Occam related products

Occam programming manual

A tutorial introduction and reference manual for the first release of the language.

Occam evaluation kit

A low cost software package intended to act as an introduction to occam. It will run on most machines, including the Apple II, IBM PC and VAX. The kit comprises an integrated full screen editor/compiler allowing medium sized occam programs to be written and run on the host, together with a language reference manual, extensive examples and installation notes.

Occam development software

For development of applications in occam, a range of support products is provided. These include compilers, together with appropriate tools, optimised for occam program development, which are intended to run on a variety of widely available hosts, generating target code for a variety of processors.

Silicon products

The IMS T424 transputer is a 32 bit microprocessor providing 10 MIPS, with 4 Kbytes of static RAM, a 32 bit multiplexed memory and peripheral interface, and four standard INMOS links.

```
CHAN keyboard AT 2:
CHAN screen AT 1:
CHAN echo,applicationIn,applicationOut:

DEF endbuffer = -3:

PAR

    -- echo process
    VAR ch:
    WHILE TRUE
        SEQ
            keyboard?ch
            echo!ch
            applicationIn!ch

    ...-- screen mixer process
    ...-- application process
```

inmos **Data card**

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

INMOS Corporation
PO BOX 1600 Colorado Springs CO 80935 USA
Telephone (303) 630 4000 Telex 910 920 4904

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

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

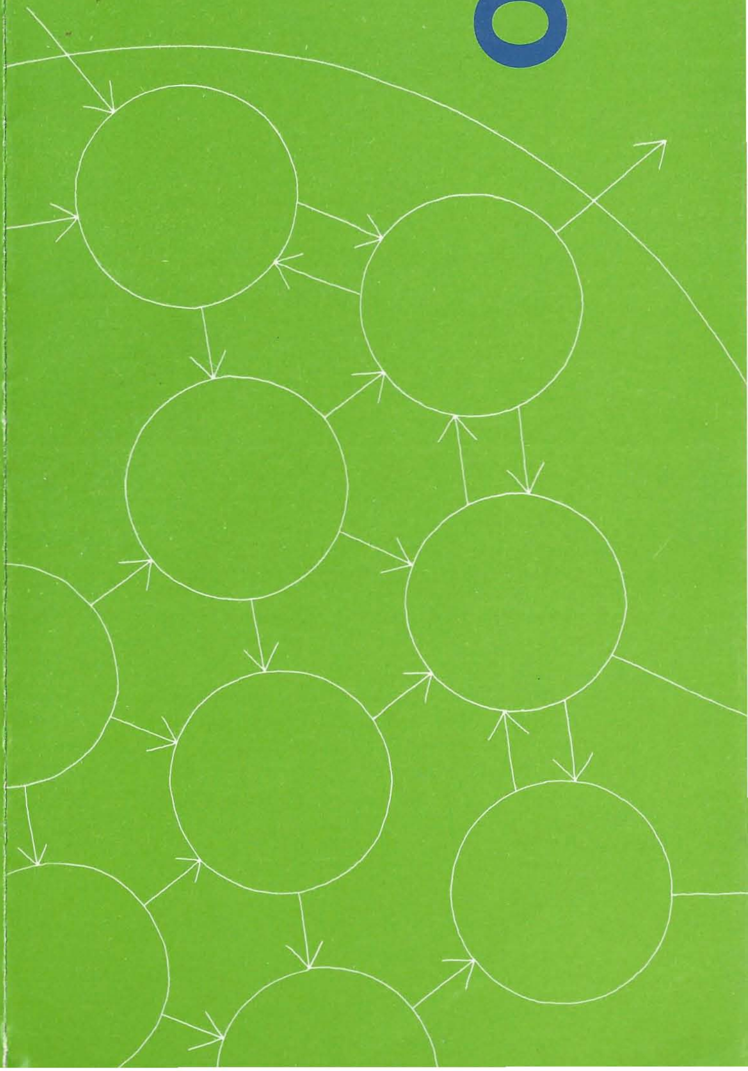
For design data refer to INMOS published Data Specifications from which these extracts are taken.

June 1984

72 OCC 007 00

Data card

occam™



Occam

Occam is a simple programming language, based on the concepts of concurrency and communication.

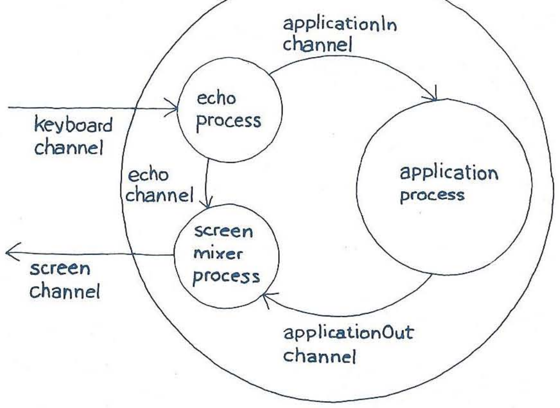
These concepts are central to today's applications of microprocessors and computers, and will play an even more important role in the future when multitudes of computers are connected to form intelligent systems.

Occam is intended for the professional design engineer/programmer.

It is simple, easy to learn and is oriented to interactive work-station based use. Occam's keywords and operators can exactly mirror system structure, enabling complex applications to be designed and programmed in a concise and readable form. As a result, design and implementation costs and times are reduced.

Occam may be used as a combination design, simulation and implementation language.

Complete hardware/software systems may be described in occam. Executing the program provides an efficient system simulation tool. The program describing the hardware components of the system can act as a formal design description for any new hardware design, while the program describing the software components may be used as the actual software for the final product.



Features

Model

Systems are described as a collection of concurrent processes, which communicate using channels.

Implementation

An occam program may be executed on a network of interconnected computers, each executing one of the concurrent processes. However, with no changes except to configuration details it may be implemented on any smaller network or a single computer, with each computer sharing its time between its set of concurrent processes.

Structure

Programs are constructed from processes combined together using constructors. The primitive processes of input, output and assignment form the lowest level of processes in a program.

Input and output

A channel provides communication between two concurrent processes. The communication is synchronised. One process must be an input process and the other an output process. The communication takes place only when both processes are ready, when the values are copied from the output process to the input process.

```
InputChannel ? char

OutputChannel ! char
```

Sequence

The sequence constructor defines a process whose component processes are executed one after the other in the order in which they appear. A sequence construct terminates after the last of its component processes has terminated.

```
SEQ
  InputChannel ? char
  OutputChannel ! char
```

Parallel

The parallel constructor defines a process whose component processes are executed concurrently.
A parallel construct terminates only when all its component processes have terminated.

```
PAR
  out1 ! 'a'
  out2 ! 'b'
```

Alternative

The alternative constructor defines a process, each component of which has an input process as its first component. The first process able to complete its input is chosen for execution. The alternative construct terminates when the chosen process terminates.

```
ALT
  in1 ? char
    out ! char
  in2 ? char
    out ! char
```

Conditional

The conditional constructor defines a process, each component of which has a condition as its first component. The conditions are tested in sequence. If a condition is found that evaluates to true that process is chosen for execution. The conditional construct terminates when the chosen process terminates.

```
IF
  x < 0
    x:= -x
  x >=0
    SKIP
```

Repetition

The repetition constructor defines a condition and a process which will be repeatedly executed until the result of evaluating the condition is false.

```
WHILE x > 0
  SEQ
    in ? x
    out ! x
```

Time

A clock local to each computer is maintained and may be accessed via the special channel TIME and used to control the execution of a process.

```
TIME ? AFTER e
```

Abstraction

A name can be given to the text of a process. The text will be substituted for all occurences of the name in textually subsequent processes. Channels, variables and other names may be used as parameters when textual substitution takes place.

```
PROC echo(CHAN in,out)=
  WHILE TRUE
    VAR x:
    SEQ
      in ? x
      out ! x:
```