

# IMS D4214B, D5214B, D7214C ANSI C toolset

## Release Notes


**READ THIS FIRST**



INMOS is a member of the SGS-THOMSON Microelectronics Group

Copyright © INMOS Limited 1991. This document may not be copied, in whole or in part, without prior written consent of INMOS.

The C compiler implementation was developed from the Perihelion Software "C" Compiler and the Codemist Norcroft "C" Compiler.

<sup>®</sup>, **inmos**<sup>®</sup>, IMS and occam are trademarks of INMOS Limited.

INMOS is a member of the SGS-THOMSON Microelectronics Group.

INMOS Document Number: 72 TDS 293 00

# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	Changes to the product	1
1.2	Validation	1
<b>2</b>	<b>Changes from previous release</b>	<b>3</b>
2.1	General additions to the product	3
2.2	Changes to tools and libraries	3
2.2.1	Changes to the ANSI C compiler icc	3
	Additions	3
	Enhancements	4
	Bugs Fixed	4
2.2.2	Changes to the ANSI C Libraries	6
	Bugs Fixed	6
2.2.3	Changes to the linker ilink	7
	Additions	7
2.2.4	Changes to the configurer icconf	8
	Enhancements	8
	Bugs Fixed	9
2.2.5	Changes to the collector icollect	10
	Additions	10
	Enhancements	10
	Bugs Fixed	10
2.2.6	Changes to the debugger idebug	11
	Command Line Options	11
	Monitor page	12
	INSPECT / MODIFY Expressions	12
	Error Messages	14
	Miscellaneous	14
	Bugs Fixed	15
	Documentation Omissions	15
2.2.7	Changes to the Makefile generator imakef	16
	Additions	16
	Enhancements	16
2.2.8	Changes to the T425 simulator isim	16
	Additions	16
	Bugs Fixed	16
2.2.9	Changes to the EPROM program converter ieprom	17
2.2.10	Changes to the processor skipping program iskip	17
2.2.11	Changes to the host file server iserver	17
	Enhancements	17
	Bugs Fixed	17
2.2.12	Changes to other tools	18

2.3	NEC PC specific changes	18
2.3.1	New keyboard layout	18
<b>3</b>	<b>Installation</b>	<b>21</b>
3.1	Introduction	21
3.1.1	If you are replacing an existing ANSI C toolset	21
3.1.2	If you are using C and occam toolsets together	21
3.1.3	Installing the iserver	21
3.1.4	Transputer hosted tools	21
3.1.5	New and changed directories	22
3.2	Server installation	22
3.2.1	PC server	22
3.2.2	Sun 3/Sun 4 servers	22
<b>4</b>	<b>Errata</b>	<b>23</b>
4.1	General	23
4.2	Tools and libraries	24
4.2.1	icc	24
4.2.2	C libraries	26
4.2.3	icconf	26
4.2.4	icollect	26
4.2.5	imakef	27
4.2.6	isim	27
4.2.7	iemit	27
4.2.8	iserver	27
4.3	Documentation errors	27
4.3.1	Reference Manual	27
4.3.2	User Manual	30
	<b>Appendices</b>	<b>33</b>
<b>A</b>	<b>Distribution kit</b>	<b>35</b>
A.1	Differences in directory contents lists	35
A.1.1	Differences for D4214B and D5214B from D4214A and D5214A	35
	Directories removed	35
	Directories added	35
A.1.2	Differences for D7214C from D7214B	36
	Directory structure	36
	NEC directory removed	36
	Distribution kit	36
<b>B</b>	<b>ispy program</b>	<b>39</b>
B.1	Network worm – ispy	39

B.1.1	Explanation of output .....	40
<b>C</b>	<b>Multibyte functions</b> .....	<b>41</b>
C.1	Introduction .....	41
C.2	Implementation .....	41
C.3	List of functions .....	42
	mblen .....	42
	mbstowcs .....	43
	mbtowc .....	44
	wcstombs .....	45
	wctomb .....	46



# 1 Introduction

This version of the INMOS ANSI C toolset is an update on the IMS D4214A, D5214A and D7214B products. As such, the documentation is the same as for those toolsets, but you should read the following notes to see what changes have been made.

## 1.1 Changes to the product

There are some general additions and enhancements to the product, some changes to the tools and libraries, and some changes in the installation directories, compared to the previous release. This release also includes bug fixes.

General changes and additions to the product are described in section 2.1.

Changes to the tools and libraries are described in section 2.2.

Changes to the installation directories are described in Appendix A.

Changes to the installation procedures, including installation of the iserver, are described in Chapter 3.

NEC specific changes are described in sections 2.3 and A.1.2.

Known problems in this release are described in Chapter 4.

## 1.2 Validation

The earlier versions of the INMOS ANSI C toolset were validated in 1990 against release 2 of the Plum-Hall validation suite. The validation was done by the British Standards Institute (BSI).

This maintenance release has not yet received a validation certificate. However, INMOS claims that this version of the toolset passes all of the tests in the Plum-Hall suite version 2.

This release also passes level 2.01, the first release of the 1991 BSI validation suite. Full validation against the 1991 suite is expected in August 1991, on the anniversary of the original validation.





# 2 Changes from previous release

This chapter describes the changes in this maintenance release from the ANSI C toolsets D4214A, D5214A, and D7214B.

Where appropriate, INMOS reference numbers are given for bugs fixed, in the format TS/*nnn*.

## 2.1 General additions to the product

- A program called `i_spy` is included in the D7214C. This enables networks of transputers to be examined and reset cleanly. Information for the user is included in Appendix B of this document.
- The source of the driver program is included in D7214C. This was an omission in earlier versions of the product.
- The source of the user bootstrap is now found in the examples directory. This was omitted from earlier versions of the product.
- The debugger better supports C++ (IMS D4217A, D5217A and D7217A).

## 2.2 Changes to tools and libraries.

### 2.2.1 Changes to the ANSI C compiler `icc`

#### Additions

- New option `P mapfile`

The compiler may be instructed, via the `P mapfile` option, to produce a map of workspace for each function defined in the file, and a map of the static area of the whole file. This map is written to the file `mapfile`.

For each function definition, a list of all local variables and formal parameters, and their position in the function's workspace is produced. The workspace requirement of the function is also listed; but note that this does not take into account any extra 'below workspace' slots required for input, output, alternation or calls.

Local variables are listed in the order in which they are allocated in workspace, not the order in which they are declared.

For the whole file, a list of all the static variables and their position in the file's static area is produced. The total static data requirement for the file is also listed.

Those static items which are globally visible are marked (`global`).

Static items introduced to allow the code to access external objects are marked `pointer to external object`.

Static items whose name has been modified by the `IMS_translate` pragma are listed under the name that is put into the object file, but are annotated with the message: (`translated from sourcename`), where `sourcename` is the name used in the source file.

- **New option `FS`**

The compiler has a new command line option (`FS`), which directs it to treat right-shifts of signed integers as arithmetic shifts. The default behaviour is, as before, to treat such shifts as logical shifts. This allows correct working of some non-ANSI conforming programs which assume that right shifts of signed values propagate the sign.

### Enhancements

- The `LF` option has been removed from the compiler. `LF` was an undocumented option and has become obsolete.
- Verbose output of assembler tidied up; it now reports code and initialised data sizes.
- Improved code generation for array accesses with multiple subscripts.
- Improved code generation for some cases of conditional (`?:`) operation.
- Improved code generation for some cases of `+` and `-`.

### Bugs Fixed

- Incorrect handling of signed bitfields on T2 fixed.
- (Ref: TS/0601/1169) If a function had no prototype and used float parameters, the workspace positions of all parameters beyond the first float were incorrectly calculated. For example:

```
float f(x, y)
float x, y;
{
    return x + y;
}

int main(void)
{
    float g;
    g = f(1.0, 2.0);
    if (g > 2.9 && g < 3.1)
        return EXIT_SUCCESS;
    else
```

```

    { printf("$$$ Failed: f(1.0, 2.0) = %f\n", g);
      return EXIT_FAILURE;
    }
}

```

This has now been fixed.

- (Ref: TS/0669/1146) Assertion failure when attempting to generate debug information for **typedefs** of function types fixed.
- (Ref: TS/0756) Assembler instruction **break** now correctly recognised in assembler inserts.
- (Ref: TS/0761/0763) Removal of last basic block in a function (under certain conditions) by flowgraph optimisation fixed.
- (Ref: TS/0717/1074) Fatal internal error **Pushing an int with no space left** when generating code for array expressions with **short** subscript for target processor with **fpu** and **dup** instruction fixed. e.g.

```

int a, b; short c; static char d[64];
a = d[c = b];

```

- (Ref: TS/0788) Incorrect code generation for  $i \geq 0$ , where  $i$  is an unsigned int, fixed. (This also caused switches on unsigned ints to be incorrectly coded in some circumstances.)
- (Ref: TS/1022) Incorrect initialisation of static data on T2 (where initial values match the representation of a pointer and are not word-aligned) fixed.
- (Ref: TS/1037) Fatal internal error **Pushing an int with no space left** when generating code for complicated **double** expressions on T2 fixed.
- (Ref: TS/1038) Fatal internal error when taking content of a multi-word object in a void context fixed, e.g.

```

struct complex { double re; double im; };
void p(void)
{
    struct complex cplx;
    (void) (cplx.re);
}

```

- (Ref: TS/1041) Incorrect handling of reads from volatile objects in a void context fixed. e.g.

```

volatile int *s;
*s;

```

(i.e. didn't read \*s previously).

- (Ref: TS/1050) Corrected optimisation which transformed two successive loads into a load and duplicate so that it does not happen when the load is from a volatile object.
- (Ref: TS/1062) Fatal internal error **Literal pool label is uninitialised** when calling a floating-point function with argument 0.0 in some circumstances (target has an fpu, constant is passed half on the register stack and half in workspace, no other part of the calling routine requires a literal pool) fixed.
- (Ref: TS/1078) Fatal internal error **tentative definition confusion** in handling of tentative definitions of static data fixed.
- (Ref: TS/1098) Assembler instruction **opr** now correctly recognised in assembler inserts.
- (Ref: TS/1133) Inaccurate compile-time constant evaluation for double precision subtract and divide operations fixed.
- (Ref: TS/1144) Spurious LINEMARK records in debug information removed.
- (Ref: TS/1149) **>, >=, <, <=** between two negative floats when compiled for TA gave an incorrect answer. Fixed.
- (Ref: TS/1184) Code generation for passing **float** parameters on register stack improved (no longer preevaluates them to a temporary if they are already addressable).
- Fatal internal error **mcrep (array 48)** when code generating for complicated subscript expressions fixed. e.g. previously failed for

```
short i = (*a)[0].s.s2;
```

## 2.2.2 Changes to the ANSI C Libraries

### Bugs Fixed

- (Ref: TS/1147) Some maths routines gave incorrect results in TB mode:
  - sinh** – when compiled for TB gave erroneous answers for arguments the modulus of which was in the range 1.0 to 710.475874. This is now corrected.
  - cosh** – when compiled for TB gave erroneous answers for arguments the modulus of which was less than 710.475874. This is now corrected.

`tanh` – when compiled for TB gave erroneous answers for arguments the modulus of which was in the range 0.5493061443 to 710.475874. This is now corrected.

- (Ref: TS/1124) `printf` gave incorrect behaviour on T2 for `%x`, `%X` and `%o`. This has now been fixed.
- (Ref: TS/764) A minor bug in `realloc` meant that on rare occasions free blocks too small to be reused were added to the free chain. This has now been fixed.
- (Ref: TS/516) The `server_transaction` function imposed a minimum length of 8 bytes on the `server` packet to be sent. This meant that the minimum transaction was 10 bytes including the two bytes for the packet length. The minimum length of `packet` is now reduced to 6 bytes which brings the minimum transaction to 8 bytes. This agrees with the true server minimum of 8 bytes.
- (Ref: TS/1197) The `exit` function automatically closes all files opened with `fopen` before exiting the program. This feature has been extended to also close all files opened using the `open` function. Previously these files were not closed by `exit`.
- (Ref: TS/1219) The functions `ProcInitClean` and `ProcAllocClean` violated the ANSI standard in that their names were visible within the user name space. These names are now hidden from the user until the `process.h` header file is included as for all other non-ANSI names. Note that this means that any programs making use of these functions will require recompilation.

### 2.2.3 Changes to the linker `ilink`

#### Additions

- **New option `EX`**

A new option (`EX`) allows the extraction of modules unlinked. The linker functions as normal except that the output will not be a single linked unit. It will in fact be the concatenation of the component modules that would have made up the linked unit, with each being unaltered. This is used for creating convenient subunits for further linking, or for extraction from libraries.

The linker `U` option and the `#REFERENCE` and `#DEFINE` commands are particularly useful for controlling the content of the output when combined with `EX`:

- `U` will allow the linker to continue even when unresolved references remain. These outstanding references remain unresolved in the output and must be resolved in some subsequent link.

- **#REFERENCE** causes a the linker to pull in the module exporting that name, and hence everything subsequently required by it.
- **#DEFINE** resolves a reference causing the linker not to pull in any module because of that symbol. Remember that there may be other external symbols in that module which may also require this. In the case of OCCAM specific references this is not possible, see below. The actual value used in the **#DEFINE** command is not significant, since no patching is done.

A main entry point need not be specified with the **EX** option and will have no effect.

In the case of OCCAM, for safety reasons the linker will not allow the separation of a module and those modules which it specifically references, they may be considered permanently fused.

## 2.2.4 Changes to the configurer `icconf`

### Enhancements

The following enhancements have been made to the configurer:

- **Improved process startup**

The configurer adds some system processes to start up the user processes. The memory used by these system processes has been significantly reduced, giving a smaller overhead.

- **System process space made available**

The configurer now supports the overlaying of system processes into any user process data areas that are large enough to accommodate the code and work space for the system processes.

This enhancement is available when configuring for boot from link or for boot from ROM.

- **Support for collector pre-patching**

The configurer now supports the ability of the latest collector to pre-patch the initialisation data used by the system processes. This means that the memory required by the system processes for their code and work space is even further reduced. This occurs automatically in all cases when the technique can be applied (i.e. when not interactively debugging).

This enhancement is available when configuring for boot from link or from boot from ROM.

- **More workspace for configurer**

The working memory required by the configurer has been reduced so that it is now possible to configure programs that are up to 30% larger in terms of the number of processes and processors they contain.

- **Improved speed**

The speed of the configurer has been increased by up to 4 times by better internal use of memory and indexing.

- **Improved memory usage checking**

Application memory usage is now checked accurately by the collector and eprom tools. The configurer checks were too pessimistic in their assumptions concerning the amount of memory used by the network loader and the size of the reserved memory for each of the transputer types. Now the configurer makes no such assumptions any more, so its checks will be optimistic. The collector and eprom tools can and do perform these checks.

- **Restriction lifted on connection statement**

A restriction concerning the way connections are used in `place` statements has now been lifted. It is now possible for the user to place a connection name representing a channel connection onto a connection name representing a link connection without needing to know how the `connect` statements defining the connection names were originally defined.

## Bugs Fixed

The following bugs have been fixed:

- The configurer no longer generates superfluous `FILE` statements for system processes in its output file if only 32 bit or 16 bit processors were defined in the network being configured.
- It is now possible to determine from the output file of the configurer whether a `CHANNEL` statement was placed on a `LINK` statement from left-to-right or from right-to-left.
- When creating a bootable file using the `T` option of the collector and the free memory size is passed into the process the size calculated by the system processes at run-time was incorrect. The free memory size calculated was slightly less than that actually available.
- The configurer does not treat the identifier `channel` as a keyword any more, it can now be used as a normal identifier.
- The configurer now prevents the user from specifying an `interface` parameter with the same name as one of its pre-defined attributes.
- Added checks into the configurer to perform a rough memory usage check for ROM memory usage when the `RA` option is specified. This check is optimistic but will still trap excessive memory usage.

## 2.2.5 Changes to the collector `icollect`

### Additions

- **New option `Y`**

A new option `Y` enables the collector to reduce the size of the overhead it applies to C programs. This happens automatically if the program has been configured with `icconf`. The option can only be used with the `T` (boot only) option. It also disables interactive debugging.

In order to support this reduction if booting from ROM, the ROM bootstrap file has been changed.

- **New option `P filename`**

A new option `P filename` has been added to provide a memory map of the final program. This helps the user find how storage is laid out and where various segments of the program reside.

### Enhancements

- **Improved system libraries**

Various optimisations have been applied to `linkboot.lib` and `sysproc.lib` to reduce the size of the bootstraps and system processes.

### Bugs Fixed

- (Ref: TS/0573) Entry offsets of greater than 32k on a PC hosted version of `icollect` are now handled correctly.
- (Ref: TS/0565) The code for rom now is aligned on a 32 bit word boundary. This is a change of specification both for the `ieprom` and the collector tools.
- (Ref: TS/0650) The `I` option produces more meaningful information.
- (Ref: TS/0721) The collector now tells the user the amount of memory used and complains if the memory size specified is too small.
- (Ref: TS/0755) Wherever numbers are accepted by the collector, both decimal and hexadecimal forms are allowed. The hexadecimal form is written with a preceding `#` symbol. Thus `#400` has the value 1024 (decimal). For compatibility with the early INMOS TDS systems, the `#` sign can also be replaced by a `$`.
- (Ref: TS/0839) Status is now returned correctly when "Input file is of incorrect type".
- (Ref: TS/1192) **D7214 PC hosted version only.** Use of a trailing `M` in the memory size given to the `/M` option was not recognised. This is now fixed.
- (Ref: TS/1193) **D7214 PC hosted version only.** The collector used not to be able to find the system libraries when building a bootfile for a boot-from-rom system. This is now fixed.



- (Ref: TS/1199) D7214 PC hosted version only. Use of the /K option to create an .rsc file for a T800 would mark it as for a T425 instead. This is now fixed.
- D7214 PC hosted version only. The PC version of the collector used to fail when ordering of segments was requested of the configurer and the code segment was to be at the lowest address. This has now been fixed. Note that this problem only affected the versions running on the PC as a host; the versions hosted on a transputer did not show it.

## 2.2.6 Changes to the debugger `idebug`

### Command Line Options

Four new command line options have been added to `idebug`: **J** **K** **GXX** **AP**.

- **J** *#hexnumber*

The **J** option takes a hexadecimal digit sequence of up to 16 digits and replicates it throughout the Data regions of a program (Stack, Static, Heap and Vectorspace as appropriate) when breakpoint debugging. The digit sequence *must* be preceded by a hash ('#') character.

**Note:** the bootstrap phase of a program may use some of the Data regions for its own purposes; consequently the pattern may have some holes in it.

Sequences with an odd number of hexadecimal digits will have a assumed hexadecimal prefix digit of 0 (e.g. #5 will be treated as #05).

```
idebug prog.btl -sr -si -b2 -j#AA
```

will load the program for breakpoint debugging and initialise the Data regions of the program to contain the byte pattern **#AA**.

- **K** *#hexnumber*

The **K** option is similar to the **J** option described above with the addition of initialising the Freespace region.

**Note:** The Freespace region is typically much larger than the other Data regions and consequently this option can take a significantly longer time to execute than the **J** option.

**Note:** the bootstrap phase of a program may use some of the Data and Freespace regions for its own purposes; consequently the pattern may have some holes in it.

- **GXX**

This option should always be used when debugging C++ programs with `idebug`.

It is necessary because the current C++ implementation uses a translator to produce C code as an intermediate step and `idebug` is not made fully aware of this.

This option instructs `idebug` to treat a C source code file as a C++ source code file when symbolically debugging it. It also helps to reduce the confusion `idebug` has in the presence of C++ header include files which produce object code.

- **AP**

This option is a replacement for the **A** option when using some other vendors' boards. It does not apply to INMOS hardware and you should contact your supplier to see whether it is applicable to your hardware.

### Monitor page

- The root processor boot link for configured programs is now checked by `idebug` to match that specified to configurer.

In breakpoint mode a warning is also issued before loading the network.

- Monitor page *Link Information* option now states when an edge has been mapped onto a link rather than stating that the link is not connected.
- Monitor page **F** command (select source file) remembers the previous filename or process number (as applicable) used with the option by use of the cursor up key.

You may also edit this string and the program command line string (**J** command) via use of other cursor keys (same as when editing a C expression, see C User Manual section 15.8.3).

- Monitor page `wdesc` register used to display `Invalid` when there was not an active process. This could be confused with an invalid `wdesc` which is flagged by an asterisk \* (i.e. not valid in the context of a user program). It has been changed to display `NotProcess` instead.

### INSPECT / MODIFY Expressions

- **Automatic Expression Pickup**

When **INSPECT** or **MODIFY** is selected, `idebug` will automatically capture the identifier which is underneath the cursor (if any). Modification of the captured expression is then permitted before applying the selected option.

In this release the automatic capture is more eager for simple `struct` / `union` member expressions which contain the `.` and `->` operators only.

This is best illustrated by example. In the following expressions, the cursor is positioned over `baz` when **INSPECT** / **MODIFY** are selected:

Program text	Expression captured
baz	baz
baz.ptr	baz.ptr
*(baz).ptr	baz
*baz.ptr	baz.ptr
baz.ptr->ptr	baz.ptr->ptr
baz.foo.ptr	baz.foo.ptr
baz->foo->ptr	baz->foo->ptr
baz[x].ptr	baz

In addition, for those captured expression which match the the program text, the cursor may be positioned anywhere on the expression before selecting INSPECT / MODIFY.

- **Displaying Large struct/unions**

The output display of `structs` and `unions` is now paged when necessary (in a similar manner to large arrays).

- **Hex Constants**

The Hex constant syntax has been expanded to accept a '%' character after the leading `0x` component of a hex constant. This provides a shorthand mechanism for specifying transputer addresses in a similar manner to that provided in the monitor page. The '%' character adds `INT_MIN` (`MOSTNEG INT`) to the hex constant using modulo arithmetic.

For example, `0x%70` produces the constant value `0x80000070` on a 32 bit transputer and `0x8070` on a 16 bit transputer.

- **De-referencing an integer (address constant indirect)**

You may now dereference a constant expression which has type `int` or `unsigned int` in a C expression when using INSPECT or MODIFY. Normally you may only dereference a pointer: this addition can save you from having to change to the Monitor page to inspect memory locations.

For example, `*0x80000000` (or `*0x%0`) would display the integer at memory location `0x80000000` on a 32 bit processor.

- **Hex Integer Print**

This has been changed to Hex Print: it now displays the hex representation of *floating* types in addition to *integral* types when enabled.

- **Scope Resolution Operator**

This capability has been added to Inspect / Modify for both C and C++ although strictly speaking, it is only defined in C++.

This operator enables you to access a global identifier which has been hidden by a local identifier of the same name. Its use is best illustrated by an example.

Consider the following program:

```
static int    foo = 42;

void example (void)
{
    int    foo = 321;
    debug_stop (); /* program will stop here */
}
```

When executed, the program will stop at `debug_stop ()`.

If you inspect `foo` the value 321 will be displayed.

If you inspect `::foo` the value 42 will be displayed.

## Error Messages

- **Changed error message**

(Probe Go:) Processor number - Invalid processor type  
has changed to:

(ProbeGo:) Processor number - Incorrect processor type  
(Ref: C User Manual page 310).

- **New error message**

**You must specify a transputer type (instead of a class)**

**Meaning:** The program you are trying to debug is for a transputer class (either TA or TB); the debugger needs to know the actual processor type (e.g. T425). You should retry using the debugger with the command line C option to specify the processor type.

## Miscellaneous

- **C library include file `errno.h`**

This header file defines `errno` via a `#define`. Unfortunately, `idebug` does not know about `#define` values: in order to inspect `errno` from within `idebug`, you should type `_IMS_errno` instead.

- **C library parallel process support**

An incorrect debug record in the asynchronous process library (`ProcRun()`, `ProcRunLow()` and `ProcRunHigh()`), has been corrected. Its effect was to disable backtracing from an asynchronous process (Ref. C User Manual section 8.14.12).

- **Support for other languages**

Full support for OCCAM has been added.

- **Support for configurers**

This version of the debugger supports the following INMOS configurers:

<code>occonf</code>	–	supplied with OCCAM toolsets
<code>icconf</code>	–	supplied with C toolsets

### Bugs Fixed

This section describes the bugs found in `idebug` which have been fixed in this release.

- (Ref: TS/0605) Configuration edges were incorrectly converted into `idebug` style virtual communication links in breakpoint mode.
- (Ref: TS/1045) Network dump didn't always switch between processors correctly when producing a network dump file.
- (Ref: TS/1058) Multiple links between processors could cause `idebug` to hang in breakpoint mode.
- (Ref: TS/1114) An attempt to dereference an incomplete type within an Inspect / Modify expression is now trapped.
- (Ref: TS/1212) A right shift (>>) within an Inspect / Modify expression is now performed correctly as a logical shift.

### Documentation Omissions

- C User Manual omits to state in chapter 15 that `idebug` checks variables for correct alignment when using INSPECT / MODIFY (e.g. an `int` must be word aligned).
- C User Manual section 15.8.3 was not explicit enough about the facilities for editing C INSPECT / MODIFY expressions. You may edit expressions by use of the cursor keys and other editing keys.
- C Reference Manual page 88 omits to state that `debug_message ()` will only display the first 80 characters of an individual message.

- C Reference Manual pages 71, 76 omits to state that the `ChanInTimeFail` and `ChanOutTimeFail` functions do not support virtual channels and cannot be used with the breakpoint debugger.

## 2.2.7 Changes to the Makefile generator `imakef`

### Additions

- **New option `NI`**

The `NI` option means that files that are on `ISEARCH` are not put into the Makefile. This means that all the system files are not present, making it much easier to read.

### Enhancements

- `imakef` now puts a line reading `##### IMAKEF CUT #####` in the Makefile. All lines above this will be untouched when the makefile is regenerated. This allows the user to introduce his own macros without them getting overwritten every time `imakef` is run.

## 2.2.8 Changes to the T425 simulator `isim`

### Additions

- **New option `N`**

The `N` option has been added to tell the simulator to do no further option interpretation after that point in the command line. The remainder of the command line is then passed to the simulated program as its command line.

### Bugs Fixed

- (Ref: TS/1048) A block move of zero bytes with invalid source or destination address was trapped by the simulator as being an illegal memory access. The simulator no longer checks the addresses if a zero sized object is being moved. Block moves are used in the `MOVE` instruction as well as being implicitly used in channel communication instructions.
- (Ref: TS/1189) Long shift left or long shift right over  $2 * \text{WORDSIZE}$  bits left the original value unchanged in the A and B registers. It now clears both registers.
- (Ref: TS/1225) Long shift right propagating sign bit from most significant word.
- (Ref: TS/1226) Long division assumed a signed value input when should have been unsigned thus would do incorrect sign stripping of the dividend and sign extension of the result.

- (Ref: TS/1227) `SETJOBREAK` and `CLRJOBREAK` assumed individual flags for high/low priority, now does the same as a real T425: there is just one flag, if set or cleared at high priority the setting is retained only whilst in high priority (the low priority value is restored on return to low priority then propagated back to high priority when next interrupted).
- (Ref: TS/1194) If a high priority process is waiting on an `ALT` and a low priority process tried to do an output to it the high priority process would disappear and the low priority process continue. Also, a high priority process waiting on an `ALT` with a high priority process doing the output to it causes the `ALT` process to re-awake but with a workspace value of `MININT`.
- (Ref: TS/0578) The use of the `XM` option now allows the simulator to remain in the root transputer awaiting more input.

### 2.2.9 Changes to the EPROM program converter `ieprom`

- New option `R`
- Option `R` causes `ieprom` to print out the absolute address of the code reference point. It can be used in conjunction with the `P` (produce memory map) qualifier of `icollect`.

### 2.2.10 Changes to the processor skipping program `iskip`

- New option `RP`

This is a new command line option which is a replacement for the `R` option when using boards from some other vendors. If you do not have an `INMOS` board, contact your supplier to see whether it is applicable.

### 2.2.11 Changes to the host file server `iserver`

#### Enhancements

- Two files have been merged; `ims_b008cmd.h` and `ims_bcmd.h` have become `ims_bcmd.h`.
- In the two files mentioned above, a number of data structures were declared, namely `B014_IO` and `B008_IO`. These structures have been replaced with one called `IMS_IO`. The old names will continue to work until the next major version of `iserver` is released (1.5).
- Two new `linkio` modules have been provided, `b016link.c` and `acsilink.c`.

#### Bugs Fixed

- Occasionally, when `iserver` received an `SpExit` request, the wrong exit code was being returned from the `iserver`. This has been fixed.

- Certain `linkio` modules did not timeout correctly. These modules have been fixed.

### 2.2.12 Changes to other tools

Cosmetic and trivial changes have been made to `ilibr`, `icvlink`, `ilist`, `icvemit`, `iemit`, and `idump`. Their operation is unaffected.

## 2.3 NEC PC specific changes

Due to compatibility problems with the NEC DOS `key` utility, special support for the toolset debugger `idebug` and simulator `isim` on NEC systems has changed.

The following are a list of changes which apply to the NEC PC :

- **New NEC ITERM file** (Ref. Delivery Manual section 2.2.8)

This has changed from `necpc.itm` to `necansi.itm`.

- **NEC specific batch files for `nidebug` and `nisim`** (Ref. Delivery Manual section 2.5)

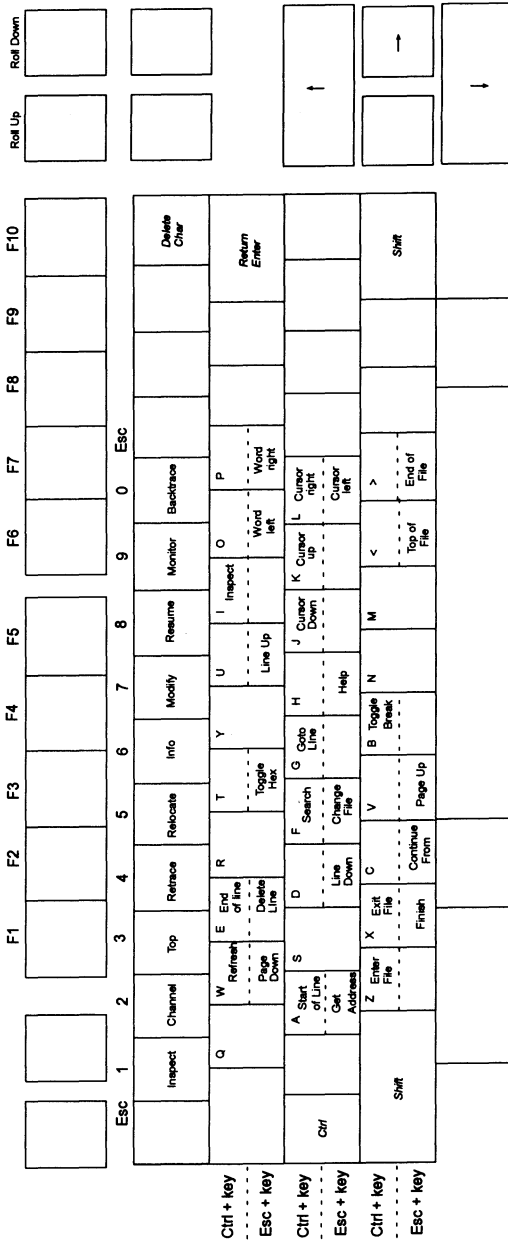
`nidebug` and `nisim` batch files are no longer present; run `idebug` and `isim` as for any other PC machine.

### 2.3.1 New keyboard layout

The keyboard layout corresponding to the replacement ITERM file `necansi.itm` is given on the following page.



# NEC keyboard layout



(RH keypad not used)



# 3 Installation

## 3.1 Introduction

### 3.1.1 If you are replacing an existing ANSI C toolset

This toolset completely replaces the previous ANSI C toolsets D4214A, IMS D5214A, and D7214B. If you already have one of these installed, then the installation will overlay the same directories with new files. However, since the content of the directories is not identical, it is recommended that you delete the original installation directory and all subdirectories. Any changes you have made to these should be recorded and repeated after this installation. The changes to the contents are listed in an appendix to these notes.

### 3.1.2 If you are using C and Occam toolsets together

The ANSI C toolset and the OCCAM toolset have some tools and libraries in common, such as the linker, librarian and debugger, 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 later version*. Tools are designed to be *backwards-compatible* so that they will work with previous releases of the TCOFF 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 date on 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.

If you are mixing OCCAM and C, information is provided in both the ANSI C toolset manual and the OCCAM toolset manual. The ANSI C toolset manual describes how to call OCCAM from a C program or from the C configuration language. The OCCAM toolset manual describes how to call C from an OCCAM program.

### 3.1.3 Installing the iserver

Note that the server installation procedure described in section 2.1 of the Delivery Manual does not apply to this toolset. The installation will not ask questions regarding which server to install. Instead you must explicitly install the server you require. Installation of the iserver is described in section 3.2.

### 3.1.4 Transputer hosted tools

There are three tools (`idebug`, `idump`, and `iskip`) that require transputer bootable files to execute on all host versions. (The PC version of the toolset requires

transputer hosted versions of some other tools as well.) These are provided in the `itools` subdirectory. If you wish to use these, then you should add the `itools` to your path after the `tools` directory or copy them into the `tools` directory as before. This is different from the previous versions, where they were copied into the `tools` directory automatically. The change has been made to make the toolset compatible with our other toolset products and to provide the user with greater flexibility in how to set up access to the tools.

### 3.1.5 New and changed directories

An appendix to these release notes lists the differences between the directory contents as provided and the lists in the appendix of the Delivery Manual.

## 3.2 Server installation

When the installation is complete, you must set up access to an `iserver`. The installation does not attempt to set one up automatically. Note that if you are replacing a previous toolset, then you should upgrade the server to one of those contained in this release. It should be release 1.42(i) or later.

One of the servers supplied should be renamed or copied onto the path.

### 3.2.1 PC server

For the PC version (D7214C) of this release, the servers are found in `\ICTOOLS\ISERVER` and are:

`isvrB04.exe` Can be used with the B004 and B008 boards with no further change.

`isvrB08.exe` Supports the B008 board, but only in conjunction with the device driver from the S708B product (qv).

`isvrnec.exe` Supports the B010 and B015 boards in NEC PCs.

If you have a non-INMOS board, then consult your supplier. It is likely, but not guaranteed, that the B004 version may work, depending on the board.

### 3.2.2 Sun 3/Sun 4 servers

For the Sun3 (D5214B) and Sun4 (D4214B) versions of the toolset, three servers are supplied. These support the IMS B011, B014, and B016 products and have the names `B011_iserver`, `B014_iserver` and `B016_iserver` respectively. You should choose the appropriate server for your hardware.

# 4 Errata

This release has the following known bugs and problems. INMOS reference numbers are given where appropriate.

## 4.1 General

- (Ref: TS/0239) The iserver used with this toolset can handle a maximum packet length of 1024 bytes. However, the C Library has a maximum packet length of 512 bytes. This may lead to errors if the iserver should send a packet longer than 512 bytes to either an application or one of the tools in this toolset. This would only occur in extreme situations, such as using the `getenv` function to obtain an environment variable which is longer than 509 characters. This bug will be fixed in future releases of the iserver.
- (Ref: TS/0539) Due to a minor problem in this release with the usage of TCOFF representation of processor capabilities between tools (e.g. hardware breakpoint support, FPU etc.), a problem arises when collecting some single processor programs with `icollect` when using the 'T' option. This problem does NOT affect the bootable code produced by the toolset. The problem manifests itself by making transputer types appear to migrate to another processor type between different tools:

Original type	Migrated type
T222	T212
T400	T425
T801	T805

This is best illustrated by an example:

```
icc foo.c /g /t400
ilink foo.tco /f startup.lnk /t400
icollect foo.lku /t
```

`ilist` will show the processor type as T425 in both `foo.tco` and `foo.lku`. More importantly, `idebug` will generate a serious error message when breakpoint debugging because it has been told to expect a T425 (and it finds a T400). When post-mortem debugging, `idebug` will generate a warning.

### Workaround

These problems primarily affect debugging where it is crucial that the transputer type used is the same as that specified in the configuration. One solution is to configure for a single processor with `icconf` (i.e. don't use

the `icollect` single processor 'T' option). If you must have single processor capability from `icollect`, the following table illustrates individual workarounds for the different processor types:

Processor Type	Workaround (for <code>icollect T</code> option)
T222	none - <code>idebug</code> will happily think the processor is a T212.
T400	debug in TB (or TA) mode and commit to T400 mode when debugged.
T801	debug in TA mode and commit to T801 mode when debugged.

- (Ref: TS/0555) The driver programs for the transputer hosted tools (except `idebug` and `idump`) monitor 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, then the driver programs may be fooled into thinking the tool has set the error flag if the error flag on one of the extra processors is already set when the tool starts.

In order to overcome the problem, you should run `ispy`, a network check program (or similar), or boot a dummy program that uses all the transputer processors in the network. Note that, once cleared, the error flag on a transputer will only become set again if you execute an erroneous program on the transputer, or if you power off the transputers. The state of the error flag is undefined (hence may be on) when the power is turned on.

- (Ref: TS/0680) The following error message can be set by any of the PC hosted tools: `Runtime error R6000 - stack overflow`. This is a general error message that can occur using ANY of the tools. The message is generated at runtime by the Microsoft C runtime system and cannot be trapped by the tool.

#### Workaround

Work around by running the tool on a transputer.

- (Ref: TS/1180) In rare circumstances, the following error message (or variations of it with the tool name `<inmos library>`) can be produced by some tools: `Fatal-<inmos library>-low level write failed 2`. Such an error can occur for instance, if the disk fills up during the writing of a file (although the tool itself will normally notice this and report the error in more detail).

## 4.2 Tools and libraries

### 4.2.1 `icc`

- (Ref: TS/0593) In the following fragment of C:

```
extern void p(void);
#pragma IMS_nolink (p);
extern void q(void)
{
    extern void p(void);
    p();
}
```

the second declaration of a function (here `p`), after a `nolink` pragma, loses the function's `nolink` status.

### Workaround

Add a second `#pragma IMS_nolink (p)` after the second declaration.

- (Ref: TS/0543) The compiler does not detect as an error the use of enumeration values that are too large for an `int` type for the 16-bit processors. For example, the following code should be erroneous:

```
enum colour { red = 100000};
```

- (Ref: TS/0517) The compiler produces a spurious warning (incorrect number of parameters) when a conversion specification in `scanf` or `printf` contains both an asterisk to denote a width of the field and an `h` to indicate a short value is to be used.
- (Ref: TS/0579) If the compiler runs out of disk space while compiling it just stops (hangs) on a PC. Escaping with CTRL-C leaves a file `T0.TMP` on the disk. No diagnostic is produced. Running with the 'I' option reports an error and terminates cleanly.
- (Ref: TS/0604) Some loops can be written which have no instructions that can allow a deschedule of the running process. These loops would have no body and a simple boolean test. They can occur either with a `while` loop or a `for` loop with no step clause and no body. For example:

```
while (value < 1)
;
```

### Workaround

Add a redundant statement to act as a body. For example:

```
while (value < 1)
    a = a;
```

- (Ref: TS/0641) If the compiler cannot write (part of) a file, then the user may not be informed of the fact. For example, if the "directory full" condition occurs when creating the `.tco` file, then no message is given.
- (Ref: TS/0669) If the source code contains a `typedef` for a function prototype, then the information generated for the debugger is incorrect. The effect is that the use of this type cannot be relied on in debugger expressions. Note that there is no problem with `typedefs` for function pointers.

#### 4.2.2 C libraries

- (Ref: TS/0606) The channel communication fault tolerant functions `ChanInChanFail`, `ChanInTimeFail`, `ChanOutChanFail`, `ChanOutTimeFail`, and `ChanReset` may not be used with virtual links when breakpoint debugging. The debugger converts all external links described to the configurer into virtual links when breakpoint debugging in order to share links with the user application program. Using these functions with virtual links will lead to undefined behaviour.

##### Workaround

In order to use these functions when breakpoint debugging you must provide real physical link addresses. This is best performed by declaring channels within the source code (i.e. not passing them in from the configurer) and initialising them with the appropriate link value defined in `channel.h`. Such placements must not conflict with any link placements described to the configurer otherwise undefined behaviour will occur.

- **Zero length messages on channels.**

If zero length messages are passed using channel library functions which take a length parameter, the results are undefined. This applies to data passed using the functions `ChanIn`, `ChanOut`, `ChanInChanFail`, `ChanInTimeFail`, `ChanOutChanFail`, and `ChanOutTimeFail`. The behaviour is a direct result of the design used to implement message passing in the transputer.

#### 4.2.3 `icconf`

- (Ref: TS/0587) The definition of the syntax of the configuration language allows the number of parameters in the interface to be zero. This is incorrect. An interface attribute must have at least one parameter. To call a process with no parameters, the interface attribute should be omitted.

#### 4.2.4 `icollect`

- (Ref: TS/0540) When using the 'T' option, the collector informs `idebug` that the type of a T801 or T805 transputer is a T800 because of an inconsistency in the internal definition. This will cause `idebug` to generate a serious error when breakpoint debugging, or a warning when post-mortem de-



bugging. Note that this is a different problem to that mentioned in section 4.1, bullet 2.

#### Workaround

Either use `icconf` to configure the program or debug in TA mode.

#### 4.2.5 `imakef`

- (Ref: TS/0529) `imakef` does not specify the name for a `.cfb` file generated by `icollect` when generating a `.bxx` file using the 'T' option. This is only a problem if you create multiple `.bxx` bootables for different transporter types from the same source in the same directory.

#### Workaround

Rename the `.cfb` file after creating each `.bxx`.

#### 4.2.6 `isim`

- (Ref: TS/0608) **D5214B/D4214B Sun 3/Sun 4 hosted versions only.** `isim` incorrectly sets the terminal mode when in the `iserver` so that the RETURN key is mapped to NEWLINE. Some routines in the `hostio` `occam` library expect a RETURN to terminate their input. These will not terminate as expected.

#### Workaround

When the program is expecting input, a RETURN can be explicitly supplied on Sun systems by the key sequence CTRL-V CTRL-M.

#### 4.2.7 `iemit`

- (Ref: TS/0626) `iemit` also uses the `ITERM` files to access the terminal. If one is not present, then it defaults to a teletype device.

#### 4.2.8 `iserver`

- (Ref: TS/0535) The help page of the `iserver` supplied with the toolset incorrectly states 'occam 2 toolset host file server', it should state 'INMOS toolset host file server'.

### 4.3 Documentation errors

#### 4.3.1 Reference Manual

- Descriptions of the multibyte functions `mblen`, `mbtowc`, `mbstowc`, `wctomb`, and `wcstombs`, are missing from Chapter 2. They are provided in Appendix C of this document.

- (Ref: TS/0632) Reference Manual, p.12. The note at the top of section 1.3.7 on the Mathematics library should read:

“Note: The following is true for all functions declared in `math.h`:

On domain errors: `errno` is set to `EDOM`;  
0.0 is returned.

On range errors: `errno` is set to `ERANGE`;  
`HUGE_VAL` is returned for overflow errors;  
`-HUGE_VAL` is returned for underflow errors. ”

- Reference Manual, p.15, section 1.3.11 ‘Standard definitions’. In the example a comment says that the value of `offsetof` is 2 for 16-bit machines. This is incorrect; it is 4 for both 16-bit and 32-bit machines.
- Reference Manual, p.18. Under ‘Characteristics of file handling’ the library filename in the third bullet item should be `stdio.h`, not `stdoio.h`.
- Reference Manual, p. 46, `assert` example. The `assert` line in the example should read:

```
assert( b != 0.0f );
```

(i.e. replace `==` by `!=`, and use floating point suffix to avoid cast).

- Reference Manual, p. 86, `ctime` function description. As with `asctime`, care must be taken when using `ctime` in a concurrent environment. Calls to `ctime` by independent unsynchronised processes may corrupt the return value.
- Reference Manual, p.129, `frexpf` function description. Contrary to that stated in the ‘Errors’ section, `frexpf` may generate domain errors. For both `frexpf` and `frexp`, domain errors are caused by NaN or Infinity.
- Reference Manual, p.199, `ProcAlloc` function description. The second paragraph should read:

“The function to be set up as a process by `ProcAlloc` should have as its first parameter a pointer to a process structure. This pointer must not be included in the calculation of `nparam`.”

- Reference Manual, p. 221, `ProcStop` function description. The description omits to state that `ProcStop` can be used in conjunction with any parallel process. This includes configurer processes as well as those started by `ProcPar` etc.
- Reference Manual, p. 250, `server_transaction` function. Under ‘Errors’ the description of error code 1 should read “Length is less than the minimum server message size of 6 bytes”.

- (Ref: TS/0631) **Reference Manual**, p. 89. The second sentence in the first paragraph of the description of the function `debug_stop` ('If the program is in HALT mode...') should be deleted. The processor does not stop when interactively debugging, only the process that executes the `debug_stop` function call.
- (Ref: TS/0667) **Reference Manual**, p.129. Under **Errors** it should say that a domain error can occur.
- (Ref: TS/1245) **Reference Manual**, p. 198. The description of the parallel function `ProcAlloc` is insufficiently clear about the size of the stack space that must be reserved for the process. The following text should be added after the description of `sp`:

“

It is important that enough space is allocate for the stack for the process. If insufficient is provided, the results are undefined. In particular, in the worst case, the runtime library needs 150 words (600 bytes for 32-bit, 300 bytes for 16-bit machines). This must be allowed for as well as the stack requirement of the user functions (cf. `max_stack_usage`).

”

This also affects the breakpoint debugging of some configuration examples, as described below.

When using the breakpoint debugger with the configurer examples that use the linked units derived from the `hostpass.c` and `multpass.c` source files then these source files should be modified to ensure they execute correctly.

The lines that have to be changed are the calls to `ProcAlloc` that allocate a structure that can be used with `ProcRun` later on. In the calls to `ProcAlloc` a stack size of 128 bytes is given which is sufficient when the code is run outside the debugger, but insufficient when the same code is run from the breakpoint debugger, because the virtual io library is used for channel communication. When running through the breakpoint debugger the value 1024 bytes should be sufficient. (Replace the value 128 in the `ProcAlloc` parameter list by the value 1024).

This stack size will also be sufficient if stack checking is enabled when compiling these source modules.

The following configuration source files import `hostpass.c` and `multpass.c`:

```

pipe.cfs          ring.cfs          tree.cfs
square.cfs       square1.cfs       zigzag.cfs

```

Programs generated from these configurations should reference the modified versions of `multpass.c` and `hostpass.c` if they are breakpoint debugged.

- **Reference Manual**, p. 359. The declaration of `main` in section B.2 should be preceded by inclusion of the channel library header file:

```
#include <channel.h>
```

#### 4.3.2 User Manual

- **User Manual**, Contents overview, p. iii. The order of Appendices D and E in the listing should be reversed. Appendix D describes bootstrap loaders and Appendix E the ISERVER protocol.
- (Ref: TS/0703) **User Manual**, p.23. In section 2.13.4 'Environment variables' the description of `IBOARDSIZE` should say "Used when loading non-configured programs" instead of "Used by `iserver`".
- (Ref: TS/0635) **User Manual**, p.77. Comment characters in the example should be `/*`, not `\*`.
- (Ref: TS/1076) **User Manual**, p.78. In section 6.5.5 'Segment ordering' the first paragraph states that there are four code segments. There are in fact five, and these are correctly listed in the succeeding paragraph.
- **User Manual**, p.79. "process" in the first example should be preceded by "define".
- **User Manual**, p.119. In section 8.12 'Debugging with `isim`' the words "2 Mbyte" should be deleted.
- **User Manual**, p.122. The wrapping round of some lines in the comment section of the listing at the top of the page is a document processing error.
- **User Manual**, p.138. In the first paragraph the name of the linker startup file for reduced library programs should be `startrd.lnk`, not `started.lnk`.
- **User Manual**, p.147. The example on page 147 is for a 16-bit transputer not a 32-bit transputer.
- **User Manual**, p.150. In the diagram the file extension for the configuration input file should be ".cfs" not ".cfg".
- **User Manual**, p.167. In the table of `IMS_on/IMS_off` parameters, the abbreviation for `scanf_checking` should be "sf", not "sc".
- **User Manual**, p. 208. The `icollect` command line in 'Examples of use' should NOT include the 't' option.
- **User Manual**, p. 236. Under section 13.2.1 'Examples of use' a note should be added to say that single transputer programs linked with the reduced library cannot be bootstrapped using the `icollect T` option and must be configured instead.

- (Ref: TS/0759) **User Manual**, p. 261. In the final sentence of the paragraph describing the `n` option (penultimate paragraph on page), 't' should read 'n'.
- **User Manual**, p. 263. Typographical error in the third paragraph; the characters "`(exit_terminate`" appear twice.
- **User Manual**, p. 299. Document processing error on first line; the characters "tt" should be deleted.
- **User Manual**, p. 316. In section 16.2.1 'Example of use' the assumed `IBOARDSIZE` should be 100000 not 100K.
- (Ref: TS/1173) **User Manual** p. 347. The documentation for `ieprom` does not state clearly enough the processor type to choose for all root transputer variants. The following list states the `ieprom` transputer type to specify for IMS transputer types:

transputer type	ieprom transputer type
M212	T2
T212	T2
T222	T2
T225	T2
T400	T4
T414	T4
T425	T4
T800	T8
T801	T8
T805	T8

- **User Manual**, p.425, p.427. The `isim N` command creates a core dump file, not a network core dump file.
- (Ref: TS/0703) **User Manual**, p.443. In section A.4.4 'Miscellaneous files' the second sentence of the `.dmp` description should read:
 

"

Created by `idump` for debugging code on the root transputer (memory dump) or by `idebug` for off-line analysis of a program on a network (network dump).

"
- (Ref: TS/0587) **User Manual**, p. 465. In the `interface` definition the list of formal attributes should be represented as a list of *one or more* items (`{1 , formal-attr }`), NOT zero or more (`{0 , formal-attr }`).
- **User Manual**, p. 502. The declaration of `main` in section F4 should be preceded by inclusion of the channel library header file:

```
#include <channel.h>
```



# Appendices





# A Distribution kit

## A.1 Differences in directory contents lists

### A.1.1 Differences for D4214B and D5214B from D4214A and D5214A

This section describes the differences in directory contents for the Sun 3 and Sun 4 versions of this maintenance release compared to the previous release.

#### Directories removed

The following files and directories which were present in the first releases of the Sun 3 (IMS D5214A) and Sun 4 (IMS D4214A) ANSI C toolsets are not present in this maintenance release:

```
install
install/install
iserver/iserver
libs/centry.lbb
libs/libc.lbb
libs/libcred.lbb
setup
source/imakef/imakef.lnk
source/iserver/b008cmd.h
tools/idebug
tools/idebug.btl
tools/idump
tools/idump.btl
tools/iskip
tools/iskip.btl
```

#### Directories added

The following files and directories have been added:

```
examples/linkboot
examples/linkboot/loader.occ
examples/linkboot/makefile.tp
examples/linkboot/userboot.asm
iserver/b016_iserver
iterms/ansi.itm
source/iserver/acsilink.c
source/iserver/b016link.c
source/iserver/iserver.lnk
source/iserver/makefile.b16
source/iserver/tosbuild.bat
source/iserver/traps.h
```

## A.1.2 Differences for D7214C from D7214B

This section describes the differences in directory contents for the PC version of this maintenance release compared to the previous release.

### Directory structure

The directory structure for this maintenance release differs slightly from the directory structure used in the previous (D7214B) release of the PC toolset. The following structure replaces that described in section 2.1 of the Delivery Manual:

D7214C directory structure:

<code>\ictools\itools</code>	The transputer bootable tools.
<code>\ictools\tools</code>	The PC hosted tools.
<code>\ictools\libs</code>	The toolset libraries and include files.
<code>\ictools\examples</code>	Examples directory.
<code>\ictools\examples\simple</code>	Simple example sources.
<code>\ictools\examples\debugger</code>	Debugger example sources.
<code>\ictools\examples\imakef</code>	<code>imakef</code> example sources.
<code>\ictools\examples\config</code>	Configurer example sources
<code>\ictools\examples\config\b008</code>	Configurer example B008 configuration files.
<code>\ictools\examples\linkboot</code>	Single processor user defined bootstrap example.
<code>\ictools\iserver</code>	The iserver executables.
<code>\ictools\source</code>	Source code.
<code>\ictools\source\iserver</code>	Server sources.
<code>\ictools\source\imakef</code>	<code>imakef</code> sources.
<code>\ictools\source\driver</code>	Bootable file driver program sources.
<code>\ictools\iterms</code>	Example item files and driver program.

### NEC directory removed

Note that the NEC directory `\ictools\nec` (Ref. Delivery Manual section A.13) has been removed in this release of the toolset and all references to it should be ignored.

### Distribution kit

- Directory `\ictools\itools`. (Ref. Delivery Manual section A.1).  
`nidebug.bat` and `nisim.bat` do not exist in this release of the toolset.

- Directory `\ictools\iterms`. (Ref. Delivery Manual section A.4).

`necansi.itm` replaces `necpc.itm`, and the file `ansi.itm` has been added. The new listing of the directory is as follows:

<code>bansi.sys</code>	Screen device driver
<code>pcansi.itm</code>	Item file for use with <code>ANSI.SYS</code>
<code>pcbansi.itm</code>	Item file for use with <code>BANSI.SYS</code>
<code>necansi.itm</code>	Item file for use with an NEC PC
<code>ansi.itm</code>	Item file for use with an ANSI terminal
<code>readme.txt</code>	Information file

- Directory `\ictools\iserver`. (Ref. Delivery Manual section A.10).

`iserver.exe` will only appear if you have copied one of the three installed `iservers` to `iserver.exe`.

- Directory `\ictools\nec`. (Ref. Delivery Manual section A.13).

This directory does not exist in this release of the toolset.

- New directories:

The directory `\ictools\examples\linkboot` has been added. The directory listing is as follows:

```
loader.occ
userboot.asm
makefile.tp
```

The directory `\ictools\source\driver` has been added. This contains the source of the driver program which is used to boot the transputer bootable versions of the tools.



# B ispy program

## B.1 Network worm – ispy

In the ANSI C toolset User Manual, p 262 and p 305, there are references to the program **ispy** which can be used to clear the error flags in networks of transputers. This program is available as part of the board support software product IMS S508 for use with the toolsets on Sun 3s and Sun 4s. However, there is no equivalent source of **ispy** for PC boards. A copy of **ispy** is included in the D7214C PC release, on a separate disk.

There are 4 copies of **ispy** as follows:

<b>ispy4s.exe</b>	small model for B008 or B004
<b>ispy4.exe</b>	large model for B008 or B004
<b>ispy8.exe</b>	large model for B008 with S708 device driver
<b>ispy nec.exe</b>	large model for B015 for the NEC PC.

A help page is available by specifying the **'/?'** command line option.

Small model versions are faster but have only 64K for code + data. Large models have more space for big networks. The **ispy8** version requires the B008 device driver (supplied with the IMS S708 software support package) to be installed with a line in **config.sys**, similar to the following:

```
DEVICE=C:\s708a\s708driv.sys /A 200 /N LINK1 /D N
```

The option **/N LINK1** is the link name and **ispy8** is invoked as follows:

```
ispy8 /L LINK1
```

Using these tools as follows:

```
ispy4s  
ispy4
```

generates output in the following format:

```
Using #200 ispy 2.33.1  
# Part rate Mb Bt [ Link0 Link1 Link2 Link3 ]  
0 T800c-20 0.05 0 [ HOST 1:1 3:1 ... ]  
1 T2 -17 0.88 1 [ ... 0:1 ... C004 ]  
3 T800c-17 0.88 1 [ ... 0:2 4:1 ... ]  
4 T414b-15 0.53 1 [ ... 3:2 5:1 ... ]  
5 T414b-15 0.48 1 [ ... 4:2 6:1 ... ]  
6 T414b-15 0.49 1 [ ... 5:2 ... ... ]
```

### B.1.1 Explanation of output

The following diagram explains the meaning of each line of the `ispdy` display. The first line of the output shown in the previous section is used as an example.

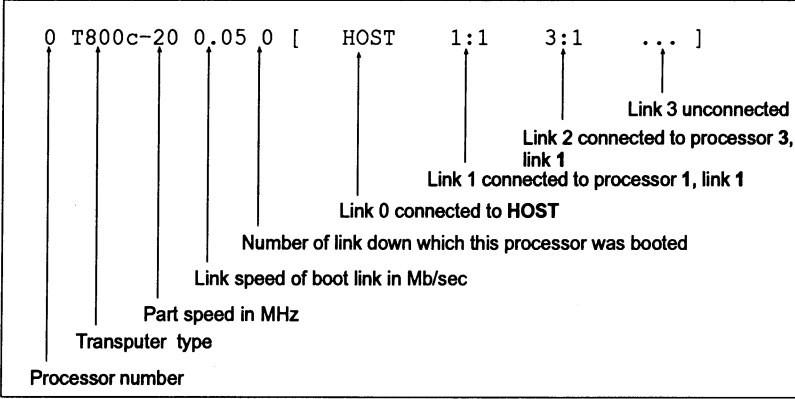


Figure F.1 Meaning of `ispdy` display

# C Multibyte functions

## C.1 Introduction

This appendix contains detailed descriptions of the ANSI multibyte functions `mblen`, `mbtowc`, `mbstowcs`, `wctomb`, and `wcstombs`, which were always present on the installation directories but were omitted from the documentation.

## C.2 Implementation

The functions provide a minimal implementation of the ANSI standard. This is considered sufficient because the current toolset supports only the standard C locale, and therefore any implementation is of limited practical value.

The functions support an implementation of wide characters in which:

```
wchar_t == int
MB_MAX_LEN == 1.
```

### C.3 List of functions

**mblen** Determines number of bytes in a multibyte character.

**Synopsis:**

```
#include <stdlib.h>
int mblen(const char *s, size_t n);
```

**Arguments:**

**const char \*s** Pointer to the multibyte character.  
**size\_t n** The maximum number of bytes to be read.

**Results:**

If **s** is not a null pointer **mblen** returns the number of bytes that are contained in the multibyte character pointed to by **s**, as long as the next **n** or fewer bytes form a valid multibyte character. In the current implementation the maximum length of a character is 1 byte.

If **s** is a null pointer **mblen** returns zero.

**Errors:**

If the specified sequence does not correspond to a valid multibyte character **mblen** returns **-1**.

**Description:**

**mblen** evaluates the number of bytes in a multibyte character. The number of bytes read is limited by **n**.



---

**mbstowcs** Converts multibyte sequence to `wchar_t` sequence.

**Synopsis:**

```
#include <stdlib.h>
size_t mbstowcs(wchar_t *pwcs, const char *s, size_t n);
```

**Arguments:**

<code>wchar_t *pwcs</code>	Pointer to the start of the array that receives the converted codes.
<code>const char *s</code>	Pointer to start of the array of multibyte characters to be converted.
<code>size_t n</code>	The maximum number of bytes to be read.

**Results:**

`mbstowcs` returns the number of array elements modified, not including any terminating zero codes.

**Errors:**

If an invalid multibyte character is encountered `mbstowcs` returns `(size_t)-1`.

**Description:**

`mbstowcs` converts a sequence of multibyte characters into a sequence of codes. It acts like the `mbtowc` function but takes as input an array of characters and returns an array of codes.

Not more than `n` codes are written into `pwcs`. If the initial and receiving objects overlap, the behaviour is undefined.

No multibyte characters that follow a NULL character are examined or converted.

**mbtowc**Converts multibyte character to type `wchar_t`.**Synopsis:**

```
#include <stdlib.h>
int mbtowc(wchar_t *pwc, const char *s, size_t n);
```

**Arguments:**

<code>wchar_t *pwc</code>	Pointer to the storage location for the converted character.
<code>const char *s</code>	Pointer to the multibyte character to be converted.
<code>size_t n</code>	The maximum number of bytes to be read.

**Results:**

If `s` is not a null pointer, `mbtowc` either returns zero (if `s` points to NULL) or returns the number of bytes that are contained in the converted multibyte character, as long as the next `n` or fewer bytes form a valid multibyte character. In the current implementation the maximum length of a character is 1 byte.

If `s` is a null pointer, `mbtowc` returns zero.

The value returned cannot be greater than `n` or the value of `MB_CUR_MAX`.

**Errors:**

`mbtowc` returns `-1` if the next `n` or fewer bytes do not form a valid multibyte character.

**Description:**

`mbtowc` converts a multibyte character to a wide character code and stores the result in the object pointed to by `pwc`.

---

**wcstombs** Converts `wchar_t` sequence to multibyte sequence.

**Synopsis:**

```
#include <stdlib.h>
size_t wcstombs(char *s, const wchar_t *pwcs, size_t n);
```

**Arguments:**

<code>char *s</code>	Pointer to the start of the array where the results will be stored.
<code>const wchar_t *pwcs</code>	Pointer to the start of the wide character sequence to be converted.
<code>size_t n</code>	The maximum number of bytes to be stored.

**Results:**

`wcstombs` returns the number of bytes modified, not including any terminating zero codes.

**Errors:**

If an invalid code is encountered `wcstombs` returns `(size_t)-1`.

**Description:**

`wcstombs` converts a sequence of wide-character codes into a sequence of multi-byte characters. It acts like the `wctomb` function but takes as input an array of codes and returns an array of characters.

Not more than `n` bytes are written into `s`. If the initial and receiving objects overlap, the behaviour is undefined.

Storage of a null character terminates the function.

**wctomb**Converts type `wchar_t` to multibyte character.**Synopsis:**

```
#include <stdlib.h>
int wctomb(char *s, wchar_t wchar);
```

**Arguments:**

<code>char *s</code>	Pointer to the array object that will receive the multibyte character.
<code>wchar_t wchar</code>	Code of wide character to be converted.

**Results:**

If `s` is not a null pointer, `wctomb` returns the number of bytes in the multibyte character corresponding to `wchar`.

If `s` is a null pointer, `wctomb` returns zero.

The value returned cannot be greater than `n` or the value of `MB_CUR_MAX`.

**Errors:**

If `wchar` does not correspond to a valid multibyte character `wctomb` returns `-1`.

**Description:**

`wctomb` converts a wide-character code to a multibyte character to and stores the result in the array pointed to by `s`. At most `MB_CUR_MAX` characters are stored.

# Addresses

## Worldwide Headquarters

INMOS Limited  
 1000 Aztec West  
 Almondsbury  
 Bristol BS12 4SQ  
 UNITED KINGDOM  
 Telephone (0454) 616616  
 Fax (0454) 617910

## Worldwide Business Centres

### USA

INMOS Business Cent  
 Headquarters (USA)  
 SGS-THOMSON Microelectronics Inc.  
 2225 Executive Circle  
 PO Box 16000  
 Colorado Springs  
 Colorado 80935-6000  
 Telephone (719) 630 4000  
 Fax (719) 630 4325

SGS-THOMSON Microelectronics Inc.  
 Sales and Marketing Headquarters (USA)  
 1000 East Bell Road  
 Phoenix  
 Arizona 85022  
 Telephone (602) 867 6100  
 Fax (602) 867 6102

INMOS Business Centre  
 SGS-THOMSON Microelectronics Inc.  
 Lincoln North  
 55 Old Bedford Road  
 Lincoln  
 Massachusetts 01773  
 Telephone (617) 259 0300  
 Fax (617) 259 4420

INMOS Business Centre  
 SGS-THOMSON Microelectronics Inc.  
 1310 Electronics Drive  
 Carrollton  
 Texas 75006  
 Telephone (214) 466 7402  
 Fax (214) 466 7352

INMOS Business Centre  
 SGS-THOMSON Microelectronics Inc.  
 9861 Broken Land Parkway  
 Suite 320  
 Columbia  
 Maryland 21045  
 Telephone (301) 995 6952  
 Fax (301) 290 7047

INMOS Business Centre  
 SGS-THOMSON Microelectronics Inc.  
 200 East Sandpointe  
 Suite 650  
 Santa Ana  
 California 92707  
 Telephone (714) 957 6018  
 Fax (714) 957 3281

INMOS Business Centre  
 SGS-THOMSON Microelectronics Inc.  
 2055 Gateway Place  
 Suite 300  
 San Jose  
 California 95110  
 Telephone (408) 452 9122  
 Fax (408) 452 0218

**EUROPE****United Kingdom**

INMOS Business Centre  
SGS-THOMSON Microelectronics Ltd.  
Planar House  
Parkway Globe Park  
Marlow  
Bucks SL7 1YL  
Telephone (0628) 890 800  
Fax (0628) 890 391

**France**

INMOS Business Centre  
SGS-THOMSON Microelectronics SA  
7 Avenue Gallieni  
BP 93  
94253 Gentilly Cedex  
Telephone (1) 47 40 75 75  
FAX (1) 47 40 79 27

**West Germany**

INMOS Business Centre  
SGS-THOMSON Microelectronics GmbH  
Bretonischer Ring 4  
8011 Grasbrunn  
Telephone (089) 46 00 60  
Fax (089) 46 00 61 40

**Italy**

INMOS Business Centre  
SGS-THOMSON Microelectronics SpA  
V.le Milanofiori  
Strada 4  
Palazzo A/4/A  
20090 Assago (MI)  
Telephone (2) 89213 1  
Fax (2) 8250449

**ASIA PACIFIC****Japan**

INMOS Business Centre  
SGS-THOMSON Microelectronics K.K.  
Nisseki Takanawa Building, 4th Floor  
18-10 Takanawa 2-chome  
Minato-ku  
Tokyo 108  
Telephone (03) 3280 4125  
Fax (03) 3280 4131

**Singapore**

INMOS Business Centre  
SGS-THOMSON Microelectronics Pte Ltd.  
28 Ang Mo Kio Industrial Park 2  
Singapore 2056  
Telephone (65) 482 14 11  
Fax (65) 482 02 40