pendant la séquence de démarrage à la mise sous tension sont données au paragraphe 3.4.3.2.

64. REGLE 3.61:
Tous les DEMANDEURS PAR DOIVENT s'assurer que BGIN* est au niveau bas dans ce délai maximal après que la tension a atteint le niveau spécifié.

65. REGLE 3.62:
SI l'entrée BGIN* d'un DEMANDEUR PAR est au niveau bas,
ALORS il DOIT mettre dans l'état haute impédance ses émetteurs de BGOUT* et des lignes AD24-AD31 et commander BREQ* et BUSY* au niveau bas dans ce délai maximal après que la tension a atteint le niveau spécifié.

66. REGLE 3.63:
SI l'entrée BGIN* d'un DEMANDEUR PAR est au niveau bas,
ALORS il DOIT placer un ID ARBITRAGE valide sur les lignes AD24-AD30, mais il NE DOIT PAS le faire avant ce délai après que la tension a atteint le niveau spécifié.

59. OBSERVATION 3.39:
Contending REQUESTERS are guaranteed that the active REQUESTER will not release BUSY* to high until this time after driving PAS* low.

60. OBSERVATION 3.40:
Contending REQUESTERS are guaranteed that the active REQUESTER will not release its contribution to the WAIT* line to high until this time after releasing BUSY* to high.

61. OBSERVATION 3.41:
Contending REQUESTERS are guaranteed that the new active REQUESTER will not release its contribution to the BREQ* line to high until this time after AC is high.

62. OBSERVATION 3.42:
Contending REQUESTERS are guaranteed that the new active REQUESTER will not drive BUSY* low until this time after it released its contribution to the BREQ* line to high.

63. RULE 3.60:
Contending REQUESTERS MUST release AD24-AD30 within this time after PAS* is high.

Table 3-8

Power-up timing specifications

Note: The sequential numbers correspond to the timing parameters given in Table 3-5.

OBSERVATION 3.43:
Additional RULES that govern the behavior of the ARBITER and of SER REQUESTERS during the power-up sequence are given in Paragraph 3.4.3.2.

64. RULE 3.61:
All PAR REQUESTERS MUST ensure that BGIN* is low within this maximum time after power is within its specified range.

65. RULE 3.62:
IF the BGIN* input to a PAR REQUESTER is low,
THEN it MUST tri-state its drivers for BGOUT* and AD24-AD31, and drive BREQ* and BUSY* low within this maximum time after power is within its specified range.

66. RULE 3.63:
IF the BGIN* input to a PAR REQUESTER is low,
THEN it MUST place a valid ARBITRATION ID on AD24-AD30, but it MUST NOT do so until this time after power is within its specified range.

67. REGLE 3.64:
Pendant la séquence de démarrage à la mise sous tension, les DEMANDEURS PAR NE DOIVENT PAS libérer BREQ* au niveau haut avant ce délai minimal après qu'ils ont placé leur ID ARBITRAGE sur les lignes AD24-AD30.

68. REGLE 3.65:
Pendant la séquence de démarrage à la mise sous tension, les DEMANDEURS PAR NE DOIVENT PAS libérer BUSY* au niveau haut avant ce délai après la détection de BREQ* au niveau haut.

69. REGLE 3.66:
Pendant la séquence de démarrage à la mise sous tension, les DEMANDEURS PAR DOIVENT maintenir leur contribution à l'ID ARBITRAGE sur les lignes AD24-AD30 au minimum pendant ce temps avant de libérer BUSY* au niveau haut.

70. REGLE 3.67:
Pendant la séquence de démarrage à la mise sous tension, les DEMANDEURS PAR DOIVENT maintenir un ID ARBITRAGE valide sur AD24-AD30 au minimum pendant ce temps après avoir détecté BUSY* au niveau haut.

71. REGLE 3.68:
Pendant la séquence de démarrage à la mise sous tension, le DEMANDEUR PAR qui a été sélectionné pour être le DEMANDEUR actif initial NE DOIT PAS commander BUSY* au niveau bas pendant ce délai après qu'il a détecté BUSY* au niveau haut.

OBSERVATION 3.44:
Pendant la séquence de démarrage à la mise sous tension, les DEMANDEURS PAR sont assurés que le DEMANDEUR actif initial ne commandera pas BUSY* au niveau bas tant qu'il n'a pas été maintenu au niveau haut au minimum pendant ce temps.

72. REGLE 3.69:
Pendant la séquence de démarrage à la mise sous tension, les DEMANDEURS PAR DOIVENT arrêter de commander AD24-AD30 dans ce délai après qu'ils ont détecté BUSY* au niveau haut.