

**NORME
INTERNATIONALE
INTERNATIONAL
STANDARD**

**CEI
IEC
60975**

Première édition
First edition
1989-08

**Format universel pour les modules objets
de microprocesseurs**

**Microprocessor universal format
for object modules**



Numéro de référence
Reference number
CEI/IEC 60975: 1989

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**

60975

Première édition
First edition
1989-08

**Format universel pour les modules objets
de microprocesseurs**

**Microprocessor universal format
for object modules**

© IEC 1989 Droits de reproduction réservés — Copyright - all rights reserved

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

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

International Electrotechnical Commission
Telefax: +41 22 919 0300

3, rue de Varembé Geneva, Switzerland
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

W

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

SOMMAIRE

	Pages
PREAMBULE	6
PREFACE	6
 Section	
1. Domaine d'application	8
2. Définitions	8
3. Références	12
4. Introduction	14
4.1 Commandes	14
4.2 Syntaxe en format libre	14
5. Constructions de base	14
5.1 Notation utilisée pour les définitions de la syntaxe	14
5.2 Objets syntaxiques élémentaires	16
6. Expressions	18
6.1 Types d'opérandes	18
6.2 Fonctions sans opérandes	18
6.3 Opérateurs monadiques et fonctions	20
6.4 Opérateurs et fonctions dyadiques	20
6.5 Opérations de manipulations de données	22
6.6 Autres opérations	22
7. Variables	24
7.1 Variable-G (adresse de début de l'exécution)	24
7.2 Variables-I (définition externe-valeurs symboliques)	24
7.3 Variables-L (limite inférieure)	24
7.4 Variables-N (noms)	26
7.5 Variables-P (pointeur de chargement)	26
7.6 Variables-R (référence de relogement)	26
7.7 Variables-S (taille de section)	26
7.8 Variables-W (registre de travail)	28
7.9 Variables-X (référence à un symbole externe)	28
8. Commandes au niveau module	28
8.1 Commande MB (début de module)	28
8.2 Commande ME (fin de module)	28
8.3 Commande DT (date et heure de création)	28
8.4 Commande AD (descripteur d'adresse)	30
9. Commentaires et somme de contrôle	32
9.1 Commentaires	32
9.2 Commande CS (somme de contrôle)	32

CONTENTS

	Page
FOREWORD	7
PREFACE	7
 Section	
1. Scope	9
2. Definitions	9
3. References	13
4. Introduction	15
4.1 Commands	15
4.2 Free-form syntax	15
5. Basic constructions	15
5.1 Notation used for syntax definitions	15
5.2 Elementary syntactic objects	17
6. Expressions	19
6.1 Types of operands	19
6.2 Functions with no operands	19
6.3 Monadic operators and functions	21
6.4 Dyadic operators and functions	21
6.5 Data manipulation operations	23
6.6 Other operations	23
7. Variables	25
7.1 G-variable (execution starting address)	25
7.2 I-variables (external definition - symbol values)	25
7.3 L-variables (low limit)	25
7.4 N-variables (name)	27
7.5 P-variables (load pointer)	27
7.6 R-variables (relocation reference)	27
7.7 S-variables (section size)	27
7.8 W-variables (working register)	29
7.9 X-variables (external symbol reference)	29
8. Module-level commands	29
8.1 MB (module begin) command	29
8.2 ME (module end) command	29
8.3 DT (date and time of creation) command	29
8.4 AD (address descriptor) command	31
9. Comments and checksum	33
9.1 Comments	33
9.2 CS (checksum) command	33

Section	Pages
10. Sectionnement	34
10.1 Commande SB (début de section)	34
10.2 Commande ST (type de section)	34
10.3 Commande SA (alignement de section)	38
11. Nom symbolique et déclaration de type	38
11.1 Commande NI (nom d'un symbole interne)	38
11.2 Commande-NX (nom d'un symbole externe)	40
11.3 Commande NN (nom)	40
11.4 Commande AT (attributs)	42
11.5 Commande TY (type)	42
12. Commande AS (affectation)	44
13. Commandes de chargement	44
13.1 Commande LD (chargement)	44
13.2 Commande IR (initialiser la base de translation)	46
13.3 Commande LR (charger relojer)	46
13.4 Commande RE (copie)	48
14. Edition de liens	48
14.1 Commande RI (conserver le symbole interne)	50
14.2 Commande WX (symbole externe faible)	50
15. Bibliothèques	50
15.1 Commande LI (spécifier une liste de recherche par défaut en bibliothèque)	50
15.2 Commande LX (bibliothèque externe)	50
16. Noyau FUMOM	52
17. Format binaire	52
18. Bibliographie	52
ANNEXES (informatives)	
ANNEXE A - Syntaxe	54
ANNEXE B - Suggestion de codage binaire	62
ANNEXE C - Utilisation des commandes et des variables par les processus	66

Section	Page
10. Sectioning	35
10.1 SB (section begin) command	35
10.2 ST (section type) command	35
10.3 SA (section alignment) command	39
11. Symbolic name and type declaration	39
11.1 NI (name of internal symbol) command	39
11.2 NX (name of external symbol) command	41
11.3 NN (name) command	41
11.4 AT (attributes) command	43
11.5 TY (type) command	43
12. AS (assignment) command	45
13. Loading commands	45
13.1 LD (load) command	45
13.2 IR (initialize relocation base) command	47
13.3 LR (load relocate) command	47
13.4 RE (replicate) command	49
14. Linkage edition	49
14.1 RI (retain internal symbol) command	51
14.2 WX (weak external symbol) command	51
15. Libraries	51
15.1 LI (specify default library search list) command	51
15.2 LX (library external) command	51
16. MUFOM kernel	53
17. Binary format	53
18. Bibliography	53
APPENDICES (informative)	
APPENDIX A - Collected syntax	55
APPENDIX B - Suggested binary encoding	63
APPENDIX C - Command and variable usage by processes	67

COMMISSION ELECTROTECHNIQUE INTERNATIONALE

FORMAT UNIVERSEL POUR LES MODULES OBJETS DE MICROPROCESSEURS

PREAMBULE

- 1) Les décisions ou accords officiels de la CEI en ce qui concerne les questions techniques, préparés par des Comités d'Etudes où sont représentés tous les Comités nationaux s'intéressant à ces questions, expriment dans la plus grande mesure possible un accord international sur les sujets examinés.
- 2) Ces décisions constituent des recommandations internationales et sont agréées comme telles par les Comités nationaux.
- 3) Dans le but d'encourager l'unification internationale, la CEI exprime le vœu que tous les Comités nationaux adoptent dans leurs règles nationales le texte de la recommandation de la CEI, dans la mesure où les conditions nationales le permettent. Toute divergence entre la recommandation de la CEI et la règle nationale correspondante doit, dans la mesure du possible, être indiquée en termes clairs dans cette dernière.

PREFACE

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

La présente norme a été reprise de la Norme 695 de l'IEEE: IEEE Trial-use Standard for Microprocessor Universal Format for Object Modules (1985).

Le texte de cette norme est issu des documents suivants:

Règle des Six Mois	Rapport de vote
47B(BC)24	47B(BC)30

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

* Le Sous-Comité 47B de la CEI est désormais transféré dans l'ISO/CEI JTC 1.

La présente norme ayant été approuvée selon les procédures de la CEI, elle est par conséquent publiée comme norme de la CEI.

INTERNATIONAL ELECTROTECHNICAL COMMISSION

MICROPROCESSOR UNIVERSAL FORMAT FOR OBJECT MODULES

FOREWORD

- 1) The formal decisions or agreements of the IEC on technical matters, prepared by Technical Committees on which all the National Committees having a special interest therein are represented, express, as nearly as possible, an international consensus of opinion on the subjects dealt with.
- 2) They have the form of recommendations for international use and they are accepted by the National Committees in that sense.
- 3) In order to promote international unification, the IEC expresses the wish that all National Committees should adopt the text of the IEC recommendation for their national rules in so far as national conditions will permit. Any divergence between the IEC recommendation and the corresponding national rules should, as far as possible, be clearly indicated in the latter.

PREFACE

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

This standard is based on IEEE Standard 695: IEEE Trial-Use Standard for Microprocessor Universal Format for Object Modules (1985).

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

Six Months' Rule	Report on Voting
47B(C0)24	47B(C0)30

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

* IEC Sub-Committee 47B has now been transferred to ISO/IEC JTC 1.

This standard was approved according to IEC procedures and is therefore published as an IEC standard.

FORMAT UNIVERSEL POUR LES MODULES OBJETS DE MICROPROCESSEURS

1. Domaine d'application

La présente norme spécifie le format de modules objets reliables, relogeables et absolus destinés à des ordinateurs de tailles de mots et d'architecture arbitraires. Deux niveaux de conformité sont spécifiés, minimal et total.

Le niveau de conformité minimal permet une souplesse suffisante pour relier des modules compilés séparément, pour traduire des adresses de manière simple et pour charger les modules objets absolus résultants avec un chargeur minimal.

Le niveau de conformité total permet toutes les fonctionnalités du niveau minimal et lui adjoint la gestion d'expressions arbitraires d'adresses, la possibilité de contrôle de type, des commandes de gestion de bibliothèques, et d'autres fonctions utiles pour une généralité complète.

Une réalisation conforme peut étendre l'ensemble des commandes ou des fonctions de FUMOM en vue d'une meilleure efficacité pour traiter les exigences spécifiques à la machine, mais les modules objets contenant de telles extensions ne doivent alors pas être réputés conformes à cette norme. De telles extensions ne sont pas spécifiées dans cette norme.

Cette norme ne spécifie pas non plus:

- 1) Les caractéristiques du langage source devant être supportées grâce aux modules FUMOM.
- 2) La structure des librairies de code relogeable.
- 3) Le format de l'équivalent binaire optionnel.
- 4) Les algorithmes pour l'édition de liens, la translation d'adresses et le chargement.
- 5) L'architecture de la machine hôte.
- 6) L'architecture de la machine cible, y compris la taille des mots, la taille de la mémoire, l'organisation de la mémoire ou le format des instructions.

Cette norme est issue de documents du domaine public et ne contient aucun élément breveté ou d'usage privé.

2. Définitions

Les définitions suivantes donnent la signification des mots techniques tels qu'ils sont utilisés dans cette norme et servent à traduire l'acception usuelle des mots définis:

code absolu: Donnée ou code machine exécutable en mémoire ou son image. A distinguer de code relogeable.

MICROPROCESSOR UNIVERSAL FORMAT FOR OBJECT MODULES

1. Scope

This standard specifies the format of linkable, relocatable, and absolute object modules for binary computers of arbitrary word size and architecture. Two levels of compliance are specified, minimum and full.

The minimum compliance level affords sufficient flexibility to link separately compiled modules, to relocate addresses in simple ways, and to load the resulting absolute object modules with a minimal loader.

The full compliance level affords all of the functionality of the minimum level and adds to it arbitrary address expression handling, type checking capability, librarian control commands, and other useful functions for full generality.

A conforming implementation may extend the command or function set of MUFOM for greater efficiency in dealing with machine-specific requirements, but object modules containing such extensions shall not be said to be conforming to this standard. Such extensions are not specified in this standard.

This standard also does not specify:

- 1) The source language features to be supported by way of MUFOM modules.
- 2) The structure of relocatable code libraries.
- 3) The format of the optional binary equivalent.
- 4) Algorithms for linking, relocating and loading.
- 5) The architecture of the host machine.
- 6) The architecture of the target machine, including word size, size of memory, memory organization or instruction format.

This standard is derived from documents in the public domain and contains no patented or proprietary material.

2. Definitions

The following definitions give the meanings of the technical words as they are used in this standard and are intended to reflect the accepted use of the defined words:

absolute code: Data or executable machine code in memory or an image thereof. Distinguish from relocatable code.

chargeur absolu: Un processus pouvant charger une ou plusieurs sections de code absolu seulement aux adresses spécifiées par les sections. Voir: relogeur.

information binaire: Configuration de bits devant être chargée en mémoire.

forme caractère: La représentation sous forme de caractère (imprimable) de l'information binaire par opposition à la représentation sous forme de configuration de bits.

somme de contrôle: Fonction déterministe du contenu d'un fichier. Si un fichier est copié et que la somme de contrôle de la copie est différente de celui de l'original, il y a eu une erreur lors de l'opération de copie.

code: Donnée ou code machine exécutable. Voir: code absolu et code relogeable.

commande: Information de commande d'un éditeur de liens, d'un relogeur ou d'un chargeur. A distinguer de code.

commentaires: Information lisible par l'homme. A distinguer d'information binaire et de forme caractère.

définition externe: Définition, à l'intérieur d'une section, d'un symbole auquel il est fait référence dans d'autres sections. Un symbole global exporté.

référence externe: Spécification, dans une section, d'un symbole qui est défini en dehors de cette section. Un symbole global importé.

format: Le langage dans lequel sont spécifiés les modules objets.

bibliothécaire: Processus qui exécute des opérations, telles que la maintenance, sur une bibliothèque.

bibliothèque: Ensemble de modules objets.

éditeur de liens: Processus qui combine plusieurs modules objets en un seul module objet satisfaisant aux liens entre ceux-ci.

pointeur de chargement: Pointeur relatif à une section géré dynamiquement par le chargeur. Il indique à quel endroit l'élément de code suivant doit être chargé. Il est initialisé à une adresse de chargement de départ.

chargeur: Synonyme de chargeur absolu.

UMA: Abréviation de "unité minimale adressable".

unité minimale adressable: Représente, pour un processeur donné, la mémoire comprise entre une adresse et la suivante. Elle n'est pas nécessairement équivalente à un mot ou à un octet. Abréviation: UMA.

module: Synonyme de module objet.

absolute loader: A process which can load one or more sections of absolute code only at the locations specified by the sections. See: relocater.

binary information: Bit patterns to be loaded into memory.

character form: The (printable) character representation of binary information as opposed to the bit pattern representation.

checksum: A deterministic function of a file's contents. If a file is copied and the checksum of the copy is different from the original, there has been an error in copying.

code: Data or executable machine code. See: absolute code and relocatable code.

command: Control information for a linker, relocater, or loader. Distinguish from code.

comments: Information that is readable by people. To be distinguished from binary information and character form.

external definition: The definition, within a section, of a symbol which is referenced by other sections. An exported global symbol.

external reference: The specification, within a section, of a symbol which is defined outside that section. An imported global symbol.

format: The language in which object modules are specified.

librarian: The process which performs operations, such as maintenance, upon a library.

library: A set of object modules.

linkage editor (or linker): A process which combines object modules into a single object module satisfying links between the object modules.

load pointer: A pointer for a section which is dynamically maintained by the loader. It indicates where the next item of code is to be loaded. It is initialized to a starting load address.

loader: Same as absolute loader.

MAU: Abbreviation for minimum addressable unit.

minimum addressable unit: For a given processor, the amount of memory located between an address and the next address. It is not necessarily equivalent to a word or a byte. Abbreviated MAU.

module: Same as object module.

FUMOM: Acronyme de Format Universel pour les Modules Objets de Microprocesseurs.

module objet: Un ensemble de sections de code absolu ou relogeable, associé à des commandes.

processus: Un programme qui est exécuté par un processeur. Dans cette norme, la dénotation de ce terme est restreinte à éditeur de liens, relogeur, évaluateur d'expressions et chargeur.

processeur: Une machine de calcul, quoique peut-être une machine virtuelle.

programme: Un ensemble de données et d'instructions qui définit les opérations à effectuer par un processeur.

code relogeable: Code indépendant de son emplacement qui peut être chargé à une adresse arbitraire de la mémoire. Il requiert généralement à la fois le relogement et l'évaluation d'expressions pour devenir du code absolu.

relogeur: Un processus qui affecte des adresses absolues à une section de code relogeable en vue de produire du code absolu.

relogement: Fonction du relogeur.

copie multiple: Le chargement de données répétitives grâce à une spécification abrégée.

section: Partie d'un programme dotée d'information auxiliaire (commandes) qui devient un segment lorsqu'elle est chargée.

segment: Région d'un seul tenant de la mémoire, dotée de frontières arbitraires.

doit: Est requis de..., sont requis de...

devrait: Est conseillé de..., sont conseillés de...

adresse de chargement de départ: L'adresse à laquelle le chargement d'une section commence.

type: Attribut d'un symbole, d'une valeur ou d'une section. La définition de type pour un symbole est laissée au réalisateur.

référence externe faible: Référence externe qui ne doit pas nécessairement être satisfaite lors de l'édition de liens.

3. Références

- [1] ANSI X3.4-1977: American National Standard Code for Information Interchange.
- [2] ANSI X3.43-1977: American National Standard Representations of Local Time of the Day for Information Interchange.
- [3] ISO 3307-1975: Echange d'information - Représentations de l'heure.

MUFOM: Acronym for Microprocessor Universal Format for Object Modules.

object module : A set of sections of absolute code or relocatable code, together with commands.

process: A program which is executed by a processor. In this standard, the denotation is restricted to linker, relocater, expression evaluator and loader.

processor: A computing machine, albeit perhaps a virtual machine.

program: A set of data and instructions which defines the operations to be performed by a processor.

relocatable code: Position-independent code which can be loaded at an arbitrary memory location. It generally requires both relocation and expression evaluation to become absolute code.

relocater: A process which assigns absolute addresses to a section of relocatable code to produce absolute code.

relocation: The function of the relocater.

replication: The loading of repetitive data by means of an abbreviated specification.

section: A part of a program with ancillary information (commands) which becomes a segment when loaded.

segment: A contiguous region in memory with arbitrary boundaries.

shall: Is required to ..., are required to...

should: Is advised to ..., are advised to...

starting load address: The address at which the loading of a section begins.

type: An attribute of a symbol, value or section. The definition of a symbol type is left to the implementer.

weak external: An external reference which need not be satisfied during linking.

3. References

- [1] ANSI X3.4-1977: American National Standard Code for Information Interchange.
- [2] ANSI X3.43-1977: American National Standard Representations of Local Time of the Day for Information Interchange.
- [3] ISO 3307-1975: Representations of Time of the Day - Information Interchange.

4. Introduction

4.1 Commandes

Un module objet FUMOM est une séquence de commandes. Ces commandes sont introduites grâce à un mnémonique de deux lettres et sont terminées par un point. Ainsi,

MBI8080.
ASP,100.
LDC30001.
ME.

est un module bien formé. Les descriptions détaillées des commandes de cet exemple sont contenues dans les sections suivantes, mais une paraphrase informelle de cet exemple est la suivante:

Le module commence "Intel 8080."

Affecter la valeur 256 (décimal) au pointeur de chargement.

Charger trois octets avec l'information binaire représentée par la spécification hexadécimale "C3," "00," et "01."

Fin du module

4.2 Syntaxe en format libre

Un module objet est traité comme une chaîne de caractères ASCII contigus conformément à la norme ANSI X3.4 [1]*. Tous les caractères ASCII de contrôle (tels que changement de ligne ou retour chariot) sont ignorés lors du traitement.

La longueur d'une commande FUMOM est variable et il n'y a aucune limitation à cette longueur. Pour représenter un module objet FUMOM comme un fichier texte sur une machine hôte (ayant une longueur de ligne limitée), une commande simple peut occuper plusieurs lignes successives (elle peut, par exemple, comprendre plusieurs changements de ligne, retours chariot ou d'autres caractères de contrôle).

5. Constructions de base

5.1 Notation utilisée pour les définitions de la syntaxe

Le métalangage utilisé pour les définitions de la syntaxe est dérivé de la forme de Backus-Naur et de la notation concernant les expressions régulières.

Les symboles terminaux (caractères littéraux indiqués exactement comme ils sont utilisés) sont inclus entre apostrophes; les symboles non terminaux (noms des formes syntaxiques) sont des mots (éventuellement munis de tirets) sans apostrophes. Des parenthèses sont utilisées pour indiquer les groupements. Les espaces ne sont pas significatifs. La concaténation s'indique par la juxtaposition des symboles. La répétition s'indique par la récursion ou par les deux métasympboles + ou *

* Les nombres entre crochets correspondent aux références indiquées dans la section 3.

4. Introduction

4.1 *Commands*

A MUFOM object module is a sequence of commands. These commands are introduced by a two-letter mnemonic and are terminated by a period. Thus,

```
MBI8080.  
ASP,100.  
LDC30001.  
ME.
```

is a well-formed module. Detailed descriptions of the commands in this example are contained in following sections, but an informal paraphrase of this example is:

Module begin "Intel 8080."

Assign load pointer value to be 256 (decimal).

Load three bytes with the binary information represented by the hexadecimal specification "C3," "00," and "01."

Module end.

4.2 *Free-form syntax*

An object module is treated as one contiguous string of ASCII characters according to ANSI standard X3.4 [1]*. All ASCII control characters (such as a line feed or a carriage return) are ignored in processing.

The length of a MUFOM command is variable and there is no limitation on that length. To represent a MUFOM object module as a text file on a host processor (with limited line size), a single command can occupy several successive lines (for example, encompass several line feeds, carriage returns, or other control characters).

5. Basic constructions

5.1 *Notation used for syntax definitions*

The metalanguage for the syntax definitions is derived from the Backus-Naur form and from regular expression notation.

Terminal symbols (literal characters shown exactly as used) are enclosed in quotation marks; nonterminal symbols (names of syntactic forms) are words (possibly hyphenated) without quotation marks. Parentheses are used to show grouping. Spaces are not significant. Concatenation is indicated by the juxtaposition of symbols. Repetition is indicated by recursion or by the two metasympols + or *

* The numbers in square brackets correspond to the references listed in Section 3.

élément * = zéro ou plusieurs occurrences de élément

élément + = une ou plusieurs occurrences de élément

L'alternative est indiquée par le métasymbole | élément 1 | élément 2:
soit élément 1 soit élément 2, exclusivement
L'option est indiquée par le métasymbole ?

, élément? : soit élément soit rien

EXEMPLE:

variable → lettre (lettre | digit)*

5.2 Objets syntaxiques élémentaires

FUMOM utilise un petit nombre de types de données pour fournir les moyens habituels pour exprimer les constructions. Ce sont les suivants:

digit → "0" | "1" | "2" | "3" | "4" | "5" | "6" | "7" | "8" | "9"

lettre-hexadécimale → "A" | "B" | "C" | "D" | "E" | "F"

digit-hexadécimal → digit | lettre-hexadécimale

nombre-hexadécimal → digit-hexadécimal +

lettre-non-hexadécimale → "G" | "H" | "I" | "J" | "K" | "L" |
"M" | "N" | "O" | "P" | "Q" | "R" |
"S" | "T" | "U" | "V" | "W" | "X" |
"Y" | "Z"

lettre → lettre-hexadécimale | lettre-non-hexadécimale

alphanumérique → lettre | digit

identificateur → lettre alphanumérique*

caractère → tout caractère ASCII graphique

longueur-chaîne-caractère → digit-hexadécimal digit-hexadécimal

chaîne-caractère → longueur-chaîne-caractère caractère*

où la valeur hexadécimale de longueur-chaîne-caractère est prise égale au nombre de caractères, qui doit être inférieur à 128, par exemple 14A STRING WITH SPACES.

variable-FUMOM → lettre-non-hexadécimale nombre-hexadécimal?

un nom qui fait référence à une variable interne d'un processus FUMOM (voir section 7).

item * = zero or more occurrences of item

item + = one or more occurrences of item

Alternation is indicated by the metasymbol | item 1 | item 2: either item 1 or item 2 not both

Election is indicated by the metasymbol ?

, item? : either item or nothing

EXAMPLE:

variable → letter (letter | digit)*

5.2 Elementary syntactic objects

MUFOM uses a small number of data types to provide standard means of expressing constructs. They are as follows:

digit → "0" | "1" | "2" | "3" | "4" | "5" | "6" | "7" | "8" | "9"

hexletter → "A" | "B" | "C" | "D" | "E" | "F"

hexdigit → digit | hexletter

hexnumber → hexdigit +

nonhexletter → "G" | "H" | "I" | "J" | "K" | "L" |
"M" | "N" | "O" | "P" | "Q" | "R" |
"S" | "T" | "U" | "V" | "W" | "X" |
"Y" | "Z"

letter → hexletter | nonhexletter

alphanum → letter | digit

identifier → letter alphanum*

character → any ASCII graphic character

char-string-length → hexdigit hexdigit

char-string → char-string-length character*

Where the hexadecimal value of string-length shall be equal to the number of characters, which shall be less than 128, for example 14A STRING WITH SPACES,

MUFOM-variable → nonhexletter hexnumber?

a name which references an internal variable of a MUFOM process (see Section 7).

6. Expressions

Les expressions sont écrites en notation Polonaise postfixée. La représentation binaire de chaque opérande ne doit pas excéder 64 bits. Une réalisation doit accepter des opérandes d'au moins 15 bits.

expression → nombre-hexadécimal | variable-FUMOM |
expression-polonaise | expression-conditionnelle

expression-polonaise → (opérande ",")* fonction

opérande → expression

fonction → "@" identificateur | opérateur

opérateur → "+" | "-" | "/" | "*" |
"<" | ">" | "=" | "#"

Lorsqu'une fonction ou un opérateur apparaît dans une expression, il spécifie le calcul à effectuer sur les valeurs données dans toute expression antérieure selon la définition de la fonction. Quelques fonctions standard sont fournies avec FUMOM. Cependant, cet ensemble sera vraisemblablement amélioré par le réalisateur de manière à plus aisément s'adapter à sa machine cible.

6.1 Types d'opérandes

Le résultat d'une opération effectuée sur les opérandes dépend des types et des valeurs de ces opérandes. Si le type ou le domaine d'un opérande n'est pas permis par l'opération qui lui est appliquée, le résultat est indéfini. Dans FUMOM, on distingue seulement deux types d'opérandes: logique et entier. Les opérandes entiers peuvent être positifs ou négatifs. Les opérandes logiques peuvent avoir seulement les deux valeurs VRAI et FAUX.

Le type d'un opérande écrit comme 'nombre-hexadécimal' est le type entier, et sa valeur est égale à celle du 'nombre-hexadécimal' considéré comme un nombre hexadécimal positif. Le type d'un opérande écrit comme une 'variable-FUMOM' dépend de la dernière affectation effectuée à la variable correspondante. Le type d'un opérande écrit comme une 'expression-polonaise' est défini ci-dessous. Le type d'un opérande écrit comme une 'expression-conditionnelle' est le type de l'opérande pour lequel la condition est vraie.

Pour les opérations logiques au niveau du bit (@AND, @OR, @XOR, @INS, @EXT), un entier FUMOM est représenté par un nombre binaire aligné sur la droite, complété par des zéros ou tronqué à gauche, et considéré comme non signé.

6.2 Fonctions sans opérandes

expression-polonaise → "@F"

expression-polonaise → "@T"

@F, @T - Ces fonctions renvoient les valeurs logiques FAUX et VRAI. Elles n'ont pas d'opérandes.

6. Expressions

Expressions are written in postfix Polish notation. The binary representation of each operand shall not exceed 64 bits. An implementation shall accept operands of at least 15 bits.

expression → hexnumber | MUFOM-variable |
polish-expr | conditional-expr

polish-expr → (operand ",")* function

operand → expression

function → "@" identifier | operator

operator → "+" | "-" | "/" | "*" |
"<" | ">" | "=" | "#"

When a function or an operator appears in an expression, it specifies the computation to be performed on the values given in any preceding expression or expressions according to the function definition. Some standard functions are provided with MUFOM. However, it is expected that the implementer will enhance this set in order to more easily accommodate his target machine.

6.1 Types of operands

The result of an operation performed on operands depends on the types and values of those operands. If the type or range of an operand is not allowed by the operation applied to it, the result is undefined. In MUFOM, only two types of operands are distinguished: logical and integer. Integer operands can be positive or negative. Logical operands can have only the two values TRUE and FALSE.

The type of an operand written as 'hexnumber' is the integer, and its value is equal to that of the 'hexnumber' considered as a positive hexadecimal number. The type of an operand written as 'MUFOM-variable' depends on the last assignment to the corresponding variable. The type of an operand written as 'polish-expr' will be defined below. The type of an operand written as 'conditional-expr' is the type of the operand for which the condition is true.

For bit-wise logical operations (@AND, @OR, @XOR, @INS, @EXT), a MUFOM integer is represented by a binary number aligned on the right, zero-filled or truncated on the left, and considered as unsigned.

6.2 Functions with no operands

polish-expr → "@F"

polish-expr → "@T"

@F, @T - These functions return the logical values FALSE and TRUE. They have no operands.

6.3 Opérateurs monadiques et fonctions

expression-polonaise → opérande " , " f-monad

f-monad → "@ABS" | "@NEG" | "@NOT" | "@ISDEF"

@ABS - Le type de 'opérande' doit être entier. Cette fonction retourne un entier qui est la valeur absolue de 'opérande'.

@NEG - Le type de 'opérande' doit être entier. Cette fonction retourne un entier qui vaut le complément arithmétique de 'opérande'.

@NOT - Si le type de 'opérande' est logique, la valeur retournée est son complément logique. Si le type de 'opérande' est entier, la valeur retournée est le complément à un de sa valeur.

@ISDEF - Cette fonction retourne la valeur logique VRAI si 'opérande' ne contient aucune variable non affectée, et retourne FAUX dans le cas contraire.

6.4 Opérateurs et fonctions dyadiques

expression-polonaise → opérande1 " , " opérande2 " , " f-dyad

f-dyad → "+" | "-" | "/" | "*" | "@MAX" | "@MIN" | "@MOD" |
"<" | ">" | "=" | "#" | "@AND" | "@OR" | "@XOR"

Pour les fonctions non commutatives, l'ordre conventionnel des opérandes est utilisé. Par exemple, dans le cas de la division, le résultat est la valeur de opérande1 divisée par la valeur de opérande2.

+, -, /, * - Ces opérateurs exécutent les opérations arithmétiques habituelles sur deux opérandes, qui doivent être entiers. Le résultat est entier. La division tronque vers zéro et son résultat est indéfini si opérande2 est nul.

@MAX, @MIN - Ces fonctions réalisent une comparaison arithmétique entre deux opérandes, qui doivent être entiers. Elles retournent un entier qui est le plus grand, ou le plus petit, des deux opérandes.

@MOD - Cette fonction accepte deux opérandes de type entier et retourne la valeur entière de opérande1 modulo opérande2. Le résultat est indéfini si l'un ou l'autre des opérandes est négatif, ou si opérande2 est nul.

<, >, =, # - Ce sont des opérateurs relationnels valables sur des entiers. L'opérateur '#' teste l'inégalité. Ils retournent VRAI ou FAUX.

@AND, @OR, @XOR - Ce sont des opérations logiques s'effectuant sur les représentations sous forme de chaînes de bits des deux opérandes ou sur les valeurs logiques VRAI ou FAUX. Si les opérandes sont des chaînes de bits, elles retournent une chaîne de bits, si ce sont des valeurs logiques, alors elles retournent VRAI ou FAUX.

6.3 Monadic operators and functions

polish-expr → operand "," monad-f

monad-f → "@ABS" | "@NEG" | "@NOT" | "@ISDEF"

@ABS - The type of 'operand' must be integer. This function returns an integer which is the absolute value of 'operand.'

@NEG - The type of 'operand' must be integer. This function returns an integer which is 'operand' arithmetically negated.

@NOT - If the type of 'operand' is logical, the value returned is its logical complement. If the type of 'operand' is integer, the value returned is the one's complement of its value.

@ISDEF - This function returns the logical value TRUE if 'operand' contains no unassigned variables, and returns FALSE otherwise.

6.4 Dyadic operators and functions

polish-expr → operand1 "," operand2 "," dyad-f

dyad-f → "+" | "-" | "/" | "*" | "@MAX" | "@MIN" | "@MOD" |
"<" | ">" | "=" | "#" | "@AND" | "@OR" | "@XOR"

For noncommutative functions, the conventional operand ordering is used. For example, in the case of division, the result is the value of operand1 divided by the value of operand2.

+, -, /, * - These perform the obvious arithmetic operations on two operands, which must be integer. The result is integer. Divide truncates toward zero, and its result is undefined if operand2 is zero.

@MAX, @MIN - These functions perform an arithmetic comparison between two operands, which must be integer. They return an integer which is the greater, or lesser, of the two operands.

@MOD - This takes two integer operands and returns the integer value operand1 modulo operand2. The result is undefined if either operand is negative, or if operand2 is zero.

<, >, =, # - These are relational operators for integer values. The operator '#' tests inequality. They return TRUE or FALSE.

@AND, @OR, @XOR - These are logical operations on the bit string representations of the values of the two operands or on the logical values TRUE or FALSE. If the operands are bit strings, a bit string is returned; if they are logical values, then TRUE or FALSE is returned.