inmos

D705B IBM/NEC PC occam 2 toolset delivery manual

INMOS Limited

72 TDS 187 00

1 Introduction

This manual provides installation instructions for the D705B occam 2 toolset for the IBM and NEC PC (and compatibles), including descriptions of parts of the toolset specific for DOS-based systems. Instructions for rebuilding the parts of the release supplied in source form are also given and changes from the previous release of the toolset are listed.

1.1 Layout of this manual

This introduction lists the prerequisites for running the DOS-based version of the toolset and summarises the contents of the release.

The rest of this manual is divided into the following chapters:

2 Installing the release: gives instructions for installing the software and setting up the system, and lists the software components.

3 DOS dependencies: provides information about DOS-specific features and requirements, including DOS specific libraries.

4 Sources supplied with the toolset: provides information about the sources supplied with the toolset, including how to rebuild the tools after modification.

5 Changes from the D705A occam toolset: outlines the differences between this and the D705A occam toolset, and lists the new tools.

6 AFSERVER libraries: describes the AFSERVER libraries which enable the D705A toolset to be used with the new host file server.

7 Known problems: lists known bugs in the software and documentation.

8 Debugger and simulator keyboard layout: shows keyboard layouts for the debugger and simulator functions on the IBM and NEC PC.

1.2 Prerequisites for running the toolset

In order to use this version of the toolset you will require:

- An IBM PC, PC/XT or PC/AT (or compatible), or an NEC PC.
- DOS version 2.0 or later (the toolset was made under DOS 3.3), with the ability to increase environment space above 160 bytes.

- About 5.6 Mbytes of free disk space (although you need not install the entire release - see section 2.1).
- An IMS B004, B008 (or similar) transputer evaluation board with an IMS T800 or T414 (Rev B or later) transputer.
- An editor and, optionally, a UNIX compatible MAKE program. Microsoft MAKE is not compatible.

1.3 Contents of this release

This release of the D705B occam 2 toolset consists of:

- A set of twelve 360 Kbyte floppy disks.
- The '*D705B occam 2 toolset delivery manual*' (this document).
- The '*occam 2 toolset user manual*' – the user and reference manual.
- The '*occam 2 toolset handbook*' – a quick reference guide.
- The '*occam 2 toolset TDS support guide*' – instructions for using the TDS – Toolset conversion tools.

2 Installing the release

The D705B occam 2 toolset comes on twelve 360 Kbyte floppy disks. The toolset software resides on disks 1 to 11, and the twelfth contains the TDS support tools. All disks are written in normal DOS low density (48 tpi) format.

You will require about 5.6 Mbytes of free space to install the entire release. Depending on the use you make of the toolset it may not be necessary to install the entire release. For example, you do not need to install libraries that you do not intend to use, and you need only install the sources if you intend to make modifications to the code.

2.1 Installation

If you have a previous version of the toolset it will be best to remove it before installing the new version.

During the installation process new directories are created. If a directory with the same name already exists (say, from a previous version) then an error will be reported. The installation will not be affected and the new toolset will simply use the old directory.

If during the installation an existing file would be overwritten (say, one from a previous version of the toolset) then you will be asked whether to proceed. To overwrite a file, type 'Y' followed by RETURN.

2.1.1 Installation procedure

To install the release first insert Disk 1 in your floppy disk drive. Then run the batch file `install.bat` on Disk 1, giving as parameters the drive letter of the floppy disk drive and the drive letter of the drive on which the toolset will be installed.

For example, if your floppy disk drive is A, and the drive on which you want the toolset installed is C, type:

```
a:install a c
```

You will then be given information on how to proceed with the installation. You will be asked a number of questions to which you can answer 'yes' by typing a 'Y' (or a 'y'), or answer 'no' by typing a 'N' (or a 'n'). A RETURN is not required. To install every file answer 'Y' to all questions.

When all the relevant disks have been read, which takes about ten minutes, and provided that none of the files already exist on the system, the installation can be allowed to proceed unattended. A full installation takes about 50 minutes on an IBM PC/AT.

The installation procedure creates a directory called `\itools`. All the programs necessary to install the toolset are copied to this directory. All the components of the toolset itself are copied into sub-directories of `\itools`. If a part of the toolset is not installed then the directory for that part will not be created.

Subdirectories of `itools` are listed in section 2.6.

2.1.2 Selective installation

The installation procedure enables you to install selected parts of the toolset. If disk space is a problem, or you do not need all the components, you can omit the parts you do not require.

When first installing the toolset it is best to install every file. If disk space is a problem, then you will be in a better position at a later date to decide which parts to delete.

The following table gives the size of each component that you can selectively install, and an indication of whether it is required for normal operation of the toolset or occam programming.

Component	**Size**	**Necessary**
Tools	1.4Mb	*yes*
Transputer based tools	0.3Mb	*no*
Compiler libraries	0.7Mb	*yes*
I/O libraries	0.4Mb	*yes*
Miscellaneous libraries	0.3Mb	*yes*
Maths libraries	0.4Mb	*for occam maths functions*
AF protocol conversion libraries	23Kb	*for AFSERVER compatibility only*
Examples	21Kb	*for first time users*
Interface code	0.4Mb	*for C, FORTRAN, and Pascal*
Sources	0.9Mb	*no*
TDS – Toolset conversion tools	0.2Mb	*if a user of TDS*

2.2 Setting system variables

This section explains how to set up the system variables necessary for normal operation of the toolset. Any DOS commands can be added to your `autoexec.bat` file so that they will be set up whenever the system is started.

An example of how to set up your environment for the toolset is given in the file `\itools\D705b.bat`.

2.2.1 Set DOS path

To be able to use the tools under DOS, the directory `\itools\tools` must be added to your DOS path. To do this use the DOS `PATH` command.

For example,in order to set the path to your system commands and then that to the toolset (on drive C), type:

```
path c:\system;c:\itools\tools
```

Note that in this example the directory `c:\system` is already part of the existing path.

2.2.2 Setting up the correct host file server

Two versions of the host file server `iserver` are supplied with this release in the `\itools\tools` directory. The two files are listed below.

File name	Server type
`iserver.exe`	Standard server for IBM PC
`neciserv.exe`	NEC PC version

To use the NEC server remove or rename the file `iserver.exe` and rename the file `neciserv.exe` to `iserver.exe`.

2.2.3 Set IBOARDSIZE

Before you can use any tool which runs on a transputer board you must set up an environment variable `IBOARDSIZE` which specifies the size of the board's memory in bytes. Tools running on transputer boards will report an error if `IBOARDSIZE` is not set up.

To set up the variable use the DOS `SET` command. For example, to set a board size of 2 Mbytes using the hexadecimal notation type:

```
set IBOARDSIZE=#200000
```

The board size can be specified in either decimal or hexadecimal. Hexadecimal numbers are preceded by the '#' character.

`IBOARDSIZE` is used by the bootstrap code to calculate the size of the free memory array passed into the program for its own use.

2.2.4 Set ITERM

The debugger and simulator tools require an environment variable `ITERM` that specifies a terminal description file. If no ITERM file is found the tools are not able to run and an error is reported. The ITERM file needs `ansi.sys` in order to function correctly.

To use the standard ITERM file supplied with the toolset (assuming drive C) type:

```
set ITERM=c:\itools\iterms\ibmpc.itm
```

A different ITERM file is supplied for the NEC PC. To set the `ITERM` variable for this system type:

```
set ITERM=c:\itools\iterms\necpc.itm
```

The simulator and debugger will report an error if they cannot find a valid ITERM file.

2.2.5 Set ISEARCH

To enable the tools to find libraries and source files on the toolset path you must set up an environment variable called `ISEARCH` to specify a hierarchy of paths to be searched. This variable should specify the standard toolset library directory (`\itools\libs\`) and any user directories that are required.

Unlike the DOS `PATH` command you must add the closing backslash '\' to the directory name. Directory names in the list may be separated by a space or a semi-colon.

For example to set up `ISEARCH` to reference the standard toolset libraries and to a user directory called `\mydir` type:

```
set ISEARCH=c:\itools\libs\ \mydir\
```

If you are using C, FORTRAN, or Pascal you should also include the directory `\itools\interf\` in order to reference the occam interface code. For example:

```
set ISEARCH=c:\itools\libs\ c:\itools\interf\ \mydir\
```

2.2.6 Board address

The default board address is 150 (hexadecimal). If your board is mapped to a different address you should set up an environment variable **TRANSPUTER** which gives the address of the board.

You can override the value specified by **TRANSPUTER** when loading programs onto a board by using the **iserver** '**SL**' option.

For example if your board is at address 200 (hexadecimal) type:

```
    set TRANSPUTER=#200
```

2.3 Transputer based tools

In the DOS based toolset some tools run on the PC, the rest on the transputer board. The following tools run on the PC:

ilibr	The librarian
ilist	The binary lister
iboot	The bootstrap tool
imakef	The Makefile generator

All other tools (except the host file server) run on the transputer board.

Transputer-based versions of the above PC hosted tools are available on the directory `\itools\transp`. To use the transputer-based versions of the tools specify the directory `\itools\transp` on the DOS path before the normal tools directory.

2.4 Environment space

The PC running DOS 2.0 and later has available 160 bytes of environment space by default. This may need to be increased in order to run the toolset.

All versions of DOS allow the environment space to be increased to a maximum of 32 Kbytes, with varying degrees of difficulty. For the commands or procedures to use on your system consult the user documentation for the specific version of DOS you are using.

For DOS versions 3.2 and later the **SHELL** command in the **config.sys** file can be used to set up an environment size when the PC is booted. For example:

```
SHELL=command.com /e:1024 /p
```

This example gives the name of the DOS command processor, sets the environment space to 1024 bytes and makes this version of the command processor permanently resident.

In DOS version 3.3 and later the command called **COMMAND** can be used to increase the environment space. For example:

```
COMMAND /e:1024 /p
```

This has a similar effect to the **SHELL** command example but is invoked from DOS.

Earlier versions of DOS require the command processor (**command.com**) to be patched. Microsoft provide a utility **SETENV** that will do this automatically.

2.5 Special considerations for the NEC PC

The NEC PC version of the toolset differs from the IBM PC version in three minor ways:

- Special files are required to set up the keyboard for the ITERM file
- A different ITERM file is used
- A different **iserver** is used
- The debugger and simulator tools are invoked using similar but distinct commands.

Differences in the use of ITERM files and **iserver** have been described previously in sections 2.2.2 and 2.2.4. The special commands that invoke the debugger and simulator are described below.

2.5.1 Invoking the debugger and simulator on the NEC PC

On the NEC PC the commands that invoke the debugger and the simulator must be prefixed with the letter 'n'. For example:

```
nisim

nidebug
```

These forms of commands *must* be used on NEC based systems so that the keyboard is set up correctly for the NEC PC ITERM file.

2.6 Software components

All tools and supporting software are installed on the directory `itools` and its subdirectories. The main subdirectories are listed in the following table.

Directory	Contents
`\itools\tools`	The executable tools
`\itools\iterms`	The ITERM files
`\itools\transp`	Transputer versions of host tools
`\itools\libs`	The toolset occam libraries
`\itools\interf`	occam interface code for non-occam programs
`\itools\examples`	occam example sources (see user manual)
`\itools\source\iserver`	Server sources
`\itools\source\imakef`	Makefile generator sources
`\itools\source\driver`	Driver program sources
`\itools\source\hostio`	Hostio library sources
`\itools\source\streamio`	Streamio library sources
`\itools\source\msdos`	DOS specific i/o library sources
`\itools\source\maths`	Maths library sources
`\itools\source\convert`	Type conversion library sources
`\itools\source\string`	String library sources
`\itools\source\process`	Process library sources
`\itools\tdstools`	Toolset – TDS conversion tools

2.6.1 Tools

The `tools` and `transp` subdirectories contain executable and bootable files for the individual tools.

2.6.2 Iterms

The `\itools\iterms` subdirectory contains example ITERM files for the IBM and NEC PCs, and keyboard setup files for the NEC PC. The files are listed below.

File	Contents
`ibmpc.itm`	IBM PC ITERM file
`necpc.itm`	NEC PC ITERM file
`doskeys.ld`	NEC PC keyboard setup file
`tds2keys.ld`	NEC PC keyboard setup file

2.6.3 Libraries

The `\itools\libs` subdirectory contains occam libraries and files of constants. The files are listed below and overleaf.

File	Contents
`afhostio.inc`	AFSERVER library constants
`afhostio.lib`	AFSERVER library
`afproc.lib`	AFSERVER process library
`convert.lib`	Type conversion library
`crc.lib`	Block CRC library
`dblmath.lib`	Double length general maths library
`hostio.inc`	Hostio library constants
`hostio.lib`	Hostio library
`hostio.liu`	Hostio library usage file
`linkaddr.inc`	Link address constants
`mathvals.inc`	Constants for use with maths libraries
`msdos.inc`	MSDOS library constants
`msdos.lib`	MSDOS library

File	Contents
`occam2h.lib`	T212 HALT compiler library
`occam2s.lib`	T212 STOP compiler library
`occam2u.lib`	T212 UNDEFINED compiler library
`occambh.lib`	T414/T425 HALT compiler library
`occambs.lib`	T414/T425 STOP compiler library
`occambu.lib`	T414/T425 UNDEFINED compiler library
`occam8h.lib`	T800 HALT compiler library
`occam8s.lib`	T800 STOP compiler library
`occam8u.lib`	T800 UNDEFINED compiler library
`process.lib`	Process library
`snglmath.lib`	Single length general maths library
`streamio.inc`	Streamio library constants
`streamio.lib`	Streamio library
`streamio.liu`	Streamio library usage file
`string.lib`	String library
`string.lib`	String library usage file
`tbmaths.lib`	T425/T414 maths library
`xlink.lib`	Extraordinary link handling library

2.6.4 Examples

The `\itools\examples` subdirectory contains the sources of example programs. The files are listed below.

File	Contents
`ctype2a.occ`	Type 2 occam interface example
`ctype2b.occ`	Type 2 occam interface example
`ctype3.occ`	Type 3 occam interface example
`simple.occ`	Simple example program
`debugex.occ`	Debugging example
`element.occ`	Sorter element
`inout.occ`	I/O process for sorter example
`sorter.occ`	Main body of single processor sorter example
`sorter.pgm`	Configuration description of multiprocessor sorter example
`sorthdr.inc`	Constants for sorter examples
`tsort.occ`	Sub-pipe for multiprocessor sorter example
`relocld.occ`	Secondary bootstrap example
`seconld.occ`	Secondary bootstrap example

2.6.5 occam interface code

The occam interface code for C, FORTRAN and Pascal programs is located in the directory `\itools\interf`. There are 106 files in all for various combinations of transputer type and error mode. A generic list is given below.

Files	Contents
`afstub.*`	Used by interface code and included for use by debugger
`mainent.*`	Type 1 interface code
`procent.*`	Type 2 interface code
`procentc.*`	Type 3 C interface code
`procentf.*`	Type 3 FORTRAN interface code
`procentp.*`	Type 3 Pascal interface code
`procents.*`	Type 3 C, FORTRAN and Pascal interface code

2.7 Sources

The sources supplied with the toolset are contained in subdirectories of `\itools\source`. Source is supplied for two of the tools, namely, `imakef` and `iserver`. The source of the driver programs for these tools is also supplied.

2.7.1 Library sources

The sources of the following libraries are provided. Directories on which they can be found are given in parentheses.

- The maths libraries (`\itools\source\maths`)
- The hostio library (`\itools\source\hostio`)
- The streamio library (`\itools\source\streamio`)
- The DOS specific i/o library (`\itools\source\msdos`)
- The process library (`\itools\source\process`)
- The string library (`\itools\source\string`)
- The type conversion library (`\itools\source\convert`)

2.7.2 Tool sources

The sources of the following tools are provided. Directories on which they can be found are given in parentheses.

- The **`iserver`** (**`\itools\source\iserver`**)
- The **`imakef`** tool (**`\itools\source\imakef`**)
- The tool driver programs (**`\itools\source\driver`**)

Instructions for building tools and libraries from source are given in sections 4.1.1 and 4.2.1.

2.8 TDS support tools

Tools for porting programs between the TDS and the toolset format are provided on the subdirectory `\itools\tdstools`. The files are listed below.

File	**Description**
`readme`	Information about the tools
`import.cut`	TDS **#SC** import/export utility
`import.top`	Readme file for **`import.cut`**
`iflat.exe`	occam file flattener tool
`idirect.b4h`	Compiler directive convertor
`idirect.exe`	Driver program for **`idirect.b4h`**
`list.exe`	Single file flattener tool
`c.set`	Set file for **`list.exe`**
`occam.set`	Set file for **`list.exe`**
`null.set`	Set file for **`list.exe`**
`lister.set`	Set file for **`list.exe`**
`streamco.lib`	Stream i/o conversion library
`streamco.liu`	Library usage file for **`streamco.lib`**

3 DOS dependencies

3.1 Driver programs

Programs which run on transputer boards are controlled by 'driver' programs which are written in C and run on the PC. These programs call the server with the correct options to boot the programs onto the transputer board.

If the tool cannot be found the driver program reports the error in the following format:

> **`Error-driver-`***error message*

The source of the driver programs can be found in the subdirectory **`\itools\source\driver`**. Details of how to rebuild the programs from source can be found in chapter 4.

3.2 Server interrupts

It is possible to interrupt the server, go to DOS to issue DOS commands, and subsequently return to the server. This has the effect of temporarily halting a program running on a transputer board. The program continues to run until access to the server is required.

To interrupt the server, use the following procedure. Remember to set the BREAK key first.

Use CTRL-BREAK (in preference to CTRL-C) in order to interrupt the program. Type 'S' at the prompt, which invokes DOS. DOS commands can now be executed as necessary.

In order to return to the server type 'exit'. This quits DOS and restarts the tool you were running previously.

When in DOS do not invoke any tool or program that runs on the transputer board, or the program running in the background will be corrupted.

The ability to interrupt the server relies on the existence of either a DOS environment variable `COMSPEC` or a DOS command file `COMMAND.COM` in order to recall DOS.

3.3 DOS specific hostio library

Library: `msdos.lib`

The MSDOS host file server library allows programs to use some facilities specific to the IBM PC. The library is supplied for the T212 and for the transputer class TA, both in UNIVERSAL error mode. A set of constants for the library are provided in the include file `msdos.inc`.

Caution: Programs that use the DOS specific hostio library will not be portable to other versions of the toolset.

Procedure	Parameter Specifiers
`dos.receive.block`	`CHAN OF SP fs, ts,` `VAL INT32 location,` `INT bytes.read, []BYTE block,` `BYTE result`
`dos.send.block`	`CHAN OF SP fs, ts,` `VAL INT32 location,` `VAL []BYTE block,` `INT len, BYTE result`
`dos.call.interrupt`	`CHAN OF SP fs, ts,` `VAL INT16 interrupt,` `VAL [dos.interrupt.regs.size]` `BYTE register.block.in,` `BYTE carry.flag,` `[dos.interrupt.regs.size] BYTE` `register.block.out,` `BYTE result`
`dos.read.regs`	`CHAN OF SP fs, ts,` `[dos.read.regs.size] BYTE` `registers,` `BYTE result`
`dos.port.read`	`CHAN OF SP fs, ts,` `VAL INT16 port.location,` `BYTE value, result`
`dos.port.write`	`CHAN OF SP fs, ts,` `VAL INT16 port.location,` `VAL BYTE value, BYTE result`

3.3.1 Procedure definitions

In the following descriptions, `fs` is the channel *from* the host file server, and `ts` is the channel *to* the host file server. The `SP` protocol used by host file server channels and the constants used by these routines are defined in the file `hostio.inc`.

`dos.receive.block`

```
PROC dos.receive.block (CHAN OF SP fs, ts,
                        VAL INT32 location,
                        INT bytes.read,
                        []BYTE block,
                        BYTE result)
```

Reads a block of length `len`, starting at `location`, from host memory.

`dos.send.block`

```
PROC dos.send.block (CHAN OF SP fs, ts,
                     VAL INT32 location,
                     VAL []BYTE block,
                     INT len, BYTE result)
```

Writes a block of data contained in `record` to host memory, starting at `location`.

`dos.call.interrupt`

```
PROC dos.call.interrupt
               (CHAN OF SP fs, ts,
                VAL INT16 interrupt,
                VAL [dos.interrupt.regs.size]
                BYTE register.block.in,
                BYTE carry.flag,
                [dos.interrupt.regs.size]
                BYTE register.block.out,
                BYTE result)
```

Invokes an interrupt call on the PC, with the processor's registers intialised to the correct values. On return from the interrupt the values stored in the processor's registers are returned, along with the value of the carry flag on the PC, stored in `flag`.

The interrupt number is specified by `interrupt`. The registers are represented by a block of bytes called the `register.block1` This block stores the values to be written to the registers, whilst `register.block2` contains the returned values. Each register value occupies 4 bytes of a block. On the IBM PC the 2 most significant bytes are ignored as this machine has only 2 byte registers (16 bit registers). The layout of registers in the block is as follows:

REGISTER	Start position in block **l.s. byte**	End position in block **m.s. byte**
`ax`	0	3
`bx`	4	7
`cx`	8	11
`dx`	12	15
`di`	16	19
`si`	20	23
`cs`	24	27
`ds`	28	31
`es`	32	35
`ss`	36	39

Note that the segment registers (`cs, ds, es and ss`) cannot be permanently changed using this routine, and are restored to their original values after the interrupt. The `cs` and `ss` registers cannot be changed.

You should intialise all the registers, (using `dos.read.regs` if necessary), or the registers will take whatever random data that already exists in the respective positions in the block.

dos.read.regs

```
PROC dos.read.regs (CHAN OF SP fs, ts,
                    [dos.read.regs.size]
                    BYTE registers,
                    BYTE result)
```

Reads the current values of the PC's registers. The values of the registers are returned as a block of bytes, each register occupying 4 bytes of the block. On the IBM PC the 2 most significant bytes are ignored as this machine has only 2 byte registers (16 bit registers). The order in which the registers appear in the block (at 4 byte intervals) is the same as for the `dos.call.interrupt`.

`dos.port.read`

```
PROC dos.port.read (CHAN OF SP fs, ts,
                    VAL INT16 port.location,
                    BYTE value, result)
```

Reads the value at the port, specified by the port address, `location`. The value returned is that of the port at the moment the port is read by the host file server. No check is made to ensure that the value received from the port is valid. On the IBM PC the 2 most significant bytes of the port address are ignored as the port address can only be represented as a 2 byte number (i.e. 16 bits).

`dos.port.write`

```
PROC dos.port.write (CHAN OF SP fs, ts,
                     VAL INT16 port.location,
                     VAL BYTE value, BYTE result)
```

Writes the given `value` at the port specified by the port address, `location`. No check is made to ensure that the value written to the port has been correctly read by the device connected to the port (if any). On the IBM PC the 2 most significant bytes of the port address are ignored as the port address can only be represented as a 2 byte number (i.e. 16 bits).

4 Sources supplied with the toolset

The following sources are supplied with the toolset:

Tools	–	The sources of the **iserver** and **imakef** tools.
Libraries	–	The sources of most of the occam libraries.
Drivers	–	The sources of the tool driver programs.
Examples	–	Example programs used in the *'occam 2 toolset user manual'*.

4.1 Tools

The sources of **imakef** and **iserver** are supplied with the toolset. You are free to modify these tools as you wish.

Both tools were compiled for this release using Microsoft C version 5.10.

4.1.1 Building the tools

To rebuild **imakef** or **iserver** from the source first move to the appropriate subdirectory under **\itools\source** or copy all the files in the subdirectory to a working directory.

The file **makefile** in the subdirectory contains build instructions for each tool. Edit the file to comment out the PC lines and all other host specific build information. Some lines must be uncommented. This applies specifically to the NEC PC, whilst certain lines must be uncommented for all other PCs. Comments inside the file explain what to do. Then run MAKE on this file.

4.2 Libraries

All library sources are written in occam. They are compiled using the toolset compiler occam and are formed into libraries using the toolset librarian **ilibr**.

4.2.1 Building the libraries

To rebuild a library from the source first move to the appropriate subdirectory under **\itools\source** or copy all the files in the subdirectory to a working directory.

Then produce a Makefile for the library using **`imakef`**. For example:

```
    imakef msdos.lib
```

This will produce the Makefile **`msdos`**. To build the library run your MAKE program on this file.

A MAKE file for **`hostio.lib`** is supplied on the subdirectory **`\itools\libs\hostio`**.

4.3 Drivers

Sources of the tool driver programs are supplied on the subdirectory **`\itools\source\driver`**.

To build all the driver programs except the **`idebug`** driver, compile the program **`driver.c`** using Microsoft C. (Version 5.10 was used to compile the driver programs for this release). The **`driver.c`** program contains instructions describing which compiler options to use for the various tools. When the file is compiled, rename the resulting **`.exe`** file with the name of the tool you wish it to drive.

To build **`idebug`** use the file **`idebug.c`**.

4.4 Examples

Descriptions and listings of the example programs can be found in the '*occam 2 toolset user manual.* They are not described here.

5 Changes from the D705A occam toolset

This is the first release of the D705B occam 2 toolset. This chapter briefly describes the differences between the D705A occam toolset and this release.

Many improvements have been made to the toolset since the D705A series was first released and are described in full in the *'occam 2 toolset user manual'*. The main changes are highlighted below.

5.1 File names

The D705B toolset uses file extensions to distinguish between code compiled for different processors and in different modes, and to distinguish between the output of the various tools.

5.2 occam language implemented

The occam language implemented by the compiler is full occam 2, with the exception of in-line `VALOF`. The configuration language has been extended so that occam source can be used within configuration definitions, with the single restriction that the source must not explicitly or implicitly call any library routine.

5.3 Libraries

The set of libraries has been extended and improved. In particular the i/o libraries have been considerably extended to make use of the new host file server.

5.3.1 Obsolete libraries

A version of the D705A `flibs` library is supplied in `afhostio.lib`, with associated constants in `afhostio.inc`, and a version of the process library in `afproc.lib`.

Warning: These libraries are supplied only to maintain compatibility with the previous release of the software, and will be discontinued in future releases. To avoid compatibility problems in the future you are strongly recommended to convert to the new libraries.

The libraries are listed in section 6.1.

5.4 Extended functionality

The following is a list of improvements made to existing tools:

compiler The compiler no longer distinguishes between compiling a library, a separate compilation unit or a main body.

The compiler can compile for a wider range of transputer types, and in different error modes.

The way code is accessed by occam from other languages has been greatly simplified.

The way in which separately compiled units are used has been simplified.

linker The linker no longer produces executable code. Use the bootstrap tool `iboot` for single transputer programs and the configurer tool `iconf` for multitransputer programs.

Separately compiled occam code no longer needs to be linked in a fixed order.

configurer The configurer allows occam source to appear inside `PROCESSOR` statements.

server The server has been completely rewritten to make it easier and more efficient to use, modify and port to other systems. Programs which communicate with the D705A server will *NOT* work with the D705B server. You have a choice between:

- converting your source to use the new libraries which go with the `iserver` – you will find these libraries better and more comprehensive than the old libraries. This should be your long term aim.
- using your existing source with the protocol convertor supplied with the new hostio library. New versions of the old AFSERVER libraries are provided to help with this.

5.5 New tools

The following tools are new and did not exist in the D705A toolset:

`imakef` – **Makefile generator** This takes the place of `tomake`, but is more comprehensive, producing a Makefile to build any type of file that can be generated by the toolset.

`iboot` – **bootstrap tool** This adds bootstrap code for single processor programs, replacing part of the linker's functionality.

`ilist`– **binary lister** This displays information about object and executable code files such as libraries, in a readable form. For example the names and parameters of library procedures can be checked.

`icheck` – **syntax checker** This performs a syntax check of occam source and produces multiple error messages.

`ilibr` – **librarian** This concatenates code produced by the compiler or linker into libraries.

`idebug` – **debugger** This is a source level post mortem debugger, suitable for debugging networks of transputers.

`isim` – **simulator** This simulates a single T414 transputer, allowing source level break pointing and single stepping of occam programs.

5.6 Interface to non-occam languages

The interface to C, FORTRAN, and Pascal has been greatly simplified. For details see chapter 9 of the '*occam 2 toolset user manual*'.

6 AFSERVER libraries

This chapter describes the libraries provided with this release of the toolset for compatibility with the D705A toolset. These libraries will not be supported in future releases of the toolset.

6.1 The AFSERVER libraries

The AFSERVER libraries are intended to enable existing programs which use the AFSERVER to be used with the D705B host file server with minimal modification. By using these libraries in conjunction with the protocol convertor `af.to.sp` in the hostio library, programs written using the D705A toolset can be used with the new server.

This is recommended for short term development only and for all future programs you should use the new hostio library.

Warning: These libraries will be discontinued in future releases and you are strongly recommended to convert to the new libraries.

The AFSERVER libraries and associated constants are contained in the following files:

File	Contents
`afhostio.inc`	Constants for afhostio library
`afhostio.lib`	Equivalent to old flibs library
`afproc.lib`	Equivalent to old process library

6.1.1 The afhostio library

The afhostio library contains the same routines as the D705A flibs library. A list of the routines is given below. Consult your existing documentation for details of the routines and how to use them. The binary lister tool `ilist` can be used to examine the procedural interfaces.

Constants for the afhostio library can be found in the file `afhostio.inc`.

Procedure definitions

```
PROC af.read.integer (CHAN OF ANY si, INT n)
```

```
PROC af.write.integer (CHAN OF ANY so, VAL INT n)
PROC af.read.record (CHAN OF ANY si, INT len,
                     []BYTE r)

PROC af.write.record (CHAN OF ANY so, VAL[]BYTE r)

PROC read.key (CHAN OF ANY si, so, INT key, result)

PROC read.key.wait (CHAN OF ANY si, so,
                    INT key, result)

PROC open.file (CHAN OF ANY si, so,
                VAL []BYTE file.name,
                VAL INT access.method, open.mode,
                VAL INT exist.mode, record.length,
                INT stream.id, result)

PROC close.stream (CHAN OF ANY si, so,
                   VAL INT stream.id, close.mode,
                   INT result)

PROC close.save (CHAN OF ANY si, so,
                 VAL INT stream.id, INT result)

PROC close.delete (CHAN OF ANY si, so,
                   VAL INT stream.id,  INT result)

PROC read.block (CHAN OF ANY si, so,
                 VAL INT stream.id, len,
                 INT bytes.read,
                 []BYTE buffer, INT result)

PROC write.block (CHAN OF ANY si, so,
                  VAL INT stream.id,
                  VAL []BYTE record,
                  INT len, result)

PROC seek (CHAN OF ANY si, so,
           VAL INT stream.id, offset,
           INT result)

PROC open.input.stream (CHAN OF ANY si, so,
                        VAL INT stream.no,
                        INT stream.id, result)
```

```
PROC open.output.stream (CHAN OF ANY si, so,
                         VAL INT stream.no,
                         INT stream.id, result)

PROC open.screen (CHAN OF ANY si, so,
                  INT id, result)

PROC open.error (CHAN OF ANY si, so,
                 INT id, result)

PROC open.keyboard (CHAN OF ANY si, so,
                    INT id, result)

PROC open.params (CHAN OF ANY si, so,
                  INT id, result)

PROC open.temp.file (CHAN OF ANY si, so,
                     VAL INT access.method,
                     VAL INT record.length,
                     INT stream.id, result)

PROC terminate.filer (CHAN OF ANY si, so,
                      INT result)

PROC set.return.result (CHAN OF ANY si, so,
                        VAL INT value, INT result)

PROC rename.file (CHAN OF ANY si, so,
                  VAL []BYTE old.name, new.name,
                  INT result)

PROC stream.access (CHAN OF ANY si, so,
                    VAL INT stream.id,
                    INT access.method, result)

PROC stream.status (CHAN OF ANY si, so,
                    VAL INT stream.id, INT result)

PROC stream.file (CHAN OF ANY si, so,
                  VAL INT stream.id,
                  INT filename.length,
                  []BYTE filename, INT result)
```

```
PROC stream.length (CHAN OF ANY si, so,
                     VAL INT stream.id,
                     INT length, result)

PROC stream.connect (CHAN OF ANY si, so,
                      VAL INT stream.id,
                      INT connection, result)

PROC run.command (CHAN OF ANY si, so,
                   VAL []BYTE command.line,
                   INT result)

PROC read.time (CHAN OF ANY si, so,
                 INT time, result)

PROC runtime.data (CHAN OF ANY si, so,
                    VAL INT option,
                    INT option.value, result)

PROC read.environment (CHAN OF ANY si, so,
                        VAL []BYTE logical.name,
                        INT len, []BYTE real.name,
                        INT result)

PROC read.core.dump (CHAN OF ANY si, so,
                      VAL INT offset, length,
                      INT len, []BYTE core.dump,
                      INT result)

PROC server.version (CHAN OF ANY si, so,
                      INT version, date,
                      INT state, result)

PROC receive.block (CHAN OF ANY si, so,
                     VAL INT location, len,
                     INT bytes.read, []BYTE buffer,
                     INT result)

PROC send.block (CHAN OF ANY si, so,
                  VAL INT location,
                  VAL []BYTE record,
                  INT len, result)
```

```
PROC call.interrupt (CHAN OF ANY si, so,
                     VAL INT interrupt,
                     VAL []BYTE register.block1,
                     INT flag, len,
                     []BYTE register.block2,
                     INT result)

PROC read.regs (CHAN OF ANY si, so,
                INT len, []BYTE buffer,
                INT result)

PROC port.read (CHAN OF ANY si, so,
                VAL INT port.location,
                INT value, result)

PROC port.write (CHAN OF ANY si, so,
                 VAL INT port.location, value,
                 INT result)
```

6.1.2 The afproc library

The afproc library contains the same routines as the D705A process library. For details of the routines consult your existing documentation or use the binary lister tool to examine the procedural interfaces.

The afproc library contains the following routines:

```
PROC IoHandler (CHAN OF ANY FromFiler, ToFiler,
                CHAN OF ANY FromUser, ToUser,
                CHAN OF ANY Debug,
                VAL BOOL DisplayDebug)

PROC AfStubFiler (CHAN OF ANY FromProgram, ToProgram)

PROC AfHostConverter (CHAN OF ANY FromFiler, ToFiler,
                      FromProgram, Toprogram)
```

7 Known problems

This chapter details known problems in the D705B occam 2 toolset. The problems are divided into three parts, documentation, software and libraries.

7.1 Documentation

On page 120 of chapter 9 (*Mixed language programming*) in the '*occam 2 toolset user manual*' the following note should be added:

The FORTRAN type 3 interface can only be used with version 2.0 of the FORTRAN compiler and libraries. For version 1.1 of FORTRAN use the C type 3 interface.

(287)

The '*occam 2 toolset user manual*' does not document error messages that may be generated by the secondary bootstrap loader. These messages are generated when the program is loaded onto the transputer board.

Messages from the secondary bootstrap loader are displayed in the following format:

`Warning - bootstrap -` *message*

where *message* can be any of the following:

Unable to read, IBOARDSIZE

The environment variable `IBOARDSIZE` is not set up.

Bad format number, IBOARDSIZE

The environment variable `IBOARDSIZE` is not set up correctly.

Insufficient memory available, ***size***

There is insufficient memory, as defined by `IBOARDSIZE`, for the program to run.

(278)

The new configurer option '`B` *percent*' is not documented in the '*occam 2 toolset user manual*'.

The configurer uses two main buffers, one for the part of the tool that performs compilation, and one for the part that performs code extraction. These can be set using the '`B` *percent*' option, where *percent* refers to the proportion of memory allocated to the code extraction buffer.

By default the available memory is divided equally between the two buffers, equivalent to invoking the configurer with the option '`B 50`'. If the message '*program too complex*' is displayed then the compiler is short of memory and the percentage should be set to less than 50; if the message '*main code buffer overflow*' is displayed then the extractor is short of memory and the percentage should be set to more than 50.

(279)

File names and directory paths must not contain the standard host system option prefix. This applies even where the operating system permits it, as is the case with UNIX.

(290)

The new **`imakef`** option '`B`' is not documented in the '*occam 2 toolset user manual*'.

The '`B`' option causes **`imakef`** to generate a command file in place of a Makefile. The command file is an ordered list of system commands that can be used as a batch file or command script to unconditionally build the object file. The file may need to be edited to conform with the host batch file format.

(291)

7.2 Software

Software problems are listed under the name of the tool. Tools are listed alphabetically.

7.2.1 `icheck` – occam 2 checker

Retyping parts of an array using the `SIZE` operator can give rise to an assert failure. For example, the following piece of code will cause the problem:

```
PROC p ([]INT x)
  INT j, k:
  []BYTE x1 RETYPES [x FROM k FOR 2]:
  []BYTE x2 RETYPES [x FROM j FOR (SIZE x) - 2]:

  SKIP
:
```

The problem is associated with the `SIZE x` part of the above. **(269)**

7.2.2 `idebug` – debugger

The debugger incorrectly locates to the line preceding a procedure call when backtracing only if the procedure call corresponds to a single byte of code. **(286)**

7.2.3 `ilist` – binary lister

The binary lister produces a spurious Warning message for the compatibility string on id tag in C, FORTRAN and Pascal code, when invoked with the '`T`' (Tag data) option. **(285)**

7.2.4 `iserver` – host file server

The PC version of the `iserver` always assumes that a board address given by the environment variable `TRANSPUTER` or the `\SL` option is hexadecimal, even if it is not preceded by the # character. **(229)**

7.2.5 `isim` – T414 simulator

`ALT`ing on the `PLACED` channels connected to the server does not always work correctly. **(267)**

When single stepping around a `WHILE` loop or a replicated `SEQ` construct the simulator will step to the line before the loop construct after executing the last line of the construct. The behaviour of the program is correct, and single stepping will bring the cursor back into the loop. **(280)**

When setting a breakpoint on a line which is a procedure call the breakpoint is set after the return of the procedure. To set a breakpoint at the procedure use the 'Set breakpoint at procedure' option on the Monitor page. **(281)**

`REAL32`, `REAL64` and `INT64` numbers cannot be displayed. **(282)**

7.2.6 occam – occam 2 compiler

An occam type cannot be retyped as its own type. (**270**)

Arrays and segments of arrays cannot be used in multiple assignments. (**271**)

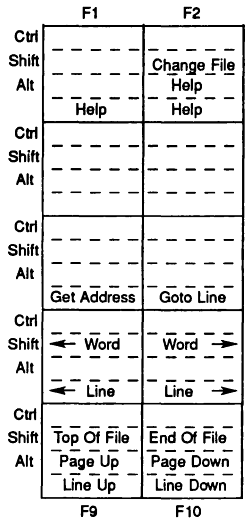
7.3 Libraries

None of the routines in `streamio.lib` recognise the `SS` protocol tags `st.initialise` and `st.help`. (**289**)

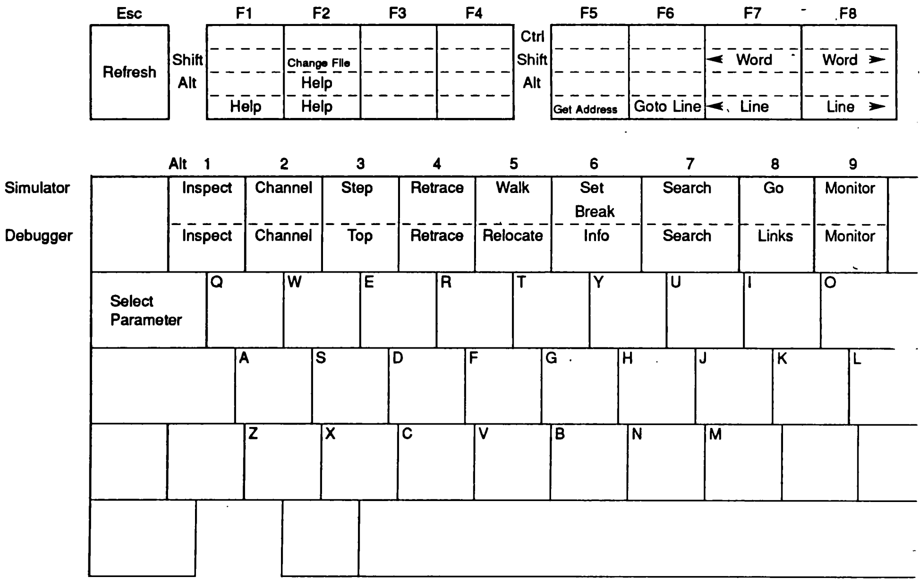
8 Debugger and simulator keyboard layout

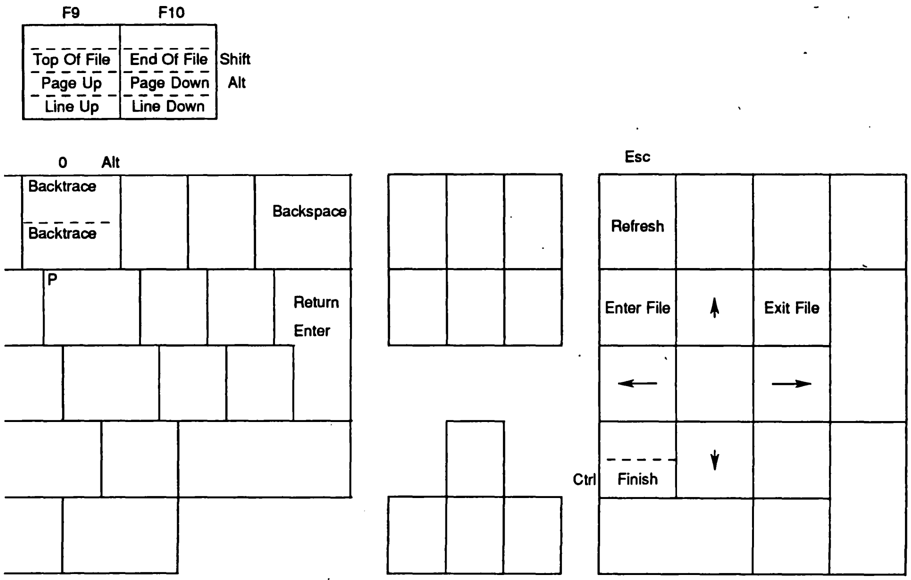
This chapter gives the keyboard layout for the debugger and simulator for both the IBM PC and compatibles, and the NEC PC.

8.1 IBM PC keyboard function keys

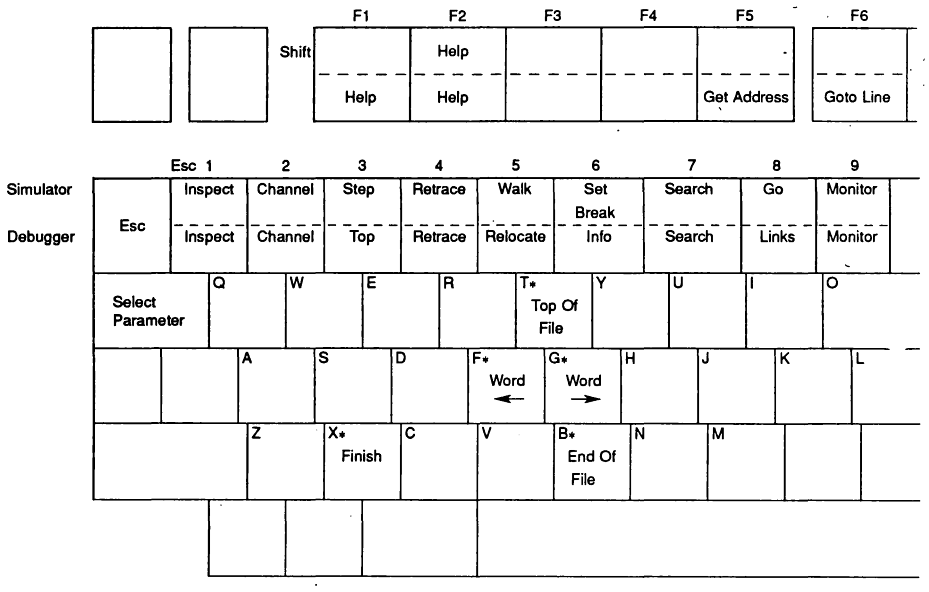


8.2 IBM PC keyboard layout



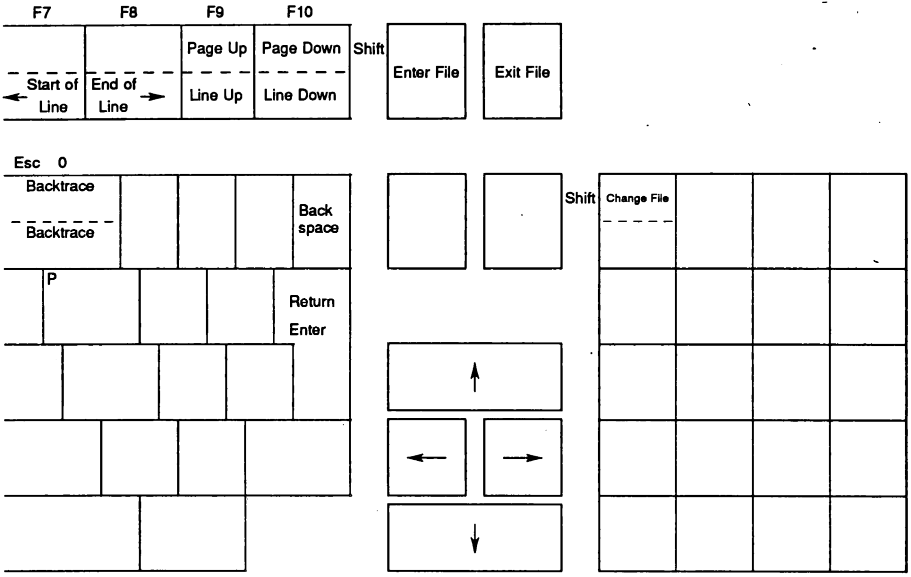

F9
F10
Top Of File
End Of File
Shift
Page Up
Page Down
Alt
Line Up
Line Down
0
Alt
Backtrace
Backtrace
Backspace
P
Return
Enter
Esc
Refresh
Enter File
Exit File
Ctrl
Finish

8.3 NEC PC keyboard layout



* Ctrl + key

Refresh = Esc Esc


F7
F8
F9
F10
Start of Line
End of Line
Page Up
Line Up
Page Down
Line Down
Shift
Enter File
Exit File
Esc 0
Backtrace
Backtrace
Back space
P
Return
Enter
Shift
Change File

INMOS Limited
1000 Aztec West
Almondsbury
Bristol BS12 4SQ
U.K.
Telephone (0454) 616616
TLX 444723

INMOS Corporation
P.O. Box 16000
Colorado Springs
Colorado 80935
U.S.A.
Telephone (719) 630 4000
TLX (Easy Link) 62944936

INMOS SARL
Immeuble Monaco
7 rue Le Corbusier
SILIC 219
94518 Rungis Cedex
France
Telephone (1) 46.87.22.01
TLX 201222

INMOS Japan K.K.
4th Floor No 1 Kowa Bldg
11-41 Akasaka 1-chome
Minato-ku
Tokyo 107
Japan
Telephone 03-505-2840
TLX J29507 TEI JPN

INMOS GmbH
Danziger Strasse 2
8057 Eching
West Germany
Telephone (089) 319 10 28
TLX 522645

72 TDS 187 00

May 1989