NORME INTERNATIONALE

INTERNATIONAL STANDARD

CEI
IEC
60713

Première édition
First edition
1981-01

Sous-programmes CAMAC

Subroutines for CAMAC

Numéro de référence
Reference number
CEI/IEC 60713: 1981

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
60713

Première édition
First edition
1981-01

Sous-programmes CAMAC

Subroutines for CAMAC

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

Commission Electrotechnique Internationale
International Electrotechnical Commission
Международная Электротехническая Комиссия

CODE PRIX
PRICE CODE U

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

SOMMAIRE

CONTENTS

Articles Pages

Clause Page

COMMISSION ÉLECTROTECHNIQUE INTERNATIONALE

SOUS-PROGRAMMES CAMAC

PRÉAMBULE

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.

PRÉFACE

La présente norme a été établie par le Comité d'Etudes N° 45 de la CEI: Instrumentation nucléaire.

Un premier projet fut discuté lors de la réunion tenue à Stockholm en 1980. A la suite de cette réunion, un projet, document 45(Bureau Central)142, fut soumis à l'approbation des Comités nationaux suivant la Règle des Six Mois en août 1980.

Les Comités nationaux des pays ci-après se sont prononcés explicitement en faveur de la publication:

Afrique du Sud (République d')	Finlande
Allemagne	France
Australie	Italie
Autriche	Pays-Bas
Belgique	Pologne
Canada	République Démocratique Allemande
Egypte	Turquie
Espagne	Union des Républiques Socialistes Soviétiques
Etats-Unis d'Amérique	

Autres publications de la CEI citées dans la présente norme:

Publications n^{os}
- 516: Système modulaire d'instrumentation pour le traitement de l'information; système CAMAC.
- 552: Système CAMAC - Organisation de systèmes multichâssis. Spécification de l'Interconnexion de branche et du contrôleur de châssis type A1.
- 640: Système CAMAC - Interface pour Interconnexion de Branche Série.
- 677: Transferts de bloc dans les systèmes CAMAC.

Autres publications:
- IML: CAMAC - Définition de l'IML, langage à utiliser dans les systèmes CAMAC, ESONE/IML/01, octobre 1974, Secrétariat ESONE, et TID-26615, janvier 1975, DOE, Washington, D.C., Etats-Unis d'Amérique.

INTERNATIONAL ELECTROTECHNICAL COMMISSION

SUBROUTINES FOR CAMAC

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 rule should, as far as possible, be clearly indicated in the latter.

PREFACE

This standard has been prepared by IEC Technical Committee No. 45: Nuclear Instrumentation.

A first draft was discussed at the meeting held in Stockholm in 1980. As a result of this meeting, a draft, Document 45(Central Office)142, was submitted to the National Committees for approval under the Six Months' Rule in August 1980.

The National Committees of the following countries voted explicitly in favour of publication:

Australia	Italy
Austria	Netherlands
Belgium	Poland
Canada	South Africa (Republic of)
Egypt	Spain
Finland	Turkey
France	Union of Soviet Socialist Republics
German Democratic Republic	United States of America
Germany	

Other IEC publications quoted in this standard:

Publications Nos. 516: A Modular Instrumentation System for Data Handling; CAMAC System.
552: CAMAC - Organization of Multi-crate Systems. Specification of the Branch-highway and CAMAC Crate Controller Type A1.
640: CAMAC - Serial Highway Interface System.
677: Block Transfers in CAMAC Systems.

Other publications: IML: CAMAC - The Definition of IML, a Language for use in CAMAC Systems, ESONE/IML/01, October 1974, ESONE Secretariat, and TID-26615, January 1975, DOE, Washington, D.C., United States of America.

SOUS-PROGRAMMES CAMAC

1. Domaine d'application

La présente norme s'applique aux sous-programmes pour les systèmes CAMAC tels qu'ils sont définis dans la Publication 516 de la CEI: Système modulaire d'instrumentation pour le traitement de l'information; système CAMAC. Son application ne doit pas s'opposer aux dispositions obligatoires de la Publication 516 de la CEI ni provoquer d'opposition à leur encontre.

2. Objet

La présente norme offre un ensemble de sous-programmes destinés à fournir des moyens généraux de communication avec les systèmes CAMAC définis dans la Publication 516 de la CEI.

Cette norme sera d'un intérêt primordial pour ceux qui souhaitent écrire leurs propres programmes de traitement de données dans un langage de haut niveau tel que le FORTRAN.

Le taux de transfert de données réalisable dépend naturellement d'un certain nombre de facteurs comme le langage utilisé, le système d'exploitation, le compilateur, les méthodes employées pour réaliser la fonction du sous-programme, le niveau d'exécution de celui-ci ainsi que des performances de l'ordinateur.

3. Généralités

Cette norme décrit un ensemble de sous-programmes normalisés permettant à différents langages de programmation de communiquer avec les systèmes CAMAC.

Intentionnellement, ces sous-programmes sont adaptés à l'utilisation du langage FORTRAN, bien qu'ils ne soient pas limités à ce seul langage. L'annexe B décrit une mise en œuvre recommandée de ces sous-programmes en FORTRAN.

La présente approche est largement fondée sur les publications de l'ESONE/IML/01 et TID-26615 sur IML: CAMAC - Définition de l'IML, langage à utiliser dans les systèmes CAMAC. Une distinction est faite entre les «déclarations», qui sont utilisées pour dénommer et spécifier les entités de l'ordinateur ou celles du CAMAC, et les «actions», qui sont employées pour réaliser les divers mouvements de données et effectuer les tests de conditions, le tout formant la portion du programme relative au CAMAC. Autant que possible, la nomenclature de l'IML-M1 a été suivie de façon à bénéficier de la routine actuelle de ce système et assurer une terminologie et un style aussi conformes que possible aux divers documents de logiciel CAMAC.

Du fait de l'emploi étendu du CAMAC avec des ordinateurs dont la longueur de mot est inférieure à 24 bits et parce qu'il existe de grandes différences entre les caractéristiques des ordinateurs et des systèmes d'exploitation, des caractéristiques spéciales dépendantes du système sont souvent exigées pour donner une plus grande efficacité ou pour utiliser au mieux les caractéristiques de ces systèmes particuliers. La partie principale de cette norme décrit des sous-programmes qui, au niveau de l'interface de l'utilisateur, dépendent seulement des caractéristiques CAMAC, et qui, par conséquent, lorsqu'ils sont écrits dans n'importe quel langage de procédure normalisé, devraient être indépendants de l'ordinateur. L'annexe A décrit des sous-programmes qui ne dépendent pas seulement du CAMAC, mais jusqu'à un

SUBROUTINES FOR CAMAC

1. Scope

This standard covers requirements for subroutines for CAMAC systems as defined in IEC Publication 516: A Modular Instrumentation System for Data Handling; CAMAC System. Its application shall not conflict or cause conflict with the mandatory requirements of IEC Publication 516.

2. Object

Recommendations are presented for a set of software subroutines to provide a general capability for communications with CAMAC systems as defined in IEC Publication 516.

They will be of primary interest to those who wish to write their own data-processing programs in a high level programming language, such as FORTRAN.

The achievable data transfer rate is of course dependent on a number of factors including the language used, the operating system, the compiler, the method and level of subroutine implementation and the computer.

3. General

This standard describes a set of standard subroutines to provide access to CAMAC facilities in a variety of computer-programming languages.

It is specifically intended that the subroutines be suitable for use with FORTRAN although they are not restricted to that language. Appendix B describes a recommended implementation of the subroutines explicitly for FORTRAN.

The present approach is based largely on ESONE/IML/01 and TID-26615 publications on IML: CAMAC - The Definition of IML, a Language for use in CAMAC Systems. A distinction is made between "declarations", which are used to name and specify computer and CAMAC entities, and "actions", which are used to implement the various data movements and condition tests which make up the CAMAC-related portion of a program. As far as possible, the nomenclature of IML-M1 has been followed in order to take advantage of existing familiarity with that system and to provide as uniform a terminology and style as possible among the various CAMAC software documents.

Because of the widespread use of CAMAC on computers with a word length of less than 24 bits and because of the great differences in computer and operating-system features, special, system-dependent features are often required to provide greater efficiency or to make appropriate use of the features of particular systems. The main body of this standard describes subroutines which, at the user interface, depend only on the features of CAMAC and therefore, when implemented in any standard procedural language, should be computer-independent. Appendix A describes subroutines which depend not only on CAMAC, but to some extent on individual computers. They cannot be made independent of the system on which they are implemented, and the user should take special precautions when it is necessary

certain point d'ordinateurs particuliers. Ils ne peuvent pas être rendus indépendants du système pour lequel ils ont été écrits, et l'utilisateur doit prendre des précautions spéciales quand il est nécessaire de les incorporer dans un programme. Un tel programme ne peut pas être transférable d'un ordinateur à un autre sans modification.

Les sous-programmes ont été groupés en trois catégories pour permettre des mises en œuvre à différents niveaux normalisés. Le niveau le plus bas requiert seulement deux sous-programmes mais donne cependant accès à la plupart des possibilités des systèmes CAMAC. Dans les niveaux plus élevés, on a ajouté des sous-programmes qui permettent des procédures d'écriture dans une terminologie plus mnémonique, offrent une meilleure manipulation des LAM, autorisent des procédures indépendantes du type d'Interconnexion de branche CAMAC utilisé et procurent des transferts de bloc efficaces.

4. Spécifications fonctionnelles

Cet article présente et décrit en détail tous les sous-programmes recommandés. Du fait que de nombreuses mises en œuvre n'ont pas besoin de l'ensemble des sous-programmes, ceux-ci ont été divisés en trois parties correspondant aux niveaux recommandés de réalisation. Le niveau A, le plus simple, fait appel aux sous-programmes du paragraphe 4.1; le niveau B, niveau intermédiaire, requiert les sous-programmes des paragraphes 4.1 et 4.2, et le niveau C qui est le plus élevé, nécessite l'emploi des sous-programmes des paragraphes 4.1, 4.2 et 4.3.

Deux dispositions CAMAC ne sont pas accessibles par ces sous-programmes: la réponse X à une action et la commande BZ. On peut néanmoins les obtenir en utilisant les sous-programmes dépendants du système décrits dans l'annexe A.

On a adopté, pour les dénominations, des conventions compatibles avec les langages de nombreux ordinateurs. Afin que l'utilisateur évite simplement les difficultés inhérentes aux dénominations, le nom de chaque sous-programme normalisé débute par la lettre «C». Le code de la seconde lettre précise sa fonction générale. Six lettres sont utilisées à cette fin:

«C» indique que le sous-programme exécute une fonction de contrôle-commande;

«D» indique que le sous-programme est une déclaration d'entité CAMAC;

«F» indique que le sous-programme transfère des mots complets de données (24 bits);

«G» indique que le sous-programme décompose une entité CAMAC désignée en ses composantes d'adresse;

«S» indique que le sous-programme transfère des mots tronqués de données (moins de 24 bits);

«T» indique que le sous-programme contrôle l'état d'un signal ou l'indication d'état.

Les lettres restantes de chaque nom (jusqu'à un maximum de six) des sous-programmes sont choisies pour leurs valeurs mnémoniques caractérisant la fonction que le sous-programme effectue.

Du fait qu'aucun langage particulier n'est pris en compte dans le corps de la norme, aucune syntaxe ne peut être définie pour un appel de sous-programme. Les sous-programmes sont décrits en fonction d'un nom de sous-programme et d'une séquence ordonnée de paramètres. Dans toute la mesure possible, chaque mise en œuvre doit conserver ses noms et l'ordre des paramètres associés. Pour la même raison, aucune forme spécifique ne peut-être définie pour les paramètres des sous-programmes. Les mots de données CAMAC sont des chaînes de 24 bits. Une mise en œuvre doit pouvoir représenter une telle chaîne. D'autres paramètres sont représentés soit en valeurs entières, soit en valeurs logiques de vérité «vrai» ou «faux». Dans

to incorporate them into a program. Such a program may not be transportable from one computer to another without modification.

The subroutines have been grouped into three subsets in order to provide different standard levels of implementation. The lowest level requires only two subroutines, but nevertheless gives access to most of the facilities which can be found in CAMAC systems. In higher levels of implementation, subroutines are added which permit procedures to be written in more mnemonic terminology, provide better handling of LAM's, permit procedures to be independent of the type of CAMAC highway used and provide efficient block-transfer capability.

4. Functional specifications

This clause introduces and describes in detail all the recommended subroutines. Since many implementations will not require the complete set, the subroutines are grouped into three subsets corresponding to the recommended implementation levels. Level A, the simplest, requires only the subroutines from Sub-clause 4.1. Level B, an intermediate level, requires the subroutines from Sub-clauses 4.1 and 4.2. Level C is the highest level and requires implementation of the subroutines from Sub-clauses 4.1, 4.2 and 4.3.

Two CAMAC facilities are not available through the use of these subroutines: the X response from an action and the BZ command. These facilities are available through the use of system-dependent subroutines described in Appendix A.

Naming conventions, compatible with the requirements of many computer languages, have been adopted. In order to make it simple for a user to avoid name conflicts, the name of each recommended subroutine begins with the letter "C". The second letter of the subroutine name is coded to indicate the general function of the subroutine. Six letters have been used for this purpose:

"C" indicates that the subroutine performs a control function;

"D" indicates that the subroutine is a declaration of a CAMAC entity;

"F" indicates that the subroutine transfers full-length (24-bit) data words;

"G" indicates that the subroutine analyzes a named CAMAC entity into its address components;

"S" indicates that the subroutine transfers short (less than 24-bit) data words;

"T" indicates that the subroutine tests the state of a signal or status indication.

The remaining letters of each subroutine name (to a maximum of six) are chosen for their mnemonic value in identifying which function the subroutine performs.

Since no particular language is assumed in the body of the standard, no syntax can be defined for a subroutine call. The subroutines are described in terms of a subroutine name and an ordered sequence of parameters. As far as possible every implementation should retain the designated name and the order of the parameters. For the same reason, no specific form can be defined for the subroutine parameters. CAMAC data words are bit strings with a length of 24 bits. An implementation must have the capacity to represent such strings. Other subroutine parameters are represented either as integer values or as the logical values "true" or "false". In this, the parameters of subroutines are described as variables or arrays of types CAMAC

cette norme, les paramètres des sous-programmes sont représentés comme des variables ou tableaux du type mot CAMAC, entier ou logique. Pour chaque réalisation, des unités de stockage appropriées et des formats de données doivent être choisis pour chacune de ces entités généralisées.

4.1 *Sous-programmes de base*

Ces deux sous-programmes, qui sont nécessaires dans toutes les mises en œuvre, forment le niveau A. Le premier fournit le moyen de définir l'adresse d'une entité CAMAC et d'y avoir accès. Le second est utilisé pour effectuer les opérations CAMAC sur les entités définies. En principe, toute entité CAMAC pour laquelle existe un mode d'accès normalisé défini est accessible par ces deux sous-programmes. En pratique, certains systèmes peuvent comporter des restrictions dans l'utilisation des contrôleurs de châssis ou de modules d'un autre système.

4.1.1 *Déclaration d'un registre CAMAC*

Nom: CDREG.

Paramètres: *ext* (adresse externe, voir paragraphe 5.1);
b (numéro de branche, voir paragraphe 5.2);
c (numéro de châssis, voir paragraphe 5.3);
n (numéro de station, voir paragraphe 5.4);
a (sous-adresse, voir paragraphe 5.5).

Fonction: CDREG réunit le numéro de branche *b*, le numéro de châssis *c*, le numéro de station *n* et la sous-adresse *a* en une forme adéquate dépendante du système utilisé et stocke le résultat en *ext.* Du fait que la méthode de codage dépend de la réalisation, les contenus de *ext* ne devraient pas être modifiés par le programme. Quelques sous-programmes (voir paragraphe 4.2) n'exigent qu'une adresse de châssis; si les paramètres *n* et *a* ont tous deux la valeur zéro, CDREG code une adresse de châssis et la stocke en *ext.*

4.1.2 *Exécution d'une action simple CAMAC*

Nom: CFSA.

Paramètres: *f* (code de fonction, voir paragraphe 5.6);
ext (adresse externe, voir paragraphe 5.1);
int (mot de données CAMAC, voir paragraphe 5.7);
q (réponse Q, voir paragraphe 5.8).

Fonction: CFSA provoque une action CAMAC spécifiée par le code de fonction *f*, qui doit être exécutée à l'adresse CAMAC spécifiée par *ext.* Si *f* contient un code de lecture ou d'écriture, un transfert de données sur 24 bits est effectué entre le registre CAMAC adressé par *ext* et l'emplacement du stockage *int* de l'ordinateur. Sans quoi *int* est ignoré. L'état de Q résultant de l'opération est stocké en *q*, «vrai» si Q = 1, «faux» si Q = 0.

4.2 *Sous-programmes d'action simple*

Ces sous-programmes, en même temps que ceux qui sont décrits dans le paragraphe 4.1, constituent la mise en œuvre de niveau B, qui fournit un ensemble complet de spécifications des actions CAMAC simples dans une forme mnémonique, concise et indépendante du type d'Interconnexion de branche ou de contrôleur de châssis. Ils permettent de déclarer les appels (LAM) et d'exécuter des actions relatives à ceux-ci en utilisant des constructions indépendantes du mode d'accès au LAM (c'est-à-dire accès de la sous-adresse ou du registre).

word, integer, or logical. For each implementation, appropriate storage units and data formats must be chosen for each of these generalized entities.

4.1 *Primary Subroutines*

These two subroutines, which are required in all implementations, make up level A. The first provides the capability to define the address of a CAMAC entity and to access it. The second is used to perform CAMAC operations on the defined entities. In principle any CAMAC entity for which there is a defined standard mode of access can be accessed through the use of these two subroutines. In practice, some systems may contain restrictions on the use of crate controllers or other system modules.

4.1.1 *Declare CAMAC Register*

Name: CDREG.

Parameters: *ext* (external address, see Sub-clause 5.1);
b (branch number, see Sub-clause 5.2);
c (crate number, see Sub-clause 5.3);
n station number, see Sub-clause 5.4);
a (subaddress, see Sub-clause 5.5).

Function: CDREG combines the branch number *b*, the crate number *c*, the station number *n*, and the subaddress *a* into a convenient system-dependent form and stores the result in *ext*. Since the method of encoding depends on the implementation, the contents of *ext* should not be modified by the program. Some subroutines (see Sub-clause 4.2) require only a crate address; if the parameters *n* and *a* are both zero, CDREG encodes a crate address and stores it in *ext*.

4.1.2 *Perform Single CAMAC Action*

Name: CFSA.

Parameters: *f* (function code, see Sub-clause 5.6);
ext (external address, see Sub-clause 5.1);
int (CAMAC data word, see Sub-clause 5.7);
q (Q response, see Sub-clause 5.8).

Function: CFSA causes the CAMAC action specified by the function code of *f* to be performed at the CAMAC address specified by *ext*. If *f* contains a read or write code, a 24-bit data transfer occurs between the CAMAC register addressed by *ext* and the computer storage location *int*. Otherwise *int* is ignored. The state of Q resulting from the operation is stored in *q*, "true" if Q = 1, "false" if Q = 0.

4.2 *Single-Action Subroutines*

These subroutines, together with those described in Sub-clause 4.1, form the level B implementation, which provides a complete facility for specifying single CAMAC actions in a way which is mnemonic, compact, and independent of the type of highway or crate controller. Facilities are provided for declaring LAM's and performing LAM actions using constructions which are independent of the LAM access mode (i.e. subaddress or register access).

4.2.1 *Emission du signal d'initialisation sur l'Interconnexion*

Nom: CCCZ.

Paramètre: *ext* (adresse externe, voir paragraphe 5.1).

Fonction: CCCZ provoque l'émission du signal d'initialisation (Z) sur l'Interconnexion du châssis spécifié par *ext.*

4.2.2 *Emission de la remise à zéro du châssis*

Nom: CCCC.

Paramètre: *ext* (adresse externe, voir paragraphe 5.1).

Fonction: CCCC provoque l'émission du signal de remise à zéro (C) sur l'Interconnexion du châssis spécifié par *ext.*

4.2.3 *Etablissement ou suppression de l'inhibition sur l'Interconnexion*

Nom: CCCI.

Paramètres: *ext* (adresse externe, voir paragraphe 5.1);
l (valeur logique de vérité, voir paragraphe 5.9).

Fonction: CCCI provoque l'établissement de l'inhibition (I) sur l'Interconnexion d'un châssis spécifié par *ext* si la valeur de *l* est à «vrai» ou sa suppression si la valeur de *l* est à «faux».

4.2.4 *Contrôle de l'inhibition sur l'Interconnexion*

Nom: CTCI.

Paramètres: *ext* (adresse externe, voir paragraphe 5.1);
l (valeur logique de vérité, voir paragraphe 5.9).

Fonction: CTCI met la valeur de *l* à «vrai» si le signal d'inhibition sur l'Interconnexion dans le châssis spécifié par *ext* est présent et met la valeur de *l* à «faux» si ce signal est absent.

4.2.5 *Mise en service ou hors service de la demande dans un châssis*

Nom: CCCD.

Paramètres: *ext* (adresse externe, voir paragraphe 5.1);
l (valeur logique de vérité, voir paragraphe 5.9).

Fonction: CCCD provoque la mise en service de la demande du châssis spécifié par *ext* si la valeur de *l* est à «vrai» et sa mise hors service si la valeur de *l* est à «faux».

4.2.6 *Contrôle de la mise en service de la demande dans un châssis*

Nom: CTCD.

Paramètres: *ext* (adresse externe, voir paragraphe 5.1);
l (valeur logique de vérité, voir paragraphe 5.9).

Fonction: CTCD met la valeur de *l* à «vrai» si la demande du châssis spécifié par *ext* est en service et met la valeur de *l* à «faux» si la demande du châssis est hors service.

4.2.1 *Generate Dataway Initialize*

Name: CCCZ.

Parameter: *ext* (external address, see Sub-clause 5.1).

Function: CCCZ causes Dataway Initialize (Z) to be generated in the crate specified by *ext.*

4.2.2 *Generate Crate Clear*

Name: CCCC.

Parameter: *ext* (external address, see Sub-clause 5.1).

Function: CCCC causes Dataway Clear (C) to be generated in the crate specified by *ext.*

4.2.3 *Set or Clear Dataway Inhibit*

Name: CCCI.

Parameters: *ext* (external address, see Sub-clause 5.1);
l (logical truth value, see Sub-clause 5.9).

Function: CCCI causes Dataway Inhibit (I) to be set in the crate specified by *ext* if the value of *l* is "true" and to be reset if the value of *l* is "false".

4.2.4 *Test Dataway Inhibit*

Name: CTCI.

Parameters: *ext* (external address, see Sub-clause 5.1);
l (logical truth value, see Sub-clause 5.9).

Function: CTCI sets the value of *l* to "true" if Dataway Inhibit is set in the crate specified by *ext* and sets the value of *l* to "false" if Dataway Inhibit is not set.

4.2.5 *Enable or Disable Crate Demand*

Name: CCCD.

Parameters: *ext* (external address, see Sub-clause 5.1);
l (logical truth value, see Sub-clause 5.9).

Function: CCCD causes Crate Demand to be enabled in the crate specified by *ext* if the value of *l* is "true" and causes Crate Demand to be disabled if the value of *l* is "false".

4.2.6 *Test Crate Demand Enabled*

Name: CTCD.

Parameters: *ext* (external address, see Sub-clause 5.1);
l (logical truth value, see Sub-clause 5.9).

Function: CTCD sets the value of *l* to "true" if Crate Demand is enabled in the crate specified by *ext* and sets the value of *l* to "false" if Crate Demand is disabled.

4.2.7 *Contrôle de la présence de la demande dans un châssis*

Nom: CTGL.

Paramètres: *ext* (adresse externe, voir paragraphe 5.1);
l (valeur logique de vérité, voir paragraphe 5.9).

Fonction: CTGL met la valeur de *l* à «vrai» si n'importe quelle demande est présente dans le châssis spécifié par *ext* et met la valeur de *l* à «faux» si aucune demande n'est présente.

4.2.8 *Déclaration de LAM*

Nom: CDLAM.

Paramètres: *lam* (identificateur de LAM, voir paragraphe 5.10);
b (numéro de branche, voir paragraphe 5.2);
c (numéro de châssis, voir paragraphe 5.3);
n (numéro de station, voir paragraphe 5.4);
m (identificateur de mode d'accès du LAM, voir paragraphe 5.11);
inta (tableau d'entiers, voir paragraphe 5.12).

Fonction: CDLAM code le numéro de branche *b*, le numéro de châssis *c*, le numéro de station *n* et toutes les autres informations nécessaires concernant un LAM sous forme d'un entier dépendant du système et stocke le résultat dans *lam*. Le paramétre *m* est interprété comme une sous-adresse si sa valeur est supérieure ou égale à zéro. Il est interprété comme la valeur négative d'une position de bit si sa valeur est inférieure à zéro. Cette information est codée dans la valeur assignée à *lam* et utilisée par les autres sous-programmes pour déterminer si le LAM est accessible par des fonctions spéciales d'accès au LAM ou par lecture et écriture des registres du groupe 2. L'information contenue dans le tableau *inta* dépend complètement de la mise en œuvre. Elle contient tout renseignement pouvant être requis par le système de l'ordinateur ou l'interface CAMAC pour permettre au programme d'accéder au LAM. L'utilisation de *inta* par CDLAM n'est pas nécessaire, mais *inta* doit quand même figurer dans la chaîne de paramètres.

4.2.9 *Mise en service ou hors service de LAM*

Nom: CCLM.

Paramètres: *lam* (identificateur de LAM, voir paragraphe 5.10);
l (valeur logique de vérité, voir paragraphe 5.9).

Fonction: CCLM entraîne la mise en service du LAM désigné par *lam* si la valeur de *l* est à «vrai» et sa mise hors service si la valeur de *l* est à «faux».

4.2.10 *Remise à zéro de LAM*

Nom: CCLC.

Paramètre: *lam* (identificateur de LAM, voir paragraphe 5.10).

Fonction: CCLC entraîne la remise à zéro du LAM désigné par *lam*.

4.2.11 *Contrôle de LAM*

Nom: CTLM.

Paramètres: *lam* (identificateur de LAM, voir paragraphe 5.10);
l (valeur logique de vérité, voir paragraphe 5.9).

4.2.7 *Test Crate Demand Present*

Name: CTGL.

Parameters: *ext* (external address, see Sub-clause 5.1);
l (logical truth value, see Sub-clause 5.9).

Function: CTGL sets the value of *l* to "true" if any demand is present in the crate specified by *ext* and sets the value of *l* to "false" if no demand is present.

4.2.8 *Declare LAM*

Name: CDLAM.

Parameters: *lam* (LAM identifier, see Sub-clause 5.10);
b (branch number, see Sub-clause 5.2);
c (crate number, see Sub-clause 5.3);
n (station number, see Sub-clause 5.4);
m (LAM access specifier, see Sub-clause 5.11);
inta (integer array, see Sub-clause 5.12).

Function: CDLAM encodes the branch number *b*, the crate number *c*, the station number *n* and all other necessary information concerning a LAM into a system-dependent, integer form and stores the result in *lam*. The parameter *m* is interpreted as a subaddress if its value is greater than or equal to zero. It is interpreted as the negative of a bit position if its value is less than zero. This information is encoded in the value assigned to *lam* and used by other subroutines to determine whether the LAM is accessed via special functions for LAM access or via reading and writing group 2 registers. The information contained in the array *inta* is completely implementation-dependent. It contains any information which may be required by the computer system or the CAMAC interface to enable the program to access the LAM. The use of *inta* by CDLAM is not required, but *inta* must appear in the parameter string regardless.

4.2.9 *Enable or Disable LAM*

Name: CCLM.

Parameters: *lam* (LAM identifier, see Sub-clause 5.10);
l (logical truth value, see Sub-clause 5.9).

Function: CCLM causes the LAM specified by *lam* to be enabled if the value of *l* is "true" and causes it to be disabled if the value of *l* is "false".

4.2.10 *Clear LAM*

Name: CCLC.

Parameter: *lam* (LAM identifier, see Sub-clause 5.10).

Function: CCLC causes the LAM specified by *lam* to be cleared.

4.2.11 *Test LAM*

Name: CTLM.

Parameters: *lam* (LAM identifier, see Sub-clause 5.10);
l (logical truth value, see Sub-clause 5.9).

Fonction: CTLM met *l* à la valeur «vrai» si le LAM désigné par *lam* est présent et met *l* à la valeur «faux» si le LAM est absent.

4.2.12 *Liaison du LAM à la procédure de service*

Nom: CCLNK.

Paramètres: *lam* (identificateur de LAM, voir paragraphe 5.10);
label (identificateur du point d'entrée, voir paragraphe 5.13).

Fonction: CCLNK associe le LAM désigné par *lam* à une procédure identifiée par le paramètre *label.* Le résultat de cette association est que la procédure sera exécutée toutes les fois que le LAM sera reconnu par le système.

4.3 *Transferts de bloc, actions multiples et déclarations inverses*

Ces sous-programmes, avec ceux qui sont décrits aux paragraphes 4.1 et 4.2, forment un système complet d'aide à la programmation CAMAC, au niveau C, avec des caractéristiques très générales et des possibilités d'exécutions efficaces des transferts de bloc et actions multiples (voir Publication 677 de la CEI: Transferts de bloc dans les systèmes CAMAC).

Tous les sous-programmes d'action de ce paragraphe emploient un «bloc de contrôle», *cb*, qui est un tableau d'entiers contenant les quatre éléments suivants:

élément 1: nombre d'opérations à répéter;
élément 2: pointage;
élément 3: identification de LAM;
élément 4: identification de canal.

Le nombre d'opérations à répéter spécifie le nombre d'actions CAMAC ou le nombre maximal de mots de données à transférer. Le pointage est renvoyé par le sous-programme et indique le nombre d'actions réellement exécutées ou le nombre de mots de données réellement transférés. Ainsi, le programme appelant peut détecter et analyser la fin prématurée d'une activité. L'identification de LAM est un identificateur codé avec la même forme et la même signification que celles qui sont renvoyées par le sous-programme CDLAM. L'identification de canal est un paramètre dépendant du système ou de la mise en œuvre et peut ne pas être exigée dans toutes les mises en œuvre. Elle est utilisée pour identifier tous les éléments relatifs à l'ordinateur nécessaires pour l'exécution de l'action CAMAC spécifiée. On peut citer, par exemple, les canaux d'entrée-sortie de l'ordinateur et d'autres éléments du système d'exploitation tels que numéros d'unités fonctionnelles.

4.3.1 *Action multiple générale*

Nom: CFGA.

Paramètres: *fa* (codes de fonction, voir paragraphe 5.14);
exta (adresses externes, voir paragraphe 5.15);
intc (tableau de données CAMAC, voir paragraphe 5.16);
qa (réponses Q, voir paragraphe 5.17);
cb (bloc de contrôle, voir paragraphe 5.18).

Fonction: CFGA provoque l'exécution d'une séquence de fonctions CAMAC déterminées par les éléments successifs de *fa* selon la séquence correspondante des adresses CAMAC déterminées par les éléments successifs de *exta.*

Toute fonction de lecture ou d'écriture dans *fa* provoque le transfert d'un mot de données CAMAC entre l'élément correspondant de *intc* et le registre CAMAC spécifié. La réponse Q pour chaque action CAMAC individuelle est stockée dans l'élément correspondant de *qa.*

Function: CTLM sets *l* to the value "true" if the LAM specified by *lam* is asserted, and sets *l* to the value "false" if the LAM is not asserted.

4.2.12 *Link LAM to Service Procedure*

Name: CCLNK.

Parameters: *lam* (LAM identifier, see Sub-clause 5.10);
label (entry point identifier, see Sub-clause 5.13).

Function: CCLNK performs an association between the LAM specified by *lam* and a procedure identified by the parameter *label.* As a result of this association the procedure will be executed whenever the LAM is recognized by the system.

4.3 *Block Transfers, Multiple Actions and Inverse Declarations*

These subroutines, together with those described in Sub-clauses 4.1 and 4.2, form a comprehensive CAMAC support system, level C, with very general features and a potential for efficient execution of block transfers and multiple actions (see IEC Publication 677: Block Transfers in CAMAC Systems).

All the action subroutines in this sub-clause employ a "control block", *cb*, which is an integer array containing four elements as follows:

element 1: repeat count;
element 2: tally;
element 3: LAM identification;
element 4: channel identification.

The repeat count specifies the number of CAMAC actions or the maximum number of data words to be transferred. The tally is returned by the subroutine and indicates the number of actions actually executed or the number of data words actually transferred. Thus, the calling program can detect and analyze a premature termination of an activity. The LAM identification is a coded identifier of the same form and interpretation as that returned by the subroutine CDLAM. The channel identification is a system-dependent or implementation-dependent parameter which may not be required in all implementations. It is used to identify any computer-related facilities necessary for the execution of the specified CAMAC action. Examples of facilities which might be required to be identified include computer input/-output channels and operating system facilities such as unit numbers.

4.3.1 *General Multiple Action*

Name CFGA.

Parameters: *fa* (function codes, see Sub-clause 5.14);
exta (external addresses, see Sub-clause 5.15);
intc (CAMAC data array, see Sub-clause 5.16);
qa (Q responses, see Sub-clause 5.17);
cb (control block, see Sub-clause 5.18).

Function: CFGA causes a sequence of CAMAC functions specified in successive elements of *fa* to be performed at a corresponding sequence of CAMAC addresses specified in successive elements of *exta.*

Any read or write function in *fa* causes a CAMAC data word to be transferred between the corresponding element of *intc* and the specified CAMAC register. The Q response for each individual CAMAC action is stored in the corresponding element of *qa.* The number of

Le nombre des actions exécutées et des éléments requis dans les tableaux *fa, exta, intc* et *qa* est donné par le premier élément de *cb*. Si le troisième élément de *cb* contient la valeur zéro, la séquence déterminée d'actions est immédiatement exécutée; s'il contient une identification de LAM, la séquence d'actions n'est pas commencée tant que le LAM n'est pas reconnu.

4.3.2 *Scrutation d'adresses*

Nom: CFMAD.

Paramètres: *f* (code de fonction, voir paragraphe 5.6);
extb (adresses externes, voir paragraphe 5.19);
intc (tableau de données CAMAC, voir paragraphe 5.16);
cb (bloc de contrôle, voir paragraphe 5.18).

Fonction: CFMAD provoque l'exécution d'une fonction CAMAC simple déterminée par la valeur de *f* selon une séquence d'adresses calculées au moyen de l'algorithme de scrutation d'adresses décrit dans la Publication 516 de la CEI, paragraphe 5.4.3.1, et la Publication 677 de la CEI, paragraphe 3.2. Dans le mode de scrutation d'adresses, la fonction spécifiée est d'abord exécutée à l'adresse indiquée par le premier élément de *extb*. Ensuite, si la réponse Q est l, la sous-adresse est incrémentée de 1 ainsi que l'index dans le tableau des données *intc*, et la fonction est exécutée à cette nouvelle sous-adresse. Lorsque la sous-adresse dépasse 15, elle est ramenée à zéro et le numéro de station est incrémenté de 1.

Si la réponse Q est 0, la sous-adresse est mise à zéro, et le numéro de station est incrémenté, mais l'index dans *intc* n'est pas modifié. L'exécution de la fonction spécifiée est tentée à la nouvelle adresse résultante et le processus est répété jusqu'à ce que soit le nombre d'actions requises (données par le premier élément de *cb*) ait été effectué, soit l'adresse calculée selon la procédure décrite ci-dessus dépasse l'adresse contenue dans le second élément de *extb*. Si le troisième élément de *cb* contient la valeur 0, l'exécution des actions CAMAC est immédiatement entreprise. S'il contient un identificateur de LAM, l'exécution n'est pas entreprise tant que le LAM spécifié n'a pas été reconnu.

La Publication 516 de la CEI définit seulement une scrutation d'adresses dans un châssis unique. Pour l'usage de cette norme, la scrutation d'adresses a été étendue aux autres composantes d'adresse, comme suit: si le numéro de station dépasse le nombre maximal de stations dans un châssis, on lui affecte la valeur 1 et le numéro de châssis est incrémenté. Si le numéro de châssis dépasse le nombre maximal de châssis dans la branche, on lui affecte la valeur 1 et le numéro de branche est incrémenté.

4.3.3 *Transfert de bloc synchronisé par le contrôleur*

Nom: CFUBC.

Paramètres: *f* (code de fonction, voir paragraphe 5.6);
ext (adresse externe, voir paragraphe 5.1);
intc (tableau de données CAMAC, voir paragraphe 5.16);
cb (bloc de contrôle, voir paragraphe 5.18).

Fonction: CFUBC provoque l'exécution de la fonction CAMAC simple spécifiée par la valeur de *f*, à l'adresse CAMAC déterminée par la valeur *ext*. Dans ce mode, l'adresse CAMAC n'est jamais changée, mais on attend du registre unique qu'il fournisse ou accepte de nombreux mots de données. Il est supposé apte à fournir ou recevoir un mot de données toutes les fois que le contrôleur l'adresse jusqu'à ce que le bloc soit terminé ou que le contrôleur mette fin au processus du fait que le nombre de transferts de données dépasse la limite fixée par le premier élément

actions performed and the number of elements required in the arrays *fa*, *exta*, *intc*, and *qa* is given by the value contained in the first element of *cb*. If the third element of *cb* contains the value zero, the specified sequence of actions is executed immediately; if it contains a LAM identification, then the sequence of actions is not initiated until the LAM is recognized.

4.3.2 *Address Scan*

Name: CFMAD.

Parameters: *f* (function code, see Sub-clause 5.6);
extb (external addresses, see Sub-clause 5.19);
intc (CAMAC data array, see Sub-clause 5.16);
cb (control block, see Sub-clause 5.18).

Function: CFMAD causes a single CAMAC function specified by the value of *f* to be executed at a succession of addresses computed using the Address Scan algorithm described in IEC Publication 516, Sub-clause 5.4.3.1, and IEC Publication 677, Sub-clause 3.2. In the Address Scan mode the specified function is executed first at the address given by the first element of *extb*. Then, if the Q response is 1, the subaddress is incremented by 1 and the index into the data array *intc* incremented by 1, and the function is executed at this new subaddress. If the subaddress is incremented beyond 15, it is set to zero, and the station number is incremented.

If the Q response is 0, the subaddress is set to zero and the station number is incremented, but the index into *intc* is not changed. Execution of the specified function is attempted at the resulting new address, and the process is repeated until either the requested number of actions (given in the first element of *cb*) has been performed or the address computed by the procedure described above exceeds the address contained in the second element of *extb*. If the third element of *cb* contains the value 0, execution of the CAMAC actions is begun immediately. If it contains a LAM identifier, execution does not begin until the specified LAM is recognized.

IEC Publication 516 defines Address Scan within a single crate only. For the purpose of this standard, Address Scan is extended to other address components as follows. If the station number is incremented beyond the number of stations in a crate, it is set to 1, and the crate number is incremented. If the crate number is incremented beyond the number of crates in the branch, the crate number is set to 1, and the branch number is incremented.

4.3.3 *Controller-Synchronized Block Transfer*

Name: CFUBC.

Parameters: *f* (function code, see Sub-clause 5.6);
ext (external address, see Sub-clause 5.1);
intc (CAMAC data array, see Sub-clause 5.16);
cb (control block, see Sub-clause 5.18).

Function: CFUBC causes the single CAMAC function given by the value of *f* to be executed at the CAMAC address specified by the value of *ext*. In this mode, the CAMAC address is never changed, but the single register is expected to supply or accept many words of data. It is assumed able to supply or accept a data word whenever the controller addresses it until the block is exhausted or the controller terminates the process because the number of data transfers exceeds the limit given by the first element of *cb*. If the third element of *cb* contains the value 0,

de *cb*. Si le troisième élément de *cb* contient la valeur 0, l'exécution des actions CAMAC est immédiatement entreprise. S'il contient un identificateur de LAM, l'exécution n'est pas entreprise tant que le LAM spécifié n'a pas été reconnu.

Le module indique que le bloc est terminé par sa réponse Q. Deux méthodes sont décrites à cet effet dans la Publication 677 de la CEI: le mode arrêt (paragraphe 3.1) et le mode arrêt sur un mot (paragraphe 4.1). En général, les deux méthodes ne sont pas mises en œuvre simultanément dans un système matériel donné et aucune confusion ne peut survenir. Cependant, si les deux méthodes sont admissibles, le quatrième élément de *cb*, qui contient des informations dépendantes du système, doit être utilisé pour spécifier dans chaque cas quelle est la méthode utilisée par le module.

4.3.4 *Transfert de bloc synchronisé par LAM*

Nom: CFUBL.

Paramètres: *f* (code de fonction, voir paragraphe 5.6);
ext (adresse externe, voir paragraphe 5.1);
intc (tableau de données CAMAC, voir paragraphe 5.16);
cb (bloc de contrôle, voir paragraphe 5.18).

Fonction: CFUBL provoque l'exécution d'une fonction CAMAC unique spécifiée par le contenu de *f* à l'adresse CAMAC spécifiée par le contenu de *ext* avec utilisation de la réponse Q et du signal de LAM tel que défini dans la Publication 677 de la CEI, paragraphe 4.2, pour le mode ULS. Dans le mode ULS, l'adresse CAMAC n'est pas changée, mais on attend de l'adresse unique qu'elle fournisse ou accepte de nombreux mots de données. Dans le mode ULS, le module établit un LAM toutes les fois qu'il est prêt à participer à un transfert de données, et le contrôleur répond avec la commande appropriée pour effectuer le transfert. N'importe quels mots de données transférés sont ou placés dans le tableau *intc*, ou extraits de ce tableau. Le contrôleur achève le processus si le nombre des exécutions dépasse la limite donnée par le contenu du premier élément de *cb* ou bien le module peut achever le processus en répondant par Q = 0. La réponse Q = 1 indique que la fonction spécifiée a été convenablement exécutée. La réponse Q = 0 indique que le transfert de données tenté n'a pas été accompli et que le processus devrait être achevé. Le LAM qui synchronise le processus est spécifié par le contenu du troisième élément de *cb*. A la fin, le nombre des réponse Q = 1 est stocké dans le second élément de *cb*.

4.3.5 *Transfert de bloc en mode répétitif*

Nom: CFUBR.

Paramètres: *f* (code de fonction, voir paragraphe 5.6);
ext (adresse externe, voir paragraphe 5.1);
intc (tableau de données CAMAC, voir paragraphe 5.16);
cb (bloc de contrôle, voir paragraphe 5.18).

Fonction: CFUBR provoque l'exécution d'une fonction CAMAC unique spécifiée par le contenu de *f*, à l'adresse CAMAC spécifiée par le contenu de *ext*, avec l'utilisation de la réponse Q telle que définie dans la Publication 516 de la CEI, paragraphe 5.4.3.2, et la Publication 677 de la CEI, paragraphe 3.3, pour le mode répétitif. Dans le mode répétitif, l'adresse CAMAC n'est jamais changée, mais on attend de l'adresse unique qu'elle accepte ou fournisse de nombreux mots de données. Q est utilisé comme signal de synchronisation. Q = 1 indique que la dernière fonction a été exécutée avec succès; Q = 0 indique que le module n'était

execution of the CAMAC actions is begun immediately. If it contains a LAM identifier, execution does not begin until the specified LAM is recognized.

The module indicates that the block is exhausted by its Q response. Two methods for doing this are described in IEC Publication 677, i.e. Stop Mode (Sub-clause 3.1) and Stop-on-Word Mode (Sub-clause 4.1). In general, both methods are not implemented in a given hardware system and no confusion should occur. However, if both methods are permissible, then the fourth element of *cb* which contains system dependent facilities must be used to specify for each case which method the module is using.

4.3.4 *LAM-Synchronized Block Transfer*

Name: CFUBL.

Parameters: *f* (function code, see Sub-clause 5.6);
ext (external address, see Sub-clause 5.1);
intc (CAMAC data array, see Sub-clause 5.16);
cb (control block, see Sub-clause 5.18).

Function: CFUBL causes the single CAMAC function specified by the contents of *f* to be executed at the CAMAC address specified by the contents of *ext*, with the usage of the Q response and the LAM signal defined as in IEC Publication 677, Sub-clause 4.2, for the ULS mode. In the ULS mode, the CAMAC address is not changed, but the single address is expected to supply or accept many words of data. In the ULS mode, the module asserts a LAM whenever it is ready to participate in a data transfer, and the controller responds with the appropriate command to effect the transfer. Any data words transferred are placed into or taken from the array *intc*. The controller terminates the process if the number of executions exceeds the limit given by the contents of the first element of *cb*, or the module may terminate the process by responding with Q = 0. The response Q = 1 indicates that the specified function was properly executed. The response Q = 0 indicates that the attempted data transfer was not completed and that the process should be terminated. The LAM which synchronizes the process is specified by the contents of the third element of *cb*. Upon termination the number of Q = 1 responses is stored in the second element of *cb*.

4.3.5 *Repeat Mode Block Transfer*

Name: CFUBR.

Parameters: *f* (function code, see Sub-clause 5.6);
ext (external address, see Sub-clause 5.1);
intc (CAMAC data array, see Sub-clause 5.16);
cb (control block, see Sub-clause 5.18).

Function: CFUBR causes the single CAMAC function specified by the contents of *f* to be executed at the CAMAC address specified by the contents of *ext* with the usage of the Q response defined as in IEC Publication 516, Sub-clause 5.4.3.2, and IEC Publication 677, Sub-clause 3.3, for the Repeat mode. In the Repeat mode, the CAMAC address is never changed, but the single address is expected to supply or accept many words of data. Q is used as a timing signal. Q = 1 indicates that the previously executed function succeeded; Q = 0 indicates that the module was not ready to execute the function and that the controller should try

pas prêt à exécuter la fonction et que le contrôleur devrait essayer de nouveau. Tous les mots de données transférés sont placés dans le tableau *intc* ou en sont extraits. Si la réponse est Q = 0, le transfert n'a pas lieu et l'index dans le tableau *intc* n'est pas modifié. Le nombre de réponses Q = 1 attendues est donné par le contenu du premier élément de *cb*. Si le troisième élément de *cb* contient zéro, le processus est immédiatement entrepris; s'il contient un identificateur de LAM, le processus n'est commencé que lorsque le LAM spécifié a été reconnu.

4.3.6 *Analyse de l'identificateur de LAM*

Nom: CGLAM.

Paramètres: *lam* (identificateur de LAM, voir paragraphe 5.10);
b (numéro de branche, voir paragraphe 5.2);
c (numéro de châssis, voir paragraphe 5.3);
n (numéro de station, voir paragraphe 5.4);
m (identificateur du mode d'accès au LAM, voir paragraphe 5.11);
inta (tableau d'entiers, voir paragraphe 5.12).

Fonction: CGLAM décompose l'identificateur *lam* de LAM, en ses constituants qui sont le numéro de branche *b*, le numéro de châssis *c*, le numéro de station *n*, la sous-adresse ou position du bit *m* et l'information dépendante du système *inta*. Ce sous-programme effectue le processus inverse réalisé par CDLAM et tous les paramètres ont la même interprétation et la même forme.

4.3.7 *Analyse de l'identificateur de registre*

Nom: CGREG.

Paramètres: *ext* (adresse externe, voir paragraphe 5.1);
b (numéro de branche, voir paragraphe 5.2);
c (numéro de châssis, voir paragraphe 5.3);
n (numéro de station, voir paragraphe 5.4);
a (sous-adresse, voir paragraphe 5.5).

Fonction: CGREG décompose l'identificateur d'adresse CAMAC *ext* en ses constituants qui sont le numéro de branche *b*, le numéro de châssis *c*, le numéro de station *n* et la sous-adresse *a*. Ce sous-programme effectue le processus inverse réalisé par CDREG et tous les paramètres ont la même interprétation et la même forme.

5. Définitions de paramètres

Les significations des paramètres utilisés dans les définitions des sous-programmes sont brièvement données dans leurs descriptions fonctionnelles. Du fait que des paramètres identiques sont utilisés dans de nombreux sous-programmes, les divers symboles sont définis et étudiés plus complètement dans les paragraphes qui suivent. Ces renseignements sont particulièrement destinés à guider les réalisateurs en espérant qu'il en résultera la plus grande uniformité possible entre les différentes réalisations.

5.1 *ext (adresse externe)*

Le symbole *ext* représente un entier qui est utilisé comme un identificateur d'une adresse externe CAMAC. L'adresse peut représenter un registre dans lequel on peut lire ou écrire, une adresse complète CAMAC à laquelle on peut avoir accès par des fonctions de commande ou de contrôle, ou une adresse de châssis. La valeur de *ext* est explicitement définie comme étant un entier. Normalement, elle peut être une version codée des composantes de l'adresse pour

again. Any data words transferred are placed into or taken from the array *intc*. If the response is Q = 0, no transfer took place, and the index into the array *intc* is not changed. The number of Q = 1 responses expected is given by the contents of the first element of *cb*. If the third element of *cb* contains zero, the process is initiated immediately; if it contains a LAM identifier, the process is initiated only when the specified LAM is recognized.

4.3.6 *Analyze LAM Identifier*

Name: CGLAM.

Parameters: *lam* (LAM identifier, see Sub-clause 5.10);
b (branch number, see Sub-clause 5.2);
c (crate number, see Sub-clause 5.3);
n (station number, see Sub-clause 5.4);
m (LAM access identifier, see Sub-clause 5.11);
inta (integer array, see Sub-clause 5.12).

Function: CGLAM decodes the LAM identifier *lam* into its component parts, consisting of the branch number *b*, the crate number *c*, the station number *n*, the subaddress or bit-position *m*, and the system-dependent information *inta*. This subroutine exactly reverses the process performed by CDLAM, and all parameters have the same interpretation and form.

4.3.7 *Analyze Register Identifier*

Name: CGREG.

Parameters: *ext* (external address, see Sub-clause 5.1);
b (branch number, see Sub-clause 5.2);
c (crate number, see Sub-clause 5.3);
n (station number, see Sub-clause 5.4);
a (subaddress, see Sub-clause 5.5).

Function: CGREG decodes the CAMAC address identifier *ext* into its component parts, consisting of the branch number *b*, the crate number *c*, the station number *n*, and the subaddress *a*. This subroutine exactly reverses the process performed by CDREG, and all parameters have the same interpretation and form.

5. Definitions of Parameters

The meanings of the parameters used in the subroutine definitions are given briefly in the subroutine function descriptions. Since similar parameters are used in many subroutines, the various symbols are defined and discussed more completely in the sub-clauses which follow. This information is intended particularly as a guide to implementors in the hope that it will result in the greatest possible degree of uniformity among different implementations.

5.1 *ext (external address)*

The symbol *ext* represents an integer which is used as an identifier of an external CAMAC address. The address may represent a register which can be read or written, a complete CAMAC address which can be accessed by control or test functions, or a crate address. The value of *ext* is explicitly defined to be an integer. Normally, it can be expected to be an encoded version of the address components, in which the coding has been selected for the

laquelle le codage a été choisi en vue de l'exécution la plus efficace des actions CAMAC par l'interface où l'on applique la mise en œuvre. D'autres possibilités sont admises cependant. Par exemple *ext* peut être un index ou un pointeur dans une structure de données dans laquelle les composantes de l'adresse CAMAC en cause sont stockées.

5.2 *b (numéro de branche)*

Le symbole *b* représente un entier qui est la composante «numéro de branche» d'une adresse CAMAC. Il peut représenter le numéro physique de la branche dans des systèmes multibranches ou des ensembles de châssis regroupés pour des raisons fonctionnelles ou autres. Dans certains systèmes, il peut être ignoré, bien qu'il doive être inclus dans la liste des paramètres, ne serait-ce que pour la compatibilité.

5.3 *c (numéro de châssis)*

Le symbole *c* représente un entier qui est la composante «numéro de châssis» d'une adresse CAMAC. Le «numéro de châssis», dans ce contexte, peut être soit le numéro physique du châssis, soit le symbole d'un entier qui est interprété par le logiciel du système de l'ordinateur pour élaborer l'information appropriée d'accès au matériel.

5.4 *n (numéro de station)*

Le symbole *n* représente un entier qui est la composante «numéro de station» d'une adresse CAMAC.

5.5 *a (sous-adresse)*

Le symbole *a* représente un entier qui est la composante «sous-adresse» d'une adresse CAMAC.

5.6 *f (code de fonction)*

Le symbole *f* représente un entier qui est le code de fonction pour une action CAMAC.

5.7 *int (mot de données CAMAC)*

Le symbole *int* représente un mot de données CAMAC stocké dans la mémoire de l'ordinateur. La forme n'est pas spécifiée mais le mot doit être stocké dans une entité adressable capable de contenir 24 bits. Dans un ordinateur ou un système programmable qui ne possède pas une unité de stockage adressable susceptible de contenir 24 bits, on doit utiliser plusieurs de ces unités.

5.8 *q (réponse Q)*

Le symbole *q* représente une valeur logique de vérité qui correspond à la réponse Q CAMAC. Elle est mise à «vrai» si la réponse Q est 1, à «faux» si la réponse Q est 0.

5.9 *l (valeur logique de vérité)*

Le symbole *l* représente une valeur logique de vérité qui peut être soit à «vrai», soit à «faux».

5.10 *lam (identificateur de LAM)*

Le symbole *lam* représente un entier utilisé comme identificateur de signal LAM CAMAC. L'information associée à l'identificateur ne doit pas seulement inclure l'adresse CAMAC, mais aussi l'information sur les moyens d'accès et de contrôle du LAM, c'est-à-dire si on y

most efficient execution of CAMAC actions on the interface to which the implementation applies. Other possibilities are allowed, however. For example *ext* may be an index or a pointer into a data structure in which the actual CAMAC address components are stored.

5.2 *b (branch number)*

The symbol *b* represents an integer which is the branch number component of a CAMAC address. It may represent a physical highway number in multiple highway systems, or it may represent sets of crates grouped together for functional or other reasons. In some systems it may be ignored, although it must be included in the parameter list for the sake of compatibility.

5.3 *c (crate number)*

The symbol *c* represents an integer which is the crate number component of a CAMAC address. "Crate number" in this context can be either the physical crate number or it can be an integer symbol which is interpreted by the computer system software to produce appropriate hardware access information.

5.4 *n (station number)*

The symbol *n* represents an integer which is the station number component of a CAMAC address.

5.5 *a (subaddress)*

The symbol *a* represents an integer which is the subaddress component of a CAMAC address.

5.6 *f (function code)*

The symbol *f* represents an integer which is the function code for a CAMAC action.

5.7 *int (CAMAC data word)*

The symbol *int* represents a CAMAC data word stored in computer memory. The form is not specified, but the word must be stored in an addressable storage entity capable of containing 24 bits. In a computer or programming system which does not have an addressable unit of storage which can contain 24 bits, multiple units must be used.

5.8 *q (Q response)*

The symbol *q* represents a logical truth value which corresponds to the CAMAC Q response. It is set to "true" if the Q response is 1, to "false" if the Q response is 0.

5.9 *l (logical truth value)*

The symbol *l* represents a logical truth value which can be either "true" or "false".

5.10 *lam (LAM identifier)*

The symbol *lam* represents an integer which is used as the identifier of a CAMAC LAM signal. The information associated with the identifier must include not only the CAMAC address but also information about the means of accessing and controlling the LAM, i.e.

accède par des fonctions sans données à une sous-adresse, ou par des fonctions lecture/écriture du groupe 2.

La valeur du *lam* est explicitement définie comme étant un entier non nul; selon la réalisation, c'est soit une représentation codée de l'information requise pour décrire le LAM, soit une clé simplement fournie pour accéder à l'information dans une structure de données-système.

La valeur 0 est utilisée pour indiquer, là où c'est convenable, qu'aucun LAM n'a été spécifié.

5.11 *m (identificateur de mode d'accès de LAM)*

Le symbole *m* représente un entier qui est utilisé pour indiquer le mode d'accès à un LAM et la composante de l'adresse d'ordre le plus bas pour accéder au LAM.

Si *m* est nul ou positif, on l'interprète comme une sous-adresse et le LAM peut être accessible par des fonctions sans données à cette sous-adresse. Si *m* est négatif, il est interprété comme la valeur négative d'une position de bit pour un LAM accessible par des fonctions de lecture, d'établissement ou de suppression de bits dans les registres du groupe 2 aux sous-adresses 12, 13 ou 14.

5.12 *inta (tableau d'entiers)*

Le symbole *inta* représente un tableau d'entiers dont ni la longueur ni le contenu ne sont définis ici. Il est supposé contenir une information dépendante du système ou de la mise en œuvre et associée à la définition d'un LAM. Si une telle information n'est pas nécessaire on n'a pas besoin d'utiliser le tableau. Cette information peut inclure des paramètres nécessaires à la liaison avec les interruptions, la spécification des événements etc. La documentation pour une mise en œuvre doit décrire les caractéristiques de tout paramètre contenu dans ce tableau.

5.13 *label (identificateur du point d'entrée)*

Le symbole *label* représente un point d'entrée dans une procédure programmée. Une telle procédure sera habituellement exécutée en réponse à la reconnaissance d'un LAM; elle peut interrompre le processus en cours d'exécution au moment même de la reconnaissance du LAM. Dans ces circonstances, la procédure doit être capable de sauvegarder et de restituer l'état de l'ordinateur de façon telle que le processus interrompu puisse être repris. Au moins une valeur des labels devrait identifier une procédure d'erreur-système pour traiter les LAM non liés aux processus de l'utilisateur.

5.14 *fa (codes de fonction)*

Le symbole *fa* représente un tableau d'entiers, chacun d'entre eux étant un code de fonction pour une action CAMAC. La longueur de *fa* est donnée par la valeur du premier élément de *cb* (voir paragraphe 5.18) au moment ou le sous-programme est exécuté.

5.15 *exta (adresse externe CAMAC)*

Le symbole *exta* représente un tableau d'entiers, chacun d'entre eux étant l'adresse d'un registre CAMAC. La forme et le contenu de l'information de chaque élement de *exta* doivent être identiques à la forme et au contenu de l'information de la quantité *ext* (voir paragraphe 5.1). La longueur de *exta* est donnée par la valeur du premier élément de *cb* (voir paragraphe 5.18) au moment où le sous-programme est exécuté.

whether it is accessed via dataless functions at a subaddress or via read/write functions in group 2 registers.

The value of *lam* is explicitly defined to be a non-zero integer; whether it is an encoded representation of the information required to describe the LAM, or simply provides a key for accessing the information in a system data structure, is an implementation decision.

The value 0 is used to indicate where appropriate, that no LAM is being specified.

5.11 *m (LAM access specifier)*

The symbol *m* represents an integer which is used to indicate the mode of access of a LAM and the lowest-order address component for LAM addressing.

If *m* is zero or positive, it is interpreted as a subaddress and the LAM is assumed to be accessed via dataless functions at this subaddress. If *m* is negative, it is interpreted as the negative of a bit position for a LAM which is accessed via reading, setting, or clearing bits in the group 2 registers at subaddresses 12, 13 or 14.

5.12 *inta (integer array)*

The symbol *inta* represents an integer array, the length and contents of which are not defined in this document. It is intended to contain system-dependent or implementation-dependent information associated with the definition of a LAM. If no such information is required, the array need not be used. This information can include parameters necessary for interrupt linkage, event specification, etc. The documentation for an implementation must describe the requirements for any parameters contained in this array.

5.13 *label (entry point identifier)*

The symbol *label* represents an entry point into a programmed procedure. Such a procedure will typically be executed in response to the recognition of a LAM, and it may interrupt the process being executed at the time of recognition of the LAM. Under these circumstances the procedure must be capable of saving and restoring the state of the computer so that the interrupted process can be resumed. At least one value of labels should identify a system error procedure which deals with LAM's not linked to user processes.

5.14 *fa (function codes)*

The symbol *fa* represents an array of integers, each of which is the function code for a CAMAC action. The length of *fa* is given by the value of the first element of *cb* (see Sub-clause 5.18) at the time the subroutine is executed.

5.15 *exta (CAMAC external address)*

The symbol *exta* represents an array of integers, each of which is a CAMAC register address. The form and information content of each element of *exta* must be identical to the form and information content of the quantity *ext* (see Sub-clause 5.1). The length of *exta* is given by the value of the first element of *cb* (see Sub-clause 5.18) at the time the subroutine is executed.

5.16 *intc (tableau de données CAMAC)*

Le symbole *intc* représente un tableau de mots de données CAMAC. Chaque élément de *intc* a la même forme que la variable du mot de données CAMAC *int* (voir paragraphe 5.7). La longueur de *intc* est donnée par la valeur du premier élément de *cb* (voir paragraphe 5.18) au moment où le sous-programme est exécuté.

5.17 *qa (réponses Q)*

Le symbole *qa* représente un tableau de valeurs de réponses Q. Chaque élément de *qa* a la même forme et peut avoir les mêmes valeurs que le paramètre *q* (voir paragraphe 5.8). La longueur de *qa* est donnée par la valeur du premier élément de *cb* (voir paragraphe 5.18) au moment où le sous-programme est exécuté.

5.18 *cb (bloc de contrôle)*

Le symbole *cb* représente un tableau d'entiers à quatre éléments. Le contenu de ces éléments est:

élément 1: nombre d'opérations à répéter;
élément 2: pointage;
élément 3: identification de LAM;
élément 4: identification de canal.

Le nombre d'opérations à répéter spécifie le nombre d'actions individuelles CAMAC ou le nombre maximal de mots de données à transférer. Quelques sous-programmes d'action multiple et de transfert de bloc permettent la fin de la séquence sur un signal provenant d'un module désigné. Dans de tels cas, le nombre d'opérations à répéter représente une limite supérieure. Le pointage est le nombre d'actions réellement exécutées ou le nombre de mots de données CAMAC réellement transférés. Si le transfert de bloc ou l'action multiple est terminée par le contrôleur par suite de l'épuisement du nombre d'opérations à répéter, le pointage sera égal à ce nombre; autrement, il peut lui être inférieur. L'identification de LAM est une valeur entière ayant les mêmes forme et contenu d'information que la variable *lam* (voir paragraphe 5.10). L'identification de canal est une valeur entière qui identifie les éléments dépendants du système nécessaires à l'exécution d'un transfert de bloc ou d'une action multiple. Ce nombre, si on en a besoin, a les mêmes forme et contenu que le paramètre *chan* (voir annexe A, paragraphe A6.2) et peut être élaboré par le sous-programme CDCHN (voir annexe A, paragraphe A3.1).

5.19 *extb (adresses externes)*

Le symbole *extb* représente un tableau d'entiers contenant des adresses CAMAC externes. Le tableau a deux éléments; le premier contient l'adresse initiale d'une action multiple en mode scrutation d'adresses, le second contient la dernière adresse autorisée à prendre part à la scrutation d'adresses. Chaque élément a les mêmes forme et contenu d'information que le paramètre *ext* (voir paragraphe 5.1).

5.16 *intc (CAMAC data array)*

The symbol *intc* represents an array of CAMAC data words. Each element of *intc* has the same form as the CAMAC data word variable *int* (see Sub-clause 5.7). The length of *intc* is given by the value of the first element of *cb* (see Sub-clause 5.18) at the time the subroutine is executed.

5.17 *qa (Q responses)*

The symbol *qa* represents an array of Q response values. Each element of *qa* has the same form and can have the same values as the parameter *q* (see Sub-clause 5.8). The length of *qa* is given by the value of the first element of *cb* (see Sub-clause 5.18) at the time the subroutine is executed.

5.18 *cb (control block)*

The symbol *cb* represents an integer array having four elements. The contents of these elements are:

element 1: repeat count;
element 2: tally;
element 3: LAM identification;
element 4: channel identification.

The repeat count specifies the number of individual CAMAC actions or the maximum number of data words to be transferred. Some multiple action and block transfer subroutines permit termination of the sequence upon a signal from the addressed module. In such cases, the repeat count represents an upper limit. The tally is the number of actions actually performed or the number of CAMAC data words actually transferred. If the block transfer or multiple action is terminated by the controller due to exhaustion of the repeat count, the tally will be equal to the repeat count; otherwise it may be less. The LAM identification is an integer value having the same form and information content as the variable *lam* (see Sub-clause 5.10). The channel identification is an integer value which identifies system-dependent facilities which may be necessary to perform the block transfer or multiple action. This number, if it is required, has the same form and content as the parameter *chan* (see Appendix A, Sub-clause A6.2) and can be created by the subroutine CDCHN (see Appendix A, Sub-clause A3.1).

5.19 *extb (external addresses)*

The symbol *extb* represents an array of integers containing external CAMAC addresses. The array has two elements; the first contains the starting address for an Address Scan multiple action, the second contains the final address which can be permitted to participate in the Address Scan sequence. Each element has the same form and information content as the parameter *ext* (see Sub-clause 5.1).

ANNEXE A

SOUS-PROGRAMMES DÉPENDANTS DU SYSTÈME

A1. Introduction

Il est souvent utile, même nécessaire, d'utiliser des sous-programmes qui dépendent étroitement de la structure d'un ordinateur particulier ou du système d'exploitation. Ces sous-programmes ne peuvent être intégralement transposables même lors de la mise en œuvre d'un langage hautement normalisé. Cependant, les fonctions qu'ils peuvent accomplir sont habituellement très similaires et peuvent être identiques dans des catégories restreintes de systèmes. Cette annexe décrit plusieurs de ces sous-programmes dépendants du système. Ils ne sont pas nécessaires à tout niveau de la réalisation, mais n'importe lequel d'entre eux peut être introduit, au gré du réalisateur. La documentation relative à une mise en œuvre déterminée doit indiquer quels sont ceux qui sont fournis et décrire complètement leurs caractéristiques et paramètres dépendants du système.

A2. Accès aux signaux spéciaux

Ces sous-programmes constituent une ouverture à des dispositions de systèmes CAMAC pour lesquelles une méthode normalisée d'accès n'est pas définie ou qui ne sont pas disponibles dans tous les systèmes.

A2.1 *Initialisation de branche*

Nom: CCINIT.

Paramètre: *b* (numéro de branche, voir paragraphe 5.2).

Fonction: CCINIT provoque l'établissement du signal d'initialisation de branche (BZ) sur la branche identifiée par le paramètre *b*. Si *b* identifie un groupe de châssis non couplés à une branche parallèle, la fonction peut être simulée de sorte que le signal d'initialisation de châssis s'établisse dans les châssis. L'utilisateur d'un système devrait être prévenu que la réponse de nombreux modules à un signal d'initialisation peut placer le système dans des états indésirables; cette disposition devrait ainsi être utilisée avec prudence.

A2.2 *Contrôle de l'état de l'action précédente*

Nom: CTSTAT.

Paramètre: *k* (code d'état, voir paragraphe A6.1).

Fonction: CTSTAT stocke un code d'état entier dans le paramètre *k*. Le code d'état stocké reflète les résultats de la dernière action exécutée dans le programme; le code a la valeur donnée par l'expression 4 e + d, où e et d sont des entiers non négatifs. La signification de d est donnée par le tableau suivant:

Valeur	Signification	
0	Q = 1	X = 1
1	Q = 0	X = 1
2	Q = 1	X = 0
3	Q = 0	X = 0

APPENDIX A

SYSTEM-DEPENDENT SUBROUTINES

A1. Introduction

It is often useful, even necessary, to utilize subroutines which depend strongly on an individual computer or operating system architecture. These subroutines cannot be made completely transportable even in an implementation for a highly standardized language. The functions they must perform are usually very similar, however, and may be identical within restricted classes of systems. This appendix describes a number of such system-dependent subroutines. They are not required to be in any level of implementation, but any of them may be supplied in an implementation at the discretion of the implementor. The documentation for a given implementation must indicate which ones are supplied and must describe completely their system-dependent features and parameters.

A2. Access to Special Signals

These subroutines give access to features of CAMAC systems for which a standard access method is not defined or which are not available in all systems.

A2.1 *Branch Initialize*

Name: CCINIT.

Parameter: *b* (branch number, see Sub-clause 5.2).

Function: CCINIT causes the branch initialize signal (BZ) to be asserted on the branch identified by the parameter *b*. If *b* identifies a group of crates not interfaced via a parallel branch, the function may be simulated by causing Crate Initialize signals to be asserted in the crates. The user of a system should be warned that the response of many modules to the initialize signal is such as to place the system in undesirable states; thus the feature should be used with caution.

A2.2 *Test Status of Preceding Action*

Name: CTSTAT.

Parameter: *k* (status code, see Sub-clause A6.1).

Function: CTSTAT stores an integer status code into the parameter *k*. The status code stored reflects the results of the last action executed in the program; the code has a value given by the expression 4 e + d, where e and d are non-negative integers. The meaning of d is given by the following table:

Value	Meaning	
0	Q = 1	X = 1
1	Q = 0	X = 1
2	Q = 1	X = 0
3	Q = 0	X = 0

Si e est nul, c'est qu'il n'y a pas d'erreur; si e est plus grand que zéro, une erreur ou une exception est indiquée. Les significations exactes de e sont dépendantes du système et doivent être précisées dans la documentation de chaque mise en œuvre.

A3. **Identificateur de canal**

Il peut être nécessaire pour quelques ordinateurs ou quelques systèmes d'exploitation d'utiliser des contrôleurs spéciaux de canal d'entrée-sortie (I/O) ou des dispositions du système d'exploitation tels que des numéros d'unités logiques et des codes de dispositifs pour accéder aux systèmes CAMAC. Les deux sous-programmes suivants donnent les moyens de spécifier ou d'obtenir de telles informations.

A3.1 *Déclaration de canal*

Nom: CDCHN.

Paramètres: *chan* (identificateur de canal, voir paragraphe A6.2);
paramètres additionnels, dépendants de la mise en œuvre.

Fonction: CDCHN code les paramètres dépendants du système sous une forme convenable et renvoie les résultats en tant que valeur de *chan.* Il spécifie les moyens en matériel et/ou logiciel d'entrée-sortie (I/O) de l'ordinateur que l'on peut utiliser et éventuellement des renseignements supplémentaires sur leurs utilisations. Les paramètres additionnels doivent être complètement décrits dans la documentation, pour chaque mise en œuvre.

A3.2 *Analyse de la déclaration de canal*

Nom: CGCHN.

Paramètres: *chan* (identificateur de canal, voir paragraphe A6.2);
paramètres additionnels, dépendants de la mise en œuvre.

Fonction: CGCHN extrait les paramètres dépendants du système de l'identificateur de canal *chan.* Il effectue le processus inverse réalisé par CDCHN.

A4. **Transferts de mot de données court**

Dans de nombreux systèmes et applications, il est gênant et inefficace d'employer la longueur totale de 24 bits d'un mot de données CAMAC et suffisant d'utiliser un mot de données tronqué. La troncature appropriée dépend du système, mais le choix le plus fréquent est déterminé par la longueur du mot de l'ordinateur. Les bits du mot de données CAMAC les plus à gauche sont supprimés tandis que les bits les plus à droite sont retenus. La longueur du mot de données CAMAC tronqué doit être spécifiée dans la documentation. Ces sous-programmes répètent les fonctions réalisées par les sous-programmes décrits dans l'article 4, à cette exception près que les mots de données CAMAC tronqués sont utilisés.

A4.1 *Exécution d'une action simple CAMAC*

Nom: CSSA.

Paramètres: *f* (code de fonction, voir paragraphe 5.6);
ext (adresse externe, voir paragraphe 5.1);
ints (mot de données CAMAC tronqué, voir paragraphe A6.3);
q (réponse Q, voir paragraphe 5.8).

Fonction: CSSA accomplit la même fonction que CFSA (voir paragraphe 4.1.2) sauf que *ints* contient un mot de données CAMAC tronqué.

If e is zero, there are no errors reported; if e is greater than zero, then some error or exception is indicated. The exact meanings of e are system-dependent and must be defined in the documentation for each implementation.

A3. Channel Identifier

It may be necessary in some computers or in some operating systems to utilize special I/O channel controllers or operating system features such as logical unit numbers and device codes to access CAMAC systems. These two subroutines provide a means to specify or interrogate such information.

A3.1 *Declare Channel*

Name: CDCHN.

Parameters: *chan* (channel identifier, see Sub-clause A6.2);
additional parameters, depending on implementation.

Function: CDCHN encodes system-dependent parameters into a convenient form and returns the results as the value of *chan.* It specifies the computer hardware and/or software I/O facilities to be used and possibly additional information relating to their use. The additional parameters must be completely described in the documentation for each implementation.

A3.2 *Analyze Channel Declaration*

Name: CGCHN.

Parameters: *chan* (channel identifier, see Sub-clause A6.2);
additional parameters, depending on implementation.

Function: CGCHN extracts the system-dependent parameters from the channel identifier *chan.* It reverses the process performed by CDCHN.

A4. Short Data-Word Transfers

In many systems and applications it is inconvenient and inefficient to utilize the full twenty-four bit length of the CAMAC data word and sufficient to use a truncated data word. The appropriate degree of truncation is dependent on the system, but the computer word length is the most common choice. The leftmost bits of the CAMAC data word are discarded, while the rightmost bits are retained. The length of a truncated CAMAC data word must be specified in the documentation. These subroutines duplicate functions performed by subroutines described in Clause 4, except that truncated CAMAC data words are used.

A4.1 *Perform Single CAMAC Action*

Name: CSSA.

Parameters: *f* (function code, see Sub-clause 5.6);
ext (external address, see Sub-clause 5.1);
ints (truncated CAMAC data word, see Sub-clause A6.3);
q (Q response, see Sub-clause 5.8).

Function: CSSA performs the same function as CFSA (see Sub-clause 4.1.2) except that *ints* contains a truncated CAMAC data word.

A4.2 *Action multiple générale*

Nom: CSGA.

Paramètres: *fa* (codes de fonction, voir paragraphe 5.14);
exta (adresses externes, voir paragraphe 5.15);
intt (tableau de données CAMAC tronquées, voir paragraphe A6.4);
qa (réponses Q, voir paragraphe 5.17);
cb (bloc de contrôle, voir paragraphe 5.18).

Fonction: CSGA accomplit la même fonction que CFGA (voir paragraphe 4.3.1) sauf que les éléments de *intt* sont des mots de données CAMAC tronqués.

A4.3 *Scrutation d'adresses*

Nom: CSMAD.

Paramètres: *f* (code de fonction, voir paragraphe 5.6);
extb (adresses externes, voir paragraphe 5.19);
intt (tableau de données CAMAC tronquées, voir paragraphe A6.4);
cb (bloc de contrôle, voir paragraphe 5.18).

Fonction: CSMAD accomplit la même fonction que CFMAD (voir paragraphe 4.3.2) sauf que les éléments de *intt* sont des mots de données CAMAC tronqués.

A4.4 *Transfert de bloc synchronisé par le contrôleur*

Nom: CSUBC.

Paramètres: *f* (code de fonction, voir paragraphe 5.6);
ext (adresses externes, voir paragraphe 5.1);
intt (tableau de données CAMAC tronquées, voir paragraphe A6.4);
cb (bloc de contrôle, voir paragraphe 5.18).

Fonction: CSUBC accomplit la même fonction que CFUBC (voir paragraphe 4.3.3) sauf que les éléments de *intt* sont des mots de données CAMAC tronqués.

A4.5 *Transfert de bloc synchronisé par LAM*

Nom: CSUBL.

Paramètres: *f* (code de fonction, voir paragraphe 5.6);
ext (adresse externe, voir paragraphe 5.1);
intt (tableau de données CAMAC tronquées, voir paragraphe A6.4);
cb (bloc de contrôle, voir paragraphe 5.18).

Fonction: CSUBL accomplit la même fonction que CFUBL (voir paragraphe 4.3.4) sauf que les éléments de *intt* sont des mots de données CAMAC tronqués.

A4.6 *Transfert de bloc en mode répétitif*

Nom: CSUBR.

Paramètres: *f* (code de fonction, voir paragraphe 5.6);
ext (adresse externe, voir paragraphe 5.1);
intt (tableau de données CAMAC tronquées, voir paragraphe A6.4);
cb (bloc de contrôle, voir paragraphe 5.18).

Fonction: CSUBR accomplit la même fonction que CFUBR (voir paragraphe 4.3.5) sauf que les éléments de *intt* sont des mots de données CAMAC tronqués.

A4.2 *General Multiple Action*

Name: CSGA.

Parameters: *fa* (function codes, see Sub-clause 5.14);
exta (external addresses, see Sub-clause 5.15);
intt (truncated CAMAC data array, see Sub-clause A6.4);
qa (Q responses, see Sub-clause 5.17);
cb (control block, see Sub-clause 5.18).

Function: CSGA performs the same function as CFGA (see Sub-clause 4.3.1) except that the elements of *intt* are truncated CAMAC data words.

A4.3 *Address Scan*

Name: CSMAD.

Parameters: *f* (function code, see Sub-clause 5.6);
extb (external addresses, see Sub-clause 5.19);
intt (truncated CAMAC data array, see Sub-clause A6.4);
cb (control block, see Sub-clause 5.18).

Function: CSMAD performs the same function as CFMAD (see Sub-clause 4.3.2) except that the elements of *intt* are truncated CAMAC data words.

A4.4 *Controller-Synchronized Block Transfer*

Name: CSUBC.

Parameters: *f* (function code, see Sub-clause 5.6);
ext (external addresses, see Sub-clause 5.1);
intt (truncated CAMAC data array, see Sub-clause A6.4);
cb (control block, see Sub-clause 5.18).

Function: CSUBC performs the same function as CFUBC (see Sub-clause 4.3.3) except that the elements of *intt* are truncated CAMAC data words.

A4.5 *LAM-Synchronized Block Transfer*

Name: CSUBL.

Parameters: *f* (function code, see Sub-clause 5.6);
ext (external address, see Sub-clause 5.1);
intt (truncated CAMAC data array, see Sub-clause A6.4);
cb (control block, see Sub-clause 5.18).

Function: CSUBL performs the same function as CFUBL (see Sub-clause 4.3.4) except that the elements of *intt* are truncated CAMAC data words.

A4.6 *Repeat Mode Block Transfer*

Name: CSUBR.

Parameters: *f* (function code, see Sub-clause 5.6);
ext (external address, see Sub-clause 5.1);
intt (truncated CAMAC data array, see Sub-clause A6.4);
cb (control block, see Sub-clause 5.18).

Function: CSUBR performs the same function as CFUBR (see Sub-clause 4.3.5) except that the elements of *intt* are truncated CAMAC data words.

A5. **Définition de l'identificateur de châssis**

Nom: CDCRT.

Paramètres: *c* (numéro de châssis, voir paragraphe 5.3);
intb (tableau d'entiers, voir paragraphe A6.5).

Fonction: CDCRT définit le numéro de châssis *c* en termes d'informations dépendantes du système et contenues dans le tableau *intb*. Il permet de changer le numéro de châssis *c* au cours d'un déroulement de procédure.

A6. **Définitions de paramètres**

Bien que de nombreux paramètres utilisés dans les sous-programmes de l'annexe A soient identiques à ceux qui sont décrits dans l'article 5 de la partie principale de cette norme, on en a introduit quelques-uns nouveaux. On décrit ci-après les paramètres se rapportant uniquement à la présente annexe.

A6.1 *k (code d'état)*

Le symbole *k* représente un entier utilisé pour fournir une information d'état concernant une action CAMAC. Les deux bits les plus à droite contiennent les compléments des réponses X et Q; les bits restant dans le mot contiennent les codes d'erreur dépendants du système. Le bit de poids le plus faible contient le complément de la réponse Q; le bit suivant à gauche (le 2 s bit) contient le complément de X.

A6.2 *chan (identificateur de canal)*

Le symbole *chan* représente un entier qui contient l'identificateur de canal.

A6.3 *ints (mot de données CAMAC tronqué)*

Le symbole *ints* représente une cellule de stockage dans la mémoire de l'ordinateur utilisée pour contenir les bits 1 à n d'un mot de données CAMAC où n est défini, pour la mise en œuvre, comme étant inférieur à vingt-quatre. Les bits de poids les plus forts du mot CAMAC ne sont pas utilisés lorsque la fonction est une LECTURE; ils sont mis à zéro si la fonction est une ÉCRITURE.

A6.4 *intt (tableau de données CAMAC tronquées)*

Le symbole *intt* représente un tableau dans la mémoire de l'ordinateur; chacun de ses éléments contient un mot de données CAMAC tronqué possédant la même forme que *ints* (voir paragraphe A6.3).

A6.5 *intb (tableau d'entiers)*

Le symbole *intb* représente un tableau d'entiers, dont la longueur et le contenu ne sont pas définis dans cette norme. Il est destiné à contenir une information dépendante du système ou de la mise en œuvre, associée à la définition d'un identificateur de numéro de châssis. L'information peut inclure le numéro physique du châssis, l'identification de l'interface de l'Interconnexion ou de l'ordinateur ou les paramètres du système d'exploitation nécessaires à l'accès. La documentation d'une mise en œuvre doit décrire les caractéristiques de tous les paramètres contenus dans ce tableau.

A5. **Define Crate Identifier**

Name: CDCRT.

Parameters: *c* (crate number, see Sub-clause 5.3);
intb (integer array, see Sub-clause A6.5).

Function: CDCRT defines the crate number *c* in terms of the system-dependent information contained in the array *intb*. It permits the meaning of the crate number *c* to be changed within a procedure execution.

A6. **Definitions of Parameters**

Although many of the parameters used in the subroutines of Appendix A are identical with those described in Clause 5 of the main body of this standard, a few new ones have been introduced. The parameters referred to only in this appendix are described below.

A6.1 *k (status code)*

The symbol *k* represents an integer used to provide status information with respect to a CAMAC action. The rightmost two bits contain the complements of the X and Q responses; the remaining bits in a word contain system-dependent error codes. The lowest-order bit contains the complement of the Q response; the next bit to the left (the 2 s bit) contains the complement of X.

A6.2 *chan (channel identifier)*

The symbol *chan* represents an integer which contains a channel identifier.

A6.3 *ints (truncated CAMAC data word)*

The symbol *ints* represents a storage cell in computer memory used to hold bits 1 to n of a CAMAC data word, where n is defined for the implementation to be less than twenty-four. Higher bits in the CAMAC word are not used when the function is a READ; they are set to zeros if the function is a WRITE.

A6.4 *intt (truncated CAMAC data array)*

The symbol *intt* represents an array in computer memory each element of which contains a truncated CAMAC data word having the same form as *ints* (see Sub-clause A6.3).

A6.5 *intb (integer array)*

The symbol *intb* represents an integer array, the length and contents of which are not defined in this standard. It is intended to contain system-dependent or implementation-dependent information associated with the definition of a crate number identifier. The information may include physical crate number, highway or computer interface identification, or operating system parameters necessary for access. The documentation for an implementation must describe the requirements for any parameters contained in this array.

ANNEXE B

MISE EN ŒUVRE EN FORTRAN

B1. Généralités

Cette annexe recommande une mise en œuvre, pour le langage FORTRAN, des sous-programmes normalisés CAMAC afin de communiquer avec des dispositifs reliés par CAMAC et d'exécuter des programmes synchronisés avec des événements associés à de tels dispositifs.

B2. Description des sous-programmes

B2.1 *Sous-programmes de base*

B2.1.1 *Déclaration d'un registre CAMAC*

Forme: CALL CDREG (*ext, b, c, n, a*)

Fonction: CDREG réunit les composants *b, c, n* et *a* en une référence à un registre unique CAMAC.

Notes de mise en œuvre

ext: le sous-programme doit renvoyer une valeur qui définit de façon unique le registre ou châssis CAMAC désigné pour d'autres sous-programmes de la mise en œuvre. Le moyen par lequel cet objectif est atteint n'est pas spécifié. La procédure la plus fréquente est de renvoyer une valeur dans laquelle les paramètres *b, c, n* et *a* ont été groupés en un seul entier avec les champs définis pour que l'exécution des actions CAMAC soit le plus efficace sur l'interface où s'effectue la mise en œuvre. D'autres possibilités sont admises cependant. Par exemple, les valeurs des paramètres pourraient être stockées dans une structure de données et un pointeur d'entrée renvoyé. Si *n* et *a* sont définis comme étant nuls, *ext* est alors défini comme étant un châssis. Si *n* et *a* ne sont pas nuls, *ext* est alors défini comme étant un registre ou une autre entité adressable.

Tables de références de registres

Il est avantageux dans de nombreuses applications d'avoir des tables de références de registres CAMAC, identiques quant à leur forme à la valeur renvoyée par CDREG et disponibles sur une base permanente, pouvant être chargées par des programmes ou sous forme de tableaux de constantes dans les programmes. De telles tables, non seulement épargnent le temps nécessaire pour définir les références à un registre chaque fois qu'un programme est exécuté, mais encore peuvent constituer des parties de tables plus grandes de définitions de dispositifs logiques. Si de telles possibilités peuvent exister dans une mise en œuvre, la documentation devra décrire comment créer ces références en utilisant le code assembleur ou par les déclarations de données du compilateur.

B2.1.2 *Exécution d'une action simple CAMAC*

Forme: CALL CFSA (*f, ext, int, q*)

Fonction: ce sous-programme accomplit une fonction simple CAMAC à une adresse unique CAMAC. Si la fonction *f* est une fonction de lecture ou d'écriture, un mot de données de 24 bits est transféré.

APPENDIX B

FORTRAN IMPLEMENTATION

B1. General

This appendix recommends an implementation of the CAMAC standard subroutines for use with FORTRAN for the purpose of communicating with CAMAC-interfaced devices and synchronizing program execution with events associated with such devices.

B2. Description of Subroutines

B2.1 *Primary Subroutines*

B2.1.1 *Declare CAMAC Register*

Form: CALL CDREG (*ext, b, c, n, a*).

Function: CDREG combines the components *b, c, n,* and *a* into a single CAMAC register reference.

Implementation notes

ext: the subroutine must return a value which uniquely defines the specified CAMAC register or crate for other subroutines in the implementation. The means by which this objective is accomplished are not specified. The most common procedure is expected to be to return a value in which the parameters *b, c, n,* and *a* have been packed into a single integer with the fields defined for the most efficient execution of CAMAC actions on the interface to which the implementation applies. Other possibilities are allowed, however. For example, the values of the parameters could be stored in a data structure and a pointer to the entry returned. If *n* and *a* are defined to be zero, then *ext* is defined to be a crate. If *n* and *a* are not zero, then *ext* is defined to be a register or other addressable entity.

Tables of Register References

It is advantageous in many applications to have tables of CAMAC register references, identical in form to the value returned by CDREG, available on a permanent basis to be loaded by programs or as arrays of constants within programs. Such tables not only save the time required to define the register references each time a program is executed but also can form parts of larger tables of logical device definitions. If such facilities can be supported by an implementation, the documentation should describe how to create these references in assembler code or in DATA statements in the compiler.

B2.1.2 *Perform Single CAMAC Action*

Form: CALL CFSA (*f, ext, int, q*).

Function: this subroutine performs a single CAMAC function at a single CAMAC address. If the function *f* is a read or a write function, a 24-bit data word is transferred.

Notes de mise en œuvre

int: la référence interne *int* doit être capable de contenir 24 bits. Si la mise en œuvre FORTRAN accepte des variables entières de 24 bits ou plus, *int* peut être une expression entière (pour l'écriture) ou une variable entière ou un élément de tableau (pour la lecture). Sinon *int* doit être un tableau d'entiers (ou une partie), avec assez d'éléments pour contenir 24 bits. Lorsqu'un mot de données CAMAC est stocké dans une variable entière ou lorsqu'une constante entière est utilisée en tant que *int*, le bit 1 CAMAC occupe la position du bit de poids faible et le bit 24 occupe la vingt-quatrième position à gauche (dans une machine binaire; dans d'autres bases, on doit faire une transformation appropriée en valeurs entières équivalentes). Si la mise en œuvre FORTRAN ne peut accepter des entiers d'une longueur d'au moins 24 bits, les bits du mot de données CAMAC sont stockés dans les éléments du tableau de telle façon que le bit 1 soit le bit de poids le plus faible dans l'élément de numéro le plus élevé. Tous les bits dans l'élément le plus haut sont remplis et les bits restants sont stockés en commençant par le bit de rang le plus faible de l'élément numéroté le plus bas suivant, etc. Pour ce qui est sans doute l'exemple le plus commun, les mots de 16 bits, on pourrait définir un tableau de longueur 2, appelé CAMDAT. Dans ce cas, les bits 1 à 16 sont stockés dans CAMDAT (2) avec le bit 1 le plus faible et le bit 16 le plus fort. Les bits 17 à 24 sont stockés dans CAMDAT (1) avec le bit 17 dans la position d'ordre la plus faible dans le mot. Les huit bits supérieurs de CAMDAT (1) sont mis à zéro.

Restrictions sur les valeurs de paramètres

Dans les systèmes à multiples utilisateurs où les moyens du CAMAC sont partagés entre plusieurs utilisateurs indépendants, il peut être nécessaire de restreindre l'accès aux contrôleurs de châssis et à certains modules en n'admettant pas que les actions de contrôleur de châssis soient émises via CFSA et en interdisant l'accès via CFSA à certains autres modules utilisés pour la commande du système. Dans des systèmes plus simples, cependant, de telles restrictions ne sont pas nécessaires et empêchent la mise en œuvre de codes d'essais et de diagnostics. Chaque réalisateur doit être juge de sa propre situation. De telles restrictions peuvent, bien sûr, être commandées en plaçant des commutateurs de mode soit pendant l'exploitation, soit à la formation du système. Lorsque de telles restrictions apparaissent nécessaires, il est recommandé que la mise en œuvre laisse au gestionnaire du système l'option d'y avoir recours ou non.

B2.2 *Sous-programmes d'action simple*

B2.2.1 *Emission du signal d'initialisation sur l'Interconnexion*

Forme: CALL CCCZ *(ext).*

Fonction: ce sous-programme provoque l'émission du signal d'initialisation (Z) sur l'Interconnexion du châssis spécifié.

B2.2.2 *Emission de la remise à zéro du châssis*

Forme: CALL CCCC *(ext).*

Fonction: ce sous-programme provoque l'émission du signal de remise à zéro (C) sur l'Interconnexion du châssis spécifié.

B2.2.3 *Etablissement ou suppression de l'inhibition sur l'Interconnexion*

Forme: CALL CCCI *(ext, l).*

Fonction: ce sous-programme provoque l'établissement de l'inhibition sur le châssis spécifié si *l* est «vrai» et sa suppression si *l* est «faux».

Implementation Notes

int: the internal reference *int* must be able to contain 24 bits. If the FORTRAN implementation supports integer variables of 24 bits or more, *int* can be an integer expression (for write) or an integer variable or array element (for read). Otherwise *int* must be an integer array (or a portion of one) with enough elements to contain 24 bits. When a CAMAC data word is stored in an integer variable, or when an integer constant is used as *int*, CAMAC bit 1 occupies the low-order bit position, and bit 24 occupies the twenty-fourth bit position to the left (in a binary machine; in other radices, an appropriate transformation to equivalent integer values must be made). If the FORTRAN implementation does not support integers with a length as great as 24 bits, then the bits of the CAMAC data word are stored in the array elements such that bit 1 is the lowest order bit in the highest numbered element. All the bits in the highest element are filled and the remaining bits are stored beginning in the lowest order bit of the next lowest numbered element, etc. For what is probably the most common example, 16-bit words, one could define an array of length 2, say CAMDAT. Then bits 1 to 16 are stored in CAMDAT (2) with bit 1 the lowest and bit 16 the highest. Bits 17 to 24 are stored in CAMDAT (1) with bit 17 in the lowest order position in the word. The upper eight bits of CAMDAT (1) are cleared to zero.

Restrictions on Parameter Values

In multiple-user systems where CAMAC facilities are shared among several independent users, it may be necessary to restrict access to crate controllers and certain modules by not allowing crate controller actions to be issued via CFSA and by forbidding access via CFSA to certain other modules used for system control. In simpler systems, however, such restrictions are unnecessary and prevent the implementation of test and diagnostic codes. Each implementor must judge his own situation. Such restrictions can, of course, be controlled by setting mode switches either during operation or at system generation. Where such restrictions appear to be necessary, it is recommended that the implementation give the system manager the option of invoking them or not.

B2.2 *Single Action Subroutines*

B2.2.1 *Generate Dataway Initialize*

Form: CALL CCCZ *(ext).*

Function: this subroutine generates the crate Initialize signal (Z) on the Dataway of the addressed crate.

B2.2.2 *Generate Crate Clear*

Form: CALL CCCC *(ext).*

Function: this subroutine generates the crate Clear signal (C) on the Dataway of the addressed crate.

B2.2.3 *Set or Clear Dataway Inhibit*

Form: CALL CCCI *(ext, l).*

Function: this subroutine sets the crate Inhibit signal in the addressed crate if *l* is "true" and clears it if *l* is "false".

B2.2.4 *Contrôle de l'inhibition sur l'Interconnexion*

Forme: CALL CTCI *(ext, l).*

Fonction: ce sous-programme renvoie la valeur «vrai» dans *l* si le signal d'inhibition est présent dans le châssis spécifié, «faux» s'il ne l'est pas.

B2.2.5 *Mise en service ou hors service des demandes de châssis*

Forme: CALL CCCD *(ext, l).*

Fonction: ce sous-programme met en service ou hors service la demande du châssis spécifié.

B2.2.6 *Contrôle de la mise en service de la demande de châssis*

Forme: CALL CTCD *(ext, l).*

Fonction: ce sous-programme renvoie dans *l* la valeur «vrai» si la demande de châssis est en service, «faux» si elle ne l'est pas.

B2.2.7 *Contrôle de la présence de demande*

Forme: CALL CTGL *(ext, l).*

Fonction: ce sous-programme renvoie dans *l* la valeur «vrai» si n'importe quel bit GL est présent dans le châssis, «faux» s'il n'y en a aucun.

B2.2.8 *Déclaration de LAM*

Forme: CALL CDLAM *(lam, b, c, n, m, inta).*

Fonction: CDLAM réunit les composantes *b, c, n, m* et *inta* dans une seule référence pour le LAM.

Notes de mise en œuvre

lam: ce sous-programme doit renvoyer une valeur qui définit de façon unique le LAM désigné pour les autres sous-programmes de la mise en œuvre. Le moyen par lequel cet objectif est atteint n'est pas spécifié. Si les paramètres nécessaires pour spécifier l'adresse d'un LAM et toute information auxiliaire utile peuvent être groupés en une variable entière unique, la valeur retournée peut alors contenir cette information. Sinon, il peut être nécessaire de fournir un index arbitraire, ou un autre code, qui identifie le LAM pour les autres sous-programmes qui y accèdent.

Affectation permanente des LAM logiques

Dans de nombreux systèmes, les LAM ne sont pas dynamiques et peuvent être connus par le système opérateur ou par les programmes entrées-sorties (I/O) CAMAC de façon plus ou moins permanente. Dans de tels cas, la fonction CDLAM n'est pas requise. Les LAM peuvent être définis via des tables qui font partie du logiciel entrées-sorties (I/O) CAMAC ou qui peuvent être chargées par un programme particulier pour identifier les LAM qu'il traite. Si une mise en œuvre particulière admet les définitions de telles tables, la documentation devra décrire comment créer ces références de LAM en utilisant le code assembleur ou les déclarations de données du compilateur.

B2.2.9 *Mise en service ou hors service de LAM*

Forme: CALL CCLM *(lam, l).*

Fonction: ce sous-programme met en service ou hors service le LAM du module désigné par *lam.*

B2.2.4 *Test Dataway Inhibit*

Form: CALL CTCI *(ext, l).*

Function: this subroutine returns the value "true" in *l* if the inhibit signal in the addressed crate is set, "false" if not.

B2.2.5 *Enable or Disable Crate Demands*

Form: CALL CCCD *(ext, l).*

Function: this subroutine enables or disables demands from the specified crate.

B2.2.6 *Test Crate Demand Enabled*

Form: CALL CTCD *(ext, l).*

Function: this subroutine returns in *l* the value "true" if crate demands are enabled, "false" if not.

B2.2.7 *Test Demand Present*

Form: CALL CTGL *(ext, l).*

Function: this subroutine returns in *l* the value "true" if any GL bit is present in the crate, "false" if not.

B2.2.8 *Declare LAM*

Form: CALL CDLAM *(lam, b, c, n, m, inta).*

Function: CDLAM combines the components *b, c, n, m,* and *inta* into a single reference to the LAM.

Implementation Notes

lam: the subroutine must return a value which uniquely defines the specified LAM for other subroutines in the implementation. The means by which this objective is accomplished are not specified. If the parameters necessary to specify the address of a LAM and any necessary auxiliary information can be packed into a single integer variable, then the value returned may contain this information. Otherwise, it may be necessary to supply an arbitrary index, or other code, which identifies the LAM to the other subroutines which access it.

Permanent Assignment of Logical LAM's

In many system LAM's are not dynamic and can be known to the operating system or the CAMAC I/O programs more or less permanently. In such cases, the function CDLAM is not required. The LAM's can be defined via tables which are part of the CAMAC I/O software or which may be loaded by a particular program to identify the LAM's with which it deals. If a particular implementation supports definition by such tables, the documentation should describe how to create LAM references using assembler code or compiler DATA statements.

B2.2.9 *Enable or Disable LAM*

Form: CALL CCLM *(lam, l).*

Function: this subroutine enables or disables the module LAM identified by *lam.*

B2.2.10 *Remise à zéro de LAM*

Forme: CALL CCLC *(lam).*

Fonction: ce sous-programme remet à zéro le LAM du module désigné par l'argument.

B2.2.11 *Contrôle de LAM*

Forme: CALL CTLM *(lam, l).*

Fonction: ce sous-programme renvoie dans *l* la valeur «vrai» si le LAM désigné est présent, «faux» s'il ne l'est pas.

B2.2.12 *Liaison du LAM à la procédure de service*

Forme: CALL CCLNK *(lam, label).*

Fonction: établit une liaison lors de l'exécution entre un LAM logique et un programme de traitement d'interruption.

B2.3 *Transferts de bloc, actions multiples et déclarations inverses*

B2.3.1 *Action multiple générale*

Forme: CALL CFGA *(fa, exta, intc, qa, cb).*

Fonction: ce sous-programme effectue une séquence de fonctions CAMAC correspondant à une séquence d'adresses CAMAC et renvoie une séquence de réponses Q. Le nombre de fonctions exécutées, d'adresses auxquelles on a accédé et de réponses Q renvoyées est donné par la valeur du paramètre *cb* (1). Les mots de données CAMAC sont transférés entre le système CAMAC et le tableau *intc* toutes les fois que la fonction spécifiée est une lecture ou une écriture. Puisque le tableau *fa* peut contenir à la fois des fonctions de lecture et d'écriture, les mots de données peuvent être écrits ou lus dans le même tableau sur appel unique à ce sous-programme. Les positions des mots de données CAMAC dans le tableau *intc* qui correspondent à des fonctions dans le tableau *fa* qui ne sont ni de lecture ni d'écriture ne sont pas accessibles par ce sous-programme.

Notes de mise en œuvre

fa: noter que, dans ce sous-programme, il est possible qu'une succession de fonctions sans rapport entre elles soient exécutées à une succession d'adresses CAMAC.

intc: la référence interne *intc*, à moins que toutes les fonctions dans le tableau *fa* ne soient des fonctions de commande et de contrôle qui n'utilisent pas de données, doit être capable de contenir un mot de données CAMAC pour toute fonction exécutée. Afin de maintenir une correspondance de position entre les valeurs dans *fa, exta, intc* et *qa*, une position dans *intc* est sautée toutes les fois qu'une fonction de commande ou de contrôle est exécutée. Si la mise en œuvre FORTRAN comporte des éléments de tableau d'entiers de 24 bits ou plus, chaque exécution de la fonction accède à un élément successif du tableau. Le mot de données CAMAC occupe les 24 bits de poids faibles de l'élément du tableau d'entiers; les bits de poids plus forts sont mis à zéro par la fonction de lecture et ignorés par celle d'écriture. Si la mise en œuvre FORTRAN comporte seulement des tableaux d'entiers dont les éléments contiennent moins de 24 bits, chaque mot de données CAMAC est alors contenu dans un ensemble d'éléments (un groupe) suffisamment grand pour contenir 24 bits. La longueur du tableau doit être assez grande pour inclure un groupe pour chaque mot de données CAMAC transféré. Chaque mot de données CAMAC est placé dans un groupe de telle façon que l'élément dans un groupe d'indice le plus haut soit rempli avec les bits de poids les plus faibles du mot de données, et

B2.2.10 *Clear LAM*

Form: CALL CCLC *(lam).*

Function: this subroutine clears the module LAM identified by the argument.

B2.2.11 *Test LAM*

Form: CALL CTLM *(lam, l).*

Function: this subroutine returns in *l* the value "true" if the addressed LAM is asserted, "false" if not.

B2.2.12 *Link LAM to Service Procedure*

Form: CALL CCLNK *(lam, label).*

Function: perform a run time association between a logical LAM and an interrupt service routine.

B2.3 *Block Transfers, Multiple Actions, and Inverse Declarations*

B2.3.1 *General Multiple Action*

Form: CALL CFGA *(fa, exta, intc, qa, cb).*

Function: this subroutine performs a sequence of CAMAC functions at a corresponding sequence of CAMAC addresses and returns a sequence of Q responses. The number of functions executed, addresses accessed, and Q responses returned is given by the value of the parameter *cb* (1). CAMAC data words are transferred between the CAMAC system and the array *intc* whenever the specified function is read or write. Since the array *fa* can contain both read and write functions, data words may be written to and read from the same array by a single call to this subroutine. CAMAC data word positions in the array *intc* which correspond to functions in the array *fa* which are neither read nor write are not accessed by this subroutine.

Implementation Notes

fa: note that in this subroutine a succession of possibly unrelated functions are executed at a succession of CAMAC addresses.

intc: the internal reference *intc,* unless all functions in the array *fa* are control or test functions, which use no data, must be able to contain a CAMAC data word for each function executed. In order to maintain the positional correspondence between values in *fa, exta, intc,* and *qa,* a position in *intc* is skipped whenever a control or test function is executed. If the FORTRAN implementation supports integer array elements of 24 bits or more, then each execution of the function accesses a successive element of the array. The CAMAC data word occupies the low-order 24 bits of the integer array element; higher order bits are cleared by a read function and ignored by a write. If the FORTRAN implementation supports only integer arrays whose elements contain fewer than 24 bits, then each CAMAC data word is contained in an integral number of elements (a "block") sufficiently large to contain 24 bits. The length of the array must be great enough to include a block for each CAMAC data word transferred. Each CAMAC data word is positioned within its block so that the element within the block with the highest index is filled with the lowest order bits of the data word, and any bits not required to contain data are at the highest order positions of the element with the lowest index. Thus in the typical 16-bit case, where a block consists of elements n and n + 1,

tous les bits non utilisés pour contenir la donnée occupent les positions de poids les plus forts de l'élément d'indice le plus bas. Ainsi, dans le cas typique des 16 bits, où un groupe est constitué par les éléments n et n + 1, l'élément n contient les bits CAMAC 24 à 17 dans ses huit bits de poids faible et l'élément n + 1 contient les bits 16 à 1.

qa: ce tableau contient les réponses Q de chaque fonction exécutée. Le format de chaque élément du tableau *qa* est identique au format de la variable *q* dans le sous-programme CFSA, c'est-à-dire qu'une valeur logique est renvoyée. Aucune action dépendante de ces valeurs n'est prise par le sous-programme.

Restrictions sur les valeurs de paramètres

Dans les systèmes à multiples utilisateurs où les moyens du CAMAC sont partagés entre de nombreux utilisateurs indépendants, il peut être nécessaire de restreindre l'accès aux contrôleurs de châssis et à certains modules en n'admettant pas que les actions de contrôleur de châssis soient émises via CFGA et en interdisant l'accès via CFGA à certains autres modules utilisés pour la commande du système. Dans des systèmes plus simples cependant, de telles restrictions ne sont pas nécessaires et empêchent la mise en œuvre de codes d'essais et de diagnostics. Chaque réalisateur doit être juge de sa propre situation. De telles restrictions peuvent, bien sûr, être commandées en plaçant des commutateurs de mode soit pendant l'exploitation, soit à la formation du système. Lorsque de telles restrictions apparaissent nécessaires, il est recommandé que la mise en œuvre laisse au gestionnaire du système l'option d'y avoir recours ou non.

Applicabilité

Ce sous-programme est surtout utile dans les systèmes où le temps mort de passage des paramètres associés à l'initialisation ou à la spécification d'une action CAMAC se trouve être très grand par rapport au temps requis pour exécuter une commande individuelle CAMAC. Les systèmes affectés de grands temps morts pour l'exécution d'entrées-sorties par le logiciel ou dans lesquels existe un processeur éloigné avec des durées de communications relativement importantes font partie de ceux qui peuvent avoir besoin de ce sous-programme.

B2.3.2 *Scrutation d'adresses*

Forme: CALL CFMAD *(f, extb, intc, cb).*

Fonction: ce sous-programme exécute une action simple CAMAC, *f*, à une succession d'adresses calculées en utilisant l'algorithme de scrutation d'adresses.

Notes de mise en œuvre

intc: voir la note de mise en œuvre sur *intc* dans le paragraphe B2.3.1.

Q: les réponses Q aux fonctions CAMAC ne sont pas sauvegardées. La réponse Q est utilisée pour commander l'algorithme d'adresse et peut ne pas fournir d'information au programme appelant.

X: une attention particulière doit être portée à la réponse X dans le mode de scrutation d'adresses. Quand un module répond Q = 0, il peut en même temps donner soit X = 1, soit X = 0; en conséquence, dans ce mode d'accès, la réponse X est ignorée lorsque la réponse Q est 0.

Restrictions sur les valeurs de paramètres

Dans les systèmes à multiples utilisateurs où les moyens du CAMAC sont partagés entre plusieurs utilisateurs indépendants, il peut être nécessaire de restreindre l'accès aux contrôleurs de châssis et à certains modules en n'admettant pas que les actions de contrôleur de châssis soient émises via CFMAD et en interdisant l'accès via CFMAD à certains autres modules utilisés pour la commande du système.

element n contains CAMAC bits 24 to 17 in its low-order eight bits and element n + 1 contains bits 16 through 1.

qa: this array contains the Q responses for each function executed. The format of each element of the array *qa* is identical to the format of the variable *q* in subroutine CFSA, i.e. a logical value is returned. No action based on these values is taken by the subroutine.

Restrictions on Parameter Values

In multiple-user systems where CAMAC facilities are shared among several independent users, it may be necessary to restrict access to crate controllers and certain modules by not allowing crate controller actions to be issued via CFGA and by forbidding access via CFGA to certain other modules used for system control. In simpler systems, however, such restrictions are unnecessary and prevent the implementation of test and diagnostic codes. Each implementor must judge his own situation. Such restrictions can, of course, be controlled by setting mode switches either during operation or at system generation. Where such restrictions appear to be necessary, it is recommended that the implementation give the system manager the option of invoking them or not.

Applicability

This subroutine is useful primarily in systems in which the overhead associated with initiating or specifying a CAMAC action is great compared with the time required to perform an individual CAMAC command. Systems which suffer from a high I/O overhead in the system software or where there is a remote intelligent processor with a relatively high communications overhead are among those in which this subroutine may be needed.

B2.3.2 *Address-Scan*

Form: CALL CFMAD *(f, extb, intc, cb).*

Function: this subroutine executes a single CAMAC operation, *f*, at a succession of addresses computed using the Address Scan algorithm.

Implementation Notes

intc: see implementation note on *intc* in Sub-clause B2.3.1.

Q: the Q responses to the CAMAC functions are not saved. The Q response is used to control the addressing algorithm and can provide no information to the calling program.

X: special attention must be paid to the X response in the Address Scan mode. When a module responds with Q = 0, it may at the same time give either X = 1 or X = 0; consequently, in this mode of access, the X response is ignored when the Q response is 0.

Restrictions on Parameter Values

In multiple-user systems where CAMAC facilities are shared among several independent users, it may be necessary to restrict access to crate controllers and certain modules by not allowing crate controller actions to be issued via CFMAD and by forbidding access via CFMAD to certain other modules used for system control.

B2.3.3 *Transfert de bloc synchronisé par le contrôleur*

Forme: CALL CFUBC *(f, ext, intc, cb).*

Fonction: ce sous-programme exécute une action simple CAMAC, *f*, à une adresse unique, avec l'usage de la réponse Q définie comme pour le mode Stop.

Notes de mise en œuvre

intc: voir la note de mise en œuvre sur *intc* dans le paragraphe B2.3.1.

Q: les réponses Q aux fonctions CAMAC ne sont pas sauvegardées. Puisque la réponse Q est utilisée par le module pour indiquer la fin du bloc, aucune autre information ne peut être fournie au programme appelant.

Restrictions sur les valeurs de paramètres

Dans les systèmes à multiples utilisateurs où les moyens du CAMAC sont partagés entre plusieurs utilisateurs indépendants, il peut être nécessaire de restreindre l'accès aux contrôleurs de châssis et à certains modules en n'admettant pas que les actions de contrôleur de châssis soient émises via CFUBC et en interdisant l'accès via CFUBC à certains autres modules utilisés pour la commande du système.

B2.3.4 *Transfert de bloc synchronisé par LAM*

Forme: CALL CFUBL *(f, ext, intc, cb).*

Fonction: ce sous-programme exécute une fonction simple CAMAC, *f*, à une adresse unique, avec l'emploi de la réponse Q et du signal LAM en mode ULS.

Notes de mise en œuvre

intc: voir la note de mise en œuvre sur *intc* dans le paragraphe B2.3.1.

A: les réponses Q aux fonctions CAMAC ne sont pas sauvegardées. Puisque la réponse Q est utilisée par le module pour indiquer la fin du bloc, aucune autre information ne peut être fournie au programme appelant.

Contrôle de LAM

Les modules conçus pour être utilisés dans ce mode doivent supprimer le LAM de synchronisation durant l'exécution d'une commande de lecture ou d'écriture.

Restrictions sur les valeurs de paramètres

Dans les systèmes à multiples utilisateurs où les moyens du CAMAC sont partagés entre plusieurs utilisateurs indépendants, il peut être nécessaire de restreindre l'accès aux contrôleurs de châssis et à certains modules en n'admettant pas que les actions de contrôleur soient émises via CFUBL et en interdisant l'accès via CFUBL à certains autres modules utilisés pour la commande du système.

B2.3.5 *Transfert de bloc en mode répétitif*

Forme: CALL CFUBR *(f, ext, intc, cb).*

Fonction: ce sous-programme exécute une fonction simple CAMAC, *f*, en mode répétitif.

Notes de mise en œuvre

intc: voir la note de mise en œuvre sur *intc* dans le paragraphe B2.3.1.

Q: les réponses Q aux fonctions CAMAC ne sont pas sauvegardées. Puisque la réponse Q est utilisée par le module pour indiquer la synchronisation des données, aucune autre information ne peut être fournie au programme appelant.

B2.3.3 *Controller-Synchronized Block Transfer*

Form: CALL CFUBC *(f, ext, intc, cb).*

Function: this subroutine executes a single CAMAC function, *f*, at a single address, with the usage of the Q response defined as for the Stop Mode.

Implementation Notes

intc: see implementation note on *intc* in Sub-clause B2.3.1.

Q: the Q responses to the CAMAC functions are not saved. Since the Q response is used by the module to indicate end of block, it can carry no other information to the calling program.

Restrictions on Parameter Values

In multiple-user systems where CAMAC facilities are shared among several independent users, it may be necessary to restrict access to crate controllers and certain modules by not allowing crate controller actions to be issued via CFUBC and by forbidding access via CFUBC to certain other modules used for system control.

B2.3.4 *LAM-Synchronized Block Transfer*

Form: CALL CFUBL *(f, ext, intc, cb).*

Function: this subroutine executes a single CAMAC function, *f*, at a single address, with the usage of the Q response and LAM signal for the ULS mode.

Implementation Notes

intc: see implementation note on *intc* in Sub-clause B2.3.1.

A: the Q responses to the CAMAC functions are not saved. Since the Q response is used by the module to indicate end of block, it can carry no other information to the calling program.

LAM Control

Modules designed to be used in this mode must clear the synchronizing LAM during the execution of a read or write command.

Restrictions on Parameter Values

In multiple-user systems where CAMAC facilities are shared among several independent users, it may be necessary to restrict access to crate controllers and certain modules by not allowing crate controller actions to be issued via CFUBL and by forbidding access via CFUBL to certain other modules used for system control.

B2.3.5 *Repeat-Mode Block Transfer*

Form: CALL CFUBR *(f, ext, intc, cb).*

Function: this subroutine executes a single CAMAC function, *f*, in the Repeat mode.

Implementation Notes

intc: see implementation note on *intc* in Sub-clause B2.3.1.

Q: the Q response to the CAMAC functions are not saved. Since the Q response is used by the module to indicate data synchronization, it can carry no other information to the calling program.

Intégrité du système

Un fonctionnement anormal ou une fonction erronée peuvent amener un module à ne jamais répondre Q = 1, provoquant ainsi une boucle sans fin. Pour éviter ce cas, on introduit dans le matériel ou le logiciel du contrôleur une limitation de temps ou de comptage de boucle.

Restrictions sur les valeurs de paramètres

Dans les systèmes à multiples utilisateurs où les moyens du CAMAC sont partagés entre plusieurs utilisateurs indépendants, il peut être nécessaire de restreindre l'accès aux contrôleurs de châssis et à certains modules en n'admettant pas que les actions de contrôleur de châssis soient émises via CFUBR et en interdisant l'accès via CFUBR à certains autres modules utilisés pour la commande du système.

B2.3.6 *Analyse de l'identificateur de LAM*

Forme: CALL CGLAM *(lam, b, c, n, m, inta).*

Fonction: CGLAM accepte un identificateur de LAM comme paramètre d'entrée et renvoie les valeurs des paramètres du matériel ou du système opérateur qui le définissent. Il exécute la transformation inverse de CDLAM.

B2.3.7 *Analyse de l'identificateur de registre*

Forme: CALL CGREG *(ext, b, c, n, a).*

Fonction: CGREG accepte un identificateur de registre, *ext,* comme paramètre d'entrée, l'analyse en ses composantes et les stocke dans *b, c, n* et *a.*

B3. Types de paramètres

On donne dans cette section les types de paramètres des sous-programmes. Se reporter à l'article 5 pour la signification de chaque paramètre.

B3.1 *Entiers simples*

Si un sous-programme renvoie une valeur dans un paramètre de ce type, elle doit être soit une variable entière, soit un élément de tableau d'entiers. Si aucune valeur n'est renvoyée, elle peut être n'importe quelle expression d'entiers. Les paramètres de cette classe sont:

a, b, c, ext, f, lam, m, n

B3.2 *Valeurs logiques simples*

Si un sous-programme renvoie une valeur dans un paramètre de ce type, elle doit être soit une variable logique, soit un élément de tableau logique. Si aucune valeur n'est renvoyée, elle peut être n'importe quelle expression logique. Les paramètres de cette classe sont:

l, q

B3.3 *Tableaux d'entiers*

Ces paramètres doivent être des tableaux d'entiers. Les paramètres de cette classe sont:

cb, exta, extb, fa, inta

B3.4 *Tableau logique*

Ce paramètre doit être un tableau logique. Le seul paramètre de cette classe est:

qa

System Integrity

A malfunction or erroneous operation may produce a situation in which a module will never respond with Q = 1, causing an unending loop. To guard against this case, a time-out or maximum repetition count should be implemented in the controller hardware or software.

Restrictions on Parameter Values

In multiple-user systems where CAMAC facilities are shared among several independent users, it may be necessary to restrict access to crate controllers and certain modules by not allowing crate controller actions to be issued via CFUBR and by forbidding access via CFUBR to certain other modules used for system control.

B2.3.6 *Analyze LAM Identifier*

Form: CALL CGLAM *(lam, b, c, n, m, inta).*

Function: CGLAM accepts a LAM identifier as input and returns the values of hardware or operating system parameters which define it. It performs the inverse transformation of CDLAM.

B2.3.7 *Analyze Register Identifier*

Form: CALL CGREG *(ext, b, c, n, a).*

Function: CGREG accepts a register identifier, *ext*, as input, analyzes it into its components, and stores them into *b*, *c*, *n*, and *a*.

B3. Parameter Types

The types of the subroutine parameters are given in this clause. See Clause 5 for the meaning of each parameter.

B3.1 *Single Integers*

If a subroutine returns a value in a parameter of this type, it must be either an integer variable or an integer array element. If no value is returned, it may be any integer expression. The parameters of this class are:

a, b, c, ext, f, lam, m, n

B3.2 *Single Logical Values*

If a subroutine returns a value in a parameter of this type, it must be either a logical variable or a logical array element. If no value is returned, it may be any logical expression. The parameters of this class are:

l, q

B3.3 *Integer Arrays*

These parameters must be integer arrays. The parameters of this class are:

cb, exta, extb, fa, inta

B3.4 *Logical Array*

This parameter must be a logical array. The only parameter of this class is:

qa

B3.5 *Mot de données CAMAC*

Ce paramètre doit être capable de stocker un mot de données CAMAC d'une longueur de 24 bits. Si la variable entière FORTRAN peut contenir 24 bits ou plus, le type pour le mot de données CAMAC est alors le même que pour un entier simple (voir le paragraphe B3.1). Si la longueur de la variable entière est plus grande que l'équivalent de 24 bits, le mot de données CAMAC est représenté comme un entier non signé avec le bit 24 représentant le poids le plus fort et le bit 1 représentant le plus faible. Si la variable entière FORTRAN ne peut pas contenir 24 bits, le paramètre doit être un tableau d'entiers suffisamment grand pour contenir 24 bits. Le mot de données CAMAC doit être représenté dans le tableau comme un entier de multiple précision avec la partie de poids le plus faible dans le dernier élément du tableau et la partie de poids le plus fort dans son premier élément. Le bit 24 du mot CAMAC est pris comme bit de plus grand poids et le bit 1 de plus petit poids. Le seul paramètre de cette classe est:

int

B3.6 *Tableau de données CAMAC*

Ce paramètre doit être un tableau d'éléments, chacun étant du même type que le mot de données CAMAC (voir le paragraphe B3.5). Le seul paramètre de cette classe est:

intc

B3.7 *Etiquette*

Elle ne peut être définie comme un type de paramètre dans les limites des normes typiques FORTRAN puisqu'elle indique une procédure prévue pour être exécutée en dehors du programme appelant. Elle doit être définie de façon appropriée pour chaque mise en œuvre. Le seul paramètre de cette classe est:

label

B3.5 *CAMAC Data Word*

This parameter must be capable of storing a CAMAC data word 24 bits in length. If the FORTRAN integer variable can contain 24 bits or more, then the type for CAMAC data word is the same as for a single integer (see Sub-clause B3.1). If the integer variable length is greater than the equivalent of 24 bits, the CAMAC data word is represented as an unsigned integer with bit 24 representing the highest order bit and bit 1 representing the lowest. If the FORTRAN integer variable cannot contain 24 bits, then the parameter must be an integer array sufficiently long to contain 24 bits. The CAMAC data word must be represented in the array as a multiple-precision integer with the lowest-order portion in the last array element and the highest-order portion in the first element. Bit 24 of the CAMAC word is taken to be the highest-order bit and bit 1 the lowest. The only parameter of this class is:

int

B3.6 *CAMAC Data Array*

This parameter must be an array of elements each of the same type as the CAMAC data word (see Sub-clause B3.5). The only parameter of this class is:

intc

B3.7 *Label*

This parameter cannot be defined or assigned a type within the bounds of strictly standard FORTRAN, since it labels a procedure which is intended to be executed out of sequence with respect to the calling program. It must be defined appropriately for each implementation. The only parameter of this class is:

label

ANNEXE C

SYMBOLES MNÉMONIQUES DU CODE DE FONCTION

Symbole	Valeur	Définition
RD1	0	Lecture du registre du groupe 1
RD2	1	Lecture du registre du groupe 2
RC1	2	Lecture et remise à zéro du registre du groupe 1
RCM	3	Lecture du complément du registre du groupe 1
TLM	8	Contrôle de LAM
CL1	9	Remise à zéro du registre du groupe 1
CLM	10	Remise à zéro de LAM
CL2	11	Remise à zéro du registre du groupe 2
WT1	16	Ecriture du registre du groupe 1
WT2	17	Ecriture du registre du groupe 2
SS1	18	Mise en place sélective du registre du groupe 1
SS2	19	Mise en place sélective du registre du groupe 2
SC1	21	Mise à zéro sélective du registre du groupe 1
SC2	23	Mise à zéro sélective du registre du groupe 2
DIS	24	Mise hors service
XEQ	25	Exécution
ENB	26	Mise en service
TST	27	Essai

APPENDIX C

FUNCTION-CODE MNEMONIC SYMBOLS

Symbol	Value	Definition
RD1	0	Read Group 1 Register
RD2	1	Read Group 2 Register
RC1	2	Read and Clear Group 1 Register
RCM	3	Read Complement of Group 1 Register
TLM	8	Test LAM
CL1	9	Clear Group 1 Register
CLM	10	Clear LAM
CL2	11	Clear Group 2 Register
WT1	16	Write Group 1 Register
WT2	17	Write Group 2 Register
SS1	18	Selective Set Group 1 Register
SS2	19	Selective Set Group 2 Register
SC1	21	Selective Clear Group 1 Register
SC2	23	Selective Clear Group 2 Register
DIS	24	Disable
XEQ	25	Execute
ENB	26	Enable
TST	27	Test

ICS 27.120

Typeset and printed by the IEC Central Office
GENEVA, SWITZERLAND