occam-2 compiler specification

Conor O'Neill and Stephen Clarke

SW-0062-3

INMOS Limited Confidential

APPROVED 19 June, 1990

Contents

1 Change history . . . 2
1.1 Changes since issue SW-0062-02 . . . 2
1.2 Changes since issue of January 24, 1990 . . . 2
2 Important information . . . 2
2.1 Changed implementation of channels . . . 2
2.2 UNDEFINED and UNIVERSAL error modes . . . 2
2.3 #SC directive . . . 2
3 Introduction . . . 3
4 Host implementation details . . . 3
5 Processor types . . . 4
6 Error modes . . . 4
7 Interactive debugging . . . 4
8 Filenames . . . 5
9 Locating Files . . . 5
10 Command line specification . . . 6
11 Compiler library . . . 8
12 Diagnostics . . . 9
12.1 List of warnings . . . 9
13 Related Documents . . . 10

1 Change history

1.1 Changes since issue SW-0062-02

- Added more command line options.

1.2 Changes since issue of January 24, 1990

- Added 'Important information' section.
- Added description of UNIVERSAL error mode.
- Added **L**, **XO**, **XM**, and **ZI** command line switches.

2 Important information

There are some major differences between this compiler and the previous occam toolset compiler (D705B etc).

2.1 Changed implementation of channels

The low level behaviour of channels has changed. This will not affect programmers working entirely in occam. However, programmers using **GUY** and **ASM** code, and **PLACE**ing channels, must be aware of the change. It may also affect programmers using **KERNEL.RUN** and its associated predefines.

2.2 UNDEFINED and UNIVERSAL error modes

The 'error mode', and the type of errors which are detected, have now been 'de-coupled'. Thus you can now run in HALT mode, but turn off most checks, etc, and it is possible to write code which will execute in either HALT or STOP mode, but still correctly trap errors.

UNDEFINED error mode is no longer supported. It can be mimicked by using the new **U** command line switch, which turns off the insertion of any run-time error checks. Hence a program compiled in HALT mode, but with this command line flag, will behave as if in occam UNDEFINED error mode.

UNIVERSAL error mode has changed its behaviour. It now is compiled to behave exactly like HALT or STOP error mode, according to whether the transputer's *HaltOnError* flag is set. It *does not* turn off all error checks like it did in the past. To do this, you should also add the **U** command line switch. Thus **X** and **U** together now mimic the old UNIVERSAL mode.

2.3 #SC directive

The **#SC** compiler directive is no longer supported. (See SW-0044 ("occam 2 Language Implementation Manual")).

3 Introduction

The occam 2 compiler is called **oc**.

The occam 2 compiler takes as input an occam 2 source file, which may refer to further source files, and compiles it into a TCOFF format (see SW-0011), or LFF format (see SW-0010), binary object file. Which format is determined when the compiler is built: it is not selectable at compiler runtime (TCOFF is the preferred).

Command line options control the target transputer and error mode.

A target processor should be specified for each compilation. The compiler supports IMS T212, T222, T225, M212, T400, T414, T425, T800, T801 and T805 processors, together with the TA, TB and TC processor classes. (See SW-0032 (“User Interface Specification”)). The compiler default is to produce code for the T414.

A target error mode should be specified for each compilation. The compiler supports HALT, STOP and UNIVERSAL error modes. The compiler default is to produce code for HALT mode.

The compiler can compile in a mode which supports interactive debugging with the INMOS debugger. Alternatively it can compile using inline sequences of transputer instructions for input and output, resulting in faster code execution.

The occam 2 source is checked for conformancy to the occam 2 language definition (see SW-0044 (“occam 2 Language Implementation Manual”)). Some extensions to the language definition in the “occam 2 Reference Manual” published by Prentice-Hall are defined. In particular, directives to include source code from other files, to reference separately compiled files and libraries, and to select compiler options.

Assembly language code inserts are also allowed by means of a language extension. They are only permissible if selected by a compiler option.

If the source does not conform to the occam 2 language definition, then the compiler will issue diagnostics, in the form of error messages, during the compilation process, and no object module file will be produced.

occam 2 source files can contain references to object code libraries, occam source to be included in the compilation, separately compiled occam code, and code produced by compilers for other languages.

Libraries and separately compiled units must be already compiled before any file which references them can itself be compiled. It is the programmer’s responsibility to ensure all components of a program are compiled in the correct order and that object code is kept up to date with changes in the source. This may be assisted by using a MAKE program in conjunction with the **imakef** tool. The compiler inserts directives into the object file to enable the linker to check that the correct modules are linked together.

4 Host implementation details

The occam 2 compiler is written in a subset of ANSI standard X3.159 C, using a subset of the X3.159 run-time library, such that it may be compiled with the following compilers: *VAX-C* under VAX-VMS, *gcc* under SunOS version 4 (both on Sun-3 and Sun-4), and *icc* (the Inmos C compiler). The occam 2 compiler is to run on SunOS, VMS and MS-DOS (?) systems, as well as 32-bit transputers attached to a host and communicating using iserver protocol (see SW-0070).

The input files must be in the host's native C system text file format; ie. sequences of ASCII characters separated by newline characters on SunOS, sequences of ASCII characters separated by carriage-return/linefeed pairs on MS-DOS, and sequences of records containing ASCII characters in stream format with a linefeed record separator and implied carrriage-return on VAX-VMS.

5 Processor types

The occam 2 compiler follows the occam toolset rule for combining processor types (see SW-0032 ("User Interface Specification")):

> Code may be called provided it is compiled for the same type or class, or it is compiled for a class which is a superset of the calling code.

6 Error modes

The occam 2 compiler follows the occam toolset rule for combining error modes (see SW-0032 ("User Interface Specification")):

> Code compiled in HALT mode may call code compiled in HALT or UNIVERSAL modes.
> Code compiled in STOP mode may call code compiled in STOP or UNIVERSAL modes.
> Code compiled in UNIVERSAL mode may call code compiled in UNIVERSAL mode.

Code compiled in UNIVERSAL mode behaves as HALT or STOP mode, according to the state of the transputer's *HaltOnError* flag.

The compiler supports the UNDEFINED (or REDUCED) error mode by use of the **`U`** command line switch, which prevents the compiler from inserting any explicit code for run time checks where it couldn't check the validity at compile time. This enables it to be used as a 'go faster' option, for cases where the code is known to be error free. Note that some checks are still performed; some transputer instructions implicitly check for erroneous conditions. For example, arithmetic overflow doesn't need any explicit checking, but is checked by the normal instructions.

7 Interactive debugging

The occam 2 compiler supports interactive debugging by default. When enabled, the compiler will generate calls to library routines to perform input and output, rather than using the transputer's instructions. It may be disabled by using the **`Y`** command line switch, which forces the compiler to use sequences of transputer instructions for input and output. When interactive debugging is selected, the compiler assumes the existence of a library called **`virtual.lib`**. See SW-0063 ("occam 2 Compiler Library Specification").

Interactive debugging must be enabled in order to use the interactive features of the debugger.

Code which has interactive debugging disabled may call code which has interactive debugging enabled, but not vice-versa.

8 Filenames

occam source files can be given any filename which is legal for the given host system. The use of the **.occ** extension for occam source, and the **.inc** extension for files containing declarations of constants and protocols, is recommended. If a source filename extension is not specified on the command line, then **.occ** will be assumed.

Output files are specified using the '**O**' option. If an output filename is not specified, the input filename is used and a **.tco** extension is added in place of any existing extension.

The compiler uses the extension of the output file as a default filename extension in a **#USE** directive.

If the Makefile generator tool **imakef** is used to assist with version control, the following extensions, dependant upon the target error mode and processor type *must* be used for the object file (see SW-0060 ("Makefile generator specification")).

Error Mode	T212 T222 M212	T225	T414	T425 T400	T800	T801 T805	TA	TB	TC
HALT	.t2h	.t3h	.t4h	.t5h	.t8h	.t9h	.tah	.tbh	.tch
STOP	.t2s	.t3s	.t4s	.t5s	.t8s	.t9s	.tas	.tbs	.tcs
UNIVERSAL	.t2x	.t3x	.t4x	.t5x	.t8x	.t9x	.tax	.tbx	.tcx

9 Locating Files

The occam 2 compiler locates files by searching a specified *directory path* on the host system. The path is specified under MS-DOS and SunOS by the environment variable **ISEARCH**, and under VAX-VMS, by the logical name **ISEARCH**. The name **ISEARCH** may be overridden by a command line option. For the syntax of **ISEARCH**, see SW-0032 ("User Interface Specification").

The search rules are:

1 If the filename contains a directory specification then the filename is used as given. Relative directory names are treated as relative to the directory in which the compiler is invoked.

2 If no directory is specified, the directory in which the compiler is invoked is assumed.

3 If the file is not present in the current directory, the path specified by the environment variable (or logical name) **ISEARCH** is searched. If there are several files of the same name on this path, the first occurrence is used.

4 If the file is not found using the above rules, then the file is assumed to be absent, and an error is reported.

If no search path has been set up then only rules 1 and 2 apply.

10 Command line specification

The command line takes the form

oc *filename* { *options* }

filename is the name of the file containing the source code. If no file extension is specified, the extension **.occ** is assumed. The file is searched along the **ISEARCH** path; see section 9. If the filename is omitted the compiler displays brief help information.

options is a list, in any order, of one or more of the options given in the following table. Each option must be preceded by '**/**' for MS-DOS and VMS based toolsets, or '**–**' for all others. Options must be separated by spaces. Options are not case sensitive.

Option	Description
A	Prevents the compiler from performing alias checking. The default is to perform alias checking. This option also disables usage checking. Note that when enabled, the compiler may insert run-time alias checks. Details of alias and usage checking rules are given in SW-0044 ("occam 2 Language Implementation Manual").
B	Displays messages in brief (single line) format.
C	Disables the generation of object code. The compiler performs syntax, semantic, alias and usage checking only.
D	Generates minimal debugging information. The default is to produce debugging data. Debugging data is required by the debugger and by the transputer simulator.
E	Disables the use of the compiler's libraries. This prevents the compilation of some programs which require 'complicated' arithmetic such as real arithmetic on a processor which does not have a floating point unit.
G	Enables the compiler to recognise the restricted range of transputer instructions, via the **GUY** and **ASM** constructs. See SW-0044 ("occam 2 Language Implementation Manual") for the list of permitted instructions.
H	Produces code in HALT error mode. This is the default compilation mode and may be omitted for HALT error mode programs.
I	Displays additional information as the compiler runs. This information includes target and error mode, and information about directives as they are processed. The default is not to display this information.
K	Disables run-time range checking. The default is to insert run-time range checking.
L	Simply load the compiler, and do nothing. Used when loading the compiler onto a transputer system.
N	Disables usage checking. The default is to perform usage checking. Usage checking is also disabled by option '**A**'. Details of usage checking rules are given in SW-0044 ("occam 2 Language Implementation Manual").
NWP	Do not warn if parameters are not used.
NWU	Do not warn if variables or routines are not used.
O *outputfile*	Specifies the name of the output file. If no output file is specified the compiler uses the input filename and adds the file extension **.tco**.
R *filename*	Redirects error and information messages to a file.

Option	Description
S	Produces code in STOP error mode.
U	Disables the insertion of all extra run-time error checking. The default is to insert run-time error checks. This is a 'stronger' option than **K**, and can be used to implement the occam UNDEFINED error mode. See section 6.
V	Prevents the compiler from producing code which has a separate vector space requirement. The default is to produce code which uses separate vector space. See SW-0044 ("occam 2 Language Implementation Manual") for details of vectorspace usage.
W	Enables the compiler to recognise the full range of transputer instructions, via the **GUY** and **ASM** constructs. See SW-0044 ("occam 2 Language Implementation Manual") for the complete list of instructions.
WD	Provide a warning whenever a name is descoped.
WO	Provide a warning whenever a run-time alias check is generated (Overlap checks).
X	Produces code in UNIVERSAL error mode.
XO	Compile a single program only. Used when the compiler is loaded onto a transputer system.
XM	Compile many programs. The compiler will loop, accepting multiple command lines from the server. Used when the compiler is loaded onto a transputer system.
Y	Disables interactive debugging with **idebug**. See section 7.

Option	Description
TA	Compile for transputer class TA .
TB	Compile for transputer class TB .
TC	Compile for transputer class TC .
T212	Compile for a T212 processor.
T222	Compile for a T222 processor. Same as **T212**.
M212	Compile for a M212 processor. Same as **T212**.
T2	Same as **T212**.
T225	Compile for a T225 processor.
T3	Same as **T225**.
T400	Compile for a T400 processor. Same as **T425**.
T414	Compile for T414 processor. This is the default processor type and may be omitted when compiling for a T414 processor.
T4	Same as **T414**.
T425	Compile for a T425 processor.
T5	Same as **T425**.
T800	Compile for a T800 processor.
T8	Same as **T800**.
T801	Compile for a T801 processor. Same as **T805**.
T805	Compile for a T805 processor.
T9	Same as **T805**.
N.B. These options are mutually exclusive. If mutually exclusive options are given an error is reported.	

Compiler diagnostic options	
Option	**Description**
Z	Print out a help page including these diagnostic options.
ZA	Print out code generator diagnostics and assembly code.
ZB	Print out assembly code.
ZC	Run compiler as far as semantic checker only (no usage checking or code generation).
ZD	Print out disassembled code (after code expansion).
ZE	Use visible compiler names rather than hiding them by appending a percent sign.
ZH	Mark the object file as 'occam harness' rather than 'occam', for use by the C style configurer.
ZI *pathname*	Use this *pathname* instead of **ISEARCH** for locating files. See section 9.
ZL	Print out lexical analyser diagnostics.
ZN	Disable all the predefined routines.
ZO	Print out interspersed source and assembly code.
ZP	Run compiler as far as parser only (no checking or code generation).
ZQ	Do not insert origin checks for imported references.
ZR	Disable generation of origin checks for exported references.
ZS	Allocate workspace by name scope, not name usage.
ZT	Print out the intermediate applicative expression tree.
ZU	Run compiler as far as usage checker only (no code generation).
ZV	Perform channel i/o by instructions, not by routine calls. Used in conjunction with **Y** option.
ZW	Do not implement arrays of channels as arrays of pointers to channels. This forces compatibility with the previous (**D705B**) toolset occam compiler.
ZX	Force 8 byte library patch size.
ZY	Force the code to be given priority when linked. Equivalent to the LINKAGE pragma. See SW-0044 ("occam 2 Language Implementation Manual").
ZZ	Compile PRI PAR using a special ALT construct.

11 Compiler library

When compiling code which contains extended type operations or conversions, or certain compiler predefined routines, the compiler makes reference to routines which are assumed to exist in a compiler library.

SW-0063 ("occam 2 Compiler Library Specification") describes the required contents of the compiler library, and the conditions under which each routine will be called.

12 Diagnostics

Compiler diagnostics take the form of warning, error and fatal error messages written to the standard error stream under VAX-VMS and SunOS, and the standard output stream under MS-DOS. These streams normally default to the user's terminal.

The compiler error messages are of the format:

Warning-oc-*filename***(***linenumber***)-***message*

Warning messages are generated whenever legal, but unorthodox, programming styles are detected. They do not prevent generation of an object file.

Error-oc-*filename***(***linenumber***)-***message*

Recoverable errors are generated whenever the compiler detects a programming error which it can repair itself. No object file is produced; the compiler will continue, but at termination it will return an error code; this allows correct termination of **make** or batch files.

Fatal-oc-*filename***(***linenumber***)-***message*

Fatal errors terminate compilation immediately. No object file is produced. The compiler will return an error code.

12.1 List of warnings

Badly formed #PRAGMA *name* **directive**
The pragma directive does not conform to the required syntax.

name **is not used**
The named variable is never used. This warning may be disabled by means of the **NWU** command line switch.

Name *name* **descopes a previous declaration**
This name descopes another name which has already been declared. This warning is only enabled by means of the **WD** command line switch.

No compatible entrypoints found in *name*
The named library contains no routines which may be called from this error mode and/or processor type.

Parameter *name* **is not used**
The named parameter is never used. This warning may be disabled by means of the **NWP** command line switch.

Placement expression for *name* **clashes with interactive debugger**
The named variable is placed on one of the transputer links. This may interfere with the INMOS interactive debugging system.

Placement expression for *name* **wraps around memory**
The calculation of the machine address for this variable has overflowed; the truncated address is used.

Routine *name* **imported by multiple #USEs**

The named routine exists in two different libraries; an implementation restriction means that this is not permitted.

Routine *name* **is not used**
The named routine is never called. This warning may be disabled by means of the **NWU** command line switch.

Run-time disjointness check inserted
number **run-time disjointness checks inserted**
The compiler has inserted run-time checks to ensure that variables are not aliased (ie that they don't overlap). This warning is only enabled by means of the **WO** command line switch.

TRANSLATE ignored: String contains NULL character
The specified string for a TRANSLATE pragma may not contain a NULL (zero) byte.

TRANSLATE ignored: Name *name* **has already been used**
You may not specifiy multiple translation strings for the same name.

TRANSLATE ignored: String *name* **has already been used**
You may not specify multiple names to be translated to the same string.

TRANSLATE ignored: Module containing *name* **has already been loaded**
The TRANSLATE pragma must precede any #USE of a library containing that string.

Unknown #PRAGMA name: *name*
The pragma name is ignored.

Workspace clashes with variable PLACED AT WORKSPACE *number*
A variable has been PLACED AT WORKSPACE *number*, and this clashes either with another placed variable, or with the compiler's workspace allocation requirements.

13 Related Documents

SW-0044 ("occam 2 Language Implementation Manual") describes the source language accepted by the compiler, and the implementation restrictions imposed by the compiler, and the standard libraries provided with the occam 2 system.

SW-0064 ("occam 2 Run time model") describes the run-time environment of an occam program, including memory allocation.

SW-0063 ("occam 2 Compiler Library Specification") describes the library routines required by the compiler.

SW-0031 ("Toolset architecture and overview").

SW-0032 ("User Interface Specification").