# NORME INTERNATIONALE

# INTERNATIONAL STANDARD

**CEI**
**IEC**
**60775**

Première édition
First edition
1983-01

## BASIC temps réel pour CAMAC

## Real-time BASIC for CAMAC

Numéro de référence
Reference number
CEI/IEC 60775: 1983

## 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**
**60775**

Première édition
First edition
1983-01

## BASIC temps réel pour CAMAC

## Real-time BASIC 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 R

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

# SOMMAIRE

# CONTENTS

COMMISSION ÉLECTROTECHNIQUE INTERNATIONALE

# BASIC TEMPS RÉEL POUR CAMAC

## PRÉAMBULE

1) Les décisions ou accords officiels de la C E I 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 C E I exprime le vœu que tous les Comités nationaux adoptent dans leurs règles nationales le texte de la recommandation de la C E I, dans la mesure où les conditions nationales le permettent. Toute divergence entre la recommandation de la C E I 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 C E I: Instrumentation nucléaire.

Un projet fut discuté lors de la réunion tenue à Tokyo en 1981. A la suite de cette réunion, un projet, document 45(Bureau Central)161, fut soumis à l'approbation des Comités nationaux suivant la Règle des Six Mois en mai 1982.

Les Comités nationaux des pays suivants se sont prononcés explicitement en faveur de la publication:

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

*Autre publication de la C E I citée dans la présente norme:*

Publication n° 516: Système modulaire d'instrumentation pour le traitement de l'information; système CAMAC.

INTERNATIONAL ELECTROTECHNICAL COMMISSION

# REAL-TIME BASIC FOR CAMAC

## FOREWORD

1) The formal decisions or agreements of the I E C 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 I E C expresses the wish that all National Committees should adopt the text of the I E C recommendation for their national rules in so far as national conditions will permit. Any divergence between the I E C recommendation and the corresponding national rules should, as far as possible, be clearly indicated in the latter.

## PREFACE

This standard has been prepared by I E C Technical Committee No. 45: Nuclear Instrumentation.

A draft was discussed at the meeting held in Tokyo in 1981. As a result of this meeting, a draft, Document 45(Central Office)161, was submitted to the National Committees for approval under the Six Months' Rule in May 1982.

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

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

*Other I E C publication quoted in this standard:*

Publication No. 516: A Modular Instrumentation System for Data Handling; CAMAC System.

# BASIC TEMPS RÉEL POUR CAMAC

## INTRODUCTION

Le BASIC temps réel pour CAMAC est le BASIC temps réel dans lequel les instructions des déclarations et du temps réel sont adaptées au matériel CAMAC. La norme BASIC temps réel définit des «déclarations de processus» et des «entrée-sortie de processus» de façon générale, indépendamment de toute interface particulière. Des parties spécifiques de la syntaxe et de la sémantique du langage sont laissées «ouvertes et définies» lors de la mise en œuvre de telle façon que pour un système matériel particulier cette mise en œuvre puisse déterminer ces zones de manière la plus appropriée. L'objet de cette norme est de fournir une norme de référence destinée à obtenir le maximum de compatibilité entre les différentes mises en œuvre de BASIC temps réel dans le système CAMAC.

On suppose que le lecteur est familiarisé avec le BASIC et le CAMAC.

## 1. Domaine d'application et objet

### 1.1 *Domaine d'application*

La présente norme s'applique aux systèmes CAMAC définis dans la Publication 516 de la C E I: Système modulaire d'instrumentation pour le traitement de l'information; système CAMAC. Rien dans la présente norme ne se substitue aux dispositions obligatoires de la Publication 516 de la C E I ni n'en étend la portée.

### 1.2 *Objet*

La norme générique pour le BASIC (désignée ici sous le nom de norme «BASIC temps réel») inclut pour le temps réel une extension qui fournit les entrée-sortie de processus des périphériques, la réponse aux événements asynchrones, la synchronisation et la communication entre activités simultanées, et la saisie des exceptions.

Cette norme BASIC temps réel est forcément générale et des parties spécifiques de la syntaxe et de la sémantique sont laissées disponibles afin de pouvoir être convenablement définies pour les processus particuliers des systèmes périphériques. L'objectif de la présente norme est de définir les syntaxes et sémantiques spécifiques applicables au CAMAC dans les régions laissées libres de la norme BASIC temps réel.

## 2. Possibilités du temps réel

Un programme temps réel est écrit sous forme d'activités simultanées qui peuvent dialoguer et se synchroniser de façon à atteindre complètement l'objectif de l'application.

Chaque activité possède des variables locales et communique avec son environnement à travers des «ports d'accès» qui sont de trois types:

1) Ports d'accès des messages qui transfèrent des données entre deux activités;
2) Ports d'accès des données partagées qui donnent accès aux données partagées qui existent en dehors des activités;
3) Ports d'accès des processus qui fournissent l'entrée-sortie CAMAC.

Egalement, des instructions sont définies pour établir un programme d'activités simultanées en temps réel et en réponse aux événements produits extérieurement et intérieurement.

# REAL-TIME BASIC FOR CAMAC

## INTRODUCTION

Real-time BASIC for CAMAC is Real-time BASIC in which the declarations and real-time statements are defined for use with CAMAC hardware. The Real-time BASIC standard defines "Process declarations" and "Process input-output" in a general way, independently of any particular interface hardware. Specific parts of the syntax and semantics of the language are left "implementation-defined" so that an implementation for a particular hardware system can define these areas in the most appropriate way. The purpose of this standard is to provide a reference, to achieve maximum compatibility between different implementations of Real-time BASIC for use with CAMAC.

It is assumed that the reader is familiar with BASIC and with CAMAC.

## 1. Scope and object

### 1.1 *Scope*

This document applies to CAMAC systems as defined in I E C Publication 516: A Modular Instrumentation System for Data Handling; CAMAC System. Nothing in this document shall supersede or extend the mandatory requirements of I E C Publication 516.

### 1.2 *Object*

The generic standard for BASIC (referred to herein as the "Real-time BASIC" standard) includes a real-time enhancement that provides input-output to process peripherals, response to asynchronous events, synchronization and communication between concurrent activities, and the handling of exceptions.

The Real-time BASIC standard is necessarily general, and specific parts of the syntax and semantics are left open to be defined appropriately for particular process peripheral systems. The purpose of this standard is to define specific syntax and semantics for use with CAMAC in areas left open in the Real-time BASIC standard.

## 2. Real-time capabilities

A real-time program is written as a number of concurrent activities which can communicate and synchronize as necessary to achieve the overall objective of the application.

Each activity has local variables, and communicates with its environment through "ports" which are of three types:

1) Message ports which transfer data between two activities;
2) Shared-data ports which allow access to shared data that exist outside the activities;
3) Process ports which provide CAMAC input-output.

Also, statements are defined to schedule concurrent activities in real-time and in response to externally and internally generated events.

## 3. Déclarations

Les instructions de déclaration sont utilisées pour définir les attributs des ports d'accès et la structure des données transférées à travers eux. Les déclarations doivent être placées en tête du programme avant toute instruction d'exécution. Elles s'appliquent à l'ensemble du programme temps réel contrairement aux instructions DIM qui sont locales pour une activité simultanée.

### 3.1 *Déclarations de structure de données*

Une structure de données est une liste de types de données, soit réels, soit entiers, ou chaîne de caractères. Elle est utilisée pour définir des listes appropriées de variables et de tableaux dans les instructions de transfert de données à travers les ports d'accès, et pour définir les unités d'accès aux données partagées. Par exemple, la déclaration d'une structure de données appelée STAT comprenant trois nombres réels, un tableau d'entiers et deux chaînes de caractères pourrait être:

200 STRUCTURE STAT: 3 OF REAL, INTEGER (100), 2 OF STRING

La syntaxe formelle des déclarations de structure des données est:

| | | |
|---|---|---|
| dec-structure-données | = | STRUCTURE nom-structure deux-points répétition? type (virgule répétition? type) * |
| nom-structure | = | lettre (lettre/chiffre) * |
| répétition | = | entier OF |
| entier | = | chiffre, chiffre * |
| type [1] | = | (REAL/INTEGER/STRING) dimensionnement? |
| dimensionnement | = | parenthèse-gauche limites parenthèse-droite |
| limites | = | entier (virgule entier)? |

### 3.2 *Déclarations CAMAC*

Les déclarations CAMAC sont les «déclarations de port d'accès de processus» dans la norme BASIC temps réel, dans le cas spécifique où le système périphérique de processus est CAMAC.

Exemples de déclarations CAMAC:

400 PROCESS INPUT WEIGHT «CAMAC (1, 3, 17, 0) (F2) (B10)»

410 PROCESS OUTPUT PANEL «CAMAC (,, 2, 4) (C4)»

420 PRODIM MPX (4)

421 PROCESS INPUT MPX (1) «CAMAC (,, 5, 0)»

422 PROCESS INPUT MPX (2) «CAMAC (,, 5, 1)»

423 PROCESS INPUT MPX (3) «CAMAC (,, 5, 2)»

424 PROCESS OUTIN MPX (4) «CAMAC (,, 7, 0) (F1)»

430 PROCESS EVENT FULL «CAMAC WEIGHT GL3»

La ligne 400 déclare un dispositif «entrée seulement» appelé WEIGHT qui se trouve à la Branche 1, Châssis 3, Station 17 et Sous-adresse 0. Sa donnée est lue par le code de fonction 2, elle est sous forme binaire en signe et taille avec une taille de 10 bits.

[1] La norme BASIC temps réel définit seulement deux types de données: numériques (décimal virgule-flottante) et chaîne de caractères. Pour de nombreuses applications CAMAC, le format numérique implique une mémoire démesurée et entraîne des calculs trop lents. En utilisation avec CAMAC, le «numérique» est subdivisé en deux types: «réel» employant un format interne décimal ou binaire et «entier» correspondant à des mots CAMAC de 24 bits. Lorsqu'ils sont utilisés en opérations numériques, les entiers sont traités comme des nombres de 24 bits en complément à deux.

## 3. Declarations

Declaration statements are used to define the attributes of ports and the structure of data transferred through them. The declarations shall be at the head of the program before any executable statements. They apply to the whole real-time program, unlike DIM statements which are local to a concurrent activity.

### 3.1 *Data structure declarations*

A data structure is a list of the data types real, integer and string. It is used to define valid lists of variables and arrays in statements transferring data through ports, and to define units of access to shared data. For example, the declaration for a data structure called STAT comprising three real numbers, an integer array and two strings could be:

```
200 STRUCTURE STAT: 3 OF REAL, INTEGER (100), 2 OF STRING
```

The formal syntax of data structure declarations is:

| | | |
|---|---|---|
| data-structure-dec | = | STRUCTURE structure-name colon repeat-count? type (comma repeat-count? type) * |
| structure-name | = | letter (letter/digit) * |
| repeat-count | = | integer OF |
| integer | = | digit digit * |
| type [1] | = | (REAL/INTEGER/STRING) dimensioning? |
| dimensioning | = | left-parenthesis bounds right-parenthesis |
| bounds | = | integer (comma integer)? |

### 3.2 *CAMAC declarations*

CAMAC declarations are the "process port declarations" in the Real-time BASIC standard, in the specific case when the process peripheral system is CAMAC.

Examples of CAMAC declarations are:

```
400 PROCESS INPUT WEIGHT "CAMAC (1, 3, 17, 0) (F2) (B10)"
410 PROCESS OUTPUT PANEL "CAMAC (,, 2, 4) (C4)"
420 PRODIM MPX (4)
421 PROCESS INPUT MPX (1) "CAMAC (,, 5, 0)"
422 PROCESS INPUT MPX (2) "CAMAC (,, 5, 1)"
423 PROCESS INPUT MPX (3) "CAMAC (,, 5, 2)"
424 PROCESS OUTIN MPX (4) "CAMAC (,, 7, 0) (F1)"
430 PROCESS EVENT FULL "CAMAC WEIGHT GL3"
```

Line 400 declares an "input only" device called WEIGHT which is at Branch 1, Crate 3, Station 17 and Sub-address 0. It is read by function code 2, and its data is in the form binary sign and magnitude with 10 magnitude bits.

[1] Real-time BASIC standard defines two data types only: numeric (decimal floating-point) and string. For many CAMAC applications the numeric format would use excessive memory and would cause calculations to be too slow. For use with CAMAC, "numeric" is subdivided into two types: "real" using decimal or binary internal format and "integer" corresponding to CAMAC 24-bit words. When used in numeric operations, integers are treated as 24-bit two's complement numbers.

Dans le second exemple, la ligne 410 déclare un dispositif «sortie seulement» appelé PANEL qui se trouve à la Station 2 et est accessible à la Sous-Adresse 4 (les numéros de Branche et de Châssis manquent — cette déclaration peut convenir pour des systèmes à châssis unique). Le code de fonction de défaut 16 est employé pour l'écriture et les données sont interprétées comme un nombre BCD de quatre chiffres.

Les lignes 420 à 424 indiquent la déclaration d'un tableau de ports d'accès CAMAC. La ligne 420 est l'instruction de dimension et les lignes 421 à 424 déclarent les attributs de chaque élément du tableau. Dans l'exemple, les éléments MPX(1), MPX(2) et MPX(3) sont les trois premières Sous-Adresses dans la Station 5, tandis que MPX(4) est la Sous-Adresse 0 dans la Station 7.

Le dernier exemple sur la ligne 430 déclare un LAM CAMAC provenant du dispositif WEIGHT et connecté à GL3. Le LAM a le nom FULL et il est employé comme un «événement» dans les instructions WAIT (voir article 4).

La partie de la déclaration à l'intérieur des guillemets est la «mise en œuvre définie» dans la norme et est définie ici pour une utilisation avec CAMAC.

Les définitions formelles sont les suivantes:

| | | |
|---|---|---|
| instruction-dim-processus | = | PRODIM dec-tableau-processus (virgule dec-tableau-processus) * |
| dec-tableau-processus | = | nom-tableau-processus dimensionnement |
| dec-port-d'accès-processus | = | PROCESS (clause-processus/clause-événement) accès-information |
| clause-processus | = | notification-es nom-port-d'accès-camac limites? (OF nom-structure)? (TIMEOUT rep-numérique)? |
| notification-es | = | INPUT/OUTPUT/OUTIN |
| nom-port-d'accès-camac | = | lettre (lettre/chiffre) * |
| rep-numérique | = | mantisse exposant? |
| mantisse | = | entier point?/entier? fraction |
| fraction | = | point entier |
| exposant | = | E (signe-plus/signe-moins)? entier |
| clause-événement | = | EVENT nom-lam |
| nom-lam | = | lettre (lettre/chiffre) * |
| information-accès [1] | = | guillemet CAMAC (dec-registre/dec-lam) guillemet |
| dec-registre [1] | = | bcna accès? format? |
| bcna [1] | = | parenthèse-gauche entier? virgule entier? virgule entier virgule entier parenthèse-droite |
| accès [1] | = | parenthèse-gauche fonction-accès (virgule fonction-accès) * parenthèse-droite |
| fonction-accès [1] | = | (F entier)/NX/nom-canal |
| nom-canal [1] | = | mise-en-œuvre-définie |
| format [1] | = | parenthèse-gauche (B/C/I) entier parenthèse-droite |
| dec-lam [1] | = | nom-port-d'accès-camac GL entier ((P/A)entier)? |

[1] «mise en œuvre-définie» dans la norme, précisée ici pour utilisation avec CAMAC.

The second example in line 410 declares an "output only" device called PANEL which is in Station 2 and is accessed at Sub-address 4 (the Branch and Crate numbers are defaulted—this declaration would be appropriate for single-crate systems). The default function code 16 is used for writing, and the data is interpreted as a BCD number with four digits.

Lines 420 to 424 show the declaration of an array of CAMAC ports. Line 420 is the dimension statement, and lines 421 to 424 declare the attributes of each element of the array. In the example the elements MPX(1), MPX(2) and MPX(3) are the first three sub-addresses in Station 5, while MPX(4) is Sub-address 0 in Station 7.

The final example on line 430 declares a CAMAC LAM emanating from the device WEIGHT, and connected to GL3. The LAM is given the name FULL and it is used as an "event" in WAIT statements (see Clause 4).

The part of a declaration inside quotation marks is "implementation-defined" in the standard, and is defined here for use with CAMAC.

The formal definitions are as follows:

| | | |
|---|---|---|
| process-dim-statement | = | PRODIM process-array-dec (comma process-array-dec) * |
| process-array-dec | = | process-array-name dimensioning |
| process-port-dec | = | PROCESS (process-clause/event-clause) access-information |
| process-clause | = | io-qualifier camac-port-name bounds? (OF structure-name)? (TIMEOUT numeric-rep)? |
| io-qualifier | = | INPUT/OUTPUT/OUTIN |
| camac-port-name | = | letter (letter/digit) * |
| numeric-rep | = | significand xrad? |
| significand | = | integer full-stop?/integer? fraction |
| fraction | = | full-stop integer |
| xrad | = | E (plus-sign/minus-sign)? integer |
| event-clause | = | EVENT lam-name |
| lam-name | = | letter (letter/digit) * |
| access-information [1] | = | quotation-mark CAMAC (register-dec/lam-dec) quotation-mark |
| register-dec | = | bcna access? format? |
| bcna [1] | = | left-parenthesis integer? comma integer? comma integer comma integer right-parenthesis |
| access [1] | = | left-parenthesis access-function (comma access-function) * right-parenthesis |
| access-function [1] | = | (F integer)/NX/ch-name |
| ch-name [1] | = | implementation-defined |
| format [1] | = | left-parenthesis (B/C/I) integer right-parenthesis |
| lam-dec [1] | = | camac-port-name GL integer ((P/A) integer)? |

[1] "implementation-defined" in the standard, defined here for use with CAMAC.

Le mot clé CAMAC est utilisé pour distinguer les déclarations CAMAC dans les systèmes à plusieurs types de périphériques.

Dans l'espace «bcna», les quatre valeurs représentent B, C, N et A, c'est-à-dire respectivement les éléments Branche, Châssis, Station et Sous-Adresse de l'adresse CAMAC. Si les deux derniers paramètres sont zéro, l'adresse représente un contrôleur de châssis destiné aux actions dans le châssis (voir paragraphe 5.3); par exemple, (1, 3, 0, 0) représente le Châssis 3 sur la Branche 1.

Deux codes de fonction au plus peuvent être spécifiés dans le domaine de l'«accès», un pour chaque groupe «lecture» et «écriture». Noter que le code de fonction est donné dans la déclaration — il n'est pas un paramètre de l'instruction d'entrée ou de sortie. Les codes de défaut pour la lecture et l'écriture sont F0 et F16. Les mots clés INPUT, OUTPUT ou OUTIN indiquent que le dispositif fournit une entrée seulement, accepte une sortie seulement ou comporte à la fois entrée et sortie.

Si un système comprend différentes sortes de modules avec ou sans possibilité d'erreur X, le branchement de l'erreur X doit être mis hors de service, soit en permanence, soit lorsqu'un module sans X est atteint. Le mot clé NX confère directement la possibilité d'être associé avec le périphérique plutôt que de la voir explicitement formulée dans chaque instruction d'entrée ou de sortie. Le nom-canal est un paramètre du système identifiant un canal de transfert de bloc de données. Il spécifie le mode de fonctionnement et, dans le cas d'un canal synchronisé sur le LAM, le LAM qui assure la synchronisation. Le canal de données par défaut est un transfert d'entrée-sortie programmé. Une méthode possible de mise en œuvre est d'utiliser une «liste de description d'environnement» fournie lors de la mise en œuvre et contenant les définitions des canaux et autres possibilités fixées qui les concernent.

Pour l'entrée-sortie de processus où la référence interne est numérique, le format précise l'interprétation numérique des données. B représente un signe et une taille binaires, avec l'entier indiquant le nombre des bits de la taille; C représente un nombre décimal codé en binaire (code 8-4-2-1) avec l'entier indiquant le nombre de chiffres décimaux dans le mot CAMAC et I représente un entier en complément à deux (l'«entier» donne le nombre total de bits, signe compris). Si aucun format n'est spécifié, l'interprétation par défaut est un entier de 24 bits.

Certains périphériques CAMAC, tels que les multiplexeurs d'exploration, ont des structures complexes de données. L'inclusion du nom-structure dans la clause-processus permet à l'analyseur du langage de vérifier la validité des listes-de-variables dans les instructions d'entrée-sortie se rapportant à de tels périphériques.

Dans le cas de dec-lam, le nom-port-d'accès-camac est le périphérique CAMAC qui est la source du LAM et l'entier qui suit le mot clé GL est la ligne à laquelle il est connecté. Si le LAM est obtenu à une Sous-adresse autre que celle qui est donnée dans la déclaration CAMAC, la Sous-adresse du LAM est donnée par un entier qui suit le mot clé A. Si le LAM est obtenu à une position de bit dans les registres A12, A13 et A14 du groupe 2, la position de bit est donnée par l'entier qui suit le mot clé P. Les positions de bit sont numérotées de 1 à 24 en commençant par le bit de poids faible.

The keyword CAMAC is used to distinguish CAMAC declarations in systems with more than one type of process peripheral.

In the "bcna" field, the four values represent B, C, N and A, i.e. the Branch, Crate, Station and Sub-address parts of the CAMAC address respectively. If the last two parameters are zero, the address represents a crate controller for use in crate actions (see Sub-clause 5.3); for example, (1, 3, 0, 0) represents Crate 3 on Branch 1.

Up to two function codes can be specified in the "access" field, one from each group "read" and "write". Note that the function code is given in the declaration—it is not a parameter of the input or output statement. The default codes for read and write are F0 and F16. The keywords INPUT, OUTPUT or OUTIN indicate that the device provides input only, accepts output only, or supports both input and output.

If a system contains a mixture of modules with or without the X-error facility, either the X-error trap must be permanently disabled, or it must be disabled when a module without X is accessed. The keyword NX allows this attribute to be associated with the device rather than stated explicitly in each input or output statement. The ch-name is a system parameter identifying a block-transfer data-channel. It specifies the mode of operation and, in the case of a LAM synchronized channel, the synchronizing LAM. The default data channel is programmed input-output. A possible method of implementation is to use an implementation supplied "environment description file" containing the definitions of channels and other relatively fixed facilities.

For process input-output where the internal reference is numeric, the format field specifies the numeric interpretation of the data. B represents binary sign and magnitude, with the integer indicating the number of magnitude bits; C represents binary coded decimal (8-4-2-1 code), with the integer indicating the number of decimal digits in the CAMAC word; and I represents two's complement integer (the "integer" gives the total number of bits, including sign). If no format is specified, the default interpretation is a 24-bit integer.

Some CAMAC devices, such as scanning multiplexers, have complex data structures. The inclusion of a structure-name in the process-clause allows the language processor to check the validity of variable-lists in input-output statements referring to such devices.

In the case of a lam-declaration, the camac-port-name is the CAMAC device that is the source of the LAM, and the integer following the keyword GL is the Graded-LAM line to which it is connected. If the LAM is accessed at a Sub-address other than that given in the CAMAC declaration, the LAM Sub-address is given by an integer following the keyword A. If the LAM is accessed at a bit position in the group 2 registers A12, A13 and A14, the bit position is given by the integer following the keyword P. Bit positions are numbered from 1 to 24 starting with the least significant.

3.3 *Modification dynamique de l'adresse*

Quelques applications demandent de pouvoir réassigner dynamiquement un élément de l'adresse d'un périphérique CAMAC. Un exemple est la demande, dans un système de haute fiabilité, de démarrage d'un périphérique entrée-sortie en attente par une commande au clavier. Pour ce faire, huit fonctions sont spécialement définies pour interroger et modifier les constituants d'une adresse CAMAC. Noter que ces fonctions sont spécifiques à l'emploi du CAMAC et qu'il n'y a rien d'équivalent dans la norme BASIC temps réel.

Les fonctions d'examen de l'adresse sont BEX, CEX, NEX et AEX qui retournent les éléments de l'adresse, respectivement B, C, N et A d'un périphérique CAMAC, par exemple:

```
520 LET G = AEX(MPX(2))
```

chargerait la variable G avec le numéro de la Station correspondant au périphérique 2 dans le tableau des ports d'accès CAMAC MPX du processus (ce serait la valeur 1 selon la déclaration à la ligne 422).

Les fonctions de modification d'adresse sont BMY, CMY, NMY et AMY qui modifient les parties de l'adresse B, C, N et A d'un périphérique CAMAC. Elles sont utilisées pour les instructions d'affectation, par exemple:

```
530 LET MPX(2) = NMY(4)
```

changerait le numéro de Station utilisé pour atteindre l'objet MPX(2), de la valeur 5 (déclaré à l'instruction 422) à la valeur 4. Il est à noter que ce sont des fonctions spéciales qui ne doivent être utilisées que pour assigner des valeurs aux éléments de l'adresse d'un périphérique CAMAC. On notera également que c'est la seule instruction dans laquelle l'objet du processus peut apparaître sur le côté gauche d'une affectation.

## 4. Activités parallèles

Une activité parallèle est définie par les instructions BASIC entre les mots clés PARACT et END PARACT. La ligne PARACT donne le nom de l'activité parallèle à utiliser dans les instructions d'ordonnancement, et une valeur d'«urgence». Un exemple d'activité parallèle serait:

```
550 PARACT WORK URGENCY 3
555   START RIG1
560   SIGNAL E6
565   WAIT TIME 17*60*60*
570   PRINT «TIME TO GO HOME»
575 END PARACT
```

L'activité parallèle s'appelle WORK et a l'urgence 3. La traduction de la valeur de l'urgence est définie lors de la mise en œuvre — elle pourrait par exemple représenter une priorité relative ou une valeur temporelle limite — mais quelle que soit l'interprétation, une valeur plus petite indique effectivement une plus grande priorité.

Une activité parallèle débute par l'instruction START. Elle peut être suspendue par une instruction WAIT et arrêtée par une instruction PARSTOP. Une instruction PARSTOP termine l'activité parallèle dans laquelle elle est inscrite, tandis que l'instruction STOP termine le programme temps réel tout entier. Si l'exécution arrive à la ligne END PARACT, l'effet est identique à celui de l'instruction PARSTOP.

3.3 *Dynamic address modification*

Some applications require the ability to re-assign dynamically the address part of a CAMAC device. An example is the requirement in a high-integrity system to activate a stand-by input-output device by a keyboard command. For this purpose eight functions are defined specifically to examine and modify the components of a CAMAC address. Note that these functions are specifically for use with CAMAC, they have no equivalent in the Real-time BASIC standard.

The address examine functions are BEX, CEX, NEX and AEX, which return the address parts B, C, N and A respectively of a CAMAC device, for example:

```
520 LET G = AEX(MPX(2))
```

would load the variable G with the Station number corresponding to device 2 in the vector of CAMAC devices MPX (this would be the value 1 following the declaration in line 422).

The address modify functions are BMY, CMY, NMY and AMY, which modify the address parts B, C, N and A of a CAMAC device. They are used in assignment statements, for example:

```
530 LET MPX(2) = NMY(4)
```

would change the Station number used to access the process object MPX(2), from the value 5 (declared in statement 422) to the value 4. Note that these are special functions that can be used only to assign values to the address part of a CAMAC device. Note also that this is the only statement in which a process object can appear on the left-hand side of an assignment.

## 4. Parallel activities

A parallel activity is defined by the BASIC statements between the keywords PARACT and END PARACT. The PARACT line gives the name of the parallel activity for use in scheduling statements, and an "urgency" value. An example of a parallel activity could be:

```
550 PARACT WORK URGENCY 3
555   START RIG1
560   SIGNAL E6
565   WAIT TIME 17*60*60*
570   PRINT "TIME TO GO HOME"
575 END PARACT
```

The parallel activity is called WORK and it has urgency 3. The interpretation of the urgency value is implementation-defined—it could for example represent a relative priority or a deadline—but whatever the interpretation a lower value effectively indicates a higher priority.

A parallel activity is started by a START statement. It can be suspended by a WAIT statement, and stopped by a PARSTOP statement. A PARSTOP statement terminates the parallel activity in which it is executed, while a STOP statement terminates the whole real-time program. If control reaches the END PARACT line the effect is the same as a PARSTOP statement.

Les définitions formelles sont les suivantes:

| | | |
|---|---|---|
| instruction-ordonnancement | = | instruction-départ / instruction-attente / instruction-signal/instruction-arrêt-activité-parallèle |
| instruction-départ | = | START nom-activité |
| nom-activité | = | lettre (lettre/chiffre) * |
| instruction-attente | = | WAIT (intervalle-attente/temps-attente/événement-attente) |
| intervalle-attente | = | DELAY expression-temps |
| expression-temps | = | expression-numérique/expression-chaîne-de-caractères |
| temps-attente | = | TIME expression-temps |
| événement-attente | = | EVENT (nom-événement/nom-lam) |
| nom-événement | = | lettre (lettre-chiffre) * |
| instruction-arrêt-activité-parallèle | = | PARSTOP |
| instruction-signal | = | SIGNAL nom-événement |

La valeur de l'expression-numérique dans une expression-temps est le nombre de secondes passées après l'heure de minuit qui précède. L'expression chaîne de caractères dans une expression-temps est une autre façon d'exprimer le temps sous forme d'heures: minutes: secondes dans un intervalle de 24 heures.

Un nom-événement est le nom d'un «événement logiciel». Un événement logiciel est implicitement déclaré par sa présence dans une instruction SIGNAL. Un événement logiciel est automatiquement annulé après sa prise en compte par une instruction WAIT EVENT.

## 5. Entrée-sortie CAMAC

### 5.1 *Transferts simples de données*

Les transferts simples de données depuis et vers le système périphérique du processus utilisent les instructions IN FROM et OUT TO comme suit:

IN FROM nom-port-d'accès-camac TO variable

OUT TO nom-port-d'accès-camac FROM expression

Par exemple, d'après la déclaration de la ligne 400, l'instruction:

600 IN FROM WEIGHT TO G

on lirait les données de la Branche 1, Châssis 3, Station 17, Sous-adresse 0, convertirait 10 bits avec le signe en bit 11 en format virgule flottante et stockerait le résultat dans la variable G.

Les définitions formelles sont les suivantes:

| | | |
|---|---|---|
| instruction-es-processus | = | instruction-entrée/instruction-sortie |
| instruction-entrée | = | IN FROM nom-port-d'accès-camac TO structure-entrée |
| structure-entrée | = | élément-structure-entrée (virgule élément-structure-entrée) * |
| élément-structure-entrée | = | variable/tableau-formel |
| instruction-sortie | = | OUT TO nom-port-d'accès-camac FROM structure-sortie |
| structure-sortie | = | élément-structure-sortie (virgule élément-structure-sortie) * |
| élément-structure-sortie | = | expression/tableau-formel |
| tableau-formel | = | tableau parenthèse-gauche virgule? parenthèse-droite |

The formal definitions are as follows:

| | | |
|---|---|---|
| scheduling-statement | = | start-statement/wait-statement/signal-statement/parstop-statement |
| start-statement | = | START activity-name |
| activity-name | = | letter (letter/digit) * |
| wait-statement | = | WAIT (wait-interval/wait-time/wait-event) |
| wait-interval | = | DELAY time-expression |
| time-expression | = | numeric-expression/string-expression |
| wait-time | = | TIME time-expression |
| wait-event | = | EVENT (event-name/lam-name) |
| event-name | = | letter (letter-digit) * |
| parstop-statement | = | PARSTOP |
| signal-statement | = | SIGNAL event-name |

The value of the numeric-expression in a time-expression is the number of seconds past the previous midnight. The string expression in a time-expression is an alternative way of expressing time in the form of hours: minutes: seconds on a 24-hour clock.

An event-name is the name of a "software event". A software event is declared implicitly by its occurrence in a SIGNAL statement. A software event is cleared automatically when it is "consumed" by a WAIT EVENT statement.

## 5. CAMAC input-output

### 5.1 *Simple data transfers*

Simple data transfers from and to the process peripheral system use IN FROM and OUT TO statements as follows:

IN FROM camac-port-name TO variable

OUT TO camac-port-name FROM expression

For example, assuming the declaration in line 400, the statement:

600 IN FROM WEIGHT TO G

would read the data from Branch 1, Crate 3, Station 17, Sub-address 0, convert 10 bits with the sign in bit 11 to floating-point form, and store the result in the variable G.

The formal definitions are as follows:

| | | |
|---|---|---|
| process-io-statement | = | in-statement/out-statement |
| in-statement | = | IN FROM camac-port-name TO in-structure |
| in-structure | = | in-structure-element (comma in-structure-element) * |
| in-structure-element | = | variable/formal-array |
| out-statement | = | OUT TO camac-port-name FROM out-structure |
| out-structure | = | out-structure-element (comma out-structure-element) * |
| out-structure-element | = | expression/formal-array |
| formal-array | = | array left-parenthesis comma? right-parenthesis |

5.2 *Transferts de bloc*

Les transferts de bloc peuvent être définis par utilisation d'un tableau formel et d'une variable numérique comme une structure-entrée ou une structure-sortie. Le tableau formel contient le bloc de données et la variable numérique contiendra, après que l'instruction de transfert de bloc sera terminée, le nombre de transferts réellement effectués. Celui-ci peut être moindre que le nombre d'éléments dans le tableau si le module signale «fin de bloc» en mettant Q = 0, ou si une exception survient. Le nombre maximal de transferts sera égal au nombre d'éléments dans le tableau. Les lignes suivantes du programme illustrent un transfert de bloc en entrée:

```
700 PROCESS INPUT M «CAMAC (,, 5, 0) (F0, CHAN3)»
710 DIM D(100)
720 IN FROM M TO D( ), C
730 IF C<>100 THEN 790
      —
      —
      —
790 PRINT C; «TRANSFERS ACCOMPLISHED»
      —
      —
```

Il est à noter qu'un tel transfert de bloc n'est pas formellement défini dans la norme BASIC temps réel. Un transfert de bloc implicite survient lorsqu'un tableau-formel apparaît, dans une structure-entrée ou dans une structure-sortie, mais aucun mécanisme signalant la fin de bloc n'est défini pour l'objet-processus, pas plus que pour mettre à la disposition du programme le nombre de transferts effectués.

5.3 *Ordres de contrôle et de commande CAMAC*

Ces ordres sont spécifiques au CAMAC et il n'y a pas d'instructions équivalentes dans la norme BASIC temps réel. Ils concernent les fonctions CAMAC «opérer» qui se rapportent à l'état des périphériques mais ne peuvent transférer des données sur les lignes de données. Ils fonctionnent au niveau du matériel dans le système CAMAC. L'ordre à exécuter est spécifié dans l'instruction. Une instruction d'ordre de contrôle et de commande présente la forme suivante:

CONTROL nom-port-d'accès-camac ordre-contrôle [1]

où:

| | | |
|---|---|---|
| ordre-contrôle | = | (F code-op)/op-module/op-châssis |
| op-module | = | ENB/DIS/CL1/CL2 |
| op-châssis | = | CZ/ENCD/DISCD/CC/CLRCI/SETCI |

Le code-op est une constante décimale représentant un des groupes «opérer» des codes de fonction. Un op-châssis est un ordre de contrôle et de commande adressé à un contrôleur de châssis, c'est-à-dire un objet-processus pour lequel les paramètres A et N sont zéro (voir paragraphe 3.2).

Un exemple d'ordre de contrôle et de commande est:

```
800 CONTROL MPX(4) ENB
```

[1] Pour cette version française, et uniquement pour les instructions, le mot «contrôle» englobe, comme en anglais, le sens de contrôle et commande.

5.2 *Block transfers*

Block transfers can be specified by using a formal array and a numeric variable as an in-structure or an out-structure. The formal array contains the block of data and the numeric variable contains, after the block transfer statement has terminated, the number of transfers actually achieved. This may be fewer than the number of elements in the array if the module signals "end of block" by setting Q = 0, or if an exception occurs. The maximum number of transfers will be equal to the number of elements in the array. The following lines of program illustrate an input block transfer:

```
700 PROCESS INPUT M "CAMAC (,, 5, 0) (F0, CHAN3)"
710 DIM D(100)
720 IN FROM M TO D( ), C
730 IF C<>100 THEN 790
      .
      .
      .
790 PRINT C; "TRANSFERS ACCOMPLISHED"
      .
      .
```

Note that block transfers are not formally defined in the Real-time BASIC standard. An implicit block transfer occurs when a formal-array appears in an in-structure or an out-structure, but no mechanism is defined for the process-object to signal end-of-block or for the number of transfers achieved to be available to the program.

5.3 *CAMAC control actions*

These actions are specific to CAMAC and no equivalent statements are defined in the Real-time BASIC standard. They concern the CAMAC "operate" functions, which refer to the status of devices but do not transfer data over the data lines. They operate at the hardware level in the CAMAC system. The action to be executed is specified in the statement. A control action statement has the following form:

CONTROL camac-port-name control-action

where:

| | | |
|---|---|---|
| control-action | = | (F op-code)/module-op/crate-op |
| module-op | = | ENB/DIS/CL1/CL2 |
| crate-op | = | CZ/ENCD/DISCD/CC/CLRCI/SETCI |

The op-code is a decimal constant representing one of the "operate" group of function codes. A crate-op is a control action addressed to a crate controller, i.e. a process-object for which the A and N parameters are zero (see Sub-clause 3.2).

An example of a control action is:

```
800 CONTROL MPX(4) ENB
```

La signification des différents mots clés est la suivante:

ENB = code de fonction 26
DIS = code de fonction 24
CL1 = code de fonction 9 (remise à zéro du registre du groupe 1)
CL2 = code de fonction 11 (remise à zéro du registre du groupe 2)
CZ = active le bus «Z» du châssis
ENCD = autorise les demandes dans le châssis
DISCD = interdit les demandes dans le châssis
CC = active le bus «remise à zéro» du châssis
CLRCI = supprime l'«inhibition-châssis»
SETCI = établit l'«inhibition-châssis»

## 6. Signaux CAMAC «Q» et «X»

Les valeurs de Q et X en retour de la dernière opération CAMAC après chaque exécution sont stockées dans les variables QCAM et XCAM. Ce sont des variables d'état du programme créées localement pour chaque activité simultanée. Elles peuvent être testées par les instructions IF, par exemple:

```
800 IF QCAM = 0 THEN 900
```

## 7. Saisie de LAM CAMAC

Un LAM CAMAC est saisi au niveau du BASIC comme un nom-lam. L'instruction 430, au paragraphe 3.2, est un exemple de déclaration-lam. On doit considérer deux aspects dans la saisie de LAM: comment l'autoriser, le mettre hors service, tester ses états par des ordres de contrôle, et comment le programme peut utiliser son apparition.

### 7.1 *Ordres de contrôle et de commande de LAM*

Les ordres de contrôle et de commande de LAM fonctionnent au niveau du matériel dans un système CAMAC. L'adresse LAM est fournie par le périphérique CAMAC associé à l'événement dans la déclaration.

La syntaxe d'un ordre de contrôle et de commande de LAM est la suivante:

```
CONTROL nom-lam ordre-lam
```

où:

ordre-lam = ENL/DISL/TEST/CLRL/MENL/MDISL/MTEST/MCLRL

La signification des différents mots clés est la suivante:

ENL : autorise LAM, soit par F26 à une sous-adresse, soit par mise en place du bit approprié dans le registre masque A13 du groupe 2 conformément à la déclaration.

DISL : met hors service LAM, soit par F24 à une sous-adresse, soit par annulation du bit approprié dans le registre masque A13 du groupe 2 conformément à la déclaration.

TEST : demande le test de LAM, soit par F8 dans une sous-adresse, soit par test de la position du bit dans le registre A14 du groupe 2. Ce test répond QCAM = 1 si le LAM est positionné, QCAM = 0 s'il est absent.

The meaning of the various keywords is as follows:

ENB = function code 26
DIS = function code 24
CL1 = function code 9 (clear group 1 register)
CL2 = function code 11 (clear group 2 register)
CZ = activate the crate "Z" bus
ENCD = enable crate demands
DISCD = disable crate demands
CC = activate the crate "clear" bus
CLRCI = remove "crate inhibit"
SETCI = set "crate inhibit"

## 6. The CAMAC "Q" and "X" signals

The values of Q and X returned for the last CAMAC operation in each activity are stored in the variables QCAM and XCAM. These are program status variables of which a local pair are created for each activity. They can be tested with IF statements, for example:

800 IF QCAM = 0 THEN 900

## 7. CAMAC LAM handling

A CAMAC LAM is handled at the BASIC level as a lam-name. Statement 430 in Sub-clause 3.2 is an example of a lam-declaration. Two aspects of LAM handling shall be considered: how to enable it, disable it and test its status by control actions, and how the program makes use of its occurrence.

### 7.1 *LAM control actions*

LAM CONTROL actions operate at the hardware level in the CAMAC system. The LAM address is supplied by the CAMAC device associated with the event in the declaration.

The syntax of a LAM CONTROL action is as follows:

CONTROL lam-name lam-action

where:

lam-action = ENL/DISL/TEST/CLRL/MENL/MDISL/MTEST/MCLRL

The meaning of the various keywords is as follows:

ENL : enable LAM, either F26 at a sub-address or set the appropriate bit in the group 2 mask register A13, according to the declaration.

DISL : disable LAM, either F24 at a sub-address or clear the appropriate bit in the group 2 mask register A13, according to the declaration.

TEST : test LAM request, either F8 at a sub-address or test the bit position in the group 2 register A14. This test leaves QCAM = 1 if the LAM is set, QCAM = 0 if clear.

CLRL : annule LAM, soit par F10 à une sous-adresse, soit par suppression du bit dans le registre A12 du groupe 2.
MENL : met en service le LAM d'un module en utilisant la sous-adresse pour contrôle général (voir paragraphe 2.3).
MDISL : met hors service le LAM d'un module, contrôle général.
MTEST : test du LAM dans le module, test général.
MCLRL: annule tous les LAM dans le module.

Il est à noter que seuls les modules disposant totalement du contrôle et commande de LAM à une sous-adresse peuvent répondre aux ordres de contrôle et commande MENL, MDISL, MTEST et MCLRL.

Noter également que ENL et DISL sont des instructions CAMAC équivalant aux instructions générales CONNECT et DISCONNECT définies dans la norme BASIC temps réel.

Exemple d'un ordre LAM CONTROL:

800 CONTROL FULL ENL

qui met en service le LAM dans la Station 17, la Sous-adresse 0 selon la déclaration dans la ligne 430, paragraphe 3.2.

7.2 *Gestion de LAM*

Un dispositif d'interruption du programme n'est pas fourni au niveau du BASIC. Un sous-programme de gestion du LAM est mis en place en tant qu'activité parallèle avec une instruction-attente désignant le LAM. Lorsque le LAM apparaît, l'activité se déroule à partir de l'instruction d'attente qui ensuite annule le LAM automatiquement. L'activité est normalement programmée pour revenir à l'instruction d'attente après la gestion de LAM.

7.3 *Modification dynamique de GL*

Les systèmes utilisant un codeur de LAM programmable doivent être capables de modifier le numéro du GL associé à un événement. La fonction GMY est utilisée à cette fin:

LET nom-lam = GMY parenthèse-gauche expression-numérique parenthèse-droite

par exemple:

940 LET FULL = GMY(7)

Ceci est une instruction spéciale d'affectation semblable aux affectations d'adresse CAMAC définies à l'article 3. C'est l'unique instruction dans laquelle un nom-événement peut apparaître sur le côté gauche de l'affectation. La fonction GMY ne peut être utilisée que dans cette instruction.

## 8. Transmission de message

Les instructions-envoyer et les instructions-recevoir sont utilisées pour transmettre les données sur des circuits-messages entre les activités-simultanées. Les circuits-messages sont établis à l'exécution par une instruction-envoyer dans une activité et une instruction-recevoir dans une autre, tout en se référant au même nom-port-d'accès-message.

Ces instructions synchronisent également les deux activités. Toutes les deux attendent sur leurs instructions-envoyer et instructions-recevoir respectives jusqu'à ce que les données soient transférées avec succès à la liste-variable dans l'activité réceptrice.

CLRL : clear LAM, either F10 at a sub-address or clear the bit in the group 2 register A12.
MENL : module enable LAM using the sub-address for overall control, see Subclause 2.3.
MDISL : module disable LAM, overall control.
MTEST : test LAM in the module, overall test.
MCLRL: clear all LAMs in the module.

Note that only modules built with the facility of overall LAM control at a sub-address will respond to the control actions MENL, MDISL, MTEST and MCLRL.

Note also that ENL and DISL are CAMAC statements equivalent to the general statements CONNECT and DISCONNECT defined in the Real-time BASIC standard.

An example of a LAM CONTROL action is:

800 CONTROL FULL ENL

which enables the LAM in Station 17, Sub-address 0 given the declaration in line 430, Sub-clause 3.2.

7.2 *LAM servicing*

A program interrupt mechanism is not provided at the BASIC level. A LAM service routine is programmed as a parallel activity with a wait-statement naming the LAM. When the LAM occurs, the activity proceeds from the wait statement, which then clears the LAM automatically. The activity is normally programmed to return to the wait statement after servicing the LAM.

7.3 *Dynamic GL modification*

Systems using a programmable LAM grader must be able to change the GL number associated with an event. The function GMY is used for this purpose:

LET lam-name = GMY left-parenthesis numeric-expression right-parenthesis

for example:

940 LET FULL = GMY(7)

This is a special assignment statement similar to the CAMAC address assignments defined in Clause 3. It is the only statement in which an event-name can appear on the left-hand side of an assignment. The function GMY can be used only in this statement.

## 8. Message passing

Send-statements and receive-statements are used to transmit data over message-paths between concurrent-activities. Message-paths are established at run-time by the execution of a send-statement in one activity and a receive-statement in another, referring to the same message-port-name.

These statements also synchronize the two activities. Both activities wait at their respective send-statement and receive-statement until the data are transferred successfully to the variable-list in the receiving activity.

Les ports-d'accès-message doivent être déclarés dans la section des déclarations au début du programme. La déclaration donne le nom du port d'accès et de la structure des données qui sont transférées à travers lui. En utilisant la structure déclarée au paragraphe 3.1, ligne 200, voici un exemple de déclaration de port-d'accès-message:

210 MESSAGE LINK OF STAT

Exemples d'instructions de message valables se conformant à cette déclaration:

1100 SEND TO LINK FROM A, X/2, 17.35, RESULTS( ), COND$, «FIRST»
1200 RECEIVE FROM LINK TO P(1), P(2), Q, I( ), COND$, A$

Noter que ces instructions doivent être dans différentes activités simultanées et que ces activités simultanées ont leurs propres variables locales, de telle sorte que COND$ à la ligne 1100 et COND$ à la ligne 1200 ne sont pas les mêmes variables.

Les définitions formelles sont:

| | | |
|---|---|---|
| dec-port-d'accès-message | = | MESSAGE nom-port-d'accès-message OF nom-structure |
| nom-port-d'accès-message | = | lettre (lettre/chiffre) * |
| instruction-envoyer | = | SEND TO nom-port-d'accès-message FROM structure-sortie (TIMEOUT expression-temps)? |
| instruction-recevoir | = | RECEIVE FROM nom-part-d'accès-message TO structure-entrée (TIMEOUT expression-temps)? |

## 9. Données partagées

Les instructions-prendre et les instructions-mettre sont utilisées pour transmettre les données à travers les ports d'accès de données partagées vers des collections de données partagées. Les ports d'accès doivent être déclarés dans la section des déclarations en tête de programme. La déclaration donne le nom du port d'accès, la structure des données transférées à travers lui et définit également une section réelle de données partagées qui existe en dehors des activités simultanées y accédant.

Les instructions-prendre et les instructions-mettre sont des opérations indivisibles. Indivisible signifie que lorsqu'une section de données partagées est atteinte par une instruction-prendre ou une instruction-mettre, aucune autre instruction-prendre ou instruction-mettre d'une autre activité simultanée ne peut accéder à cette section de données partagées avant que l'accès original soit achevé et sans se soucier d'un quelconque re-ordonnancement des activités effectuées par le système d'exploitation.

Exemple de ces instructions:

140 SHARED FLIGHT(10) OF STAT
1110 GET FROM FLIGHT(3) TO X,Y,ALL, MOD( ), ID$, IQ$
1210 PUT TO FLIGHT(I) FROM 3,Y,Z*1.5,CON( ), «NEXT», A$

Les définitions formelles sont:

| | | |
|---|---|---|
| dec-port-d'accès-données | = | SHARED nom-port-d'accès-données dimensionnement? OF nom-structure |
| nom-port-d'accès-données | = | lettre (lettre/chiffre) * |
| instruction-prendre | = | GET FROM nom-port-d'accès-données TO structure-entrée |
| instruction-mettre | = | PUT TO nom-port-d'accès-données FROM structure-sortie |

Message-ports shall be declared in the declaration section at the beginning of the program. The declaration gives the name of the port and the structure of the data transferred through it. Using the structure declared in Sub-clause 3.1, line 200, an example of a message-port declaration is:

210 MESSAGE LINK OF STAT

Examples of valid message statements conforming to this declaration are:

1100 SEND TO LINK FROM A, X/2, 17.35, RESULTS( ), COND$, "FIRST"
1200 RECEIVE FROM LINK TO P(1), P(2), Q, I( ), COND$, A$

Note that these statements shall be in different concurrent activities, and that concurrent activities have their own local variables so that COND$ in line 1100 and COND$ in line 1200 are not the same variable.

The formal definitions are:

| | | |
|---|---|---|
| message-port-dec | = | MESSAGE message-port-name OF structure-name |
| message-port-name | = | letter (letter/digit) * |
| send-statement | = | SEND TO message-port-name FROM out-structure (TIMEOUT time-expression)? |
| receive-statement | = | RECEIVE FROM message-port-name TO in-structure (TIMEOUT time-expression)? |

## 9. Shared data

Get-statements and put-statements are used to transmit data through shared-data ports to collections of shared data. The ports shall be declared in the declaration section at the head of the program. The declaration gives the name or the port, the structure of the data transferred through it, and also defines a real section of shared data that exists outside the concurrent activities accessing it.

Get-statements and put-statements are indivisible operations. Indivisible means that when a section of shared data is accessed by a get-statement or a put-statement, no other get-statement or put-statement in another concurrent activity can access that section of shared data until the original access is complete, regardless of any rescheduling of activities carried out by the operating system.

Examples of these statements are:

140 SHARED FLIGHT(10) OF STAT
1110 GET FROM FLIGHT(3) TO X,Y,ALL, MOD( ), ID$, IQ$
1210 PUT TO FLIGHT(I) FROM 3,Y,Z*1.5,CON( ), "NEXT", A$

The formal definitions are:

| | | |
|---|---|---|
| data-port-dec | = | SHARED data-port-name dimensioning? OF structure-name |
| data-port-name | = | letter (letter/digit) * |
| get-statement | = | GET FROM data-port-name TO in-structure |
| put-statement | = | PUT TO data-port-name FROM out-structure |

## 10. Manipulation de bit

La norme BASIC temps réel définit une manipulation de bit en termes de «chaînes de caractères binaires». Une chaîne de caractères binaires est une séquence de caractères «0» et «1». Un programme d'application utilise les opérations habituelles de découpage et de concaténation pour positionner et tester les bits; par exemple, s'il est demandé de tester le bit 5 d'une combinaison binaire stockée dans la variable B$, on utilisera des instructions telles que:

IF B$(5:5) = «1» THEN 2000
ou IF SEG$(B$,5,5) = «1» THEN 2000

qui dépendent de la mise en œuvre. Dans de nombreuses applications CAMAC, de telles opérations seraient trop lentes et impliqueraient trop de mémoire; en ce cas, les fonctions binaires suivantes sont recommandées:

AND: parenthèse-gauche expression-numérique virgule expression-numérique parenthèse-droite

OR : parenthèse-gauche expression-numérique virgule expression-numérique parenthèse-droite

XOR: parenthèse-gauche expression-numérique virgule expression-numérique parenthèse-droite

NOT: parenthèse-gauche expression-numérique parenthèse-droite

Les expressions-numériques ne doivent contenir que des constantes et des entiers.

## 10. Bit manipulation

The Real-time BASIC standard defines bit manipulation in terms of "bit strings". A bit string is a sequence of the characters "0" and "1". An application program uses the normal string operations of segmentation and concatenation to set and test bits, for example if it is required to test bit 5 of a bit pattern stored in the variable B$, statements such as:

IF B$(5:5) = "1" THEN 2000
or IF SEG$(B$,5,5) = "1" THEN 2000

are used depending on the implementation. For many CAMAC applications this approach will be too slow and occupy too much memory, in which case the following bit-functions are recommended:

AND: left-parenthesis numeric-expression comma numeric-expression right-parenthesis

OR : left-parenthesis numeric-expression comma numeric-expression right-parenthesis

XOR: left-parenthesis numeric-expression comma numeric-expression right-parenthesis

NOT: left-parenthesis numeric-expression right-parenthesis

The numeric-expressions shall contain only numeric constants and integers.

# ANNEXE A

## MÉTHODE DE DÉFINITION DE LA SYNTAXE

Les conventions utilisées dans les définitions formelles sont celles qui sont employées dans les normes ISO appropriées. Ces conventions sont entièrement décrites dans ces normes; cependant, une brève description de la méthode pour définir la syntaxe est donnée ci-après.

Le métalangage syntaxique utilisé découle du BNF. La syntaxe est déterminée par des séries de «règles d'élaboration» qui définissent des éléments syntaxiques en termes d'autres éléments syntaxiques de façon hiérarchique jusqu'à atteinte d'un «symbole terminal». Un symbole terminal est un élément du langage en cours de définition, c'est-à-dire du BASIC temps réel pour CAMAC. On utilise aussi certains symboles particuliers dont les définitions sont données ci-après:

= sépare la partie gauche de la partie droite dans une règle d'élaboration

? l'élément syntaxique qui précède est facultativement présent

* l'élément syntaxique qui précède est facultativement présent un nombre arbitraire de fois (y compris les temps nuls)

(et) les symboles ( ) sont utilisés pour grouper les éléments syntaxiques en une seule expression

/ sépare des options

Des espaces et nouvelles mises à la ligne sont employées pour améliorer la lisibilité des définitions; elles n'ont pas de signification syntaxique.

Les exemples suivants illustrent l'emploi de quelques-uns de ces symboles:

structure-sortie = élément-structure-sortie
(virgule élément-structure-sortie) *
élément-structure-sortie = expression/tableau-formel

Cela signifie qu'une structure-sortie est une liste d'éléments-structure-sortie. S'il y a plus d'un article, ils sont séparés par des virgules. Chaque article peut être soit une expression, soit un tableau formel.

Un exemple d'une structure-sortie répondant à cette définition est:

A + 2, B ( ), C$

# APPENDIX A

## METHOD OF SYNTAX DEFINITION

The conventions used in the formal definitions are those employed in the relevant ISO standards. The conventions are explained fully in those standards, but a brief description of the method of syntax definition is given below.

The syntactic metalanguage used is derived from BNF. The syntax is defined by a series of "production rules" that define syntactic elements in terms of other syntactic elements in a hierarchical manner, until a "terminal symbol" is reached. A terminal symbol is an element of the language being defined, that is Real-time BASIC for CAMAC. Certain special symbols are used whose meaning is defined below:

= separates the left part from the right part in a production rule

? the preceding syntactic element is optionally present

* the preceding syntactic element is optionally present an arbitrary number of times (including zero times)

(and) the symbols ( ) are used to group syntactic elements into a single unit

/ separates alternatives

Spaces and new lines are used to improve legibility of the definitions; they have no syntactic significance.

The following example illustrates the use of some of these symbols:

```
out-structure          =  out-structure-element
                          (comma out-structure-element) *
out-structure-element  =  expression/formal-array
```

This means that an out-structure is a list of out-structure-elements. If there is more than one item in the list, the items are separated by commas. Each item can be either an expression or a formal array.

An example of an out-structure satisfying this definition is:

A + 2, B ( ), C$

# ANNEXE B

## LES DÉFINITIONS FORMELLES

Cette annexe donne la liste de toutes les définitions formelles dans l'ordre alphabétique avec référence, dans la marge de gauche, à l'article ou au paragraphe dans lequel elles sont introduites.

| | | | |
|---|---|---|---|
| 3.2 | accès | = | parenthèse-gauche fonction-accès (virgule fonction-accès) * parenthèse-droite |
| 3.2 | bcna | = | parenthèse-gauche entier? virgule entier? virgule entier virgule entier parenthèse-droite |
| 3.2 | clause-événement | = | EVENT nom-lam |
| 3.2 | clause-processus | = | notification-es nom-port-d'accès-camac limites? (OF nom-structure)? (TIMEOUT rep-numérique) |
| 3.2 | dec-lam | = | nom-port-d'accès-camac GL entier ((P/A) entier)? |
| 9. | dec-port-d'accès-données | = | SHARED nom-port-d'accès-données dimensionnement? OF nom-structure |
| 8. | dec-port-d'accès-message | = | MESSAGE nom-port-d'accès-message OF nom-structure |
| 3.2 | dec-port-d'accès-processus | = | PROCESS (clause-processus/clause-événement) accès-information |
| 3.2 | dec-registre | = | bcna accès? format? |
| 3.1 | dec-structure-données | = | STRUCTURE nom-structure deux-points répétition? type (virgule répétition? type) * |
| 3.2 | dec-tableau-processus | = | nom-tableau-processus dimensionnement |
| 3.1 | dimensionnement | = | parenthèse-gauche limites parenthèse-droite |
| 5. | élément-structure-entrée | = | variable/tableau-formel |
| 5. | élément-structure-sortie | = | expression/tableau-formel |
| 3.1 | entier | = | chiffre chiffre * |
| 4. | événement-attente | = | EVENT (nom-événement/nom-lam) |
| 3.2 | exposant | = | E (signe-plus/signe-moins)? entier |
| 4. | expression-temps | = | expression-numérique/expression-chaîne-de-caractères |
| 3.2 | fonction-accès | = | (F entier)/NX/nom-canal |
| 3.2 | format | = | parenthèse-gauche (B/C/I) entier parenthèse-droite |
| 3.2 | fraction | = | point entier |
| 3.2 | information-accès | = | guillemet CAMAC (dec-registre/dec-lam) guillemet |
| 4. | instruction-arrêt-activité parallèle | = | PARSTOP |
| 4. | instruction attente | = | WAIT (intervalle-attente/temps-attente/événement-attente) |
| 4. | instruction-départ | = | START nom-activité |
| 3.2 | instruction-dim-processus | = | PRODIM dec-tableau-processus (virgule dec-tableau-processus) * |

# APPENDIX B

## THE FORMAL DEFINITIONS

This appendix lists all the formal definitions in alphabetical order, with a reference in the left-hand margin to the clause or sub-clause in which they are introduced.

| | | | |
|---|---|---|---|
| 3.2 | access | = | left-parenthesis access-function (comma access-function) * right-parenthesis |
| 3.2 | access-function | = | (F integer)/NX/ch-name |
| 3.2 | access-information | = | quotation-mark CAMAC (register-dec/lam-dec) quotation-mark |
| 4. | activity-name | = | letter (letter/digit) * |
| 3.2 | bcna | = | left-parenthesis integer? comma integer? comma integer comma integer right-parenthesis |
| 3.1 | bounds | = | integer (comma integer)? |
| 3.2 | camac-port-name | = | letter (letter/digit) * |
| 3.2 | ch-name | = | implementation-defined |
| 5.3 | control-action | = | (F op-code)/module-op/crate-op |
| 5.3 | crate-op | = | CZ/ENCD/DISCD/CC/CLRCI/SETCI |
| 9. | data-port-dec | = | SHARED data-port-name dimensioning? OF structure-name |
| 9. | data-port-name | = | letter (letter/digit) * |
| 3.1 | data-structure-dec | = | STRUCTURE structure-name colon repeat-count? type (comma repeat-count? type) * |
| 3.1 | dimensioning | = | left-parenthesis bounds right-parenthesis |
| 3.2 | event-clause | = | EVENT lam-name |
| 4. | event-name | = | letter (letter/digit) * |
| 5. | formal-array | = | array left-parenthesis comma? right-parenthesis |
| 3.2 | format | = | left-parenthesis (B/C/I) integer right-parenthesis |
| 3.2 | fraction | = | full-stop integer |
| 9. | get-statement | = | GET FROM data-port-name TO in-structure |
| 5. | in-statement | = | IN FROM camac-port-name TO in-structure |
| 5. | in-structure | = | in-structure-element (comma in-structure-element) * |
| 5. | in-structure-element | = | variable/formal-array |
| 3.1 | integer | = | digit digit * |
| 3.2 | io-qualifier | = | INPUT/OUTPUT/OUTIN |
| 7.1 | lam-action | = | ENL/DISL/TEST/CLRL/MENL/ MDISL/MTEST/MCLRL |

| | | | |
|---|---|---|---|
| 5. | instruction-entrée | = | IN FROM nom-port-d'accès-camac<br>TO structure-entrée |
| 8. | instruction-envoyer | = | SEND TO nom-port-d'accès-message<br>FROM structure-sortie<br>(TIMEOUT expression-temps)? |
| 5. | instruction-es-processus | = | instruction-entrée/instruction-sortie |
| 9. | instruction-mettre | = | PUT TO nom-port-d'accès-données<br>FROM structure-sortie |
| 4. | instruction-ordonnancement | = | instruction-départ/instruction-attente/<br>instruction-signal/instruction-arrêt-<br>activité-parallèle |
| 9. | instruction-prendre | = | GET FROM nom-port-d'accès-données<br>TO structure-entrée |
| 8. | instruction-recevoir | = | RECEIVE FROM nom-port-d'accès-message<br>TO structure-entrée<br>(TIMEOUT expression-temps)? |
| 4. | instruction-signal | = | SIGNAL nom-événement |
| 5. | instruction-sortie | = | OUT TO nom-port-d'accès-camac<br>FROM structure-sortie |
| 4. | intervalle-attente | = | DELAY expression-temps |
| 3.1 | limites | = | entier (virgule entier)? |
| 3.2 | mantisse | = | entier point?/entier? fraction |
| 4. | nom-activité | = | lettre (lettre/chiffre) * |
| 3.2 | nom-canal | = | mise-en-œuvre-définie |
| 4. | nom-événement | = | lettre (lettre/chiffre) * |
| 3.2 | nom-lam | = | lettre (lettre/chiffre) * |
| 3.1 | nom-structure | = | lettre (lettre/chiffre) * |
| 3.2 | nom-port-d'accès-camac | = | lettre (lettre/chiffre) * |
| 9. | nom-port-d'accès-données | = | lettre (lettre/chiffre) * |
| 8. | nom-port-d'accès-message | = | lettre (lettre/chiffre) * |
| 3.2 | notification-es | = | INPUT/OUTPUT/OUTIN |
| 5.3 | op-châssis | = | CZ/ENCD/DISCD/CC/CLRCI/SETCI |
| 5.3 | op-module | = | ENB/DIS/CL1/CL2 |
| 5.3 | ordre-contrôle | = | (F code-op)/op-module/op-châssis |
| 7.1 | ordre-lam | = | ENL/DISL/TEST/CLRL/<br>MENL/MDISL/MTEST/MCLRL |
| 3.1 | répétition | = | entier OF |
| 3.2 | rep-numérique | = | mantisse exposant? |
| 5. | structure-entrée | = | élément-structure-entrée<br>(virgule élément-structure-entrée) * |
| 5. | structure-sortie | = | élément-structure-sortie<br>(virgule élément-structure-sortie) * |
| 5. | tableau-formel | = | tableau parenthèse-gauche virgule?<br>parenthèse-droite |
| 4. | temps-attente | = | TIME expression-temps |
| 3.1 | type | = | (REAL/INTEGER/STRING) dimensionnement? |

| | | | |
|---|---|---|---|
| 3.2 | lam-dec | = | camac-port-name GL integer ((P/A) integer)? |
| 3.2 | lam-name | = | letter (letter/digit) * |
| 8. | message-port-dec | = | MESSAGE message-port-name<br>OF structure-name |
| 8. | message-port-name | = | letter (letter/digit) * |
| 5.3 | module-op | = | ENB/DIS/CL1/CL2 |
| 3.2 | numeric-rep | = | significand xrad? |
| 5. | out-statement | = | OUT TO camac-port-name<br>FROM out-structure |
| 5. | out-structure | = | out-structure-element<br>(comma out-structure-element) * |
| 5. | out-structure-element | = | expression/formal-array |
| 4. | parstop-statement | = | PARSTOP |
| 3.2 | process-array-dec | = | process-array-name dimensioning |
| 3.2 | process-clause | = | io-qualifier camac-port-name bounds?<br>(OF structure-name)?<br>(TIMEOUT numeric-rep)? |
| 3.2 | process-dim-statement | = | PRODIM process-array-dec<br>(comma process-array-dec) * |
| 5. | process-io-statement | = | in-statement/out-statement |
| 3.2 | process-port-dec | = | PROCESS (process-clause/event-clause)<br>access-information |
| 9. | put-statement | = | PUT TO data-port-name<br>FROM out-structure |
| 8. | receive-statement | = | RECEIVE FROM message-port-name<br>TO in-structure<br>(TIMEOUT time-expression)? |
| 3.2 | register-dec | = | bcna access? format? |
| 3.1 | repeat-count | = | integer OF |
| 4. | scheduling-statement | = | start-statement/wait-statement/<br>signal-statement/parstop-statement |
| 8. | send-statement | = | SEND TO message-port-name<br>FROM out-structure<br>(TIMEOUT time-expression)? |
| 4. | signal-statement | = | SIGNAL event-name |
| 3.2 | significand | = | integer full-stop?/integer? fraction |
| 4. | start-statement | = | START activity-name |
| 3.1 | structure-name | = | letter (letter/digit) * |
| 4. | time-expression | = | numeric-expression/string-expression |
| 3.1 | type | = | (REAL/INTEGER/STRING) dimensioning? |
| 4. | wait-event | = | EVENT (event-name/lam-name) |
| 4. | wait-interval | = | DELAY time-expression |
| 4. | wait-statement | = | WAIT (wait-interval/wait-time/wait-event) |
| 4. | wait-time | = | TIME time-expression |
| 3.2 | xrad | = | E (plus-sign/minus-sign)? integer |

# ANNEXE C

## MOTS CLÉS CAMAC, FONCTIONS ET INSTRUCTIONS

Cette annexe donne la liste de tous les nouveaux mots clés et instructions dans l'ordre alphabétique, avec référence, dans la marge de gauche, à l'article ou au paragraphe dans lequel ils sont introduits.

| | | |
|---|---|---|
| 3.2 | A | la sous-adresse pour contrôle total de LAM |
| 3.3 | AEX | une fonction d'examen d'une adresse CAMAC |
| 3.3 | AMY | une fonction de modification d'une adresse CAMAC |
| 10. | AND | une fonction logique |
| 3.2 | B | signe binaire et sa taille |
| 3.3 | BEX | une fonction d'examen d'une adresse CAMAC |
| 3.3 | BMY | une fonction de modification d'une adresse CAMAC |
| 3.2 | C | format BCD |
| 3.2 | CAMAC | signifie une entité CAMAC |
| 5.3 | CC | une action de contrôle et de commande CAMAC |
| 3.3 | CEX | une fonction d'examen d'une adresse CAMAC |
| 5.3 | CLRCI | une action de contrôle et de commande CAMAC |
| 7.1 | CLRL | une action de contrôle et de commande de LAM |
| 5.3 | CL1 | une action de contrôle et de commande CAMAC |
| 5.3 | CL2 | une action de contrôle et de commande CAMAC |
| 3.3 | CMY | une fonction de modification d'une adresse CAMAC |
| 5.3, 7.1 | CONTROL | mot clé pour une instruction de contrôle et de commande |
| 5.3 | CZ | une action de contrôle et de commande CAMAC |
| 4. | DELAY | introduit un retard temporel dans une instruction-attente |
| 5.3 | DIS | une action de contrôle et de commande CAMAC |
| 5.3 | DISCD | une action de contrôle et de commande CAMAC |
| 7.1 | DISL | une action de contrôle et de commande de LAM |
| 5.3 | ENB | une action de contrôle et de commande CAMAC |
| 5.3 | ENCD | une action de contrôle et de commande CAMAC |
| 4. | END PARACT | termine le code d'une activité parallèle |
| 7.1 | ENL | une action de contrôle et de commande de LAM |
| 3.2, 4. | EVENT | introduit un nom-événement |
| 3.2 | F | introduit un code de fonction CAMAC |
| 9. | GET | mot clé pour une instruction d'entrée en données partagées |
| 3.2 | GL | introduit un numéro de «GL» CAMAC |
| 7.3 | GMY | une fonction pour changer le numéro GL LAM |
| 3.2 | I | complément à deux... format entier |
| 5.1 | IN FROM | mot clé pour une instruction d'entrée CAMAC |
| 3.2 | INPUT | un registre CAMAC fournissant l'entrée seulement |
| 3.1 | INTEGER | un type de données dans une déclaration de structure |
| 7.1 | MCLR | une action de contrôle et de commande de LAM |

# APPENDIX C

## CAMAC KEYWORDS, FUNCTIONS AND STATEMENTS

This appendix lists all the new keywords and statements in alphabetical order, with a reference in the left-hand margin to the clause or sub-clause in which they are introduced.

| | | |
|---|---|---|
| 3.2 | A | the sub-address for overall LAM control |
| 3.3 | AEX | a function to examine a CAMAC address |
| 3.3 | AMY | a function to modify a CAMAC address |
| 10. | AND | a logical function |
| 3.2 | B | binary sign and magnitude |
| 3.3 | BEX | a function to examine a CAMAC address |
| 3.3 | BMY | a function to modify a CAMAC address |
| 3.2 | C | BCD format |
| 3.2 | CAMAC | signifies a CAMAC entity |
| 5.3 | CC | a CAMAC CONTROL action |
| 3.3 | CEX | a function to examine a CAMAC address |
| 5.3 | CLRCI | a CAMAC CONTROL action |
| 7.1 | CLRL | a LAM CONTROL action |
| 5.3 | CL1 | a CAMAC CONTROL action |
| 5.3 | CL2 | a CAMAC CONTROL action |
| 3.3 | CMY | a function to modify a CAMAC address |
| 5.3, 7.1 | CONTROL | keyword for a CONTROL statement |
| 5.3 | CZ | a CAMAC CONTROL action |
| 4. | DELAY | introduces a time delay in a wait-statement |
| 5.3 | DIS | a CAMAC CONTROL action |
| 5.3 | DISCD | a CAMAC CONTROL action |
| 7.1 | DISL | a LAM CONTROL action |
| 5.3 | ENB | a CAMAC CONTROL action |
| 5.3 | ENCD | a CAMAC CONTROL action |
| 4. | END PARACT | terminates the code for a parallel activity |
| 7.1 | ENL | a LAM CONTROL action |
| 3.2, 4. | EVENT | introduces an event-name |
| 3.2 | F | introduces a CAMAC function code |
| 9. | GET | keyword for a shared-data input statement |
| 3.2 | GL | introduces a CAMAC "GL" number |
| 7.3 | GMY | a function to change a LAM GL number |
| 3.2 | I | two's complement... integer format |
| 5.1 | IN FROM | keyword for a CAMAC input statement |
| 3.2 | INPUT | a CAMAC register provides input only |
| 3.1 | INTEGER | a data-type in a structure declaration |
| 7.1 | MCLR | a LAM CONTROL action |

| | | |
|---|---|---|
| 7.1 | MDISL | une action de contrôle et de commande de LAM |
| 7.1 | MENL | une action de contrôle et de commande de LAM |
| 8. | MESSAGE | introduit une déclaration de port d'accès-message |
| 7.1 | MTEST | une action de contrôle et de commande de LAM |
| 3.3 | NEX | une action d'examen d'une adresse CAMAC |
| 3.3 | NMY | une action de modification d'une adresse CAMAC |
| 10. | NOT | une fonction logique |
| 3.2 | NX | un mot clé indiquant qu'aucun test d'erreur sur X ne doit être exécuté lorsqu'on a accès à un module |
| 10. | OR | une fonction logique |
| 3.2 | OUTIN | un registre CAMAC comportant à la fois une entrée et une sortie |
| 3.2 | OUTPUT | un registre CAMAC acceptant une entrée seulement |
| 5.1 | OUT TO | une instruction CAMAC de sortie |
| 3.2 | P | introduit la position d'un bit de LAM |
| 4. | PARACT | introduit une section de code pour une activité parallèle |
| 4. | PARSTOP | une instruction d'ordonnancement |
| 3.2 | PROCESS | introduit une déclaration de port d'accès d'E/S CAMAC |
| 3.2 | PRODIM | introduit une déclaration de tableau CAMAC |
| 9. | PUT | mot clé pour une instruction de sortie en données partagées |
| 6. | QCAM | une variable de système pour ranger l'état du signal «Q» CAMAC |
| 3.1 | REAL | un type de données dans une déclaration de structure |
| 8. | RECEIVE | une instruction d'entrée de message |
| 8. | SEND | mot clé pour une instruction de sortie de message |
| 5.3 | SETCI | une action de contrôle et de commande CAMAC |
| 9. | SHARED | définit une section de données partagées par des activités parallèles |
| 4. | SIGNAL | mot clé pour une instruction-signal |
| 4. | START | instruction pour démarrer une activité parallèle |
| 3.1 | STRUCTURE | introduit une liste de types de données définissant une structure |
| 3.1 | STRING | un type de données dans une déclaration de structure |
| 7.1 | TEST | une action de contrôle et de commande de LAM |
| 4. | TIME | introduit une expression-temps dans une instruction-attente |
| 3.2, 8. | TIMEOUT | introduit une valeur de délai pour les E/S à travers les ports d'accès |
| 4. | WAIT | mot clé pour une instruction-attente |
| 6. | XCAM | une variable de système pour ranger l'état du signal «X» CAMAC |
| 10. | XOR | une fonction logique |

| | | |
|---|---|---|
| 7.1 | MDISL | a LAM CONTROL action |
| 7.1 | MENL | a LAM CONTROL action |
| 8. | MESSAGE | introduces a message-port declaration |
| 7.1 | MTEST | a LAM CONTROL action |
| 3.3 | NEX | a function to examine a CAMAC address |
| 3.3 | NMY | a function to modify a CAMAC address |
| 10. | NOT | a logical function |
| 3.2 | NX | a keyword indicating that no X error check should be performed when the module is accessed |
| 10. | OR | a logical function |
| 3.2 | OUTIN | a CAMAC register supports both input and output |
| 3.2 | OUTPUT | a CAMAC register accepts input only |
| 5.1 | OUT TO | a CAMAC output statement |
| 3.2 | P | introduces the bit position of a LAM |
| 4. | PARACT | introduces a section of code for a parallel activity |
| 4. | PARSTOP | a scheduling statement |
| 3.2 | PROCESS | introduces a CAMAC I/O port declaration |
| 3.2 | PRODIM | introduces a CAMAC array |
| 9. | PUT | keyword for a shared-data output statement |
| 6. | QCAM | a system variable for staticising the CAMAC "Q" signal |
| 3.1 | REAL | a data-type in a structure declaration |
| 8. | RECEIVE | a message input statement |
| 8. | SEND | keyword for a message output statement |
| 5.3 | SETCI | a CAMAC CONTROL action |
| 9. | SHARED | defines a section of data shared by parallel activities |
| 4. | SIGNAL | keyword for a signal-statement |
| 4. | START | statement to start a parallel activity |
| 3.1 | STRUCTURE | introduces a list of data-types defining a structure |
| 3.1 | STRING | a data-type in a structure declaration |
| 7.1 | TEST | a LAM CONTROL action |
| 4. | TIME | introduces a time-expression in a wait-statement |
| 3.2, 8. | TIMEOUT | introduces a time-out value for I/O through ports |
| 4. | WAIT | keyword for a wait-statement |
| 6. | XCAM | a system variable for staticising the CAMAC "X" signal |
| 10. | XOR | a logical function |

**ICS 27.120**

Typeset and printed by the IEC Central Office
GENEVA, SWITZERLAND