

**NORME
INTERNATIONALE
INTERNATIONAL
STANDARD**

**CEI
IEC
1052**

Première édition
First edition
1991-04

CEI 1052 ROUTINES STANDARDS FASTBUS

**Routines standards utilisables avec le système
d'acquisition de données FASTBUS**

IEC 1052 FASTBUS STANDARD ROUTINES

**Standard Routines for use with FASTBUS
data acquisition system**



Numéro de référence
Reference number
CEI/IEC 1052: 1991

Numéros des publications

Depuis le 1^{er} 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* (IEV).

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
1052**

Première édition
First edition
1991-04

CEI 1052 ROUTINES STANDARDS FASTBUS

**Routines standards utilisables avec le système
d'acquisition de données FASTBUS**

IEC 1052 FASTBUS STANDARD ROUTINES

**Standard Routines for use with FASTBUS
data acquisition system**

© CEI 1991 Droits de reproduction réservés — Copyright — all rights reserved

Aucune partie de cette publication ne peut être reproduite ni utilisée sous quelque forme que ce soit et par aucun procédé, électronique ou mécanique, y compris la photocopie et les microfilms, sans l'accord écrit de l'éditeur.

No part of this publication may be reproduced or utilized in any form or by any means, electronic or mechanical, including photocopying and microfilm, without permission in writing from the publisher.

Bureau Central de la Commission Electrotechnique Internationale 3, rue de Varembe Genève, Suisse



Commission Electrotechnique Internationale
International Electrotechnical Commission
Международная Электротехническая Комиссия

CODE PRIX
PRICE CODE

XE

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

SOMMAIRE

PREAMBULE	1
PREFACE	1
Section 1. Objet, domaine d'application et introduction	2
1.1 Objet et domaine d'application	2
1.2 Interprétation de la présente norme	2
1.3 Vue d'ensemble du document	2
Section 2. Concepts de base	4
2.1 Terminologie	4
2.2 Environnement	5
2.2.1 Vue d'ensemble	5
2.2.2 Définition de l'environnement	6
2.2.3 Identificateur d'environnement	6
2.2.4 Modes d'exécution	6
2.2.5 Accès au FASTBUS dans un système multi-utilisateur	6
2.2.6 Session FASTBUS	7
2.2.7 Routines d'action	7
2.2.8 Moment d'appel et moment d'exécution	7
2.3 Conventions de dénomination	7
2.4 Paramètres des routines: types et direction	7
2.5 Paramètres des tampons	9
2.6 Catégories des routines	10
Section 3. Gestion de l'environnement et exécution différée	11
3.1 Ouvrir et fermer une session FASTBUS	11
3.1.1 Ouverture d'une session FASTBUS	11
3.1.2 Clôture d'une session FASTBUS	12
3.2 Routines de gestion de l'environnement	12
3.2.1 Création d'un environnement en mode d'exécution immédiate	12
3.2.2 Création d'un environnement en mode d'exécution différée	12
3.2.3 Libération d'un environnement	13
3.2.4 Initialisation d'un environnement	13
3.2.5 Copie d'un environnement	14
3.2.6 Obtention d'un environnement	14
3.2.7 Restauration d'un environnement	14
3.3 Exécution différée et validation des listes	15
3.3.1 Exécution d'une liste	15
3.3.2 Validation d'une liste	15
3.3.3 Obtention du pointeur courant dans une liste spécifiée	15
Section 4. Paramètres opérationnels	17
4.1 Introduction	17
4.1.1 Paramètres opérationnels dans le mode d'exécution différée	17
4.2 Définition des paramètres opérationnels	17
4.3 Routines pour les paramètres opérationnels	18
4.3.1 Initialisation des paramètres opérationnels	18
4.3.2 Réglage d'un paramètre opérationnel	18
4.3.3 Obtention d'un paramètre opérationnel	18
4.3.4 Lecture d'un paramètre opérationnel	19
4.3.5 Ecriture d'un paramètre opérationnel	19
4.3.6 Empilage des paramètres opérationnels courants	20
4.3.7 Dépilage des paramètres opérationnels courants	20
4.4 Paramètres opérationnels globaux et de gestion d'erreur	20
4.4.1 Identificateur du port du contrôleur	20

CONTENTS

FOREWORD	1
PREFACE	1
Section 1. Object, Scope and Introduction	2
1.1 Scope and Object	2
1.2 Interpretation of this Standard	2
1.3 Document Overview	2
Section 2. Basic Concepts	4
2.1 Terminology	4
2.2 Environment	5
2.2.1 Overview	5
2.2.2 Definition of the Environment	6
2.2.3 Environment Identifier	6
2.2.4 Execution Modes	6
2.2.5 Access to FASTBUS in a Multi-user System	6
2.2.6 FASTBUS Session	7
2.2.7 Action Routines	7
2.2.8 Call Time and Execution Time	7
2.3 Naming Conventions	7
2.4 Parameters to Routines: Type and Direction	7
2.5 Buffer Parameters	9
2.6 Categories of Routines	10
Section 3. Environment Management and Delayed Execution	11
3.1 Open and Close FASTBUS Session	11
3.1.1 Open FASTBUS Session	11
3.1.2 Close FASTBUS Session	12
3.2 Environment Management Routines	12
3.2.1 Create an Immediate Execution Mode Environment	12
3.2.2 Create a Delayed Execution Mode Environment	12
3.2.3 Release Environment	13
3.2.4 Reset Environment	13
3.2.5 Copy an Environment	14
3.2.6 Get an Environment	14
3.2.7 Set an Environment	14
3.3 Delayed Execution and List Validation	15
3.3.1 Execute a List	15
3.3.2 Validate a List	15
3.3.3 Get Current Pointer in Specified List	15
Section 4. Operational Parameters	17
4.1 Introduction	17
4.1.1 Operational Parameters in Delayed Execution Mode	17
4.2 Definition of Operational Parameters	17
4.3 Operational Parameter Routines	18
4.3.1 Initialize Operational Parameter	18
4.3.2 Set Operational Parameter	18
4.3.3 Get Operational Parameter	18
4.3.4 Read Operational Parameter	19
4.3.5 Write Operational Parameter	19
4.3.6 Push Current Operational Parameters onto Stack	20
4.3.7 Pop Current Operational Parameters off Stack	20
4.4 Overall and Error Handling Operational Parameters	20
4.4.1 Controller Port Identifier	20

4.4.2	Absence d'attente de fin d'exécution des actions	21
4.4.3	Retard à l'exécution	21
4.4.4	Taille de l'environnement	21
4.4.5	Limitation de la génération des comptes rendus d'état	21
4.4.6	Limitation du code de retour au niveau de sévérité	22
4.4.7	Seuil de sévérité pour le traitement des erreurs	22
4.4.8	Seuil de sévérité pour les rapports d'erreur	22
4.4.9	Seuil de sévérité d'exception	22
4.4.10	Seuil de sévérité des messages	23
4.4.11	Concision des messages de rapport	23
4.4.12	Rapport sur chaque action	23
4.5	Paramètres opérationnels du protocole FASTBUS, des temporisateurs et des réessais	23
4.5.1	Niveau d'arbitrage	24
4.5.2	Arbitrage dans le protocole d'accès assuré	24
4.5.3	Arbitrage dans le protocole d'accès prioritaire	24
4.5.4	Conservation de la maîtrise du bus pour un bloc complet	24
4.5.5	Dimension des paquets dans un transfert de bloc FASTBUS	25
4.5.6	Transfert en pipe-line	25
4.5.7	Période des cycles de données dans un transfert en pipe-line	25
4.5.8	Dispositif à NTA fixe	25
4.5.9	Transfert des données en mots courts	26
4.5.10	Longueur des mots courts	26
4.5.11	Contrôle de la génération de la parité	26
4.5.12	Mise en service de l'adressage géographique	26
4.5.13	Absence de cycle d'arbitrage	26
4.5.14	Absence de cycle d'adressage primaire	27
4.5.15	Absence de cycle d'adresse secondaire pour les opérations dans l'espace données	27
4.5.16	Absence de cycle d'adresse secondaire pour les opérations dans l'espace CSR	27
4.5.17	Absence de cycle de données	27
4.5.18	Conservation de la maîtrise du bus	28
4.5.19	Conservation de la maîtrise du bus si AR n'est pas positionné	28
4.5.20	Maintien de la connexion d'adresse	28
4.5.21	Maintien des signaux du maître après une erreur	28
4.5.22	Durée du temporisateur long	29
4.5.23	Mise hors service du temporisateur long	29
4.5.24	Maintien du temporisateur long	29
4.5.25	Durée du temporisateur d'attente	29
4.5.26	Mise hors service du temporisateur d'attente	30
4.5.27	Durée du temporisateur d'adresse	30
4.5.28	Mise hors service du temporisateur d'adresse	30
4.5.29	Durée du temporisateur de données	30
4.5.30	Mise hors service du temporisateur de données	30
4.5.31	Durée du temporisateur logiciel	31
4.5.32	Mise hors service du temporisateur logiciel	31
4.5.33	Maintien du temporisateur logiciel	31
4.5.34	Nombre de réessais	31
4.5.35	Délai de réessai	31
Section 5.	Tampons de données	33
5.1	Paramètres d'accès à un tampon	33
5.1.1	Tampons définis comme une variable ou un tableau	33
5.1.2	Tampons définis comme une valeur	33
5.1.3	Tampons définis par l'identificateur d'un tampon séquentiel	33
5.2	Routines des tampons séquentiels de données	34
5.2.1	Déclaration d'un tampon interne	34
5.2.2	Déclaration d'un tampon externe	34
5.2.3	Libération d'un tampon séquentiel interne de données	34
5.2.4	Lecture du pointeur d'un tampon séquentiel	35
5.2.5	Ecriture du pointeur d'un tampon séquentiel	35

4.4.2 Do Not Wait for Completion of Action(s)	21
4.4.3 Delay Execution	21
4.4.4 Environment Size	21
4.4.5 Limit Status Generation	21
4.4.6 Return Code contains only the Severity	22
4.4.7 Error Handler Severity Threshold	22
4.4.8 Error Reporting Severity Threshold	22
4.4.9 Exception Severity Threshold	22
4.4.10 Message Severity Threshold	23
4.4.11 Report Terse Message	23
4.4.12 Report On Each Action	23
4.5 FASTBUS Protocol, Timeout and Retry Operational Parameters	23
4.5.1 Arbitration Level	24
4.5.2 Assured Access Arbitration Protocol	24
4.5.3 Prioritized Access Arbitration Protocol	24
4.5.4 Hold Mastership for Whole Block	24
4.5.5 FASTBUS Block Transfer Blocklet Size	25
4.5.6 Pipelined Transfer	25
4.5.7 Pipelined Transfer Data Cycle Clock Time	25
4.5.8 Fixed NTA Device	25
4.5.9 Transfer Data as Short Words	26
4.5.10 Short Word Size	26
4.5.11 Control of Parity Generation	26
4.5.12 Enable Geographic Addressing	26
4.5.13 No Arbitration Cycle	26
4.5.14 No Primary Address Cycle	27
4.5.15 No Secondary Address Cycle for Data Space Operations	27
4.5.16 No Secondary Address Cycle for CSR Space Operations	27
4.5.17 No Data Cycle	27
4.5.18 Hold Bus Mastership	28
4.5.19 Hold Bus Mastership if AR not Asserted	28
4.5.20 Hold Address Connection	28
4.5.21 Hold Master Signals on Error	28
4.5.22 Long Timer	29
4.5.23 Disable Long Timer	29
4.5.24 Maintain Long Timer	29
4.5.25 Wait Timer	29
4.5.26 Disable Wait Timer	30
4.5.27 Address Timer	30
4.5.28 Disable Address Timer	30
4.5.29 Data Timer	30
4.5.30 Disable Data Timer	30
4.5.31 Software Timer	31
4.5.32 Disable Software Timer	31
4.5.33 Maintain Software Timer	31
4.5.34 Number of Retries	31
4.5.35 Retry Time	31
Section 5. Data Buffers	33
5.1 Buffer Access Arguments	33
5.1.1 Buffer specified as a Variable or Array	33
5.1.2 Buffer specified as a Value	33
5.1.3 Buffer specified as a Sequential Buffer ID	33
5.2 Sequential Data Buffer Routines	34
5.2.1 Declare Internal Buffer	34
5.2.2 Declare External Buffer	34
5.2.3 Release Internal Sequential Data Buffer	34
5.2.4 Read Sequential Buffer Pointer	35
5.2.5 Write Sequential Buffer Pointer	35

5.2.6 Déplacement du pointeur d'un tampon séquentiel	35
Section 6. Routines de transaction simples	37
6.1 Conditions s'appliquant aux routines de transaction et aux routines complexes	37
6.1.1 Arbitrage	37
6.1.2 Adressage primaire	37
6.1.3 Cycle d'écriture de l'adresse secondaire	38
6.1.4 Cycles de données	38
6.1.5 Transfert de bloc	38
6.1.6 Déconnexion	39
6.2 Routines de transaction simples FASTBUS	39
6.2.1 Transfert d'un mot unique de données	39
6.2.2 Transfert de bloc	40
6.2.3 Routines d'adresse secondaire	41
6.2.4 Lecture de la longueur du dernier transfert de données	42
Section 7. Routines de transaction complexes	43
7.1 Accès à la table de routage de l'interconnexion de segment	43
7.2 Déplacement de données entre des dispositifs FASTBUS	43
7.3 Lecture-modification-écriture dans un registre FASTBUS	44
7.4 Routines de concentration de données	45
7.4.1 Lecture des données dans des dispositifs adressés séquentiellement	45
7.4.2 Lecture des données dans un dispositif qui répond à une scrutation des contacts T	46
7.4.3 Attribution d'un dispositif FASTBUS	46
7.4.4 Libération d'un dispositif FASTBUS	46
7.5 Envoi d'un message d'interruption FASTBUS	47
Section 8. Routines primitives d'action FASTBUS	48
8.1 Introduction	48
8.2 Paramètres	48
8.3 Routines de cycles individuels	48
8.3.1 Arbitrage	48
8.3.2 Relâcher la maîtrise du bus	49
8.3.3 Cycle d'adressage primaire	49
8.3.4 Déconnexion	49
8.3.5 Transfert d'un mot unique	50
8.3.6 Transfert de bloc	50
8.4 Accès aux lignes FASTBUS	51
8.4.1 Lecture de ligne(s) individuelle(s) du bus	52
8.4.2 Écriture de ligne(s) individuelle(s) du bus	52
8.4.3 Lecture de ligne(s) interne(s)	52
Section 9. Routines des SR et des messages d'interruption FASTBUS	54
9.1 Introduction	54
9.2 Demande de service FASTBUS	54
9.2.1 Effacer le positionnement de SR	55
9.2.2 Connecter une routine à SR	55
9.2.3 Déconnecter une routine de SR	55
9.2.4 Mettre en service la connexion SR	56
9.2.5 Mettre hors service la connexion SR	56
9.3 Routines des messages d'interruption FASTBUS	56
9.3.1 Connecter une routine à FIR	56
9.3.2 Déconnecter une routine de FIR	57
9.3.3 Mettre en service la connexion FIR	57
9.3.4 Mettre hors service la connexion FIR	58
Section 10. Synchronisation, ressources système et routines des ports	59
10.1 Outils de synchronisation	59
10.1.1 Attendre la fin d'une opération	59

5.2.6 Displace Sequential Buffer Pointer	35
Section 6. Simple Transaction Routines	37
6.1 Conditions Governing Transaction and Compound Routines	37
6.1.1 Arbitration	37
6.1.2 Primary Address	37
6.1.3 Secondary Address Write Cycle	38
6.1.4 Data Cycles	38
6.1.5 Block Transfers	38
6.1.6 Disconnection	39
6.2 Simple FASTBUS Transaction Routines	39
6.2.1 Single Data Word Transfer	39
6.2.2 Block Transfer	40
6.2.3 Secondary Address Routines	41
6.2.4 Read Length of Last Data Transfer	42
Section 7. Compound Transaction Routines	43
7.1 Access Segment Interconnect Route Table	43
7.2 Move Data between FASTBUS Devices	43
7.3 Read-Modify-Write a FASTBUS Location	44
7.4 Data Gathering Routines	45
7.4.1 Read Data from Sequential Device Addresses	45
7.4.2 Read Data from Devices that Respond to a T-pin Scan	46
7.4.3 Allocate a FASTBUS Device	46
7.4.4 Deallocate a FASTBUS Device	46
7.5 Send a FASTBUS Interrupt Message	47
Section 8. Primitive FASTBUS Action Routines	48
8.1 Introduction	48
8.2 Parameters	48
8.3 Single Cycle Routines	48
8.3.1 Arbitrate	48
8.3.2 Release Bus Mastership	49
8.3.3 Primary Address Cycle	49
8.3.4 Disconnection	49
8.3.5 Single Word Transfers	50
8.3.6 Block Transfers	50
8.4 FASTBUS Line Access	51
8.4.1 Read Single Bus Line(s)	52
8.4.2 Write Single Bus Line(s)	52
8.4.3 Read Internal Line(s)	52
Section 9. FASTBUS SR and Interrupt Message Routines	54
9.1 Introduction	54
9.2 FASTBUS Service Request	54
9.2.1 Clearing the SR Assertion	55
9.2.2 Connect Routine to SR	55
9.2.3 Disconnect Routine from SR	55
9.2.4 Enable SR Connections	56
9.2.5 Disable SR Connections	56
9.3 FASTBUS Interrupt Message Routines	56
9.3.1 Connect Routine to FIR	56
9.3.2 Disconnect Routine from FIR	57
9.3.3 Enable FIR Connections	57
9.3.4 Disable FIR Connections	58
Section 10. Synchronization, system resource and port Routines	59
10.1 Synchronization Tools	59
10.1.1 Wait for Completion of Operation	59

10.1.2	Contrôler l'état d'exécution	59
10.1.3	Connecter la routine de fin d'exécution	60
10.1.4	Déconnecter la routine de fin d'exécution	60
10.2	Réinitialiser un port FASTBUS	60
10.3	Allocation d'un port FASTBUS	61
10.3.1	Allouer un port FASTBUS	61
10.3.2	Désallouer un port FASTBUS	61
10.3.3	Allouer un port FASTBUS avec temporisateur	61
10.4	Obtenir le numéro de version	61
10.5	Rapporter une erreur du port FASTBUS	62
10.6	Emettre un message d'erreur du port FASTBUS	62
Section 11.	Traitement des états et des erreurs	63
11.1	Introduction	63
11.2	Codes d'erreur et de retour	63
11.3	Etats résumés et complémentaires	63
11.4	Génération limitée d'informations d'état	64
11.5	Etat en mode d'exécution différée	64
11.6	Sévérité	64
11.6.1	FB_SEV_SUCCESS (FSSUCC)	64
11.6.2	FB_SEV_INFO (FSINFO)	64
11.6.3	FB_SEV_WARNING (FSWARN)	65
11.6.4	FB_SEV_ERROR (FSERR)	65
11.6.5	FB_SEV_FATAL (FSFTL)	65
11.6.6	FB_SEV_NEVER (FSNEV)	65
11.7	Régler la sévérité d'un code d'erreur	65
11.8	Réponses aux erreurs à l'exécution	66
11.8.1	FB_RESP_IGNORE	66
11.8.2	FB_RESP_ABORT	66
11.8.3	FB_RESP_RETRY_ABORT	66
11.8.4	FB_RESP_RETRY_IGNORE	67
11.8.5	FB_RESP_ABORT_ACTION	67
11.8.6	FB_RESP_RETRY_ABORT_ACTION	67
11.8.7	Sévérité des codes d'erreur produits avec la réponse d'erreur associée	67
11.8.8	Fixer et obtenir la réponse à une erreur	67
11.9	Rapport automatique des erreurs	68
11.9.1	Connecter une routine utilisateur de gestion d'erreur	69
11.9.2	Déconnecter une routine utilisateur de gestion d'erreur	69
11.9.3	Détails des routines utilisateur de gestion d'erreur	69
11.10	Accès à l'information de l'état résumé	70
11.10.1	Décoder l'état résumé	70
11.11	Accès à l'information d'état complémentaire	70
11.11.1	Décoder l'information d'état complémentaire	71
11.11.2	Décoder l'état complémentaire dans l'exécution d'une liste	71
11.11.3	Trouver l'information d'état complémentaire	72
11.11.4	Trouver un état complémentaire particulier dans une liste	72
11.12	Routines pour la manipulation des codes d'erreur	73
11.12.1	Contrôler la sévérité d'un code d'erreur	73
11.12.2	Tester le code d'état dans un code d'erreur	73
11.12.3	Tester la sévérité dans un code d'erreur	73
11.12.4	Retourner les codes d'erreur d'une sévérité spécifiée	73
11.13	Rapporter une erreur FASTBUS	74
11.14	Emettre un message d'erreur	74
Section 12.	Codes d'erreur	75
12.1	Ensemble standard de codes d'erreur	75
Annexe A.	Récapitulation des noms des routines et des paramètres	84
A.1	Routines rangées par nom court	86

10.1.2 Check Completion State	59
10.1.3 Connect Completion Routine	60
10.1.4 Disconnect Completion Routine	60
10.2 Reset FASTBUS Port	60
10.3 FASTBUS Port Allocation	61
10.3.1 Allocate FASTBUS Port	61
10.3.2 Deallocate FASTBUS Port	61
10.3.3 Allocate FASTBUS Port with Timeout	61
10.4 Get Version Numbers	61
10.5 Report a Port FASTBUS Error	62
10.6 Generate a Port Error Message	62
Section 11. Status and error handling	63
11.1 Introduction	63
11.2 Error and Return Codes	63
11.3 Summary and Supplementary Status	63
11.4 Restricting Generation of Status Information	64
11.5 Delayed Mode Execution Status	64
11.6 Severity	64
11.6.1 FB_SEV_SUCCESS (FSSUCC)	64
11.6.2 FB_SEV_INFO (FSINFO)	64
11.6.3 FB_SEV_WARNING (FSWARN)	65
11.6.4 FB_SEV_ERROR (FSERR)	65
11.6.5 FB_SEV_FATAL (FSETL)	65
11.6.6 FB_SEV_NEVER (FSNEV)	65
11.7 Set the Severity of an Error Code	65
11.8 Response to Errors in Execution	66
11.8.1 FB_RESP_IGNORE	66
11.8.2 FB_RESP_ABORT	66
11.8.3 FB_RESP_RETRY_ABORT	66
11.8.4 FB_RESP_RETRY_IGNORE	67
11.8.5 FB_RESP_ABORT_ACTION	67
11.8.6 FB_RESP_RETRY_ABORT_ACTION	67
11.8.7 Severity of Error Codes generated with Associated Error Response	67
11.8.8 Specify and Determine Response to an Error	67
11.9 Automatic Error Reporting	68
11.9.1 Connect User Error Handling Routine	69
11.9.2 Disconnect User Error Handling Routine	69
11.9.3 Details of User Error Handling Routine	69
11.10 Access to Summary Status Information	70
11.10.1 Decode Summary Status	70
11.11 Access to Supplementary Status Information	70
11.11.1 Decode Supplementary Status Information	71
11.11.2 Decode Supplementary Status from Execution of a List	71
11.11.3 Find Supplementary Status Information	72
11.11.4 Find Specific Supplementary Status within a List	72
11.12 Routines for the Manipulation of Error Codes	73
11.12.1 Check Severity of an Error Code	73
11.12.2 Test Status Code in an Error Code	73
11.12.3 Test Severity in an Error Code	73
11.12.4 Return Error Code of Specified Severity	73
11.13 Report a FASTBUS Error	74
11.14 Generate an Error Message	74
Section 12. Error Codes	75
12.1 Standard Set of Error Codes	75
Appendix A. Summary of Routine Names and Parameters	84
A.1 Routines sorted by Short Name	86

A.2 Routines rangées par nom long	89
Annexe B. Liste des noms réservés	92
B.1 Liste alphabétique par nom long	92
B.2 Liste alphabétique par nom court	97
Annexe C. Types de données en FORTRAN 77	103
Index	104

Withdrawing
IECNORM.COM: Click to view the full PDF of IEC 61052:1991

A.2 Routines sorted by Long Name	89
Appendix B. List of reserved Names	92
B.1 List sorted by Long Name	92
B.2 List sorted by Short Name	97
Appendix C. FORTRAN 77 data types	103
Index	104

Withd 2021
IECNORM.COM: Click to view the full PDF of IEC 61052:1991

COMMISSION ELECTROTECHNIQUE INTERNATIONALE**CEI 1052 ROUTINES STANDARDS FASTBUS****Routines standards utilisables avec le système
d'acquisition de données FASTBUS****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 vœu 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 Comité d'Etudes n° 45 de la CEI: Instrumentation nucléaire.

Le texte de cette norme est issu des documents suivants:

Règle des Six Mois	Rapport de vote
45(BC)198	45(BC)205

Le rapport de vote indiqué dans le tableau ci-dessus donne toute information sur le vote ayant abouti à l'approbation de cette norme.

La publication suivante de la CEI est citée dans la présente norme:

Publication n° 935 (1990): CEI 935 FASTBUS - Système modulaire d'acquisition rapide de données.

INTERNATIONAL ELECTROTECHNICAL COMMISSION

IEC 1052 FASTBUS STANDARD ROUTINES

Standard Routines for use with FASTBUS data acquisition system

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 IEC Technical Committee No. 45: Nuclear Instrumentation.

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

Six Months' Rule	Report on Voting
45(CO)198	45(CO)205

Full information on the voting for the approval of this standard can be found in the Voting Report indicated in the above table.

The following IEC publication is quoted in this standard:

Publication No. 935 (1990): IEC 935 FASTBUS - Modular high speed data acquisition system.

Section 1. Objet, domaine d'application et introduction

La présente norme définit des routines standards pour l'accès logiciel à un système FASTBUS, système standard modulaire pour le contrôle et l'acquisition rapide de données, défini dans la norme CEI 935. On suppose que le lecteur est familier avec le système FASTBUS.

1.1 Objet et domaine d'application

Les routines standards définies dans cette norme sont applicables aux systèmes FASTBUS conformes à la norme CEI 935. Le but est de fournir des routines standards qui aient la même signification pour tous les utilisateurs et ainsi de minimiser la duplication des efforts logiciels et de maximiser l'interchangeabilité des systèmes, des sous-systèmes et du logiciel FASTBUS.

1.2 Interprétation de la présente norme

Les clauses utilisant le mot "doit" sont obligatoires.

Les clauses utilisant le mot "devrait" indiquent une pratique recommandée ou préférée qui doit être suivie sauf s'il y a de sérieuses raisons de ne pas le faire.

Les clauses utilisant le mot "peuvent" indiquent une pratique permise et laissent la liberté de choix au réalisateur.

Les éléments de cette norme qui sont définis comme "dépendant de la réalisation" doivent être décrits avec précision dans la documentation associée à la réalisation.

Aucune réalisation contrevenant à une clause obligatoire ne peut être considérée comme une réalisation conforme à cette norme.

Aucune licence ou autre permission n'est nécessaire pour utiliser cette norme.

1.3 Vue d'ensemble du document

La section 2 décrit certains concepts fondamentaux communs à toutes les routines.

La section 3 se rapporte aux routines propres à la gestion et à l'utilisation de l'environnement FASTBUS.

La section 4 décrit les paramètres opérationnels qui font partie de l'environnement FASTBUS.

La section 5 décrit les modes de tampon qui sont utilisés pour contrôler l'accès des routines d'action aux tampons de données de l'utilisateur.

La section 6 décrit les routines qui exécutent des transactions simples sur le FASTBUS.

La section 7 décrit les routines qui exécutent des transactions complexes sur le FASTBUS.

La section 8 décrit les routines qui exécutent les primitives des actions FASTBUS.

La section 9 décrit des routines de gestion des événements asynchrones du FASTBUS, de réponse aux demandes de service et de réponse aux messages d'interruption.

Section 1. Object, Scope and Introduction

This standard defines standard routines for software access to a FASTBUS system, a standardized modular high-speed data acquisition and control system defined in IEC 935. It is assumed that the reader is familiar with the FASTBUS system.

1.1 Scope and Object

The standard routines defined in this standard are applicable to FASTBUS systems as defined in the IEC 935. The object is to provide standard routines that have the same meaning to all users so as to minimize duplication of software effort, and maximize interchangeability of FASTBUS systems, subsystems and software.

1.2 Interpretation of this Standard

Clauses using the word "shall" are mandatory.

Clauses using the word "should" indicate recommended or preferred practice to be followed unless there are sound reasons to the contrary.

Clauses using the word "may" indicate permitted practice and leave freedom of choice to the implementer.

Parts of an implementation related to features defined as "implementation dependent" in this standard shall be clearly described in the documentation associated with the Implementation.

No Implementation contravening any mandatory clause is to be referred to as an Implementation of this standard.

No licence or other permission is needed in order to use this standard.

1.3 Document Overview

Section 2 describes some basic concepts common to all routines.

Section 3 deals with routines involved in the management and use of the FASTBUS environment.

Section 4 describes the Operational Parameters which form part of the FASTBUS environment.

Section 5 describes the buffer modes which are used to control access to user data buffers by action routines.

Section 6 describes routines for carrying out simple transactions on FASTBUS.

Section 7 describes routines for carrying out compound transactions on FASTBUS.

Section 8 describes routines for carrying out primitive FASTBUS actions.

Section 9 describes routines for handling asynchronous events on FASTBUS, responding to service requests and responding to interrupt messages.

La section 10 décrit des routines spécifiques soit au système soit au port.

La section 11 décrit les réponses de la réalisation aux erreurs. On décrit des routines pour contrôler celles-ci et pour extraire les informations d'état concernant les actions précédentes.

La section 12 contient la liste des codes d'erreur et leur sévérité.

Withdrawing
IECNORM.COM: Click to view the full PDF of IEC 61052:1991

Section 10 describes routines that are either system or port dependent.

Section 11 describes the response of the implementation to errors. Routines are described for controlling this and for extracting status information about previous actions.

Section 12 contains a list of all the error codes along with their severity.

Withd
IECNORM.COM: Click to view the full PDF of IEC 61052:1991

Section 2. Concepts de base

2.1 Terminologie

La présente norme utilise différents termes qui possèdent une signification particulière en FASTBUS. Des termes tels que "pipe-line", "diffusion", "segment", etc., sont définis dans la norme FASTBUS. Ces définitions ne seront pas répétées ici.

Les routines de cette norme sont décrites comme une communication entre "l'utilisateur" qui les appelle et la "réalisation" qui est appelée. La portée d'une session FASTBUS définit "l'utilisateur" ou le "processus utilisateur". Certains termes généraux sont définis ci-dessous:

Les "actions FASTBUS" lisent ou manipulent directement les lignes FASTBUS sur un segment. Les réalisations FASTBUS ne supportent pas forcément toutes les actions de ce type qui sont, en principe, possibles.

Une "primitive d'action FASTBUS" est l'une des cinq activités logiques élémentaires possibles:

- Cycle d'arbitrage.
- Cycle d'adressage primaire.
- Cycle de données d'adresse secondaire.
- Cycle de données (incluant l'adresse secondaire).
- Libération du dispositif (séquence de fin).
- Libération du bus.

Toutes les réalisations FASTBUS ne supportent pas forcément l'exécution de chaque activité élémentaire isolée, mais, pour pouvoir être qualifiées conformes à cette norme, toutes les réalisations doivent supporter les opérations et les transactions telles qu'elles sont définies ci-dessous.

Une "opération" est un cycle d'adressage primaire suivi par aucun, un ou plusieurs cycles de données et par une séquence de fin.

Une "opération simple" est une opération où seul le premier cycle de données après le cycle d'adressage primaire peut spécifier une adresse secondaire.

Une "transaction" est constituée par la prise de la maîtrise du bus suivie par l'exécution d'une ou de plusieurs opérations suivies par la libération du bus.

Une "transaction simple" est une transaction contenant l'exécution d'une seule opération simple.

Le "contrôleur" est l'origine des appels aux actions FASTBUS décrites dans cette norme.

Un "port" FASTBUS couple un contrôleur avec un segment FASTBUS.

Il peut exister de multiples chemins entre le contrôleur et un segment FASTBUS. On doit utiliser un port_id pour distinguer ces différents chemins.

La "norme FASTBUS" qui est utilisée ici est la norme CEI 935: FASTBUS, système modulaire d'acquisition rapide de données.

Section 2. Basic Concepts

2.1 Terminology

This standard uses several terms which have specific meaning for FASTBUS. Such terms as "pipelined", "broadcast", "Segment", etc., are defined in the FASTBUS standard. Their definitions are not repeated here.

The routines in this standard are described as a communication between the "User", who calls them, and the "Implementation", which is called. The scope of a FASTBUS session defines the "User" or "User Process". Some general terms are defined below:

"FASTBUS Actions" directly read or manipulate FASTBUS lines on a Segment. Not all implementations to FASTBUS support all of the actions of this type that are, in principle, possible.

A "Primitive FASTBUS Action" is one of the following five possible logically complete subactivities:

- Arbitration Cycle.
- Primary Address Cycle.
- Secondary Address Data Cycle.
- Data Cycle.
- Device Release (Termination Sequence).
- Bus Release.

Not all implementations to FASTBUS support the execution of each subactivity singly but all implementations shall support Operations and Transactions as defined below in order to qualify as an Implementation of this standard.

An "Operation" is a primary address cycle followed by zero or more Data cycles and a Termination Sequence.

A "Simple Operation" is an Operation in which only the first Data cycle after the primary address cycle may specify a secondary address.

A "Transaction" is the gaining of bus mastership followed by the execution of one or more Operations followed by release of the bus.

A "Simple Transaction" is a Transaction involving the execution of a single Simple Operation.

The "Controller" is the originator of the FASTBUS Action calls described in this standard.

A FASTBUS "Port" couples a Controller to a FASTBUS Segment.

Multiple paths may exist between the Controller and a FASTBUS Segment. A port_id shall be used to distinguish between different paths.

The "FASTBUS standard" as used herein is IEC 935: FASTBUS, Modular High-Speed Data Acquisition System.

2.2 Environnement

2.2.1 Vue d'ensemble

La norme FASTBUS définit un protocole où un maître gagne tout d'abord le contrôle du bus, puis établit une connexion avec un ou plusieurs dispositifs esclaves et alors exécute des transferts de données. Le maître peut rompre la connexion avec les esclaves sans relâcher le contrôle du segment et il peut alors établir une connexion avec un esclave différent. Le maître peut, n'importe quand, libérer le bus pour qu'un autre maître puisse gagner le contrôle du bus dès que toutes les connexions d'adresses ont été relâchées.

Toutes ces actions prennent place dans le contexte d'un environnement sur lequel l'utilisateur a certains moyens de contrôle. Le niveau de contrôle qui est disponible pour l'utilisateur variera d'une réalisation à l'autre. Le contrôleur peut ou non modifier les paramètres affectant les actions FASTBUS, et le système logiciel du processeur permet ou non d'utiliser certaines opérations ou techniques. Pour autoriser le large éventail de ressources matérielles et logicielles qui peuvent être disponibles dans différentes applications, cette norme distingue les actions qui prennent place sur le segment FASTBUS lui-même et l'environnement dans lequel elles se produisent. Des routines sont définies pour contrôler d'une part la nature de l'environnement et, d'autre part, l'activité qui doit prendre place sur le segment FASTBUS.

L'activité du FASTBUS est déterminée par des instructions du contrôleur. Le contrôleur peut être un gros système de calcul avec de nombreux périphériques standards, comme des disques ou des bandes magnétiques (en plus du FASTBUS), et avec un système d'exploitation multi-utilisateur et multi-tâche qui fournit la possibilité d'une large gamme d'activités simultanées telles que le développement de programmes et l'acquisition de données. A l'autre extrémité du spectre, il peut être un processeur inclus dans un dispositif FASTBUS ne possédant peut-être même pas un système d'exploitation rudimentaire, mono-tâche, mono-utilisateur. Un tel processeur peut utiliser des programmes qui ont été préparés ailleurs et qui peuvent être stockés dans une mémoire morte.

Certaines des instructions du contrôleur produiront des cycles de divers types sur le FASTBUS lui-même, tandis que d'autres contrôleront l'environnement dans lequel les cycles FASTBUS se dérouleront. Cet environnement comprend, non exhaustivement, le niveau d'arbitrage utilisé par le maître lorsqu'il demande la maîtrise du bus, différents retards de temporisation, l'information fixant le maintien ou non de la connexion d'adresse à la fin du cycle de données courant, celle fixant la libération ou non du bus à la fin de l'opération courante, etc. Cette norme offre la possibilité de spécifier ces valeurs bien qu'un port FASTBUS donné n'ait pas forcément besoin d'être capable de les modifier toutes.

Un autre aspect de l'environnement définit comment le système se comporte après avoir exécuté un appel à une routine FASTBUS. On offre la possibilité de réponse automatique aux conditions d'erreurs et de rapport automatique des erreurs. L'utilisateur a le contrôle, selon des conditions définies, des réponses du matériel et du logiciel et fixe ce qui doit ou non être rapporté.

L'environnement définit également l'origine des instructions vers le port FASTBUS. Ces instructions peuvent venir, soit immédiatement des routines lorsque le programme les exécute, soit provenir de listes d'instructions construites précédemment. L'environnement détermine si, à l'appel d'une routine FASTBUS, une liste doit être construite ou non.

2.2 Environment

2.2.1 Overview

The FASTBUS standard defines a protocol whereby a Master first gains control of the bus, establishes a connection to one or more Slave devices and then executes data transfers. The Master may break the connection to the Slaves without releasing control of the segment and Master may establish a connection to a different Slave. The Master may, at any time, release the bus so that another Master may gain control of the bus once any address connection is released.

All these actions take place within the context of an Environment over which the User has some measure of control. Just what degree of control is available to the User will vary from one Implementation to another. The Controller may or may not be able to vary parameters affecting FASTBUS Actions and the processor's software system may or may not permit certain operations or techniques to be used. In order to allow for the wide range of hardware and software resources that may be available in different applications, this standard distinguishes between actions that take place on the FASTBUS Segment itself and the Environment in which they occur. Routines are defined to control both the nature of the Environment and the activity which is to take place on the FASTBUS Segment.

The activity on FASTBUS is determined by instructions from the Controller. The Controller may be a large scale computing system with many standard peripherals such as disks and magnetic tape units (in addition to FASTBUS) and with a multi-user, multi-tasking operating system which provides facilities for a wide range of simultaneous activities such as program development and data acquisition. At the other end of the spectrum may be the processor embedded in a FASTBUS device having, if at all, only a rudimentary single-user, single-task operating system. Such a processor may run programs which have been prepared elsewhere and which may be stored in read-only memory.

Some of the instructions from the Controller will result in cycles of various types taking place on the FASTBUS itself while others will control the Environment in which the FASTBUS cycles take place. This Environment includes, but is not limited to, the arbitration level used by the Master when vying for bus mastership, various time-out delays, whether or not the address connection is to be maintained at the end of the current data cycles, whether or not the bus is to be released at the end of the current Operation, etc. Provision is made in this standard for specifying these values, although a given FASTBUS Port need not be able to implement changes to them all.

Another aspect of the Environment specifies how the system behaves after completing a call to a FASTBUS routine. Provision is made for automatic responses to error conditions and automatic reporting of errors. The User has control over hardware and software responses to defined conditions as well as what does and does not get reported.

The Environment also specifies the source of the instructions to the FASTBUS Port. These instructions may come immediately from the routines as they are encountered by the program or they may come from Lists of instructions that have been previously constructed. Whether or not a List is constructed at FASTBUS routine invocation time is determined by the Environment.

2.2.2 Définition de l'environnement

"L'environnement" est l'information que la réalisation conserve pour le compte de l'utilisateur. Cette information se compose de cinq parties:

1. Un ensemble de "paramètres opérationnels", qui prennent des valeurs par défaut et peuvent être fixés explicitement par l'utilisateur (voir section 4).
2. Les informations contrôlant les réponses de la réalisation aux erreurs et autres conditions non prévues (voir sections 4 et 11).
3. Les informations relatives aux tampons séquentiels (voir section 5).
4. Les informations d'état produites par la réalisation décrivant les actions passées et les erreurs (voir section 11).
5. Une "liste" des actions FASTBUS. Celle-ci est utilisée seulement dans le mode d'exécution différée, comme un enregistrement d'actions à exécuter plus tard. Cette liste n'est pas présente dans un environnement qui ne supporte pas le mode d'exécution différée (voir section 3).

2.2.3 Identificateur d'environnement

L'environnement est référencé en utilisant le paramètre "environment_id" (identificateur) qui est transmis à la plupart des routines FASTBUS. Une réalisation doit fournir un environnement par défaut ayant l'environnement_id de FB_DEFAULT EID. D'autres environnements peuvent être créés en appelant une des routines de l'article 3.2. Un utilisateur peut travailler simultanément avec différents environnements.

2.2.4 Modes d'exécution

Un environnement peut être dans l'un des deux modes d'exécution suivants, le "mode d'exécution immédiate" ou le "mode d'exécution différée". Le mode d'exécution est représenté par l'état du paramètre opérationnel FB_PAR_DELAY_EXEC qui peut être respectivement FB_FALSE ou FB_TRUE. Le mode d'exécution initial dépend de la manière dont l'environnement est créé. Un environnement créé "immédiat" doit toujours rester en mode d'exécution immédiate. Un environnement créé "différé" peut ultérieurement être changé en mode d'exécution immédiate et vice versa en modifiant FB_PAR_DELAY_EXEC.

Une réalisation doit obligatoirement supporter le mode d'exécution immédiate, le mode d'exécution différée est optionnel.

Le mode d'exécution affecte ce qui arrive lorsqu'une routine d'action est appelée.

Les actions provoquées par les routines d'action où l'identificateur de l'environnement n'est pas passé comme paramètre doivent toujours être exécutées immédiatement.

2.2.5 Accès au FASTBUS dans un système multi-utilisateur

Le verrouillage des adresses et du bus FASTBUS autorise l'usage correct du bus par des utilisateurs multiples. La réalisation du logiciel doit gérer l'exclusion mutuelle de plusieurs utilisateurs accédant au FASTBUS à travers le même port. La réalisation doit éviter que d'autres utilisateurs accèdent au bus par un port où un utilisateur possède un verrouillage de bus (GK positionné) ou un verrouillage d'adresse (verrouillage AS-AK).

L'attribution explicite d'un port est une opération de haut niveau qui fournit à l'utilisateur la garantie d'usage exclusif du port FASTBUS pour exécuter une série d'opérations sans conflit sur le port. On notera que la seule attribution du port ne garantit pas l'usage exclusif du bus (voir article 10.3).

2.2.2 Definition of the Environment

The "Environment" is that information which the Implementation keeps on behalf of the User. This information consists of five parts:

1. A set of "Operational Parameters", which take default values and may be set explicitly by the User (see section 4).
2. Information controlling the response of the Implementation to errors and other unsolicited conditions (see sections 4 and 11).
3. Information related to Sequential Buffers (see section 5).
4. Status information generated by the Implementation describing past actions and errors (see section 11).
5. A "List" of FASTBUS Actions. This is used in Delayed Execution Mode only, as a record of actions to be executed later. The List is not present for an Environment which does not support Delayed Execution Mode (see section 3).

2.2.3 Environment Identifier

The Environment is referred to by using an environment id (identifier) parameter that is transmitted to most FASTBUS routines. An Implementation shall provide a default Environment having the environment id of FB_DEFAULT_EID. Other Environments may be created by calling one of the routines in clause 3.2. A User may work with several Environments concurrently.

2.2.4 Execution Modes

An Environment may be in either of two execution modes, "Immediate Execution Mode" or "Delayed Execution Mode". The execution mode is represented by the state of the Operational Parameter FB_PAR_DELAY_EXEC being FB_FALSE or FB_TRUE, respectively. The initial execution mode depends on how the Environment is created. An Environment created "immediate" shall always remain in Immediate Execution mode. An Environment created "delayed" may subsequently be changed to and from Immediate Execution Mode by altering FB_PAR_DELAY_EXEC.

It is mandatory for an Implementation to support Immediate Execution Mode, whereas support for Delayed Execution Mode is optional.

The execution mode affects what happens when an Action Routine is called.

Actions caused by Action Routines where the Environment identifier is not passed as a parameter shall always be executed immediately.

2.2.5 Access to FASTBUS in a Multi-user System

The FASTBUS bus and address locks provide for proper use of the bus by multiple masters. The software Implementation shall handle the mutual exclusion of multiple users accessing FASTBUS via the same Port. The Implementation shall prevent other users from accessing the bus through a Port through which another User has a bus lock (GK asserted) or address lock (AS-AK lock).

Explicit port allocation is a high level operation which provides for the user to guarantee exclusive use of the FASTBUS Port in order to perform a series of operations without contention on the Port. It should be noted that Port allocation alone does not guarantee exclusive use of the bus itself (see clause 10.3).

2.2.6 Session FASTBUS

Un appel à la routine FB_OPEN, décrite à la section 3, définit une session FASTBUS. Pendant cette session, l'utilisateur doit disposer de l'environnement par défaut (spécifié en utilisant l'environnement id FB_DEFAULT_EID). Toute allocation de port ou de dispositif (voir section 10) agit sur l'ensemble de la session FASTBUS associée.

2.2.7 Routines d'action

Dans les descriptions suivantes, certaines routines sont indiquées comme des "routines d'action". Ce sont les routines qui ont une "action" associée affectant le FASTBUS ou les "tampons de données" ou les deux. Toutes les routines dont les noms incluent READ ou WRITE (par opposition à GET ou SET, voir article 4.1) sont des routines d'action.

En mode d'exécution immédiate, lorsqu'une routine d'action est appelée, l'action est immédiatement exécutée. Les informations d'état concernant le succès ou l'échec de l'action sont mémorisées dans l'environnement (historique d'état, voir section 11). L'action exacte à exécuter dépend de la routine appelée, mais aussi de l'état des paramètres opérationnels. Le transfert de données par une routine d'action se fait de ou dans un tampon utilisateur suivant le "mode de tampon" défini (voir section 5).

En mode d'exécution différée, lorsqu'une routine d'action est appelée, l'action n'est pas exécutée immédiatement. L'action associée à la routine, conditionnée par l'état des paramètres opérationnels à cet instant, est ajoutée à la "liste" des actions et pourra être exécutée plus tard. Les codes de retour de la routine ne donnent qu'une indication sur le succès de l'addition de l'action à la liste, mais à ce niveau on ne produit aucun historique d'état, tout historique d'état antérieur est conservé intact. A ce niveau, aucun transfert de données ne se fait de ou dans les tampons de données (sauf pour WRITE d'un mot unique de données dans le mode FB_BUFFER_VALUE - voir section 5).

La routine FB_LIST_EXECUTE provoque l'exécution, dans l'ordre, de toutes les actions mémorisées dans une liste. Lorsque cette routine est appelée, un état historique est produit et les transferts de données se font de ou dans des tampons de données utilisateurs suivant chaque action.

2.2.8 Moment d'appel et moment d'exécution

Lorsque l'on décrit les effets des routines d'action, ces deux termes sont utilisés, pour que la même définition puisse s'appliquer aux modes d'exécution immédiate ou différée. "Moment d'appel" se réfère au moment où la routine d'action est appelée, et "moment d'exécution" se réfère au moment où les actions associées se produisent. En mode d'exécution immédiate, le moment d'exécution est de fait le moment de l'appel, tandis qu'en mode d'exécution différée, le moment d'exécution est le moment où FB_LIST_EXECUTE est appelé.

2.3 Conventions de dénomination

A chaque entité est assigné un nom descriptif long (jusqu'à 32 caractères) et un nom court (jusqu'à 6 caractères). Tous les noms longs commencent par FB_ et tous les noms courts par la lettre F. Tous les deux sont en majuscules dans la présente norme.

Une réalisation doit reconnaître les noms courts et devrait aussi reconnaître les noms longs.

2.4 Paramètres des routines: types et direction

Le nom, le type de données et la direction des paramètres sont mis en tableau pour chaque routine comme suit:

2.2.6 FASTBUS Session

A call to the routine FB_OPEN, described in section 3, defines a FASTBUS Session. Within this session the User shall have available a single Default Environment (specified by using the Environment_id FB_DEFAULT_EID). Any Port or device allocations (see section 10) act over the whole associated FASTBUS session.

2.2.7 Action Routines

In the descriptions of routines which follow, some routines are labelled as being "Action Routines". These are routines which have an associated "action" affecting FASTBUS or "Data Buffers" or both. All routines whose names include READ or WRITE (instead of GET or SET, see clause 4.1) are Action Routines.

In Immediate Execution Mode, when an Action Routine is called, the action is performed immediately. Status information about the success or failure of the action is stored in the Environment (Status History, see section 11). The exact action to be performed depends on the routine called, and also on the state of the Operational Parameters. Data transfer by Action Routines takes place to or from any User buffers according to a defined "Buffer Mode" (see section 5).

In Delayed Execution Mode, when an Action Routine is called, the action does not take place immediately. Instead, the action associated with the routine, as conditioned by the state of the Operational Parameters at the time, is added to the "List" of actions and can be executed at any later time. The Return Code of the routine only gives an indication of the success of adding the action to the List, but at this stage, no status history is generated, any previous Status History being left intact. At this stage, no data transfer takes place into or out of the Data Buffers (except for single word WRITE data in FB_BUFFER_VALUE mode - see section 5).

The FB_LIST_EXECUTE routine causes all the actions stored in a List to be performed in order. When this routine is called, a Status History is generated, and data transfers take place into and out of the User Data Buffers as appropriate for each action.

2.2.8 Call Time and Execution Time

In discussing the effects of Action Routines, two terms are used so that the same definition may apply to Immediate and Delayed Execution Modes. "Call Time" refers to the time when an Action Routine is called and "Execution Time" refers to the time when the associated action takes place. In Immediate Execution Mode, Execution Time is therefore the same as Call Time, whereas in Delayed Execution Mode, Execution Time is the time when FB_LIST_EXECUTE is called.

2.3 Naming Conventions

Each entity is assigned a descriptive long (up to 32 characters) name as well as a short (up to six characters) name. All long names start with FB_ and all short ones with the letter F. Both are in upper case in this standard.

An Implementation shall recognize the short names and should also recognize the long names.

2.4 Parameters to Routines: Type and Direction

The name, data type and direction of routine parameters are tabulated for each routine as follows:

Paramètres:

< nom >	< type >	< direction >
< nom >	< type >	< direction >
...		

Le champ < nom > n'est utilisé que pour référencer ce paramètre à l'intérieur de la description de la routine courante. Il est toujours en minuscules dans cette norme.

Le champ < type > indique l'utilisation et, pour chaque réalisation, la mémoire nécessaire pour le paramètre. Il commence toujours par FB_ suivi de minuscules.

De nombreux paramètres sont mémorisés dans des mots de 32 bits. Cependant, dans de nombreux cas, les valeurs que peuvent prendre les arguments sont limitées. Certaines valeurs, comme `environment_id`, sont attribuées par les routines et ne peuvent être modifiées ou manipulées par l'utilisateur. D'autres types ne peuvent avoir que les valeurs symboliques définies dans cette norme. La représentation de ces valeurs en mémoire dépend de la réalisation.

Les champs < type > avec leurs utilisations sont les suivants:

FB_boolean	Prend les valeurs FB_FALSE ou FB_TRUE.
FB_buffer_mode	Définit la signification du paramètre correspondant <code>buffer_specifier</code> (voir section 5).
FB_environment_id	La valeur FB_DEFAULT_EID peut toujours être utilisée. Les seules autres valeurs qui peuvent être utilisées sont discutées à l'article 3.2.
FB_error_code	Un code d'erreur (voir section 11). Contient les informations d'état et la sévérité.
FB_integer	Prend toute valeur de -2^{31} à $2^{31}-1$.
FB_line	Identifie un signal ou un groupe de signaux sur le FASTBUS. Les valeurs possibles sont données à l'article 8.4.
FB_list_pointer	Une position dans une liste d'actions. La représentation dépend de la réalisation.
FB_operator	Identifie l'opération logique ou arithmétique qui doit se produire pendant une lecture-modification-écriture FASTBUS (voir section 7).
FB_parameter_id	Identifie un paramètre opérationnel particulier. Peut prendre toutes les valeurs décrites à la section 4 plus des valeurs dépendant de la réalisation.
FB_parameter_value	La valeur d'un paramètre opérationnel. La gamme des valeurs autorisées dépend du paramètre et est donnée à la section 4.
FB_port_id	Identifie une connexion particulière entre un contrôleur et un segment FASTBUS.
FB_response_code	Décrit une action à prendre résultant des conditions indiquées par le matériel FASTBUS. Voir article 11.8.
FB_routine	Un paramètre utilisé pour passer une routine utilisateur comme argument à une routine FASTBUS.
FB_seq_buffer_id	Un identificateur pour un tampon séquentiel, décrit à la section 5.
FB_severity	Décrit la sévérité d'une condition d'erreur. Les valeurs possibles sont données à l'article 11.6.
FB_word	Un mot de 32 bits d'une signification non définie.

Parameters:

< name >	< type >	< direction >
< name >	< type >	< direction >
...		

The < name > field is used only to refer to the parameter within the description of the routine that follows. It is always in lower case in this standard.

The < type > field indicates the usage and, for each Implementation, the storage requirement for the parameter. It always starts with FB_ followed by lower case letters.

Many parameters are stored as 32-bit words. However, in many cases, the values the arguments can take are restricted. Some values, like environment_ids, are assigned by the routines and should not be altered or manipulated by the user. Other types shall only have symbolic values as defined in this standard. The conversion of these names to stored values is Implementation dependent.

The < type > fields, with their usage, are as follows:

FB_boolean	Takes values FB_FALSE or FB_TRUE.
FB_buffer_mode	Defines the significance of a related buffer_specifier parameter (see section 5).
FB_environment_id	The value FB_DEFAULT_EID may always be used. The only other values which may be used are discussed in clause 3.2.
FB_error_code	An Error Code (see section 11). Contains status information and a Severity.
FB_integer	Takes any value from -2^{31} to $2^{31}-1$.
FB_line	Identifies a signal or group of signals on FASTBUS. Takes any value given in clause 8.4.
FB_list_pointer	A position within a List of actions. Representation is implementation dependent.
FB_operator	Identifies the arithmetic or logical operation that is to take place during a FASTBUS Read-Modify-Write (see section 7).
FB_parameter_id	Identifies a particular Operational Parameter. Takes any of the values listed in section 4 or an implementation dependent value.
FB_parameter_value	The value of an Operational Parameter. The range of values allowed depends on the parameter and is given in section 4.
FB_port_id	Identifies a particular connection between a Controller and a FASTBUS Segment.
FB_response_code	Describes an action to be taken as a result of specified conditions in the FASTBUS hardware. See clause 11.8.
FB_routine	A parameter used to pass a User routine as an argument to a FASTBUS routine.
FB_seq_buffer_id	A Sequential Buffer identifier, described in section 5.
FB_severity	Describes the Severity of an error condition. Takes any value listed in clause 11.6.
FB_word	A 32-bit word of undefined significance.

Les types peuvent être étendus par le préfixe "array_of_", qui indique un tableau de dimension non spécifiée. La dimension du tableau en octets est toujours donnée par un paramètre séparé.

La représentation de ces types dans un environnement FORTRAN 77 est donnée dans l'annexe C.

Le champ <direction> donne la direction du flux d'information et doit être l'un des suivants:

IN La routine lit mais ne modifie pas l'argument.

OUT La routine retourne la valeur à l'argument ou à l'entité spécifiée.

A titre d'exemple, la routine FRD ou FB_READ_DAT possède les paramètres suivants:

return_code	FB_error_code	(OUT)	
environment_id	FB_environment_id	(IN)	
primary_address	FB_word	(IN)	
secondary_address	FB_word	(IN)	
buffer_mode	FB_buffer_mode	(IN)	
buffer_specifier	FB_word	(OUT)	(FBVAR)
	ou FB_seq_buffer_id	(IN)	(FBSEQ)

Cela indique que le premier paramètre est un code de retour, renvoyé par la routine, le second paramètre est un identificateur de contrôle passé à la routine et non renvoyé. Les troisième et quatrième paramètres sont des mots de 32 bits passés à la routine. Les paramètres buffer_mode et buffer_specifier définissent les sources de données ou leurs destinations dans le contrôleur et sont discutés ci-dessous. Le buffer_specifier peut prendre l'une des formes de la liste.

2.5 Paramètres des tampons

Chaque routine d'action possède parmi ses paramètres deux paramètres de tampon:

buffer_mode	FB_buffer_mode	(IN)
buffer_specifier	< type >	< direction >

La valeur du paramètre buffer_mode détermine la signification du second paramètre, son type et quelquefois sa direction.

Les trois modes de tampon possibles, pour les transferts des données, permettent d'utiliser une variable, ou une valeur spécifiée (littérale), ou un tampon séquentiel numéroté. Ces modes sont définis dans le tableau suivant:

Valeur buffer_mode		Buffer_specifier	
(nom long)	(court)	< type >	< direction >
FB_BUFFER_VAR	FBVAR	(array_of_)FB_word	IN ou OUT
FB_BUFFER_VALUE	FBVAL	FB_word	IN
FB_BUFFER_SEQ	FBSEQ	FB_seq_buffer_id	IN

Les modes de tampon et leurs utilisations dans un environnement de mode d'exécution immédiate ou différée sont discutés en détail à la section 5.

L'annexe C donne la représentation représentée suggérée des types de données FASTBUS dans un environnement FORTRAN 77.

The type may be extended by the prefix "array_of_", which indicates an array of unspecified size. The size of the array in bytes is always given as a separate parameter.

The representation of these types in a FORTRAN 77 Environment is given in appendix C.

The <direction> field gives the direction of information flow and shall be one of the following:

IN The routine reads, but does not modify, the argument.
OUT The routine returns the value to the argument or to the entity specified.

As an example, the routine FRD or FB_READ_DAT has the following parameters:

return_code	FB_error_code	(OUT)	
environment_id	FB_environment_id	(IN)	
primary_address	FB_word	(IN)	
secondary_address	FB_word	(IN)	
buffer_mode	FB_buffer_mode	(IN)	
buffer_specifier	FB_word	(OUT)	(FBVAR)
	or FB_seq_buffer_id	(IN)	(FBSEQ)

This indicates that the first parameter is a Return Code, returned by the routine, the second parameter is a control identifier passed to the routine and not returned. The third and fourth parameters are 32 bit words passed to the routine. The buffer_mode and buffer_specifier parameters define the data source or destination in the Controller and are discussed below. The buffer_specifier may take one form OR the other(s) listed for each routine.

2.5 Buffer Parameters

Any Action Routine has among its parameters two buffer parameters:

buffer_mode	FB_buffer_mode	(IN)
buffer_specifier	< type >	< direction >

Here, the value of the Buffer Mode parameter determines the significance of the second parameter, its type and possibly its direction.

The three possible Buffer Modes allow a variable, or a specified value (literal), or a numbered sequential buffer to be used for data transfer. The modes are tabulated below:

Buffer_mode Value		Buffer_specifier	
(long name)	(short)	< type >	< direction >
FB_BUFFER_VAR	FBVAR	(array_of_)FB_word	IN or OUT
FB_BUFFER_VALUE	FBVAL	FB_word	IN
FB_BUFFER_SEQ	FBSEQ	FB_seq_buffer_id	IN

The Buffer Modes and their use with Delayed and Immediate Execution Mode Environments are discussed in more detail in section 5.

Appendix C gives the suggested representation of the FASTBUS data types in FORTRAN 77 Environment.

2.6 Catégories des routines

Chaque routine de cette norme est affectée à une des catégories suivantes:

Les routines de la catégorie A sont obligatoires pour toutes les réalisations.

Les routines de la catégorie B sont obligatoires pour toutes les réalisations qui supportent l'environnement du mode d'exécution différée.

Les routines de la catégorie C doivent être réalisées par ceux des contrôleurs FASTBUS sur lesquels on peut ordonner l'exécution de cycles simples FASTBUS ou accéder individuellement aux lignes des signaux FASTBUS.

Les routines appropriées dans la catégorie I doivent être réalisées lorsqu'un contrôleur peut se comporter en esclave et recevoir des messages d'interruption FASTBUS ou peut exécuter des actions initialisées par le positionnement du signal SR.

Les routines de la catégorie S dépendent du système et de la réalisation. Leur réalisation n'est pas obligatoire et leur fonctionnement ne doit pas obligatoirement être le même dans toutes les réalisations.

Les routines de la catégorie E sont optionnelles pour toutes les réalisations.

2.6 Categories of Routines

Each routine in this standard is assigned to one of several categories.

Those routines in Category A are mandatory for any Implementation.

Those routines in Category B are mandatory for any Implementation that supports a Delayed Execution Mode Environment.

The routines in Category C shall be implemented by those FASTBUS Controllers that can be instructed to perform Single FASTBUS cycles or access individual FASTBUS signal lines.

The appropriate routines in Category I shall be implemented when a Controller is to act as a Slave and receive FASTBUS interrupt messages or can perform actions initiated by assertion of the SR signal.

The routines in Category S are system and Implementation dependent. Their Implementation is not mandatory and their Operation might not be the same for all Implementations.

The routines in Category E are optional for any Implementation.

Section 3. Gestion de l'environnement et exécution différée

Une réalisation particulière des routines FASTBUS peut dépendre de l'architecture du contrôleur et des ports. La principale différence entre les réalisations résulte du fait que le mode d'exécution différée est supporté ou non; cette différence se retrouve dans l'environnement (voir section 2).

Toutes les réalisations doivent fournir un environnement par défaut ayant pour `environment_id` la valeur `FB_DEFAULT_EID`. Cet environnement doit être en mode d'exécution immédiate uniquement.

Dans les réalisations qui ne supportent pas le mode d'exécution différée, l'environnement par défaut ainsi que tous ceux créés par l'utilisateur doivent contenir les paramètres opérationnels, les réponses d'erreur et les informations d'état et, si nécessaire, les informations sur les tampons séquentiels. Dans les réalisations qui supportent le mode d'exécution différée, tout environnement créé par l'utilisateur en mode d'exécution différée doit être capable de contenir en plus une liste des actions à exécuter.

3.1 Ouvrir et fermer une session FASTBUS

Les deux routines, décrites ci-dessous, doivent être appelées par tout programme au début et à la fin d'une session FASTBUS.

3.1.1 Ouverture d'une session FASTBUS

La routine suivante doit être appelée par l'utilisateur avant toute autre routine de cette norme. Des paramètres dépendant de la réalisation peuvent être utilisés pour spécifier les détails sur la manière d'utiliser le FASTBUS.

Aucun paramètre opérationnel ne doit affecter cette routine.

Nom: FBOPEN ou FB_OPEN

Catégorie: A

Paramètres:

<code>return_code</code>	<code>FB_error_code</code> (OUT)
autres paramètres	Dépendent de la réalisation

Effets:

Cet appel peut être utilisé pour effectuer l'initialisation logicielle et matérielle du contrôleur, mais ne doit exécuter aucune action FASTBUS sur le bus.

Sans affecter le segment FASTBUS ni exécuter aucune opération FASTBUS, tous les paramètres opérationnels de l'environnement par défaut doivent prendre leurs valeurs par défaut.

Si `FB_OPEN` a été appelé plus récemment que `FB_CLOSE`, un `FB_CLOSE` (voir ci-après) est implicitement effectué.

Section 3. Environment Management and Delayed Execution

A particular Implementation of the FASTBUS Routines may depend on the architecture of the Controller and Port. The major difference between Implementations is whether Delayed Execution Mode is supported or not; this difference is reflected in the Environment (see section 2).

All Implementations shall provide a default Environment associated with the environment_id value of FB_DEFAULT_EID. This Environment shall only be an Immediate Execution Mode Environment.

In Implementations which do not support Delayed Execution Mode, the default Environment as well as any Environment created by the User shall contain the Operational Parameters, error response and status information, and Sequential Buffer information if relevant. In Implementations which support Delayed Execution Mode, any Environments created by the User with Delayed Execution Mode shall be capable of additionally containing a List of actions to be performed.

3.1 Open and Close FASTBUS Session

Two routines are described which shall be called by any program before and at the end of a FASTBUS session.

3.1.1 Open FASTBUS Session

The following routine shall be called by the User prior to any other routine in this standard. Implementation dependent parameters may be used to specify details of how FASTBUS is to be used.

No Operational Parameters shall affect this routine.

Name: FBOPEN or FB_OPEN

Category: A

Parameters:

return_code	FB_error_code (OUT)
further parameters	Implementation dependent

Effect:

This call may be used to perform software and hardware Controller initialization, but shall not perform any FASTBUS Actions on the bus.

Without affecting the FASTBUS Segment or performing any FASTBUS Operation, all Operational Parameters in the default Environment shall take their default states.

If FB_OPEN has been called more recently than FB_CLOSE, an FB_CLOSE (see below) shall be implicitly performed.

3.1.2 Clôture d'une session FASTBUS

Lorsqu'il n'a plus besoin d'utiliser le FASTBUS, l'utilisateur doit appeler la routine suivante:

Nom: FBCLOS ou FB_CLOSE

Catégorie: A

Paramètres:

return_code FB_error_code (OUT)

Effets:

Si l'utilisateur a laissé le port FASTBUS dans un état où il positionne certaines lignes, ces lignes sont libérées. Toutes les valeurs d'environnement_id allouées sont implicitement désallouées. Aucun paramètre opérationnel ne doit affecter cette routine. Si, après un appel à la routine FB_CLOSE, l'utilisateur appelle une routine de cette norme avant d'appeler à nouveau FB_OPEN, une erreur doit être émise.

Aucun historique d'état n'est produit.

3.2 Routines de gestion de l'environnement

3.2.1 Création d'un environnement en mode d'exécution immédiate

Nom: FCIENV ou FB_CREATE_IMMEDIATE_ENVIRONMENT

Catégorie: A

Paramètres:

return_code FB_error_code (OUT)
environment_id FB_environment_id (OUT) (voir texte)

Effets:

On doit créer ou assigner un environnement dans l'état défini par défaut par la réalisation, qui soit dans le mode d'exécution immédiate (FB_PAR_DELAY_EXEC est FB_FALSE). Pour un environnement créé de cette manière, le paramètre opérationnel FB_PAR_DELAY_EXEC doit être accessible en lecture seule. La valeur d'environnement_id retournée doit être utilisée comme un paramètre d'entrée pour spécifier l'environnement aux autres routines.

Pour les réalisations qui allouent la mémoire, cette routine doit rendre disponible l'espace nécessaire à l'environnement et doit retourner un environnement_id.

(Dans les réalisations où l'utilisateur doit gérer l'allocation mémoire, le paramètre environment_id pointe sur un tableau alloué par l'utilisateur pour mémoriser la définition de l'environnement et l'identifier. Dans ce cas, la direction du paramètre est IN.)

3.2.2 Création d'un environnement en mode d'exécution différée

Nom: FCDENV ou FB_CREATE_DELAYED_ENVIRONMENT

Catégorie: B

3.1.2 Close FASTBUS Session

When use of FASTBUS is no longer required, the User shall call the following routine:

Name: FBCLOS or FB_CLOSE

Category: A

Parameters:

return_code FB_error_code (OUT)

Effect:

If the User has left the FASTBUS Port in a state in which it is asserting any lines, those lines shall be deasserted. All allocated environment_id values are implicitly deallocated. No Operational Parameters shall affect this routine. If, after a call to the FB_CLOSE routine, the User calls any routines defined in this standard before calling the FB_OPEN routine again, an error shall be generated.

No Status History is generated.

3.2 Environment Management Routines

3.2.1 Create an Immediate Execution Mode Environment

Name: FCIENV or FB_CREATE_IMMEDIATE_ENVIRONMENT

Category: A

Parameters:

return_code FB_error_code (OUT)
environment_id FB_environment_id (OUT) (see text)

Effect:

An Environment shall be created or assigned which is in the state defined as default by the Implementation, and is in Immediate Execution Mode (FB_PAR_DELAY_EXEC is FB_FALSE). For Environments created this way the FB_PAR_DELAY_EXEC Operational parameter shall be read only. The value of the environment_id returned shall be used as an input parameter to specify the Environment to other routines.

For Implementations which handle memory allocation, this routine shall make available the required space for the Environment and shall return an environment_id.

(In Implementations where the User must handle memory allocation, the environment_id parameter points to a User allocated array to be used to store and identify the Environment definition. In this case the direction of the parameter is IN.)

3.2.2 Create a Delayed Execution Mode Environment

Name: FCDENV or FB_CREATE_DELAYED_ENVIRONMENT

Category: B

Paramètres:

return_code	FB_error_code	(OUT)	
environment_id	FB_environment_id	(OUT)	(voir texte)
maximum_calls	FB_integer	(IN)	

Effets:

Un environnement dans le mode d'exécution différée est créé ou assigné. Le paramètre opérationnel FB_PAR_DELAY_EXEC a la valeur FB_TRUE, le reste l'environnement est dans l'état défini par défaut par la réalisation. Pour un environnement créé de cette manière, le paramètre opérationnel FB_PAR_DELAY_EXEC doit être accessible en lecture et en écriture. L'environnement_id associé doit être défini et utilisé comme un paramètre d'entrée pour spécifier l'environnement aux autres routines de cette norme. Le paramètre maximum_calls doit définir le nombre maximal d'actions FASTBUS qui peuvent être placées dans la liste.

L'environnement créé doit contenir une liste vide ou nulle. Pour les réalisations qui allouent la mémoire, cette routine doit rendre disponible l'espace nécessaire à l'environnement et doit retourner un environment_id.

(Dans les réalisations où l'utilisateur doit gérer l'allocation mémoire, le paramètre environment_id pointe sur un tableau alloué par l'utilisateur pour mémoriser la définition de l'environnement et sa liste associée, et l'identifier. Dans ce cas, la direction du paramètre est IN.)

3.2.3 Libération d'un environnement

Nom: FRLENV ou FB_RELEASE_ENVIRONMENT

Catégorie: A

Paramètres:

return_code	FB_error_code	(OUT)
environment_id	FB_environment_id	(IN)

Effets:

Toutes les ressources système associées à l'environnement spécifié par environment_id sont libérées. Cet environment_id doit devenir illicite pour tout appel ultérieur aux routines de cette norme.

3.2.4 Initialisation d'un environnement

Nom: FRSENV ou FB_RESET_ENVIRONMENT

Catégorie: A

Paramètres:

return_code	FB_error_code	(OUT)
environment_id	FB_environment_id	(IN)

Effets:

L'environnement spécifié par environment_id doit être placé dans l'état qui est défini par défaut par la réalisation. Pour les réalisations par traitement de listes, l'environnement doit contenir une liste vide ou nulle. Cet environment_id doit rester valable pour tout appel ultérieur aux routines de cette norme.

Parameters:

return_code	FB_error_code	(OUT)	
environment_id	FB_environment_id	(OUT)	(see text)
maximum_calls	FB_integer	(IN)	

Effect:

A Delayed Execution Mode Environment shall be created or assigned. The Operational Parameter FB_PAR_DELAY_EXEC has a value of FB_TRUE, and otherwise the Environment is in the state defined as default by the implementation. For Environments created this way the FB_PAR_DELAY_EXEC Operational Parameter shall be read/write. An associated environment_id shall be defined and used to specify the Environment to other routines in this standard. The parameter maximum_calls shall specify the maximum number of FASTBUS Actions that may be placed in the List.

The Environment created shall contain an empty or null List. For Implementations that handle memory allocation, this routine shall make available the required space for the Environment and shall return an environment_id.

(In Implementations where the User must handle memory allocation, the environment_id parameter points to a User allocated array to be used to store and identify the Environment definition and its associated List. In this case the direction of the parameter is IN.)

3.2.3 Release Environment

Name: FRLENV or FB_RELEASE_ENVIRONMENT

Category: A

Parameters:

return_code	FB_error_code	(OUT)
environment_id	FB_environment_id	(IN)

Effect:

All system resources associated with the Environment specified by the environment_id shall be released. The environment_id shall become invalid for use in subsequent calls to the Routines in this standard.

3.2.4 Reset Environment

Name: FRSENV or FB_RESET_ENVIRONMENT

Category: A

Parameters:

return_code	FB_error_code	(OUT)
environment_id	FB_environment_id	(IN)

Effect:

The Environment specified by the environment_id shall be reset to the state which is defined as default by the Implementation. In List processing Implementations, the Environment shall contain an empty or null List. The environment_id shall remain valid for subsequent calls to the routines in this standard.

3.2.5 Copie d'un environnement

Nom: FCOENV ou FB_COPY_ENVIRONMENT

Catégorie: E

Paramètres:

return_code	FB_error_code	(OUT)
from_environment_id	FB_environment_id	(IN)
to_environment_id	FB_environment_id	(OUT) (voir texte)

Effets:

Cette routine doit créer un nouvel environnement identifié par to_environment_id identique à l'état actuel de l'environnement spécifié par from_environment_id. Les caractéristiques de environment_id décrites aux paragraphes 3.2.1 et 3.2.2 pour l'environnement des modes d'exécution immédiate ou différée doivent respectivement s'appliquer à to_environment_id.

3.2.6 Obtention d'un environnement

Nom: FGTEENV ou FB_GET_ENVIRONMENT

Catégorie: E

Paramètres:

return_code	FB_error_code	(OUT)
environment_id	FB_environment_id	(IN)
user_array	array_of_FB_word	(IN)
array_size	FB_integer	(IN)
byte_count	FB_integer	(OUT)

Effets:

L'information contenue dans l'environnement défini par environment_id doit être transférée dans le tableau user_array. Ce tableau doit avoir une longueur en octets de array_size. Le nombre d'octets réellement transférés doit être retourné dans byte_count. Si le tableau utilisateur n'est pas assez grand pour contenir l'environnement spécifié, une erreur doit être renvoyée et la valeur retournée dans byte_count doit être le nombre d'octets nécessaire à un transfert sans erreur.

3.2.7 Restauration d'un environnement

Nom: FSTENV ou FB_SET_ENVIRONMENT

Catégorie: E

Paramètres:

return_code	FB_error_code	(OUT)
environment_id	FB_environment_id	(IN)
user_array	array_of_FB_word	(IN)
byte_count	FB_integer	(IN)

Effets:

Byte_count octets d'informations doivent être transférés du tableau user_array dans l'environnement spécifié par environment_id.

3.2.5 Copy an Environment

Name: FCOENV or FB_COPY_ENVIRONMENT

Category: E

Parameters:

return_code	FB_error_code	(OUT)
from_environment_id	FB_environment_id	(IN)
to_environment_id	FB_environment_id	(OUT) (see text)

Effect:

This routine shall create a new Environment identified by to_environment_id identical to the current state of the Environment specified by the from_environment_id. The descriptions in sub-clauses 3.2.1 and 3.2.2 of the environment_id for Immediate and Delayed Execution Mode Environments respectively shall apply to the to_environment_id.

3.2.6 Get an Environment

Name: FGTEENV or FB_GET_ENVIRONMENT

Category: E

Parameters:

return_code	FB_error_code	(OUT)
environment_id	FB_environment_id	(IN)
user_array	array_of_FB_word	(IN)
array_size	FB_integer	(IN)
byte_count	FB_integer	(OUT)

Effect:

Information in the Environment specified by the environment_id shall be transferred to the array User_array. This array shall have a length of array_size bytes. The actual number of bytes transferred shall be returned as the byte_count. If the User_array is not large enough to contain the specified Environment an error shall be returned and the returned value of byte_count shall be the number of bytes required for an error-free transfer.

3.2.7 Set an Environment

Name: FSTENV or FB_SET_ENVIRONMENT

Category: E

Parameters:

return_code	FB_error_code	(OUT)
environment_id	FB_environment_id	(IN)
user_array	array_of_FB_word	(IN)
byte_count	FB_integer	(IN)

Effect:

Information of byte_count bytes in the array user_array shall be transferred to the Environment specified by the environment_id.

Ces informations devraient avoir été mémorisées dans `user_array` par un précédent appel à `FB_GET_ENVIRONMENT`. Pour les réalisations qui allouent la mémoire, un nouvel environnement est créé, identifié par le paramètre retourné `environment_id`. Pour les réalisations dans lesquelles l'utilisateur doit gérer l'allocation de la mémoire, le paramètre `environment_id` pointe sur un tableau défini par l'utilisateur qui contiendra la description de l'environnement. On remarquera que l'utilisation de certains modes de tampon peut engendrer des problèmes de relocation qui doivent être réglés par l'utilisateur.

3.3 Exécution différée et validation des listes

3.3.1 Exécution d'une liste

Nom: FLEXEC ou FB_LIST_EXECUTE

Catégorie: B

Paramètres:

<code>return_code</code>	<code>FB_error_code</code> (OUT)
<code>environment_id</code>	<code>FB_environment_id</code> (IN)

Effets:

Cette routine provoque la validation de la liste dans l'environnement défini par `environment_id` (si elle n'a pas déjà été validée) et ensuite son exécution (voir article 2.2). L'état du paramètre opérationnel `FB_PAR_DELAY_EXEC` doit être ignoré.

Un historique d'état (voir section 11) est produit.

3.3.2 Validation d'une liste

Nom: FLVAL ou FB_LIST_VALIDATE

Catégorie: E

Paramètres:

<code>return_code</code>	<code>FB_error_code</code> (OUT)
<code>environment_id</code>	<code>FB_environment_id</code> (IN)

Effets:

La réussite de cette routine doit indiquer, pour autant que l'on puisse le déterminer, que l'exécution de la liste dans l'environnement défini par `environment_id` sera réussie sur le port_id courant.

3.3.3 Obtention du pointeur courant dans une liste spécifiée

Nom: FLGPTR ou FB_LIST_GET_POINTER

Catégorie: B

Paramètres:

<code>return_code</code>	<code>FB_error_code</code> (OUT)
<code>environment_id</code>	<code>FB_environment_id</code> (IN)
<code>list_pointer</code>	<code>FB_list_pointer</code> (OUT)

The information should have been stored in the user_array by a previous call to FB_GET_ENVIRONMENT. For Implementations which handle memory allocation, a new Environment identified by the returned parameter environment_id is created. For Implementations in which the User must manage memory allocation the environment_id parameter points to a User-defined array which will contain the description of the Environment. Note that the use of some of the Buffer Modes can lead to relocatability problems which must be solved by the User.

3.3 Delayed Execution and List Validation

3.3.1 Execute a List

Name: FLEXEC or FB_LIST_EXECUTE

Category: B

Parameters:

return_code	FB_error_code (OUT)
environment_id	FB_environment_id (IN)

Effect:

This routine shall cause the List in the Environment specified by environment_id to be validated (if not previously validated) and then to be executed (see clause 2.2). The state of the Operational Parameter FB_PAR_DELAY_EXEC shall be ignored.

A Status History (see section 11) is generated.

3.3.2 Validate a List

Name: FLVAL or FB_LIST_VALIDATE

Category: E

Parameters:

return_code	FB_error_code (OUT)
environment_id	FB_environment_id (IN)

Effect:

Successful execution of this routine shall indicate that, as far as can be determined, the List in the Environment specified by environment_id will execute successfully for the current port_id.

3.3.3 Get Current Pointer in Specified List

Name: FLGPTR or FB_LIST_GET_POINTER

Category: B

Parameters:

return_code	FB_error_code (OUT)
environment_id	FB_environment_id (IN)
list_pointer	FB_list_pointer (OUT)

Effets:

Cette routine doit renvoyer un pointeur vers la première place libre dans la liste associée à l'environnement désigné.

La valeur retournée peut être utilisée, après l'exécution de la liste, pour extraire les informations d'état concernant la liste (voir section 11).

Withdrawing
IECNORM.COM: Click to view the full PDF of IEC 61052:1991

Effect:

This routine shall return the pointer to the next free place in the List associated with the named Environment.

The value returned may be used, after List execution, to extract status information about the List (see section 11).

Withdrawing
IECNORM.COM: Click to view the full PDF of IEC 61052:1991

Section 4. Paramètres opérationnels

4.1 Introduction

Les paramètres opérationnels permettent d'ajuster l'environnement pour une application particulière. Lorsque l'on crée un environnement, il contient un ensemble de paramètres opérationnels qui possèdent tous leur valeur par défaut qui peut dépendre de la réalisation. La valeur courante d'un paramètre opérationnel peut être modifiée par la routine FB_PAR_SET et sa valeur courante lue par les routines FB_PAR_GET et FB_PAR_READ.

4.1.1 Paramètres opérationnels dans le mode d'exécution différée

Il y a deux types de paramètres opérationnels dont la différence ne devient apparente que dans des environnements en mode d'exécution différée. Le premier type: les paramètres opérationnels globaux et de gestion d'erreur (article 4.4) doivent, par leur nature même, rester inchangés pendant l'exécution d'une liste. Le deuxième type: les paramètres opérationnels du protocole FASTBUS (article 4.5) se rapportent directement à l'activité sur le FASTBUS et peuvent être modifiés par une routine d'action qui fait partie de la liste. Les paramètres opérationnels globaux et de gestion d'erreur ne peuvent être modifiés que par la routine FB_PAR_SET tandis que ceux du protocole FASTBUS peuvent être changés par les routines FB_PAR_SET et FB_PAR_WRITE.

La routine FB_PAR_SET redéfinit la valeur courante d'un paramètre opérationnel et ainsi, pour un environnement en mode d'exécution différée, redéfinit l'environnement de départ pour l'exécution de la liste. La routine FB_PAR_WRITE permet de changer les paramètres opérationnels du protocole FASTBUS pendant l'exécution d'une liste mais sans modifier l'ensemble des paramètres opérationnels pris en compte lorsque la liste sera réexécutée. L'effet d'une action FB_PAR_WRITE peut être changé entre deux exécutions successives d'une liste en modifiant la valeur correspondante dans le tampon indiqué. La routine FB_PAR_READ permet d'examiner les paramètres opérationnels du protocole FASTBUS pendant l'exécution d'une liste.

4.2 Définition des paramètres opérationnels

Les paramètres opérationnels définissent une partie de l'environnement. Les paramètres opérationnels associés à différents environnement ids sont complètement indépendants. Un paramètre opérationnel est représenté par une valeur unique sur 32 bits. Certains paramètres opérationnels ne prennent que les valeurs FB_TRUE et FB_FALSE. Certains paramètres opérationnels peuvent être accessibles seulement en lecture. L'écriture d'un paramètre en lecture seule à la valeur fixée par la réalisation ne doit pas être considérée comme une erreur. Une réalisation peut posséder des paramètres opérationnels spécifiques.

La forme courte des noms de tous les paramètres opérationnels définis ci-dessous doit être reconnue par toutes les réalisations.

Une réalisation de cette norme n'a pas besoin de donner accès à tous les paramètres opérationnels. Le retour d'une erreur doit indiquer que l'on s'est référé à des paramètres opérationnels qui ne sont pas supportés par la réalisation matérielle ou logicielle (voir section 12).

Une réalisation doit définir des valeurs par défaut pour tous les paramètres qu'elle supporte. Les valeurs par défaut, lorsqu'elles sont définies dans cette norme, doivent être utilisées par toutes les réalisations.

Section 4. Operational Parameters

4.1 Introduction

The Operational Parameters allow an Environment to be tailored for particular applications. When an Environment is created it is given a set of Operational Parameters all of which have their, possibly implementation dependent, default values. The current value of an Operational Parameter can be changed by the FB_PAR_SET routine and its current value determined by the FB_PAR_GET and FB_PAR_READ routines.

4.1.1 Operational Parameters in Delayed Execution Mode

There are two types of Operational Parameters whose difference becomes apparent only for Delayed Execution Mode Environments. The first type, the Overall and Error Handling Operational Parameters (clause 4.4) must, by their very nature, remain unchanged during the execution of a List. The second type, the FASTBUS Protocol Operational Parameters (clause 4.5), directly relate to activity on the FASTBUS and can be changed by an action routine which is part of a List. The Overall and Error Handling Operational Parameters can only be changed by the FB_PAR_SET routine while the FASTBUS Protocol ones can be changed by both the FB_PAR_SET and FB_PAR_WRITE routines.

The FB_PAR_SET routine redefines the current value of an Operational Parameter and hence, for Delayed Execution Mode Environments, redefines the starting environment for List Execution. The FB_PAR_WRITE routine allows the FASTBUS Protocol Operational Parameters to be changed during the execution of a List but without affecting the set of Operational Parameters in effect when the List is reexecuted. The effect of any FB_PAR_WRITE action may be changed between successive executions of a List by changing the value in the referenced data buffer. The FB_PAR_READ routine allows the FASTBUS Protocol Operational Parameters to be examined during List execution.

4.2 Definition of Operational Parameters

Operational Parameters define part of the Environment. Operational Parameters associated with different environment_ids are completely independent. An Operational Parameter is represented by a single 32-bit value. Some Operational Parameters take only the values FB_TRUE and FB_FALSE. Some Operational Parameters may be Read-only. The setting of a Read-only parameter to its Implementation fixed value shall not be considered to be an error. There may also be Implementation specific Operational Parameters.

The short form of the names of all the Operational Parameters defined below shall be recognized by any Implementation.

A given Implementation of this standard need not provide access to all Operational Parameters. Operational Parameters that are not supported by the software or hardware Implementation shall, if referred to, return an error indicating this (see section 12).

An Implementation shall define default values for all supported parameters. Default values when defined in this standard shall be used by all Implementations.

Une réalisation peut restreindre l'accès à certains paramètres opérationnels (et retourner un code d'erreur indiquant une insuffisance de privilège pour l'opération tentée lorsqu'un essai est fait pour y accéder). Cela sera particulièrement nécessaire lorsque la réalisation permet des utilisateurs multiples.

4.3 Routines pour les paramètres opérationnels

Les routines suivantes doivent être utilisées pour accéder aux paramètres opérationnels. Ces paramètres doivent toujours être référencés par leur nom tel qu'il est défini dans les paragraphes suivants.

4.3.1 Initialisation des paramètres opérationnels

Nom: FBPINI ou FB_PAR_INIT

Catégorie: A

Paramètres:

return_code	FB_error_code (OUT)
environment_id	FB_environment_id (IN)
parameter_id	FB_parameter_id (IN)

Effets:

Cette routine doit restaurer le paramètre opérationnel défini par parameter_id à sa valeur par défaut définie dans l'environnement actuel par la réalisation. Si parameter_id vaut FB_PAR_ALL (FPALL), tous les paramètres doivent être fixés à leurs valeurs par défaut.

4.3.2 Réglage d'un paramètre opérationnel

Nom: FBPSET ou FB_PAR_SET

Catégorie: A

Paramètres:

return_code	FB_error_code (OUT)
environment_id	FB_environment_id (IN)
parameter_id	FB_parameter_id (IN)
parameter_value	FB_parameter_value (IN)

Effets:

Lorsqu'elle est appelée, cette routine doit assigner la valeur parameter_value au paramètre opérationnel défini par parameter_id dans l'environnement spécifié. Après l'appel à cette routine, parameter_value spécifié doit devenir la valeur courante du paramètre opérationnel désigné.

4.3.3 Obtention d'un paramètre opérationnel

Nom: FBPGET ou FB_PAR_GET

Catégorie: A

An Implementation may restrict access to certain Operational Parameters (and return an Error Code indicating insufficient privilege for the attempted operation when an attempt is made to access them). This will be especially necessary when an Implementation allows multiple Users.

4.3 Operational Parameter Routines

The following routines shall be used to provide access to Operational Parameters. The parameters shall always be referred to by their names as defined in the following subclauses.

4.3.1 Initialize Operational Parameter

Name: FBPINI or FB_PAR_INIT

Category: A

Parameters:

return_code	FB_error code	(OUT)
environment_id	FB_environment_id	(IN)
parameter_id	FB_parameter_id	(IN)

Effect:

This routine shall cause the Operational Parameter specified by parameter_id to be restored to its Implementation defined default value in the specified Environment. If parameter_id is set to FB_PAR_ALL (FPALL) all parameters shall be set to their default values.

4.3.2 Set Operational Parameter

Name: FBPSET or FB_PAR_SET

Category: A

Parameters:

return_code	FB_error_code	(OUT)
environment_id	FB_environment_id	(IN)
parameter_id	FB_parameter_id	(IN)
parameter_value	FB_parameter_value	(IN)

Effect:

When called, this routine shall cause the Operational Parameter specified by parameter_id, in the specified Environment, to be assigned the value parameter_value. After a call to this routine, the specified parameter_value shall become the current value for the named Operational Parameter.

4.3.3 Get Operational Parameter

Name: FBPGET or FB_PAR_GET

Category: A

Paramètres:

return_code	FB_error_code	(OUT)
environment_id	FB_environment_id	(IN)
parameter_id	FB_parameter_id	(IN)
parameter_value	FB_parameter_value	(OUT)

Effets:

Lorsqu'elle est appelée, cette routine doit renvoyer dans parameter_value la valeur actuelle du paramètre opérationnel désigné dans l'environnement spécifié.

4.3.4 Lecture d'un paramètre opérationnel

Nom: FBPRD ou FB_PAR_READ

Catégorie: B, Routine d'action

Paramètres:

return_code	FB_error_code	(OUT)
environment_id	FB_environment_id	(IN)
parameter_id	FB_parameter_id	(IN)
buffer_mode	FB_buffer_mode	(IN)
buffer_specifier	FB_word	(OUT) (FBVAR)
	ou FB_seq_buffer_id	(IN) (FBSEQ)

Action:

Quel que soit le mode d'exécution de l'environnement spécifié, cette routine doit placer, dans le tampon de données indiqué, la valeur effective au moment de l'exécution du paramètre opérationnel désigné. Cette routine ne doit être utilisée que pour les paramètres opérationnels de l'article 4.5.

4.3.5 Ecriture d'un paramètre opérationnel

Nom: FBPWRT ou FB_PAR_WRITE

Catégorie: B, Routine d'action

Paramètres:

return_code	FB_error_code	(OUT)
environment_id	FB_environment_id	(IN)
parameter_id	FB_parameter_id	(IN)
buffer_mode	FB_buffer_mode	(IN)
buffer_specifier	FB_word	(OUT) (FBVAR)
	ou FB_seq_buffer_id	(IN) (FBSEQ)

Action:

Quel que soit le mode d'exécution de l'environnement spécifié, cette routine doit, au moment de l'exécution, remplacer la valeur courante du paramètre opérationnel désigné par la valeur sise dans le tampon de données indiqué. Les effets de FB_PAR_WRITE dans une liste ne doivent pas se maintenir de l'exécution de cette liste à la suivante. Cette routine ne doit être utilisée que pour les paramètres opérationnels de l'article 4.5.

Parameters:

return_code	FB_error_code	(OUT)
environment_id	FB_environment_id	(IN)
parameter_id	FB_parameter_id	(IN)
parameter_value	FB_parameter_value	(OUT)

Effect:

When called this routine shall cause the current value of the named Operational Parameter in the specified Environment to be returned as the parameter_value.

4.3.4 Read Operational Parameter

Name: FBPRD or FB_PAR_READ

Category: B, Action Routine

Parameters:

return_code	FB_error_code	(OUT)
environment_id	FB_environment_id	(IN)
parameter_id	FB_parameter_id	(IN)
buffer_mode	FB_buffer_mode	(IN)
buffer_specifier	FB_word	(OUT) (FBVAR)
	or FB_seq_buffer_id	(IN) (FBSEQ)

Action:

Regardless of the Execution Mode of the specified Environment, this routine shall at Execution Time place the current value of the named Operational Parameter into the specified data buffer. This routine shall only be used for the Operational Parameters in clause 4.5.

4.3.5 Write Operational Parameter

Name: FBPWRT or FB_PAR_WRITE

Category: B, Action Routine

Parameters:

return_code	FB_error_code	(OUT)
environment_id	FB_environment_id	(IN)
parameter_id	FB_parameter_id	(IN)
buffer_mode	FB_buffer_mode	(IN)
buffer_specifier	FB_word	(OUT) (FBVAR)
	or FB_seq_buffer_id	(IN) (FBSEQ)

Action:

Regardless of the Execution Mode of the specified Environment, this routine shall at Execution Time cause the current value of the named Operational Parameter to be changed to the Execution Time value in the specified data buffer. The effects of FB_PAR_WRITE within a List shall not carry over from one execution of a List to the next. This routine shall only be used for the Operational Parameters in clause 4.5.

4.3.6 Empilage des paramètres opérationnels courants

Nom: FBPUSH ou FB_PAR_PUSH

Catégorie: E

Paramètres:

return_code	FB_error_code	(OUT)
environment_id	FB_environment_id	(IN)

Effets:

Lorsqu'elle est appelée, cette routine doit pousser les paramètres opérationnels courants de l'environnement spécifié sur une pile interne des paramètres opérationnels spécifique de l'environnement.

4.3.7 Dépilage des paramètres opérationnels courants

Nom: FBPOP ou FB_PAR_POP

Catégorie: E

Paramètres:

return_code	FB_error_code	(OUT)
environment_id	FB_environment_id	(IN)

Effets:

Lorsqu'elle est appelée, cette routine doit remplacer les paramètres opérationnels de l'environnement désigné par ceux retirés du sommet de la pile interne des paramètres opérationnels spécifique de l'environnement.

4.4 Paramètres opérationnels globaux et de gestion d'erreur

Les paramètres opérationnels globaux sont ceux associés à des activités ou des entités qui ne sont pas directement reliées aux détails de l'activité sur le FASTBUS lui-même (ceux-ci sont décrits plus loin à l'article 4.5). Par exemple, l'identité d'un port FASTBUS, la décision d'attendre la fin d'une action FASTBUS avant le retour à l'appelant, la taille et le mode d'exécution d'un environnement sont des paramètres opérationnels globaux. Les paramètres opérationnels de gestion d'erreur sont utilisés pour contrôler le rapport des comptes rendus d'état. Ils sont discutés en détail à la section 11. Les termes "état résumé" et "état complémentaire" sont définis à l'article 11.3.

4.4.1 Identificateur du port du contrôleur

Nom: FPPORT ou FB_PAR_PORT

Valeur: définie par la réalisation

Défaut: défini par la réalisation

Ce paramètre opérationnel doit identifier le contrôleur FASTBUS et le port à travers lequel toute action sera exécutée. Un contrôleur donné peut posséder plus d'un port FASTBUS. Ce paramètre, qui dépend de la réalisation, définit le port à utiliser pour les demandes d'action suivantes lorsque plusieurs chemins sont disponibles vers le FASTBUS.

4.3.6 Push Current Operational Parameters onto Stack

Name: FBPUSH or FB_PAR_PUSH

Category: E

Parameters:

return_code	FB_error_code	(OUT)
environment_id	FB_environment_id	(IN)

Effect:

When called, this routine shall replace the current Operational Parameters of the specified Environment onto an internal Environment specific Operational Parameter stack.

4.3.7 Pop Current Operational Parameters off Stack

Name: FBPOP or FB_PAR_POP

Category: E

Parameters:

return_code	FB_error_code	(OUT)
environment_id	FB_environment_id	(IN)

Effect:

When called, this routine shall replace the Operational Parameters of the named Environment by the ones removed from the top of the internal Environment specific Operational Parameter stack.

4.4 Overall and Error Handling Operational Parameters

The Overall Operational Parameters are those associated with activities or entities not directly related to the details of the activity on the FASTBUS itself. (These latter are described in clause 4.5.) Hence the identity of a FASTBUS Port, whether to wait for completion of FASTBUS activity before returning to the caller and the size and execution mode of the Environment are all Overall Operational Parameters. The Error Handling Operational Parameters are used to control status reporting. They are discussed in more detail in section 11. The terms Summary Status and Supplementary Status are defined in clause 11.3.

4.4.1 Controller Port Identifier

Name: FPPORT or FB_PAR_PORT

Values: Implementation dependent

Default: Implementation dependent

This Operational Parameter shall identify the FASTBUS Controller and Port through which any Actions are performed. It is possible that a given Controller has more than one FASTBUS Port. This Implementation dependent parameter specifies which Port to use for subsequent requests for Actions when more than one path to FASTBUS is available.

Exécution en mode différée:

Remarquer que le changement de port à utiliser par un environnement donné, après que la liste des actions à exécuter a été construite, peut entraîner l'affectation au port de paramètres n'existant pas ou illégaux. Cette difficulté n'apparaîtra que lors de l'appel suivant à FB_LIST_EXECUTE.

4.4.2 Absence d'attente de fin d'exécution des actions

Nom: FPNOWT ou FB_PAR_NO_WAIT
Valeur: FB_TRUE ou FB_FALSE
Défaut: FB_FALSE

Si ce paramètre opérationnel vaut FB_FALSE, toute routine d'action ou appel à FB_LIST_EXECUTE ne doit retourner à la routine d'appel que lorsque l'action demandée est finie, que ce soit avec ou sans erreur. S'il vaut FB_TRUE, le retour vers la routine d'appel doit se faire immédiatement après que l'action demandée a démarré, éventuellement avant qu'elle soit terminée.

4.4.3 Retard à l'exécution

Nom: FPDLAY ou FB_PAR_DELAY_EXEC
Valeur: FB_TRUE ou FB_FALSE
Défaut: FB_FALSE - pour un environnement créé en mode immédiat (lecture seulement)
FB_TRUE - pour un environnement créé en mode différé (lecture/écriture)

Ce paramètre opérationnel définit le mode d'exécution de l'environnement. Il a la valeur FB_FALSE pour un environnement en mode d'exécution immédiate ou la valeur FB_TRUE pour un environnement en mode d'exécution différée.

4.4.4 Taille de l'environnement

Nom: FPENVS ou FB_PAR_ENVIRONMENT_SIZE
Valeur: définie par la réalisation
Défaut: défini par la réalisation

Ce paramètre en lecture seule doit être la taille courante, en octets, de l'environnement.

4.4.5 Limitation de la génération des comptes rendus d'état

Nom: FPNSTA ou FB_PAR_NO_STATUS
Valeur: FB_TRUE ou FB_FALSE
Défaut: défini par la réalisation

Fixer ce paramètre opérationnel à FB_TRUE permet une exécution plus rapide des routines d'action ou de FB_LIST_EXECUTE, en sacrifiant les informations d'état détaillées sur les actions exécutées.

Si ce paramètre opérationnel a la valeur FB_FALSE, chaque routine d'action ou appel à FB_LIST_EXECUTE doit mémoriser à la fois les informations de l'état résumé et de l'état supplémentaire (voir section 11). S'il est FB_TRUE, seul l'état résumé doit être mémorisé.

Delayed Mode Execution:

Note that changing the Port to be used by a given Environment, after the List of Actions to be performed has been built, may lead to unimplemented or illegal parameters being assigned to the Port. This difficulty may only be revealed at the next call to FB_LIST_EXECUTE.

4.4.2 Do Not Wait for Completion of Action(s)

Name: FPNOWT or FB_PAR_NO_WAIT
Values: FB_TRUE or FB_FALSE
Default: FB_FALSE

If this Operational Parameter is set to FB_FALSE any Action Routine or call to FB_LIST_EXECUTE shall only return to the calling routine once the requested Actions are completed - either with or without error. If it is set to FB_TRUE the return to the calling routine shall be made immediately after the requested Actions are started, possibly before they are complete.

4.4.3 Delay Execution

Name: FPDLAY or FB_PAR_DELAY_EXEC
Values: FB_TRUE or FB_FALSE
Default: FB_FALSE - for an Environment created in Immediate Mode (read only)
 FB_TRUE - for an Environment created in Delayed Execution Mode (read/write)

This Operational Parameter specifies the Execution Mode of the Environment. It has the value FB_FALSE for an Immediate Execution Mode Environment or the value FB_TRUE for a Delayed Execution Mode Environment.

4.4.4 Environment Size

Name: FPENVS or FB_PAR_ENVIRONMENT_SIZE
Values: Implementation dependent
Default: Implementation dependent

This read-only parameter shall be the current size in bytes of the Environment.

4.4.5 Limit Status Generation

Name: FPNSTA or FB_PAR_NO_STATUS
Values: FB_TRUE or FB_FALSE
Default: Implementation dependent

Setting this Operational Parameter to FB_TRUE allows for faster execution of Action Routines or FB_LIST_EXECUTE, with the sacrifice of detailed status information about the actions performed.

If this Operational Parameter has the value FB_FALSE, each Action routine or call to FB_LIST_EXECUTE shall store both Summary and Supplementary Status information (see section 11). If it is FB_TRUE only the Summary Status shall be stored.

4.4.6 Limitation du code de retour au niveau de sévérité

Nom: FPRSEV ou FB_PAR_RETURN_SEVERITY
Valeur: FB_TRUE ou FB_FALSE
Défaut: FB_FALSE

Fixer ce paramètre opérationnel à FB_TRUE permet une exécution plus rapide des routines d'action ou de FB_LIST_EXECUTE, en sacrifiant le détail des codes de retour.

Si ce paramètre opérationnel vaut FB_TRUE, le code de retour renvoyé doit seulement rapporter la sévérité des erreurs rencontrées. Ce sera l'un des codes d'erreur: FB_ERR_SUCCESS, FB_ERR_INFO, FB_ERR_WARN, FB_ERR_ERROR ou FB_ERR_FATAL. Des informations d'état plus détaillées peuvent être accédées par un appel à l'une des routines de décodage d'état de la section 11.

Si ce paramètre opérationnel a la valeur FB_FALSE, le code de retour doit être un code d'erreur contenant les informations d'état et la sévérité associée.

4.4.7 Seuil de sévérité pour le traitement des erreurs

Nom: FPHATH ou FB_PAR_HANDLER_THRESHOLD
Valeur: toute valeur du type FB_severity
Défaut: FB_SEV_WARNING

Ce paramètre opérationnel doit définir le seuil de sévérité pour lequel le programme de traitement d'erreur de l'utilisateur doit être appelé. Si le code d'erreur le plus sévère résultant de l'exécution d'une action ou d'une liste d'actions est égal ou supérieur à ce seuil, un programme utilisateur de traitement d'erreur (connecté en utilisant la routine FB_STATUS_CONNECT) doit être appelé avant que le rapporteur automatique des erreurs ne fonctionne.

4.4.8 Seuil de sévérité pour les rapports d'erreur

Nom: FPRETH ou FB_PAR_REPORT_THRESHOLD
Valeur: toute valeur du type FB_severity
Défaut: FB_SEV_WARNING

Ce paramètre opérationnel doit définir le seuil de sévérité pour lequel le rapport automatique des erreurs se produit. La manière de rapporter les erreurs dépend de la réalisation.

Si la sévérité du code de retour de toute routine appelée dans cette norme (telle qu'elle est modifiée par tout appel à un programme utilisateur de traitement des erreurs) est égale ou supérieure à ce seuil, le rapport automatique de l'erreur ou des erreurs doit s'effectuer avant de revenir de cette routine.

4.4.9 Seuil de sévérité d'exception

Nom: FPEXTH ou FB_PAR_EXCEPTION_THRESHOLD
Valeur: toute valeur du type FB_severity
Défaut: FB_SEV_ERROR

Ce paramètre opérationnel doit définir le seuil de sévérité pour lequel une condition d'exception - généralement l'arrêt du programme - doit être engendrée, une fois qu'un rapport automatique d'erreur est terminé.

Les détails sur cette exception dépendront de la réalisation.

4.4.6 Return Code contains only the Severity

Name: FPRSEV or FB_PAR_RETURN_SEVERITY
Values: FB_TRUE or FB_FALSE
Default: FB_FALSE

Setting this Operational Parameter to FB_TRUE allows for faster execution of Action Routines or FB_LIST_EXECUTE, with the sacrifice of a more detailed Return Code.

If this Operational Parameter is FB_TRUE the Return Code returned by a routine shall only report the Severity of any error encountered. That is, it will be one of the Error Codes FB_ERR_SUCCESS, FB_ERR_INFO, FB_ERR_WARN, FB_ERR_ERROR or FB_ERR_FATAL. More detailed status information may be available through a call to one of the status decoding routines in section 11.

If this Operational Parameter has the value FB_FALSE, the Return Code shall be an Error Code containing status information and the associated Severity.

4.4.7 Error Handler Severity Threshold

Name: FPHATH or FB_PAR_HANDLER_THRESHOLD
Values: any value of type FB_severity
Default: FB_SEV_WARNING

This Operational Parameter shall specify the Severity threshold at which the User error handler will be called. If the most severe Error Code resulting from execution of an Action or List of Actions is equal to or greater than this threshold, a User error handler (connected using the routine FB_STATUS_CONNECT) shall be called before the Automatic Error Reporter acts.

4.4.8 Error Reporting Severity Threshold

Name: FPRETH or FB_PAR_REPORT_THRESHOLD
Values: any value of type FB_severity
Default: FB_SEV_WARNING

This Operational Parameter shall specify the Severity threshold at which the automatic reporting of errors will occur. The means of reporting the errors is Implementation dependent.

If the Severity of the Return Code from any routine call in this standard (as modified by any call to a User error handler) is equal to or greater than this threshold, automatic reporting of the error or errors shall occur before the return from the routine.

4.4.9 Exception Severity Threshold

Name: FPEXTH or FB_PAR_EXCEPTION_THRESHOLD
Values: any value of type FB_severity
Default: FB_SEV_ERROR

Once any automatic error reporting has been completed, this Operational Parameter shall specify the Severity threshold at which an Exception Condition - usually program termination - will be generated.

The details of this exception shall be Implementation dependent.

Si la sévérité du code de retour de toute routine appelée dans cette norme (telle qu'elle est modifiée par tout appel à un programme utilisateur de traitement des erreurs) est égale ou supérieure à ce seuil, l'exception doit être signalée.

4.4.10 Seuil de sévérité des messages

Nom: FPMETH ou FB_PAR_MESSAGE_THRESHOLD
Valeur: toute valeur du type FB_severity
Défaut: FB_SEV_SUCCESS

Ce paramètre opérationnel doit définir le seuil de sévérité pour lequel le rapport automatique des codes individuels d'erreur se produit.

Le code de retour de la routine appelée (tel qu'il est modifié par tout appel à un programme utilisateur de traitement des erreurs) et tous les codes d'erreur dans les états résumés ou les états supplémentaires, dont la sévérité est égale ou supérieure à ce seuil, doivent être rapportés.

4.4.11 Concision des messages de rapport

Nom: FPRSUM ou FB_PAR_REPORT_TERSE
Valeur: FB_TRUE ou FB_FALSE
Défaut: FB_FALSE

Ce paramètre opérationnel doit contrôler la quantité d'information d'état qui sera rapportée par chaque appel au rapporteur automatique des erreurs.

S'il est FB_FALSE, toutes les informations d'état disponibles associées à chaque code d'erreur à rapporter seront rapportées. S'il est FB_TRUE, le rapporteur automatique d'erreur rapportera seulement un simple message.

Les détails du format et du contenu de ce message dépendent de la réalisation.

4.4.12 Rapport sur chaque action

Nom: FPRACT ou FB_PAR_REPORT_ACTIONS
Valeur: FB_TRUE ou FB_FALSE
Défaut: FB_FALSE

Ce paramètre opérationnel doit contrôler si, lorsqu'une action est exécutée, une information est rapportée par le rapporteur automatique des erreurs.

S'il est FB_FALSE, aucune information n'est rapportée.

S'il est FB_TRUE, le rapporteur automatique d'erreur rapportera pour chaque action lorsqu'elle se produit.

Les détails du format et du contenu de ce message dépendent de la réalisation.

4.5 Paramètres opérationnels du protocole FASTBUS, des temporisateurs et des réessais

Les paramètres opérationnels du protocole FASTBUS conditionnent l'action exécutée par le port sur le FASTBUS. On peut accéder à ces paramètres par les routines FB_PAR_GET, FB_PAR_SET, FB_PAR_READ et FB_PAR_WRITE. Les actions des routines FB_PAR_READ et FB_PAR_WRITE peuvent être différées selon l'environnement, alors que les actions de FB_PAR_GET et FB_PAR_SET ne le peuvent pas.

If the Severity of the Return Code from any routine call in this standard (as modified by any call to a User error handler) is equal to or greater than this threshold, the exception condition shall be signalled.

4.4.10 Message Severity Threshold

Name: FPMETH or FB_PAR_MESSAGE_THRESHOLD
Values: any value of type FB_severity
Default: FB_SEV_SUCCESS

This Operational Parameter shall specify the Severity threshold at which the automatic reporting of individual Error Codes will occur.

The Return Code from the routine call (as modified by any call to a User error handler), and any Error Codes in the Summary or Supplementary Status, whose Severity is equal to or greater than this threshold, shall be reported.

4.4.11 Report Terse Message

Name: FPRSUM or FB_PAR_REPORT_TERSE
Values: FB_TRUE or FB_FALSE
Default: FB_FALSE

This Operational Parameter shall control the amount of status information that will be reported in any call to the automatic error reporter.

If it is FB_FALSE, all available status information associated with any Error Codes to be reported will be reported. If it is FB_TRUE, the automatic error reporter will only report a single message.

The details of the contents and size of the message are Implementation dependent.

4.4.12 Report On Each Action

Name: FPRACT or FB_PAR_REPORT_ACTIONS
Values: FB_TRUE or FB_FALSE
Default: FB_FALSE

This Operational Parameter shall control whether as actions are performed information is reported by the automatic error reporter.

If it is FB_FALSE, no such information is reported.

If it is FB_TRUE, the automatic error reporter will report on each action as it occurs.

The details of the contents and size of the message are Implementation dependent.

4.5 FASTBUS Protocol, Timeout and Retry Operational Parameters

The FASTBUS Protocol Operational Parameters condition the action on FASTBUS as performed by the Port. These parameters can be accessed by the FB_PAR_GET, FB_PAR_SET, FB_PAR_READ and FB_PAR_WRITE routines. The actions of the FB_PAR_READ and FB_PAR_WRITE routines may be delayed depending on the Environment but the actions of the FB_PAR_GET and FB_PAR_SET are not.

La norme FASTBUS définit quatre temporisateurs dans un maître FASTBUS. On dispose de paramètres opérationnels pour contrôler ces temporisateurs lorsqu'ils peuvent être contrôlés par une réalisation. Un temporisateur logiciel global est aussi défini. Par défaut tous les temporisateurs doivent être en service.

Si une réponse SS non prévue se produit pendant un cycle FASTBUS, la réalisation peut répondre en réessayant le cycle erroné. On décrit à la section 11 les moyens de spécifier cette réponse. On dispose de paramètres opérationnels pour spécifier le nombre de réessais et l'intervalle de temps entre chacun.

4.5.1 Niveau d'arbitrage

Nom: FPARBL ou FB_PAR_ARBITRATION_LEVEL
Valeur: 0 à 63
Défaut: défini par la réalisation

Ce paramètre doit spécifier, pour les appels ultérieurs à des routines d'action, la valeur à utiliser dans le registre de niveau d'arbitrage (CSR#8 <05.00>) du maître associé. Ce registre contient, en plus du niveau d'arbitrage du maître, certaines informations concernant le protocole à utiliser par le maître lorsqu'il positionne AR (voir la norme FASTBUS, CEI 935, paragraphe 6.3.1 et article 8.11).

Ce paramètre ainsi que les suivants, qui affectent le processus d'arbitrage, ne devraient être définis que par l'hôte.

4.5.2 Arbitrage dans le protocole d'accès assuré

Nom: FPASSA ou FB_PAR_ASSURED_ACCESS
Valeur: FB_TRUE ou FB_FALSE
Défaut: FB_FALSE

Ce paramètre doit définir l'état de CSR#8 <07>, le bit d'accès assuré, dans le registre de niveau d'arbitrage pour les actions postérieures à l'appel de la routine. Si ce paramètre vaut FB_TRUE, ce bit doit être positionné, et s'il vaut FB_FALSE, ce bit doit être remis à zéro. Lorsque CSR#8 <07> = 1, le maître doit suivre le protocole d'accès assuré lorsqu'il positionne AR.

4.5.3 Arbitrage dans le protocole d'accès prioritaire

Nom: FPPRIA ou FB_PAR_PRIORITIZED_ACCESS
Valeur: FB_TRUE ou FB_FALSE
Défaut: FB_FALSE

Ce paramètre doit définir l'état de CSR#8 <06>, le bit d'accès prioritaire, dans le registre de niveau d'arbitrage pour les actions postérieures à l'appel de la routine. Si ce paramètre vaut FB_TRUE, ce bit doit être positionné, et s'il vaut FB_FALSE, ce bit doit être remis à zéro. Lorsque CSR#8 <06> = 1, le maître doit suivre le protocole d'accès prioritaire lorsqu'il positionne AR.

4.5.4 Conservation de la maîtrise du bus pour un bloc complet

Nom: FPWBLK ou FB_PAR_WHOLE_BLOCK
Valeur: FB_TRUE ou FB_FALSE
Défaut: défini par la réalisation

Ce paramètre doit contrôler si la maîtrise du FASTBUS doit être relâchée pendant les opérations suivantes de transfert de bloc.

The FASTBUS standard defines four timers in a FASTBUS Master. Operational Parameters are provided to control these timers when they may be controlled by an Implementation. An overall software timer is also defined. By default all timers shall be enabled.

If an unexpected SS response occurs during a FASTBUS cycle, the Implementation may respond by retrying the cycle in error. Means for specifying this response are described in section 11. Operational Parameters are defined to specify the number of retries and the time interval between each one.

4.5.1 Arbitration Level

Name: FPARBL or FB_PAR_ARBITRATION_LEVEL
Values: 0 to 63
Default: Implementation dependent

This parameter shall specify the value to be used in the associated Master's Arbitration Level Register (CSR#8 <05:00>) for subsequent Action Routine calls. Besides containing the Master's Arbitration Level this register also contains information concerning the protocol to be used by the Master when asserting AR (see FASTBUS standard, IEC 935, subclause 6.3.1 and clause 8.11).

This parameter, as well as those below that affect the arbitration process, should only be set by the Host.

4.5.2 Assured Access Arbitration Protocol

Name: FPASSA or FB_PAR_ASSURED_ACCESS
Values: FB_TRUE or FB_FALSE
Default: FB_FALSE

This parameter shall specify the state of CSR#8 <07>, the Assured Access bit in the Arbitration Level Register for subsequent Action Routine calls. If FB_TRUE this bit shall be set, and if FB_FALSE the bit shall be clear. When CSR#8 <07> = 1 the Master shall follow the Assured Access protocol when asserting AR.

4.5.3 Prioritized Access Arbitration Protocol

Name: FPPRIA or FB_PAR_PRIORITIZED_ACCESS
Values: FB_TRUE or FB_FALSE
Default: FB_FALSE

This parameter shall specify the state of CSR#8 <06>, the Prioritized Access bit in the Arbitration Level Register for subsequent Action Routine calls. If FB_TRUE this bit shall be set, and if FB_FALSE the bit shall be clear. When CSR#8 <06> = 1 the Master shall follow the Prioritized Access protocol when asserting AR.

4.5.4 Hold Mastership for Whole Block

Name: FPWBLK or FB_PAR_WHOLE_BLOCK
Values: FB_TRUE or FB_FALSE
Default: Implementation dependent

This parameter shall control whether or not FASTBUS mastership will be released during any subsequent Block Transfer Operation.

S'il vaut `FB_FALSE`, le maître peut, par tout moyen disponible, découper le transfert du bloc complet en un certain nombre de morceaux successifs (paquets - voir paragraphe 6.1.5).

Les paramètres opérationnels `FB_PAR_HOLD_BUS`, `FB_AR_HOLD_BUS_NO_AR`, `FB_PAR_HOLD_AS` et `FB_PAR_HOLD_ON_ERROR` définis ci-après ont priorité sur ce paramètre pour déterminer si la maîtrise du bus doit être relâchée.

4.5.5 Dimension des paquets dans un transfert de bloc FASTBUS

Nom: `FPBLSZ` ou `FB_PAR_BLOCKLET_SIZE`
Valeur: nombre de cycles de données FASTBUS (transitions DS)
Défaut: défini par la réalisation

Ce paramètre doit définir la dimension maximale des paquets pour les transferts de blocs FASTBUS suivants.

4.5.6 Transfert en pipe-line

Nom: `FPPIPE` ou `FB_PAR_PIPELINE`
Valeur: `FB_TRUE` ou `FB_FALSE`
Défaut: `FB_FALSE`

Lorsque ce paramètre a la valeur `FB_TRUE`, chaque transfert de blocs FASTBUS suivant doit être exécuté en mode de transfert en pipe-line ($MS=3$).

La période entre les transitions du signal de cadencement des données (`DS(u)` à `DS(d)`) provient du paramètre `FB_PAR_CLOCK` si la réalisation a accès à ce paramètre.

4.5.7 Période des cycles de données dans un transfert en pipe-line

Nom: `FPCLK` ou `FB_PAR_CLOCK`
Valeur: temps entre les transitions de DS en nanosecondes
Défaut: défini par la réalisation

Lorsque les données sont transférées par un transfert en pipe-line, les transitions du signal DS se produisent à une période déterminée par le maître. Si cette période peut être réglée par logiciel, elle doit être définie par ce paramètre.

4.5.8 Dispositif à NTA fixe

Nom: `FPFNTA` ou `FB_PAR_FIXED_NTA`
Valeur: `FB_TRUE` ou `FB_FALSE`
Défaut: `FB_FALSE`

Ce paramètre indique, lorsqu'il vaut `FB_TRUE`, que le transfert de données se fait avec un dispositif FASTBUS qui n'incrémente pas le registre NTA (par exemple un dispositif ressemblant à un FIFO).

De tels dispositifs reçoivent ou émettent de multiples mots de données à une adresse interne unique. Le port qui échange des données avec un tel dispositif doit en tenir compte pour pouvoir réessayer une opération si une erreur se produit, ou lorsqu'il divise un bloc de données en plusieurs blocs de longueur plus faible.

Remarquer que toute opération vers un tel dispositif ne peut être correctement réessayée que si le dispositif possède en plus le registre de sauvegarde décrit à l'article 11.3 de la norme FASTBUS. (La présence de ce registre de sauvegarde est transparente pour le maître.)

If it has the value `FB_FALSE`, the Master may by any mechanism available to it break the transfer of the entire block into a number of transfers of sequential parts of the block (blocklets - see subclause 6.1.5).

The Operational Parameters `FB_PAR_HOLD_BUS`, `FB_PAR_HOLD_BUS_NO_AR`, `FB_PAR_HOLD_AS` and `FB_PAR_HOLD_ON_ERROR` defined below have precedence, overriding this parameter in determining whether or not the release of bus mastership occurs.

4.5.5 FASTBUS Block Transfer Blocklet Size

Name: `FPBLSZ` or `FB_PAR_BLOCKLET_SIZE`
Values: number of FASTBUS data cycles (DS transitions)
Default: Implementation dependent

This parameter shall specify the maximum blocklet size for subsequent FASTBUS Block Transfers.

4.5.6 Pipelined Transfer

Name: `FPPIPE` or `FB_PAR_PIPELINE`
Values: `FB_TRUE` or `FB_FALSE`
Default: `FB_FALSE`

If this parameter has the value `FB_TRUE`, any subsequent FASTBUS Block Transfers shall be executed in Pipelined Transfer mode ($MS=3$).

The interval between data signal timing transitions ($DS(u)$ to $DS(d)$) is taken from the parameter `FB_PAR_CLOCK` if the Implementation has access to it.

4.5.7 Pipelined Transfer Data Cycle Clock Time

Name: `FPCLK` or `FB_PAR_CLOCK`
Values: time between DS transitions in nanoseconds
Default: Implementation dependent

When data are transferred by a pipelined transfer, the transitions on the DS signal occur at a rate determined by the Master. If this rate is software settable, it shall be defined by this parameter.

4.5.8 Fixed NTA Device

Name: `FPFNTA` or `FB_PAR_FIXED_NTA`
Values: `FB_TRUE` or `FB_FALSE`
Default: `FB_FALSE`

This parameter, when in the `FB_TRUE` state, shall be used to indicate that the data transfer is from or to a FASTBUS Device that does not increment the NTA register ('FIFO like' devices, for example).

Such a Device accepts or sends multiple data words at a single Internal Address. The Port transferring data to or from such a Device must be aware of this in order to correctly retry the Operation if an error occurs, and when dividing a block transfer into several block transfers of shorter length.

Note that any Operation to such a device can only be correctly retried if, in addition, the device implements a protective buffer register as described in the FASTBUS standard clause 11.3. (The presence of the protective buffer is transparent to the Master.)

4.5.9 Transfert des données en mots courts

Nom: FPSDW ou FB_PAR_SHORT_DATA_WORD
Valeur: FB_TRUE ou FB_FALSE
Défaut: FB_FALSE

Lorsque ce paramètre vaut FB_FALSE, chaque mot de données transféré avec le tampon de données spécifié doit avoir 32 bits de large. Si ce paramètre vaut FB_TRUE, chaque mot de données doit avoir la largeur FB_PAR_SHORT_WORD_SIZE.

Les FB_PAR_SHORT_WORD_SIZE bits d'informations sont transférés sur les bits de poids faibles des lignes adresse/données du FASTBUS. Les autres bits des lignes AD sont mis à zéro par une écriture FASTBUS et sont ignorés dans une lecture FASTBUS. Ce paramètre ne doit s'appliquer qu'à un cycle de données FASTBUS et non à un cycle d'adressage primaire ou secondaire.

4.5.10 Longueur des mots courts

Nom: FPSWDS ou FB_PAR_SHORT_WORD_SIZE
Valeur: définie par la réalisation
Défaut: défini par la réalisation

Ce paramètre opérationnel, défini par la réalisation et en lecture seule, donne la dimension en bits des mots de données courts.

4.5.11 Contrôle de la génération de la parité

Nom: FPPRTY ou FB_PAR_PARITY
Valeur: FB_PARITY_ODD (FPPODD), produit une parité impaire
FB_PARITY_EVEN (FPPEVN), produit une parité paire
FB_PARITY_NONE (FPPNON), ne produit pas de parité
Défaut: défini par la réalisation (doit être impaire ou absente)

Ce paramètre doit être utilisé pour contrôler la génération de la parité par le port FASTBUS.

Remarquer que la norme FASTBUS demande une parité impaire ou absente - une parité paire ne devrait être émise que pour des raisons de diagnostic.

4.5.12 Mise en service de l'adressage géographique

Nom: FPEG ou FB_PAR_EG_UP
Valeur: FB_TRUE ou FB_FALSE
Défaut: FB_FALSE

Lorsque ce paramètre opérationnel vaut FB_TRUE, le port doit monter la ligne EG pour mettre en service la reconnaissance de l'adresse géographique quand il émet le cycle d'adressage primaire de l'opération demandée. Si ce port ne peut pas commander la ligne EG, ce paramètre ne doit pas être supporté.

4.5.13 Absence de cycle d'arbitrage

Nom: FPNOAR ou FB_PAR_NO_ARBITRATION
Valeur: FB_TRUE ou FB_FALSE
Défaut: FB_FALSE

Lorsque ce paramètre opérationnel vaut FB_TRUE, aucun cycle d'arbitrage ne doit être inclus dans la transaction FASTBUS demandée.

4.5.9 Transfer Data as Short Words

Name: FPSDW or FB_PAR_SHORT_DATA_WORD
Values: FB_TRUE or FB_FALSE
Default: FB_FALSE

If this parameter has the value FB_FALSE, each data word transferred to or from the specified Data Buffer shall be 32 bits wide. If this parameter is set to FB_TRUE, each data word shall have a width of FB_PAR_SHORT_WORD_SIZE.

FB_PAR_SHORT_WORD_SIZE bits are transferred over the low order bits of the FASTBUS Address-Data lines. The remaining bits on the AD lines are cleared by a FASTBUS write and ignored by a FASTBUS read. This parameter shall apply only to FASTBUS data cycles and not to primary or secondary address cycles.

4.5.10 Short Word Size

Name: FPSWDS or FB_PAR_SHORT_WORD_SIZE
Values: Implementation dependent
Default: Implementation dependent

This Read-only Operational Parameter is the Implementation dependent size, in bits, of a short data word.

4.5.11 Control of Parity Generation

Name: FPPRTY or FB_PAR_PARITY
Values: FB_PARITY_ODD (FPPODD), ODD parity is generated
 FB_PARITY_EVEN (FPPEVN), EVEN parity is generated
 FB_PARITY_NONE (FPPNON), NO parity is generated
Default: Implementation dependent (should be ODD or NONE)

This parameter shall be used to control parity generation by the FASTBUS port.

Note that the FASTBUS standard calls for odd or no parity - even parity should not be generated except for diagnostic reasons.

4.5.12 Enable Geographic Addressing

Name: FPEG or FB_PAR_EG_UP
Values: FB_TRUE or FB_FALSE
Default: FB_FALSE

If this Operational Parameter has the value FB_TRUE, the Port shall raise the EG line to enable Geographic address recognition when carrying out the Primary address cycle of the requested Operation. If the Port does not have control of the EG line this parameter shall be unsupported.

4.5.13 No Arbitration Cycle

Name: FPNOAR or FB_PAR_NO_ARBITRATION
Values: FB_TRUE or FB_FALSE
Default: FB_FALSE

If this Operational Parameter has the value FB_TRUE, no Arbitration cycle shall be included in the requested FASTBUS Transaction.

Ce paramètre est ignoré par les routines primitives des actions FASTBUS de la section 8.

4.5.14 Absence de cycle d'adressage primaire

Nom: FPNOPA ou FB_PAR_NO_PRIM_ADDR
Valeur: FB_TRUE ou FB_FALSE
Défaut: FB_FALSE

Lorsque ce paramètre opérationnel vaut FB_TRUE, aucun cycle d'adressage primaire ne doit être inclus dans l'opération FASTBUS demandée. Le paramètre d'adressage primaire sera ignoré.

Si une opération est à exécuter avec ce paramètre fixé à FB_TRUE, et qu'aucune connexion d'adresse n'a été établie antérieurement, une erreur doit être retournée et l'opération ne doit pas être exécutée.

Ce paramètre est ignoré par les routines primitives des actions FASTBUS de la section 8.

4.5.15 Absence de cycle d'adresse secondaire pour les opérations dans l'espace données

Nom: FPNOSA ou FB_PAR_NO_SEC_ADDR
Valeur: FB_TRUE ou FB_FALSE
Défaut: FB_FALSE

Ce paramètre opérationnel ne doit être utilisé que pour une opération dans l'espace données FASTBUS. Si ce paramètre a la valeur FB_TRUE, aucun cycle d'adresse secondaire ne doit être exécuté dans l'opération FASTBUS. Le paramètre d'adresse secondaire doit être ignoré. Si ce paramètre a la valeur FB_FALSE, le cycle d'adresse secondaire sera exécuté.

Ce paramètre est ignoré par les routines primitives des actions FASTBUS de la section 8.

4.5.16 Absence de cycle d'adresse secondaire pour les opérations dans l'espace CSR

Nom: FPNOSC ou FB_PAR_NO_SEC_ADDR_CSR
Valeur: FB_TRUE ou FB_FALSE
Défaut: FB_FALSE

Ce paramètre opérationnel ne doit être utilisé que pour une opération dans l'espace CSR FASTBUS. Si ce paramètre a la valeur FB_TRUE, aucun cycle d'adresse secondaire ne doit être exécuté dans l'opération FASTBUS. Le paramètre d'adresse secondaire doit être ignoré. Si ce paramètre a la valeur FB_FALSE, le cycle d'adresse secondaire sera exécuté.

Remarquer que la norme FASTBUS demande, dans la plupart des cas, qu'une opération FASTBUS dans l'espace CSR inclut un cycle d'adresse secondaire. Ce paramètre est ignoré par les routines primitives des actions FASTBUS de la section 8.

4.5.17 Absence de cycle de données

Nom: FPNODA ou FB_PAR_NO_DATA_CYCLE
Valeur: FB_TRUE ou FB_FALSE
Défaut: FB_FALSE

Si ce paramètre a la valeur FB_TRUE, aucun cycle de données ne doit être exécuté dans l'opération FASTBUS demandée.

This parameter is ignored by the Primitive FASTBUS Action Routines in section 8.

4.5.14 No Primary Address Cycle

Name: FPNOPA or FB_PAR_NO_PRIM_ADDR
Values: FB_TRUE or FB_FALSE
Default: FB_FALSE

If this Operational Parameter has the value FB_TRUE, no primary address cycle shall be included in the requested FASTBUS Operation. The primary address parameter shall be ignored.

If the Operation to be executed has this parameter set to FB_TRUE, and no address connection has been established previously, an error shall be returned and the Operation shall not be executed.

This parameter is ignored by the Primitive FASTBUS Action Routines in section 8.

4.5.15 No Secondary Address Cycle for Data Space Operations

Name: FPNOSA or FB_PAR_NO_SEC_ADDR
Values: FB_TRUE or FB_FALSE
Default: FB_FALSE

This Operational Parameter shall act only on FASTBUS Data Space Operations. If it has the value FB_TRUE, no secondary address cycle shall be performed as part of the FASTBUS Operation. The secondary address parameter shall be ignored. If it has the value FB_FALSE, the secondary address cycle shall be carried out.

This parameter is ignored by the Primitive FASTBUS Action Routines in section 8.

4.5.16 No Secondary Address Cycle for CSR Space Operations

Name: FPNOSC or FB_PAR_NO_SEC_ADDR_CSR
Values: FB_TRUE or FB_FALSE
Default: FB_FALSE

This Operational Parameter shall act only on FASTBUS CSR Space Operations. If it has the value FB_TRUE, no secondary address cycle shall be performed as part of the FASTBUS Operation. The secondary address parameter shall be ignored. If it has the value FB_FALSE, the secondary address cycle shall be carried out.

Note that the FASTBUS standard requires, in almost all cases, that Operations in CSR space include a secondary address cycle. This parameter is ignored by the Primitive FASTBUS Action Routines in section 8.

4.5.17 No Data Cycle

Name: FPNODA or FB_PAR_NO_DATA_CYCLE
Values: FB_TRUE or FB_FALSE
Default: FB_FALSE

If this parameter has the value FB_TRUE, no data cycle shall be executed as part of the requested FASTBUS Operation.

Ce paramètre est ignoré par les routines primitives des actions FASTBUS de la section 8.

4.5.18 Conservation de la maîtrise du bus

Nom: FPHOLD ou FB_PAR_HOLD_BUS
Valeur: FB_TRUE ou FB_FALSE
Défaut: FB_FALSE

Si ce paramètre a la valeur FB_TRUE, la maîtrise du FASTBUS doit être maintenue à la fin de l'opération spécifiée, c.-à-d. que GK est maintenu positionné. Autrement, la maîtrise du bus doit être relâchée au plus tard à la fin de l'opération, sauf si une erreur FASTBUS se produit et que le paramètre FB_PAR_HOLD_BUS_ON_ERROR a la valeur FB_TRUE.

Ce paramètre est ignoré par les routines primitives des actions FASTBUS de la section 8.

4.5.19 Conservation de la maîtrise du bus si AR n'est pas positionné

Nom: FPHOAR ou FB_PAR_HOLD_BUS_NO_AR
Valeur: FB_TRUE ou FB_FALSE
Défaut: FB_FALSE

Si ce paramètre a la valeur FB_TRUE, la maîtrise du FASTBUS doit être maintenue à la fin de l'opération spécifiée si l'on ne détecte pas AR positionné sur le bus. Aussitôt que l'on détecte que AR est positionné, le bus doit être libéré. Autrement la maîtrise du bus doit être relâchée au plus tard à la fin de l'opération sauf si le paramètre opérationnel FB_PAR_HOLD_BUS a la valeur FB_TRUE, ou qu'une erreur FASTBUS se produit et que le paramètre FB_PAR_HOLD_BUS_ON_ERROR a la valeur FB_TRUE.

Ce paramètre est ignoré par les routines primitives des actions FASTBUS de la section 8.

4.5.20 Maintien de la connexion d'adresse

Nom: FPHOAS ou FB_PAR_HOLD_AS
Valeur: FB_TRUE ou FB_FALSE
Défaut: FB_FALSE

Si ce paramètre a la valeur FB_TRUE la connexion d'adresse FASTBUS doit être maintenue à la fin de l'opération, c.-à-d. que AS est maintenu positionné. Autrement la connexion d'adresse doit être relâchée à la fin de l'opération sauf si une erreur fatale FASTBUS se produit et que l'on a choisi l'option FB_PAR_HOLD_ON_ERROR.

Ce paramètre est ignoré par les routines primitives des actions FASTBUS de la section 8.

4.5.21 Maintien des signaux du maître après une erreur

Nom: FPHOER ou FB_PAR_HOLD_BUS_ON_ERROR
Valeur: FB_TRUE ou FB_FALSE
Défaut: FB_FALSE

Si ce paramètre a la valeur FB_TRUE et qu'une opération se termine par suite d'une erreur FASTBUS, le port doit maintenir tous les signaux positionnés à ce moment. Chaque signal doit être maintenu jusqu'à ce qu'il soit explicitement libéré.

This parameter is ignored by the Primitive FASTBUS Action Routines in section 8.

4.5.18 Hold Bus Mastership

Name: FPHOLD or FB_PAR_HOLD_BUS
Values: FB_TRUE or FB_FALSE
Default: FB_FALSE

If this parameter has the value FB_TRUE, FASTBUS mastership shall be maintained at the completion of the specified Operation, i.e. GK is held asserted. Otherwise mastership shall be released at or before Operation completion, unless a FASTBUS error occurs and the FB_PAR_HOLD_BUS_ON_ERROR parameter has the value FB_TRUE.

This parameter is ignored by the Primitive FASTBUS Action Routines in section 8.

4.5.19 Hold Bus Mastership if AR not Asserted

Name: FPHOAR or FB_PAR_HOLD_BUS_NO_AR
Values: FB_TRUE or FB_FALSE
Default: FB_FALSE

If this parameter has the value FB_TRUE, FASTBUS mastership shall be maintained at the completion of the specified Operation if AR is not detected as asserted on the bus. As soon as AR is detected as asserted, the bus is released. Otherwise mastership shall be released at or before Operation completion, unless the Operational Parameter FB_PAR_HOLD_BUS is FB_TRUE, or a FASTBUS error occurs and the FB_PAR_HOLD_BUS_ON_ERROR parameter has the value FB_TRUE.

This parameter is ignored by the Primitive FASTBUS Action Routines in section 8.

4.5.20 Hold Address Connection

Name: FPHOAS or FB_PAR_HOLD_AS
Values: FB_TRUE or FB_FALSE
Default: FB_FALSE

If this parameter has the value FB_TRUE, any FASTBUS address connection shall be maintained at the end of the Operation, i.e. AS is held asserted. Otherwise the address connection shall be released at Operation completion, unless a fatal FASTBUS error occurs and the FB_PAR_HOLD_ON_ERROR option is selected.

This parameter is ignored by the Primitive FASTBUS Action Routines in section 8.

4.5.21 Hold Master Signals on Error

Name: FPHOER or FB_PAR_HOLD_BUS_ON_ERROR
Values: FB_TRUE or FB_FALSE
Default: FB_FALSE

If this parameter has the value FB_TRUE, and an Operation terminates because of a FASTBUS error, the Port shall maintain any currently asserted signals. The signals shall be maintained until explicitly released.

4.5.22 Durée du temporisateur long

Nom: FPLOT ou FB_PAR_LONG_TIMER
Valeur: temps de temporisation en nanosecondes
Défaut: défini par la réalisation

Si la durée du temporisateur long peut être fixée par programme, elle doit être définie par ce paramètre. Si elle peut être lue, ce paramètre devrait contenir sa valeur. Si un déclenchement du temporisateur se produit, le code d'erreur approprié doit être renvoyé.

La valeur du temporisateur long sera également écrite dans le CSR#1C du port (s'il existe).

Le temporisateur long est démarré au début d'une action ou d'une liste (sauf s'il est déjà en route à cause du paramètre FB_PAR_LONG_TIMER_ON qui est FB_TRUE), ou lorsque sa valeur est fixée par le paramètre FB_PAR_LONG_TIMER. Il est arrêté et remis à zéro à la fin d'une action ou d'une liste, sauf si le paramètre opérationnel FB_PAR_LONG_TIMER_ON est FB_TRUE (voir paragraphe 4.5.24).

Remarquer que le temporisateur long contrôle la totalité de la transaction FASTBUS et est ainsi fonctionnellement équivalent aux temporisateurs d'arbitrage et de GK.

4.5.23 Mise hors service du temporisateur long

Nom: FPDLOT ou FB_PAR_DISABLE_LONG_TIMER
Valeur: FB_TRUE ou FB_FALSE
Défaut: FB_FALSE

Ce paramètre doit être mis à FB_TRUE pour mettre hors service le temporisateur long (dans CSR#9 < 04>).

4.5.24 Maintien du temporisateur long

Nom: FPOLOT ou FB_PAR_LONG_TIMER_ON
Valeur: FB_TRUE ou FB_FALSE
Défaut: FB_FALSE

Ce paramètre doit être mis à FB_TRUE pour supprimer la remise à zéro du temporisateur à la fin de l'exécution d'une action ou d'une liste et le redémarrage du temporisateur au début de l'exécution d'une action ou d'une liste. (C.-à-d. que le temporisateur continue à tourner.)

4.5.25 Durée du temporisateur d'attente

Nom: FPWTT ou FB_PAR_WT_TIMER
Valeur: temps de temporisation en nanosecondes
Défaut: défini par la réalisation

Si la durée du temporisateur d'attente peut être fixée par programme, elle doit être définie par ce paramètre. Si elle peut être lue, ce paramètre devrait contenir sa valeur. Ce temporisateur est démarré lorsque le maître détecte que le signal WT du FASTBUS est positionné, et est remis à zéro et inhibé lorsque WT n'est pas positionné. Si un déclenchement du temporisateur se produit, le code d'erreur approprié doit être renvoyé.

La valeur du temporisateur d'attente sera également écrite dans le CSR#1D du port (s'il existe).

4.5.22 Long Timer

Name: FPLOT or FB_PAR_LONG_TIMER
Values: timeout time in nanoseconds
Default: Implementation dependent

If the long timer period is software settable, it shall be specified by this parameter. If it is readable this parameter should contain its value. If the timeout occurs the appropriate Error Code shall be returned.

The Long Timer value will also be written to CSR#1C of the Port (if it is implemented).

The Long Timer is started at the beginning of an Action or a List (unless it is already running due to the parameter FB_PAR_LONG_TIMER_ON being FB_TRUE), or when its value is set using the parameter FB_PAR_LONG_TIMER. It is stopped and reset at the end of an Action or List, unless the Operational Parameter FB_PAR_LONG_TIMER_ON is FB_TRUE (see subclause 4.5.24).

Note that the Long Timer acts over complete FASTBUS Transactions and thus provides the functionality of Arbitration or GK timers.

4.5.23 Disable Long Timer

Name: FPDLOT or FB_PAR_DISABLE_LONG_TIMER
Values: FB_FALSE or FB_TRUE
Default: FB_FALSE

This parameter shall be set to FB_TRUE to disable (in CSR#9<04>) the long timer timeout.

4.5.24 Maintain Long Timer

Name: FPLOT or FB_PAR_LONG_TIMER_ON
Values: FB_FALSE or FB_TRUE
Default: FB_FALSE

This parameter shall be set to FB_TRUE to suppress the resetting of the timer at the end of an Action or List execution and the restarting of the timer at the beginning of an Action or List execution. (I.e. the timer continues ticking.)

4.5.25 Wait Timer

Name: FPWTT or FB_PAR_WT_TIMER
Values: timeout time in nanoseconds
Default: Implementation dependent

If the wait timer period is software settable, it shall be specified by this parameter. If it is readable, it should be returned by this parameter. This timer is started when the master detects the FASTBUS WT signal asserted, and is reset and inhibited when WT is not asserted. If this timeout occurs, the appropriate Error Code shall be returned.

The wait timer value will also be written to CSR#1D of the Port (if it is implemented).

4.5.26 Mise hors service du temporisateur d'attente

Nom: FPDWTT ou FB_PAR_DISABLE_WT_TIMER
Valeur: FB_TRUE ou FB_FALSE
Défaut: FB_FALSE

Ce paramètre doit être mis à FB_TRUE pour mettre hors service le temporisateur d'attente (dans CSR#9 <05>).

4.5.27 Durée du temporisateur d'adresse

Nom: FPAKT ou FB_PAR_AK_TIMER
Valeur: temps de temporisation en nanosecondes
Défaut: défini par la réalisation

Si la durée du temporisateur d'adresse peut être fixée par programme, elle doit être définie par ce paramètre. Si elle peut être lue, ce paramètre devrait contenir sa valeur. Chaque fois que le port produit une transition sur AS (AS(u) ou AS(d)), ce temporisateur est démarré pour attendre la transition correspondante de AK. Si un déclenchement du temporisateur se produit, le code d'erreur approprié doit être renvoyé.

La valeur du temporisateur d'adresse sera également écrite dans le CSR#1E du Port (s'il existe).

4.5.28 Mise hors service du temporisateur d'adresse

Nom: FPDAKT ou FB_PAR_DISABLE_AK_TIMER
Valeur: FB_TRUE ou FB_FALSE
Défaut: FB_FALSE

Ce paramètre doit être mis à FB_TRUE pour mettre hors service le temporisateur d'adresse (dans CSR#9 <06>).

4.5.29 Durée du temporisateur de données

Nom: FPDKT ou FB_PAR_DK_TIMER
Valeur: temps de temporisation en nanosecondes
Défaut: défini par la réalisation

Si la durée du temporisateur de données peut être fixée par programme, elle doit être définie par ce paramètre. Si elle peut être lue, ce paramètre devrait contenir sa valeur. Chaque fois que le port produit une transition sur DS, le temporisateur est démarré pour attendre la transition correspondante de DK. Si un déclenchement du temporisateur se produit, le code d'erreur approprié doit être renvoyé.

La valeur du temporisateur de données sera également écrite dans le CSR#1F du Port (s'il existe).

4.5.30 Mise hors service du temporisateur de données

Nom: FPDDKT ou FB_PAR_DISABLE_DK_TIMER
Valeur: FB_TRUE ou FB_FALSE
Défaut: FB_FALSE

Ce paramètre doit être mis à FB_TRUE pour mettre hors service le temporisateur de données (dans CSR#9 <07>).

4.5.26 Disable Wait Timer

Name: FPDWTT or FB_PAR_DISABLE_WT_TIMER
Values: FB_FALSE or FB_TRUE
Default: FB_FALSE

This parameter shall be set to FB_TRUE to disable (in CSR#9<05>) the Wait timer timeout.

4.5.27 Address Timer

Name: FPAKT or FB_PAR_AK_TIMER
Values: timeout time in nanoseconds
Default: Implementation dependent

If the address timer period is software settable, it shall be specified by this parameter. If it is readable, it should be returned by this parameter. Whenever the Port generates a transition on AS (AS(u) or AS(d)), a timer is started for the receipt of the responding AK transition. If a timeout occurs the appropriate Error Code shall be returned.

The address timer value will also be written to CSR#1E of the Port (if it is implemented).

4.5.28 Disable Address Timer

Name: FPDAKT or FB_PAR_DISABLE_AK_TIMER
Values: FB_FALSE or FB_TRUE
Default: FB_FALSE

This parameter shall be set to FB_TRUE to disable (in CSR#9<06>) the address timer timeout.

4.5.29 Data Timer

Name: FPDKT or FB_PAR_DK_TIMER
Values: timeout time in nanoseconds
Default: Implementation dependent

If the data timer period is software settable, it shall be specified by this parameter. If it is readable, it should be returned by this parameter. Whenever the Port generates a transition on DS, a timer is started for the receipt of the responding DK transition. If a timeout occurs the appropriate Error Code shall be returned.

The data timer value will also be written to CSR#1F of the Port (if it is implemented).

4.5.30 Disable Data Timer

Name: FPDDKT or FB_PAR_DISABLE_DK_TIMER
Values: FB_FALSE or FB_TRUE
Default: FB_FALSE

This parameter shall be set to FB_TRUE to disable (in CSR#9<07>) the data timer timeout.

4.5.31 Durée du temporisateur logiciel

Nom: FPSOFT ou FB_PAR_SOFT_TIMER
Valeur: temps de temporisation en millisecondes
Défaut: défini par la réalisation

Ce paramètre fournit une définition uniforme pour un ou plusieurs temporisateurs logiciels qui seront présents dans chaque réalisation. Il fournit un moyen pour renvoyer le contrôle à la routine d'appel lorsque l'exécution d'une action ou d'une liste est trop longue.

Il n'est pas mis hors service à travers le réseau FASTBUS en positionnant un bit de CSR#9.

L'expiration du temporisateur logiciel ne doit produire aucune modification dans l'état du FASTBUS.

Le temporisateur logiciel est démarré au début d'une action ou d'une liste (sauf s'il est déjà en route à cause du paramètre FB_PAR_SOFT_TIMER_ON qui est FB_TRUE), ou lorsque sa valeur est fixée par le paramètre FB_PAR_SOFT_TIMER. Il est arrêté et remis à zéro à la fin d'une action ou d'une liste, sauf si le paramètre opérationnel FB_PAR_SOFT_TIMER_ON est FB_TRUE (voir paragraphe 4.5.33).

4.5.32 Mise hors service du temporisateur logiciel

Nom: FPDSOT ou FB_PAR_DISABLE_SOFT_TIMER
Valeur: FB_TRUE ou FB_FALSE
Défaut: FB_FALSE

Ce paramètre doit être mis à FB_TRUE pour mettre hors service le temporisateur logiciel.

4.5.33 Maintien du temporisateur logiciel

Nom: FPOSOT ou FB_PAR_SOFT_TIMER_ON
Valeur: FB_TRUE ou FB_FALSE
Défaut: FB_FALSE

Ce paramètre doit être mis à FB_TRUE pour supprimer la remise à zéro du temporisateur à la fin de l'exécution d'une action ou d'une liste et le redémarrage du temporisateur au début de l'exécution d'une action ou d'une liste.

4.5.34 Nombre de réessais

Nom: FPRTRY ou FB_PAR_NUM_RETRY
Valeur: nombre de réessais
Défaut: 0

Ce paramètre doit définir le nombre de fois où un cycle FASTBUS doit être réessayé, lorsqu'il provoque un code d'erreur avec une sévérité différente de FB_SEV_SUCCESS, et que le code de réponse associé spécifie que l'on devrait effectuer des réessais (voir article 11.6).

Remarquer que toute action peut être réessayée au niveau de l'application de l'utilisateur si l'on désire un plus grand nombre de réessais.

4.5.35 Délai de réessai

Nom: FPRTRT ou FB_PAR_ADDRESS_CYCLE_RETRY_TIME
Valeur: délai moyen en nanosecondes
Défaut: défini par la réalisation

4.5.31 Software Timer

Name: FPSOFT or FB_PAR_SOFT_TIMER
Value: timeout time in milliseconds
Default: implementation dependent

This parameter provides a uniform definition for one or several software timers that will be present in any implementation. It provides a means of allowing control to return to the calling routine when an Action or List takes too long to execute.

It is not disabled through the FASTBUS network by setting a bit in CSR#9.

Expiration of the Software Timer shall not cause any change in state of the FASTBUS.

The Software Timer is started at the beginning of an Action or a List (unless it is already running due to the parameter FB_PAR_SOFT_TIMER_ON being FB_TRUE), or when its value is set using the parameter FB_PAR_SOFT_TIMER. It is stopped and reset at the end of an Action or List, unless the Operational Parameter FB_PAR_SOFT_TIMER_ON is FB_TRUE (see subclause 4.5.33).

4.5.32 Disable Software Timer

Name: FPDSOT or FB_PAR_DISABLE_SOFT_TIMER
Values: FB_FALSE or FB_TRUE
Default: FB_FALSE

This parameter shall be set to FB_TRUE to disable the timeout of the software timer.

4.5.33 Maintain Software Timer

Name: FPOSOT or FB_PAR_SOFT_TIMER_ON
Values: FB_FALSE or FB_TRUE
Default: FB_FALSE

This parameter shall be set to FB_TRUE to suppress the resetting of the timer at the end of an Action or List execution and the restarting of the timer at the beginning of an Action or List execution.

4.5.34 Number of Retries

Name: FPRTRY or FB_PAR_NUM_RETRY
Values: number of retries
Default: 0

This parameter shall specify the number of times a FASTBUS cycle leading to an Error Code with a Severity not equal to FB_SEV_SUCCESS shall be retried if the associated Response Code (see clause 11.6) specifies that retries should be attempted.

Note that all actions can be retried at the User application level if more retries are desired.

4.5.35 Retry Time

Name: FPRTRT or FB_PAR_ADDRESS_CYCLE_RETRY_TIME
Values: average delay time in nanoseconds
Default: Implementation dependent

Lorsqu'un cycle FASTBUS d'adresse secondaire ou de données est réessayé, l'essai doit être fait aussi rapidement que possible. Cependant, lorsqu'un cycle d'adressage primaire est réessayé, il est quelquefois nécessaire d'attendre un certain temps entre chaque essai. Si le code de réponse était $SS=1$ (réseau occupé), le maître ne devrait pas réessayer le cycle d'adressage immédiatement, pour éviter un blocage potentiel du système. Ce paramètre est utilisé pour fixer la valeur moyenne du délai aléatoire entre les réessais.

Withdrawing
IEC NORM.COM: Click to view the full PDF of IEC 61052:1991

When a FASTBUS data or secondary address cycle is retried, the retry shall be done as fast as possible. However, when a Primary address Cycle is retried it is sometimes necessary to wait a finite length of time between each attempt. If the SS Code was SS=1 (Network Busy) the Master should not retry the address cycle immediately, to avoid potential system deadlock. This parameter is used to define an average value for the random delay that should occur between retries.

Withdrawn
IECNORM.COM: Click to view the full PDF of IEC 61052:1991

Section 5. Tampons de données

5.1 Paramètres d'accès à un tampon

Pour toutes les routines d'action qui produisent un transfert de données avec un "tampon de données", une paire d'arguments définit le mode d'accès et l'emplacement du tampon. Le premier de ces arguments ("buffer_mode") définit l'interprétation du second argument. Ce second argument ("buffer_specifier") peut être l'identificateur d'un tampon séquentiel, l'identificateur d'une variable ou d'un tableau, ou la valeur de la donnée elle-même. Les modes de tampon autorisés, définis à la section 2, sont décrits ci-dessous.

5.1.1 Tampons définis comme une variable ou un tableau

Ce mode de tampon doit être disponible pour le mode d'exécution immédiate. Il n'est pas obligatoire pour le mode d'exécution différée.

Lorsque le paramètre buffer_mode a la valeur FB_BUFFER_VAR, le buffer_specifier doit être une variable. Au moment de l'exécution, les données doivent être écrites ou lues dans la variable spécifiée.

5.1.2 Tampons définis comme une valeur

Ce mode de tampon ne doit être permis que pour les opérations d'écriture d'un mot unique et doit être disponible pour les environnements en mode d'exécution différée. Il n'est pas obligatoire pour les environnements en mode d'exécution immédiate.

Lorsque le paramètre buffer_mode a la valeur FB_BUFFER_VALUE, le buffer_specifier doit être la valeur du mot de donnée unique. Au moment de l'exécution, on écrit la valeur de ce mot au moment de l'appel.

5.1.3 Tampons définis par l'identificateur d'un tampon séquentiel

Ce mode de tampon doit être disponible pour le mode d'exécution différée. Il n'est pas obligatoire pour le mode d'exécution immédiate.

Lorsque le paramètre buffer_mode a la valeur FB_BUFFER_SEQ, des actions successives FASTBUS avec les mêmes buffer_specifiers (identificateurs de tampon séquentiel) écrivent ou lisent en séquence dans le tampon séquentiel correspondant. Les routines optionnelles FB_READ_BUFFER_POINTER, FB_WRITE_BUFFER_POINTER, FB_DISPLACE_BUFFER_POINTER définies dans cette section donnent accès au pointeur courant dans le tampon séquentiel de données.

Un tampon séquentiel peut être déclaré du type interne ou externe.

Un "tampon interne" doit être accessible directement par le programme appelant. (Comme cette norme ne fournit aucun mécanisme pour partager des paramètres entre les environnements, les tampons internes de données ne doivent être utilisés que par un seul environnement à la fois.)

Les "tampons externes" de données peuvent ne pas être directement accessibles au programme les utilisant et leur disponibilité est optionnelle.

La valeur de buffer_id est fournie par l'utilisateur et doit être unique dans un environnement. Un "tampon séquentiel" doit être déclaré en utilisant une des routines décrites ci-dessous avant l'exécution de l'environnement qui s'y réfère.

Section 5. Data Buffers

5.1 Buffer Access Arguments

For each Action Routine that results in the transfer of data to or from a "Data Buffer", a pair of arguments specifies the access mode and location of the buffer. The first argument of the pair ("buffer_mode") specifies how the second argument is to be interpreted. The second argument of the pair ("buffer_specifier") is either a Sequential Buffer identifier, a variable or array identifier, or the data value itself. The allowed Buffer Modes, defined in section 2, are explained below.

5.1.1 Buffer specified as a Variable or Array

Support for this Buffer Mode shall be provided for Immediate Execution Mode. It is not mandatory for Delayed Execution Mode.

When the buffer_mode parameter has the value FB_BUFFER_VAR, the buffer_specifier shall be a variable. The data shall be written from or read to the specified variable at Execution Time.

5.1.2 Buffer specified as a Value

This Buffer Mode shall be allowed only for single word write Operations and shall be provided for Delayed Execution Mode Environments. It is not mandatory for Immediate Execution Mode Environments.

When the buffer_mode has the value FB_BUFFER_VALUE, the buffer_specifier shall be a single word data value, whose value at Call Time is written at Execution Time.

5.1.3 Buffer specified as a Sequential Buffer ID

Support for this Buffer Mode shall be provided for Delayed Execution Mode. It is not mandatory for Immediate Execution Mode.

When the Buffer Mode is set to FB_BUFFER_SEQ, successive FASTBUS Actions with the same buffer_specifiers (sequential buffer ids) write to or read from the same Sequential Buffer in sequence. The optional routines FB_READ_BUFFER_POINTER, FB_WRITE_BUFFER_POINTER, FB_DISPLACE_BUFFER_POINTER defined in this section provide access to a sequential buffer's current data location pointer.

A sequential buffer may be declared to be of type Internal or External.

An "Internal Buffer" shall be directly accessible to the calling program. (Since this standard provides no mechanism for accessing parameters between Environments, internal Data Buffers shall be referenced by only one Environment at a time.)

Support for "External Buffers", where the Data Buffer may not be directly accessible to the program using it, is optional.

The value of the buffer_id is provided by the User and shall be unique within the Environment. A "Sequential Buffer" shall be declared using one of the routines described below before execution of any operations that references it.

5.2 Routines des tampons séquentiels de données

5.2.1 Déclaration d'un tampon interne

Nom: FBDSEQ ou FB_DECLARE_SEQ_BUFFER

Catégorie: B

Paramètres:

return_code	FB_error_code	(OUT)
environment_id	FB_environment_id	(IN)
seq_buffer_id	FB_seq_buffer_id	(IN)
buffer_array	FB_word_array	(IN)
buffer_size	FB_integer	(IN)

Effets:

Cette routine doit associer l'identificateur spécifié du tampon séquentiel (*seq_buffer_id*) au tampon interne spécifié (*buffer_array*) contenant un nombre d'octets égal à *buffer_size*. Le pointeur du tampon est réinitialisé. Le tampon interne est local et directement accessible par le programme d'appel. Les actions FASTBUS, dont le *buffer_mode* est FB_BUFFER_SEQ et dont les *buffer_specifier* ont la même valeur *seq_buffer_id*, doivent utiliser le même tampon interne.

5.2.2 Déclaration d'un tampon externe

Nom: FBDEXT ou FB_DECLARE_EXT_SEQ_BUFFER

Catégorie: E

Paramètres:

return_code	FB_error_code	(OUT)
environment_id	FB_environment_id	(IN)
seq_buffer_id	FB_seq_buffer_id	(IN)
buffer_location	FB_word_array	(IN)
buffer_size	FB_integer	(IN)

Effets:

Cette routine doit associer l'identificateur spécifié du tampon séquentiel au tampon externe spécifié caractérisé par l'argument *buffer_location* contenant un nombre d'octets égal à *buffer_size*. Ce tampon est externe et peut ne pas être directement accessible par l'utilisateur (par exemple il peut être situé dans le port). Toutes les opérations FASTBUS, dont le *buffer_mode* est FB_BUFFER_SEQ et dont les *buffer_specifier* ont la même valeur *seq_buffer_id*, doivent utiliser le même tampon externe. Les paramètres *buffer_location* et *buffer_size* dépendent de la réalisation.

5.2.3 Libération d'un tampon séquentiel interne de données

Nom: FBREL ou FB_RELEASE_SEQ_BUFFER

Catégorie: B

Paramètres:

return_code	FB_error_code	(OUT)
environment_id	FB_environment_id	(IN)
seq_buffer_id	FB_seq_buffer_id	(IN)

5.2 Sequential Data Buffer Routines

5.2.1 Declare Internal Buffer

Name: FBDSEQ or FB_DECLARE_SEQ_BUFFER

Category: B

Parameters:

return_code	FB_error_code	(OUT)
environment_id	FB_environment_id	(IN)
seq_buffer_id	FB_seq_buffer_id	(IN)
buffer_array	FB_word_array	(IN)
buffer_size	FB_integer	(IN)

Effect:

This routine shall associate the specified Sequential Buffer Identifier (seq_buffer_id) with the specified Internal Buffer (buffer_array) containing buffer_size bytes. The buffer pointer is reset. The Internal Buffer is local to and directly accessible by the calling program. FASTBUS Actions with their buffer_mode set to FB_BUFFER_SEQ and buffer_specifier set to the same value as seq_buffer_id shall use the same Internal Buffer.

5.2.2 Declare External Buffer

Name: FBDEXT or FB_DECLARE_EXT_SEQ_BUFFER

Category: E

Parameters:

return_code	FB_error_code	(OUT)
environment_id	FB_environment_id	(IN)
seq_buffer_id	FB_seq_buffer_id	(IN)
buffer_location	FB_word_array	(IN)
buffer_size	FB_integer	(IN)

Effect:

This routine shall associate the specified Sequential Buffer Identifier with the specified External Buffer, characterized by the argument buffer_location, of size buffer_size bytes. The buffer is external to and may not be directly accessible by the User (it may be located in the Port for example). Any FASTBUS Operations with their buffer_mode set to FB_BUFFER_SEQ and buffer_specifier set to the same value as seq_buffer_id shall use the same External Buffer. The buffer_location and buffer_size parameters shall be Implementation dependent.

5.2.3 Release Internal Sequential Data Buffer

Name: FBREL or FB_RELEASE_SEQ_BUFFER

Category: B

Parameters:

return_code	FB_error_code	(OUT)
environment_id	FB_environment_id	(IN)
seq_buffer_id	FB_seq_buffer_id	(IN)

Effets:

Cette routine doit libérer l'identificateur du tampon spécifié pour une utilisation ultérieure. Sa valeur devient invalide jusqu'à ce qu'elle soit redéclarée comme un tampon interne ou externe comme ci-dessus.

5.2.4 Lecture du pointeur d'un tampon séquentiel

Nom: FBRPTR ou FB_READ_BUFFER_POINTER

Catégorie: E, Routine d'action

Paramètres:

return_code	FB_error_code	(OUT)	
environment_id	FB_environment_id	(IN)	
seq_buffer_id	FB_seq_buffer_id	(IN)	
buffer_mode	FB_buffer_mode	(IN)	
buffer_specifier	FB_word	(OUT)	(FBVAR, FBVAL)
	ou FB_seq_buffer_id	(IN)	(FBSEQ)

Action:

Cette routine doit copier le pointeur du tampon séquentiel courant associé à seq_buffer_id dans le tampon de données spécifié par les paramètres d'accès au tampon, buffer_mode et buffer_specifier. La valeur du pointeur du tampon séquentiel doit être donnée, en octets, par rapport au début du tampon séquentiel. Lorsqu'il pointe sur la première position du tampon séquentiel, sa valeur doit être zéro.

5.2.5 Ecriture du pointeur d'un tampon séquentiel

Nom: FBWPTR ou FB_WRITE_BUFFER_POINTER

Catégorie: E, Routine d'action

Paramètres:

return_code	FB_error_code	(OUT)	
environment_id	FB_environment_id	(IN)	
seq_buffer_id	FB_seq_buffer_id	(IN)	
buffer_mode	FB_buffer_mode	(IN)	
buffer_specifier	FB_word	(IN)	(FBVAR, FBVAL)
	ou FB_seq_buffer_id	(IN)	(FBSEQ)

Action:

La valeur contenue dans le tampon de données, spécifié par les paramètres d'accès buffer_mode et buffer_specifier, est mise dans le pointeur de données du tampon séquentiel spécifié. La valeur indiquée doit être en octets par rapport au début du tampon séquentiel. Lorsque la valeur est égale à zéro, le pointeur doit être placé sur la première position du tampon séquentiel.

5.2.6 Déplacement du pointeur d'un tampon séquentiel

Nom: FBDPTR ou FB_DISPLACE_BUFFER_POINTER

Catégorie: E, Routine d'action

Effect:

This routine shall release the specified buffer identifier for future use. Its value becomes invalid until it has been redeclared as an internal or external buffer as above.

5.2.4 Read Sequential Buffer Pointer

Name: FBRPTR or FB_READ_BUFFER_POINTER

Category: E, Action Routine

Parameters:

return_code	FB_error_code	(OUT)	
environment_id	FB_environment_id	(IN)	
seq_buffer_id	FB_seq_buffer_id	(IN)	
buffer_mode	FB_buffer_mode	(IN)	
buffer_specifier	FB_word	(OUT)	(FBVAR, FBVAL)
	or FB_seq_buffer_id	(IN)	(FBSEQ)

Action:

This routine shall copy the current sequential buffer pointer associated with seq_buffer_id to the Buffer specified by the buffer access parameters buffer_mode and buffer_specifier. The value of the sequential buffer pointer shall be in bytes relative to the beginning of the Sequential Buffer. When pointing at the first location of the Sequential Buffer its value shall be zero.

5.2.5 Write Sequential Buffer Pointer

Name: FBWPTR or FB_WRITE_BUFFER_POINTER

Category: E, Action Routine

Parameters:

return_code	FB_error_code	(OUT)	
environment_id	FB_environment_id	(IN)	
seq_buffer_id	FB_seq_buffer_id	(IN)	
buffer_mode	FB_buffer_mode	(IN)	
buffer_specifier	FB_word	(IN)	(FBVAR, FBVAL)
	or FB_seq_buffer_id	(IN)	(FBSEQ)

Action:

The pointer to the data in the specified Sequential Buffer is set to the value contained in the Data Buffer specified by the buffer access parameters buffer_mode and buffer_specifier. The value specified shall be in bytes relative to the beginning of the Sequential Buffer. When pointing at the first location of the Sequential Buffer its value shall be zero.

5.2.6 Displace Sequential Buffer Pointer

Name: FBDPTR or FB_DISPLACE_BUFFER_POINTER

Category: E, Action Routine

Paramètres:

return_code	FB_error_code	(OUT)
environment_id	FB_environment_id	(IN)
seq_buffer_id	FB_seq_buffer_id	(IN)
buffer_mode	FB_buffer_mode	(IN)
buffer_specifier	FB_word	(IN) (FBVAR, FBVAL)
	ou FB_seq_buffer_id	(IN) (FBSEQ)

Action:

Au moment de l'exécution, cette routine doit déplacer le pointeur du tampon séquentiel courant associé à seq_buffer_id d'une valeur (en octets) contenue dans le tampon de données spécifié par les paramètres d'accès au tampon buffer_mode et buffer_specifier.

Parameters:

return_code	FB_error_code	(OUT)	
environment_id	FB_environment_id	(IN)	
seq_buffer_id	FB_seq_buffer_id	(IN)	
buffer_mode	FB_buffer_mode	(IN)	
buffer_specifier	FB_word	(IN)	(FBVAR, FBVAL)
	or FB_seq_buffer_id	(IN)	(FBSEQ)

Action:

At Execution Time, this routine shall displace the current sequential buffer pointer associated with seq_buffer_id, by the value (in bytes) contained in the Data Buffer specified by the buffer access parameters buffer_mode and buffer_specifier.

Section 6. Routines de transaction simples

Les routines de cette section font appel à des transactions FASTBUS qui sont des séquences de cycles FASTBUS.

6.1 Conditions s'appliquant aux routines de transaction et aux routines complexes

Cet article donne les conditions dans lesquelles différents cycles FASTBUS sont explicitement ou implicitement mis en oeuvre par les routines de cette section et de la suivante. Les cycles eux-mêmes sont décrits dans la norme FASTBUS.

6.1.1 Arbitrage

L'arbitrage doit être automatiquement exécuté si le port ne possède pas la maîtrise du bus et si le paramètre FB_PAR_NO_ARBITRATION est FB_FALSE. Si ce paramètre est FB_TRUE, ou si le port a déjà la maîtrise du bus, l'arbitrage doit être supprimé.

Le niveau d'arbitrage utilisé doit être spécifié par le paramètre opérationnel FB_PAR_ARBITRATION_LEVEL.

Avant de positionner AR, un maître doit prendre en compte: si le paramètre opérationnel FB_PAR_ASSURED_ACCESS est FB_TRUE, l'état de la ligne AI, ou si le paramètre FB_PAR_PRIORITIZED_ACCESS est FB_TRUE, le niveau d'arbitrage du maître actuel.

6.1.2 Adressage primaire

Un cycle d'adressage primaire doit être exécuté lorsque l'on appelle la routine d'action, sauf si le paramètre opérationnel FB_PAR_NO_PRIM_ADDR est FB_TRUE. L'adresse primaire utilisée doit être la valeur (au moment de l'appel) du paramètre "primary_address" de la routine.

Si le paramètre opérationnel FB_PAR_EG_UP est FB_TRUE lorsque la routine est appelée, la ligne EG du bus, si nécessaire, doit être positionnée par le port pendant le cycle d'adressage primaire.

Le réessai automatique en cas d'erreur doit être spécifié en utilisant la routine FB_STATUS_SET_RESPONSE décrite à la section 11.

Après chaque cycle d'adressage primaire, le bus doit être libéré (c.-à-d. GK redescendu) avant ou pendant le dernier cycle de données de l'opération sauf si, au moment de l'appel de la routine, le paramètre FB_PAR_HOLD_BUS était FB_TRUE, ou si FB_PAR_HOLD_BUS_ON_ERROR était FB_TRUE et qu'une erreur s'est produite qui empêcherait des opérations FASTBUS ultérieures de s'exécuter, ou que FB_PAR_HOLD_BUS_NO_AR était FB_TRUE et que AR n'a pas été détecté positionné.

Les conditions d'erreur associées au cycle d'adressage primaire sont:

FEAKTO ou FB_ERR_AK_TIMEOUT

Pas de réponse AK dans la limite de temps

FEASSx ou FB_ERR_PRIMARY_ADDRESS_SSx

Réponse SS non prévue

Section 6. Simple Transaction Routines

Routines in this section invoke FASTBUS Transactions that are sequences of FASTBUS cycles.

6.1 Conditions Governing Transaction and Compound Routines

This clause gives the conditions under which various FASTBUS cycles are explicitly or implicitly invoked by the routines in this and the following section. The cycles themselves are described in the FASTBUS standard.

6.1.1 Arbitration

Arbitration shall be automatically performed if the Port does not have bus mastership, and if the parameter FB_PAR_NO_ARBITRATION is FB_FALSE. If this parameter is FB_TRUE, or if the Port already has bus mastership, arbitration shall be suppressed.

The arbitration level used shall be specified by the Operational Parameter FB_PAR_ARBITRATION_LEVEL.

Before asserting AR, a Master shall take account of the state of the AI line if the Operational Parameter FB_PAR_ASSURED_ACCESS is FB_TRUE or alternatively the arbitration level of the current Master, if the parameter FB_PAR_PRIORITIZED_ACCESS is FB_TRUE.

6.1.2 Primary Address

A primary address cycle shall be performed unless the Operational Parameter FB_PAR_NO_PRIM_ADDR is FB_TRUE when the Action Routine is called. The primary address used shall be the value (at call time) of the "primary_address" parameter to the routine.

If the Operational Parameter FB_PAR_EG_UP is FB_TRUE when the routine is called, the EG bus line shall, if appropriate, be asserted by the Port during the primary address cycle.

Automatic retries on error shall be specified by using the FB_STATUS_SET_RESPONSE routine described in section 11.

After each primary address cycle the bus shall be released (i.e. GK lowered) before or during the last data cycle of the Operation, unless, at the time of the routine call, the parameter FB_PAR_HOLD_BUS was FB_TRUE, or FB_PAR_HOLD_BUS_ON_ERROR was FB_TRUE and an error has occurred which would prevent further FASTBUS Operations from being executed, or FB_PAR_HOLD_BUS_NO_AR was FB_TRUE and AR has not been detected as asserted.

Error conditions associated with primary address cycles are:

FEAKTO or FB_ERR_AK_TIMEOUT No AK response within time limit

FEASSx or FB_ERR_PRIMARY_ADDRESS_SSx Unexpected SS response

6.1.3 Cycle d'écriture de l'adresse secondaire

Pour les routines dans l'espace données, si le paramètre opérationnel FB_PAR_NO_SEC_ADDR est FB_FALSE, le cycle d'adressage primaire doit être suivi par un cycle d'adresse secondaire. Si ce paramètre est FB_TRUE, le cycle d'adresse secondaire doit être omis et le paramètre secondary_address ignoré.

Pour les routines dans l'espace CSR, si le paramètre opérationnel FB_PAR_NO_SEC_ADDR_CSR est FB_FALSE, le cycle d'adressage primaire doit être suivi par un cycle d'adresse secondaire. Si ce paramètre est FB_TRUE, le cycle d'adresse secondaire doit être omis et le paramètre secondary_address ignoré.

L'adresse secondaire utilisée doit être la valeur (au moment de l'appel) du paramètre "secondary_address" de la routine (sauf spécification contraire dans la description de la routine).

Le réessai automatique en cas d'erreur doit être spécifié en utilisant la routine FB_STATUS_SET_RESPONSE décrite à la section 11.

Les conditions d'erreur associées au cycle d'adresse secondaire sont:

FESATO ou FB_ERR_SECONDARY_ADDRESS_TIMEOUT	Pas de réponse DK dans la limite de temps
FESSx ou FB_ERR_SECONDARY_ADDRESS_SSx	Réponse SS non prévue

6.1.4 Cycles de données

Des cycles de données sont mis en oeuvre par la plupart des routines de cette section, comme indiqué ci-après pour chaque routine, sauf si FB_PAR_NO_DATA_CYCLE est FB_TRUE lorsque la routine est appelée.

Le réessai automatique en cas d'erreur doit être spécifié en utilisant la routine FB_STATUS_SET_RESPONSE décrite à la section 11.

Lorsque le paramètre FB_PAR_SHORT_DATA_WORD est FB_TRUE, les mots de moins de 32 bits sont, à chaque cycle de données, répartis dans le tampon de données ou reformés à partir de celui-ci (voir section 5).

Le paramètre opérationnel FB_PAR_PIPELINE doit déterminer si une routine de transfert par bloc exécute un transfert de bloc ou un transfert en pipe-line.

Les conditions d'erreur associées au cycle de données sont:

FEDKTO ou FB_ERR_DK_TIMEOUT	Pas de réponse DK dans la limite de temps
FEDSSx ou FB_ERR_DATA_SSx	Mauvaise réponse SS
FEBSS2 ou FB_ERR_BLOCK_DATA_SS2	Réponse SS2 à un transfert de bloc

6.1.5 Transfert de bloc

Lorsqu'il contrôle un transfert de bloc, un maître peut suivre différentes stratégies, selon les possibilités du matériel et la valeur de différents paramètres opérationnels.

Si le paramètre opérationnel FB_PAR_WHOLE_BLOCK est FB_TRUE, le maître doit transférer le bloc entier en une opération ininterrompue.

Si ce paramètre est FB_FALSE, le maître peut diviser le transfert en transferts d'un certain nombre de paquets plus petits. La dimension (maximale) de ces paquets est définie par le paramètre opérationnel FB_PAR_BLOCKLET_SIZE.

6.1.3 Secondary Address Write Cycle

For Data Space routines, if the Operational Parameter FB_PAR_NO_SEC_ADDR is FB_FALSE, the primary address cycle shall be followed by a secondary address cycle. If this Operational Parameter is FB_TRUE, the secondary address cycle shall be omitted and the secondary_address parameter ignored.

For CSR Space routines, if the Operational Parameter FB_PAR_NO_SEC_ADDR_CSR is FB_FALSE, the primary address cycle shall be followed by a secondary address cycle. If this Operational Parameter is FB_TRUE, the secondary address cycle shall be omitted and the secondary_address parameter ignored.

The secondary address used shall be the value (at Call Time) of the "secondary_address" parameter to the routine (unless stated otherwise in the routine description).

Automatic retries on error shall be specified by using the FB_STATUS_SET_RESPONSE routine described in section 11.

Error conditions associated with secondary address cycles are:

FESATO or FB_ERR_SECONDARY_ADDRESS_TIMEOUT	No DK response within time limit
FESSSx or FB_ERR_SECONDARY_ADDRESS_SSx	Unexpected SS response

6.1.4 Data Cycles

Data cycles are invoked by most of the routines in this section, as stated below for each routine, unless FB_PAR_NO_DATA_CYCLE is FB_TRUE when the routine is called.

Automatic retries on error shall be specified by using the FB_STATUS_SET_RESPONSE routine described in section 11.

The parameter FB_PAR_SHORT_DATA_WORD, when FB_TRUE, causes words of less than 32 bits to be packed into, or unpacked from, the Data Buffer by data cycles (see section 5).

The FB_PAR_PIPELINE Operational Parameter shall determine whether block transfer or pipelined transfers are performed by the block transfer routines.

Error conditions associated with Data cycles are:

FEDKTO or FB_ERR_DK_TIMEOUT	No DK response within time limit
FEDSSx or FB_ERR_DATA_SSx	Bad SS response
FEBSS2 or FB_ERR_BLOCK_DATA_SS2	SS2 response on a block transfer

6.1.5 Block Transfers

When controlling a block transfer, a Master can follow one of several possible strategies, depending on its hardware capabilities and on the values of several Operational Parameters.

The Master shall transfer the entire block in one uninterrupted Operation, if the Operational Parameter FB_PAR_WHOLE_BLOCK is FB_TRUE.

If this parameter is FB_FALSE, the Master may divide the transfer up into the transfer of a number of smaller blocklets. The (maximum) size of these blocklets is defined by the Operational Parameter FB_PAR_BLOCKLET_SIZE.

Après que chaque paquet a été transféré, le maître doit soit relâcher le bus et participer au cycle d'arbitrage suivant, soit examiner l'état de AR et ne relâcher le bus que si AR est positionné.

Les maîtres qui réagissent à l'état de AR peuvent, au lieu d'utiliser des paquets de dimension fixe, surveiller continuellement l'état de AR pendant le transfert et réagir correctement lorsque AR se trouve positionné. Cette stratégie est plus efficace lorsque les autres maîtres potentiels obéissent au protocole d'accès prioritaire pour positionner AR.

6.1.6 Déconnexion

Après une opération effectuée par une routine de cette section, il doit y avoir une déconnexion implicite de l'esclave, sauf si le paramètre FB_PAR_HOLD_AS était FB_TRUE au moment de l'appel, ou si FB_PAR_HOLD_BUS_ON_ERROR était FB_TRUE et qu'une erreur s'est produite qui empêcherait des opérations FASTBUS ultérieures de s'exécuter.

6.2 Routines de transaction simples FASTBUS

6.2.1 Transfert d'un mot unique de données

Nom:	FRD	ou	FB_READ_DAT	Lire un mot dans l'espace données
	FWD	ou	FB_WRITE_DAT	Ecrire un mot dans l'espace données
	FRC	ou	FB_READ_CSR	Lire un mot dans l'espace CSR
	FWC	ou	FB_WRITE_CSR	Ecrire un mot dans l'espace CSR
	FRDM	ou	FB_READ_DAT_MULT	Lire un mot dans l'espace données. Ecouteurs multiples
	FWDM	ou	FB_WRITE_DAT_MULT	Ecrire un mot dans l'espace données. Ecouteurs multiples
	FRCM	ou	FB_READ_CSR_MULT	Lire un mot dans l'espace CSR. Ecouteurs multiples
	FWCM	ou	FB_WRITE_CSR_MULT	Ecrire un mot dans l'espace CSR. Ecouteurs multiples

Catégorie: A, Routine d'action

Paramètres:

return_code	FB_error_code	(OUT)	
environment_id	FB_environment_id	(IN)	
primary_address	FB_word	(IN)	
secondary_address	FB_word	(IN)	
buffer_mode	FB_buffer_mode	(IN)	
buffer_specifier	FB_word	(IN)	FWxx (FBVAR, FBVAL)
	ou FB_word	(OUT)	FRxx (FBVAR)
	ou FB_seq_buffer_id	(IN)	(FBSEQ)

Action:

Un transfert d'un seul mot de données doit être exécuté.

Le code MS utilisé pour le cycle d'adressage primaire et la valeur des lignes RD pour les cycles de données dépendent comme suit de la routine appelée:

After each blocklet has been transferred, the Master shall either release the bus and participate in the following Arbitration cycle, or examine the state of AR and only release the bus if AR is found to be asserted.

Masters that can react to the state of AR may, rather than using a fixed blocklet size, continuously monitor the state of AR during the transfer and react appropriately when AR becomes asserted. This strategy is most effective when all other potential Masters obey the Prioritized Access protocol for asserting AR.

6.1.6 Disconnection

After an Operation invoked by a routine in this section, there shall be an implicit disconnection from the slave unless the parameter FB_PAR_HOLD_AS was FB_TRUE at the time of the call, or unless FB_PAR_HOLD_BUS_ON_ERROR was FB_TRUE and an error has occurred which would prevent further FASTBUS Operations from being executed.

6.2 Simple FASTBUS Transaction Routines

6.2.1 Single Data Word Transfer

Name:	FRD	or	FB_READ_DAT	Read Word from Data Space
	FWD	or	FB_WRITE_DAT	Write Word to Data Space
	FRC	or	FB_READ_CSR	Read Word from CSR Space
	FWC	or	FB_WRITE_CSR	Write Word to CSR Space
	FRDM	or	FB_READ_DAT_MULT	Read Word, Data Space, Multi-listener
	FWDM	or	FB_WRITE_DAT_MULT	Write Word, Data Space, Multi-listener
	FRCM	or	FB_READ_CSR_MULT	Read Word, CSR Space, Multi-listener
	FWCM	or	FB_WRITE_CSR_MULT	Write Word, CSR Space, Multi-listener

Category: A, Action Routine

Parameters:

return_code	FB_error_code	(OUT)	
environment_id	FB_environment_id	(IN)	
primary_address	FB_word	(IN)	
secondary_address	FB_word	(IN)	
buffer_mode	FB_buffer_mode	(IN)	
buffer_specifier	FB_word	(IN)	FWxx (FBVAR, FBVAL)
	or FB_word	(OUT)	FRxx (FBVAR)
	or FB_seq_buffer_id	(IN)	(FBSEQ)

Action:

A single data word transfer shall take place.

The MS code used for the Primary address cycle and the RD line value for the Data cycle shall depend as follows on the routine called:

Routine	MS	RD
FB_READ_DAT	MS = 0	RD = 1
FB_WRITE_DAT	MS = 0	RD = 0
FB_READ_CSR	MS = 1	RD = 1
FB_WRITE_CSR	MS = 1	RD = 0
FB_READ_DAT_MULT	MS = 2	RD = 1
FB_WRITE_DAT_MULT	MS = 2	RD = 0
FB_READ_CSR_MULT	MS = 3	RD = 1
FB_WRITE_CSR_MULT	MS = 3	RD = 0

Les cycles de données doivent avoir MS = 0.

Les considérations gérant l'exécution des cycles individuels, définies à l'article 6.1, doivent s'appliquer.

6.2.2 Transfert de bloc

Nom: FRDB ou FB_READ_DAT_BLOCK Lire un bloc, espace données
 FWDB ou FB_WRITE_DAT_BLOCK Ecrire un bloc, espace données
 FRCB ou FB_READ_CSR_BLOCK Lire un bloc, espace CSR
 FWCB ou FB_WRITE_CSR_BLOCK Ecrire un bloc, espace CSR
 FRDBM ou FB_READ_DAT_BLOCK_MULT Lire un bloc, espace données.
 Ecouteurs multiples
 FWDBM ou FB_WRITE_DAT_BLOCK_MULT Ecrire un bloc, espace données.
 Ecouteurs multiples
 FRCBM ou FB_READ_CSR_BLOCK_MULT Lire un bloc, espace CSR.
 Ecouteurs multiples
 FWCBM ou FB_WRITE_CSR_BLOCK_MULT Ecrire un bloc, espace CSR.
 Ecouteurs multiples

Catégorie: A, Routine d'action

Paramètres:

return_code	FB_error_code	(OUT)	
environment_id	FB_environment_id	(IN)	
primary_address	FB_word	(IN)	
secondary_address	FB_word	(IN)	
buffer_mode	FB_buffer_mode	(IN)	
buffer_specifier	array_of_FB_word	(IN)	FWxx (FBVAR)
	ou array_of_FB_word	(OUT)	FRxx (FBVAR)
	ou FB_seq_buffer_id	(IN)	(FBSEQ)
max_bytes	FB_integer	(IN)	

Action:

Un transfert de bloc doit être exécuté.

Le code MS utilisé pour le cycle d'adressage primaire et la valeur des lignes RD pour le transfert de bloc dépendent comme suit de la routine appelée:

Routine	MS	RD
FB_READ_DAT	MS = 0	RD = 1
FB_WRITE_DAT	MS = 0	RD = 0
FB_READ_CSR	MS = 1	RD = 1
FB_WRITE_CSR	MS = 1	RD = 0
FB_READ_DAT_MULT	MS = 2	RD = 1
FB_WRITE_DAT_MULT	MS = 2	RD = 0
FB_READ_CSR_MULT	MS = 3	RD = 1
FB_WRITE_CSR_MULT	MS = 3	RD = 0

The Data cycle shall have MS = 0.

The considerations governing the execution of the individual cycles defined in clause 6.1 shall apply.

6.2.2 Block Transfer

Name: FRDB or FB_READ_DAT_BLOCK Read Block, Data Space
 FWDB or FB_WRITE_DAT_BLOCK Write Block, Data Space
 FRCB or FB_READ_CSR_BLOCK Read Block, CSR Space
 FWCB or FB_WRITE_CSR_BLOCK Write Block, CSR Space
 FRDBM or FB_READ_DAT_BLOCK_MULT Read Block, Data Space, Multi-listener
 FWDBM or FB_WRITE_DAT_BLOCK_MULT Write Block, Data Space, Multi-listener
 FRCBM or FB_READ_CSR_BLOCK_MULT Read Block, CSR Space, Multi-listener
 FWCBM or FB_WRITE_CSR_BLOCK_MULT Write Block, CSR Space, Multi-listener

Category: A, Action Routine

Parameters:

return_code	FB_error_code	(OUT)	
environment_id	FB_environment_id	(IN)	
primary_address	FB_word	(IN)	
secondary_address	FB_word	(IN)	
buffer_mode	FB_buffer_mode	(IN)	
buffer_specifier	array_of_FB_word	(IN)	FWxx (FBVAR)
	or array_of_FB_word	(OUT)	FRxx (FBVAR)
	or FB_seq_buffer_id	(IN)	(FBSEQ)
max_bytes	FB_integer	(IN)	

Action:

A block transfer shall take place.

The MS code used for the primary address cycle and the RD line value for the block transfer shall depend on the routine called as follows:

Routine	MS	RD
FB_READ_DAT_BLOCK	MS = 0	RD = 1
FB_WRITE_DAT_BLOCK	MS = 0	RD = 0
FB_READ_CSR_BLOCK	MS = 1	RD = 1
FB_WRITE_CSR_BLOCK	MS = 1	RD = 0
FB_READ_DAT_BLOCK_MULT	MS = 2	RD = 1
FB_WRITE_DAT_BLOCK_MULT	MS = 2	RD = 0
FB_READ_CSR_BLOCK_MULT	MS = 3	RD = 1
FB_WRITE_CSR_BLOCK_MULT	MS = 3	RD = 0

Le code MS utilisé pour le transfert de bloc dépend du paramètre opérationnel FB_PAR_PIPELINE. S'il est FB_FALSE, MS = 1. S'il est FB_TRUE, MS = 3.

Les considérations gérant l'exécution des cycles individuels, définies à l'article 6.1, doivent s'appliquer.

6.2.3 Routines d'adresse secondaire

Nom: FRDSA ou FB_READ_DAT_SA Lire l'adresse secondaire, espace données
 FWDSA ou FB_WRITE_DAT_SA Ecrire l'adresse secondaire, espace données
 FRCSA ou FB_READ_CSR_SA Lire l'adresse secondaire, espace CSR
 FWCSA ou FB_WRITE_CSR_SA Ecrire l'adresse secondaire, espace CSR

Catégorie: A, Routine d'action

Paramètres:

return code	FB_error_code	(OUT)	
environment_id	FB_environment_id	(IN)	
primary_address	FB_word	(IN)	
buffer_mode	FB_buffer_mode	(IN)	
buffer_specifier	FB_word	(IN)	FWxSA (FBVAR, FBVAL)
	ou FB_word	(OUT)	FRxSA (FBVAR)
	ou FB_seq_buffer_id	(IN)	(FBSEQ)

Action:

Un transfert d'un seul mot de données du NTA ou dans celui-ci doit être exécuté. Le code MS utilisé pour le cycle d'adressage primaire et la valeur des lignes RD pour les cycles de données dépendent comme suit de la routine appelée:

Routine	MS	RD
FB_READ_DAT_SA	MS = 0	RD = 1
FB_WRITE_DAT_SA	MS = 0	RD = 0
FB_READ_CSR_SA	MS = 1	RD = 1
FB_WRITE_CSR_SA	MS = 1	RD = 0

Le cycle de données doit avoir MS = 2. Les considérations gérant l'exécution des cycles individuels, définies à l'article 6.1, doivent s'appliquer, sauf que le cycle d'adresse secondaire peut impliquer une lecture ou une écriture, avec les données transférées au moment de l'exécution de ou vers le tampon de données.

Routine	MS	RD
FB_READ_DAT_BLOCK	MS=0	RD=1
FB_WRITE_DAT_BLOCK	MS=0	RD=0
FB_READ_CSR_BLOCK	MS=1	RD=1
FB_WRITE_CSR_BLOCK	MS=1	RD=0
FB_READ_DAT_BLOCK_MULT	MS=2	RD=1
FB_WRITE_DAT_BLOCK_MULT	MS=2	RD=0
FB_READ_CSR_BLOCK_MULT	MS=3	RD=1
FB_WRITE_CSR_BLOCK_MULT	MS=3	RD=0

The MS code used for the block transfer shall depend on the FB_PAR PIPELINE Operational Parameter. If it is FB_FALSE then MS=1. If it is FB_TRUE then MS=3.

The considerations governing the execution of the individual cycles defined in clause 6.1 shall apply.

6.2.3 Secondary Address Routines

Name: FRDSA or FB_READ_DAT_SA Read Data Space secondary address
 FWDSA or FB_WRITE_DAT_SA Write Data Space secondary address
 FRCSA or FB_READ_CSR_SA Read CSR Space secondary address
 FWCSA or FB_WRITE_CSR_SA Write CSR Space secondary address

Category: A, Action Routine

Parameters:

return_code	FB_error_code	(OUT)	
environment_id	FB_environment_id	(IN)	
primary_address	FB_word	(IN)	
buffer_mode	FB_buffer_mode	(IN)	
buffer_specifier	FB_word	(IN)	FWxSA (FBVAR, FBVAL)
	or FB_word	(OUT)	FRxSA (FBVAR)
	or FB_seq_buffer_id	(IN)	(FBSEQ)

Action:

A single word data transfer from or to the NTA register shall take place. The MS code used for the primary address cycle and the RD line value for the DS cycle shall depend as follows on the routine called:

Routine	MS	RD
FB_READ_DAT_SA	MS=0	RD=1
FB_WRITE_DAT_SA	MS=0	RD=0
FB_READ_CSR_SA	MS=1	RD=1
FB_WRITE_CSR_SA	MS=1	RD=0

The Data cycle shall have MS=2. The considerations governing the execution of the individual cycles defined in clause 6.1 shall apply, except that the secondary address cycle may involve a read or write, with the data transferred to or from the Data Buffer at Execution Time.

Le paramètre opérationnel FB_PAR_NO_SEC_ADDR doit être pris en compte. Ainsi, s'il est FB_TRUE, aucune action, autre que la séquence de fin, ne suit le cycle d'adressage primaire.

6.2.4 Lecture de la longueur du dernier transfert de données

Nom: FRLen ou FB_READ_LENGTH

Catégorie: E, Routine d'action

Paramètres:

return_code	FB_error_code	(OUT)	
environment_id	FB_environment_id	(IN)	
buffer_mode	FB_buffer_mode	(IN)	
buffer_specifier	FB_word	(OUT)	(FBVAR)
	ou FB_seq_buffer_id	(IN)	(FBSEQ)

Action:

Le nombre d'octets transférés de ou vers le tampon utilisateur pendant la dernière action FASTBUS (dans l'environnement spécifié) qui implique un identificateur de tampon utilisateur doit être retourné (au moment de l'exécution) dans le tampon de données de FB_READ_LENGTH.

The Operational Parameter FB_PAR_NO_SEC_ADDR shall be active. Thus, if it is FB_TRUE, no action follows the primary address cycle other than the termination sequence.

6.2.4 Read Length of Last Data Transfer

Name: FRLen or FB_READ_LENGTH

Category: E, Action Routine

Parameters:

return_code	FB_error_code	(OUT)	
environment_id	FB_environment_id	(IN)	
buffer_mode	FB_buffer_mode	(IN)	
buffer_specifier	FB_word	(OUT)	(FBVAR)
	or FB_seq_buffer_id	(IN)	(FBSEQ)

Action:

The number of bytes transferred to or from the user buffer during the last FASTBUS Action (in the specified Environment) which involved a user buffer specifier shall be returned (at Execution Time) in the Data Buffer of FB_READ_LENGTH.

Section 7. Routines de transaction complexes

Les routines définies dans cette section sont un sous-ensemble de toutes les transactions complexes FASTBUS possibles. Les actions FASTBUS élémentaires de la transaction sont exécutées dans les conditions décrites à l'article 6.1.

7.1 Accès à la table de routage de l'interconnexion de segment

Les routines suivantes sont définies pour accéder à la table de routage de l'interconnexion de segment.

Nom: FRRT ou FB_READ_ROUTE_TABLE (mot unique)
 FWRT ou FB_WRITE_ROUTE_TABLE (mot unique)
 FRRTB ou FB_READ_ROUTE_TABLE_BLOCK
 FWRTB ou FB_WRITE_ROUTE_TABLE_BLOCK

Catégorie: E, Routine d'action

Paramètres:

return_code	FB_error_code	(OUT)	
environment_id	FB_environment_id	(IN)	
primary_address	FB_word	(IN)	
route_address	FB_word	(IN)	
buffer_mode	FB_buffer_mode	(IN)	
buffer_specifier	FB_word	(IN)	FWRT (FBVAR, FBVAL)
	ou FB_word	(OUT)	FRRT (FBVAR)
	ou array_of_FB_word	(IN)	FWRTB (FBVAR)
	ou array_of_FB_word	(OUT)	FRRTB (FBVAR)
	ou FB_seq_buffer_id	(IN)	toutes (FBSEQ)
max_bytes	FB_integer	(IN)	FxRTB seulement

Action:

La partie esclave de l'interconnexion de segment, à l'adresse géographique *primary_address*, doit être adressée dans l'espace CSR. Un cycle d'adresse secondaire spécifiant l'adresse 40h est suivi par un cycle d'écriture de données avec MS=0 et la valeur *route_address*, pour définir l'adresse (de départ) à utiliser pour un accès à la table de routage. Ce cycle d'écriture de données doit être suivi par un cycle d'adresse secondaire spécifiant l'adresse 41h - le registre de données de la table de routage. Un cycle de données avec MS=0 doit être exécuté pour FB_READ_ROUTE_TABLE et pour FB_WRITE_ROUTE_TABLE, ou un transfert de bloc avec MS=1 doit être exécuté pour FB_READ_ROUTE_TABLE_BLOCK et FB_WRITE_ROUTE_TABLE_BLOCK. Les données doivent être transférées de ou vers le tampon de données spécifié.

7.2 Déplacement de données entre des dispositifs FASTBUS

Nom: FMOVD ou FB_MOVE_DAT Transférer un mot dans l'espace données
 FMOCV ou FB_MOVE_CSR Transférer un mot dans l'espace CSR
 FMOVDB ou FB_MOVE_DAT-BLOCK Transférer un bloc dans l'espace données
 FMOCVB ou FB_MOVE_CSR_BLOCK Transférer un bloc dans l'espace CSR

Catégorie: E, Routine d'action

Section 7. Compound Transaction Routines

The routines defined in this section are a subset of the many possible compound FASTBUS Transactions. The individual FASTBUS Actions are executed under the same conditions as described in clause 6.1.

7.1 Access Segment Interconnect Route Table

The following routines are defined to perform accesses to the Route Table of a Segment Interconnect (SI).

Name: FRRT or FB_READ_ROUTE_TABLE (single word)
 FWRT or FB_WRITE_ROUTE_TABLE (single word)
 FRRTB or FB_READ_ROUTE_TABLE_BLOCK
 FWRTB or FB_WRITE_ROUTE_TABLE_BLOCK

Category: E, Action Routine

Parameters:

return_code	FB_error_code	(OUT)	
environment_id	FB_environment_id	(IN)	
primary_address	FB_word	(IN)	
route_address	FB_word	(IN)	
buffer_mode	FB_buffer_mode	(IN)	
buffer_specifier	FB_word	(IN)	FWRT (FBVAL, FBVAR)
	or FB_word	(OUT)	FRRT (FBVAR)
	or array_of_FB_word	(IN)	FWRTB (FBVAR)
	or array_of_FB_word	(OUT)	FRRTB (FBVAR)
	or FB_seq_buffer_id	(IN)	any (FBSEQ)
max_bytes	FB_integer	(IN)	FxRTB only

Action:

The slave section of the Segment Interconnect at geographical address primary_address shall be addressed in CSR space. A secondary address cycle specifying address 40h is followed by a write data cycle with MS=0 and value route_address, to specify the (starting) address to be used for route table accesses. This write data cycle shall be followed by a secondary address cycle specifying address 41h - the route table data register. A data cycle with MS=0 shall be carried out for FB_READ_ROUTE_TABLE and FB_WRITE_ROUTE_TABLE or a block transfer with MS=1 shall be carried out for FB_READ_ROUTE_TABLE_BLOCK and FB_WRITE_ROUTE_TABLE_BLOCK. The data shall be transferred to or from the specified Data Buffer.

7.2 Move Data between FASTBUS Devices

Name: FMOVD or FB_MOVE_DAT Transfer Word in Data space
 FMOVCS or FB_MOVE_CSR Transfer Word in CSR space
 FMOVDB or FB_MOVE_DAT_BLOCK Transfer Block in Data space
 FMOVCSB or FB_MOVE_CSR_BLOCK Transfer Block in CSR space

Category: E, Action Routine

Paramètres:

return_code	FB_error_code	(OUT)	
environment_id	FB_environment_id	(IN)	
primary_address1	FB_word	(IN)	
secondary_address1	FB_word	(IN)	
primary_address2	FB_word	(IN)	
secondary_address2	FB_word	(IN)	
buffer_mode	FB_buffer_mode	(IN)	FMOVxB seulement
buffer_specifier	FB_word	(OUT)	FMOVxB (FBVAR)
	ou FB_seq_buffer_id	(IN)	FMOVxB (FBSEQ)
max_bytes	FB_integer	(IN)	FMOVxB seulement

Action:

Une opération de lecture (de mot ou de bloc) doit être exécutée sur un esclave en utilisant les paramètres *primary_address1* et *secondary_address1*. Si la lecture se termine sans erreurs, les données lues dans cette opération doivent être écrites dans un autre esclave, en utilisant les paramètres *primary_address2* et *secondary_address2*. Si l'on utilise un tampon séquentiel comme tampon intermédiaire de données, le pointeur du tampon doit être déplacé entre la lecture et l'écriture pour que la donnée qui vient d'être lue soit réécrite.

Les considérations de l'article 6.1 s'appliquent à la fois à l'opération de lecture et à celle d'écriture.

Pour le transfert d'un mot unique, les données doivent être lues dans un tampon interne à la réalisation - aucun paramètre de tampon ne doit être inclus dans l'appel à la routine. Pour les transferts de bloc, un tampon intermédiaire doit être spécifié.

7.3 Lecture-modification-écriture dans un registre FASTBUS

Nom: FMODD ou FB MODIFY_DAT Modifier un mot dans l'espace données
 FMODC ou FB MODIFY_CSR Modifier un mot dans l'espace CSR

Catégorie: E, Routine d'action

Paramètres:

return_code	FB_error_code	(OUT)
environment_id	FB_environment_id	(IN)
primary_address	FB_word	(IN)
secondary_address	FB_word	(IN)
operator_id	FB_operator	(IN)
operand	FB_word	(IN)

Action:

Un mot doit être lu dans le registre de l'esclave défini par les paramètres *primary_address* et *secondary_address*. La maîtrise du bus doit être maintenue pendant que l'opération arithmétique ou logique spécifiée par *operator_id* est exécutée, en utilisant la valeur lue comme premier paramètre, et l'opérande comme second paramètre de *operator_id*. Le résultat de cette opération est réécrit dans le même registre de l'esclave. La maîtrise du bus est alors relâchée.

Les *operator_ids* définis sont:

Parameters:

return_code	FB_error_code	(OUT)	
environment_id	FB_environment_id	(IN)	
primary_address1	FB_word	(IN)	
secondary_address1	FB_word	(IN)	
primary_address2	FB_word	(IN)	
secondary_address2	FB_word	(IN)	
buffer_mode	FB_buffer_mode	(IN)	FMOVxB only
buffer_specifier	FB_word	(OUT)	FMOVxB (FBVAR)
	or FB_seq_buffer_id	(IN)	FMOVxB (FBSEQ)
max_bytes	FB_integer	(IN)	FMOVxB only

Action:

A read (Word or Block) Operation shall be performed on a slave using the parameters primary_address1 and secondary_address1. If the read completes without error, the data read in that Operation shall then be written to another Slave, using the parameters primary_address2 and secondary_address2. If a Sequential Buffer is used as an intermediate Data Buffer, the buffer pointer shall be displaced between the read and the write so that the same data that has been read is written back out.

The considerations in clause 6.1 shall apply to both the read and the write Operations.

For a single word transfer, the data shall be read into an internal Data Buffer in the implementation - no buffer parameters shall be included in the call to the move routine. For block transfers, an intermediate buffer shall be specified.

7.3 Read-Modify-Write a FASTBUS Location

Name: FMODD or FB_MODIFY_DAT Modify Word in Data space
 FMODC or FB_MODIFY_CSR Modify Word in CSR space

Category: E, Action Routine

Parameters:

return_code	FB_error_code	(OUT)
environment_id	FB_environment_id	(IN)
primary_address	FB_word	(IN)
secondary_address	FB_word	(IN)
operator_id	FB_operator	(IN)
operand	FB_word	(IN)

Action:

A word shall be read from the slave register specified by the parameters primary_address and secondary_address. Mastership of the bus shall be maintained while the arithmetic or logical Operation specified by operator_id is carried out using the value read as the first parameter, and the operand as the second parameter of the operator_id. The result of this Operation is written back to the same register in the Slave. Bus mastership is then released.

Defined operator_ids are:

Nom court	Nom long	Signification
FMSET	FB_MOD_SET	Positionne le bit défini par l'opérande
FMCLR	FB_MOD_CLEAR	Met à zéro le bit défini par l'opérande
FMAND	FB_MOD_AND	Fonction ET entre les données lues et l'opérande
FMOR	FB_MOD_OR	Fonction OU entre les données lues et l'opérande
FMXOR	FB_MOD_XOR	Fonction OU inclusif entre les données lues et l'opérande
FMNOT	FB_MOD_NOT	Effectue le complément à "1" des données lues
FMRSH	FB_MOD_RSHIFT	Décalle les données lues vers la droite d'une quantité définie par l'opérande
FMLSH	FB_MOD_LSHIFT	Décalle les données lues vers la gauche d'une quantité définie par l'opérande

7.4 Routines de concentration de données

Ces routines sont utilisées pour exécuter une lecture séquentielle de blocs de données provenant de différents dispositifs FASTBUS. Chaque routine inclut une procédure différente de sélection de l'adresse FASTBUS à partir de laquelle la lecture sera initialisée.

7.4.1 Lecture des données dans des dispositifs adressés séquentiellement

Nom: FRDSEQ ou FB_READ_DAT_SEQUENTIAL Lecture séquentielle dans l'espace données
 FRCSEQ ou FB_READ_CSR_SEQUENTIAL Lecture séquentielle dans l'espace CSR

Catégorie: E, Routine d'action

Paramètres:

return_code	FB_error_code	(OUT)	
control_id	FB_control_id	(IN)	
start_primad	FB_word	(IN)	
last_primad	FB_word	(IN)	
primad_step	FB_word	(IN)	
secondary_address	FB_word	(IN)	
buffer_mode	FB_buffer mode	(IN)	
buffer_specifier	array_of FB_word	(OUT)	(FBVAR)
	ou FB_seq_buffer_id	(IN)	(FBSEQ)
buffer_size	FB_integer	(IN)	
max_bytes	FB_integer	(IN)	

Action:

Un transfert de bloc d'une longueur maximale de max_bytes doit être exécuté à l'adresse spécifiée par start_primad et secondary_address. Les données doivent être transférées de ou vers le tampon de données indiqué. Sans relâcher la maîtrise du bus, le port libère l'esclave connecté et doit calculer une nouvelle adresse primaire en ajoutant primad_step au paramètre courant d'adresse primaire. Si la nouvelle adresse primaire ne dépasse pas last_primad, un autre transfert de bloc d'un maximum de max_bytes doit être effectué, et les données placées à la suite de celles du transfert de bloc précédent. Le processus doit être répété jusqu'à ce que last_primad soit dépassé, à ce moment le bus est relâché.

Short Name	Long Name	Meaning
FMSET	FB_MOD_SET	Set bit positions specified by operand
FMCLR	FB_MOD_CLEAR	Clear bit positions specified by operand
FMAND	FB_MOD_AND	AND data read with operand
FMOR	FB_MOD_OR	OR data read with operand
FMXOR	FB_MOD_XOR	XOR data read with operand
FMNOT	FB_MOD_NOT	Perform ones complement on data read
FMRSH	FB_MOD_RSHIFT	Shift data read to the right with length of shift specified by operand
FMLSH	FB_MOD_LSHIFT	Shift data read to the left with length of shift specified by operand

7.4 Data Gathering Routines

These routines are used to perform a block data read sequentially from several different FASTBUS Devices. Each routine includes a different selection procedure for the FASTBUS addresses from which the block read will be initiated.

7.4.1 Read Data from Sequential Device Addresses

Name: FRDSEQ or FB_READ_DAT_SEQUENTIAL Read Sequential in Data Space

FRCSEQ or FB_READ_CSR_SEQUENTIAL Read Sequential in CSR Space

Category: E, Action Routine

Parameters:

return_code	FB_error_code	(OUT)
control_id	FB_control_id	(IN)
start_primad	FB_word	(IN)
last_primad	FB_word	(IN)
primad_step	FB_word	(IN)
secondary_address	FB_word	(IN)
buffer_mode	FB_buffer_mode	(IN)
buffer_specifier	array_of_FB_word	(OUT) (FBVAR)
	or FB_seq_buffer_id	(IN) (FBSEQ)
buffer_size	FB_integer	(IN)
max_bytes	FB_integer	(IN)

Action:

A block transfer of up to max_bytes shall be carried out at the address specified by start_primad and secondary_address. The data shall be transferred to or from the specified Data Buffer. Without releasing bus mastership, the Port releases the attached Slave and shall calculate a new primary address by adding primad_step to the current primary address parameter. If the new primary address does not exceed last_primad, another block transfer of up to max_bytes shall be carried out, and the data concatenated with the data of the previous block transfer. This process shall be repeated until last_primad is exceeded, at which time the bus is released.

7.4.2 Lecture des données dans un dispositif qui répond à une scrutation des contacts T

Nom: FRDTPS ou FB_READ_DAT_TP_SCAN Scrutation, lecture dans l'espace données
FRCTPS ou FB_READ_CSR_TP_SCAN Scrutation, lecture dans l'espace CSR

Catégorie: E, Routine d'action

Paramètres:

return_code	FB_error_code	(OUT)
environment_id	FB_environment_id	(IN)
primary_address	FB_word	(IN)
secondary_address	FB_word	(IN)
buffer_mode	FB_buffer_mode	(IN)
buffer_specifier	FB_word	(OUT) (FBVAR)
	ou FB_seq_buffer_id	(IN) (FBSEQ)
buffer_size	FB_integer	(IN)
max_bytes	FB_integer	(IN)

Action:

Un cycle d'adressage primaire en diffusion utilisant le paramètre `primary_address`, et $MS=2$ (FRDTPS) ou $MS=3$ (FRCTPS) doit être suivi par un cycle de lecture de données d'un mot unique avec $MS=0$. L'adresse primaire doit avoir le bit global (bit 1) à zéro et doit correspondre à l'un des cas 3, 3a, 4, 5 ou 6 de la Table III de la norme FASTBUS.

Pour chaque bit positionné dans le mot lu, on construit une adresse géographique composée du champ GP de `primary_address` spécifié, ajouté à la valeur codée de la position de ce bit du mot de données. A chacune de ces adresses, on exécute un cycle d'adresse secondaire utilisant la valeur du paramètre `secondary_address`, suivi par une opération de lecture de données en transfert de bloc d'un maximum de `max_bytes`. Les données provenant des transferts de bloc successifs sont réunies dans le tampon de données spécifié.

7.4.3 Attribution d'un dispositif FASTBUS

Nom: FDVAL ou FB_DEVICE_ALLOCATE

Catégorie: E, Routine d'action

Paramètres:

return_code	FB_error_code	(OUT)
environment_id	FB_environment_id	(IN)
primary_address	FB_word	(IN)

Action:

Un cycle d'adressage primaire dans l'espace CSR doit être suivi par un cycle d'adresse secondaire spécifiant l'adresse 4. Un cycle d'écriture d'un mot unique de données est ensuite exécuté avec la propre adresse FASTBUS du port comme donnée. Les considérations de l'article 6.1 s'appliquent à l'exception de la valeur de l'adresse secondaire qui est toujours 4.

7.4.4 Libération d'un dispositif FASTBUS

Nom: FDVDE ou FB_DEVICE_DEALLOCATE

Catégorie: E, Routine d'action

7.4.2 Read Data from Devices that Respond to a T-pin Scan

Name: FRDTPS or FB_READ_DAT_TP_SCAN Scan, read Data space
 FRCTPS or FB_READ_CSR_TP_SCAN Scan, read CSR space

Category: E, Action Routine

Parameters:

return_code	FB_error_code	(OUT)
environment_id	FB_environment_id	(IN)
primary_address	FB_word	(IN)
secondary_address	FB_word	(IN)
buffer_mode	FB_buffer_mode	(IN)
buffer_specifier	FB_word	(OUT) (FBVAR)
	or FB_seq_buffer_id	(IN) (FBSEQ)
buffer_size	FB_integer	(IN)
max_bytes	FB_integer	(IN)

Action:

A broadcast primary address cycle using the parameter primary_address, and MS=2 (FRDTPS) or MS=3 (FRCTPS) shall be followed by a single-word read Data cycle with MS=0. The primary_address shall have the Global bit (bit 1) clear and shall be as specified for Case 3, 3a, 4, 5 or 6 in Table III of the FASTBUS standard.

For each bit set in the word read, a Geographical address is constructed consisting of the GP field of the specified primary_address, plus the encoded value of the position of a set bit in the data word. At each such address, a secondary address cycle using the value of the parameter secondary_address, followed by a block transfer read data Operation of no more than max_bytes, shall be carried out. The data from successive block transfers shall be concatenated in the specified Data Buffer.

7.4.3 Allocate a FASTBUS Device

Name: FDVAL or FB_DEVICE_ALLOCATE

Category: E, Action Routine

Parameters:

return_code	FB_error_code	(OUT)
environment_id	FB_environment_id	(IN)
primary_address	FB_word	(IN)

Action:

A primary address cycle to CSR space shall be followed by a secondary address cycle specifying address 4. A single-word write data cycle is then performed with the Port's own FASTBUS address as data. The considerations in clause 6.1 shall apply, except that the secondary address value is always 4.

7.4.4 Deallocate a FASTBUS Device

Name: FDVDE or FB_DEVICE_DEALLOCATE

Category: E, Action Routine

Paramètres:

return_code	FB_error_code	(OUT)
environment_id	FB_environment_id	(IN)
primary_address	FB_word	(IN)

Action:

Un cycle d'adressage primaire dans l'espace CSR doit être suivi par un cycle d'adresse secondaire spécifiant l'adresse 0. Un cycle d'écriture d'un mot unique de données est ensuite exécuté avec la valeur 80000 (hexadécimal). Les considérations de l'article 6.1 s'appliquent à l'exception de la valeur de l'adresse secondaire qui est toujours 0.

7.5 Envoi d'un message d'interruption FASTBUS

Nom: FBFIM ou FB_SEND_FIM

Catégorie: E, Routine d'action

Paramètres:

return_code	FB_error_code	(OUT)
environment_id	FB_environment_id	(IN)
primary_address	FB_word	(IN)
receiver_bloc_num	FB_word	(IN)
buffer_mode	FB_buffer_mode	(IN)
buffer_specifier	array_of FB word	(IN) (FBVAR)
	ou FB seq buffer_id	(IN) (FBSEQ)
message_len_bytes	FB integer	(IN)

Action:

Cette routine provoque l'exécution d'un transfert de données en bloc, comme c'est décrit au paragraphe 6.2.2, d'une longueur égale à message_len_bytes/4 de mots de 32 bits.

L'adressage primaire doit être dans l'espace CSR et peut être un adressage géographique, logique ou de diffusion. L'adresse secondaire doit être construite par la routine à partir de la valeur spécifiée par le numéro du bloc de réception d'interruption (voir la norme FASTBUS, article 8.2).

Le message doit être copié par la réalisation dans un tampon interne, et les quatre bits de poids faible du premier mot de données doivent être remplacés par la valeur message_len_bytes/4-1, le nombre de mots de données qui suivent dans le transfert.

La réalisation doit remplacer le deuxième mot du message par l'adresse du maître qui envoie le message.

Parameters:

return_code	FB_error_code	(OUT)
environment_id	FB_environment_id	(IN)
primary_address	FB_word	(IN)

Action:

A primary address cycle to CSR space shall be followed by a secondary address cycle specifying address 0. A single word write data cycle with data 80000 (hexadecimal) shall then be carried out. The considerations in clause 6.1 shall apply, except that the secondary address value is always 0.

7.5 Send a FASTBUS Interrupt Message

Name: FBFIM or FB_SEND_FIM

Category: E, Action Routine

Parameters:

return_code	FB_error_code	(OUT)
environment_id	FB_environment_id	(IN)
primary_address	FB_word	(IN)
receiver_block_num	FB_word	(IN)
buffer_mode	FB_buffer_mode	(IN)
buffer_specifier	array_of_FB_word	(IN) (FBVAR)
	or FB_seq_buffer_id	(IN) (FBSEQ)
message_len_bytes	FB_integer	(IN)

Action:

This routine shall cause a block data transfer, exactly as described in subclause 6.2.2, of length = message_len_bytes/4 32-bit words, to be performed.

The primary address shall be in CSR space and may be a Geographical, Logical or Broadcast Address. The secondary address shall be constructed by the routine from the specified value of the Interrupt Receiver Block Number (see FASTBUS standard, clause 8.2).

The message shall be copied by the Implementation to an internal buffer, and the low-order 4 bits of the first data word shall be replaced by the value of message_len_bytes/4-1, the number of data words to follow in the transfer.

The Implementation should replace the second word of the message with the address of the Master sending the message.

Section 8. Routines primitives d'action FASTBUS

8.1 Introduction

Les routines primitives d'action FASTBUS sont divisées en deux groupes. Les routines de cycle individuel sont traitées à l'article 8.3 et les routines manipulant les lignes individuelles sont traitées à l'article 8.4.

Après son exécution, une primitive d'action FASTBUS peut laisser le FASTBUS dans un état où il ne peut être utilisé par d'autres utilisateurs (voir paragraphe 2.2.5).

8.2 Paramètres

Les paramètres opérationnels suivants (décrits à la section 4) ne s'appliquent pas aux routines de cette section.

FB_PAR_NO_WAIT
 FB_PAR_NO_ARBITRATION
 FB_PAR_NO_PRIM_ADDR
 FB_PAR_NO_SEC_ADDR
 FB_PAR_NO_SEC_ADDR_CSR
 FB_PAR_NO_DATA_CYCLE
 FB_PAR_HOLD_BUS
 FB_PAR_HOLD_BUS_NO_AR
 FB_PAR_HOLD_AS

On ne doit relâcher, à la fin d'une primitive d'action FASTBUS, ni la maîtrise du bus, ni la connexion d'adresse sauf si cela est explicitement demandé par l'opération.

8.3 Routines de cycles individuels

Ces routines forment un ensemble complet à partir duquel on peut construire les routines de transaction.

Lorsqu'une routine de cycle individuel provoque une transition sur les lignes AS ou DS, elle doit attendre l'acceptation par les transitions AK ou DK et contrôler les codes SS associés.

8.3.1 Arbitrage

Nom: FCARB ou FB_CYCLE_ARBITRATE

Catégorie: C, Routine d'action

Paramètres:

return_code	FB_error_code	(OUT)
environment_id	FB_environment_id	(IN)

Action:

Cette routine provoque la demande d'arbitrage du maître pour la maîtrise du bus (voir article 6.1).

Section 8. Primitive FASTBUS Action Routines

8.1 Introduction

Primitive FASTBUS Action Routines are divided into two groups. Single cycle routines are dealt with in clause 8.3, and individual line handling routines are dealt with in clause 8.4.

On completion a Primitive FASTBUS Action can leave FASTBUS in a condition in which it cannot be used by other users (see subclause 2.2.5).

8.2 Parameters

The following Operational Parameters (described in section 4) do not apply to routines in this section:

FB_PAR_NO_WAIT
 FB_PAR_NO_ARBITRATION
 FB_PAR_NO_PRIM_ADDR
 FB_PAR_NO_SEC_ADDR
 FB_PAR_NO_SEC_ADDR_CSR
 FB_PAR_NO_DATA_CYCLE
 FB_PAR_HOLD_BUS
 FB_PAR_HOLD_BUS_NO_AR
 FB_PAR_HOLD_AS.

Neither FASTBUS mastership, nor any address connection shall be released at the end of a Primitive FASTBUS Action unless explicitly called for by the Operation.

8.3 Single Cycle Routines

These routines form a complete set out of which the Transaction routines can be constructed.

When a Single Cycle routine causes a transition on the AS or DS line, it shall wait for the acknowledging AK or DK transition, and check the associated SS code.

8.3.1 Arbitrate

Name: FCARB or FB_CYCLE_ARBITRATE

Category: C, Action Routine

Parameters:

return_code	FB_error_code	(OUT)
environment_id	FB_environment_id	(IN)

Action:

This routine shall cause the Master to arbitrate for bus mastership (see clause 6.1).

8.3.2 Release Bus Mastership

Name: FCREL or FB_CYCLE_RELEASE_BUS

Category: C, Action Routine

Parameters:

return_code FB_error_code (OUT)
environment_id FB_environment_id (IN)

Action:

This routine shall cause the GK line to be deasserted by the Master.

8.3.3 Primary Address Cycle

Names: FCPD or FB_CYCLE_PA_DAT Primary address to data space
FCPC or FB_CYCLE_PA_CSR Primary address to CSR space
FCPDM or FB_CYCLE_PA_DAT_MULT Primary address to data space
Multi-listener
FCPCM or FB_CYCLE_PA_CSR_MULT Primary address to CSR space
Multi-listener

Category: C, Action Routine

Parameters:

return_code FB_error_code (OUT)
environment_id FB_environment_id (IN)
buffer_mode FB_buffer_mode (IN)
buffer_specifier FB_word (IN) (FBVAR, FBVAL)
or FB_seq_buffer_id (IN) (FBSEQ)

Action:

A FASTBUS primary address cycle shall be performed as described in subclause 6.1.2 using data in the specified data buffer as the primary address, with the associated MS code depending on the routine called as follows:

Routine	MS
FB_CYCLE_PA_DAT	MS = 0
FB_CYCLE_PA_CSR	MS = 1
FB_CYCLE_PA_DAT_MULT	MS = 2
FB_CYCLE_PA_CSR_MULT	MS = 3

8.3.4 Disconnection

Name: FCDISC or FB_CYCLE_DISCONNECT

Category: C, Action Routine

Parameters:

return_code FB_error_code (OUT)
environment_id FB_environment_id (IN)

Action:

Tous les signaux doivent être retirés du bus par le maître, sauf GK.

8.3.5 Transfert d'un mot unique

Nom: FCRW ou FB_CYCLE_READ_WORD Lecture d'un mot unique
 FCRSA ou FB_CYCLE_READ_SA Lecture de l'adresse secondaire
 FCWW ou FB_CYCLE_WRITE_WORD Ecriture d'un mot unique
 FCWSA ou FB_CYCLE_WRITE_SA Ecriture de l'adresse secondaire

Catégorie: C, Routine d'action

Paramètres:

return_code	FB_error_code	(OUT)	
environment_id	FB_environment_id	(IN)	
buffer_mode	FB_buffer_mode	(IN)	
buffer_specifier	FB_word	(IN)	FCWxx (FBVAR, FBVAL)
	ou FB_word	(OUT)	FCRxx (FBVAR)
	ou FB_seq_buffer_id	(IN)	(FBSEQ)

Action:

Un cycle FASTBUS unique de données ou d'adresse secondaire doit être exécuté et les données transférées à partir du tampon de données ou dans celui-ci comme décrit au paragraphe 6.2.1. Les codes RD et MS associés dépendent comme suit de la routine appelée:

Routine	RD	MS
FB_CYCLE_READ_WORD	RD=1	MS=0
FB_CYCLE_READ_SA	RD=1	MS=2
FB_CYCLE_WRITE_WORD	RD=0	MS=0
FB_CYCLE_WRITE_SA	RD=0	MS=2

8.3.6 Transfert de bloc

Nom: FCRB ou FB_CYCLE_READ_BLOCK
 FCWB ou FB_CYCLE_WRITE_BLOCK

Catégorie: C, Routine d'action

Paramètres:

return_code	FB_error_code	(OUT)	
environment_id	FB_environment_id	(IN)	
buffer_mode	FB_buffer_mode	(IN)	
buffer_specifier	array_of_FB_word	(IN)	FCWB (FBVAR)
	ou array_of_FB_word	(OUT)	FCRB (FBVAR)
	ou FB_seq_buffer_id	(IN)	(FBSEQ)
max_bytes	FB_integer	(IN)	

Action:

Un transfert de bloc FASTBUS doit être exécuté comme décrit au paragraphe 6.2.2. La direction du transfert de données dépend comme suit de la routine appelée:

Action:

All signals except GK shall be removed from the bus by the Master.

8.3.5 Single Word Transfers

Name: FCRW or FB_CYCLE_READ_WORD Read single word
 FCRSA or FB_CYCLE_READ_SA Read secondary address
 FCWW or FB_CYCLE_WRITE_WORD Write single word
 FCWSA or FB_CYCLE_WRITE_SA Write secondary address

Category: C, Action Routine

Parameters:

return_code	FB_error_code	(OUT)	
environment_id	FB_environment_id	(IN)	
buffer_mode	FB_buffer_mode	(IN)	
buffer_specifier	FB_word	(IN)	FCWxx (FBVAR, FBVAL)
	or FB_word	(OUT)	FCRxx (FBVAR)
	or FB_seq_buffer_id	(IN)	(FBSEQ)

Action:

A single FASTBUS data or secondary address cycle shall be performed and data transferred to or from the Data Buffer as described in subclause 6.2.1. The associated RD and MS codes shall depend on the routine called as follows:

Routine	RD	MS
FB_CYCLE_READ_WORD	RD=1	MS=0
FB_CYCLE_READ_SA	RD=1	MS=2
FB_CYCLE_WRITE_WORD	RD=0	MS=0
FB_CYCLE_WRITE_SA	RD=0	MS=2

8.3.6 Block Transfers

Name: FCRB or FB_CYCLE_READ_BLOCK
 FCWB or FB_CYCLE_WRITE_BLOCK

Category: C, Action Routine

Parameters:

return_code	FB_error_code	(OUT)	
environment_id	FB_environment_id	(IN)	
buffer_mode	FB_buffer_mode	(IN)	
buffer_specifier	array_of_FB_word	(IN)	FCWB (FBVAR)
	or array_of_FB_word	(OUT)	FCRB (FBVAR)
	or FB_seq_buffer_id	(IN)	(FBSEQ)
max_bytes	FB_integer	(IN)	

Action:

A FASTBUS Block Transfer shall be performed as described in subclause 6.2.2. The data transfer direction shall depend on the routine called as follows:

Routine	RD
FB_CYCLE_READ_BLOCK	RD=0
FB_CYCLE_WRITE_BLOCK	RD=1

Le code MS utilisé dépend comme suit du paramètre FB_PAR_PIPELINE:

FB_PAR_PIPELINE = FB_FALSE	spécifie MS = 1
FB_PAR_PIPELINE = FB_TRUE	spécifie MS = 3

La longueur du transfert de données peut être déterminée en appelant la routine FB_READ_LENGTH décrite au paragraphe 6.2.4.

8.4 Accès aux lignes FASTBUS

Ces routines forment un ensemble complet, dans le sens qu'elles permettent d'exécuter toutes les opérations FASTBUS, incluant les fonctions de diagnostic et de test.

Lorsque les routines de cet article provoquent une transition sur les lignes d'échantillonnage AS et DS, il n'y a pas de contrôle implicite des lignes SS ou d'acceptation correspondantes.

Une réalisation qui fournit les routines FB_LINE_READ et FB_LINE_WRITE doit définir des noms pour toutes les lignes, cependant elle n'est pas obligée de permettre l'accès à chacune des lignes, et peut renvoyer un code d'erreur lorsque l'accès au signal FASTBUS spécifié n'est pas supporté.

Dans les routines ci-après, la gamme de valeurs qui peut être prise par le paramètre du type FB_integer est déterminée par la valeur du paramètre FB_LINE_, selon le tableau suivant:

Tableau 1 (page 1 / 2). Définition des types FB_LINE et de la gamme associée		
Nom de la ligne		Gamme de valeurs
Court	Long	
FBLAS	FB_LINE_AS	0, 1
FBLAK	FB_LINE_AK	0, 1
FBLEG	FB_LINE_EG	0, 1
FBLMS	FB_LINE_MS	0 ... 7
FBLAD	FB_LINE_AD	0 ... 2 ³² -1
FBLSS	FB_LINE_SS	0 ... 7
FBLDS	FB_LINE_DS	0, 1
FBLDK	FB_LINE_DK	0, 1
FBLRD	FB_LINE_RD	0, 1
FBLPE	FB_LINE_PE	0, 1
FBLPA	FB_LINE_PA	0, 1
FBLWT	FB_LINE_WT	0, 1
FBLAR	FB_LINE_AR	0, 1
FBLAG	FB_LINE_AG	0, 1
FBLAL	FB_LINE_AL	0 ... 63
FBLGK	FB_LINE_GK	0, 1

Routine	RD
FB_CYCLE_READ_BLOCK	RD=0
FB_CYCLE_WRITE_BLOCK	RD=1

The MS code used shall depend on the FB_PAR_PIPELINE parameter as follows:

FB_PAR_PIPELINE = FB_FALSE specifies MS = 1
 FB_PAR_PIPELINE = FB_TRUE specifies MS = 3

The length of the data transfer can be determined by calling routine FB_READ_LENGTH described in subclause 6.2.4.

8.4 FASTBUS Line Access

These routines form a complete set, in that they allow any FASTBUS Operation, including diagnostic and test functions, to be performed.

When routines in this clause cause a transition on the AS or DS strobe lines, there is no implied checking of the corresponding acknowledgement or SS lines.

An Implementation that provides the routines FB_LINE_READ and FB_LINE_WRITE shall define names for all the lines, however it need not support access to every line, and may return an error code indicating that access to the specified FASTBUS signal is unsupported.

In the routines below, the range of values which may be taken by the parameter of type FB integer is determined by the value of the parameter of type FB_LINE_, according to the following table:

Table 1 (Page 1 of 2). Definition of FB_LINE type and associated ranges		
Line name		Range of values
Short	Long	
FBLAS	FB_LINE_AS	0, 1
FBLAK	FB_LINE_AK	0, 1
FBLEG	FB_LINE_EG	0, 1
FBLMS	FB_LINE_MS	0 ... 7
FBLAD	FB_LINE_AD	0 ... 2 ³² -1
FBLSS	FB_LINE_SS	0 ... 7
FBLDS	FB_LINE_DS	0, 1
FBLDK	FB_LINE_DK	0, 1
FBLRD	FB_LINE_RD	0, 1
FBLPE	FB_LINE_PE	0, 1
FBLPA	FB_LINE_PA	0, 1
FBLWT	FB_LINE_WT	0, 1
FBLAR	FB_LINE_AR	0, 1
FBLAG	FB_LINE_AG	0, 1
FBLAL	FB_LINE_AL	0 ... 63
FBLGK	FB_LINE_GK	0, 1

Tableau 1 (page 2 / 2). Définition des types FB_LINE et de la gamme associée		
Nom de la ligne		Gamme de valeurs
Court	Long	
FBLAI	FB_LINE_AI	0, 1
FBLSR	FB_LINE_SR	0, 1
FBLRB	FB_LINE_RB	0, 1
FBLBH	FB_LINE_BH	0, 1
FBLGA	FB_LINE_GA	0 ... 31
FBLTP	FB_LINE_TP	0, 1

8.4.1 Lecture de ligne(s) individuelle(s) du bus

Nom: FLR ou FB_LINE_READ

Catégorie: C, Routine d'action

Paramètres:

return_code	FB_error_code	(OUT)	
environment_id	FB_environment_id	(IN)	
line	FB_line	(IN)	
buffer_mode	FB_buffer_mode	(IN)	
buffer_specifier	FB_word	(OUT)	(FBVAR)
	ou FB_seq_buffer_id	(IN)	(FBSEQ)

Action:

L'état actuel sur le FASTBUS de la ou des lignes référencées par le paramètre "line" doit être retourné vers le tampon de données spécifié.

8.4.2 Ecriture de ligne(s) individuelle(s) du bus

Nom: FLW ou FB_LINE_WRITE

Catégorie: C, Routine d'action

Paramètres:

return_code	FB_error_code	(OUT)	
environment_id	FB_environment_id	(IN)	
line	FB_line	(IN)	
buffer_mode	FB_buffer_mode	(IN)	
buffer_specifier	FB_word	(OUT)	(FBVAR, FBVAL)
	ou FB_seq_buffer_id	(IN)	(FBSEQ)

Action:

La ou les lignes référencées par le paramètre "line" doivent être positionnées suivant la valeur donnée par le tampon de données spécifié.

8.4.3 Lecture de ligne(s) interne(s)

Nom: FLRI ou FB_LINE_READ_INTERNAL

Catégorie: C, Routine d'action

Table 1 (Page 2 of 2). Definition of FB_LINE type and associated ranges		
Line name		Range of values
Short	Long	
FBLAI	FB_LINE_AI	0, 1
FBLSR	FB_LINE_SR	0, 1
FBLRB	FB_LINE_RB	0, 1
FBLBH	FB_LINE_BH	0, 1
FBLGA	FB_LINE_GA	0 ... 31
FBLTP	FB_LINE_TP	0, 1

8.4.1 Read Single Bus Line(s)

Name: FLR or FB_LINE_READ

Category: C, Action Routine

Parameters:

return_code	FB_error_code	(OUT)	
environment_id	FB_environment_id	(IN)	
line	FB_line	(IN)	
buffer_mode	FB_buffer_mode	(IN)	
buffer_specifier	FB_word	(OUT)	(FBVAR)
	or FB_seq_buffer_id	(IN)	(FBSEQ)

Action:

The current state on FASTBUS of the line(s) referred to by the "line" parameter shall be returned in the specified Data Buffer.

8.4.2 Write Single Bus Line(s)

Name: FLW or FB_LINE_WRITE

Category: C, Action Routine

Parameters:

return_code	FB_error_code	(OUT)	
environment_id	FB_environment_id	(IN)	
line	FB_line	(IN)	
buffer_mode	FB_buffer_mode	(IN)	
buffer_specifier	FB_word	(OUT)	(FBVAR, FBVAL)
	or FB_seq_buffer_id	(IN)	(FBSEQ)

Action:

The line(s) referred to by the line parameter shall assert the value given in the specified Data Buffer.

8.4.3 Read Internal Line(s)

Name: FLRI or FB_LINE_READ_INTERNAL

Category: C, Action Routine

Paramètres:

return_code	FB_error_code	(OUT)	
environment_id	FB_environment_id	(IN)	
line	FB_line	(IN)	
buffer_mode	FB_buffer_mode	(IN)	
buffer_specifier	FB_word	(OUT)	(FBVAR)
	ou FB_seq_buffer_id	(IN)	(FBSEQ)

Action:

L'état actuel des signaux vers ou produit par le port FASTBUS et qui correspondent à la ou aux lignes FASTBUS référencées par le paramètre "line" doit être retourné vers le tampon de données spécifié.

La signification exacte de la valeur retournée dépend de la réalisation.

Parameters:

return_code	FB_error_code	(OUT)	
environment_id	FB_environment_id	(IN)	
line	FB_line	(IN)	
buffer_mode	FB_buffer_mode	(IN)	
buffer_specifier	FB_word	(OUT)	(FBVAR)
	or FB_seq_buffer_id	(IN)	(FBSEQ)

Action:

The current state of a signal(s) internal to and generated by the FASTBUS Port and which is related to the FASTBUS line(s) referred to by the line parameter shall be returned in the specified Data Buffer.

The exact significance of the value returned is Implementation dependent.

Section 9. Routines des SR et des messages d'interruption FASTBUS

9.1 Introduction

Les routines de cette section fournissent un lien entre le logiciel dans un maître, et, soit la détection du positionnement du signal SR du FASTBUS, par le port FASTBUS associé, soit de la réception d'un message d'interruption FASTBUS.

Il y a deux protocoles standards d'interruption en FASTBUS:

1. Tout module peut émettre le signal de demande de service à n'importe quel moment; cela peut être détecté par un maître FASTBUS.
2. Un maître peut envoyer un message d'interruption FASTBUS d'un maximum de 16 mots dans le bloc de réception des interruptions dans un dispositif de réception.

(Cela est décrit en détail à la section 9 de la norme FASTBUS.)

9.2 Demande de service FASTBUS

Le signal de demande de service peut provoquer l'appel asynchrone d'une routine utilisateur (par une interruption du contrôleur) ou bien son état peut être scruté.

Le signal SR peut être scruté en utilisant la routine primitive FB_LINE_READ (FLR). Un utilisateur peut connecter une routine utilisateur qui sera appelée sans synchronisation avec le positionnement de SR.

Un des paramètres fourni pour cette connexion est "l'adresse de la source SR". Ce doit être l'adresse géographique du module positionnant SR pour qui la routine utilisateur doit être appelée, ou FB_SR_DEFAULT (voir plus loin).

La manière dont on détermine le dispositif qui positionne SR dépend du logiciel et du matériel. Cette norme n'exige pas que la réalisation soit capable de déterminer la source du SR, mais il est recommandé, si possible, qu'elle soit capable de le faire. Une réalisation qui a la connaissance de la topologie du système FASTBUS peut exécuter une scrutation des contacts "T". Autrement la réalisation peut scruter chaque adresse de source de SR pour laquelle une connexion a été faite, pour déterminer qui positionne SR (éventuellement plus d'un dispositif).

Pour autoriser le cas où la réalisation ne peut déterminer l'adresse de la source du SR, et pour l'utilisateur qui désire se connecter à tous les positionnements de SR, qu'elle qu'en soit la source, on doit disposer de la pseudo-adresse de source de SR: FB_SR_DEFAULT. Si une routine est connectée à cette adresse de source de SR, la routine utilisateur doit être appelée si l'on a trouvé aucun des dispositifs SR explicitement connectés positionnant SR, ou si la réalisation ne peut déterminer la source du positionnement de SR.

Des routines sont fournies qui permettent à l'utilisateur privilégié de mettre en ou hors service la connexion SR sur un port particulier. Par défaut, la connexion SR est en service dès qu'une connexion explicite est faite.

Section 9. FASTBUS SR and Interrupt Message Routines

9.1 Introduction

The routines in this section provide the link between the software in a Master, and either the detection by its associated FASTBUS Port of the assertion of the FASTBUS SR signal, or the receipt of a FASTBUS Interrupt Message.

There are two standard interrupt protocols in FASTBUS:

1. Any Module can raise the Service Request signal at any time; this may be detected by a FASTBUS Master.
2. A Master may send a FASTBUS Interrupt Message of up to 16 words in length to an Interrupt Receiver Block in a receiving Device.

(These are described in detail in the FASTBUS standard section 9.)

9.2 FASTBUS Service Request

The Service Request signal may cause a user routine to be called asynchronously (via an interrupt to the controller) or its state may be polled.

The SR signal may be polled using the primitive routine `FB_LINE_READ` (FLR). A User may connect a User routine to be called asynchronously on assertion of SR.

One of the parameters provided for this connection is the "SR Source Address". This shall be the Geographical address of the Module raising SR for which the User's routine is to be called, or `FB_SR_DEFAULT` (see below).

The process by which devices asserting SR is determined is hardware and software dependent. This standard does not require that the Implementation be able to determine the source of SR but it is recommended that, if possible, it should be able to do so. An Implementation with knowledge of the FASTBUS system topology can perform a T-pin Scan. Otherwise the implementation can poll each SR Source Address for which a connection has been made, to determine who is asserting the SR (possibly more than one device).

To allow for the case where the Implementation cannot determine the SR Source Address, and for the User who wants to connect to all assertions of SR, regardless of source, an SR Source Address of `FB_SR_DEFAULT` shall be provided. If a routine is connected to this SR Source Address, the User routine shall be called if none of the explicitly connected SR devices are found to be asserting SR, or the Implementation cannot determine the source of the SR assertion.

Routines are provided to allow the privileged User to enable or disable SR connections from a particular Port. SR connections are by default enabled, once the explicit connection is made.

9.2.1 Effacer le positionnement de SR

La réalisation devrait mettre hors service le positionnement de SR dans le dispositif avant d'appeler la routine connectée (en utilisant les bits de CSR#0 décrits dans la norme FASTBUS au paragraphe 8.3.2). C'est de la responsabilité de la routine connectée de remettre à zéro la source de SR et de remettre en service le positionnement de SR dans le dispositif.

9.2.2 Connecter une routine à SR

Nom: FBSRC ou FB_SR_CONNECT

Catégorie: I

Paramètres:

return_code	FB_error_code	(OUT)
SR_source address	FB_word	(IN)
port_id	FB_port_id	(IN)
routine	FB_routine	(IN)

Effets:

Lorsqu'un SR se produit et que la SR_source address spécifiée positionne SR, on doit appeler la routine indiquée si le port est en service. Si la SR_source address est indiquée comme FR_SR_DEFAULT, chaque détection d'un SR provenant de sources qui ne sont pas par ailleurs explicitement connectées sera passée à la routine indiquée.

Le paramètre passé à la routine connectée doit être le suivant:

SR_source address	FB_word	(IN)
-------------------	---------	------

SR_source address doit être l'adresse primaire de l'esclave qui positionne SR sauf si FB_SR_DEFAULT a été envoyé à FB_SR_CONNECT et que FB_SR_DEFAULT doit aussi être envoyé à la routine utilisateur.

9.2.3 Déconnecter une routine de SR

Nom: FBSRD ou FB_SR_DISCONNECT

Catégorie: I

Paramètres:

return_code	FB_error_code	(OUT)
SR_source_address	FB_word	(IN)
port_id	FB_port_id	(IN)

Effets:

Après que cette routine a été appelée, les connexions établies précédemment, pour le port_id spécifié, entre le positionnement de SR par un dispositif à l'adresse géographique SR_source_address et la routine utilisateur indiquée doivent être rompues. Ainsi, le positionnement de SR par le dispositif ne provoquera plus l'appel à la routine utilisateur indiquée. Si l'on spécifie la SR_source_address de FB_SR_SOURCE_ALL, toutes les connexions faites pour le port indiqué doivent être rompues.

9.2.1 Clearing the SR Assertion

The Implementation should disable the SR assertion in the device asserting SR before calling the connected routine (using CSR#0 bits described in the FASTBUS standard subclause 8.3.2). It is the responsibility of the connected routine to reset the SR source and reenale SR assertion in the device.

9.2.2 Connect Routine to SR

Name: FBSRC or FB_SR_CONNECT

Category: I

Parameters:

return_code	FB_error_code	(OUT)
SR_source_address	FB_word	(IN)
port_id	FB_port_id	(IN)
routine	FB_routine	(IN)

Effect:

When an SR occurs and the specified SR_source_address is asserting SR, the specified routine shall be called if the Port is enabled. If the SR_source_address is specified as FB_SR_DEFAULT, all detection of SR from sources that are not otherwise explicitly connected will be passed to the specified routine.

The parameter passed to the connected routine shall be the following:

SR_source_address	FB_word	(IN)
-------------------	---------	------

SR_source_address shall be the Primary Address of the slave asserting SR except that if FB_SR_DEFAULT was passed to FB_SR_CONNECT then FB_SR_DEFAULT shall also be passed to the user routine.

9.2.3 Disconnect Routine from SR

Name: FBSRD or FB_SR_DISCONNECT

Category: I

Parameters:

return_code	FB_error_code	(OUT)
SR_source_address	FB_word	(IN)
port_id	FB_port_id	(IN)

Effect:

After this routine has been called, the previously established connection for the specified port_id between the assertion of SR by the device at Geographical Address SR_source_address and a User specified routine shall be broken. Hence the assertion of SR by the device will not cause a User specified routine to be called. If an SR_source_address of FB_SR_SOURCE_ALL is specified, all such connections made for the specified Port shall be broken.

9.2.4 Mettre en service la connexion SR

Nom: FBSREN ou FB_SR_ENABLE

Catégorie: I

Paramètres:

return_code	FB_error_code	(OUT)
port_id	FB_port_id	(IN)

Effets:

Le port indiqué doit être mis en service pour répondre au signal SR par un appel aux routines connectées.

9.2.5 Mettre hors service la connexion SR

Nom: FBSRDS ou FB_SR_DISABLE

Catégorie: I

Paramètres:

return_code	FB_error_code	(OUT)
port_id	FB_port_id	(IN)

Effets:

Après que cette routine a été appelée, la réalisation ne doit plus appeler aucune routine connectée lorsque le positionnement de SR est détecté par le port_id indiqué. Tous les SR en attente doivent être ignorés.

9.3 Routines des messages d'interruption FASTBUS

Les routines suivantes sont utilisées pour déclarer une routine à appeler lors de la réception d'un message d'interruption FASTBUS sur un port FASTBUS. On suppose que l'utilisateur n'a pas l'accès direct au récepteur d'interruption FASTBUS. La réception des messages et la remise en service de receveur d'interruption sont faites par la réalisation, qui filtre et répartit les messages arrivés aux utilisateurs.

Si un port a été mis hors service pour la réception des messages d'interruption, aucune des routines connectées ne sera appelée. Par défaut, un port doit être mis en service pour recevoir les messages et les transférer aux routines connectées.

9.3.1 Connecter une routine à FIR

Nom: FBFIRC ou FB_FIR_CONNECT

Catégorie: I

Paramètres:

return_code	FB_error_code	(OUT)
port_id	FB_port_id	(IN)
receiver_block_num	FB_integer	(IN)
filter_mask	FB_word	(IN)
filter_value	FB_word	(IN)
filter_word_num	FB_integer	(IN)
routine	FB_routine	(IN)

9.2.4 Enable SR Connections

Name: FBSREN or FB_SR_ENABLE

Category: I

Parameters:

return_code	FB_error_code	(OUT)
port_id	FB_port_id	(IN)

Effect:

The specified Port shall be enabled to respond to the SR signal by invoking any connected routines.

9.2.5 Disable SR Connections

Name: FBSRDS or FB_SR_DISABLE

Category: I

Parameters:

return_code	FB_error_code	(OUT)
port_id	FB_port_id	(IN)

Effect:

After this routine has been called, the Implementation shall not call any connected routine because of the assertion of SR detected by the specified port_id. Any pending SRs shall be ignored.

9.3 FASTBUS Interrupt Message Routines

The following routines are used to declare a routine to be called on receipt of a FASTBUS interrupt message by the FASTBUS Port. It is assumed that the User does not have direct access to the FASTBUS interrupt receiver. The receiving of messages and reenabling of the Interrupt receiver are done by the Implementation, which filters and dispatches the incoming messages to the Users.

If a Port has been disabled from receiving Interrupt Messages, none of the connected routines will be called. By default a Port shall be enabled to receive the messages and transfer them to connected routines.

9.3.1 Connect Routine to FIR

Name: FBFIRC or FB_FIR_CONNECT

Category: I

Parameters:

return_code	FB_error_code	(OUT)
port_id	FB_port_id	(IN)
receiver_block_num	FB_integer	(IN)
filter_mask	FB_word	(IN)
filter_value	FB_word	(IN)
filter_word_num	FB_integer	(IN)
routine	FB_routine	(IN)

Effets:

Cette routine doit établir la connexion entre la routine spécifiée et le bloc de réception d'interruption spécifié du port désigné. Une valeur de 0 pour le receiver_block_num doit définir le bloc de réception des interruptions de CSR#100 à CSR#10F, une valeur de 1, le bloc de CSR#110 à CSR#11F, etc.

Lorsqu'un message est reçu par le bloc de réception d'interruption spécifié, le contenu du mot filter_word_num du message d'interruption doit être traité par une fonction "ET" avec le filter_mask et le résultat comparé à filter_value. Si les deux sont égaux, la routine doit être appelée, sinon aucune autre action n'est prise.

Toutes les routines connectées qui satisfont aux critères seront appelées dans un ordre qui dépend de la réalisation.

Les paramètres passés à la routine connectée sont les suivants:

receiver_block_no	FB_integer	(IN)
message_buffer	array_of_FB_word	(IN)
message_length_bytes	FB_integer	(IN)
port_id	FB_port_id	(IN)

9.3.2 Déconnecter une routine de FIR

Nom: FBFIRD ou FB_FIR_DISCONNECT

Catégorie: I

Paramètres:

return_code	FB_error_code	(OUT)
port_id	FB_port_id	(IN)
receiver_block_num	FB_integer	(IN)
filter_mask	FB_word	(IN)
filter_value	FB_word	(IN)
filter_word_num	FB_integer	(IN)
routine	FB_routine	(IN)

Effets:

Cette routine fait que la réalisation cesse d'associer la routine définie associée au filtre spécifié à la réception d'un message d'interruption dans le bloc de réception des messages d'interruption du port indiqué.

9.3.3 Mettre en service la connexion FIR

Nom: FBFIRE ou FB_FIR_ENABLE

Catégorie: I

Paramètres:

return_code	FB_error_code	(OUT)
port_id	FB_port_id	(IN)

Effets:

La réalisation doit être réactivée pour accepter des messages d'interruption FASTBUS sur le port indiqué par port_id et les passer aux routines connectées correspondantes.

Effect:

This routine shall make a connection between the specified routine and the specified interrupt receiver block of the identified Port. A value of 0 for receiver_block_num shall specify the interrupt receiver block at CSR#100 to CSR#10F, a value of 1, the block from CSR#110 to CSR#11F, etc.

When a message is received by the specified interrupt receiver block, the contents of the filter_word_num word of the interrupt message shall be ANDed with the filter_mask and the result compared with filter_value. If the two are equal, the routine shall be called, otherwise no further action is taken.

All connected routines that satisfy the criterion shall be called in an Implementation dependent order.

The parameters passed to the connected routine shall be the following:

receiver_block_no	FB_integer	(IN)
message_buffer	array_of_FB_word	(IN)
message_length_bytes	FB_integer	(IN)
port_id	FB_port_id	(IN)

9.3.2 Disconnect Routine from FIR

Name: FBFIRD or FB_FIR_DISCONNECT

Category: I

Parameters:

return_code	FB_error_code	(OUT)
port_id	FB_port_id	(IN)
receiver_block_num	FB_integer	(IN)
filter_mask	FB_word	(IN)
filter_value	FB_word	(IN)
filter_word_num	FB_integer	(IN)
routine	FB_routine	(IN)

Effect:

This routine shall cause the Implementation to no longer associate the specified routine associated with the specified filter, with the receipt of an interrupt message in the interrupt receiver block of the specified Port.

9.3.3 Enable FIR Connections

Name: FBFIRE or FB_FIR_ENABLE

Category: I

Parameters:

return_code	FB_error_code	(OUT)
port_id	FB_port_id	(IN)

Effect:

The Implementation shall be enabled to accept FASTBUS Interrupt Messages on the Port specified by port_id and pass them on to the appropriately connected routines.

9.3.4 Mettre hors service la connexion FIR

Nom: FBFIRS ou FB_FIR_DISABLE

Catégorie: I

Paramètres:

return_code	FB_error_code	(OUT)
port_id	FB_port_id	(IN)

Effets:

La réalisation met hors service les appels à toute routine utilisateur connectée résultant de messages reçus sur le port indiqué.

9.3.4 Disable FIR Connections

Name: FBFIRS or FB_FIR_DISABLE

Category: I

Parameters:

return_code	FB_error_code	(OUT)
port_id	FB_port_id	(IN)

Effect:

The Implementation is disabled from invoking any connected user routines as a result of a message being received by the specified Port.

Section 10. Synchronisation, ressources système et routines des ports

Les ressources système et les routines des ports fournissent un accès aux outils de synchronisation communs ou aux fonctions spécifiques au port d'une manière indépendante de la machine.

Remarquer que l'accès aux routines des ressources système affecteront généralement les autres utilisateurs sur des systèmes multi-utilisateurs, et peuvent nécessiter un certain type de privilège. L'erreur FB_ERR_NO_PRIVILEGE doit être renvoyée si l'utilisateur n'est pas autorisé à accéder à cette fonction.

10.1 Outils de synchronisation

Ces routines permettent à l'utilisateur de contrôler l'état d'une routine d'action ou d'appeler FB_LIST_EXECUTE. En général cette possibilité est très utile lorsque le paramètre opérationnel FB_PAR_NO_WAIT est FB_TRUE et lorsque le retour à la routine d'appel est effectué avant que l'action demandée ait été exécutée. L'utilisateur peut directement déclarer une routine, qui sera appelée à l'exécution, et un test ou attente pour l'exécution d'une routine d'action ou l'action de FB_LIST_EXECUTE.

10.1.1 Attendre la fin d'une opération

Nom: FCOMWT ou FB_COMPLETION_WAIT

Catégorie: S

Paramètres:

return_code	FB_error_code	(OUT)
environment_id	FB_environment_id	(IN)

Effets:

Cette routine doit attendre la fin de toutes les opérations associées à environment_id spécifié.

10.1.2 Contrôler l'état d'exécution

Nom: FCOMTS ou FB_COMPLETION_TEST

Catégorie: S

Paramètres:

return_code	FB_error_code	(OUT)
environment_id	FB_environment_id	(IN)
boolean_value	FB_boolean	(IN)

Effets:

Cette routine doit contrôler si toutes les opérations spécifiées pour l'environnement ont été terminées. Une valeur booléenne doit être retournée avec FB_TRUE si toutes les opérations sont terminées et FB_FALSE autrement.

Section 10. Synchronization, system resource and port Routines

System Resource and Port routines provide access to common synchronization tools or Port specific functions in a machine independent fashion.

Note that access to system resource routines will generally affect other Users on a multi-user system and may require some type of privilege. The error FB_ERR_NO_PRIVILEGE shall be returned if the User is not allowed access to the function.

10.1 Synchronization Tools

These routines allow a User to check on the status of an Action Routine or a call to FB_LIST_EXECUTE. In general, this capability is most useful when the Operational Parameter FB_PAR_NO_WAIT is FB_TRUE and when the return to the calling routine is made before the requested action has been completed. The User may directly declare a routine to be called on completion, and test or wait for the completion of an Action Routine or action of FB_LIST_EXECUTE.

10.1.1 Wait for Completion of Operation

Name: FCOMWT or FB_COMPLETION_WAIT

Category: S

Parameters:

return_code	FB_error_code	(OUT)
environment_id	FB_environment_id	(IN)

Effect:

This routine shall wait for completion of all Operations associated with the specified environment_id.

10.1.2 Check Completion State

Name: FCOMTS or FB_COMPLETION_TEST

Category: S

Parameters:

return_code	FB_error_code	(OUT)
environment_id	FB_environment_id	(IN)
boolean_value	FB_boolean	(IN)

Effect:

This routine shall check whether all Operations specified for the Environment have been completed. A boolean value shall be returned which is FB_TRUE if all Operations have been completed and FB_FALSE otherwise.

10.1.3 Connecter la routine de fin d'exécution

Nom: FBCOMC ou FB_COMPLETION_CONNECT

Catégorie: S

Paramètres:

return_code	FB_error_code	(OUT)
environment_id	FB_environment_id	(IN)
routine	FB_routine	(IN)

Effets:

Cette routine réalise la connexion entre la routine indiquée et la fin de l'exécution de la routine d'action ou de la liste d'actions. Une seule routine de fin d'exécution peut être connectée à un instant donné. La tentative de connecter une autre routine doit renvoyer une erreur.

10.1.4 Déconnecter la routine de fin d'exécution

Nom: FBCOMD ou FB_COMPLETION_DISCONNECT

Catégorie: S

Paramètres:

return_code	FB_error_code	(OUT)
environment_id	FB_environment_id	(IN)

Effets:

Cette routine fait que la réalisation n'associe pas plus longtemps la routine connectée à la fin d'exécution de la routine d'action ou de la liste d'actions.

10.2 Réinitialiser un port FASTBUS

Nom: FBPRST ou FB_PORT_RESET

Catégorie: A, Routine d'action

Paramètres:

return_code	FB_error_code	(OUT)
port_id	FB_port_id	(IN)

Effets:

Cette routine doit réinitialiser le port FASTBUS. Les effets détaillés de cette réinitialisation sont définis par la réalisation.

Tous les signaux positionnés sur le bus par le port seront retirés.

Réinitialiser un port affectera les autres utilisateurs. Normalement, dans un système multi-utilisateurs, pour pouvoir exécuter avec succès cette routine, l'utilisateur aura besoin de privilège.

10.1.3 Connect Completion Routine

Name: FBCOMC or FB_COMPLETION_CONNECT

Category: S

Parameters:

return_code	FB_error_code	(OUT)
environment_id	FB_environment_id	(IN)
routine	FB_routine	(IN)

Effect:

This routine shall make a connection between the specified routine and the completion of an Action Routine or List of actions. Only one completion routine may be connected at any time. An attempt to connect another routine shall return with an error.

10.1.4 Disconnect Completion Routine

Name: FBCOMD or FB_COMPLETION_DISCONNECT

Category: S

Parameters:

return_code	FB_error_code	(OUT)
environment_id	FB_environment_id	(IN)

Effect:

This routine shall cause the Implementation to no longer associate the connected routine with the completion of an Action Routine or List of actions.

10.2 Reset FASTBUS Port

Name: FBPRST or FB_PORT_RESET

Category: A, Action Routine

Parameters:

return_code	FB_error_code	(OUT)
port_id	FB_port_id	(IN)

Effect:

This routine shall reset the FASTBUS Port. The detailed effect of this reset shall be defined by the implementation.

All signals asserted by the Port on the bus shall be deasserted.

Resetting a Port will affect other Users. In a multi-user system the User will usually require privileges in order to successfully execute this routine.

10.3 Allocation d'un port FASTBUS

Dans un environnement multi-utilisateurs, un utilisateur peut utiliser les routines suivantes pour allouer et désallouer l'usage exclusif d'un port.

10.3.1 Allouer un port FASTBUS

Nom: FNPALL ou FB_PORT_ALLOCATE

Catégorie: S, I

Paramètres:

return_code	FB_error_code	(OUT)
port_id	FB_port_id	(IN)

Effets:

Cette routine doit tenter d'allouer l'exclusivité du port indiqué à l'utilisateur associé à la routine d'appel. En cas d'insuccès, une erreur doit être renvoyée. L'attribution d'un port affectera les autres utilisateurs qui ne pourront plus y accéder.

10.3.2 Désallouer un port FASTBUS

Nom: FNPDEA ou FB_PORT_DEALLOCATE

Catégorie: S, I

Paramètres:

return_code	FB_error_code	(OUT)
port_id	FB_port_id	(IN)

Effets:

Cette routine doit désallouer le port indiqué.

10.3.3 Allouer un port FASTBUS avec temporisateur

Nom: FNPALT ou FB_PORT_ALLOCATE_TMO

Catégorie: S, I

Paramètres:

return_code	FB_error_code	(OUT)
port_id	FB_port_id	(IN)
timeout_parameter	temporisation en millisecondes	

Effets:

Cette routine doit tenter d'allouer l'exclusivité du port indiqué au processus associé à la routine d'appel. En cas d'insuccès au bout du temps spécifié, une erreur doit être renvoyée.

10.4 Obtenir le numéro de version

Nom: FBVERS ou FB_GET_SOFTWARE_VERSION

Catégorie: A

10.3 FASTBUS Port Allocation

In a multi-user Environment, the following routines can be used by a user to allocate and deallocate a Port for its exclusive use.

10.3.1 Allocate FASTBUS Port

Name: FNPALL or FB_PORT_ALLOCATE

Category: S, I

Parameters:

return_code	FB_error_code	(OUT)
port_id	FB_port_id	(IN)

Effect:

This routine shall attempt to allocate exclusively the specified Port to the User associated with the calling routine. If unsuccessful, an error shall be returned. Allocating a port will affect other Users by preventing their access to it.

10.3.2 Deallocate FASTBUS Port

Name: FNPDEA or FB_PORT_DEALLOCATE

Category: S, I

Parameters:

return_code	FB_error_code	(OUT)
port_id	FB_port_id	(IN)

Effect:

This routine shall deallocate the specified Port.

10.3.3 Allocate FASTBUS Port with Timeout

Name: FNPALT or FB_PORT_ALLOCATE_TMO

Category: S, I

Parameters:

return_code	FB_error_code	(OUT)
port_id	FB_port_id	(IN)
timeout_parameter	timeout time in milliseconds	

Effect:

This routine shall attempt to allocate exclusively the specified Port to the User associated with the calling routine. If unsuccessful within the specified time, an error shall be returned.

10.4 Get Version Numbers

Name: FBVERS or FB_GET_SOFTWARE_VERSION

Category: A

Paramètres:

return_code	FB_error_code	(OUT)
port_id	FB_port_id	(IN)
hw_type	FB_word	(OUT)
hw_version	FB_word	(OUT)
sw_type	FB_word	(OUT)
sw_version	FB_word	(OUT)

Effets:

Cette routine doit renvoyer le type et le numéro de version du matériel et du logiciel utilisés sur le port indiqué.

Le type du matériel doit être fixé égal à l'ID FASTBUS du port du dispositif. La version matérielle peut être assignée par le réalisateur.

Le type du logiciel doit être assigné d'une manière semblable à l'ID du dispositif comme cela est défini dans la norme FASTBUS au paragraphe 8.3.1. Ainsi, les valeurs pour les bits de poids fort sur les 16 bits de l'ID logiciel seront affectés par le président du Comité NIM aux organisations qui le demandent, lesquelles assigneront les valeurs.

La version logicielle est affectée par le réalisateur et doit être incluse dans la documentation qui accompagne la réalisation.

10.5 Rapporter une erreur du port FASTBUS

Nom: FBPRPT ou FB_PORT_STATUS_REPORT

Catégorie: E

Paramètres:

return_code	FB_error_code	(OUT)
port_id	FB_port_id	(IN)
error_code	FB_error_code	(IN)

Effets:

Un message d'erreur sera émis à partir du paramètre error_code. On prendra des mesures selon la réalisation pour attirer l'attention de l'utilisateur.

10.6 Emettre un message d'erreur du port FASTBUS

Nom: FBPTRN ou FB_PORT_STATUS_TRANSLATE

Catégorie: E

Paramètres:

return_code	FB_error_code	(OUT)
port_id	FB_port_id	(IN)
error_code	FB_error_code	(IN)
message	FB_string	(OUT)

Effets:

Un message d'erreur sera émis à partir du paramètre error_code. Le résultat est renvoyé dans le paramètre message.

Parameters:

return_code	FB_error_code	(OUT)
port_id	FB_port_id	(IN)
hw_type	FB_word	(OUT)
hw_version	FB_word	(OUT)
sw_type	FB_word	(OUT)
sw_version	FB_word	(OUT)

Effect:

This routine shall return the type and the version number of the hardware and software in use at the specified Port.

The hardware type should be set equal to the FASTBUS ID of the Port device. The hardware version may be assigned by the implementer.

Software types are allocated in a similar manner to Device IDs as defined in the FASTBUS standard subclause 8.3.1. That is, values for the most significant bits of a 16 bit software ID will be assigned by the Chairman of the NIM Committee to requesting organizations, who will then assign values.

The software version is assigned by the implementer and shall be included in the documentation that accompanies the Implementation.

10.5 Report a Port FASTBUS Error

Name: FBPRPT or FB_PORT_STATUS_REPORT

Category: E

Parameters:

return_code	FB_error_code	(OUT)
port_id	FB_port_id	(IN)
error_code	FB_error_code	(IN)

Effect:

An error message shall be generated from the error_code parameter. Some Implementation dependent action shall be taken to bring this to the user attention.

10.6 Generate a Port Error Message

Name: FBPTRN or FB_PORT_STATUS_TRANSLATE

Category: E

Parameters:

return_code	FB_error_code	(OUT)
port_id	FB_port_id	(IN)
error_code	FB_error_code	(IN)
message	FB_string	(OUT)

Effect:

An error message shall be generated from the error_code parameter. The result is returned in the message parameter.

Section 11. Traitement des états et des erreurs

11.1 Introduction

Cette section décrit les routines de contrôle du traitement des erreurs rencontrées pendant l'exécution des routines standards et pour extraire l'information détaillée sur les conditions dans lesquelles elles se sont produites.

11.2 Codes d'erreur et de retour

L'appel de chaque routine de cette norme doit provoquer la génération d'un code d'erreur qui consiste en un code d'état et une sévérité.

Un code d'état définit en termes généraux le résultat de l'exécution d'une routine. Les codes d'état indiquent que la routine s'est exécutée normalement ou que des difficultés identifiées se sont produites. En plus, les codes d'état fournissent des informations sur les détails et les conditions finales de l'action exécutée. La "sévérité" d'un code d'erreur est utilisée comme déclenchement pour différentes actions de rapport automatique des erreurs. Pratiquement toutes les routines de cette norme produisent un "code de retour" qui, par défaut, est un code d'erreur et qui contient ainsi à la fois les informations d'état et de sévérité (paragraphe 4.4.6). La sévérité peut être changée de sa valeur par défaut par un appel à `FB_STATUS_SET_SEVERITY` ou par le résultat d'une réponse d'erreur (voir ci-dessous). Sauf pour des modifications de la sévérité, le code d'erreur émis pour un état résumé devrait être un de ceux de la section 12.

11.3 Etats résumés et complémentaires

En plus du code de retour, dans le mode d'exécution immédiate, les routines d'action produisent des informations d'état consistant en un code d'erreur, un nombre total d'octets lus et un nombre total d'octets écrits par la routine. Cette information est référencée comme "l'état résumé". (L'un ou les deux comptes d'octets peuvent ne pas être appropriés pour une routine particulière.)

Un état résumé doit toujours être disponible à travers un appel à `FB_STATUS_GET_SUMMARY`.

Pour chaque routine d'action, des informations supplémentaires d'état peuvent être accumulées et mémorisées en tant que "état complémentaire" associé à l'état résumé. Pratiquement, l'état complémentaire dépendra du contrôleur et de la réalisation.

Chaque information de l'état complémentaire doit contenir un code d'erreur.

Les éléments optionnels disponibles comme informations d'état complémentaires incluent un paramètre numérique associé qualifiant le code d'erreur et un numéro d'index qui, d'une manière dépendant de la réalisation, indique la partie de l'action de la routine qui est rapportée.

La sévérité de l'état résumé doit être égale ou plus sévère que celle contenue dans chaque état complémentaire associé.

Cet ensemble des états résumés et complémentaires est appelé "l'historique d'état".

Section 11. Status and error handling

11.1 Introduction

This section describes routines for controlling the handling of errors encountered during execution of the standard routines and for extracting detailed information about conditions that have occurred.

11.2 Error and Return Codes

The calling of each routine in this standard shall cause an "Error Code" to be generated which consists of a Status Code and its Severity.

A Status Code defines in general terms the result of a routine's execution. Status codes indicate whether the routine is executed normally or whether some identified difficulty occurred. In addition Status Codes provide information on the details and final conditions of the Action performed. The "Severity" of an Error Code is used as a trigger for various automatic error reporting actions. Nearly every routine in this standard generates a "Return Code" which, by default, is an Error Code and thus contains both Status and Severity information (subclause 4.4.6). The Severity may be changed from its default value by a call to `FB_STATUS_SET_SEVERITY` or by the result of an error response (see below). Apart from modifications to the Severity, the Error Code generated for a Summary Status should be one of those in section 12.

11.3 Summary and Supplementary Status

In addition to a Return Code, in Immediate Execution Mode the Action Routines produce status information consisting of an Error Code, the total number of bytes read and the total number of bytes written by the routine. This information is referred to as the "Summary Status". (One or both of the byte counts may not be relevant to any particular routine).

A Summary Status shall always be available through a call to `FB_STATUS_GET_SUMMARY`.

Additional status information for each Action Routine may be gathered and stored as "Supplementary Status" associated with the Summary Status. The Supplementary Status will typically be Controller and Implementation dependent.

Each supplementary status item shall contain an Error Code.

Optional items available as supplementary status information include an associated numeric parameter qualifying the Error Code and an index number which, in an Implementation dependent way, indicates the part of the routine's action that is being reported on.

The Severity of the Summary Status shall be equal to or more severe than that contained in any associated Supplementary Status.

This set of Summary and Supplementary Status is known as the "Status History".

Le rapport automatique des erreurs (voir ci-dessous) utilise le niveau de sévérité dans l'état résumé comme les informations de tout état complémentaire pour déterminer ce qui doit ou non être rapporté. On dispose de routines pour fixer les détails disponibles dans l'historique d'état.

11.4 Génération limitée d'informations d'état

Le paramètre opérationnel FB_PAR_NO_STATUS, lorsqu'il est FB_TRUE, doit limiter l'accumulation d'informations d'état au seul état résumé pour accélérer l'exécution des actions. Aucun état complémentaire n'est mémorisé.

Certaines réalisations peuvent ne pas être capables de fournir des états complémentaires. Pour de telles réalisations, le paramètre opérationnel FB_PAR_NO_STATUS devrait être un paramètre en lecture seule dont la valeur par défaut est FB_TRUE.

11.5 Etat en mode d'exécution différée

Lorsqu'une routine d'action quelconque, sauf pour FB_LIST_EXECUTE, est appelée dans un environnement de mode d'exécution différée, il n'est émis qu'un code de retour. Aucun historique d'état n'est produit par l'appel à la routine.

L'exécution d'une liste, par l'appel à FB_LIST_EXECUTE, doit provoquer la génération de l'état résumé qui doit contenir le code d'erreur de la première apparition de l'erreur de plus haute sévérité rencontrée.

En plus de produire cet état résumé, l'exécution d'une liste peut également, si le paramètre opérationnel FB_PAR_NO_STATUS est FB_FALSE, produire un état complémentaire pour chaque action de la liste.

11.6 Sévérité

Chaque code d'erreur inclut une sévérité associée. Les sévérités définies en ordre de sévérité croissante doivent être comme suit:

Nom long de sévérité	Nom court de sévérité
FB_SEV_SUCCESS	FSSUCC
FB_SEV_INFO	FSINFO
FB_SEV_WARNING	FSWARN
FB_SEV_ERROR	FSERR
FB_SEV_FATAL	FSFTL
FB_SEV_NEVER	FSNEV

11.6.1 FB_SEV_SUCCESS (FSSUCC)

La fonction demandée par la routine a été exécutée avec succès.

11.6.2 FB_SEV_INFO (FSINFO)

La fonction demandée par la routine a été exécutée, mais avec certaines qualifications, par exemple, à la fin de l'activité FASTBUS demandée, la maîtrise du bus a été conservée.

The automatic error reporter (see below) makes use of the Severity level in the Summary Status as well as information in any Supplementary Status to determine what is and what is not reported. There are routines supplied to determine the details available from the Status History.

11.4 Restricting Generation of Status Information

The Operational Parameter FB_PAR_NO_STATUS, when FB_TRUE, shall restrict the accumulation of status information to only the Summary Status in order to speed the execution of actions. No Supplementary Status is stored.

Some Implementations may not be able to generate any Supplementary Status. For such Implementations the Operational Parameter FB_PAR_NO_STATUS should be made a read-only parameter whose default is FB_TRUE.

11.5 Delayed Mode Execution Status

When any Action Routine, except for FB_LIST_EXECUTE, is called with a Delayed Execution Mode Environment, it generates only a Return Code. No Status History is generated by the call to the routine.

The execution of a List, through a call to FB_LIST_EXECUTE, shall result in the generation of Summary Status which shall contain the error code of the first occurrence of the highest Severity error encountered.

In addition to producing this Summary Status, List execution may also, if the FB_PAR_NO_STATUS Operational Parameter is FB_FALSE, generate Supplementary Status for each action in the List.

11.6 Severity

Each Error Code includes an associated Severity. The defined Severities in order of increasing Severity shall be as follows:

Severity Long Name	Severity Short Name
FB_SEV_SUCCESS	FSSUCC
FB_SEV_INFO	FSINFO
FB_SEV_WARNING	FSWARN
FB_SEV_ERROR	FSERR
FB_SEV_FATAL	FSFTL
FB_SEV_NEVER	FSNEV

11.6.1 FB_SEV_SUCCESS (FSSUCC)

The function requested of the routine was carried out successfully.

11.6.2 FB_SEV_INFO (FSINFO)

The function requested of the routine was carried out, but with some qualification. For example, at the termination of the requested FASTBUS activity bus mastership was maintained.

11.6.3 FB_SEV_WARNING (FSWARN)

La fonction demandée par la routine a été terminée en dépit de l'apparition de quelque(s) erreur(s). L'erreur ne rend pas un traitement ultérieur sans signification ou impossible. Par exemple, le changement demandé d'un paramètre de temporisation n'a pas été exécuté car la modification n'était pas supportée par le matériel.

11.6.4 FB_SEV_ERROR (FSERR)

La fonction demandée n'a pas été exécutée avec succès. Typiquement, l'erreur était d'un type qui rend les appels ou les actions futurs du FASTBUS impossibles ou leurs résultats imprévisibles. Par exemple, un cycle d'adressage primaire a reçu une réponse SS non nulle ou un paramètre passé était illégal.

11.6.5 FB_SEV_FATAL (FSFTL)

La fonction demandée n'a pas été exécutée avec succès. Cet échec était suffisamment important pour justifier l'arrêt du programme.

11.6.6 FB_SEV_NEVER (FSNEV)

Une des utilisations de la hiérarchie des sévérités est de fournir une méthode pour appliquer des seuils ajustables pour l'initiation de certaines actions spécifiées.

Cette sévérité, *FB_SEV_NEVER*, doit être disponible pour que les conditions de seuil ne soient jamais remplies.

11.7 Régler la sévérité d'un code d'erreur

La routine suivante sera utilisée pour modifier la sévérité qui doit être associée ultérieurement à un code d'erreur à la place de sa valeur courante ou par défaut.

En mode d'exécution différée, la sévérité associée au code d'erreur doit être celle qui est effective lorsque la liste *FB_LIST_EXECUTE* est appelée, plutôt que lorsqu'elle fut construite.

Nom: FSSSEV ou FB_STATUS_SET_SEVERITY

Catégorie: E

Paramètres:

return_code	FB_error_code	(OUT)
environment_id	FB_environment_id	(IN)
error_code	FB_error_code	(IN)
severity	FB_severity	(IN)

Effets:

La sévérité associée au code d'erreur spécifié doit être fixée à la valeur donnée. Cette valeur surcharge la sévérité assignée par défaut. Les sévérités par défaut associées à chaque code d'erreur sont présentées à la section 12.

Les codes d'erreurs dont la sévérité a été modifiée par l'action de la réponse d'erreur doivent avoir la sévérité la moins sévère entre la dernière réponse et celle fixée par cette routine.

Les codes d'erreur peuvent ne pas tous avoir leur sévérité altérée. Un essai pour le faire provoquera le retour d'une erreur.

11.6.3 FB_SEV_WARNING (FSWARN)

The function requested of the routine was completed despite the occurrence of some error(s). The error did not make further processing meaningless or impossible. For example, a requested change of a timeout parameter was not carried out because the change was not supported by the hardware.

11.6.4 FB_SEV_ERROR (FSERR)

The function requested was not carried out successfully. The error was, typically, of a type which makes future FASTBUS calls or actions impossible or their results unpredictable. For example, a primary address cycle received a non-zero SS response or a passed parameter was illegal.

11.6.5 FB_SEV_FATAL (FSFTL)

The function requested was not carried out successfully. This failure was of sufficient importance to warrant termination of the program.

11.6.6 FB_SEV_NEVER (FSNEV)

One of the uses of a hierarchy of Severities is to supply a method of applying an adjustable threshold for the initiation of certain specified actions.

This Severity, FB_SEV_NEVER, shall be provided so that a threshold condition can never be met.

11.7 Set the Severity of an Error Code

The following routine shall be used to change the Severity to be subsequently associated with an Error Code from its current or default value.

In Delayed Execution Mode, the Severity associated with an Error Code shall be the current Severity when FB_LIST_EXECUTE is called, rather than when the list was built.

Name: FSSSEV or FB_STATUS_SET_SEVERITY

Category: E

Parameters:

return_code	FB_error_code	(OUT)
environment_id	FB_environment_id	(IN)
error_code	FB_error_code	(IN)
severity	FB_severity	(IN)

Effect:

The Severity associated with the specified Error Code shall be set to the value given. This value overrides the default severity assigned. The default Severity associated with each Error Code is tabulated in section 12.

Error Codes whose Severity has been modified by the action of the error response shall have the least severe of the severities from the latter response and that set by this routine.

Not all Error Codes may have their Severity altered. An attempt to do so will result in an error being returned.

11.8 Réponses aux erreurs à l'exécution

Pendant l'exécution de chaque cycle FASTBUS, le maître peut recevoir l'un des huit codes d'état possibles de l'esclave; suivant les circonstances, il peut ou non les considérer comme une erreur. Il est également possible que le cycle ne se termine pas ou même ne démarre pas à cause de l'absence de retour au maître de la réponse au dialogue dans la période du temporisateur, ou que quelque autre temporisateur matériel ou logiciel ait déclenché avant que le cycle ou l'opération soit achevé. Les "codes de réponse" sont utilisés pour contrôler la réponse à de telles conditions. Les réponses spécifiques à des conditions matérielles données devraient inclure celles définies ci-dessous.

Les réponses définies doivent être:

Codes de réponse	
Nom long	Nom court
FB_RESP_IGNORE	FBRIG
FB_RESP_ABORT	FBRAB
FB_RESP_RETRY_ABORT	FBRAB
FB_RESP_RETRY_IGNORE	FBRIG
FB_RESP_ABORT_ACTION	FBRABA
FB_RESP_RETRY_ABORT_ACTION	FBRRAA

En mode d'exécution différée, lorsqu'une liste d'actions peut être exécutée, l'activation des réponses FB_RESP_ABORT et FB_RESP_RETRY_ABORT engendre la fin de la liste d'actions. L'activation des codes de réponse supplémentaires, FB_RESP_ABORT_ACTION et FB_RESP_RETRY_ABORT_ACTION, engendre la fin de l'action qui a rencontré une condition d'erreur, mais avec la continuation de la liste d'actions.

En mode d'exécution immédiate, FB_RESP_ABORT_ACTION se comporte comme FB_RESP_ABORT et FB_RESP_RETRY_ABORT_ACTION se comporte comme FB_RESP_RETRY_ABORT. Cependant, comme cela est analysé ci-après, la sévérité du code d'erreur qui en résultera sera différente.

11.8.1 FB_RESP_IGNORE

Si une condition matérielle se produit dont le code de réponse associé est FB_RESP_IGNORE, la réalisation doit essayer de continuer l'action comme si aucune erreur ne s'était produite.

En général, l'erreur rencontrée peut empêcher toute continuation d'avoir un sens.

11.8.2 FB_RESP_ABORT

Cela doit être la réponse par défaut d'une réalisation à toutes les conditions d'erreur matérielle sauf pour SS=0 et, dans les transferts de bloc, SS=2. Lorsque cela est le code de réponse spécifié, lors de la détection de condition matérielle, l'action ou la liste d'actions est abandonnée.

11.8.3 FB_RESP_RETRY_ABORT

Lorsqu'une condition matérielle se produit pour laquelle cela est le code de réponse spécifié, l'action FASTBUS qui a provoqué l'erreur est réessayée. Si, après le nombre d'essais spécifié par la valeur courante du paramètre opérationnel FB_PAR_NUM_RETRY, la condition d'erreur est toujours présente, l'action ou la liste d'actions est abandonnée.

11.8 Response to Errors in Execution

During the execution of each FASTBUS cycle one of up to eight possible Slave Status codes will be received by the Master which, depending on the circumstances, may or may not be considered to be an error. It is also possible that the cycle fails to complete or even start because of a failure of the Master to receive the return part of the handshake within the timeout period, or that some other hardware or software timer period expired before the cycle or Operation completed. The "Response Codes" are used to control the response to such conditions. The specified response to a given hardware condition should include those defined below.

The defined responses shall be:

Response Codes	
Long Name	Short Name
FB_RESP_IGNORE	FBRIG
FB_RESP_ABORT	FBRAB
FB_RESP_RETRY_ABORT	FBRAB
FB_RESP_RETRY_IGNORE	FBRIG
FB_RESP_ABORT_ACTION	FBRABA
FB_RESP_RETRY_ABORT_ACTION	FBRRAA

In Delayed Execution Mode, when a List of actions may be executed, activation of the responses FB_RESP_ABORT and FB_RESP_RETRY_ABORT results in termination of the List of actions. Activation of the extra Response Codes, FB_RESP_ABORT_ACTION and FB_RESP_RETRY_ABORT_ACTION, results in termination of the action which has encountered an error condition, but with continuation of the List of actions.

In Immediate Execution Mode, FB_RESP_ABORT_ACTION acts the same as FB_RESP_ABORT and FB_RESP_RETRY_ABORT_ACTION acts the same as FB_RESP_RETRY_ABORT. However, as discussed below, the Severity of the resulting Error Code will be different.

11.8.1 FB_RESP_IGNORE

If a hardware condition occurs whose associated Response Code is FB_RESP_IGNORE, the Implementation shall attempt to continue the action as if no error had occurred.

In general, the encountered error may prevent any continuation from being meaningful.

11.8.2 FB_RESP_ABORT

This shall be the default response of an Implementation to all hardware error conditions, except SS=0 and, on a block transfer, SS=2. When this is the specified Response Code for a detected hardware condition, the action or List of actions is aborted.

11.8.3 FB_RESP_RETRY_ABORT

When a hardware condition occurs for which this is the specified Response Code, the FASTBUS Action that produced the error is retried. If after the number of retries specified by the current value of the Operational Parameter FB_PAR_NUM_RETRY the error condition is still occurring, the action or List of actions is aborted.

11.8.4 FB_RESP_RETRY_IGNORE

Lorsqu'une condition matérielle se produit pour laquelle cela est le code de réponse spécifié, l'action FASTBUS qui a provoqué l'erreur est réessayée. Si, après le nombre d'essais fixé par la valeur courante du paramètre opérationnel FB_PAR_NUM_RETRY, la condition d'erreur est toujours présente, l'erreur est ignorée et l'action ou la liste d'actions est continuée comme pour FB_RESP_IGNORE.

11.8.5 FB_RESP_ABORT_ACTION

Lorsqu'une condition matérielle se produit pour laquelle cela est le code de réponse spécifié, l'action est abandonnée. Si une liste d'actions est en train de s'exécuter, l'action suivante de la liste est essayée.

11.8.6 FB_RESP_RETRY_ABORT_ACTION

Lorsqu'une condition matérielle se produit pour laquelle cela est le code de réponse spécifié, l'action FASTBUS qui a provoqué l'erreur est réessayée. Si, après le nombre d'essais spécifié par la valeur courante du paramètre opérationnel FB_PAR_NUM_RETRY, la condition d'erreur est toujours présente, l'action est abandonnée. Si une liste d'actions est en train de s'exécuter, l'action suivante de la liste est essayée.

11.8.7 Sévérité des codes d'erreur produits avec la réponse d'erreur associée

Si la valeur du code de réponse produit la continuation d'une action ou d'un ensemble d'actions plutôt que l'abandon de toutes les actions, un code d'erreur supplémentaire de FB_ERR_IGNORE (si l'erreur est ignorée), FB_ERR_RETRY (si un réessai est effectué), ou FB_ERR_ABORT_ACTION (si une seule action dans une liste est abandonnée) sera produit.

Le code d'erreur pour l'erreur originelle et le code d'erreur supplémentaire ont tous deux leur sévérité associée, mais dans ce cas les deux codes d'erreur doivent porter la moins sévère des deux. Puisque la sévérité de FB_ERR_RETRY est FB_SEV_WARN, la sévérité d'un code d'erreur dont la sévérité fixée ou par défaut est FB_SEV_ERROR est descendue lorsque le réessai est réussi.

Les sévérités par défaut des codes d'erreur associés aux codes de réponse sont:

Code d'erreur	Sévérité assignée
FB_ERR_IGNORE	FB_SEV_SUCCESS
FB_ERR_RETRY	FB_SEV_INFO
FB_ERR_ABORT_ACTION	FB_SEV_WARN

En plus, s'il est disponible, le nombre de réessais devrait être enregistré dans les informations d'état complémentaire.

11.8.8 Fixer et obtenir la réponse à une erreur

Les routines suivantes doivent être utilisées pour fixer les codes de réponse et y accéder. La première routine permet de fixer le code de réponse pour une condition donnée et la seconde permet d'obtenir le code de réponse actuel pour une condition donnée.

Nom: FSSR ou FB_STATUS_SET_RESPONSE
FSGR ou FB_STATUS_GET_RESPONSE

Catégorie: E

11.8.4 FB_RESP_RETRY_IGNORE

When a hardware condition occurs for which this is the specified Response Code, the FASTBUS Action that produced the error is retried. If after the number of retries specified by the current value of the Operational Parameter FB_PAR_NUM_RETRY the error condition persists, the error is ignored and the action or List of actions is continued as for FB_RESP_IGNORE.

11.8.5 FB_RESP_ABORT_ACTION

When a hardware condition occurs for which this is the specified Response Code, the action is aborted. If a List of actions is being executed, the next action in the List is attempted.

11.8.6 FB_RESP_RETRY_ABORT_ACTION

When a hardware condition occurs for which this is the specified Response Code, the FASTBUS Action that produced the error is retried. If after the number of retries specified by the current value of the Operational Parameter FB_PAR_NUM_RETRY the error condition persists, the action is aborted. If a List of actions is being executed, the next action in the List is attempted.

11.8.7 Severity of Error Codes generated with Associated Error Response

If the value of a Response Code results in continuation of an action or set of actions rather than aborting of all the actions, extra Error Codes from FB_ERR_IGNORE (if the error was ignored), FB_ERR_RETRY (if retries were done), or FB_ERR_ABORT_ACTION (if a single action in a List was aborted), shall be generated.

The Error Code for the original error and the extra Error Code both have associated Severities, but in this case both Error Codes shall carry the less severe of the two. Since the Severity of FB_ERR_RETRY is FB_SEV_WARN, the Severity of an Error Code whose default or Set Severity is FB_SEV_ERROR is lowered when retries are successful.

The default Severities of the Error Codes associated with the Response Codes are:

Error Code	Severity Assigned
FB_ERR_IGNORE	FB_SEV_SUCCESS
FB_ERR_RETRY	FB_SEV_INFO
FB_ERR_ABORT_ACTION	FB_SEV_WARN

In addition, if available, the number of retries should be recorded in the Supplementary Status information.

11.8.8 Specify and Determine Response to an Error

The following routines shall be used to specify and access the Response Codes. The first routine allows the Response Code for a given condition to be specified and the second allows the current Response Code for a given condition to be determined.

Names: FSSR or FB_STATUS_SET_RESPONSE
FSGR or FB_STATUS_GET_RESPONSE

Category: E

Paramètres:

return_code	FB_error_code	(OUT)
environment_id	FB_environment_id	(IN)
hardware_error	FB_error_code	(IN)
error_response	FB_response_code	(IN) (FSSR)
	ou FB_response_code	(OUT) (FSGR)

Le paramètre hardware_error doit prendre une des valeurs suivantes (le nom long correspondant de la section 12 peut également être utilisé):

FEASS0, FEASS1, FEASS2, FEASS3, FEASS4, FEASS5, FEASS6, FEASS7
 FESS0, FESS1, FESS2, FESS3, FESS4, FESS5, FESS6, FESS7
 FEDSS0, FEDSS1, FEDSS2, FEDSS3, FEDSS4, FEDSS5, FEDSS6, FEDSS7
 FEWTO, FEBSS2, FEDPE, FEBOV, FEAUTO, FESATO, FEDKTO

où FExSSy est le code d'état (SS) reçu par le maître pour un adressage primaire (x=A), un adressage secondaire (x=S) ou un cycle de données (x=D) ayant la valeur y (y=0 à 7).

Effets:

La réponse à une condition associée au code d'erreur spécifié doit être fixée en utilisant la routine FB_STATUS_SET_RESPONSE. La réponse par défaut aux déclenchements des temporisateurs et à toutes les réponses SS non nulles, sauf pour SS=2 pour des cycles de données dans un transfert de bloc, doit être FB_RESP_ABORT. La réponse par défaut à toute réponse SS=0 doit être FB_RESP_IGNORE. La réponse par défaut à SS=2 dans le cas de cycles de données dans un transfert de bloc doit être FB_RESP_ABORT_ACTION.

La dernière valeur fixée, ou si elle n'a pas été fixée, la valeur par défaut de la réponse du système à une condition matérielle d'un type donné, doit être retournée par FB_STATUS_GET_RESPONSE.

11.9 Rapport automatique des erreurs

Si une réalisation offre le "rapport automatique des erreurs", elle doit le faire de la manière suivante:

Après l'exécution d'une routine, le rapporteur automatique des erreurs doit être appelé avant de rendre le contrôle à l'utilisateur. Le rapporteur automatique des erreurs doit en premier examiner la sévérité du code d'erreur dans l'état résumé et la comparer avec celle indiquée par FB_PAR_HANDLER_THRESHOLD.

Si elle est égale ou supérieure à FB_PAR_HANDLER_THRESHOLD et que le gestionnaire d'erreur de l'utilisateur a été connecté comme décrit ci-dessous, le gestionnaire d'erreur de l'utilisateur doit être appelé.

Ensuite, la sévérité du code d'erreur dans l'état résumé est comparée avec celle indiquée par FB_PAR_REPORT_THRESHOLD. Si elle est égale ou supérieure à la valeur de ce seuil, le rapporteur automatique des erreurs doit continuer, autrement le contrôle est retourné à l'utilisateur.

Si le code d'erreur dans l'état résumé a une sévérité égale ou supérieure à FB_PAR_MESSAGE_THRESHOLD, un message sur cette erreur doit être émis. Si FB_PAR_REPORT_TERSE est FB_FALSE, un rapport doit également être émis pour les codes d'erreur de l'état complémentaire dont la sévérité est aussi égale ou supérieure à FB_PAR_MESSAGE_THRESHOLD. Si FB_PAR_REPORT_TERSE est FB_TRUE, aucun rapport automatique de l'état complémentaire ne sera émis. Remarque que si le paramètre opérationnel FB_PAR_NO_STATUS est FB_TRUE, seul l'état résumé aura été créé.

Parameters:

return_code	FB_error_code	(OUT)
environment_id	FB_environment_id	(IN)
hardware_error	FB_error_code	(IN)
error_response	FB_response_code	(IN) (FSSR)
	or FB_response_code	(OUT) (FSGR)

The hardware_error parameter shall take one of the following values (the corresponding long names from section 12 may also be used):

FEASS0, FEASS1, FEASS2, FEASS3, FEASS4, FEASS5, FEASS6, FEASS7
 FESSS0, FESSS1, FESSS2, FESSS3, FESSS4, FESSS5, FESSS6, FESSS7
 FEDSS0, FEDSS1, FEDSS2, FEDSS3, FEDSS4, FEDSS5, FEDSS6, FEDSS7
 FEWTT0, FEBSS2, FEDPE, FEBOV, FEAUTO, FESATO, FEDKTO

where FExSSy is the status (SS) code received by the Master for a primary address (x = A), secondary address (x = S) or data cycle (x = D) having a value of y (y = 0 to 7).

Effect:

The response to a condition associated with the specified Error Code shall be set by using the FB_STATUS_SET_RESPONSE routine. The default response to timeouts and all non-zero SS responses, except SS = 2 for block transfer data cycles, shall be FB_RESP_ABORT. The default response to all SS = 0 responses shall be FB_RESP_IGNORE. The default response to SS = 2 for block transfer data cycles shall be FB_RESP_ABORT_ACTION.

The last value set or, if none has been set, the default value of the response of the system to the hardware condition of the given type, shall be returned by FB_STATUS_GET_RESPONSE.

11.9 Automatic Error Reporting

If an Implementation provides for the "automatic reporting of errors", it shall act in the following fashion.

After the execution of a routine the automatic error reporter shall be invoked before returning control to the User. The automatic error reporter shall first examine the Severity of the Error Code in the Summary Status and compare it with that specified by FB_PAR_HANDLER_THRESHOLD.

If it equals or exceeds FB_PAR_HANDLER_THRESHOLD and the User error handler has been connected as described below, the User error handler shall be called.

Next the Severity of the Error Code in the Summary Status is compared with that specified by FB_PAR_REPORT_THRESHOLD. If it equals or exceeds this threshold value the automatic error reporting shall continue, otherwise control shall be returned to the User.

If the Error Code in the Summary Status has a Severity equal to or exceeding FB_PAR_MESSAGE_THRESHOLD a report of the error shall be generated. If FB_PAR_REPORT_TERSE is FB_FALSE reports shall also be generated for any Supplementary Status Error Codes whose Severity also equals or exceeds the FB_PAR_MESSAGE_THRESHOLD. If FB_PAR_REPORT_TERSE is FB_TRUE, no automatic reporting of Supplementary Status shall be carried out. Note that if the Operational Parameter FB_PAR_NO_STATUS is FB_TRUE only the Summary Status has been generated.

Le moyen utilisé par le rapporteur automatique des erreurs pour enregistrer et/ou visualiser les messages dépendra de la réalisation.

Enfin, si le code d'erreur de l'état résumé a une sévérité qui est égale ou supérieure à la valeur fixée par `FB_PAR_EXCEPTION_THRESHOLD`, une condition d'interruption doit être émise ou une mesure doit être prise pour arrêter l'exécution du programme appelant.

11.9.1 Connecter une routine utilisateur de gestion d'erreur

Nom: FSCON ou FB_STATUS_CONNECT

Catégorie: E

Paramètres:

return_code	FB_error_code	(OUT)
environment_id	FB_environment_id	(IN)
User_routine	FB_routine	(IN)

Effets:

La routine utilisateur de gestion d'erreur User_routine doit être associée à l'environnement spécifié. Toute routine précédemment connectée au même environment_id est automatiquement déconnectée.

11.9.2 Déconnecter une routine utilisateur de gestion d'erreur

Nom: FSDISC ou FB_STATUS_DISCONNECT

Catégorie: E

Paramètres:

return_code	FB_error_code	(OUT)
environment_id	FB_environment_id	(IN)

Effets:

Après l'exécution de cette routine, aucune routine utilisateur de gestion d'erreur ne doit être associée à l'environnement spécifié.

11.9.3 Détails des routines utilisateur de gestion d'erreur

Nom: (routine utilisateur de gestion d'erreur)

Catégorie: E

Paramètres:

return_code	FB_error_code	(OUT)
return_error	FB_error_code	(OUT)
input_error	FB_error_code	(IN)
environment_id	FB_environment_id	(IN)

Effets:

La routine utilisateur de gestion d'erreur doit avoir les paramètres ci-dessus. Le code d'erreur retourné doit être utilisé par le rapporteur automatique d'erreur à la place de la valeur dans l'état résumé. Si le gestionnaire d'erreur utilisateur retourne un return_code indiquant que l'exécution n'a pas été réussie, le rapporteur automatique d'erreur doit ignorer le code d'erreur renvoyé par cette routine.

The medium used by the automatic error reporter to log and/or display messages is Implementation dependent.

Finally, if the Error Code in a Summary Status has a Severity that equals or exceeds the value specified by `FB_PAR_EXCEPTION_THRESHOLD`, then an exception condition shall be raised or action shall be taken to discontinue execution of the calling program.

11.9.1 Connect User Error Handling Routine

Name: FSCON or FB_STATUS_CONNECT

Category: E

Parameters:

return_code	FB_error_code	(OUT)
environment_id	FB_environment_id	(IN)
User_routine	FB_routine	(IN)

Effect:

The User error handling routine User_routine shall be associated with the specified Environment. Any routine previously connected to the same environment_id is automatically disconnected.

11.9.2 Disconnect User Error Handling Routine

Name: FSDISC or FB_STATUS_DISCONNECT

Category: E

Parameters:

return_code	FB_error_code	(OUT)
environment_id	FB_environment_id	(IN)

Effect:

After the execution of this routine no User error handling routine shall be associated with the named Environment.

11.9.3 Details of User Error Handling Routine

Name: (User error handling routine)

Category: E

Parameters:

return_code	FB_error_code	(OUT)
return_error	FB_error_code	(OUT)
input_error	FB_error_code	(IN)
environment_id	FB_environment_id	(IN)

Effect:

The User error handling routine shall have the above parameters. The returned Error Code shall be used by the automatic error reporter in place of the value in the Summary Status. If the User error handler returns a return_code indicating that its execution was not successful, the automatic error reporter shall ignore the Error Code returned from the routine.

La routine utilisateur ne doit appeler aucune routine d'action.

11.10 Accès à l'information de l'état résumé

Un état résumé est produit chaque fois qu'une action d'une routine d'action est exécutée dans l'environnement du mode d'exécution immédiate ou que la routine FB_LIST_EXECUTE est appelée. Les routines suivantes décodent l'information de l'état résumé.

11.10.1 Décoder l'état résumé

Nom: FGSUM ou FB_STATUS_GET_SUMMARY

Catégorie: A

Paramètres:

return_code	FB_error_code	(OUT)
environment_id	FB_environment_id	(IN)
error_code	FB_error_code	(OUT)
bytes_read	FB_integer	(OUT)
bytes_written	FB_integer	(OUT)

Effets:

Cette routine doit retourner les valeurs suivantes de l'environnement spécifié:

error_code	<i>Le code d'erreur avec la sévérité la plus forte enregistrée dans l'environnement.</i>
bytes_read	<i>Le nombre total d'octets lus dans un tampon de données résultant de l'appel d'une routine d'action ou de l'appel à FB_LIST_EXECUTE.</i>
bytes_written	<i>Le nombre total d'octets écrits dans un tampon de données résultant de l'appel d'une routine d'action ou de l'appel à FB_LIST_EXECUTE.</i>

11.11 Accès à l'information d'état complémentaire

En plus de l'information dans l'état résumé, une réalisation peut obtenir des informations additionnelles sur l'exécution d'une action - connues sous le nom "d'état complémentaire". Le nombre et le type des éléments de cet état complémentaire dépend de la réalisation. On ne peut donc pas donner une définition exacte de ces paramètres.

Un mécanisme pour accéder à une telle information est ainsi défini: chaque appel à une des routines ci-dessous doit retourner un seul enregistrement des informations de l'état complémentaire.

Chaque enregistrement est conçu pour être constitué d'un code d'erreur qui décrit l'état et sa sévérité associée, éventuellement un "associated_parameter" et un paramètre "where occurred" qui indique quelle action ou quelle partie d'action a provoqué le code d'erreur. Il est de la responsabilité de l'utilisateur de savoir quel code d'erreur possède un paramètre associé.

Par exemple, le code d'erreur informatif FB_ERR_LAST_PAD devrait avoir comme paramètre associé la valeur de la dernière adresse primaire utilisée.

Si la routine utilisée pour accéder à l'enregistrement de l'état complémentaire ne peut pas fournir plus d'informations, elle doit retourner un code d'erreur FB_ERR_NO_MORE_STATUS.

The User routine shall not call any Action Routine.

11.10 Access to Summary Status Information

Summary Status is generated whenever the action of an Action Routine is executed with an Immediate Execution Mode Environment or the routine FB_LIST_EXECUTE is called. The following routine decodes the information in the Summary Status.

11.10.1 Decode Summary Status

Name: FSGSUM or FB_STATUS_GET_SUMMARY

Category: A

Parameters:

return_code	FB_error_code	(OUT)
environment_id	FB_environment_id	(IN)
error_code	FB_error_code	(OUT)
bytes_read	FB_integer	(OUT)
bytes_written	FB_integer	(OUT)

Effect:

This routine shall return the following values from the specified Environment:

error_code The Error Code with the highest Severity recorded in the Environment.

bytes_read The total number of bytes read into any Data Buffers as a result of the Action Routine call or call to FB_LIST_EXECUTE.

bytes_written The total number of bytes written out of any Data Buffers as a result of the Action Routine call or call to FB_LIST_EXECUTE.

11.11 Access to Supplementary Status Information

In addition to the information in the Summary Status an Implementation may gather additional information about the execution of actions - known as the "Supplementary Status". The amount and type of this Supplementary Status will be dependent on the Implementation. Thus an exact definition of all the parameters cannot be made.

A mechanism for accessing such information is defined such that each call to one of the routines below shall return a single record of Supplementary Status information.

Each record is conceived of as consisting of an Error Code which describes the status and its associated severity, an "associated_parameter", if applicable, and a "where_occurred" parameter which indicates which action, or which part of an action, gave rise to the Error Code. It is the responsibility of the user to be aware of which Error Codes have an associated parameter.

For example, the informational Error Code FB_ERR_LAST_PAD should have as its associated parameter the value of the last primary address used.

If the routine used to access the Supplementary Status records can supply no more information, it shall return an Error Code FB_ERR_NO_MORE_STATUS.

Des appels répétés à la routine d'accès peuvent être effectués pour extraire toute l'information d'état disponible.

Etat complémentaire provenant de l'exécution d'une liste:

Lorsque l'on accède à l'état complémentaire d'un appel à FB_LIST EXECUTE, les paramètres where_occurred et list_pointer sont pointés sur l'action dans la liste à laquelle les informations d'état s'appliquent. En particulier, le list_pointer (un paramètre d'entrée vers les routines de décodage d'état ci-dessous) peut être obtenu pendant que la liste est construite par la routine FB_LIST_GET_POINTER.

11.11.1 Décoder l'information d'état complémentaire

Nom: FSGSUP ou FB_STATUS_GET_SUPPLEMENTARY

Catégorie: E

Paramètres:

return_code	FB_error_code	(OUT)
environment_id	FB_environment_id	(IN)
error_code	FB_error_code	(OUT)
associated_parameter	FB_word	(OUT)
where_occurred	FB_integer	(OUT)

Effets:

Cette routine doit retourner les autres informations d'état sur l'action.

Le premier appel à cette routine doit retourner le même code d'erreur que celui retourné par FB_STATUS_GET_SUMMARY.

Informations sur des actions exécutées en mode d'exécution différée:

Le premier appel à cette routine qui fournit une information sur une action particulière dans une liste doit renvoyer un code d'erreur équivalent à celui qui aurait été disponible dans l'état résumé si l'action avait été exécutée en mode d'exécution immédiate.

11.11.2 Décoder l'état complémentaire dans l'exécution d'une liste

Nom: FSGLSU ou FB_STATUS_LIST_SUPPLEMENTARY

Catégorie: E

Paramètres:

return_code	FB_error_code	(OUT)
environment_id	FB_environment_id	(IN)
list_pointer	FB_list_pointer	(IN)
error_code	FB_error_code	(OUT)
associated_parameter	FB_word	(OUT)
where_occurred	FB_integer	(OUT)

Effets:

Cette routine doit retourner l'information d'état détaillé sur l'historique d'état pour l'appel à une action spécifiée dans une liste d'une manière identique à FB_STATUS_GET_SUPPLEMENTARY.

Repeated calls to the access routines can be used to extract all the available status information.

Supplementary Status from List execution:

When the Supplementary Status information from a call to FB_LIST_EXECUTE is accessed, the where_occurred and list_pointer parameters are pointers to the action within the List to which the status information applies. In particular, the list_pointer (an input parameter to the status decoding routines below) may be obtained while the List is being built through use of the FB_LIST_GET_POINTER routine.

11.11.1 Decode Supplementary Status Information

Name: FSGSUP or FB_STATUS_GET_SUPPLEMENTARY

Category: E

Parameters:

return_code	FB_error_code	(OUT)
environment_id	FB_environment_id	(IN)
error_code	FB_error_code	(OUT)
associated_parameter	FB_word	(OUT)
where_occurred	FB_integer	(OUT)

Effect:

This routine shall return further status information about an action.

The first call to this routine shall return the same Error Code as that returned by FB_STATUS_GET_SUMMARY.

Information from Actions Executed in Delayed Execution Mode:

The first call to this routine that results in information about a particular action within a List shall return an Error Code equivalent to that which would have been available in the Summary Status if the action had been executed in Immediate Execution Mode.

11.11.2 Decode Supplementary Status from Execution of a List

Name: FSGLSU or FB_STATUS_LIST_SUPPLEMENTARY

Category: E

Parameters:

return_code	FB_error_code	(OUT)
environment_id	FB_environment_id	(IN)
list_pointer	FB_list_pointer	(IN)
error_code	FB_error_code	(OUT)
associated_parameter	FB_word	(OUT)
where_occurred	FB_integer	(OUT)

Effect:

This routine shall return detailed status information from the Status History for a specified action call within a List in a manner identical to FB_STATUS_GET_SUPPLEMENTARY.

Le list_pointer de liste doit être celui renvoyé par un appel à FB_LIST_GET_POINTER effectué immédiatement avant l'appel à la routine, dont on obtient l'état de l'action.

Le premier appel à cette routine qui fournit une information sur une action particulière dans une liste doit renvoyer un code d'erreur équivalent à celui qui aurait été disponible dans l'état résumé si l'action avait été exécutée en mode d'exécution immédiate.

11.11.3 Trouver l'information d'état complémentaire

Nom: FSFSUP ou FB_FIND_SUPPLEMENTARY

Catégorie: E

Paramètres:

return_code	FB_error_code	(OUT)
environment_id	FB_environment_id	(IN)
error_code	FB_error_code	(IN)
associated_parameter	FB_word	(OUT)
where_occurred	FB_integer	(OUT)

Effets:

Cette routine fonctionne d'une manière identique à FB_STATUS_GET_SUPPLEMENTARY, sauf que le paramètre error_code est fourni par l'appelant. Cette routine permet à l'utilisateur d'extraire un enregistrement particulier de l'information d'état complémentaire.

Si le code d'erreur indiqué n'est pas présent dans l'état complémentaire, l'erreur FB_ERR_STATUS_NOT_FOUND est retournée.

11.11.4 Trouver un état complémentaire particulier dans une liste

Nom: FSFLSU ou FB_FIND_LIST_SUPPLEMENTARY

Catégorie: E

Paramètres:

return_code	FB_error_code	(OUT)
environment_id	FB_environment_id	(IN)
list_pointer	FB_list_pointer	(IN)
error_code	FB_error_code	(IN)
associated_parameter	FB_word	(OUT)
where_occurred	FB_integer	(OUT)

Effets:

Cette routine fonctionne d'une manière identique à FB_STATUS_LIST_SUPPLEMENTARY, sauf que le paramètre error_code est fourni par l'appelant, pour pouvoir extraire tout état complémentaire particulier qui serait présent.

Cette routine doit seulement retourner l'information pour une action où le code d'erreur et le pointeur de liste correspondent à ceux spécifiés par l'utilisateur.

Si aucune autre présence de error_code n'est trouvée dans l'état complémentaire, l'erreur FB_ERR_STATUS_NOT_FOUND sera retournée (voir plus haut).

The list_pointer shall be that returned by a call to FB_LIST_GET_POINTER done immediately before the call to the routine for whose action the status is to be obtained.

The first call to this routine that results in information about a particular action within a List shall return an Error Code equivalent to that which would have been available in the Summary Status if the action had been executed in Immediate Execution Mode.

11.11.3 Find Supplementary Status Information

Name: FSFSUP or FB_FIND_SUPPLEMENTARY

Category: E

Parameters:

return_code	FB_error_code	(OUT)
environment_id	FB_environment_id	(IN)
error_code	FB_error_code	(IN)
associated_parameter	FB_word	(OUT)
where_occurred	FB_integer	(OUT)

Effect:

This routine acts identically to FB_STATUS_GET_SUPPLEMENTARY, except that the error_code parameter is supplied by the caller. The routine allows the user to extract a specified record of Supplementary Status information.

If the specified Error Code is not present in the Supplementary Status the error FB_ERR_STATUS_NOT_FOUND is returned.

11.11.4 Find Specific Supplementary Status within a List

Name: FSFLSU or FB_FIND_LIST_SUPPLEMENTARY

Category: E

Parameters:

return_code	FB_error_code	(OUT)
environment_id	FB_environment_id	(IN)
list_pointer	FB_list_pointer	(IN)
error_code	FB_error_code	(IN)
associated_parameter	FB_word	(OUT)
where_occurred	FB_integer	(OUT)

Effect:

This routine acts identically to FB_STATUS_LIST_SUPPLEMENTARY, except that the error_code parameter is supplied by the caller, in order to determine whether specific Supplementary Status is present.

The routine shall return information only for an action where the Error Code and List pointer match that specified by the User.

If no further occurrence of the error_code is found in the supplementary status the error FB_ERR_STATUS_NOT_FOUND shall be returned (see above).

11.12 Routines pour la manipulation des codes d'erreur

Remarquer que `FB_STATUS_SEVERITY`, `FB_STATUS_MATCH` et `FB_STATUS_THRESHOLD` sont des fonctions et non des sous-programmes.

11.12.1 Contrôler la sévérité d'un code d'erreur

Nom: `FSEVER` ou `FB_STATUS_SEVERITY`

Catégorie: E

Paramètres:

`error_code` `FB_error_code` (IN)

Effets:

`FB_STATUS_SEVERITY` est une fonction qui retourne la sévérité du `error_code` indiqué.

11.12.2 Tester le code d'état dans un code d'erreur

Nom: `FMATCH` ou `FB_STATUS_MATCH`

Catégorie: E

Paramètres:

`error_code_1` `FB_error_code` (IN)
`error_code_2` `FB_error_code` (IN)

Effets:

Cette fonction doit retourner `FB_TRUE` si les deux codes d'erreur sont identiques (en excluant la sévérité) et `FB_FALSE` s'ils ne le sont pas.

11.12.3 Tester la sévérité dans un code d'erreur

Nom: `FSTHR` ou `FB_STATUS_THRESHOLD`

Catégorie: E

Paramètres:

`error_code_1` `FB_error_code` (IN)
`severity` `FB_severity` (IN)

Effets:

Cette fonction doit retourner `FB_TRUE` si la sévérité du code d'erreur est égale ou supérieure à la sévérité indiquée et `FB_FALSE` si elle ne l'est pas.

11.12.4 Retourner les codes d'erreur d'une sévérité spécifiée

Nom: `FSENC` ou `FB_STATUS_ENCODE`

Catégorie: E

11.12 Routines for the Manipulation of Error Codes

Note that `FB_STATUS_SEVERITY`, `FB_STATUS_MATCH` and `FB_STATUS_THRESHOLD` are functions - not subroutines.

11.12.1 Check Severity of an Error Code

Name: `FSEVER` or `FB_STATUS_SEVERITY`

Category: E

Parameters:

error_code FB_error_code (IN)

Effect:

`FB_STATUS_SEVERITY` is a function which shall return the Severity of the specified error_code.

11.12.2 Test Status Code in an Error Code

Name: `FMATCH` or `FB_STATUS_MATCH`

Category: E

Parameters:

error_code_1 FB_error_code (IN)
error_code_2 FB_error_code (IN)

Effect:

This function shall return `FB_TRUE` if the two Error Codes are identical (exclusive of Severity) and `FB_FALSE` if they are not.

11.12.3 Test Severity in an Error Code

Name: `FSTHR` or `FB_STATUS_THRESHOLD`

Category: E

Parameters:

error_code_1 FB_error_code (IN)
severity FB_severity (IN)

Effect:

This function shall return `FB_TRUE` if the Severity of the Error Code equals or exceeds the specified Severity and `FB_FALSE` if it does not.

11.12.4 Return Error Code of Specified Severity

Name: `FSENC` or `FB_STATUS_ENCODE`

Category: E

Paramètres:

new_error_code	FB_error_code	(OUT)
error_code	FB_error_code	(IN)
severity	FB_severity	(IN)

Effets:

Cette routine doit retourner un code d'erreur qui a la même information d'état que celle de error_code mais avec la sévérité indiquée.

11.13 Rapporter une erreur FASTBUS

Nom: FSRPT ou FB_STATUS_REPORT

Catégorie: E

Paramètres:

return_code	FB_error_code	(OUT)
environment_id	FB_environment_id	(IN)
error_code	FB_error_code	(IN)
associated_parameter	FB_word	(IN)
where_occurred	FB_integer	(IN)

Effets:

Un message d'erreur doit être émis à partir des paramètres error_code, associated_parameter et where_occurred. Certaines mesures dépendant de la réalisation doivent être prises pour attirer l'attention de l'utilisateur.

11.14 Emettre un message d'erreur

Nom: FSTRAN ou FB_STATUS_TRANSLATE

Catégorie: E

Paramètres:

return_code	FB_error_code	(OUT)
environment_id	FB_environment_id	(IN)
error_code	FB_error_code	(IN)
associated_parameter	FB_word	(IN)
where_occurred	FB_integer	(IN)
message	FB_string	(OUT)

Effets:

Un message d'erreur doit être émis à partir des paramètres error_code, associated_parameter et where_occurred. Le résultat est retourné dans le paramètre message.

Parameters:

new_error_code	FB_error_code	(OUT)
error_code	FB_error_code	(IN)
severity	FB_severity	(IN)

Effect:

This routine shall return an Error Code with the same status information as that in error_code, but with the specified Severity.

11.13 Report a FASTBUS Error

Name: FSRPT or FB_STATUS_REPORT

Category: E

Parameters:

return_code	FB_error_code	(OUT)
environment_id	FB_environment_id	(IN)
error_code	FB_error_code	(IN)
associated_parameter	FB_word	(IN)
where_occurred	FB_integer	(IN)

Effect:

An error message shall be generated from the error_code, associated_parameter and where_occurred parameters. Some Implementation dependent action shall be taken to bring this to the User's attention.

11.14 Generate an Error Message

Name: FSTRAN or FB_STATUS_TRANSLATE

Category: E

Parameters:

return_code	FB_error_code	(OUT)
environment_id	FB_environment_id	(IN)
error_code	FB_error_code	(IN)
associated_parameter	FB_word	(IN)
where_occurred	FB_integer	(IN)
message	FB_string	(OUT)

Effect:

An error message shall be generated from the error_code, associated_parameter and where_occurred parameters. The result is returned in the message parameter.

Section 12. Codes d'erreur

12.1 Ensemble standard de codes d'erreur

Comme discuté à la section 11, un code d'erreur contient les informations d'état (code d'état) - décrivant certaines conditions qui se sont produites à l'exécution d'un appel à une routine de cette norme - et une sévérité.

Les codes d'erreur définis dans cette section tentent de couvrir toutes les conditions d'état possibles qui peuvent résulter de l'exécution des routines de cette norme. Chaque code d'erreur défini possède une sévérité associée par défaut avec des conditions d'état données.

Toutes les réalisations doivent définir les noms courts des codes d'erreur décrits dans cette section.

Si une réalisation n'inclut pas `FB_STATUS_SET_SEVERITY` et/ou `FB_STATUS_SET_RESPONSE` (voir section 11), la sévérité de chaque code d'erreur sera toujours celle définie ci-dessous.

Les routines `FB_STATUS_SET_RESPONSE` et `FB_STATUS_SET_SEVERITY` peuvent changer la sévérité associée à un code d'état. Lorsque cela se produit, la sévérité du code d'erreur sera changée.

FB_ERR_ACCESS_DATA_BUFFER Nom court: FEBUF, Sévérité : FB_SEV_ERROR

Pendant un transfert de données de ou vers un tampon de données, une erreur d'accès s'est produite. Cela peut arriver lors de l'accès à un tampon aléatoire, séquentiel interne ou externe.

FB_ERR_ABORT_ACTION Nom court: FEABA, Sévérité : FB_SEV_WARNING

Dans une liste d'actions exécutées par un appel à `FB_LIST_EXECUTE`, une action a été terminée prématurément par l'apparition d'une erreur, mais la liste d'actions n'a pas été abandonnée. Le code d'erreur correspondant à l'erreur originelle devrait être disponible en conjonction avec ce code d'erreur.

La sévérité de `FB_ERR_ABORT_ACTION` doit être utilisée pour modifier la sévérité rapportée pour l'erreur originelle.

FB_ERR_ADDRESS_RELEASE_TIMEOUT Nom court: FEAkDO, Sévérité: FB_SEV_ERROR

AK(d) ne s'est pas produit après AS(d) pendant la période du temporisateur.

FB_ERR_AK_TIMEOUT Nom court: FEAkTO, Sévérité : FB_SEV_ERROR

Le temporisateur AK a déclenché.

FB_ERR_ALREADY_CONNECT Nom court: FEACON, Sévérité : FB_SEV_ERROR

Un essai pour connecter une routine a échoué car elle était déjà connectée.

FB_ERR_BAD_PARAMETER_VALUE Nom court: FEBPRV, Sévérité : FB_SEV_WARNING

Dans l'appel à `FB_PAR_SET`, la valeur du paramètre donné n'est pas correcte, mais la valeur (éventuellement modifiée) est quand même acceptée et l'environnement modifié.

Section 12. Error Codes

12.1 Standard Set of Error Codes

As discussed in section 11, an Error Code contains status information (Status Code) - describing some conditions that occurred on execution of a call to a Routine in this standard - and a Severity.

The Error Codes defined in this section attempt to cover all possible status conditions that might result from executing the routines in this standard. Each Error Code defined has the Severity associated by default with the given status condition.

All Implementations shall define the short names of the Error Codes listed in this section.

If an Implementation does not include FB_STATUS_SET_SEVERITY and/or FB_STATUS_SET_RESPONSE (see section 11), the Severity of each Error Code will always be as listed below.

The routines FB_STATUS_SET_RESPONSE and FB_STATUS_SET_SEVERITY may change the Severity to be associated with a Status Code. When this occurs, the Severity of the Error Code will be changed.

FB_ERR_ACCESS_DATA_BUFFER Short name: FEBUF, Severity : FB_SEV_ERROR

During a transfer of data to or from a Data Buffer an access error occurred. This could be while accessing a random, internal or external sequential buffer.

FB_ERR_ABORT_ACTION Short name: FEABA, Severity : FB_SEV_WARNING

Within a List of actions executed through a call to FB_LIST_EXECUTE, an action was terminated prematurely due to the occurrence of an error, but the List of actions was not aborted. The error code corresponding to the original error should be available in conjunction with this error code.

The Severity of the FB_ERR_ABORT_ACTION shall be used to modify the Severity reported for the original error.

FB_ERR_ADDRESS_RELEASE_TIMEOUT Short name: FEAKDO, Severity: FB_SEV_ERROR

AK(d) did not occur after AS(d) within the timeout period.

FB_ERR_AK_TIMEOUT Short name: FEAKTO, Severity : FB_SEV_ERROR

The AK timer timed out.

FB_ERR_ALREADY_CONNECT Short name: FEACON, Severity : FB_SEV_ERROR

An attempt to connect a routine failed because there was already a connection.

FB_ERR_BAD_PARAMETER_VALUE Short name: FEBPRV, Severity : FB_SEV_WARNING

In calling FB_PAR_SET the value of the given parameter is not proper, but the value (possibly transformed) is accepted anyway and the Environment changed.

FB_ERR_BLOCK_DATA_SS2	Nom court: FEBSS2,	Sévérité : FB_SEV_INFO
Cycle de données avec SS = 2 pendant un transfert de bloc (voir FB_ERR_DATA_SS2).		
FB_ERR_BUFFER_OVERFLOW	Nom court: FEBOV,	Sévérité : FB_SEV_ERROR
L'action demandée va faire ou a fait déborder le tampon de données.		
FB_ERR_CLOSED	Nom court: FECLSD,	Sévérité : FB_SEV_ERROR
La session FASTBUS est terminée. Aucune routine sauf FB_OPEN ne peut être appelée à partir de cet état. (Egalement retourné par FB_CLOSE si la session n'est pas ouverte.)		
FB_ERR_CONNECT	Nom court: FECON,	Sévérité : FB_SEV_ERROR
Un essai pour connecter une routine a échoué et il n'y avait pas de connexion antérieure.		
FB_ERR_DATA_PARITY_ERROR	Nom court: FEDPE,	Sévérité : FB_SEV_ERROR
On a rencontré une erreur de parité dans un cycle de lecture de données FASTBUS. (SS peut avoir une valeur quelconque y compris 0.)		
FB_ERR_DATA_SS0	Nom court: FEDSS0,	Sévérité : FB_SEV_SUCCESS
Sur un cycle de données FASTBUS un code SS = 0 a été retourné.		
FB_ERR_DATA_SS1	Nom court: FEDSS1,	Sévérité : FB_SEV_ERROR
Sur un cycle de données FASTBUS un code SS = 1 a été retourné.		
FB_ERR_DATA_SS2	Nom court: FEDSS2,	Sévérité : FB_SEV_ERROR
Sur un cycle de données FASTBUS un code SS = 2 a été retourné. Remarquer que lorsqu'un transfert de bloc est exécuté, une réponse SS = 2 devrait, par défaut, émettre l'erreur FB_ERR_BLOCK_DATA_SS2, dont la sévérité associée est FB_SEV_INFO.		
FB_ERR_DATA_SS3	Nom court: FEDSS3,	Sévérité : FB_SEV_ERROR
Sur un cycle de données FASTBUS un code SS = 3 a été retourné.		
FB_ERR_DATA_SS4	Nom court: FEDSS4,	Sévérité : FB_SEV_ERROR
Sur un cycle de données FASTBUS un code SS = 4 a été retourné.		
FB_ERR_DATA_SS5	Nom court: FEDSS5,	Sévérité : FB_SEV_ERROR
Sur un cycle de données FASTBUS un code SS = 5 a été retourné.		
FB_ERR_DATA_SS6	Nom court: FEDSS6,	Sévérité : FB_SEV_ERROR
Sur un cycle de données FASTBUS un code SS = 6 a été retourné.		
FB_ERR_DATA_SS7	Nom court: FEDSS7,	Sévérité : FB_SEV_ERROR
Sur un cycle de données FASTBUS un code SS = 7 a été retourné.		
FB_ERR_DISCONNECT	Nom court: FEDCON,	Sévérité : FB_SEV_ERROR
Un essai pour déconnecter une routine a échoué. Aucune connexion n'existe actuellement.		

FB_ERR_BLOCK_DATA_SS2 Short name: FEBSS2, Severity : FB_SEV_INFO

Data cycle SS = 2 during a block transfer (see FB_ERR_DATA_SS2).

FB_ERR_BUFFER_OVERFLOW Short name: FEBOV, Severity : FB_SEV_ERROR

The requested action will overflow or overflowed the Data Buffer.

FB_ERR_CLOSED Short name: FECLSD, Severity : FB_SEV_ERROR

The FASTBUS session is closed. No other routine than FB_OPEN may be called from this state. (Also returned by FB_CLOSE if the session is not open.)

FB_ERR_CONNECT Short name: FECON, Severity : FB_SEV_ERROR

An attempt to connect a routine failed and there was no previous connection.

FB_ERR_DATA_PARITY_ERROR Short name: FEDPE, Severity : FB_SEV_ERROR

On a FASTBUS read data cycle a parity error was encountered. (SS may have any value including 0.)

FB_ERR_DATA_SS0 Short name: FEDSS0, Severity : FB_SEV_SUCCESS

On a FASTBUS data cycle an SS code of 0 was returned.

FB_ERR_DATA_SS1 Short name: FEDSS1, Severity : FB_SEV_ERROR

On a FASTBUS data cycle an SS code of 1 was returned.

FB_ERR_DATA_SS2 Short name: FEDSS2, Severity : FB_SEV_ERROR

On a FASTBUS data cycle SS = 2 was returned. Note that when a block transfer is being executed an SS = 2 should, by default, generate the error FB_ERR_BLOCK_DATA_SS2, which has an associated Severity of FB_SEV_INFO.

FB_ERR_DATA_SS3 Short name: FEDSS3, Severity : FB_SEV_ERROR

On a FASTBUS data cycle SS = 3 was returned.

FB_ERR_DATA_SS4 Short name: FEDSS4, Severity : FB_SEV_ERROR

On a FASTBUS data cycle SS = 4 was returned.

FB_ERR_DATA_SS5 Short name: FEDSS5, Severity : FB_SEV_ERROR

On a FASTBUS data cycle SS = 5 was returned.

FB_ERR_DATA_SS6 Short name: FEDSS6, Severity : FB_SEV_ERROR

On a FASTBUS data cycle SS = 6 was returned.

FB_ERR_DATA_SS7 Short name: FEDSS7, Severity : FB_SEV_ERROR

On a FASTBUS data cycle SS = 7 was returned.

FB_ERR_DISCONNECT Short name: FEDCON, Severity : FB_SEV_ERROR

An attempt to disconnect a routine failed. No connection may currently exist.

FB_ERR_DK_RELEASE_TIMEOUT Nom court: FEDKDO, Sévérité : FB_SEV_ERROR

DK(d) ne s'est pas produite après DS(d) pendant la période du temporisateur.

FB_ERR_DK_TIMEOUT Nom court: FEDKTO, Sévérité : FB_SEV_ERROR

Une transition sur DK ne s'est pas produite après une transition sur DS pendant la période du temporisateur.

FB_ERR_ENV_ACTIVE Nom court: FEENAC, Sévérité : FB_SEV_ERROR

L'environnement spécifié est en fonctionnement et ne peut pas être accédé. Cette erreur peut se produire si l'on essaie d'accéder à un environnement tandis qu'une action ou un ensemble d'actions s'exécutent avec le paramètre opérationnel FB_PAR_NO_WAIT à FB_TRUE.

FB_ERR_ENV_ID_OVERFLOW Nom court: FEEIOV, Sévérité : FB_SEV_ERROR

Il n'est pas possible de créer un environnement car le nombre maximal d'environnements a déjà été créé.

FB_ERR_ENV_OVERFLOW Nom court: FEENOV, Sévérité : FB_SEV_ERROR

Il n'y a pas la place suffisante pour créer un environnement.

FB_ERR_ENV_RELEASE Nom court: FEEREL, Sévérité : FB_SEV_ERROR

Un essai pour libérer un environnement a échoué. L'environnement_id était cependant valable.

FB_ERR_ERROR Nom court: FEERR, Sévérité: FB_SEV_ERROR

Une erreur s'est produite avec la sévérité FB_SEV_ERROR. Des informations supplémentaires peuvent être disponibles dans l'historique d'état s'il existe.

FB_ERR_FATAL Nom court: FEFTL, Sévérité: FB_SEV_FATAL

Une erreur s'est produite avec la sévérité FB_SEV_FATAL. Des informations supplémentaires peuvent être disponibles dans l'historique d'état s'il existe.

FB_ERR_HARDWARE_ERROR Nom court: FEHERR, Sévérité : FB_SEV_ERROR

Une erreur qui dépend de la réalisation s'est produite dans le port matériel. Toutes informations supplémentaires disponibles peuvent être obtenues dans l'historique d'état.

FB_ERR_HARDWARE_WARNING Nom court: FEHWRN, Sévérité : FB_SEV_WARNING

Une mise en garde qui dépend de la réalisation s'est produite dans le port matériel. Toutes informations supplémentaires disponibles peuvent être obtenues dans l'historique d'état.

FB_ERR_HOLDING_ADDRESS Nom court: FEHDAS, Sévérité : FB_SEV_INFO

Une connexion d'adresse (AS/AK chaînés) est maintenue, soit parce que le paramètre opérationnel FB_PAR_HOLD_AS ou FB_HOLD_BUS_ON_ERROR est FB_TRUE, ou parce qu'une action primitive FASTBUS a été exécutée.

FB_ERR_HOLDING_BUS Nom court: FEHDFB, Sévérité : FB_SEV_INFO

La maîtrise du FASTBUS est maintenue (GK est positionné par le port). Cela est soit parce qu'un paramètre opérationnel FB_PAR_HOLD_BUS, FB_PAR_HOLD_BUS_NO_AR, ou

FB_ERR_DK_RELEASE_TIMEOUT Short name: FEDKDO, Severity: FB_SEV_ERROR

DK(d) did not occur after DS(d) within the timeout period.

FB_ERR_DK_TIMEOUT Short name: FEDKTO, Severity : FB_SEV_ERROR

A transition in DK did not occur after a transition in DS within the timeout period.

FB_ERR_ENV_ACTIVE Short name: FEENAC, Severity : FB_SEV_ERROR

The specified Environment is in use and may not be accessed. This error may occur if an access to an Environment is attempted while an Action or set of Actions is being executed with the Operational Parameter FB_PAR_NO_WAIT set to FB_TRUE.

FB_ERR_ENV_ID_OVERFLOW Short name: FEEIOV, Severity : FB_SEV_ERROR

It is not possible to create an environment because the maximum number of environments has already been created.

FB_ERR_ENV_OVERFLOW Short name: FEENOV, Severity : FB_SEV_ERROR

There is insufficient space to create an Environment.

FB_ERR_ENV_RELEASE Short name: FEEREL, Severity : FB_SEV_ERROR

An attempt to release an Environment failed. The environment_id was, however, valid.

FB_ERR_ERROR Short name: FEERR, Severity : FB_SEV_ERROR

An error occurred with Severity FB_SEV_ERROR. Additional information may be available in the Status History if it exists.

FB_ERR_FATAL Short name: FEFTL, Severity: FB_SEV_FATAL

An error occurred with Severity FB_SEV_FATAL. Additional information may be available in the Status History if it exists.

FB_ERR_HARDWARE_ERROR Short name: FEHERR, Severity : FB_SEV_ERROR

An Implementation dependent Port hardware error occurred. Any available additional information about the error may be obtained from the Status History.

FB_ERR_HARDWARE_WARNING Short name: FEHWRN, Severity : FB_SEV_WARNING

An Implementation dependent Port hardware warning error occurred. Any available additional information may be obtained from the Status History.

FB_ERR_HOLDING_ADDRESS Short name: FEHDAS, Severity : FB_SEV_INFO

An address connection (AS/AK lock) is being maintained, either because the Operational Parameter FB_PAR_HOLD_AS or FB_HOLD_BUS_ON_ERROR is FB_TRUE, or because a Primitive FASTBUS Action has been executed.

FB_ERR_HOLDING_BUS Short name: FEHDFB, Severity : FB_SEV_INFO

FASTBUS mastership is being maintained (GK is asserted by the Port). This is either because the Operational Parameter FB_PAR_HOLD_BUS, or

FB_HOLD_BUS_ON_ERROR est **FB_TRUE**, ou parce qu'une action primitive FASTBUS a été exécutée.

FB_ERR_IGNORE Nom court: FEIGNO, Sévérité : **FB_SEV_SUCCESS**

Une erreur qui a été ignorée s'est produite. Des réessais ont peut-être été effectués. Le code d'erreur correspondant à l'erreur originelle devrait être disponible avec ce code d'erreur.

La sévérité de **FB_ERR_IGNORE** doit être utilisée pour modifier la sévérité rapportée pour l'erreur originelle.

FB_ERR_ILL_PARAMETER_VALUE Nom court: FEIPRV, Sévérité : **FB_SEV_ERROR**

Dans l'appel à **FB_SET_PAR**, la valeur d'un paramètre opérationnel donné n'est pas permise. Aucun changement n'est fait dans l'environnement.

FB_ERR_ILL_SEQ_BUFFER_ID Nom court: FEILSI, Sévérité : **FB_SEV_ERROR**

Renvoyé par **FB_DECLARE_SEQ_BUFFER** ou **FB_DECLARE_EXT_SEQ_BUFFER**; la valeur de **buffer_id** séquentiel n'est pas valable pour la réalisation.

FB_ERR_IMPLEMENTATION Nom court: FEIMP, Sévérité : **FB_SEV_ERROR**

Une erreur inattendue dans la réalisation s'est produite. Aucune autre information n'est disponible.

FB_ERR_INFO Nom court: FEINFO, Sévérité: **FB_SEV_INFO**

Une erreur s'est produite avec une sévérité **FB_SEV_INFO**. Des informations supplémentaires peuvent se trouver dans l'historique d'état s'il existe.

FB_ERR_INVALID_ENV_ID Nom court: FEINEI, Sévérité : **FB_SEV_ERROR**

Il n'y a pas d'environnement qui correspond à l'environnement_id donné.

FB_ERR_INVALID_LIST_POINTER Nom court: FEINLI, Sévérité : **FB_SEV_ERROR**

Une valeur non valable a été spécifiée pour le pointeur de liste.

FB_ERR_INVALID_PORT_ID Nom court: FEINPO, Sévérité : **FB_SEV_ERROR**

Il n'y a pas de port correspondant au port_id donné.

FB_ERR_INVALID_SEQ_BUFFER_ID Nom court: FEINSI, Sévérité : **FB_SEV_ERROR**

Il n'y a pas de tampon séquentiel (interne ou externe) qui correspond au **buffer_id** séquentiel donné.

FB_ERR_LAST_PRIMARY_ADDRESS Nom court: FELPAD, Sévérité : **FB_SEV_INFO**

La valeur de la dernière connexion d'adresse primaire dans une action ou un appel à **FB_LIST_EXECUTE**.

FB_ERR_LAST_SECONDARY_ADDRESS Nom court: FELSAD, Sévérité : **FB_SEV_INFO**

La valeur de la dernière connexion d'adresse secondaire dans une action ou un appel à **FB_LIST_EXECUTE**.

FB_PAR_HOLD_BUS_NO_AR, or **FB_HOLD_BUS_ON_ERROR** is **FB_TRUE**, or because a Primitive FASTBUS Action has been executed.

FB_ERR_IGNORE Short name: FEIGNO, Severity : FB_SEV_SUCCESS

An error occurred which was ignored. Retries may have been attempted. The error code corresponding to the original error should be available in conjunction with this error code.

The Severity of the **FB_ERR_IGNORE** shall be used to modify the Severity reported for the original error.

FB_ERR_ILL_PARAMETER_VALUE Short name: FEIPRV, Severity: FB_SEV_ERROR

In calling **FB_SET_PAR** the value of the given Operational Parameter is not allowed. No change was made in the Environment.

FB_ERR_ILL_SEQ_BUFFER_ID Short name: FEILSI, Severity: FB_SEV_ERROR

Returned from **FB_DECLARE_SEQ_BUFFER** or **FB_DECLARE_EXT_SEQ_BUFFER**; the value of the sequential buffer_id is invalid for the Implementation.

FB_ERR_IMPLEMENTATION Short name: FEIMP, Severity : FB_SEV_ERROR

An unexpected error in the Implementation has been encountered. No other information is available.

FB_ERR_INFO Short name: FEINFO, Severity: FB_SEV_INFO

An error occurred with Severity **FB_SEV_INFO**. Additional information may be available in the Status History if it exists.

FB_ERR_INVALID_ENV_ID Short name: FEINEI, Severity : FB_SEV_ERROR

There is no Environment that corresponds to the given environment_id.

FB_ERR_INVALID_LIST_POINTER Short name: FEINLI, Severity : FB_SEV_ERROR

An invalid value was specified for the List Pointer.

FB_ERR_INVALID_PORT_ID Short name: FEINPO, Severity : FB_SEV_ERROR

There is no Port that corresponds to the given port_id.

FB_ERR_INVALID_SEQ_BUFFER_ID Short name: FEINSI, Severity : FB_SEV_ERROR

There is no sequential buffer (internal or external) that corresponds to the given sequential buffer_id.

FB_ERR_LAST_PRIMARY_ADDRESS Short name: FELPAD, Severity : FB_SEV_INFO

The value of the last primary address connection in the action or call to **FB_LIST_EXECUTE**.

FB_ERR_LAST_SECONDARY_ADDRESS Short name: FELSAD, Severity: FB_SEV_INFO

The value of the last secondary address in the action or call to **FB_LIST_EXECUTE**.

FB_ERR_LAST_LIST_POINTER Nom court: FELPTR, Sévérité : FB_SEV_ERROR

La valeur finale du pointeur de liste dans un appel à FB_LIST_EXECUTE.

FB_ERR_LIST_OVERFLOW Nom court: FELOV, Sévérité : FB_SEV_ERROR

Dans la construction d'une liste, l'espace alloué dans l'environnement pour la liste a été dépassé.

FB_ERR_LONG_TIMEOUT Nom court: FELOTO, Sévérité : FB_SEV_ERROR

Le temporisateur long a déclenché.

FB_ERR_LONG_TIMER_ON Nom court: FELOON, Sévérité : FB_SEV_INFO

Le temporisateur long est maintenu (c.-à-d. continue à compter).

FB_ERR_NO_ADDRESS_CONNECTION Nom court: FENOAS, Sévérité : FB_SEV_ERROR

Un cycle de données ou un cycle d'adresse secondaire a été tenté alors qu'il n'y avait aucune connexion d'adresse (AS/AK chaîné).

FB_ERR_NO_MORE_STATUS Nom court: FENMST, Sévérité : FB_SEV_ERROR

Il n'y a pas davantage d'état complémentaire pour cette action ou cette liste, ou pour le pointeur de liste spécifié.

FB_ERR_NO_PRIVILEGE Nom court: FENPRV, Sévérité : FB_SEV_ERROR

L'appelant a un privilège insuffisant pour exécuter la routine appelée. Ce code d'erreur doit être retourné, dans une réalisation qui supporte des utilisateurs multiples, lorsqu'une routine est appelée dont les effets peuvent s'appliquer à d'autres utilisateurs du système.

FB_ERR_NO_SUPPLEMENTARY_STATUS Nom court: FENSST, Sévérité: FB_SEV_INFO

Aucun état complémentaire n'est présent. Soit aucune routine d'action n'a encore été appelée dans cet environnement; soit le paramètre opérationnel FB_PAR_NO_STATUS était actif lorsque la dernière routine d'action a été appelée. Ce code d'erreur sera retourné lors d'un appel à FB_STATUS_FIND SUPPLEMENTARY ou à FB_STATUS_FIND_LIST SUPPLEMENTARY lorsque le code d'erreur de l'état complémentaire demandé n'a pas été trouvé. Il sera également retourné lors d'un appel à FB_STATUS_GET_LIST SUPPLEMENTARY ou FB_STATUS_FIND_LIST SUPPLEMENTARY, lorsqu'il n'y a pas d'état complémentaire correspondant au pointeur de liste spécifié.

FB_ERR_NORMAL Nom court: FENORM, Sévérité : FB_SEV_SUCCESS

Fin d'exécution normale. Aucune condition inattendue ne s'est produite. Des informations supplémentaires peuvent être disponibles dans l'historique d'état.

FB_ERR_NOT_CONNECTED Nom court: FENCON, Sévérité : FB_SEV_WARNING

Un essai de déconnexion d'une routine a échoué car il n'y avait pas de connexion.

FB_ERR_NOT_HOLDING_MASTERSHIP Nom court: FENHMA, Sévérité : FB_SEV_SUCCESS

La maîtrise du FASTBUS n'est pas conservée.

FB_ERR_LAST_LIST_POINTER Short name: FELPTR, Severity : FB_SEV_ERROR

The final value of the list pointer in a call to FB_LIST_EXECUTE.

FB_ERR_LIST_OVERFLOW Short name: FELOV, Severity : FB_SEV_ERROR

In constructing a List, the space allocated in the Environment for the List has been exceeded.

FB_ERR_LONG_TIMEOUT Short name: FELOTO, Severity : FB_SEV_ERROR

The long timer timed out.

FB_ERR_LONG_TIMER_ON Short name: FELOON, Severity : FB_SEV_INFO

The long timer being maintained (i.e. continues to tick).

FB_ERR_NO_ADDRESS_CONNECTION Short name: FENOAS, Severity : FB_SEV_ERROR

A data cycle or secondary address cycle was attempted while no address connection (AS/AK lock) was present.

FB_ERR_NO_MORE_STATUS Short name: FENMST, Severity : FB_SEV_ERROR

There is no more Supplementary Status available for this Action or List, or specified list pointer.

FB_ERR_NO_PRIVILEGE Short name: FENPRV, Severity : FB_SEV_ERROR

The caller has insufficient privilege to execute the called routine. This error code shall be returned by an Implementation that supports multi-users when a routine is called whose effect would impact other Users of the system.

FB_ERR_NO_SUPPLEMENTARY_STATUS Short name: FENSST, Severity: FB_SEV_INFO

No Supplementary Status is present. Either no Action Routine has yet been called using this Environment or the Operational Parameter FB_PAR_NO_STATUS was in force when the last Action Routine was called. This error code shall be returned from a call to FB_STATUS_FIND_SUPPLEMENTARY or FB_STATUS_FIND_LIST_SUPPLEMENTARY when the requested Supplementary Status Error Code was not found. It shall also be returned from a call to FB_STATUS_GET_LIST_SUPPLEMENTARY or FB_STATUS_FIND_LIST_SUPPLEMENTARY, when there is no Supplementary Status corresponding to the specified List Pointer.

FB_ERR_NORMAL Short name: FENORM, Severity : FB_SEV_SUCCESS

Normal completion. No unexpected condition occurred. Additional information may be available in the Status History.

FB_ERR_NOT_CONNECTED Short name: FENCON, Severity : FB_SEV_WARNING

An attempt to disconnect a routine failed because there was no connection.

FB_ERR_NOT_HOLDING_MASTERSHIP Short name: FENHMA, Severity: FB_SEV_SUCCESS

Mastership of the FASTBUS is not being held.

FB_ERR_NOT_MASTER

Nom court: FENOMA, Sévérité : FB_SEV_ERROR

Un cycle FASTBUS, autre qu'un cycle d'arbitrage, a été essayé tandis que le port n'avait pas la maîtrise du bus. Cela peut se produire lorsque des routines d'action primitives sont appelées dans un ordre incorrect, ou que le paramètre opérationnel FB_PAR_NO_ARBITRATION est FB_TRUE, mais que le port ne conserve pas la maîtrise du bus.

FB_ERR_OPEN

Nom court: FEOPEN, Sévérité : FB_SEV_WARNING

Un appel à FB_OPEN a été fait pendant une session FASTBUS qui était déjà ouverte. La session a été fermée et réouverte. L'environnement_id qui a été alloué pendant la session précédente a été libéré.

FB_ERR_OPERATION_PENDING

Nom court: FEPEND, Sévérité : FB_SEV_WARNING

L'action ou l'ensemble d'actions demandées n'ont pas été terminées. Ce code d'erreur est retourné lorsqu'une routine d'action est appelée dans un environnement en mode d'exécution immédiate ou que l'on appelle FB_LIST_EXECUTE, avec un paramètre opérationnel FB_PAR_NO_WAIT à FB_TRUE. Ce code d'erreur indique que l'action ou les actions demandées ont été démarrées sans erreur fatale.

FB_ERR_PIPELINE_TRANSFER

Nom court: FERPL, Sévérité : FB_SEV_ERROR

Ce code d'erreur n'est renvoyé que pour les transferts en pipe-line. Il indique qu'un problème, qui est apparu pendant l'exécution du transfert, ne peut être rapporté par un code d'erreur plus détaillé. Par exemple, le nombre de transitions DK reçues ne correspond pas au nombre de transitions DS.

FB_ERR_POINTER_OVERFLOW

Nom court: FEPOV, Sévérité : FB_SEV_ERROR

Un essai a été fait pour déplacer le pointeur d'un tampon séquentiel en dehors de la frontière du tampon.

FB_ERR_PORT_ACTIVE

Nom court: FEPAC, Sévérité : FB_SEV_ERROR

Un essai pour exécuter une action n'a pas été possible car le port était actif. Cela peut se produire dans un système multi-utilisateurs si le port exécute une action demandée par un autre utilisateur et ne met pas en queue les demandes d'utilisation.

FB_ERR_PORT_ALREADY_ALLOCATED

Nom court: FEPAL, Sévérité : FB_SEV_ERROR

Un essai pour allouer un port n'a pas été réussi car le port était déjà alloué à un autre utilisateur.

FB_ERR_PORT_NOT_ALLOCATED

Nom court: FEPNAL, Sévérité : FB_SEV_ERROR

Un essai pour exécuter une action qui nécessite l'allocation du port a été rejeté car le port n'était pas alloué, ou un essai pour désallouer un port qui n'était pas alloué.

FB_ERR_PORT_NOT_AVAILABLE

Nom court: FEPNOP, Sévérité : FB_SEV_ERROR

Le port n'est pas utilisable car il n'est pas dans un état opérationnel.

FB_ERR_PORT_OWNED

Nom court: FEPOWN, Sévérité : FB_SEV_INFO

Une demande pour allouer un port a été faite pour un port déjà alloué à l'utilisateur courant.

FB_ERR_NOT_MASTER

Short name: FENOMA, Severity : FB_SEV_ERROR

A FASTBUS cycle, other than Arbitration, was attempted while the Port did not have Mastership of the bus. This may occur when Primitive Action Routines are not called in the required order, or the Operational Parameter FB_PAR_NO_ARBITRATION is FB_TRUE, but the Port is not maintaining bus mastership.

FB_ERR_OPEN

Short name: FEOPEN, Severity : FB_SEV_WARNING

A call to FB_OPEN was made during a FASTBUS session that was already opened. The session was closed and then re-opened. Environment_ids that had been allocated during the previous session are released.

FB_ERR_OPERATION_PENDING

Short name: FEPEND, Severity : FB_SEV_WARNING

The requested Action or set of Actions has not yet completed. This error code is returned when an Action Routine is called with an Immediate Execution Mode Environment or FB_LIST_EXECUTE is called, with the Operational Parameter FB_PAR_NO_WAIT set to FB_TRUE. This error code indicates that the requested Action(s) have been started without a fatal error.

FB_ERR_PIPELINE_TRANSFER

Short name: FERPL, Severity : FB_SEV_ERROR

This error code is returned only for Pipeline transfers. It indicates a problem occurred during execution of the transfer, that could not be reported through any more detailed error code. For example, the number of DK transitions received disagreed with the number of DS transitions.

FB_ERR_POINTER_OVERFLOW

Short name: FEPOV, Severity : FB_SEV_ERROR

An attempt was made to move a sequential buffer pointer outside the buffer boundary.

FB_ERR_PORT_ACTIVE

Short name: FEPAC, Severity : FB_SEV_ERROR

An attempt to execute an action was not possible because the Port was active. This may occur in a multi-user system if a Port is executing an action requested by another User and does not queue requests for its use.

FB_ERR_PORT_ALREADY_ALLOCATED Short name: FEPAL, Severity: FB_SEV_ERROR

An attempt to allocate a Port was not successful because the Port was already allocated to another User.

FB_ERR_PORT_NOT_ALLOCATED

Short name: FEPNAL, Severity : FB_SEV_ERROR

An attempt to perform an action that requires Port allocation was rejected because the Port was not allocated, or an attempt was made to deallocate a Port that was not allocated.

FB_ERR_PORT_NOT_AVAILABLE

Short name: FEPNOP, Severity : FB_SEV_ERROR

The Port is not available for use because it is not in an Operational state.

FB_ERR_PORT_OWNED

Short name: FEPOWN, Severity : FB_SEV_INFO

A request to allocate a port was made for a port that is already allocated by the present user.

FB_ERR_PRIMARY_ADDRESS_SS0 Nom court: FEASS0, Sévérité : FB_SEV_SUCCESS

Un SS = 0 s'est produit dans un cycle d'adressage primaire.

FB_ERR_PRIMARY_ADDRESS_SS1 Nom court: FEASS1, Sévérité : FB_SEV_ERROR

Un SS = 1 s'est produit dans un cycle d'adressage primaire.

FB_ERR_PRIMARY_ADDRESS_SS2 Nom court: FEASS2, Sévérité : FB_SEV_ERROR

Un SS = 2 s'est produit dans un cycle d'adressage primaire.

FB_ERR_PRIMARY_ADDRESS_SS3 Nom court: FEASS3, Sévérité : FB_SEV_ERROR

Un SS = 3 s'est produit dans un cycle d'adressage primaire.

FB_ERR_PRIMARY_ADDRESS_SS4 Nom court: FEASS4, Sévérité : FB_SEV_ERROR

Un SS = 4 s'est produit dans un cycle d'adressage primaire.

FB_ERR_PRIMARY_ADDRESS_SS5 Nom court: FEASS5, Sévérité : FB_SEV_ERROR

Un SS = 5 s'est produit dans un cycle d'adressage primaire.

FB_ERR_PRIMARY_ADDRESS_SS6 Nom court: FEASS6, Sévérité : FB_SEV_ERROR

Un SS = 6 s'est produit dans un cycle d'adressage primaire.

FB_ERR_PRIMARY_ADDRESS_SS7 Nom court: FEASS7, Sévérité : FB_SEV_ERROR

Un SS = 7 s'est produit dans un cycle d'adressage primaire.

FB_ERR_READ_ONLY_PARAMETER Nom court: FEROPA, Sévérité : FB_SEV_ERROR

Un essai a été fait pour fixer un paramètre opérationnel en lecture seule.

FB_ERR_RETRY Nom court: FERTY, Sévérité: FB_SEV_INFO

Au moins un réessai, provoqué par une erreur, s'est produit pendant l'action demandée. L'action a finalement été un succès. Le code d'erreur correspondant à l'erreur originelle devrait être disponible avec ce code d'erreur.

La sévérité de FB_ERR_RETRY doit être utilisée pour modifier la sévérité rapportée pour l'erreur originelle.

FB_ERR_SAD_TIMEOUT Nom court: FESATO, Sévérité : FB_SEV_ERROR

Un temporisateur DK a déclenché pendant un cycle d'adresse secondaire.

FB_ERR_SECONDARY_ADDRESS_SS0 Nom court: FESSS0, Sévérité : FB_SEV_SUCCESS

Un SS = 0 s'est produit dans un cycle d'adresse secondaire.

FB_ERR_SECONDARY_ADDRESS_SS1 Nom court: FESSS1, Sévérité : FB_SEV_ERROR

Un SS = 1 s'est produit dans un cycle d'adresse secondaire.

FB_ERR_SECONDARY_ADDRESS_SS2 Nom court: FESSS2, Sévérité : FB_SEV_ERROR

Un SS = 2 s'est produit dans un cycle d'adresse secondaire.

FB_ERR_PRIMARY_ADDRESS_SS0	Short name: FEASS0,	Severity : FB_SEV_SUCCESS
An SS = 0 occurred on a primary address cycle.		
FB_ERR_PRIMARY_ADDRESS_SS1	Short name: FEASS1,	Severity : FB_SEV_ERROR
An SS = 1 occurred on a primary address cycle.		
FB_ERR_PRIMARY_ADDRESS_SS2	Short name: FEASS2,	Severity : FB_SEV_ERROR
An SS = 2 occurred on a primary address cycle.		
FB_ERR_PRIMARY_ADDRESS_SS3	Short name: FEASS3,	Severity : FB_SEV_ERROR
An SS = 3 occurred on a primary address cycle.		
FB_ERR_PRIMARY_ADDRESS_SS4	Short name: FEASS4,	Severity : FB_SEV_ERROR
An SS = 4 occurred on a primary address cycle.		
FB_ERR_PRIMARY_ADDRESS_SS5	Short name: FEASS5,	Severity : FB_SEV_ERROR
An SS = 5 occurred on a primary address cycle.		
FB_ERR_PRIMARY_ADDRESS_SS6	Short name: FEASS6,	Severity : FB_SEV_ERROR
An SS = 6 occurred on a primary address cycle.		
FB_ERR_PRIMARY_ADDRESS_SS7	Short name: FEASS7,	Severity : FB_SEV_ERROR
An SS = 7 occurred on a primary address cycle.		
FB_ERR_READ_ONLY_PARAMETER	Short name: FEROPA,	Severity : FB_SEV_ERROR
An attempt was made to set a read only Operational Parameter.		
FB_ERR_RETRY	Short name: FERTY,	Severity: FB_SEV_INFO
At least one retry occurred during the requested Action due to the occurrence of an error. The action was finally successful. The error code corresponding to the original error should be available in conjunction with this error code. The Severity of the FB_ERR_RETRY shall be used to modify the Severity reported for the original error.		
FB_ERR_SAD_TIMEOUT	Short name: FESATO,	Severity : FB_SEV_ERROR
A DK timer time-out occurred during a secondary address cycle.		
FB_ERR_SECONDARY_ADDRESS_SS0	Short name: FESSS0,	Severity: FB_SEV_SUCCESS
An SS = 0 occurred on a secondary address cycle.		
FB_ERR_SECONDARY_ADDRESS_SS1	Short name: FESSS1,	Severity: FB_SEV_ERROR
An SS = 1 occurred on a secondary address cycle.		
FB_ERR_SECONDARY_ADDRESS_SS2	Short name: FESSS2,	Severity: FB_SEV_ERROR
An SS = 2 occurred on a secondary address cycle.		

FB_ERR_SECONDARY_ADDRESS_SS3 Nom court: FESSS3, Sévérité : FB_SEV_ERROR

Un SS = 3 s'est produit dans un cycle d'adresse secondaire.

FB_ERR_SECONDARY_ADDRESS_SS4 Nom court: FESSS4, Sévérité : FB_SEV_ERROR

Un SS = 4 s'est produit dans un cycle d'adresse secondaire.

FB_ERR_SECONDARY_ADDRESS_SS5 Nom court: FESSS5, Sévérité : FB_SEV_ERROR

Un SS = 5 s'est produit dans un cycle d'adresse secondaire.

FB_ERR_SECONDARY_ADDRESS_SS6 Nom court: FESSS6, Sévérité : FB_SEV_ERROR

Un SS = 6 s'est produit dans un cycle d'adresse secondaire.

FB_ERR_SECONDARY_ADDRESS_SS7 Nom court: FESSS7, Sévérité : FB_SEV_ERROR

Un SS = 7 s'est produit dans un cycle d'adresse secondaire.

FB_ERR_SEQ_BUFFER_ACTIVE Nom court: FESBAC, Sévérité : FB_SEV_ERROR

Le tampon séquentiel correspondant au `buffer_id` séquentiel donné est en usage et ne peut être accédé par les routines qui modifient le pointeur du tampon.

FB_ERR_SEQ_BUFFER_EXISTS Nom court: FESBEX, Sévérité : FB_SEV_WARNING

Pour un appel à `FB_DECLARE_SEQ_BUFFER` ou `FB_DECLARE_EXT_SEQ_BUFFER`, le tampon séquentiel correspondant au `buffer_id` séquentiel donné existe déjà. Le tampon séquentiel a été redéclaré et les pointeurs réinitialisés.

FB_ERR_SEQ_BUFFER_ID_OVERFLOW Nom court: FESIOV, Sévérité : FB_SEV_ERROR

Il n'y a pas suffisamment de place pour déclarer un autre tampon séquentiel.

FB_ERR_SOFT_TIMER_ON Nom court: FESOON, Sévérité : FB_SEV_INFO

Le temporisateur logiciel est maintenu (c.-à-d. continue à compter).

FB_ERR_SOFTWARE_TIMEOUT Nom court: FESOTO, Sévérité : FB_SEV_ERROR

Le temporisateur logiciel a déclenché.

FB_ERR_STATUS_NOT_FOUND Nom court: FENSF, Sévérité : FB_SEV_ERROR

Aucune information d'état ne peut être trouvée pour la valeur d'entrée du pointeur de liste.

FB_ERR_STATUS_OVERFLOW Nom court: FESOV, Sévérité : FB_SEV_ERROR

Une place insuffisante est disponible pour mémoriser l'historique d'état qui vient d'être produit.

FB_ERR_SUCCESS Nom court: FESUCC, Sévérité : FB_SEV_SUCCESS

Une condition imprévue s'est produite qui n'affecte pas l'exécution de l'action. Des informations supplémentaires peuvent être disponibles dans l'historique d'état.

FB_ERR_SECONDARY_ADDRESS_SS3 Short name: FESSS3, Severity: FB_SEV_ERROR

An SS = 3 occurred on a secondary address cycle.

FB_ERR_SECONDARY_ADDRESS_SS4 Short name: FESSS4, Severity: FB_SEV_ERROR

An SS = 4 occurred on a secondary address cycle.

FB_ERR_SECONDARY_ADDRESS_SS5 Short name: FESSS5, Severity: FB_SEV_ERROR

An SS = 5 occurred on a secondary address cycle.

FB_ERR_SECONDARY_ADDRESS_SS6 Short name: FESSS6, Severity: FB_SEV_ERROR

An SS = 6 occurred on a secondary address cycle.

FB_ERR_SECONDARY_ADDRESS_SS7 Short name: FESSS7, Severity: FB_SEV_ERROR

An SS = 7 occurred on a secondary address cycle.

FB_ERR_SEQ_BUFFER_ACTIVE Short name: FESBAC, Severity : FB_SEV_ERROR

The sequential buffer corresponding to the given sequential buffer_id is in use and may not be accessed by routines that affect the buffer pointer.

FB_ERR_SEQ_BUFFER_EXISTS Short name: FESBEX, Severity : FB_SEV_WARNING

For a call to FB_DECLARE_SEQ_BUFFER or FB_DECLARE_EXT_SEQ_BUFFER, the sequential buffer corresponding to the given sequential buffer_id already existed. The sequential buffer was re-declared and the pointers re-initialized.

FB_ERR_SEQ_BUFFER_ID_OVERFLOW Short name: FESIOV, Severity : FB_SEV_ERROR

There is insufficient space to declare another sequential buffer.

FB_ERR_SOFT_TIMER_ON Short name: FESOON, Severity : FB_SEV_INFO

The software timer being maintained (i.e. continues to tick).

FB_ERR_SOFTWARE_TIMEOUT Short name: FESOTO, Severity : FB_SEV_ERROR

The software timer timed out.

FB_ERR_STATUS_NOT_FOUND Short name: FENSF, Severity : FB_SEV_ERROR

No Status information could be found for the input value of the List Pointer.

FB_ERR_STATUS_OVERFLOW Short name: FESOV, Severity : FB_SEV_ERROR

Insufficient space is available to store the Status History being generated.

FB_ERR_SUCCESS Short name: FESUCC, Severity : FB_SEV_SUCCESS

An unexpected condition occurred which did not affect execution of the Actions. Additional information may be available in the Status History.

FB_ERR_UNKNOWN Nom court: FEOOPS, Sévérité : FB_SEV_ERROR

Un code d'erreur inconnu a été rencontré.

FB_ERR_UNKNOWN_PARAMETER Nom court: FEUNPR, Sévérité : FB_SEV_ERROR

Un paramètre inconnu (non défini par la norme ni par la réalisation) a été passé à une routine.

FB_ERR_UNSPORT_OPERATION Nom court: FEUPOP, Sévérité : FB_SEV_ERROR

L'action demandée n'est pas supportée par le port actuel. Ce code peut être émis en mode d'exécution immédiate, ou en mode d'exécution différée pendant la validation ou l'exécution d'une liste.

FB_ERR_UNSPARAMETER Nom court: FEUPAR, Sévérité : FB_SEV_WARNING

Un paramètre opérationnel non supporté a été passé à une routine standard.

FB_ERR_UNSRoutine Nom court: FEUROU, Sévérité : FB_SEV_ERROR

Une routine définie dans cette norme, qui n'est pas supportée par la réalisation, a été appelée.

FB_ERR_USER_ARRAY_OVERFLOW Nom court: FEUAOV, Sévérité : FB_SEV_ERROR

Un appel à FB_GET_ENVIRONMENT a provoqué un débordement d'un tableau utilisateur.

FB_ERR_WAIT_TIMEOUT Nom court: FEWTTO, Sévérité : FB_SEV_ERROR

Le temporisateur d'attente (WAIT) a déclenché.

FB_ERR_WARNING Nom court: FEWARN, Sévérité : FB_SEV_WARNING

Une erreur s'est produite avec la sévérité FB_SEV_WARNING. Des informations supplémentaires peuvent être disponibles dans l'historique d'état s'il existe.

FB_ERR_UNKNOWN Short name: FEOOPS, Severity : FB_SEV_ERROR

An unknown error code has been encountered.

FB_ERR_UNKNOWN_PARAMETER Short name: FEUNPR, Severity: FB_SEV_ERROR

An unknown parameter (not defined by the standard or the Implementation) was passed to a routine.

FB_ERR_UNSPORT_OPERATION Short name: FEUPOP, Severity: FB_SEV_ERROR

The requested action is not supported on the current Port. This code may be generated in Immediate Execution Mode, or in Delayed Execution Mode during validation or execution of a List.

FB_ERR_UNSPARAMETER Short name: FEUPAR, Severity : FB_SEV_WARNING

An unsupported Operational Parameter was passed to a standard routine.

FB_ERR_UNSRoutine Short name: FEURQU, Severity : FB_SEV_ERROR

A routine defined in this standard was called that is unsupported by the implementation.

FB_ERR_USER_ARRAY_OVERFLOW Short name: FEUAOV, Severity : FB_SEV_ERROR

A call to FB_GET_ENVIRONMENT resulted in a User array overflow.

FB_ERR_WAIT_TIMEOUT Short name: FEWTTO, Severity : FB_SEV_ERROR

The WAIT timer timed out.

FB_ERR_WARNING Short name: FEWARN, Severity : FB_SEV_WARNING

An error occurred with Severity FB_SEV_WARNING. Additional information may be available in the Status History if it exists.

Annexe A. Récapitulation des noms des routines et des paramètres

Cette annexe récapitule les noms des routines définies dans cette norme et les arguments de chaque routine. Les routines sont placées dans l'ordre où elles apparaissent dans cette norme.

La liste suivante donne la signification de certains arguments:

retcod	= Return_code
eid	= Environment_id
pad	= Primary_Address
sad	= Secondary_Address
bufmode	= Buffer_Mode
buf	= buffer_specifier

Nom court	Nom long	
FBOPEN ou FB_OPEN		[retcod, le reste dépend de la réalisation]
FBCLOS ou FB_CLOSE		[retcod]
FCIENV ou FB_CREATE_IMMEDIATE_ENVIRONMENT		[retcod, eid]
FCDENV ou FB_CREATE_DELAYED_ENVIRONMENT		[retcod, eid, maxcalls]
FRLENV ou FB_RELEASE_ENVIRONMENT		[retcod, eid]
FRSENV ou FB_RESET_ENVIRONMENT		[retcod, eid]
FCOENV ou FB_COPY_ENVIRONMENT		[retcod, from_eid, to_eid]
FGTENV ou FB_GET_ENVIRONMENT		[retcod, eid, array, array_size, byte_count]
FSTENV ou FB_SET_ENVIRONMENT		[retcod, eid, array, byte_count]
FLEXEC ou FB_LIST_EXECUTE		[retcod, eid]
FLVAL ou FB_LIST_VALIDATE		[retcod, eid]
FLGPTR ou FB_LIST_GET_POINTER		[retcod, eid, list_pointer]
FBPINI ou FB_PAR_INIT		[retcod, eid, param_id]
FBPSET ou FB_PAR_SET		[retcod, eid, param_id, param_value]
FBPGET ou FB_PAR_GET		[retcod, eid, param_id, param_value]
FBPRD ou FB_PAR_READ		[retcod, eid, param_id, bufmode, buf]
FBPWRT ou FB_PAR_WRITE		[retcod, eid, param_id, bufmode, buf]
FBPUSH ou FB_PAR_PUSH		[retcod, eid]
FBPOP ou FB_PAR_POP		[retcod, eid]
FBDSEQ ou FB_DECLARE_SEQ_BUFFER		[retcod, eid, bufid, buffer, buf_size]
FBDEXT ou FB_DECLARE_EXT_SEQ_BUFFER		[retcod, eid, bufid, buffer, buf_size]
FBREL ou FB_RELEASE_SEQ_BUFFER		[retcod, eid, bufid]
FBRPTR ou FB_READ_BUFFER_POINTER		[retcod, eid, bufid, bufmode, buf]
FBWPTR ou FB_WRITE_BUFFER_POINTER		[retcod, eid, bufid, bufmode, buf]
FBDPTR ou FB_DISPLACE_BUFFER_POINTER		[retcod, eid, bufid, bufmode, buf]
FRD ou FB_READ_DAT		[retcod, eid, pad, sad, bufmode, buf]
FWD ou FB_WRITE_DAT		[retcod, eid, pad, sad, bufmode, buf]
FRC ou FB_READ_CSR		[retcod, eid, pad, sad, bufmode, buf]
FWC ou FB_WRITE_CSR		[retcod, eid, pad, sad, bufmode, buf]
FRDM ou FB_READ_DAT_MULT		[retcod, eid, pad, sad, bufmode, buf]
FWDM ou FB_WRITE_DAT_MULT		[retcod, eid, pad, sad, bufmode, buf]
FRCM ou FB_READ_CSR_MULT		[retcod, eid, pad, sad, bufmode, buf]
FWCM ou FB_WRITE_CSR_MULT		[retcod, eid, pad, sad, bufmode, buf]
FRDB ou FB_READ_DAT_BLOCK		[retcod, eid, pad, sad, bufmode, buf, maxbytes]
FWDB ou FB_WRITE_DAT_BLOCK		[retcod, eid, pad, sad, bufmode, buf, maxbytes]

Appendix A. Summary of Routine Names and Parameters

This appendix provides a summary of the Routine names defined in this standard and the arguments for each routine. The Routines are listed in the order in which they appear in the standard.

The meaning of some of the arguments listed is as follows:

retcod	= Return_code
eid	= Environment_id
pad	= Primary_Address
sad	= Secondary_Address
bufmode	= Buffer_Mode
buf	= buffer_specifier

Short Name	Long Name	
FBOPEN or FB_OPEN		[retcod, rest are implementation dependent]
FBCLOS or FB_CLOSE		[retcod]
FCIENV or FB_CREATE_IMMEDIATE_ENVIRONMENT		[retcod, eid]
PCDENV or FB_CREATE_DELAYED_ENVIRONMENT		[retcod, eid, maxcalls]
FRLENV or FB_RELEASE_ENVIRONMENT		[retcod, eid]
FRSENV or FB_RESET_ENVIRONMENT		[retcod, eid]
FCOENV or FB_COPY_ENVIRONMENT		[retcod, from_eid, to_eid]
FGTENV or FB_GET_ENVIRONMENT		[retcod, eid, array, array_size, byte_count]
FSTENV or FB_SET_ENVIRONMENT		[retcod, eid, array, byte_count]
FLEXEC or FB_LIST_EXECUTE		[retcod, eid]
FLVAL or FB_LIST_VALIDATE		[retcod, eid]
FLGPTR or FB_LIST_GET_POINTER		[retcod, eid, list_pointer]
FBPINI or FB_PAR_INIT		[retcod, eid, param_id]
FBPSET or FB_PAR_SET		[retcod, eid, param_id, param_value]
FBPGET or FB_PAR_GET		[retcod, eid, param_id, param_value]
FBPRD or FB_PAR_READ		[retcod, eid, param_id, bufmode, buf]
FBPWR or FB_PAR_WRITE		[retcod, eid, param_id, bufmode, buf]
FBPUSH or FB_PAR_PUSH		[retcod, eid]
FBPOP or FB_PAR_POP		[retcod, eid]
FBDSEQ or FB_DECLARE_SEQ_BUFFER		[retcod, eid, bufid, buffer, buf_size]
FBDEXT or FB_DECLARE_EXT_SEQ_BUFFER		[retcod, eid, bufid, buffer, buf_size]
FBREL or FB_RELEASE_SEQ_BUFFER		[retcod, eid, bufid]
FBRPTR or FB_READ_BUFFER_POINTER		[retcod, eid, bufid, bufmode, buf]
FBWPTR or FB_WRITE_BUFFER_POINTER		[retcod, eid, bufid, bufmode, buf]
FBDPTR or FB_DISPLACE_BUFFER_POINTER		[retcod, eid, bufid, bufmode, buf]
FRD or FB_READ_DAT		[retcod, eid, pad, sad, bufmode, buf]
FWD or FB_WRITE_DAT		[retcod, eid, pad, sad, bufmode, buf]
FRC or FB_READ_CSR		[retcod, eid, pad, sad, bufmode, buf]
FWC or FB_WRITE_CSR		[retcod, eid, pad, sad, bufmode, buf]
FRDM or FB_READ_DAT_MULT		[retcod, eid, pad, sad, bufmode, buf]
FWDM or FB_WRITE_DAT_MULT		[retcod, eid, pad, sad, bufmode, buf]
FRCM or FB_READ_CSR_MULT		[retcod, eid, pad, sad, bufmode, buf]
FWCM or FB_WRITE_CSR_MULT		[retcod, eid, pad, sad, bufmode, buf]
FRDB or FB_READ_DAT_BLOCK		[retcod, eid, pad, sad, bufmode, buf, maxbytes]
FWDB or FB_WRITE_DAT_BLOCK		[retcod, eid, pad, sad, bufmode, buf, maxbytes]

FRCB	ou FB_READ_CSR_BLOCK	[retcod, eid, pad, sad, bufmode, buf, maxbytes]
FWCB	ou FB_WRITE_CSR_BLOCK	[retcod, eid, pad, sad, bufmode, buf, maxbytes]
FRDBM	ou FB_READ_DAT_BLOCK_MULT	[retcod, eid, pad, sad, bufmode, buf, maxbytes]
FWDBM	ou FB_WRITE_DAT_BLOCK_MULT	[retcod, eid, pad, sad, bufmode, buf, maxbytes]
FRCBM	ou FB_READ_CSR_BLOCK_MULT	[retcod, eid, pad, sad, bufmode, buf, maxbytes]
FWCBM	ou FB_WRITE_CSR_BLOCK_MULT	[retcod, eid, pad, sad, bufmode, buf, maxbytes]
FRDSA	ou FB_READ_DAT_SA	[retcod, eid, pad, bufmode, buf]
FWDSA	ou FB_WRITE_DAT_SA	[retcod, eid, pad, bufmode, buf]
FRCSA	ou FB_READ_CSR_SA	[retcod, eid, pad, bufmode, buf]
FWCSA	ou FB_WRITE_CSR_SA	[retcod, eid, pad, bufmode, buf]
FRLEN	ou FB_READ_LENGTH	[retcod, eid, bufmode, buf]
FRRT	ou FB_READ_ROUTE_TABLE	[retcod, eid, pad, route_sad, bufmode, buf]
FWRT	ou FB_WRITE_ROUTE_TABLE	[retcod, eid, pad, route_sad, bufmode, buf]
FRRTB	ou FB_READ_ROUTE_TABLE_BLOCK	[retcod, eid, pad, route_sad, bufmode, buf, maxbytes]
FWRTB	ou FB_WRITE_ROUTE_TABLE_BLOCK	[retcod, eid, pad, route_sad, bufmode, buf, maxbytes]
FMOVD	ou FB_MOVE_DAT	[retcod, eid, pad1, sad1, pad2, sad2]
FMOVC	ou FB_MOVE_CSR	[retcod, eid, pad1, sad1, pad2, sad2]
FMOVDB	ou FB_MOVE_DAT_BLOCK	[retcod, eid, pad1, sad1, pad2, sad2, bufmode, buf, maxbytes]
FMOVCB	ou FB_MOVE_CSR_BLOCK	[retcod, eid, pad1, sad1, pad2, sad2, bufmode, buf, maxbytes]
FMODD	ou FB_MODIFY_DAT	[retcod, eid, pad, sad, oper_id, operand]
FMODC	ou FB_MODIFY_CSR	[retcod, eid, pad, sad, oper_id, operand]
FRDSEQ	ou FB_READ_DAT_SEQUENTIAL	[retcod, eid, start_pad, end_pad, step_pad, sad, bufmode, buf, buf_size, maxbytes]
FRCSEQ	ou FB_READ_CSR_SEQUENTIAL	[retcod, eid, start_pad, end_pad, step_pad, sad, bufmode, buf, buf_size, maxbytes]
FRDTPS	ou FB_READ_DAT_TP_SCAN	[retcod, eid, pad, sad, bufmode, buf, buf_size, maxbytes]
FRCTPS	ou FB_READ_CSR_TP_SCAN	[retcod, eid, pad, sad, bufmode, buf, buf_size, maxbytes]
FDVAL	ou FB_DEVICE_ALLOCATE	[retcod, eid, pad]
FDVDE	ou FB_DEVICE_DEALLOCATE	[retcod, eid, pad]
FBFIM	ou FB_SEND_FIM	[retcod, eid, pad, receiver_block, bufmode, buf, message_len]
FCARB	ou FB_CYCLE_ARBITRATE	[retcod, eid]
FCREL	ou FB_CYCLE_RELEASE_BUS	[retcod, eid]
FCPD	ou FB_CYCLE_PA_DAT	[retcod, eid, bufmode, buf]
FCPC	ou FB_CYCLE_PA_CSR	[retcod, eid, bufmode, buf]
FCPDM	ou FB_CYCLE_PA_DAT_MULT	[retcod, eid, bufmode, buf]
FCPCM	ou FB_CYCLE_PA_CSR_MULT	[retcod, eid, bufmode, buf]
FCDISC	ou FB_CYCLE_DISCONNECT	[retcod, eid]
FCRW	ou FB_CYCLE_READ_WORD	[retcod, eid, bufmode, buf]
FCRSA	ou FB_CYCLE_READ_SA	[retcod, eid, bufmode, buf]
FCWW	ou FB_CYCLE_WRITE_WORD	[retcod, eid, bufmode, buf]
FCWSA	ou FB_CYCLE_WRITE_SA	[retcod, eid, bufmode, buf]
FCRB	ou FB_CYCLE_READ_BLOCK	[retcod, eid, bufmode, buf, maxbytes]
FCWB	ou FB_CYCLE_WRITE_BLOCK	[retcod, eid, bufmode, buf, maxbytes]
FLR	ou FB_LINE_READ	[retcod, eid, line, bufmode, buf]

FRCB	or FB_READ_CSR_BLOCK	[retcod, eid, pad, sad, bufmode, buf, maxbytes]
FWCB	or FB_WRITE_CSR_BLOCK	[retcod, eid, pad, sad, bufmode, buf, maxbytes]
FRDBM	or FB_READ_DAT_BLOCK_MULT	[retcod, eid, pad, sad, bufmode, buf, maxbytes]
FWDBM	or FB_WRITE_DAT_BLOCK_MULT	[retcod, eid, pad, sad, bufmode, buf, maxbytes]
FRCBM	or FB_READ_CSR_BLOCK_MULT	[retcod, eid, pad, sad, bufmode, buf, maxbytes]
FWCBM	or FB_WRITE_CSR_BLOCK_MULT	[retcod, eid, pad, sad, bufmode, buf, maxbytes]
FRDSA	or FB_READ_DAT_SA	[retcod, eid, pad, bufmode, buf]
FWDSA	or FB_WRITE_DAT_SA	[retcod, eid, pad, bufmode, buf]
FRCSA	or FB_READ_CSR_SA	[retcod, eid, pad, bufmode, buf]
FWCSA	or FB_WRITE_CSR_SA	[retcod, eid, pad, bufmode, buf]
FRLEN	or FB_READ_LENGTH	[retcod, eid, bufmode, buf]
FRRT	or FB_READ_ROUTE_TABLE	[retcod, eid, pad, route_sad, bufmode, buf]
FWRT	or FB_WRITE_ROUTE_TABLE	[retcod, eid, pad, route_sad, bufmode, buf]
FRRTB	or FB_READ_ROUTE_TABLE_BLOCK	[retcod, eid, pad, route_sad, bufmode, buf, maxbytes]
FWRTB	or FB_WRITE_ROUTE_TABLE_BLOCK	[retcod, eid, pad, route_sad, bufmode, buf, maxbytes]
FMOVD	or FB_MOVE_DAT	[retcod, eid, pad1, sad1, pad2, sad2]
FMOVC	or FB_MOVE_CSR	[retcod, eid, pad1, sad1, pad2, sad2]
FMOVDB	or FB_MOVE_DAT_BLOCK	[retcod, eid, pad1, sad1, pad2, sad2, bufmode, buf, maxbytes]
FMOVCB	or FB_MOVE_CSR_BLOCK	[retcod, eid, pad1, sad1, pad2, sad2, bufmode, buf, maxbytes]
FMODD	or FB_MODIFY_DAT	[retcod, eid, pad, sad, oper_id, operand]
FMODC	or FB_MODIFY_CSR	[retcod, eid, pad, sad, oper_id, operand]
FRDSEQ	or FB_READ_DAT_SEQUENTIAL	[retcod, eid, start_pad, end_pad, step_pad, sad, bufmode, buf, buf_size, maxbytes]
FRCSEQ	or FB_READ_CSR_SEQUENTIAL	[retcod, eid, start_pad, end_pad, step_pad, sad, bufmode, buf, buf_size, maxbytes]
FRDTPS	or FB_READ_DAT_TP_SCAN	[retcod, eid, pad, sad, bufmode, buf, buf_size, maxbytes]
FRCTPS	or FB_READ_CSR_TP_SCAN	[retcod, eid, pad, sad, bufmode, buf, buf_size, maxbytes]
FDVAL	or FB_DEVICE_ALLOCATE	[retcod, eid, pad]
FDVDE	or FB_DEVICE_DEALLOCATE	[retcod, eid, pad]
FBFIM	or FB_SEND_FIM	[retcod, eid, pad, receiver_block, bufmode, buf, message_len]
FCARB	or FB_CYCLE_ARBITRATE	[retcod, eid]
FCREL	or FB_CYCLE_RELEASE_BUS	[retcod, eid]
FCPD	or FB_CYCLE_PA_DAT	[retcod, eid, bufmode, buf]
FCPC	or FB_CYCLE_PA_CSR	[retcod, eid, bufmode, buf]
FCPDM	or FB_CYCLE_PA_DAT_MULT	[retcod, eid, bufmode, buf]
FCPCM	or FB_CYCLE_PA_CSR_MULT	[retcod, eid, bufmode, buf]
FCDISC	or FB_CYCLE_DISCONNECT	[retcod, eid]
FCRW	or FB_CYCLE_READ_WORD	[retcod, eid, bufmode, buf]
FCRSA	or FB_CYCLE_READ_SA	[retcod, eid, bufmode, buf]
FCWW	or FB_CYCLE_WRITE_WORD	[retcod, eid, bufmode, buf]
FCWSA	or FB_CYCLE_WRITE_SA	[retcod, eid, bufmode, buf]
FCRB	or FB_CYCLE_READ_BLOCK	[retcod, eid, bufmode, buf, maxbytes]
FCWB	or FB_CYCLE_WRITE_BLOCK	[retcod, eid, bufmode, buf, maxbytes]
FLR	or FB_LINE_READ	[retcod, eid, line, bufmode, buf]

FLW	ou FB_LINE_WRITE	[retcod, eid, line, bufmode, buf]
FLRI	ou FB_LINE_READ_INTERNAL	[retcod, eid, line, bufmode, buf]
FBSRC	ou FB_SR_CONNECT	[retcod, SRsource, port, routine]
FBSRD	ou FB_SR_DISCONNECT	[retcod, SRsource]
FBSREN	ou FB_SR_ENABLE	[retcod, port]
FBSRDS	ou FB_SR_DISABLE	[retcod, port]
FBFIRC	ou FB_FIR_CONNECT	[retcod, port, receiver, filter_mask, filter_value, filter_word, routine]
FBFIRD	ou FB_FIR_DISCONNECT	[retcod, port, receiver, routine]
FBFIRE	ou FB_FIR_ENABLE	[retcod, port]
FBFIRS	ou FB_FIR_DISABLE	[retcod, port]
FBCOMC	ou FB_COMPLETION_CONNECT	[retcod, eid, routine]
FBCOMD	ou FB_COMPLETION_DISCONNECT	[retcod, eid]
FCOMWT	ou FB_COMPLETION_WAIT	[retcod, eid]
FCOMTS	ou FB_COMPLETION_TEST	[retcod, eid, result]
FBPRST	ou FB_PORT_RESET	[retcod, port]
FNPALL	ou FB_PORT_ALLOCATE	[retcod, port]
FNPDEA	ou FB_PORT_DEALLOCATE	[retcod, port]
FNPALT	ou FB_PORT_ALLOCATE_TMO	[retcod, port, timeout]
FBPRPT	ou FB_PORT_STATUS_REPORT	[retcode, port, error_code]
FBPTRN	ou FB_PORT_STATUS_TRANSLATE	[retcode, port, error_code, message]
FBVERS	ou FB_GET_SOFTWARE_VERSION	[retcod, port, hw_type, hw_version, sw_type, sw_version]
FSSSEV	ou FB_STATUS_SET_SEVERITY	[retcod, eid, error_code, severity]
FSSR	ou FB_STATUS_SET_RESPONSE	[retcod, eid, error_code, response]
FSGR	ou FB_STATUS_GET_RESPONSE	[retcod, eid, error_code, response]
FSCON	ou FB_STATUS_CONNECT	[retcod, eid, routine]
FSDISC	ou FB_STATUS_DISCONNECT	[retcod, eid]
FSGSUM	ou FB_STATUS_GET_SUMMARY	[retcod, eid, error_code, bytes_read, bytes_written]
FSGSUP	ou FB_STATUS_GET_SUPPLEMENTARY	[retcod, eid, error_code, param, pointer]
FSGLSU	ou FB_STATUS_LIST_SUPPLEMENTARY	[retcod, eid, list_pointer, error_code, param, pointer]
FSFSUP	ou FB_FIND_SUPPLEMENTARY	[retcod, eid, error_code, param, pointer]
FSFLSU	ou FB_FIND_LIST_SUPPLEMENTARY	[retcod, eid, list_pointer, error_code, param, pointer]
FSEVER	ou FB_STATUS_SEVERITY	[error_code]
FMATCH	ou FB_STATUS_MATCH	[error_code1, error_code2]
FSTHR	ou FB_STATUS_THRESHOLD	[error_code, severity]
FSENC	ou FB_STATUS_ENCODE	[new_error_code, old_error_code, severity]
FSRPT	ou FB_STATUS_REPORT	[retcod, eid, error_code, param, pointer]
FSTRAN	ou FB_STATUS_TRANSLATE	[retcod, eid, error_code, param, pointer, message]

A.1 Routines rangées par nom court

Nom court	Nom long	
FBCLOS	ou FB_CLOSE	[retcod]
FBCOMC	ou FB_COMPLETION_CONNECT	[retcod, eid, routine]
FBCOMD	ou FB_COMPLETION_DISCONNECT	[retcod, eid]
FBDEXT	ou FB_DECLARE_EXT_SEQ_BUFFER	[retcod, eid, bufid, buffer, buf_size]
FBDPTR	ou FB_DISPLACE_BUFFER_POINTER	[retcod, eid, bufid, bufmode, buf]
FBDSEQ	ou FB_DECLARE_SEQ_BUFFER	[retcod, eid, bufid, buffer, buf_size]
FBFIM	ou FB_SEND_FIM	[retcod, eid, pad, receiver_block, bufmode, buf, message_len]

FLW	or FB_LINE_WRITE	[retcod, eid, line, bufmode, buf]
FLRI	or FB_LINE_READ_INTERNAL	[retcod, eid, line, bufmode, buf]
FBSRC	or FB_SR_CONNECT	[retcod, SRsource, port, routine]
FBSRD	or FB_SR_DISCONNECT	[retcod, SRsource]
FBSREN	or FB_SR_ENABLE	[retcod, port]
FBSRDS	or FB_SR_DISABLE	[retcod, port]
FBFIRC	or FB_FIR_CONNECT	[retcod, port, receiver, filter_mask, filter_value, filter_word, routine]
FBFIRD	or FB_FIR_DISCONNECT	[retcod, port, receiver, routine]
FBFIRE	or FB_FIR_ENABLE	[retcod, port]
FBFIRS	or FB_FIR_DISABLE	[retcod, port]
FBCOMC	or FB_COMPLETION_CONNECT	[retcod, eid, routine]
FBCOMD	or FB_COMPLETION_DISCONNECT	[retcod, eid]
FCOMWT	or FB_COMPLETION_WAIT	[retcod, eid]
FCOMTS	or FB_COMPLETION_TEST	[retcod, eid, result]
FBPRST	or FB_PORT_RESET	[retcod, port]
FNPALL	or FB_PORT_ALLOCATE	[retcod, port]
FNPDEA	or FB_PORT_DEALLOCATE	[retcod, port]
FNPALT	or FB_PORT_ALLOCATE_TMO	[retcod, port, timeout]
FBPRPT	or FB_PORT_STATUS_REPORT	[retcode, port, error_code]
FBPTRN	or FB_PORT_STATUS_TRANSLATE	[retcode, port, error_code, message]
FBVERS	or FB_GET_SOFTWARE_VERSION	[retcod, port, hw_type, hw_version, sw_type, sw_version]
FSSSEV	or FB_STATUS_SET_SEVERITY	[retcod, eid, error_code, severity]
FSSR	or FB_STATUS_SET_RESPONSE	[retcod, eid, error_code, response]
FSGR	or FB_STATUS_GET_RESPONSE	[retcod, eid, error_code, response]
FSCON	or FB_STATUS_CONNECT	[retcod, eid, routine]
FSDISC	or FB_STATUS_DISCONNECT	[retcod, eid]
FSGSUM	or FB_STATUS_GET_SUMMARY	[retcod, eid, error_code, bytes_read, bytes_written]
FSGSUP	or FB_STATUS_GET_SUPPLEMENTARY	[retcod, eid, error_code, param, pointer]
FSGLSU	or FB_STATUS_LIST_SUPPLEMENTARY	[retcod, eid, list_pointer, error_code, param, pointer]
FSFSUP	or FB_FIND_SUPPLEMENTARY	[retcod, eid, error_code, param, pointer]
FSFLSU	or FB_FIND_LIST_SUPPLEMENTARY	[retcod, eid, list_pointer, error_code, param, pointer]
FSEVER	or FB_STATUS_SEVERITY	[error_code]
FMATCH	or FB_STATUS_MATCH	[error_code1, error_code2]
FSTHR	or FB_STATUS_THRESHOLD	[error_code, severity]
FSENC	or FB_STATUS_ENCODE	[new_error_code, old_error_code, severity]
FSRPT	or FB_STATUS_REPORT	[retcod, eid, error_code, param, pointer]
FSTRAN	or FB_STATUS_TRANSLATE	[retcod, eid, error_code, param, pointer, message]

A.1 Routines sorted by Short Name

Short Name	Long Name	
FBCLOS	or FB_CLOSE	[retcod]
FBCOMC	or FB_COMPLETION_CONNECT	[retcod, eid, routine]
FBCOMD	or FB_COMPLETION_DISCONNECT	[retcod, eid]
FBDEXT	or FB_DECLARE_EXT_SEQ_BUFFER	[retcod, eid, bufid, buffer, buf_size]
FBDPTR	or FB_DISPLACE_BUFFER_POINTER	[retcod, eid, bufid, bufmode, buf]
FBDSEQ	or FB_DECLARE_SEQ_BUFFER	[retcod, eid, bufid, buffer, buf_size]
FBFIM	or FB_SEND_FIM	[retcod, eid, pad, receiver_block, bufmode, buf, message_len]

FBFIRC	ou FB_FIR_CONNECT	[retcod, port, receiver, filter_mask, filter_value, filter_word, routine]
FBFIRD	ou FB_FIR_DISCONNECT	[retcod, port, receiver, routine]
FBFIRE	ou FB_FIR_ENABLE	[retcod, port]
FBFIRS	ou FB_FIR_DISABLE	[retcod, port]
FBOPEN	ou FB_OPEN	[retcod, reste dépend de la réalisation]
FBPGET	ou FB_PAR_GET	[retcod, eid, param_id, param_value]
FBPINI	ou FB_PAR_INIT	[retcod, eid, param_id]
FBPOP	ou FB_PAR_POP	[retcod, eid]
FBPRD	ou FB_PAR_READ	[retcod, eid, param_id, bufmode, buf]
FBPRST	ou FB_PORT_RESET	[retcod, port]
FBPSET	ou FB_PAR_SET	[retcod, eid, param_id, param_value]
FBPUSH	ou FB_PAR_PUSH	[retcod, eid]
FBPWRT	ou FB_PAR_WRITE	[retcod, eid, param_id, bufmode, buf]
FBREL	ou FB_RELEASE_SEQ_BUFFER	[retcod, eid, bufid]
FBRPTR	ou FB_READ_BUFFER_POINTER	[retcod, eid, bufid, bufmode, buf]
FBSRC	ou FB_SR_CONNECT	[retcod, SRsource, port, routine]
FBSRD	ou FB_SR_DISCONNECT	[retcod, SRsource]
FBSRDS	ou FB_SR_DISABLE	[retcod, port]
FBSREN	ou FB_SR_ENABLE	[retcod, port]
FBVERS	ou FB_GET_SOFTWARE_VERSION	[retcod, port, hw_type, hw_version, sw_type, sw_version]
FBWPTR	ou FB_WRITE_BUFFER_POINTER	[retcod, eid, bufid, bufmode, buf]
FCARB	ou FB_CYCLE_ARBITRATE	[retcod, eid]
FCDENV	ou FB_CREATE_DELAYED_ENVIRONMENT	[retcod, eid, maxcalls]
FCDISC	ou FB_CYCLE_DISCONNECT	[retcod, eid]
FCIENV	ou FB_CREATE_IMMEDIATE_ENVIRONMENT	[retcod, eid]
FCOENV	ou FB_COPY_ENVIRONMENT	[retcod, from_eid, to_eid]
FCOMTS	ou FB_COMPLETION_TEST	[retcod, eid, result]
FCOMWT	ou FB_COMPLETION_WAIT	[retcod, eid]
FCPC	ou FB_CYCLE_PA_CSR	[retcod, eid, bufmode, buf]
FCPCM	ou FB_CYCLE_PA_CSR_MULT	[retcod, eid, bufmode, buf]
FCPD	ou FB_CYCLE_PA_DAT	[retcod, eid, bufmode, buf]
FCPDM	ou FB_CYCLE_PA_DAT_MULT	[retcod, eid, bufmode, buf]
FCRB	ou FB_CYCLE_READ_BLOCK	[retcod, eid, bufmode, buf, maxbytes]
FCREL	ou FB_CYCLE_RELEASE_BUS	[retcod, eid]
FCRSA	ou FB_CYCLE_READ_SA	[retcod, eid, bufmode, buf]
FCRW	ou FB_CYCLE_READ_WORD	[retcod, eid, bufmode, buf]
FCWB	ou FB_CYCLE_WRITE_BLOCK	[retcod, eid, bufmode, buf, maxbytes]
FCWSA	ou FB_CYCLE_WRITE_SA	[retcod, eid, bufmode, buf]
FCWW	ou FB_CYCLE_WRITE_WORD	[retcod, eid, bufmode, buf]
FDVAL	ou FB_DEVICE_ALLOCATE	[retcod, eid, pad]
FDVDE	ou FB_DEVICE DEALLOCATE	[retcod, eid, pad]
FGTENV	ou FB_GET_ENVIRONMENT	[retcod, eid, array, array_size, byte_count]
FLEXEC	ou FB_LIST_EXECUTE	[retcod, eid]
FLGPTR	ou FB_LIST_GET_POINTER	[retcod, eid, list_pointer]
FLR	ou FB_LINE_READ	[retcod, eid, line, bufmode, buf]
FLRI	ou FB_LINE_READ_INTERNAL	[retcod, eid, line, bufmode, buf]
FLVAL	ou FB_LIST_VALIDATE	[retcod, eid]
FLW	ou FB_LINE_WRITE	[retcod, eid, line, bufmode, buf]
FMATCH	ou FB_STATUS_MATCH	[error_code1, error_code2]
FMODC	ou FB_MODIFY_CSR	[retcod, eid, pad, sad, oper_id, operand]
FMODD	ou FB_MODIFY_DAT	[retcod, eid, pad, sad, oper_id, operand]
FMOVC	ou FB_MOVE_CSR	[retcod, eid, pad1, sad1, pad2, sad2]
FMOVCB	ou FB_MOVE_CSR_BLOCK	[retcod, eid, pad1, sad1, pad2, sad2, bufmode, buf_size, maxbytes]
FMOVD	ou FB_MOVE_DAT	[retcod, eid, pad1, sad1, pad2, sad2]

FBFIRC	or FB_FIR_CONNECT	[retcod, port, receiver, filter_mask, filter_value, filter_word, routine]
FBFIRD	or FB_FIR_DISCONNECT	[retcod, port, receiver, routine]
FBFIRE	or FB_FIR_ENABLE	[retcod, port]
FBFIRS	or FB_FIR_DISABLE	[retcod, port]
FBOPEN	or FB_OPEN	[retcod, rest are implementation dependent]
FBPGET	or FB_PAR_GET	[retcod, eid, param_id, param_value]
FBPINI	or FB_PAR_INIT	[retcod, eid, param_id]
FBPOP	or FB_PAR_POP	[retcod, eid]
FBPRD	or FB_PAR_READ	[retcod, eid, param_id, bufmode, buf]
FBPRST	or FB_PORT RESET	[retcod, port]
FBPSET	or FB_PAR SET	[retcod, eid, param_id, param_value]
FBPUSH	or FB_PAR PUSH	[retcod, eid]
FBPWRT	or FB_PAR WRITE	[retcod, eid, param_id, bufmode, buf]
FBREL	or FB_RELEASE_SEQ_BUFFER	[retcod, eid, bufid]
FBRPTR	or FB_READ_BUFFER_POINTER	[retcod, eid, bufid, bufmode, buf]
FBSRC	or FB_SR_CONNECT	[retcod, SRsource, port, routine]
FBSRD	or FB_SR_DISCONNECT	[retcod, SRsource]
FBSRDS	or FB_SR_DISABLE	[retcod, port]
FBSREN	or FB_SR_ENABLE	[retcod, port]
FBVERS	or FB_GET_SOFTWARE_VERSION	[retcod, port, hw_type, hw_version, sw_type, sw_version]
FBWPTR	or FB_WRITE_BUFFER_POINTER	[retcod, eid, bufid, bufmode, buf]
FCARB	or FB_CYCLE_ARBITRATE	[retcod, eid]
FCDENV	or FB_CREATE_DELAYED_ENVIRONMENT	[retcod, eid, maxcalls]
FCDISC	or FB_CYCLE_DISCONNECT	[retcod, eid]
FCIENV	or FB_CREATE_IMMEDIATE_ENVIRONMENT	[retcod, eid]
FCOENV	or FB_COPY_ENVIRONMENT	[retcod, from_eid, to_eid]
FCOMTS	or FB_COMPLETION_TEST	[retcod, eid, result]
FCOMWT	or FB_COMPLETION_WAIT	[retcod, eid]
FCPC	or FB_CYCLE_PA_CSR	[retcod, eid, bufmode, buf]
FCPCM	or FB_CYCLE_PA_CSR_MULT	[retcod, eid, bufmode, buf]
FCPD	or FB_CYCLE_PA_DAT	[retcod, eid, bufmode, buf]
FCPDM	or FB_CYCLE_PA_DAT_MULT	[retcod, eid, bufmode, buf]
FCRB	or FB_CYCLE_READ_BLOCK	[retcod, eid, bufmode, buf, maxbytes]
FCREL	or FB_CYCLE_RELEASE_BUS	[retcod, eid]
FCRSA	or FB_CYCLE_READ_SA	[retcod, eid, bufmode, buf]
FCRW	or FB_CYCLE_READ_WORD	[retcod, eid, bufmode, buf]
FCWB	or FB_CYCLE_WRITE_BLOCK	[retcod, eid, bufmode, buf, maxbytes]
FCWSA	or FB_CYCLE_WRITE_SA	[retcod, eid, bufmode, buf]
FCWW	or FB_CYCLE_WRITE_WORD	[retcod, eid, bufmode, buf]
FDVAL	or FB_DEVICE_ALLOCATE	[retcod, eid, pad]
FDVDE	or FB_DEVICE_DEALLOCATE	[retcod, eid, pad]
FGTENV	or FB_GET_ENVIRONMENT	[retcod, eid, array, array_size, byte_count]
FLEXEC	or FB_LIST_EXECUTE	[retcod, eid]
FLGPTR	or FB_LIST_GET_POINTER	[retcod, eid, list_pointer]
FLR	or FB_LINE_READ	[retcod, eid, line, bufmode, buf]
FLRI	or FB_LINE_READ_INTERNAL	[retcod, eid, line, bufmode, buf]
FLVAL	or FB_LIST_VALIDATE	[retcod, eid]
FLW	or FB_LINE_WRITE	[retcod, eid, line, bufmode, buf]
FMATCH	or FB_STATUS_MATCH	[error_code1, error_code2]
FMODC	or FB_MODIFY_CSR	[retcod, eid, pad, sad, oper_id, operand]
FMODD	or FB_MODIFY_DAT	[retcod, eid, pad, sad, oper_id, operand]
FMOVC	or FB_MOVE_CSR	[retcod, eid, pad1, sad1, pad2, sad2]
FMOVCB	or FB_MOVE_CSR_BLOCK	[retcod, eid, pad1, sad1, pad2, sad2, bufmode, buf_size, maxbytes]
FMOVD	or FB_MOVE_DAT	[retcod, eid, pad1, sad1, pad2, sad2]

FMOVDB	ou FB_MOVE_DAT_BLOCK	[retcod, eid, pad1, sad1, pad2, sad2, bufmode, buf_size, maxbytes]
FNPALL	ou FB_PORT_ALLOCATE	[retcod, port]
FNPALT	ou FB_PORT_ALLOCATE_TMO	[retcod, port, timeout]
FNPDEA	ou FB_PORT_DEALLOCATE	[retcod, port]
FRC	ou FB_READ_CSR	[retcod, eid, pad, sad, bufmode, buf]
FRCB	ou FB_READ_CSR_BLOCK	[retcod, eid, pad, sad, bufmode, buf, maxbytes]
FRCBM	ou FB_READ_CSR_BLOCK_MULT	[retcod, eid, pad, sad, bufmode, buf, maxbytes]
FRCM	ou FB_READ_CSR_MULT	[retcod, eid, pad, sad, bufmode, buf]
FRCSA	ou FB_READ_CSR_SA	[retcod, eid, pad, bufmode, buf]
FRCSEQ	ou FB_READ_CSR_SEQUENTIAL	[retcod, eid, start_pad, end_pad, step_pad, sad, bufmode, buf, buf_size, maxbytes]
FRCTPS	ou FB_READ_CSR_TP_SCAN	[retcod, eid, pad, sad, bufmode, buf, sad, bufmode, buf, buf_size, maxbytes]
FRD	ou FB_READ_DAT	[retcod, eid, pad, sad, bufmode, buf]
FRDB	ou FB_READ_DAT_BLOCK	[retcod, eid, pad, sad, bufmode, buf, maxbytes]
FRDBM	ou FB_READ_DAT_BLOCK_MULT	[retcod, eid, pad, sad, bufmode, buf, maxbytes]
FRDM	ou FB_READ_DAT_MULT	[retcod, eid, pad, sad, bufmode, buf]
FRDSA	ou FB_READ_DAT_SA	[retcod, eid, pad, bufmode, buf]
FRDSEQ	ou FB_READ_DAT_SEQUENTIAL	[retcod, eid, start_pad, end_pad, step_pad, sad, bufmode, buf, buf_size, maxbytes]
FRDTPS	ou FB_READ_DAT_TP_SCAN	[retcod, eid, pad, sad, bufmode, buf, buf_size, maxbytes]
FRLEN	ou FB_READ_LENGTH	[retcod, eid, bufmode, buf]
FRLENV	ou FB_RELEASE_ENVIRONMENT	[retcod, eid]
FRRT	ou FB_READ_ROUTE_TABLE	[retcod, eid, pad, route_sad, bufmode, buf]
FRRTB	ou FB_READ_ROUTE_TABLE_BLOCK	[retcod, eid, pad, route_sad, bufmode, buf_size, maxbytes]
FRSENV	ou FB_RESET_ENVIRONMENT	[retcod, eid]
FSCON	ou FB_STATUS_CONNECT	[retcod, eid, routine]
FSDISC	ou FB_STATUS_DISCONNECT	[retcod, eid]
FSENC	ou FB_STATUS_ENCODE	[new_error_code, old_error_code, severity]
FSEVER	ou FB_STATUS_SEVERITY	[error_code]
FSFLSU	ou FB_FIND_LIST_SUPPLEMENTARY	[retcod, eid, list_pointer, error_code, param, pointer]
FSFSUP	ou FB_FIND_SUPPLEMENTARY	[retcod, eid, error_code, param, pointer]
FSGLSU	ou FB_STATUS_LIST_SUPPLEMENTARY	[retcod, eid, list_pointer, error_code, param, pointer]
FSGR	ou FB_STATUS_GET_RESPONSE	[retcod, eid, error_code, response]
FSGSUM	ou FB_STATUS_GET_SUMMARY	[retcod, eid, error_code, bytes_read, bytes_written]
FSGSUP	ou FB_STATUS_GET_SUPPLEMENTARY	[retcod, eid, error_code, param, error_code, param, pointer]
FSRPT	ou FB_STATUS_REPORT	[retcod, eid, error_code, param, pointer]
FSSR	ou FB_STATUS_SET_RESPONSE	[retcod, eid, error_code, response]
FSSSEV	ou FB_STATUS_SET_SEVERITY	[retcod, eid, error_code, severity]
FSTENV	ou FB_SET_ENVIRONMENT	[retcod, eid, array, byte_count]
FSTHR	ou FB_STATUS_THRESHOLD	[error_code, severity]
FSTRAN	ou FB_STATUS_TRANSLATE	[retcod, eid, error_code, param, pointer, message]
FWC	ou FB_WRITE_CSR	[retcod, eid, pad, sad, bufmode, buf]
FWCB	ou FB_WRITE_CSR_BLOCK	[retcod, eid, pad, sad, bufmode, buf, maxbytes]