
**Programming languages — C —
Extensions to support embedded
processors**

*Langages de programmation — C — Extensions pour supporter les
processeurs intégrés*

Withdrawn

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.

Withdrawn

© ISO/IEC 2004

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

Page

FOREWORD	VI
INTRODUCTION	VII
1 SCOPE	1
2 REFERENCES	1
3 CONFORMANCE	1
4 FIXED-POINT ARITHMETIC	2
4.1 Overview and principles of the fixed-point data types	2
4.1.1 The data types	2
4.1.2 Spelling of the new keywords	3
4.1.3 Overflow and Rounding	4
4.1.4 Type conversion, usual arithmetic conversions	5
4.1.5 Fixed-point constants	6
4.1.6 Operations involving fixed-point types	7
4.1.7 Fixed-point functions	9
4.1.8 Fixed-point definitions <stdfix.h>	11
4.1.9 Formatted I/O functions for fixed-point arguments	11
4.2 Detailed changes to ISO/IEC 9899:1999	12
5 NAMED ADDRESS SPACES AND NAMED-REGISTER STORAGE CLASSES	37
5.1 Overview and principles of named address spaces	37
5.1.1 Additional address spaces	37
5.1.2 Address-space type qualifiers	37
5.1.3 Address space nesting and rules for pointers	38
5.1.4 Standard library support	39
5.2 Overview and principles of named-register storage classes	39
5.2.1 Access to machine registers	39
5.2.2 Named-register storage-class specifiers	39
5.2.3 Ensuring correct side effects via objects allocated in registers	41
5.2.4 Relationship between named registers and I/O-register designators	41
5.3 Detailed changes to ISO/IEC 9899:1999	41
6 BASIC I/O HARDWARE ADDRESSING	49
6.1 Rationale	49
6.1.1 Basic Standardization Objectives	49

6.2 Terminology	49
6.3 Basic I/O Hardware addressing header <iohw.h>	51
6.3.1 Standardization principles	51
6.3.2 The abstract model	52
6.4 Specifying I/O registers	54
6.4.1 I/O-register designators	54
6.4.2 Accesses to individual I/O registers	54
6.4.3 I/O register buffers	55
6.4.4 I/O groups	56
6.4.5 Direct and indirect designators	57
6.4.6 Operations on I/O groups	57
6.5 Detailed changes to ISO/IEC 9899:1999	58
ANNEX A - FIXED-POINT ARITHMETIC	65
A.1 Fixed-point datatypes	65
A.1.1 Introduction	65
A.2. Number of data bits in _Fract versus _Accum	68
A.3 Possible Data Type Implementations	69
A.4 Overflow and Rounding	70
A.5 Type conversions, usual arithmetic conversions	71
A.6 Operations involving fixed-point types	72
A.7 Exception for 1 and -1 Multiplication Results	72
A.8 Linguistic Variables and unsigned _Fract: an example of unsigned fixed-point	73
ANNEX B - NAMED ADDRESS SPACES AND NAMED-REGISTER STORAGE CLASSES	74
B.1 Embedded systems extended memory support	74
B.1.1 Modifiers for named address spaces	74
B.1.2 Application-defined multiple address space support	75
B.1.3 I/O register definition for intrinsic or user defined address spaces	76
ANNEX C - IMPLEMENTING THE <IOHW.H> HEADER	78
C.1 General	78
C.1.1 Recommended steps	78
C.1.2 Compiler considerations	78
C.2 Overview of I/O Hardware Connection Options	79
C.2.1 Multi-Addressing and I/O Register Endianness	79
C.2.2 Address Interleaving	80
C.2.3 I/O Connection Overview:	80
C.2.4 Generic buffer index	81

C.3.	I/O-register designators for different I/O addressing methods	82
C.4	Atomic operation	83
C.5	Read-modify-write operations and multi-addressing cases.	83
C.6	I/O initialization	84
C.7	Intrinsic Features for I/O Hardware Access	85
ANNEX D - MIGRATION PATH FOR <IOHW.H> IMPLEMENTATIONS		86
D.1	Migration path for <iohw.h> implementations	86
D.2	<iohw.h> implementation based on C macros	86
D.2.1	The access specification method	86
D.2.2	An <iohw.h> implementation technique	87
D.2.3	Features	87
D.2.4	The <iohw.h> header	88
D.2.5	The user's I/O-register designator definitions	91
D.2.6	The driver function	92
ANNEX E - FUNCTIONALITY NOT INCLUDED IN THIS TECHNICAL REPORT		93
E.1	Circular buffers	93
E.2	Complex data types	94
E.3	Consideration of BCD data types for Embedded Systems	94
E.4	Modwrap overflow	94
ANNEX F - C++ COMPATIBILITY AND MIGRATION ISSUES		96
F.1	Fixed-point Arithmetic	96
F.2	Multiple Address Spaces Support	96
F.3	Basic I/O Hardware Addressing	96

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 joint technical committee are circulated to national bodies for voting. Publication as an International Standard requires approval by at least 75 % of the national bodies casting a vote.

In exceptional circumstances, the joint technical committee may propose the publication of a Technical Report of one of the following types:

- type 1, when the required support cannot be obtained for the publication of an International Standard, despite repeated efforts;
- type 2, when the subject is still under technical development or where for any other reason there is the future but not immediate possibility of an agreement on an International Standard;
- type 3, when the joint technical committee has collected data of a different kind from that which is normally published as an International Standard ("state of the art", for example).

Technical Reports of types 1 and 2 are subject to review within three years of publication, to decide whether they can be transformed into International Standards. Technical Reports of type 3 do not necessarily have to be reviewed until the data they provide are considered to be no longer valid or useful.

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

ISO/IEC TR 18037, which is a Technical Report of type 2, was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 22, *Programming languages, their environments and system software interfaces*.

Introduction

In the fast growing market of embedded systems there is an increasing need to write application programs in a high-level language such as C. Basically there are two reasons for this trend: programs for embedded systems become more complex (and hence are difficult to maintain in assembly language), and processor models for embedded systems have a decreasing lifespan (which implies more frequent re-adapting of applications to new instruction sets). The code re-usability achieved by C-level programming is considered to be a major step forward in addressing these issues.

Various technical areas have been identified where functionality offered by processors (such as DSPs) that are used in embedded systems cannot easily be exploited by applications written in C. Examples are fixed-point operations, usage of different memory spaces, low level I/O operations and others. The current proposal addresses only a few of these technical areas.

Embedded processors are often used to analyze analogue signals and process these signals by applying filtering algorithms to the data received. Typical applications can be found in all wireless devices. The common data type used in filtering algorithms is the fixed-point data type, and in order to achieve the necessary speed, embedded processors are often equipped with special hardware for fixed-point data. The C language (as defined in ISO/IEC 9899:1999) does not provide support for fixed-point arithmetic operations, currently leaving programmers with no option but to handcraft most of their algorithms in assembly language. This Technical Report specifies a fixed-point data type for C, definable in a range of precision and saturation options. Optimizing C compilers can generate highly efficient code for fixed-point data as easily as for integer and floating-point data.

Many embedded processors have multiple distinct banks of memory and require that data be grouped in different banks to achieve maximum performance. Ensuring the simultaneous flow of data and coefficient data to the multiplier/accumulator of processors designed for FIR filtering, for example, is critical to their operation. In order to allow the programmer to declare the memory space from which a specific data object must be fetched, this Technical Report specifies basic support for multiple address spaces. As a result, optimizing compilers can utilize the ability of processors that support multiple address spaces, for instance, to read data from two separate memories in a single cycle to maximize execution speed.

As the C language has matured over the years, various extensions for accessing basic I/O hardware (*iohw*) registers have been added to address deficiencies in the language. Today almost all C compilers for freestanding environments and embedded systems support some method of direct access to *iohw* registers from the C source level. However, these extensions have not been consistent across dialects.

This Technical Report provides an approach to codifying common practice and providing a single uniform syntax for basic *iohw* register addressing.

Programming languages — C — Extensions to support embedded processors

1 Scope

This Technical Report specifies a series of extensions of the programming language C (as specified by ISO/IEC 9899:1999) to support features commonly found in embedded processors. It deals with the following topics: extensions to support fixed-point arithmetic, named address spaces, and basic I/O hardware addressing.

Each clause in this Technical Report deals with a specific topic. The first subclauses of clauses 4, 5 and 6 contain a technical description of the features of the topic. These subclauses provide an overview but do not contain all the fine details. The last subclause of each clause contains the editorial changes to ISO/IEC 9899:1999 necessary to fully specify the topic in ISO/IEC 9899:1999, and thereby provides a complete definition. Additional explanation and rationale are provided in the Annexes.

2 References

The following referenced documents are indispensable for the application of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

ISO/IEC 9899:1999, *Programming languages – C*