

This is a preview - [click here to buy the full publication](#)

INTERNATIONAL STANDARD

ISO/IEC 9496

Fourth edition
2003-12-15

CHILL — The ITU-T programming language

CHILL — Le langage de programmation de l'UIT-T

Reference number
ISO/IEC 9496:2003(E)



© ISO/IEC 2003

PDF disclaimer

This PDF file may contain embedded typefaces. In accordance with Adobe's licensing policy, this file may be printed or viewed but shall not be edited unless the typefaces which are embedded are licensed to and installed on the computer performing the editing. In downloading this file, parties accept therein the responsibility of not infringing Adobe's licensing policy. The ISO Central Secretariat accepts no liability in this area.

Adobe is a trademark of Adobe Systems Incorporated.

Details of the software products used to create this PDF file can be found in the General Info relative to the file; the PDF-creation parameters were optimized for printing. Every care has been taken to ensure that the file is suitable for use by ISO member bodies. In the unlikely event that a problem relating to it is found, please inform the Central Secretariat at the address given below.

© ISO/IEC 2003

All rights reserved. Unless otherwise specified, 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 either ISO at the address below or ISO's member body in the country of the requester.

ISO copyright office
Case postale 56 • CH-1211 Geneva 20
Tel. + 41 22 749 01 11
Fax + 41 22 749 09 47
E-mail copyright@iso.org
Web www.iso.org

Published in Switzerland

CONTENTS

	<i>Page</i>
1 Introduction	1
1.1 General	1
1.2 Language survey	1
1.3 Modes and classes	2
1.4 Locations and their accesses	3
1.5 Values and their operations	3
1.6 Actions	4
1.7 Input and output	4
1.8 Exception handling	4
1.9 Time supervision	5
1.10 Program structure	5
1.11 Concurrent execution	5
1.12 General semantic properties	6
1.13 Implementation options	6
2 Preliminaries	7
2.1 The metalanguage	7
2.2 Vocabulary	8
2.3 The use of spaces	9
2.4 Comments	9
2.5 Format effectors	9
2.6 Compiler directives	10
2.7 Names and their defining occurrences	10
3 Modes and classes	12
3.1 General	12
3.2 Mode definitions	13
3.3 Mode classification	16
3.4 Discrete modes	17
3.5 Real modes	20
3.6 Powerset modes	22
3.7 Reference modes	22
3.8 Procedure modes	23
3.9 Instance modes	24
3.10 Synchronization modes	25
3.11 Input-Output Modes	26
3.12 Timing modes	28
3.13 Composite modes	29
3.14 Dynamic modes	37
3.15 Moreta Modes	38
4 Locations and their accesses	45
4.1 Declarations	45
4.2 Locations	47
5 Values and their operations	54
5.1 Synonym definitions	54
5.2 Primitive value	55
5.3 Values and expressions	70

		<i>Page</i>
6	Actions.....	79
	6.1 General.....	79
	6.2 Assignment action.....	79
	6.3 If action.....	81
	6.4 Case action.....	81
	6.5 Do action.....	83
	6.6 Exit action.....	86
	6.7 Call action.....	87
	6.8 Result and return action.....	90
	6.9 Goto action.....	90
	6.10 Assert action.....	91
	6.11 Empty action.....	91
	6.12 Cause action.....	91
	6.13 Start action.....	91
	6.14 Stop action.....	91
	6.15 Continue action.....	92
	6.16 Delay action.....	92
	6.17 Delay case action.....	92
	6.18 Send action.....	93
	6.19 Receive case action.....	94
	6.20 CHILL built-in routine calls.....	97
7	Input and Output.....	102
	7.1 I/O reference model.....	102
	7.2 Association values.....	104
	7.3 Access values.....	104
	7.4 Built-in routines for input output.....	105
	7.5 Text input output.....	112
8	Exception handling.....	120
	8.1 General.....	120
	8.2 Handlers.....	121
	8.3 Handler identification.....	121
9	Time supervision.....	122
	9.1 General.....	122
	9.2 Timeoutable processes.....	122
	9.3 Timing actions.....	122
	9.4 Built-in routines for time.....	124
10	Program Structure.....	125
	10.1 General.....	125
	10.2 Reaches and nesting.....	127
	10.3 Begin-end blocks.....	129
	10.4 Procedure specifications and definitions.....	129
	10.5 Process specifications and definitions.....	134
	10.6 Modules.....	134
	10.7 Regions.....	135
	10.8 Program.....	135
	10.9 Storage allocation and lifetime.....	136
	10.10 Constructs for piecewise programming.....	136
	10.11 Genericity.....	141

	<i>Page</i>	
11	Concurrent execution.....	144
11.1	Processes, tasks, threads and their definitions.....	144
11.2	Mutual exclusion and regions	145
11.3	Delaying of a thread.....	148
11.4	Re-activation of a thread	148
11.5	Signal definition statements	148
11.6	Completion of Region and Task locations	149
12	General semantic properties.....	149
12.1	Mode rules.....	149
12.2	Visibility and name binding	160
12.3	Case selection.....	167
12.4	Definition and summary of semantic categories	169
13	Implementation options	173
13.1	Implementation defined built-in routines	173
13.2	Implementation defined integer modes	173
13.3	Implementation defined floating point modes.....	173
13.4	Implementation defined process names	173
13.5	Implementation defined handlers.....	173
13.6	Implementation defined exception names.....	173
13.7	Other implementation defined features	173
	Appendix I – Character set for CHILL	175
	Appendix II – Special symbols	176
	Appendix III – Special simple name strings	177
III.1	Reserved simple name strings	177
III.2	Predefined simple name strings.....	178
III.3	Exception names	178
	Appendix IV – Program examples.....	179
IV.1	Operations on integers.....	179
IV.2	Same operations on fractions	179
IV.3	Same operations on complex numbers	180
IV.4	General order arithmetic.....	180
IV.5	Adding bit by bit and checking the result.....	180
IV.6	Playing with dates	181
IV.7	Roman numerals.....	182
IV.8	Counting letters in a character string of arbitrary length.....	183
IV.9	Prime numbers	184
IV.10	Implementing stacks in two different ways, transparent to the user.....	184
IV.11	Fragment for playing chess	185
IV.12	Building and manipulating a circularly linked list	188
IV.13	A region for managing competing accesses to a resource	189
IV.14	Queuing calls to a switchboard	190
IV.15	Allocating and deallocating a set of resources	190
IV.16	Allocating and deallocating a set of resources using buffers	192
IV.17	String scanner1	194
IV.18	String scanner2.....	195
IV.19	Removing an item from a double linked list	196
IV.20	Update a record of a file.....	196
IV.21	Merge two sorted files.....	197
IV.22	Read a file with variable length records.....	198
IV.23	The use of spec modules	199
IV.24	Example of a context.....	199
IV.25	The use of prefixing and remote modules	199

	<i>Page</i>
IV.26 The use of text i/o.....	200
IV.27 A generic stack.....	201
IV.28 An abstract data type.....	202
IV.29 Example of a spec module.....	202
IV.30 Object-Orientation: Modes for Simple, Sequential Stacks.....	202
IV.31 Object-Orientation: Mode Extension: Simple, Sequential Stack with Operation "Top".....	204
IV.32 Object-Orientation: Modes for Stacks with Access Synchronization.....	204
Appendix V – Decommited features.....	206
V.1 Free directive.....	206
V.2 Integer modes syntax.....	206
V.3 Set modes with holes.....	206
V.4 Procedure modes syntax.....	206
V.5 String modes syntax.....	207
V.6 Array modes syntax.....	207
V.7 Level structure notation.....	207
V.8 Map reference names.....	207
V.9 Based declarations.....	207
V.10 Character string literals.....	207
V.11 Receive expressions.....	207
V.12 Addr notation.....	207
V.13 Assignment syntax.....	207
V.14 Case action syntax.....	207
V.15 Do for action syntax.....	207
V.16 Explicit loop counters.....	208
V.17 Call action syntax.....	208
V.18 RECURSEFAIL exception.....	208
V.19 Start action syntax.....	208
V.20 Explicit value receive names.....	208
V.21 Blocks.....	208
V.22 Entry statement.....	208
V.23 Register names.....	208
V.24 Recursive attribute.....	208
V.25 Quasi cause statements and quasi handlers.....	209
V.26 Syntax of quasi statements.....	209
V.27 Weakly visible names and visibility statements.....	209
V.28 Weakly visible names and visibility statements.....	209
V.29 Pervasiveness.....	209
V.30 Seizing by modulation name.....	209
V.31 Predefined simple name strings.....	209
Appendix VI – Index of production rules.....	210

Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work. In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1.

International Standards are drafted in accordance with the rules given in the ISO/IEC Directives, Part 2.

The main task of the joint technical committee is to prepare International Standards. Draft International Standards adopted by the technical committee are circulated to the member bodies for voting. Publication as an International Standard requires approval by at least 75 % of the member bodies casting a vote.

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO shall not be held responsible for identifying any or all such patent rights.

ISO/IEC 9496 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology, Subcommittee SC 22, Programming languages, their environments and system software interfaces*, in collaboration with ITU-T. The identical text is published as ITU-T Rec. Z.200.

This fourth edition cancels and replaces the third edition (ISO/IEC 9496:1998), which has been technically revised.

INTERNATIONAL STANDARD**ITU-T RECOMMENDATION****CHILL – THE ITU-T PROGRAMMING LANGUAGE**

This Recommendation | International Standard defines the ITU-T programming language CHILL. When CHILL was first defined in 1980 "CHILL" stood for CCITT High Level Language.

The following subclauses of this clause introduce some of the motivations behind the language design and provide an overview of the language features.

For information concerning the variety of introductory and training material on this subject, the reader is referred to the Manuals, "Introduction to CHILL" and "CHILL user's manual".

An alternative definition of CHILL, in a strict mathematical form (based on the VDM notation), is available in the Manual entitled "Formal definition of CHILL".

1.1 General

CHILL is a strongly typed, block structured language designed primarily for the implementation of large and complex embedded systems.

CHILL was designed to:

- enhance reliability and run time efficiency by means of extensive compile-time checking;
- be sufficiently flexible and powerful to encompass the required range of applications and to exploit a variety of hardware;
- provide facilities that encourage piecewise and modular development of large systems;
- cater for real-time applications by providing built-in concurrency and time supervision primitives;
- permit the generation of highly efficient object code;
- be easy to learn and use.

The expressive power inherent in the language design allows engineers to select the appropriate constructs from a rich set of facilities such that the resulting implementation can match the original specification more precisely.

Because CHILL is careful to distinguish between static and dynamic objects, nearly all the semantic checking can be achieved at compile time. This has obvious run time benefits. Violation of CHILL dynamic rules results in run-time exceptions which can be intercepted by an appropriate exception handler (however, generation of such implicit checks is optional, unless a user defined handler is explicitly specified).

CHILL permits programs to be written in a machine independent manner. The language itself is machine independent; however, particular compilation systems may require the provision of specific implementation defined objects. It should be noted that programs containing such objects will not, in general, be portable.