

**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

AVA Documents
Document Sharing Hub

Numéros des publications

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

Publications consolidées

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

Validité de la présente publication

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

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

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

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

Terminologie, symboles graphiques et littéraux

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

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

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

Numbering

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

Consolidated publications

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

Validity of this publication

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

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

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

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

Terminology, graphical and letter symbols

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

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

* See web site address on title page.

**NORME
INTERNATIONALE
INTERNATIONAL
STANDARD**

**CEI
IEC
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



Documents

Document Sharing Hub

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

Withdrawn

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

Withdrawn

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.

Withdrawn

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.

Withdrawn



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.

Withdrawing

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.

Withdawn

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)
environnement_id FB_environnement_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 environnement_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: ERSENV 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).

Withdrawn



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).

Withdrawn

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.