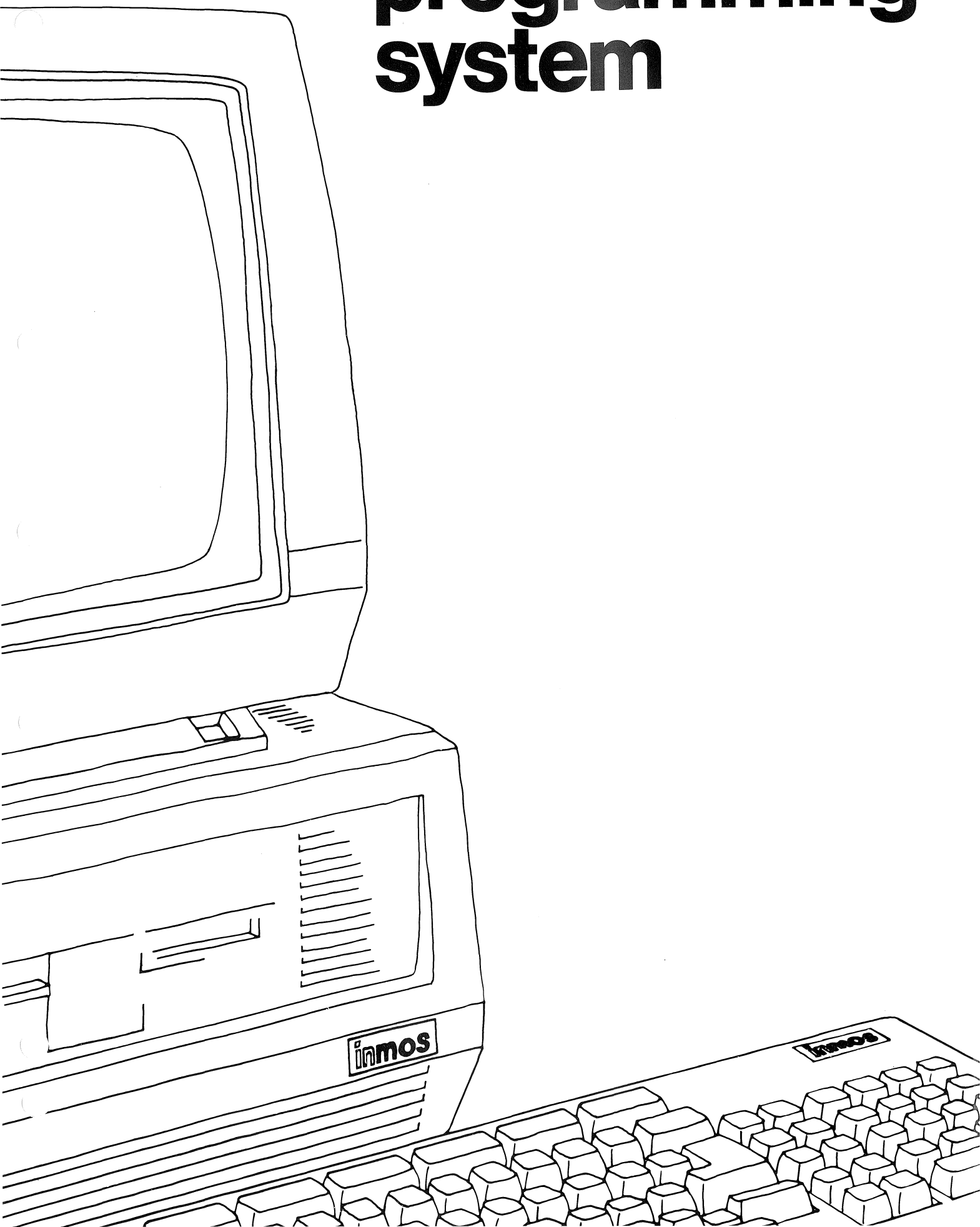
# occam® programming system



## Occam

Occam is the first commercially available language to be based on the concepts of concurrency and communication. These concepts enable today's applications of microprocessors to be implemented more efficiently. They are essential for tomorrow's systems built from multiple interconnected transputers.

Occam is easy to understand and encourages structured programming, with clean interfaces between concurrent processes. Its syntax is specially designed for interactive use.

Occam overcomes the problems of programming microprocessors by formalising the notions of input, output, interrupts, priority and real–time. By introducing these ideas into the language, it is not necessary to use real–time executives or machine level debugging.

An occam program naturally reflects the structure of the application, describing how the hardware is arranged and providing the specification and implementation of each component.

Occam has a formal basis, and its use improves confidence in the correctness of a program. Programs can be transformed, preserving functionality, in the same way as logic functions can be transformed using Boolean algebra.

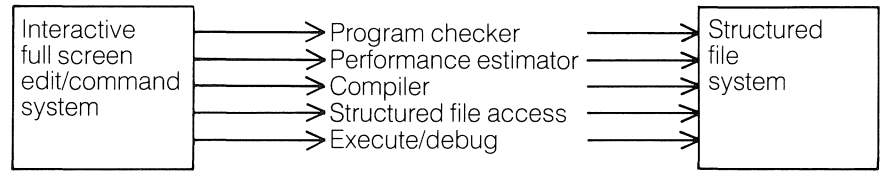
Product quality benefits from the use of occam for all stages of design and implementation, and from the natural use of structured programming.

## The occam programming system

The occam programming system provides a complete programming environment for the generation of reliable, well structured and efficient programs. The structured editing facilities provided by the fully integrated user interface mirror the hierarchical structure of the occam application under development. The integrated system provides a secure mechanism for separate compilation and the use of separate files, allowing fast editing and re–compilation for minor program changes.

Major system functions are provided as components of a toolkit. The individual tools are invoked by function keys, and the integrated editing system allows the application of any tool to any part of the program under design.

The benefits of the occam programming system combine to provide excellent design productivity, and greatly increase confidence in the timely and accurate implementation of highly concurrent and real time systems.



**Occam and its programming environment**

Occam is based on the principles of concurrency and communication. Concurrency allows the many components of a system to execute simultaneously while the communications allow them to work together.

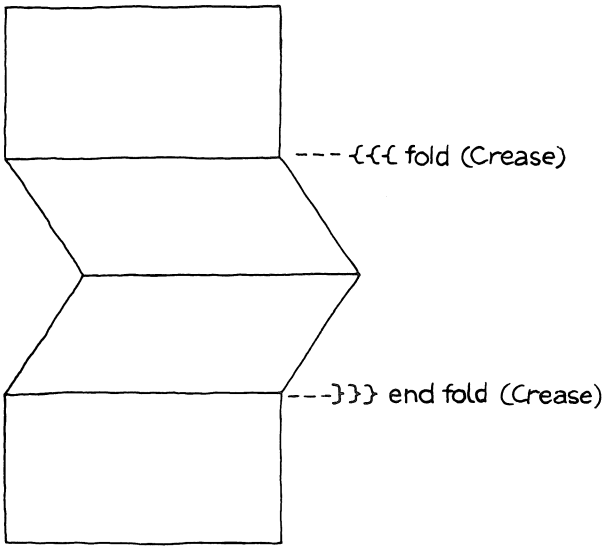
Occam can capture the hierarchical structure of a system by allowing an interconnected set of processes to be regarded from the outside as a single process. In order to assist the programmer in the creation of a program, INMOS has based the programming environment on the same notion of hierarchical structure.

**User interface**

The user interface is based on a full screen structured editor, exploiting the concept of folding. Folding provides a very effective method of navigating around and viewing selected parts of a large design, and yet operates within the constraints of an ordinary text vdu. The programmer's dependence on hard copy listings is much reduced, and the system is able to exploit the structure to provide facilities such as compilation control (exactly those parts of a program which have been changed are recompiled), navigation to the part of the program where an error (compiletime or runtime) was detected, as well as the editing convenience of being able to reorganise the major structure of a program simply and easily.

**Folding**

Folding is analogous to taking a letter or document with headed paragraphs and then folding the paper such that the text of the paragraph is invisible, leaving only the heading visible.




With a suitably folded program the programmer can see the structure of the program at a glance without any distracting detail. Individual sections can then be unfolded so that the detail can be worked on. The user can create and delete folds at will as the text of the program is created and edited. The editor imposes no restriction on folding, which can be nested to any depth.

A program, such as the example described in the occam brochure, would initially be displayed as a single fold:

```
... system
```

On opening the fold, the program would be displayed as:

```
{{{  system
CHAN Echo, App.in, App.out:
PAR
  ... keyboard handler
  ... application
  ... screen handler
}}}
```

The screen handler could now be displayed by itself, eg

```
{{{  screen handler
... declarations and initialisation
  WHILE running
    SEQ
      ... reset alarm clock
      ALT
        ... deal with Echo channel
        ... deal with App.out channel
        ... and timeout
}}}
```

It is now possible to view how input from the application is treated:

```
{{{  screen handler
... declarations and initialisation
  WHILE running
    SEQ
      ... reset alarm clock
      ALT
        ... deal with Echo channel
        {{{ deal with App.out channel
        App.out ? ch
          IF
            ch = terminating.character
              running := FALSE
            TRUE
              Screen ! ch
        }}}
        ... and timeout
}}}
```

**Secure checking**

At any time the user may check the syntax of the source code. This will locate the first error in the source and automatically position the cursor at the appropriate point in the text, opening folds as required, to allow the user to make the correction. The usage of variables and channels by concurrent processes are also checked at the same time, to ensure that variables are accessed by only one process at a time, and that each channel is use for output in one process and for input in one other process.

Some errors (for example array bounds violation) cannot be detected at compile time. In all cases, the occam programming system provides a safe environment in that either the editor/checker catches the error, or it is detected at runtime.

**Secure compilation**

The contents of a fold may be compiled separately. Separate compilation can significantly reduce the recompilation time after a minor change to the program. The compiled code is stored in a separate sub–fold within the fold being compiled. The integrated environment provides automatic recompilation of any part of the program which has been edited (or compiled with a previous version of the compiler) and ensures that source and object code are kept in step.

**Running programs**

The compiled code is linked with an appropriate kernel, which implements the concurrency necessary to execute occam and provides an interface to the host computer's operating system for access to the terminal and files.

To aid the debugging of occam programs, the occam programming system provides the ability to locate the position, in the occam source, of the cause of a runtime error. This can be used in cases such as deadlock, array bounds range check, divide by zero, `STOP`.

**File folds**

Special function keys apply tags to a fold to indicate that it should be filed separately. Filed folds are only read into the system when needed, reducing disk traffic and improving system performance.

**Editing facilities**

In addition to the specialised facilities, the editor supports the normal editing functions of cursor movement, screen position control, character insertion and deletion (including undelete), line (and therefore fold) movemement, deletion (including undelete) and duplication, search and replace (applied to the contents of a fold), and output of listings.

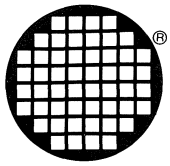
**Performance estimator**

A performance estimator is included which performs a static analysis of the program within fold and reports transputer code size and execution time.

**Toolsets**

The toolsets provided will depend on the host operating system environment. In early releases, a fixed toolset is provided, in subsequent releases, the user will have the ability to select an appropriate toolset and develop further tools in occam.

inmos®



INMOS Limited
Whitefriars
Lewins Mead
Bristol BS1 2NP
England
Telephone (0272) 290861
Telex 444723

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

INMOS Corporation
PO Box 16000
Colorado Springs
CO 80935
USA
Telephone (303) 630 4000
TWX 910 920 4904

INMOS SARL
Immeuble Monaco
7 rue Le Corbusier
SILIC 219
94518 Rungis Cedex
France
Telephone (1) 687 22 01
Telex 201222



September 1985
72-OCC-028-000