inmos®

IMS D4305A Sun4
IMS D6305A VAX/VMS
IMS D7305A IBM 386 PC
occam 2 Toolset
Release Notes

SGS-THOMSON MICROELECTRONICS

INMOS is a member of the SGS–THOMSON Microelectronics Group

INMOS document number: 72 TDS 365 02

Contents

1 Release notes

1.1 Introduction

This occam 2 toolset is a replacement for the existing Dx205 toolsets. It supports the same language and features as before, but in addition, there is extra support for embedded systems applications.

This document describes the major differences, provides general installation information, and lists the known errors. These notes are intended as a guide; the manuals define the functionality in detail.

1.1.1 Licenses

Some variants (e.g. D4305A) of this product are supplied with a licence manager. Please read your delivery manual before attempting to install and use the software.

1.1.2 Notation

For ease of reading, all options are given in the form `-x` as they would be written for UNIX systems. For MS-DOS and VAX/VMS, the corresponding option would be written as `/x`.

1.2 Hosts

The toolset is supplied in three variants:

D4305A Sun 4 running SunOS 4.1.1 or later.

D6305A VAX under VMS.

D7305A for IBM-PC 386 compatibles with a minimum of 4M bytes running DOS 5.

The INQUEST Toolset (Dx300B) is designed to operate in conjunction with the Dx305 toolset.

Note: This and later versions of the toolset do not support either the Sun 3 or the PC-AT 286. However, the D5205 (Sun 3) and D7205 (PC-AT 286) will continue to be available for some time. Please contact your distributor for further details.

1.3 `iserver` – important changes

The version of the host server supplied with this release has changed considerably since the previous version.

There is just one program per toolset and you no longer need to decide which one to use. To handle the different possible methods of accessing transputers, it now obtains information from a connection database. This must exist and its name provided in the environment variable `ICONDB`. Examples of such a database are supplied under the `connect` subdirectory and you should edit a suitable one for your installation.

The `TRANSPUTER` environment variable has changed its meaning, and its value must be changed to reflect the entry in the connection database that you wish to use by default.

Please see the chapter on the `iserver` in the *occam 2 Toolset Reference Manual.*

1.4 Language extensions and changes

The language supported continues to be occam 2 as defined in the *occam 2 Reference Manual.*

1.4.1 Language additions and clarifications

In some cases there was some ambiguity or shortcoming on the definition. Some of these have been resolved.

- Usage and alias checking pragmas. New pragmas have been implemented which permit the programmer to specify that particular variables break the usage and/or alias checking rules. A meaning has been given to programs which use this technique, so that a future compiler which performs more aggressive optimization will not break programs which abide to the appropriate rules.
- It is now possible to name a `PROTOCOL` of type `ANY`; simple `CHAN OF ANY` is now obsolescent. `CHAN OF ANY` is currently accepted but may be removed in future releases. A warning is issued whenever it is seen, however this warning may be disabled by means of a compiler command line switch.
- A new `#PRAGMA COMMENT` has been added.
- New facilities have been added for the `ASM` construct to facilitate low level programming.

1.4.2 Changes from previous version

- In a counted array channel input, the length and the array are now considered to be received simultaneously. Therefore no part of the array expression, e.g. the length of a slice, may depend on the length of the counted array communication.

- Many fixed size limits inside the compiler have now been removed. E.g. it is now possible to have any number of variables in a single procedure. A command line switch has been added to control the maximum size of object file which can be produced.

1.5 Virtual routing configurer

The configurer in this toolset contains the equivalent functionality to the virtual routing configurer as provided in the product Dx224A. This enables the user to write a configuration connecting channels in the application without regard to the number of hardware links available. The configurer will insert extra processes that will do the multiplexing and routing of messages to support the requested channel connections. The user also has some control over which links are used for this multiplexing support.

The configurer supports a command line switch to force it to behave in the same way as the Dx205 configurer.

The debugger `idebug` has been modified to make it easier to debug programs that have been configured with the virtual routing software.

1.6 Embedded systems help

1.6.1 Memory map file

A new tool `imap` is provided which can produce a complete record of where all procedures and functions reside in memory on every node of a network. It operates by reading map files produced by the compiler, linker and collector and computing the locations of all the known names.

It produces a list for each processor, the contents of the memory in increasing order of address, stating what the sections of memory are for.

Then it lists all the known names in alphabetical order showing where they reside on each node. Optionally, the names which are internal to the occam compiler support library can be suppressed, in order to reduce the size of the listings.

The listings from the compiler and linker have been changed in order to supply `imap` with the information it needs. In addition, the workspace requirements of each procedure and function are given as are the positions of local variables within the workspace.

1.6.2 Placing code and data

The configurer has new features to enable the user to place code and data areas at specified locations in memory. It can also be told not to use the on-chip memory and leave it free for the application.

A new attribute is added to a processor, `reserved :=` *size*. This reserves *size* bytes from the lowest address and neither any part of the application nor its bootstrap will be placed in the reserved area automatically. Hence, an application can ensure that the tools do not use on-chip memory, or any other contiguous area starting at *mint*. This attribute is applied to each processor in a network independently. The default is to reserve no memory.

Three other new attributes are added to a processor:

```
location.ws := addr
location.vs := addr
location.code := addr
```

These specify the exact location, by address, of each of the workspace, vectorspace and code respectively. They take precedence over the `order` attribute which retains its meaning.

These attributes are placed in the mapping section of the configuration and are used in an analogous manner to similar ones found in `icconf` of the Dx314 product.

1.6.3 Bootstrap sources

The sources of all the bootstraps and network loaders are provided with the toolset. These are commented in full, so that the user can modify them if necessary.

Some of the sources are in assembler. The user will need the Dx314 product in order to assemble them if changes to them are required.

1.7 Sundry improvements

`INT16` arithmetic on 32-bit processors has been improved, to be both faster and more compact.

The T426 is now supported by all tools.

Some additions have been made to what is allowed in the ASM construct.

`idebug` can now debug boot-from-ROM, run-in-RAM programs.

1.8 Cautions

There are two hosts that are discontinued with this release, viz. the IBM PC (286-AT compatible plus TRAM, XT etc.) and the Sun 3.

The option `-t` on the collector is strongly deprecated. The INQUEST toolset will not support programs that have been generated using it. Nor will the forthcoming

T9000 toolset. The configuration systems for both the T9000 and the T-series will converge in the next release, when the provision of `-t` on the collector will be discontinued. To a large extent, there are already alternative ways of getting equivalent functionality in this release. Any shortcomings will be remedied in the next release.

Special care must be taken when using the occam 2 toolset Dx305 and the old ANSI C toolset Dx214 to perform mixed language programming. If this old version of the ANSI C toolset is being used, then virtual channels must NOT be used. Also, post-mortem debugging is still possible for mixed language programs, but interactive debugging cannot be done. You must use the `callc.lib` library from the Dx205 release.

A new version of the ANSI-C toolset (Dx314) has recently been released which is fully compatible with this version of the occam 2 toolset. This contains a new copy of the `callc.lib` library.

On MS-DOS, the previous toolsets used a driver called `BANSI.SYS`. This has been removed, along with `pcbansi.itm`, since it is incompatible with later versions of both DR-DOS and MS-DOS.

The tools `icvlink` and `icvemit` have been removed from the toolset, as have all the TDS conversion tools.

The use of `GUY` for assembly inserts is strongly deprecated. The compiler gives a warning message when it finds `GUY` and the description has been removed from the manuals. `ASM` is the recommended method of incorporating assembly code.

There is a change in the default manner of booting programs from link. The Dx205 toolset used whatever space was available from low memory, Dx305 loads its bootstraps and loader into high memory, as defined by the configuration language '`memsize`' attribute. To retain the Dx205 behavior, a new option `-bm` has been added to the collector.

When collecting with the `-t` option, the value of `IBOARDSIZE` must not be greater than the actual memory present on the hardware. When collecting with the `-t` and `-m` options, the value of the `-m` option must not be greater than the actual amount of memory present.

The `F` folding editor has been removed. Contact your supplier to obtain the latest version of `F`.

1.9 Installation

1.9.1 If you are replacing an existing occam 2 toolset

This toolset completely replaces the previous occam toolsets D4205A, D6205A and D7205A. The installation of this toolset will generate a new directory and subdirectories. The older one should be deleted.

If you have made any changes to the previous toolset structure, these should be recorded and re-implemented in the new structure.

1.9.2 If you are using occam 2 and ANSI–C together

The occam and ANSI-C toolsets have some common components, such as the linker, librarian, debugger, eprom tools, and the system library `sysproc.lib`. The installation procedures for each toolset create separate directory structures. However, if you are intending to mix languages, it is important to know which version of the tools to use.

When choosing a tool, *always use the latest version.* Tools are designed to be *backwards compatible* so that they will work with previous releases of the toolsets. A tool will not necessarily work in conjunction with tools from a later toolset, especially if there is a later version of the tool in the later toolset.

Consult the documentation in the Release Notes for each toolset to determine which is the later release. If you wish to use both toolsets, put the *later* release on your search path *before* the earlier one. If you decide to combine the tools into a common directory, then, for the tools and libraries which are in both toolsets, always use the later one.

1.9.3 Transputer bootable files

This toolset does not contain any of the tools as transputer bootables except for `idebug`, `iskip` and `idump`. These three need a transputer to execute.

None of the other tools are provided as transputer executable files. They execute on the host.

For an IBM PC compatible host, D7205 relied on a transputer to execute the compiler. This is not true in D7305; the requirements of the PC are now that it has an Intel 386 or 486 compatible processor, runs DOS 5.0 or later, and has at least 4 MB of memory.

1.10 Library source

If you wish to rebuild or customize the occam 2 libraries the following information will be of use. It assumes that you are experienced in the use of the toolset.

To determine which routines in which error mode and for which transputer types are in a library, study the output of `ilist` on the library in question, e.g. `ilist string.lib -e`. This will also show whether the sources were compiled with the `Y` option or not. All supplied libraries are built compiled with minimum debug data included (the `D` option on `oc`). The maths libraries `snglmath.lib`, `dblmath.lib` and `tbmaths.lib` are built with vector space and run-time checks disabled (the `V` and `U` options on `oc`). `crc.lib` is built with run-time checks disabled (the `U` option on `oc`).

This is, in general, enough information to rebuild a library. However, one complication arises. In some libraries, `hostio.lib` for example, some of the routines use other routines in the same library. In such a case the lower level routines must be compiled first and made into a (temporary) library which the other routines then `#USE` in order to access them. Finally, once all the library components have been successfully compiled, the whole library can be built, and the temporary library can be deleted.

The libraries which are built in the manner described above are **hostio.lib**, **tbmaths.lib**, and **convert.lib**. To rebuild one of these libraries you need to proceed as follows. Examine the source code of the library routines; if you see the `#USE` of a library with an unknown name, it is one of these temporary libraries (e.g. **utilstb.lib** in **tbmaths.lib** sources). By looking through the sources of the routines using this library, you can determine its required contents. Construct an appropriate **.lbb** file to make the temporary library. Then compile the appropriate routines and combine them into a library using **ilibr**. You can then compile all the routines which depend on this library. Proceed until everything has been compiled successfully, and then use **ilibr** to combine everything together to make the desired library.

Note: that some whole libraries depend upon others (e.g. **hostio.lib** depends on **convert.lib**). Therefore if you rebuild **convert.lib**, you will also need to rebuild **hostio.lib**. The '**.liu**' files for the libraries list the dependencies.

1.11 Transputer error flag

The driver program for the transputer hosted **iskip** tool monitors the error flag as the tool executes in order to catch any internal errors of the tool should they occur. If your hardware is configured as a **Down** system and consists of more than one transputer, the driver program may be fooled into thinking that **iskip** has set the error flag, if the error flag on one of the extra processors is already set when the tool is executed.

In order to overcome this problem, you should run a network check program, one may have been supplied by your hardware vendor, or boot a (dummy) program that uses all of the processors in the network.

Note: that once cleared, an error flag on a transputer will only become set again if you execute an erroneous program on the transputer or you power on the transputer again.

1.12 Bugs fixed

The following bugs in previous versions of the toolset are now fixed.

TS/1179 If the argument to `ASSERT` was an `INLINE FUNCTION`, which expanded to a constant, then it was never tested.

TS/1205, TS/1294 Various problems with counted array communication.

TS/1206 Compiler inserted incorrect opcode for *fpuclrerr* in T800 STOP mode.

TS/1243 The version of `ATAN2` in `tbmaths.lib` returned *-pi* rather than *+pi* when given arguments `( negative.number, -0.0(REAL32) ).`

TS/1251 Compiler was overly strict on indentation of continuation lines.

TS/1254 Nested replicated `ALT`s, nested inside other `ALT`s, could cause the compiler to crash.

TS/1257, TS/1263 Various problems with constructed tables.

TS/1274 Complicated double length integer expressions could be compiled incorrectly.

TS/1276 The compiler didn't trap writes to free variables of functions used in an abbreviation.

TS/1307 `INLINE` procedures in configuration source code could cause arbitrary errors.

TS/1308 `MOVE2D`, where the first argument is a subscript of an array, failed.

TS/1315 Complicated `INT64` to `REAL64` conversions broke.

TS/1324 Array abbreviations took an incorrect size when the real size was a 'non-local' variable.

TS/1333 Double length comparisons against negative constants were incorrect.

TS/1340 Assignment of multi-dimensional `INT16` array elements failed.

TS/1361 Incorrectly permitted associativity of `AND` and `OR`.

TS/1367 `INT16` to `BYTE` conversions on 32-bit processor failed to correctly sign-extend the source.

TS/1405 Complicated floating point expressions involving some constants could overflow the floating point stack.

TS/1439 Segments evaluated as part of the result of a `VALOF` could be incorrectly evaluated.

TS/1467 `BYTE` sized `PORT`s caused a *word* read or write on the transputer memory bus.

TS/1495, TS/1604, TS/1730 Various problems with multi-dimensional channel arrays in the configurer.

TS/1522 Segments used on the right hand side of an `OR` may not have been range checked correctly.

TS/1546 Relational operators in double length expressions could cause the compiler to fail.

TS/1571 `INT64 AFTER` was calculated incorrectly on 16-bit processors.

TS/1586 The configurer could fail on networks of mixed 16-bit and 32-bit processors.

TS/1606 Storing a result of a multiple-result function directly into an array element could fail.

TS/1618 Complicated subscripting of constant tables could cause the configurer to fail.

TS/1626 When the length of a slice to be input was itself a 'complicated' expression, the compiler could generate incorrect code.

TS/1632 Floating point expressions with common sub-expressions could fail.

TS/1649, TS/1855 On a T225, incorrect code was generated when subscripting an array whose elements were of type `INT32` or `REAL32`.

TS/1687 When references were made from `.pgm` files to files on remote file systems, then the names generated for those files by `imakef` in makefiles and linker indirect files were sometimes incorrect.

TS/1723 If abbreviations appeared before the timer input in an `ALT`, incorrect code could be generated.

TS/1771 Zero length slices could cause a compiler failure.

TS/1751 Abbreviations and `RETYPES` of `PLACED` channels failed.

TS/1893 If a function used the replicator variable of a replicated ALT as a free variable, i.e. not passed as a parameter, and that function was called as part of a guard of that ALT, or as part of the channel expression, then the function did not obtain the correct value of the replicator variable.

TS/1900 Complicated expressions in ALT guards were sometimes evaluated incorrectly.

TS/1905 On a T2 as the root processor, attempts to boot from ROM and run in ROM did not always work.

TS/1907 `REAL32` arithmetic on TB class processors in STOP mode now acts correctly for overflow and divide by zero.

TS/1932, TS/2008 The configurer calculated incorrect values when passing constant arrays of INTs, or when RETYPING INTs, on a 16-bit processor, It also failed to detect passing out-of-range values. These errors are now caught at configuration time.

TS/1962 The library routines `REAL32GT`, `REAL32EQ`, `REAL64GT` and `REAL64EQ` did not always give the correct result when supplied with a `NaN`.

TS/1965 Some assignments to segments of arrays were not range checked properly.

TS/1968 `INT16` operations in STOP error mode did not check for error correctly.

TS/1970 `MOSTNEG INT64` divided (or remainder) by 1 on a 16-bit processor failed.

TS/1972 `MOSTNEG INT16` to `REAL64` conversion failed on 16-bit processors.

TS/1980, TS/2055 Some illegal `RETYPES` were permitted.

TS/1981 Conversion of small denormalized `REAL64`s to `REAL32`s on 16-bit processors is now correct.

TS/1982 In some cases a `PRI PAR` was permitted when nested inside another `PRI PAR`.

TS/1995, TS/2006 STOP mode code on T8 processors could fail if another process set the floating point error flag.

TS/2020 `MOSTNEG INT`*xx* divided by or remainder by `-1` was treated incorrectly.

TS/2052 The EPROM programming software has been improved.

TS/2060 The `U` compiler switch incorrectly disabled `ASSERT` run-time checks.

TS/2066 Abbreviations of replicated PAR replicators gave the wrong value.

TS/2075 `STRINGTOINT` and associated routines returned a wrong value when illegal characters were found in a string which began with a minus sign.

1.13 Errata

1.13.1 occam compiler — oc

TS/1407 The behavior of the compiler when constant folding floating point expressions involving NaNs at compile-time is not exactly the same as that of the transputer at run-time.

TS/1984 The compiler permits processes of different priorities to read the same timer.

TS/2093 The compiler does not support segments of constant arrays.

INSdi01337 `ASSERT(false)` will not stop in STOP error mode if the 'U' compiler option is specified.

1.13.2 Configurer — `occonf`

INSdi01578 The configurer doesn't check that code has been compiled with interactive debugging enabled when virtual channels are being used at configuration level.

INSdi01583 If a configuration uses an (illegal) link number between 4 and 31, the configurer crashes.

1.13.3 Collector — `icollect`

INSdi01579 The collector fails when trying to find its libraries for a STOP mode unconfigured ROM program.

1.13.4 Simulator — `isim`

INSdi01340 The simulator incorrectly sets the (simulated) error flag when calculating:

```
(MOSTNEG INT) REM (-1).
```

1.13.5 Debugger — `idebug`

TS/1477 The object files for certain types of processor are identical to those of others, and the debugger cannot distinguish them. If the program is configured, there is no problem. If you *must* use the non-configured case (i.e. `-t` option to the collector), then do the following for debugging:

For T222 there is no problem. The debugger can treat a T222 as a T212, which is what the object file is marked as.

For T400, T426 Compile the code in TB mode, and use `idebug`'s `-c` switch to specify T400 or T426. Re-compile for T400 or T426 when debugged.

For T801 Compile the code in TA mode, and use `idebug`'s `-c` switch to specify T801. Re-compile for T801 when debugged.

TS/1862 Low priority user processes using software virtual links are promoted temporarily to a high priority process as they perform the communication operation. If you locate to such a process via the `idebug` Monitor page - 'Display software virtual links option', `idebug` cannot tell if they were originally at high or at low priority: `idebug` will always locate to what it believes is a high priority process. For further details consult the `idebug` reference chapter in the *occam 2 Toolset Reference Manual*, where the Monitor page '`Z`' option is described.

1.13.6 File lister — `ilist`

TS/1477 The file lister cannot distinguish between some processor types when listing some file formats. In these cases, the T400 and T426 processors are shown as a T425; the T801 is shown as a T805 and the T222 as a T212. The code the compilers generate is identical for these alternative processor types and so the object files do not mark any further differences.

1.13.7 Makefile generator — `imakef`

TS/1544 The makefile generator may produce a core dump or other system error when building a makefile for a `.liu` file when the `.lbb` file is missing.

1.13.8 File server — `iserver`

IS/0051 If a string is written to the screen, or a file, using the `write.string.nl` routines, and the string contains a NUL character (ASCII code 0), then that is used as the end of the string.

IS/0053 When a request is made for an environment variable, whose content is longer than 507 bytes, the server attempts to send more bytes in the return message than the runtime systems can accept. Typically, the error flag on the transputer will be set.

The maximum length of an environment variable should be 507 bytes.

IS/0057 The iserver doesn't accept command lines greater than 256 characters.

1.13.9 Libraries

TS/1292 If `so.getenv` is used to read an environment variable which is bigger than 507 bytes, `spr.operation.failed` is sometimes returned instead of

`spr.buffer.overflow`.

INSdi01337 `DEBUG.ASSERT(false)` in STOP mode will not stop when `idebug` is not being used.