



# DSP-C and Embedded C Extensions to DWARF

ACE Associated Compiler Experts by

Version: 2008.4  
Date: February 7, 2008  
Status: release  
Confidentiality: public  
Reference: CoSy-8117P-dwarf

## **DSP-C and Embedded C Extensions to DWARF**

by ACE Associated Compiler Experts bv.

© Copyright 1998-2005,2007-2008 by ACE Associated Compiler Experts bv,  
Amsterdam, the Netherlands.

© Copyright 1998-2005,2007-2008 by ACE Associated Computer Experts bv,  
Amsterdam, the Netherlands.

All rights reserved. No part of this document may be copied, photocopied, reproduced or translated in any way, without prior written consent of ACE Associated Compiler Experts bv.

Every care has been taken in manufacturing the supplied product and its documentation. ACE Associated Compiler Experts bv will neither assume responsibility for any damages caused by the use of its products, nor accept warranty or update claims, unless stated explicitly otherwise in a special agreement.

The information contained in this document is subject to change without notice.

Printed in the Netherlands, March 19, 2008.

Many of the designations used by manufacturers and vendors to distinguish their product are trademarks. ACE Associated Compiler Experts bv has made every attempt to supply trademark information of manufacturers and their products mentioned in this document. ACE Associated Compiler Experts bv also recognizes any trademarks used in this document but not mentioned below.

### **Trademark notices**

CoSy<sup>®</sup> is a registered trademark of ACE Associated Computer Experts bv.

---

# Acknowledgments

Supporting the DSP-C language in the DWARF debugging formats has been realized by close co-operation between ACE and its industrial partners.

The *Multiple Source File Support* extension described in this document has been made possible by Philips Semiconductors NV.

# Contents

<b>1</b>	<b>Introduction</b>	<b>5</b>
<b>2</b>	<b>DWARF 1.0</b>	<b>6</b>
2.1	GENERAL DESCRIPTION . . . . .	6
2.1.1	Attribute Types . . . . .	6
2.1.2	Location Information . . . . .	6
2.1.3	Type Attributes . . . . .	7
2.1.4	Modified Types . . . . .	7
2.2	DEBUGGING INFORMATION ENTRIES . . . . .	7
2.2.1	Compilation Unit Entries . . . . .	7
2.2.2	Program Variable Entries . . . . .	8
2.2.3	Space Translation Table . . . . .	8
2.3	DATA REPRESENTATION . . . . .	8
2.3.1	Attribute Types . . . . .	8
2.3.2	Location Atoms . . . . .	9
2.3.3	Fundamental Types . . . . .	9
2.3.4	Type Modifiers . . . . .	10
2.4	Source Languages . . . . .	10
2.5	Address Range Table . . . . .	10
2.6	Space Translation Table . . . . .	10
<b>3</b>	<b>Multiple Source File Support</b>	<b>11</b>
3.1	Compilation Unit Entries . . . . .	11
3.2	Line Number Table . . . . .	12
<b>4</b>	<b>DWARF 2.0</b>	<b>13</b>
4.1	Compilation Unit Entries . . . . .	13
4.2	Base Type Entries . . . . .	13
4.3	Type Modifier Entries . . . . .	14
4.4	Memory spaces . . . . .	14
	<b>Bibliography</b>	<b>15</b>

# Chapter 1

## Introduction

In 1997, an international partnership between industrial partners, led by ACE, embarked on extending the ISO C language definition to include support for facilities as found on many DSP processor architectures, such as fixed point data types, circular buffer support and multiple memory spaces. This extension has become a de facto standard, under the name DSP-C (see the *DSP-C Definition Document* [2]). Moreover, the extension has been input to the ISO standardization committee, leading to an extended version of ISO-C, commonly known as Embedded C (see *Embedded C Definition*) [6]. For a comparison of both languages, check *Embedded C, Implementation Comparison to DSP-C* [3].

Although the DSP-C and Embedded C language extensions allow DSP programs to be written in a high level language ensuring portability while being able to be compiled efficiently, it does not cater for debugging programs in the development stage. For this, it was needed to define how to represent the language extensions in symbolic debug information formats of DWARF.

The DSP release of the CoSy[1] compiler development system supports DSP-C and Embedded C debugging information in the extended DWARF formats. The CoSy product is commercially available for compiler development from ACE Associated Compiler Experts by (<http://www.ace.nl>).

This document describes the extensions defined for two versions of the DWARF symbolic debug specification. For the remainder of this document, DWARF 1.0 format refers to the DWARF format version 1, as defined in *DWARF Debugging Information Format* [4]. When referring to the DWARF 2.0 format, the DWARF format version 2 is meant, as defined in *DWARF Debugging Information Format* [5].

Next to the extensions specific for DSP-C and Embedded C support, this document describes an addition made to the DWARF 1.0 format, enabling the compiler to represent line number information originating from more than a single file.

## Chapter 2

# DWARF 1.0

In this section, the extensions to the existing DWARF 1.0 are defined to support the DSP-C and Embedded C languages. Wherever possible, the section names are the same as the corresponding sections in the *DWARF Debugging Information Format* document (see [4]).

## 2.1 GENERAL DESCRIPTION

### 2.1.1 Attribute Types

The table of attribute types allows user-defined names to be added. Table 2.1 indicates which names are added.

AT_ACE_space	AT_ACE_spaces
--------------	---------------

Table 2.1: Attribute names

### 2.1.2 Location Information

The table of location atoms allows user-defined atoms to be added. Table 2.2 indicates which atoms are added. The `OP_ACE_SADDR` atom is a new form of a location atom, defined as: An

OP_ACE_SADDR
--------------

Table 2.2: Location atoms

”`saddr(spaceidx)`” atom indicates that the address on the stack is a relocated or relocatable address within the space *spaceidx*. The specified space *spaceidx* attribute is a 2-byte argument value.

### 2.1.3 Type Attributes

The table of fundamental types allows user-defined types to be added. Table 2.3 indicates which types are added.

FT_ACE_short_fixed	FT_ACE_signed_short_fixed
FT_ACE_unsigned_short_fixed	FT_ACE_fixed
FT_ACE_signed_fixed	FT_ACE_unsigned_fixed
FT_ACE_long_fixed	FT_ACE_signed_long_fixed
FT_ACE_unsigned_long_fixed	FT_ACE_short_accum
FT_ACE_signed_short_accum	FT_ACE_unsigned_short_accum
FT_ACE_accum	FT_ACE_signed_accum
FT_ACE_unsigned_accum	FT_ACE_long_accum
FT_ACE_signed_long_accum	FT_ACE_unsigned_long_accum
FT_ACE_long_long	FT_ACE_signed_long_long
FT_ACE_unsigned_long_long	

Table 2.3: Fundamental types

### 2.1.4 Modified Types

The table of type modifiers allows user-defined modifiers to be added. Table 2.4 indicates which modifiers are added.

MOD_ACE_saturate	MOD_ACE_circular
MOD_ACE_space	MOD_ACE_modwrap

Table 2.4: Type modifiers

## 2.2 DEBUGGING INFORMATION ENTRIES

### 2.2.1 Compilation Unit Entries

The table of language names defines an additional language, known as Embedded C. This is to differentiate the recognition of DSP-C versus Embedded C implementation, where a debugger should show different type names for e.g. the fixed point data types (e.g. in DSP-C, `__fixed` and `__accum`, where in Embedded C the types are named `_Fract` and `_Accum`). For DSP-C, no language number was added. The occurrence of a fixed-point fundamental type, in combination with `LANG_C` or `LANG_C89` can be used to determine whether the language is DSP-C.

LANG_ACE_EMBEDDED_C
---------------------

Table 2.5: Language names

A new table, the space translation table, is added (see Section 2.2.3), defining the translation of memory space attributes, memory space names and addressing methods. Under normal circumstances, the same table will be present in every module of an application (e.g. because all modules are compiled with the same compiler). The extension does however not require the tables to be identical across all modules (e.g. suppose a 'generic' compiler mode which is compatible, but defines a different space table).

A reference to this translation data concerning the compilation unit is made using the `AT_spaces` attribute of which the value is a reference to a table of memory space translations.

The space translation table is optional. When the table is present, all `AT_ACE_space` attributes and `OP_ACE_SADDR` location atoms specify an index in this table. When the table is not present, the meaning of the `AT_ACE_space` attributes and `OP_ACE_SADDR` location atoms are target implementation defined.

## 2.2.2 Program Variable Entries

The location of a variable may be specified with a `AT_ACE_space` attribute, specifying in which memory space the object is defined.

For the meaning of this attribute value, see Section 2.2.1.

## 2.2.3 Space Translation Table

Objects are located in memory. DWARF 1.0 assumes one contiguous memory space in which all objects and code is located.

A compiler may allow a flexible way to handle slightly different memory configurations, where the mapping of names to addressing modes are specified at compilation time.

A debugger must address the different configurations used, and needs to know what the locations of objects are, what memory space names are used, and how these map to addressing methods to read from memory.

The DWARF 1.0 format does not specify this relation, and since the configuration must be flexible, the information can best be added to a new section `.debug.space` in the object format. The `.debug` section contains a reference to this information (`AT_ACE_spaces`).

The space translation table (see also 2.6) is an optional part of this extension. When not present, space references get a target implementation defined meaning (see also 2.2.1).

## 2.3 DATA REPRESENTATION

### 2.3.1 Attribute Types

The encoding of additional attributes is specified in Table 2.6.

Attribute name	Form	Value
AT_ACE_space	halfword	(0x2000 FORM_DATA2)
AT_ACE_spaces	word	(0x2010 FORM_DATA4)

Table 2.6: Additional attribute encodings

### 2.3.2 Location Atoms

Additional location atoms are specified in Table 2.7.

Atom name	Value
OP_ACE_SADDR	0xe0

Table 2.7: Location atom encodings

### 2.3.3 Fundamental Types

New type encodings are given in Table 2.8.

Type name	Value
FT_ACE_short_fixed	0x8000
FT_ACE_signed_short_fixed	0x8001
FT_ACE_unsigned_short_fixed	0x8002
FT_ACE_fixed	0x8003
FT_ACE_signed_fixed	0x8004
FT_ACE_unsigned_fixed	0x8005
FT_ACE_long_fixed	0x8006
FT_ACE_signed_long_fixed	0x8007
FT_ACE_unsigned_long_fixed	0x8008
FT_ACE_short_accum	0x8009
FT_ACE_signed_short_accum	0x800A
FT_ACE_unsigned_short_accum	0x800B
FT_ACE_accum	0x800C
FT_ACE_signed_accum	0x800D
FT_ACE_unsigned_accum	0x800E
FT_ACE_long_accum	0x800F
FT_ACE_signed_long_accum	0x8010
FT_ACE_unsigned_long_accum	0x8011
FT_ACE_long_long	0x8012
FT_ACE_signed_long_long	0x8013
FT_ACE_unsigned_long_long	0x8014

Table 2.8: Type encodings

### 2.3.4 Type Modifiers

Modifier name	Value
MOD_ACE_saturate	0x80
MOD_ACE_circular	0x81
MOD_ACE_space	0x82
MOD_ACE_modwrap	0x83

Table 2.9: Type modifier encodings

Type modifiers are specified in Table 2.9. The `MOD_ACE_space` modifier has a 2-byte argument, specifying the memory space index value, when the space table section is present. When the space table is not present, the value has a target implementation defined meaning.

## 2.4 Source Languages

The new added language is defined in Table 2.10.

Language name	Value
LANG_ACE_EMBEDDED	0x8000

Table 2.10: Language encodings

## 2.5 Address Range Table

The address range table represents addresses which need to be attributed with memory space information, similar to the way the new location atom `OP_ACE_SADDR` is introduced to attribute location atoms with information to specify the memory space, .

The tuples in this table are redefined, and consist of a 2-byte space specification, an address (in the size appropriate for the given architecture) and a 4-byte length.

Note that the presence of this table is defined optional in the DWARF 1.0 specification.

## 2.6 Space Translation Table

Each set of entries in the table of space translations contained in the `.debug_space` section consists of a 4-byte offset into the `.debug` section followed by a series of tuples. Each tuple consists of a 2-byte index value, followed by a string of non-null bytes terminated by one null byte (memory space name), followed by a 4-byte addressing mode value. Each set is terminated by a 2-byte halfword containing the value 0.

## Chapter 3

# Multiple Source File Support

The DWARF 1.0 format can only represent lines within a single module, whose source name is defined in the `TAG_compile_unit`. If code in the compiled module originates from include files, then this line information cannot be represented. Typically, this is caused by code originating from `#include`'d files or by using the `#line` directive in the program source.

A second issue which is not covered by the DWARF 1.0 definition, occurs when code can be reordered by a locator program. The DWARF 1.0 format assumes that the generator of the debug information knows the first code address in a module and all code locations can be defined using an unsigned offset value from that point.

### 3.1 Compilation Unit Entries

A `TAG_compile_unit` entry is extended to own one or more `TAG_source_file`<sup>1</sup> entries.

A source file entry has the following attributes:

1. An `AT_sibling` attribute.
2. An `AT_low_pc` attribute whose value is the relocated address of the first machine instruction represented by the accompanying line number information.
3. An `AT_high_pc` attribute whose value is the relocated address of the first location past the last machine instruction represented by the accompanying line number information.
4. An `AT_name` attribute whose value is a null-terminated string containing the full or relative path name of the source file to which the accompanying line number information applies.
5. An `AT_stmt_list` attribute whose value is a reference to a table of line number information.

---

<sup>1</sup> The tags `TAG_compile_unit` and `TAG_source_file` are identical in the DWARF 1.0 definition, but are used here to differentiate their uses

While scanning through the sequence of instructions, whenever the next instruction implies a change in source file, a new `TAG_source_file` including a private line number table is started. The `TAG_source_file` entries may not be nested.

The program code covered by a `TAG_source_file` must be consecutive in memory. For situations where the debug information generator is uncertain whether the covered code is a consecutive range <sup>2</sup>, it should also introduce a new `TAG_source_file`.

### 3.2 Line Number Table

The contents of the Line Number Table is not extended. The only extension made is that for every `TAG_source_file` declared within the `TAG_compile_unit`, a separate Line Number Table will be present.

---

<sup>2</sup> For example, a code generator may place every procedure in a program within a separate code section. In this case it depends on the locator algorithm whether the next procedure begins on the next address or not. It is advised to start a new `TAG_source_file`

## Chapter 4

# DWARF 2.0

This chapter shows the additions to DWARF 2.0 [5] to support the DSP-C (see the *DSP-C Definition Document* [2]) and Embedded C languages.

### 4.1 Compilation Unit Entries

A new language name is added, in order to allow debuggers to distinguish an DSP-C application from an Embedded C implementation. The latter main importance is because of the representation differences of the used type modifiers, and the difference in the type promotion system. For DSP-C, no language number was added. The occurrence of a fixed-point type, in combination with `DW_LANG_C` or `DW_LANG_C89` can be used to determine whether the language is DSP-C.

<code>DW_LANG_ACE_EMBEDDED_C</code>	Embedded C, including ACE extensions	0x8000
-------------------------------------	--------------------------------------	--------

Table 4.1: Additional Language Names

### 4.2 Base Type Entries

Two new base types are added as indicated in Table 4.2.

Name	Meaning	Value
<code>DW_ATE_ACE_signed_fixed</code>	signed Fixed Point type	0x80
<code>DW_ATE_ACE_unsigned_fixed</code>	unsigned Fixed Point type	0x81

Table 4.2: Additional Base Type Entries

More information must be given to specify Fixed Point types than e.g. an integer type. For this reason, a new attribute is introduced.

A Fixed Point type entry has a `DW_AT_byte_size` attribute, whose value is a constant, describing the size in bytes of the storage unit used to represent an object of the given type.

If the value of an object of Fixed Point type does not fully occupy the storage unit described by the byte size attribute, the base type has a `DW_AT_bit_size` attribute. The bit size attribute describes the actual size in bits to represent the value of the Fixed Point type (i.e. including the sign, integral and scale information).

Additionally, a Fixed Point type has a `DW_AT_ACE_scale` attribute, whose value is a constant, describing the size in bits of the scale of the Fixed Point type (see Table 4.3).

Attribute name	Value	Classes
<code>DW_AT_ACE_scale</code>	0x2000	constant

Table 4.3: Attribute Encoding

### 4.3 Type Modifier Entries

Three new modifiers are added as indicated in Table 4.4.

Tag	Meaning	Value
<code>DW_TAG_ACE_saturated_type</code>	<code>_Sat</code> qualified type	0x4080
<code>DW_TAG_ACE_circular_type</code>	<code>_Circ</code> qualified type	0x4081
ACE reserved modifier number		0x4082
ACE reserved modifier number		0x4083
<code>DW_TAG_ACE_modwrap_type</code>	<code>_Modwrap</code> qualified type	0x4084

Table 4.4: Additional Type Modifier Entries

The qualifiers in above table represent the notation as used in the Embedded C language. In DSP-C, these are written `__sat`, `__circ`. DSP-C does not define anything similar to the `_Modwrap` keyword.

### 4.4 Memory spaces

The support of memory spaces is implemented by using the existing attributes `AT_segment` and `AT_address_class` in the following way.

The `AT_segment` attribute is used at points where a location of an object is described. The value of the attribute is a target defined number identifying the memory space where the object is located.

The `AT_address_class` attribute is used within pointer type definitions. Again, the value of the attribute is a target defined number identifying the memory space which the pointer points to.

# Bibliography

- [1] ACE Associated Compiler Experts bv. CoSy Compiler Development System, <http://www.ace.nl/products/cosy.htm>. 1994-2002.
- [2] ACE Associated Compiler Experts bv. DSP-C, An extension to ISO/IEC IS 9899:1990. Ref. CoSy-8025-dsp-c, 2005.
- [3] ACE Associated Compiler Experts bv. Embedded C, Implementation Comparison to DSP-C. Ref. CoSy-8141-EmbeddedC, 2008.
- [4] UNIX International. Dwarf debugging information format. Technical Report 1.1.0, UNIX International, Waterview Corporate Center, 20 Waterview Boulevard, Parsippany, NJ 07054, October 1992.
- [5] UNIX International. Dwarf debugging information format. Technical Report 2.0.0, UNIX International, Waterview Corporate Center, 20 Waterview Boulevard, Parsippany, NJ 07054, July 1993.
- [6] ISO/IEC. ISO/IEC TR 18037, Programming languages – C – Extensions to support embedded processors. 2008.