IMS D7205
IBM/NEC PC
occam 2 Toolset
delivery manual

INMOS Limited

72 TDS 245 01

March 1991

Copyright © INMOS Limited 1991

, inmos, IMS and occam are trademarks of INMOS Limited.

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

INMOS document number: 72 TDS 245 01

Contents

1 Introduction

This manual provides installation instructions for the IMS D7205 occam 2 toolset for the IBM PC (and compatibles) and the NEC PC. This delivery manual deals with PC-specific parts of the toolset.

1.1 Layout of this manual

Chapter 1 Introduction: (this chapter) summarises the contents of the release and describes its prerequisites.

Chapter 2 Installing the release: provides installation instructions for this release.

Chapter 3 Confidence testing: contains a simple procedure to check that the installation has been done correctly.

Chapter 4 Building from sources: contains details of how to use the sources supplied with the release to rebuild the libraries, server, and makefile generator.

Chapter 5 Changes from previous releases: contains details of the changes from previous versions of the occam 2 toolset such as the D705B.

Appendix A Distribution kit: contains a list of the components of the release.

Appendix B Debugger function keys: shows the positions of the debugger and simulator function keys on the IBM and NEC PC keyboard.

1.2 Prerequisites for running the toolset (IBM PC)

In order to use the occam 2 toolset you will require:

- An IBM PC, PC/XT or PC/AT (or compatible).
- DOS version 3.0 or later.
- About 10 Mbytes of free disk space.
- An IMS B004 or B008 (or similar) transputer board with an INMOS 32-bit transputer such as an IMS T800 or T414 (Rev B or later), with 2 Mbytes or more of memory.

Note: for interactive debugging a 32-bit transputer is used to run the debugger, so to use this feature of the toolset at least one additional transputer will be needed.

Although not absolutely necessary, it is also strongly recommended that you have a version of the **make** system building tool. Some versions of **make** compatible with the toolset are listed in the user manual, part 1 section 21.1.

1.3 Prerequisites for running the toolset (NEC PC)

In order to use the occam 2 toolset you will require:

- An NEC PC.
- DOS version 3.0 or later.
- About 10 Mbytes of free disk space.
- An IMS B010 or B015 (or similar) transputer board with an INMOS 32-bit transputer such as an IMS T800 or T414 (Rev B or later), with 2 Mbytes or more of memory.

Note: for interactive debugging a 32-bit transputer is used to run the debugger, so to use this feature of the toolset at least one additional transputer will be needed.

Although not absolutely necessary, it is also strongly recommended that you have a version of the **make** system building tool. Some versions of **make** compatible with the toolset are listed in the user manual, part 1 section 21.1.

1.4 Contents of this release

The major software and documentation components of the D7205 occam 2 toolset release are as follows:

- A set of 14 360 Kbyte 5.25 inch floppy disks.
- A set of 7 720 Kbyte 3.5 inch floppy disks.
- The *occam 2 toolset delivery manual* (this document).
- The *occam 2 toolset user manual – parts 1 and 2*.
- The *occam 2 toolset handbook*.

- The *TDS support guide* to aid in upgrading from the TDS.
- The *F editor* user manual.
- The *occam 2 reference manual*.
- The *Tutorial introduction to occam programming*.
- The *occam User Group Information*.
- The *occam User Group Enrolment Form*.
- The *occam User Groups*.
- Product registration form.

A customer checklist is provided listing all components of the release.

The toolset provides a collection of tools and libraries to support compilation, execution and debugging of occam 2 programs on transputer networks. The facilities provided by the toolset are described in the *occam 2 toolset user manual*. The *occam 2 toolset handbook* provides a handy reference guide to the tools and libraries.

A collection of tools and libraries is provided to aid users in upgrading from the INMOS Transputer Development System (TDS) products: IMS D700D and D700E. The facilities provided by these tools are described in the *TDS support guide*. If you are not currently a TDS user you can ignore these tools.

A preliminary version of a folding editor called 'F' is also provided with this toolset. This editor is intended primarily for previous users of the TDS who have grown used to this style of editing. However, it may be of interest to other toolset users who have not yet chosen an editor for use with the toolset. The usage of F is described in the *F editor* user manual.

Additional copies of the toolset user manual can be ordered; the product number is IMS D0205.

1.5 Compatibility with previous releases

This release is not object compatible with previous INMOS occam products such as the D705B; it uses a new object format known as TCOFF (Transputer Common Object File Format). Source files should be recompiled with the new tools. Chapter 5 of this delivery manual describes the main changes in the source languages accepted by this toolset, over the D705B.

For circumstances where it is not possible to recompile source, there is a conversion tool called **icvlink** which may be used to convert object files produced by the D705B (LFF format) to object files suitable for use with this toolset. **icvlink** is described in the user manual, part 1, chapter 13.

For users of the TDS, the TDS source file structures must be converted so that they are suitable for compilation with the toolset. See the *TDS support guide* for details.

This release is compatible with the D7214 ANSI C toolset supplied by INMOS, which also uses the TCOFF object format. Together these toolsets can be used for mixed language programming in C and occam.

1.6 Notes for users of the D7214 ANSI C toolset

The D7205 occam toolset and the D7214 ANSI C toolset have some tools in common, such as the linker, librarian and debugger. The installation procedures for the D7205 create a totally separate directory structure for the occam tools. However, for users of both toolsets, it is necessary to know which revision of the tool to choose.

When choosing which version of a tool to use *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; for example the debugger supplied with the ANSI C toolset will not necessarily support occam.

Consult the release notes to determine which revisions of the ANSI C toolset this release post-dates. If you are using both toolsets, put the *later* release on your path before the earlier one. If you decide to combine the tools into a common directory, then, for tools 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 user manual and the occam toolset user 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.

2 Installing the release

This release of the IMS D7205 occam 2 toolset comes on 14 360K 5.25 inch floppy disks and on 7 720K 3.5 inch floppy disks. You will require about 10 Mbytes of free space to install the entire release.

The whole release will be installed by the installation procedure, to make the process as easy as possible. You can delete some of the components after installation if you wish.

2.1 Installation

To install the release first insert Disk 1 in your floppy disk drive. Next run the batch file, **install.bat**, on Disk 1, giving as parameters the drive letter of the floppy disk drive and 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 instructions to put each disk in turn into the floppy disk drive. The contents of each floppy disk will be copied onto the destination drive. The installation procedure will then extract the files of the toolset from the archives and place them in appropriate directories.

If the installation has been successful the following messages will be displayed at the end of installation:

```
INSTALLATION COMPLETE
```

The installation procedure creates a directory called **\d7205**. 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 **\d7205**, as shown in the following table:

Directory	Contents
\d7205\itools	The transputer-hosted tools.
\d7205\tools	The PC-hosted tools.
\d7205\libs	The toolset libraries and include files.
\d7205\iterms	Example ITERM files.
\d7205\examples	Examples directory.
\d7205\examples\bootstrp	Bootstrap example sources.
\d7205\examples\idebug	Debugger example sources.
\d7205\examples\mixed	Mixed language example sources.
\d7205\examples\mixconf	Mixed language configuration example sources.
\d7205\examples\oc	occam compiler example sources.
\d7205\examples\occonf	Configurer example sources.
\d7205\examples\sorter	Sorter example sources.
\d7205\iserver	The **iserver** executables.
\d7205\source	Source code.
\d7205\source\iserver	Server sources.
\d7205\source\imakef	Makefile generator sources.
\d7205\source\driver	Driver sources.
\d7205\source\libs	Library sources(in sub directories).
\d7205\tdstools	The TDS to toolset conversion utilities.
\d7205\f	The F folding editor
\d7205\install	Installation scripts.

Appendix A describes these directories and their files in more detail.

The release installation procedure installs everything onto the hard disk. Certain parts of the toolset release may be removed from the hard disk if disk space is a problem. The following table indicates which parts of the release are essential for its correct operation.

Component	Necessary
itools	yes
tools	yes †
libs	yes
iterms	yes
iserver	yes
source	no
examples	no
tdstools	no
f	no

† Some of the tools are provided in both PC-hosted and transputer-hosted versions.

PC-hosted versions are ones which run on the PC. Transputer-hosted versions run on the transputer board. For each transputer-hosted tool there is a 'bootable file' with the extension `.btl` which contains the code to run on the transputer, and a driver program, with the extension `.exe` which runs on the PC. By typing the name of the tool, the driver program is invoked, and the transputer version of the tool is loaded and run on the transputer.

In general the PC-hosted ones will be faster to start up, as the transputer-hosted ones have first to be loaded onto the transputer board. However, the transputer-hosted ones can use the full memory available on the transputer board, and therefore can be used to build larger systems than the tools limited by the memory model on the PC. So in general you should use the PC-hosted tools in preference to start with, but move to the transputer-hosted versions if you encounter memory problems. If you are happy to use just the transputer-based tools then the `tools` directory may be deleted.

Do NOT delete the `itools` directory as it contains some tools for which hosted equivalents do not exist. Some tools, such as the debugger `idebug` and the dump tool `idump` were designed to work only on the transputer. Other tools, such as the occam compiler `oc` and the configurer `occonf` cannot run hosted on the PC because of the memory limitations of the PC.

If you are not upgrading from the TDS (Transputer Development System) then you will not need the `tdstools` directory, and this may be deleted.

If you do not require the F folding editor, then the `f` directory may be deleted.

The `source` and `examples` directories are not strictly needed for the operation

of the toolset, but it is recommended that you leave them installed, at least initially, as they provide many useful examples of occam programs, some of which are referred to in the manual.

Having installed the toolset you have to set up some environment variables before you can use any of the tools. The setup process is described in the next section.

2.2 Setting up the toolset for use

This section explains how to set up the environment necessary to use the toolset. It describes the basic changes to the system configuration file **config.sys** which you should make before you attempt to use the toolset and shows how to set up the necessary environment variables.

2.2.1 Choosing an iserver for use

All tools which run on the transputer are loaded by a program running on the PC known as the **iserver**. As well as loading the program it supplies access to the PC's terminal and file system. User programs are also loaded and served by the **iserver**.

Three compiled versions of the **iserver** are provided in the toolset release, in the directory **\d7205\iserver**. One of these must be chosen for use with your toolset. The three versions are as follows:

isvrb04.exe	For the IBM PC with an IMS B004 or B008 board or equivalent.
isvrb08.exe	For the IBM PC with an IMS B008 board and device driver.
isvrnec.exe	For the NEC PC with an IMS B010 or B015 board.

Note that the IMS B008 version of **iserver** should only be used if you have installed the IMS B008 device driver in the IMS S708 (which is the B008 software support product). Otherwise use the IMS B004 server **isvrb04.exe**.

To set up the appropriate **iserver** for use copy the selected server into the file **iserver.exe**; for example, to select the B004 server **isvrb04.exe**:

```
cd \d7205\iserver
copy isvrb04.exe iserver.exe
```

2.2.2 Setting the FILES variable

The **FILES** command in your system configuration file **config.sys** should be changed or added to specify at least 20 simultaneously open files. For example:

```
FILES=20
```

Note: Any other file handling software used on the system (such as PC-NFS) should also be reset to accept at least 20 simultaneously open files.

2.2.3 Setting the correct PATH

You should add the DOS commands given below to your **autoexec.bat** file so that they will be set up whenever you switch on your PC.

If you wish to use the transputer based tools then only add the directories **\d7205\iserver** and **\d7205\itools** to your DOS path. To do this use the DOS **path** command.

If hosted tools are required add **\d7205\tools** before **\d7205\itools**.

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

```
PATH=C:\DOS;C:\D7205\ISERVER;C:\D7205\ITOOLS
```

The above command sets up the path to find the transputer based tools only.

2.2.4 Setting an alternative iserver

The transputer-hosted tools are started by a driver program which runs the **iserver** to load the appropriate tool and run it on the transputer board. A copy of this driver exists for each of the transputer-hosted tools.

It is possible to arrange for the driver to pick up a different version of **iserver** from that on the path. An alternative **iserver** can be used by defining the file to be used in the **ISERVER** environment variable. If the **ISERVER** environment variable is defined on the system then the **iserver** is referenced by the environment variable rather than from the DOS path. If **ISERVER** is not defined on the system, then the DOS path, set up using the **path** command, is used to find the server.

For example:

To use a server called **myserver.exe** from your **\bin** directory on drive C, use the following definition:

```
set ISERVER=C:\BIN\MYSERVER.EXE
```

2.2.5 Using the TDS conversion tools and F editor

The TDS conversion tools and F editor are contained within their own directory structures for easy deletion by users who do not require them. Each of these directories contains a **tools** and **itools** subdirectory containing the PC-hosted and transputer-hosted versions of the tools respectively.

To make use of these tools you can do *one* of the following:

- Copy the tools into the **\d7205\tools** and **\d7205\itools** directories as appropriate; these will then be picked up from the DOS path.
- Include the appropriate **\d7205\tdstools** and **\d7205\f** subdirectories in your **PATH** definition.

2.2.6 Setting the board memory size

Before you can use any tool or user program which runs on your transputer evaluation board you must set up an environment variable, **IBOARDSIZE**, giving the size of the memory on the board (in bytes). To do this use the DOS **set** command. For example, to set a board size to 2 Mbytes type:

```
set IBOARDSIZE=#200000
```

You may give either a decimal or hexadecimal (preceded by '#') number. On keyboards without '#', the '$' character can be used instead. Leading and trailing spaces are prohibited.

If **IBOARDSIZE** is specified incorrectly, for example as a character, string or with leading or trailing spaces, the system defaults to a board size of 0 (zero) and it is not possible to run a program. If **IBOARDSIZE** is explicitly set to a very small value a similar error may occur.

Note: that setting very small board sizes may cause some tools to hang. This is an important point to remember when developing programs for 16-bit transputers such as the T222. Remember to reset the boardsize after testing the 16-bit transputer program as some of the tools will hang if run with a small value of **IBOARDSIZE**.

2.2.7 Setting a file system search path

To enable the tools to find libraries and include files you must set up an environment variable called **ISEARCH**. This environment variable normally contains the standard library and include file directory (**\d7205\libs**), together with any user directories as required.

Note: that unlike the DOS path you must add the closing backslash, '\', to a directory name.

Directories may be separated by a space or a semi-colon. For example to set up **ISEARCH** to point to the standard include files and libraries and to a user directory called **\mydir** type the following DOS command:

```
set ISEARCH=C:\D7205\LIBS\;C:\MYDIR\
```

2.2.8 Setting root memory size for idebug

The amount of memory on the root transputer must be defined for **idebug**, using the environment variable **IDEBUGSIZE**. This variable is set up in the same way as **IBOARDSIZE** (see section 2.2.6) and should be set to the available memory. Leading and trailing spaces are prohibited.

The debugger requires at least 1 Mbyte of memory on the root transputer to operate correctly, but 2 Mbytes or more is recommended.

2.2.9 Setting an alternative board address

The default PC bus address used by the **iserver** for locating the transputer board is **#150** for the IBM PC and **#D0** for the NEC PC. If your transputer board resides at a different address in the PC bus you should set up the environment variable **TRANSPUTER** which gives the address of the board. The address must be given in hexadecimal.

For example, if your transputer board is at address **#200**, use the following command:

```
set TRANSPUTER=200
```

You can also use the **iserver** '**SL**' option to override the address specified by the **TRANSPUTER** environment variable.

2.2.10 ITERM support for the debugger and simulator

The interactive debugger **idebug** and the simulator **isim** drive the PC screen, and have a set of PC function keys which are used for user input. These programs are portable to a variety of terminals, and use a terminal setup file, known as "ITERM" to determine the screen and keyboard characteristics. This section describes how to set up the ITERM for the IBM PC. If you are using a NEC PC then read the next section instead.

In order to use the debugger or simulator you will need to use a screen device driver that can recognise ANSI *escape sequences*. You should use either **ansi.sys** which is supplied with DOS, or **bansi.sys** which is supplied with this toolset release.

The normal **ansi.sys** supplied with a PC does not support features such as *insert line* and *delete line*. Without such features, the debugger has to redraw the whole screen whenever it scrolls. In order to overcome such limitations this release includes a replacement for **ansi.sys** which is known as **bansi.sys**. This is compatible with **ansi.sys**; it merely provides extra functionality.

Once installed it may be used with tools other than the debugger or simulator (for example, with the F folding editor).

In order to install **bansi.sys** you will need to place the following in your **config.sys** file (if you prefer to install **ansi.sys** you must add the appropriate **DEVICE** line for it) :-

```
DEVICE=C:\D7205\ITERMS\BANSI.SYS
```

This line should replace a similar line that references **ansi.sys** (if it was present).

You will need to re-boot the PC in order for the **bansi.sys** device driver to be used.

You should then set your ITERM to use **pcbansi.itm** if you have installed **bansi.sys** and **pcansi.itm** if you have installed **ansi.sys**.

```
set ITERM=C:\D7205\ITERMS\PCBANSI.ITM
```

The keyboard assignment created by this ITERM is shown in Appendix B.

2.2.11 ITERM on the NEC PC

If you are using a NEC PC computer then you must set your **ITERM** environment variable to use **necansi.itm**.

For example:

```
set ITERM=C:\D7205\ITERMS\NECANSI.ITM
```

The keyboard assignment created by this ITERM is shown in Appendix B. For the NEC PC a simple keyboard assignment has been chosen, which does not make any use of the function keys. Use of the function keys requires the keyboard to be initialised. The keyboard assignment provided will allow NEC PC users to get started with the minimum of complication. You can develop a more sophisticated ITERM using the function keys if you wish.

2.2.12 Summary of setup process

Here is a summary of the setup steps needed before the toolset may be used:

1 Set up an appropriate **iserver** for your PC and board type.

2 Set up the **FILES** variable in **config.sys**.

3 Set up the **PATH** variable.

4 Set up the **ISERVER** environment variable, if desired.

5 Set up the **IBOARDSIZE** and **IDEBUGSIZE** environment variables.

6 Set up the **ISEARCH** environment variable.

7 Set up the **TRANSPUTER** environment variable, if required.

8 Install **BANSI.SYS**, if required.

9 Set up the **ITERM** environment variable.

The environment variable commands should be placed in **autoexec.bat** so that they are set up every time the PC is switched on.

It may be necessary to increase the amount of environment space available; the next section describes how to do this.

2.3 Environment space

The PC may not have enough 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 the DOS command line.

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

2.4 Using the F editor

If you wish to use the F editor you will need to set up some environment variables and some host-specific absolute file names. These are discussed both in the **read.me** file included with the software and in the F editor manual.

The environment variables are **FTERM** which must reference an appropriate extended ITERM file for the host keyboard and screen you are using and, optionally, **FSTART** which may be used to override the pointer to the F start up file which is found in the F section of the ITERM file.

The supporting files required by the F editor are the following:

An ITERM file	`pcfold.itm`
A keyboard help file	`pcfold.hlp`
A startup file	`fold.stp`
A command file	`fold.cmd`

These are supplied in the directory `\d7205\f\data`.

The ITERM file must be identified by the `FTERM` environment variable. The keyboard help file is by default in a file with the same root name as the ITERM, but with the suffix `.hlp`. The startup file is identified by a line in the ITERM file or by the `FSTART` environment variable. The choice between these depends on the extent to which the same file is to be shared between different users, or the same user using different directories or different terminals. The command file is identified by a line in the startup file. The choice between an absolute or local file name for this is left to the user. See the *F User Guide*.

As an initial confidence test of the installation of F, go to a directory containing the ITERM file appropriate to your environment and call the F editor to edit this file. If you are new to folding you may like to edit the F tutorial file (`\d7205\f\data\ftutor.occ`), which will guide you through the facilities offered by the editor.

For the NEC PC a batch file called `necf.bat` can be run to start up the editor. This sends a file to the terminal to initialise the function keys. This file needs to be adapted to work correctly with the latest versions of DOS for the NEC PC.

2.5 Server interrupts

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

To interrupt the server, use the following procedure. Remember to enable the BREAK key first using the `DOS break` command.

Use CTRL-BREAK (in preference to CTRL-C) in order to interrupt the program. A prompt will appear. Type 'S' at the prompt, which enters a new DOS command processor. DOS commands can now be executed as necessary.

In order to return to the server type `exit`. This quits the DOS command processor and restarts the `iserver`.

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.

The user manual refers to the "host break key". For the PC toolset products this is CTRL-BREAK.

2.6 Driver program errors

The transputer based tools are executed through a driver program which itself generates error messages. For example:

```
Fatal-driver- unable to execute 'oc', Arg list too long
```

In this example the messages indicates that the DOS limit on the length of the command line has been exceeded.

Driver errors are generated for limitations or errors such as a command line too long, denial of read/write access to a file, and file or directory not found.

2.7 Transputer error flag

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, 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 is executed. In order to overcome this problem, you should run a network check program, such as **ispy**, or boot a (dummy) program that uses all of the processors in the network.

The **ispy** program is provided as part of the board support software for INMOS *i*q systems products. These products are available separately through your local INMOS distributor.

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

3 Confidence testing

This chapter describes a short procedure which may be followed to check that installation has been done correctly.

1 Set the current disk to the same disk as the toolset has been installed on. For example, if the compiler has been installed in directory **C:\D7205**, do this:

```
D>c:

C>
```

2 Set the current directory to a convenient directory for doing this test. For example:

```
C>cd \mine

C>
```

3 Copy the example **simple.occ** file to the current directory:

```
C>copy \d7205\examples\oc\simple.occ
     1 File(s) copied

C>
```

4 Use **imakef** to build a makefile which will build you a bootable file in **/ta** mode (this produces a version which will run on any 32-bit transputer):

```
C>imakef simple.bah

C>
```

If, instead of the **C>** prompt, the computer outputs the following, or something similar —

```
Error - iserver - protocol error ...
```

— it is likely that there has been some error in setting up the transputer board. In particular, please check that the wire links, accessible from the back of the PC, have been correctly installed. The transputer board's documentation should help with this.

It may also be necessary to change the default address at which the transputer is assumed to be in the PC's bus as the compiler plus the other tools are loaded at the address **#150**. For example, if your transputer board is at address **#200** then it will be necessary to add the option '**/sl 200**' to the command line when running the tools (or set the environment variable **TRANSPUTER** to 200).

5 Now use a suitable **make** tool to build the example program. Details of which **make** tools are suitable can be found in the *occam 2 Toolset User manual*. If you don't have a suitable **make** tool at present you should simply type the lines below starting **oc**, **ilink** and **icollect**.

```
C>make -f simple.mak
oc simple /ta /h /o simple.tah

ilink /f simple.lah /ta /h /o simple.cah

icollect simple.cah /t /o simple.bah

C>
```

6 Finally, the program can be run:

```
C>iserver /sb simple.bah
Please type your name :Bruce
Hello Bruce

C>
```

The output '**Hello Bruce**' comes from the **simple.occ** example program and the name you type in.

4 Building from sources

The occam 2 toolset is supplied with the source of some of the libraries, the makefile generator, the server and the driver program. This chapter gives advice on how to rebuild the binaries.

The sources are supplied in subdirectories of **\d7205\source**.

4.1 The driver program

The driver program (see section 2.1) is supplied as a single C source file called **driver.c**. This can be recompiled using Microsoft C version 5.0.

4.2 The Makefile generator

The makefile generator program **imakef** is supplied as a collection of C source files and two makefiles. The makefile **makefile.pc** compiles **imakef** for the the PC and requires Microsoft C version 5.0 or later. To recompile type:

```
C>make -f makefile.pc

C>
```

The makefile **makefile.tp** compiles **imakef** for the transputer and requires the INMOS ANSI C toolset (IMS D7214). To recompile type:

```
C>make -f makefile.tp

C>
```

4.3 The server

The **iserver** source is supplied as a collection of C source files and a collection of makefiles which compile the **iserver** for the appropriate host.

The makefile **makefile.pc** compiles **iserver** for the PC and requires Microsoft C version 5.0. To recompile type:

```
C>make -f makefile.pc

C>
```

The sources exist for building the file server to communicate with the following transputer board products:

IMS B004 (INMOS Ltd)
IMS B008 (INMOS Ltd)
IMS B010 (INMOS Ltd)
IMS B011 (INMOS Ltd)
IMS B014 (INMOS Ltd)
IMS B015 (INMOS Ltd)
IMS B016 (INMOS Ltd)
C3 QT0 (Caplin Cybernetics Corp.)

Makefiles are supplied for building the **iserver** for all the different board types supported. Source for using the file server under the Helios operating system (Perihelion Software Ltd) is also included.

4.4 The libraries

The libraries are more complicated to rebuild. If you wish to rebuild or customise the libraries the following information will be of use. It assumes that you are experienced in the use of the toolset.

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

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

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

everything has been compiled successfully, and then **ilibr** everything together to make the desired library.

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

5 Changes from previous releases

The D7205 occam 2 toolset is a new set of occam tools, giving a considerable improvement over the D705B toolset release. A summary of the major improvements is provided below:

- The toolset is compatible with the INMOS ANSI C toolset release (IMS D7214).
- The facilities for mixing C and occam have been improved. As well as the ways of calling C processes from occam supported in the D705B, a number of additional procedures are available which allow calling of C functions directly. Even functions making use of static variables can be called.
- Multiple error detection is now done by the occam compiler rather than by a separate tool (**icheck**).
- Interactive debugging on the target network is now supported, including the setting of breakpoints, and the ability to examine and modify the state of processes.
- There is an improved configuration language, supporting separate descriptions of hardware and software, and automatic placement of channels on links.
- Procedures and functions can now be defined as **INLINE**, causing the compiler to insert code inline at a call.
- There is a new form of assembler insert, introduced by the keyword **ASM**, which provides better facilities than the old **GUY** keyword. The **GUY** form of insert has been retained in this release for compatibility.
- Tools for creating EPROM program files are now included in the toolset.
- A folding editor called 'F' is included in the package.

The object file format has been changed since the previous release; the tools now support an object file format called TCOFF (Transputer Common Object File Format) which is becoming a standard across all INMOS toolsets. A conversion tool called **icvlink** is provided to convert old format object files into the new format, but to use the new features of the toolset existing code should be recompiled into the new format.

5.1 Adaptations required to existing program source

The occam programming language implemented by the toolset is essentially the same as that implemented by the D705B, with a number of extensions. The main change is in the configuration language.

The changes required to configuration programs include:

- Preparing a separate description of the hardware, including the allocation of names to processors in the network.
- Declaring the host connection, from which the network is booted, and placing channels on that link.
- Altering the **PLACED PAR** to refer to the processor names rather than allocating them numbers and types, and removing all of the channel placements on link numbers (which can now be determined automatically by the configurer). If the specific placement of a channel on a link is needed, the link must be given a name in the hardware description; that name can then be used in a **PLACE** statement in the software description.
- Combining the hardware and software descriptions in a file, with the appropriate keywords to distinguish them.

There are many changes in the compilation and configuration tools, which have been re-designed and re-implemented since the D705B. The tools have a greater number of options, to allow more flexibility and control over the code being generated. The creation of bootable files for processor networks is done in two stages, using the tools **occonf** and **icollect**. The **icollect** tool is also used in place of **iboot** to create a bootable program for a single processor.

The debugger **idebug** has been extended to support interactive debugging of programs in a network. This requires changes in the code generated by the compilation tools. Interactive debugging code is produced by default; the tool option **Y** is available on a number of tools to suppress the generation of interactive debugging support, and generate more efficient code.

In spite of these changes, the structure of source files which go to make up an occam program is similar to that of the D705B. Once the required changes have been made to the configuration source, the **imakef** tool can be used to generate a make file containing all the tool invocations required to build the bootable file. The file extension conventions of **imakef** are the same as in the D705B, although the tools themselves do not specifically require these conventions. Running the resultant make file may indicate a number of small changes required to the source files, which it should be possible to resolve from information available in the user manual.

5.1.1 Changes in the implementation of occam

There are a number of changes in the implementation of occam. These are described in this section.

5.1.2 Changed implementation of channels

PLEASE READ THIS SECTION CAREFULLY. IF YOU THINK THIS APPLIES TO YOU PLEASE READ THE APPROPRIATE SECTION OF THE USER MANUAL BEFORE ATTEMPTING TO COMPILE AND RUN occam PROGRAMS WITH THIS TOOLSET.

The low level behaviour of channels has changed since the previous occam toolset release. This will not affect programmers working entirely in occam. However, programmers using **GUY** and **ASM** code, and **PLACE**ing channels, must be aware of the change. For example, programs containing an array of channels placed on the transputer links must be changed. The change may also affect programmers using **KERNEL.RUN** and its associated predefines.

The changes are described in section 10.1.3 of the **Low level programming** chapter of the user manual. The ability to manipulate channels has been removed from the **GUY** assembly facility, to ensure that code manipulating channels is recoded in **ASM**.

5.1.3 UNDEFINED and UNIVERSAL error modes

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

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

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

The system of error modes is described in detail in the user manual.

5.1.4 Transputer types

The support for transputer types and classes has been extended to incorporate new transputer types introduced since the D705B. It has also changed its implementation to allow more flexible creation of libraries, and to allow the creation of programs which are guaranteed to run on a range of transputer types. The main difference for existing programs is that **TA** code can no longer be called from **T8** code, as the method used to return floating point results from functions differs between the classes. There is no longer a **TC** class.

The system of transputer types is described in detail in the user manual.

5.1.5 #SC directive

The #SC compiler directive is no longer supported.

5.1.6 Stricter checking of PRI PAR

The compiler does stricter checking of nested **PRI PAR** constructs, including checks across separate compilation boundaries, to enforce the occam rule that a **PRI PAR** may not be nested inside another **PRI PAR**.

5.1.7 Single processor programs

In the D705B toolset, single processor programs could have bootstraps added by a tool called **iboot**, without requiring configuration. The equivalent tool in this toolset is **icollect**, with a **T** option. Unlike **iboot**, it enforces the rule that the server channels must be **CHAN OF SP**.

5.1.8 Usage checking rules

The implementation of usage checking has changed slightly in this release of the occam compiler. In particular it enforces the rule that: for a channel array passed into a procedure, all channels must be used in the same direction. Usage checking can still be disabled if absolutely necessary.

5.2 Changes in libraries

This section lists the changes and bug fixes to the libraries of the occam 2 toolset since the D705B and equivalents. This section is in the form of a set of technical notes on the changes to the library implementations.

5.2.1 Compiler libraries

General: It has been enforced that no routine uses vector space.

General: There used to be a number of entry points that were not documented for use, but which were used by the compiler. These entry points were actually callable in user's source code. All such entry points have either been removed completely or given illegal occam identifiers so as not to be legally callable.

FLOATING.UNPACK and **DFLOATING.UNPACK** for T2 and TA: arguments of infinities, NaNs and zeroes used not to cause any error – fixed so that they **CAUSEERROR**.

UNPACKSN for any 32-bit transputer now exists, whereas it was previously only available for TB.

ROUNDSN for any 32-bit transputer now exists, whereas it was previously only available for TB.

FRACMUL for any 16-bit transputer now exists, whereas it was previously only available for 32-bit transputers.

5.2.2 debug.lib

This is a new library. However it does contain the routine **DEBUG.TIMER** which has been moved here from **process.lib**.

DEBUG.TIMER used to be called **debug.timer** and used to be available only for 32-bit transputers; it is now available on all transputers. Further, it used to terminate after one day if not explicitly stopped; now it will only terminate when explicitly stopped.

5.2.3 convert.lib

General: It has been enforced that no routine uses vector space.

REAL32TOSTRING and **REAL64TOSTRING** for all transputers: formatting and rounding have been much improved.

5.2.4 crc.lib

General: It has been enforced that no routine uses vector space.

5.2.5 hostio.lib

General: Toolset User Manual documentation much improved, especially noting the fact that a server dependent failure result for most routines is ≥ **spr.operation.failed** rather than identically equal to it.

so.write.string, so.write.string.nl, so.fwrite.string, and **so.fwrite.string.nl** for all transputers: used to be a maximum limit to the size of string that these routines could write – fixed so that there is no limit.

so.read.echo.hex.int, so.read.echo.hex.int64 and **so.read.echo.any.int** for all transputers: used not to accept lower case hex digits – fixed.

af.to.sp has been removed from this library.

so.popen.read for all transputers: used to attempt to open a file the name of which resulted from a truncation of part of the environment string and **filename** if their concatenation was not long enough for **full.name**; further, angle brackets were not recognised in a directory specifier (relevant for the VAX) – both fixed.

so.fwrite.hex.int32, so.write.hex.int32, so.read.echo.hex.int32, so.read.echo.int32, so.fwrite.int32 and **so.write.int32** are new routines to this library.

so.fwrite.int, so.write.int, so.fwrite.int64 and **so.write.int64** used to accept negative field widths as equivalent to zero – fixed so that a negative field width is an error.

so.multiplexor and **so.overlapped.multiplexor** have been renamed as **so.pri.multiplexor** and **so.overlapped.pri.multiplexor**, respectively. Further, it is now valid for **SIZE queue** in the latter to be zero, in which case input is waited for on **stopper**.

so.multiplexor and **so.overlapped.multiplexor** have been replaced by new, similar, routines. The old versions of these routines (see above) were not fair multiplexors; the new versions attempt to show some degree of fairness.

5.2.6 snglmath.lib

General: It has been enforced that no routine uses vector space.

POWER for T2: the size of the domain before infinity is returned has been slightly extended – change.

5.2.7 dblmath.lib

General: It has been enforced that no routine uses vector space.

DPOWER for all transputers: the size of the domain before infinity is returned has been slightly extended – change.

5.2.8 tbmaths.lib

General: It has been enforced that no routine uses vector space.

General: There used to be a number of entry points that were not documented for use, but which were, and still are, used as support by the major routines of this library. These entry points were actually callable in user's source code. All such entry points have been given illegal occam identifiers so as not to be legally callable.

5.2.9 process.lib

debug.timer has been moved from this library into **debug.lib** and has been renamed as **DEBUG.TIMER**.

As this library now only contains evaluation board support processes, which are useful only to a minority of customers, it has been removed from the this release. The source code for the processes is included in the **\d7205\examples\oc** directory, for those customers still requiring this software.

5.2.10 streamio.lib

ss.write.int and **ss.write.hex.int** for all transputers: used to only allow up to 15/10 characters, respectively, to be written out – fixed to allow unlimited field widths.

ss.write.int64 and **ss.write.hex.int64** for all transputers: used to only allow up to 32/20 characters, respectively, to be written out – fixed to allow unlimited field widths.

ss.write.real32 for all transputers: used to write out up to 40 characters – changed to allow no more than 24.

ss.write.real64 for all transputers: used to write out up to 60 characters – changed to allow no more than 30.

ss.scrstream.to.array for all transputers: used not to store all of the string into the array on receipt of the **st.out.string** tag – fixed.

so.keystream.from.kbd for all transputers: used to accept negative values for **ticks.per.poll** – fixed so that negative values are an error.

so.keystream.from.file for all transputers: used to always return ≥ **spr.operation.failed** on reaching the end of file – fixed.

so.keystream.from.stdin for all transputers: used to return ≥ **spr.operation.failed** on reading the end of file from the keyboard – fixed.

ss.write.int and **ss.write.int64** for all transputers: used to accept negative values for a field width as equivalent to zero – fixed so that negative values for the field width is an error.

ss.scrstream.multiplexor is a new routine to this library.

5.2.11 string.lib

General: It has been enforced that no routine uses vector space.

5.2.12 xlink.lib

General: It has been enforced that no routine uses vector space.

Appendices

A Distribution kit

This appendix lists the files which make up the distribution kit for this version of the occam 2 toolset. Each filename is accompanied by a short description of the file's function.

A.1 Directory `\d7205\itools`

This directory contains the transputer-hosted tools.

`oc.btl`	occam compiler transputer-bootable code
`oc.exe`	occam compiler driver program
`ilink.btl`	linker code
`ilink.exe`	linker driver program
`ilibr.btl`	librarian code
`ilibr.exe`	librarian driver program
`icollect.btl`	collector code
`icollect.exe`	collector driver program
`ilist.btl`	lister code
`ilist.exe`	lister driver program
`occonf.btl`	configurer code
`occonf.exe`	configurer driver program
`idebug.btl`	debugger code
`idebug.exe`	debugger driver program
`ieprom.btl`	EPROM file constructor code
`ieprom.exe`	EPROM file constructor driver program
`isim.btl`	simulator code
`isim.exe`	simulator driver program
`icvemit.btl`	memory description format converter code
`icvemit.exe`	memory description format converter driver program
`iemit.btl`	external memory interface program code
`iemit.exe`	external memory interface program driver program
`imakef.btl`	makefile generator code
`imakef.exe`	makefile generator driver program
`idump.btl`	core dumper code
`idump.exe`	core dumper driver program

iskip.btl	skip loader code
iskip.exe	skip loader driver program
icvlink.btl	link format converter code
icvlink.exe	link format converter driver program

A.2 Directory \d7205\tools

This directory contains the PC-hosted tools.

icollect.exe	collector PC executable
icvemit.exe	memory description format converter PC executable
icvlink.exe	link format converter PC executable
iemit.exe	external memory interface program PC executable
ieprom.exe	EPROM file constructor PC executable
ilibr.exe	librarian PC executable
ilink.exe	linker PC executable
ilist.exe	lister PC executable
imakef.exe	makefile generator PC executable

A.3 Directory \d7205\libs

This directory contains the libraries and include files.

callc.lib	calling C from occam library
convert.lib	string conversion library
crc.lib	CRC library
dblmath.lib	64-bit arithmetic library
debug.lib	debug procedure library
hostio.inc	host i/o library header file
hostio.lib	host i/o library
hostio.liu	host i/o library usage file
linkaddr.inc	link address include file
linkboot.lib	link booting library

mathvals.inc	maths value definitions
msdos.inc	MSDOS library include file
msdos.lib	MSDOS library
occam2.lib	occam compiler library for T2
occam8.lib	occam compiler library for T8
occama.lib	occam compiler library for TA, TB and T5
occamutl.lib	occam compiler utilities library
occam2.lnk	linker file for using occam library for T2
occam8.lnk	linker file for using occam library for T8
occama.lnk	linker file for using occam library for TA, TB and T5
occonfio.lib	configurer i/o library
romboot.lib	boot from ROM library
snglmath.lib	32-bit arithmetic library
streamio.inc	stream i/o include file
streamio.lib	stream i/o conversion library
streamio.liu	stream i/o conversion library usage file
string.lib	string handling library
string.liu	string handling library usage file
sysproc.lib	system library
tbmaths.lib	TB maths library
ticks.inc	transputer clock tick rate include file
virtual.lib	occam compiler i/o library
xlink.lib	extraordinary link handling library

A.4 Directory \d7205\iterms

bansi.sys	screen device driver
pcansi.itm	ITERM file for use with ANSI.SYS
pcbansi.itm	ITERM file for use with BANSI.SYS
necansi.itm	ITERM file for use with an NEC PC
ansi.itm	ITERM file for use with a standard ANSI terminal
readme.txt	information file

A.5 Directory \d7205\examples\bootstrp

This directory contains source files of example bootstraps.

A.6 Directory \d7205\examples\idebug

This directory contains the example sources for using the debugger.

A.7 Directory \d7205\examples\mixed

This directory contains examples of mixed language programming.

A.8 Directory \d7205\examples\mixconf

This directory contains examples of mixed language configuration.

A.9 Directory \d7205\examples\oc

This directory contains examples of using the occam compiler.

A.10 Directory \d7205\examples\occonf

This directory contains examples of using the configurer.

A.11 Directory \d7205\examples\sorter

This directory contains the source of the sorter example program.

A.12 Directory \d7205\iserver

isvrb04.exe	host file server and loader program (IBM PC version for B004)
isvrb08.exe	host file server and loader program (IBM PC version for B008)
isvrnec.exe	host file server and loader program (NEC PC version)
iserver.exe	the server which was selected at installation time

A.13 Directory \d7205\source\iserver

Contains the sources for the host file server.

A.14 Directory \d7205\source\imakef

Contains the sources for the makefile generator.

A.15 Directory \d7205\source\driver

Contains the source of the driver program.

A.16 Directory \d7205\source\libs\convert

Contains the source of the **convert** library.

A.17 Directory \d7205\source\libs\hostio

Contains the source of the **hostio** library.

A.18 Directory \d7205\source\libs\maths

Contains the source of the maths libraries.

A.19 Directory \d7205\source\libs\msdos

Contains the source of the **msdos** library.

A.20 Directory \d7205\source\libs\streamio

Contains the source of the **streamio** library.

A.21 Directory \d7205\source\libs\string

Contains the source of the **string** library.

A.22 Directory \d7205\tdstools\itools

idirect.btl	directive converter code
idirect.exe	directive converter driver program
iflat.btl	TDS to toolset converter code
iflat.exe	TDS to toolset converter driver program
iflist.btl	TDS lister code
iflist.exe	TDS lister driver program

A.23 Directory \d7205\tdstools\tools

iflat.exe	PC-hosted TDS to toolset converter
iflist.exe	PC-hosted TDS lister

A.24 Directory \d7205\tdstools\libs

streamco.lib	stream conversion library
streamco.liu	stream conversion library usage file

A.25 Directory \d7205\tdstools\set

clist.set	driver file for the **iflist** program
null.set	driver file for the **iflist** program
occam.set	driver file for the **iflist** program
trans.idi	driver file for the **idirect** program

A.26 Directory \d7205\tdstools\streamco

Contains source of the TDS to toolset stream conversion library.

A.27 Directory \d7205\f\itools

f.btl	folding editor code
f.exe	folding editor driver program
necf.bat	NEC PC editor batch file

A.28 Directory \d7205\f\tools

f.exe	PC-hosted folding editor
necf.bat	NEC PC editor batch file

A.29 Directory \d7205\f\data

bansi.sys	screen device driver
fold.cmd	editor command file
fold.stp	editor startup file
ftutor.occ	editor tutor
ascii.itm	ITERM for ASCII keyboard
ascii.hlp	keyboard help file for ASCII keyboard
read.me	information file
pcfold.itm	ITERM file for PC terminal
necfold.itm	ITERM file for NEC terminal
pcfold.hlp	keyboard help file for PC terminal
necfold.hlp	keyboard help file for NEC terminal
necini25.lis	NEC initialisation file
tds3keys.ld	editor keyboard setup file
tds3keys.tbl	editor keyboard setup table
doskeys.ld	keyboard reset file
doskeys.tbl	keyboard reset table

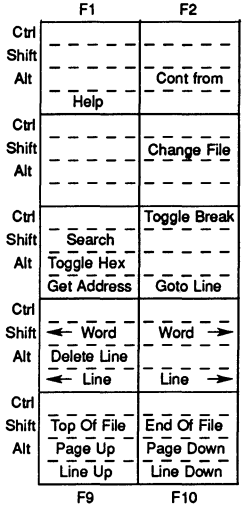
A.30 Directory \d7205\install

Contains source files associated with installing this release.

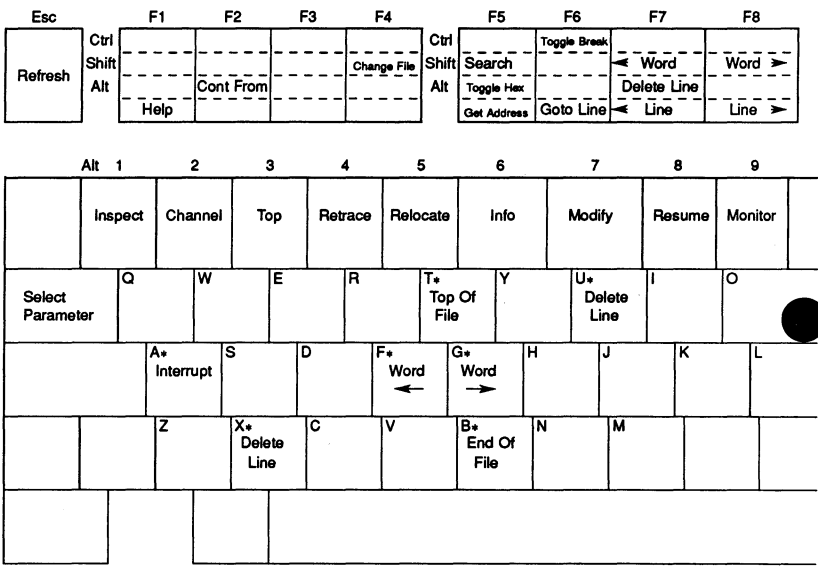
B Debugger function keys

This appendix gives the keyboard assignments for the debugger symbolic functions for both the IBM PC and compatibles (**PCANSI.ITM** and **PCBANSI.ITM**), and the NEC PC (**NECPC.ITM**). Some of the keys are applicable to the simulator as well.

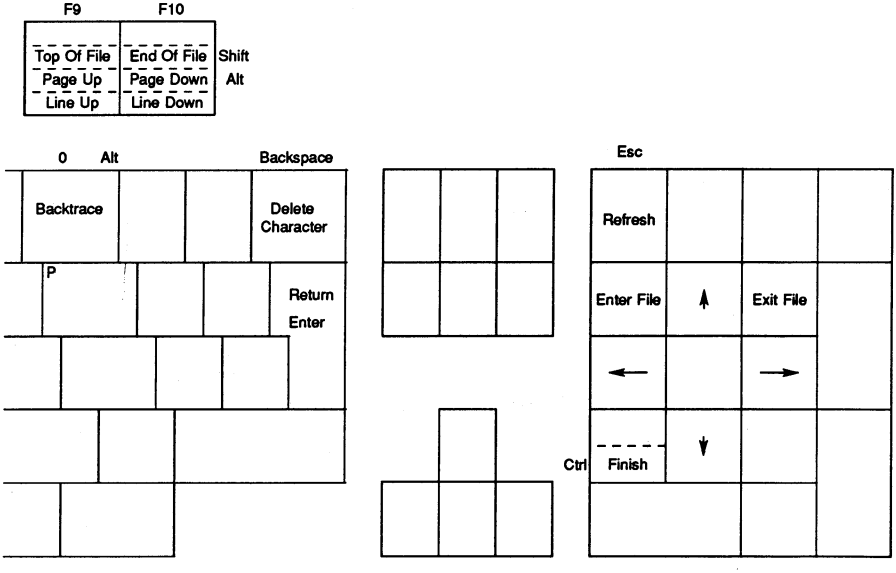
B.1 IBM PC LH-keypad



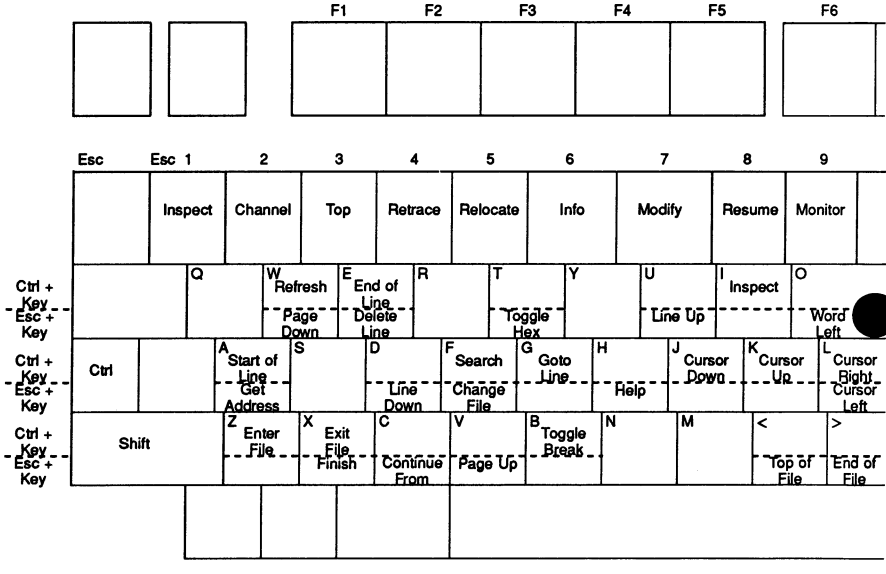
B.2 IBM PC main keyboard

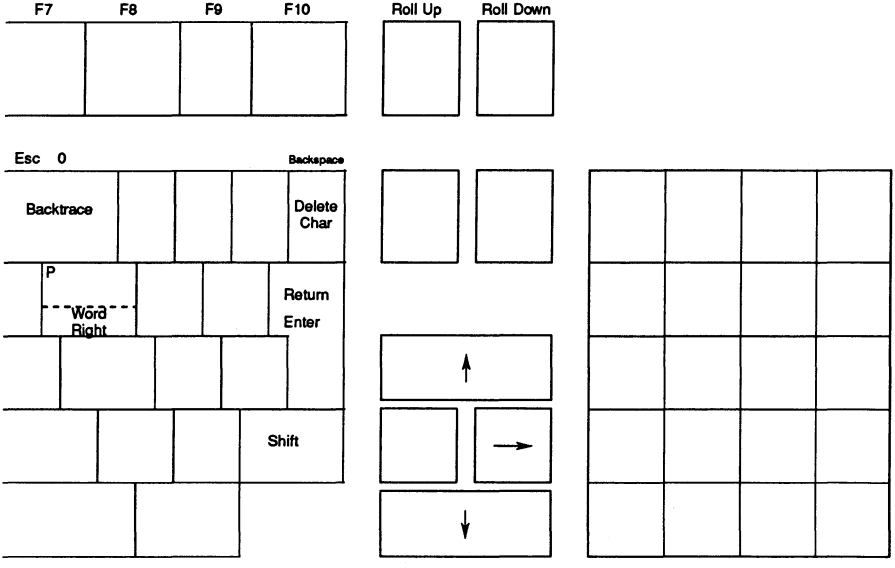


* Ctrl + key


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

B.3 NEC PC keyboard layout




F7
F8
F9
F10
Roll Up
Roll Down
Esc 0
Backspace
Backtrace
Delete Char
P
Word Right
Return
Enter
Shift

inmos

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 Centre
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.
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

INMOS Business Centre
SGS-THOMSON Microelectronics Inc.
1310 Electronics Drive
Carrollton
Texas 75006
Telephone (214) 466 8844
Fax (214) 466 7352

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

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 10

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