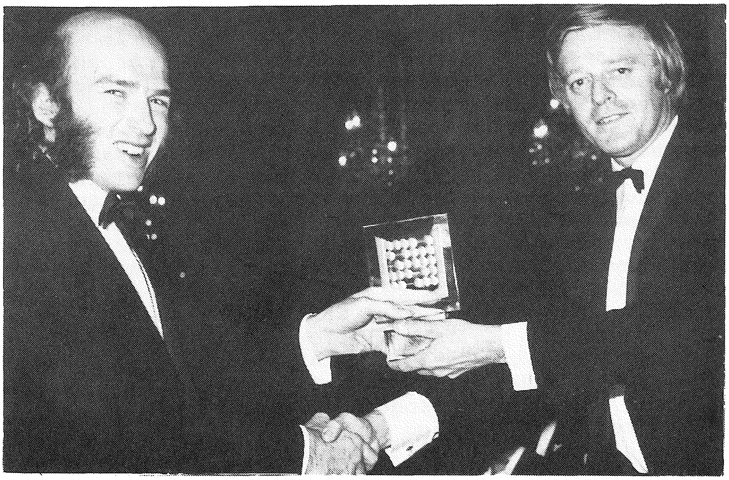
# occam™ user group · newsletter

No. 2 January 1985

CONTENTS


David May receiving the British Computer Society Award for technical innovation from John Butcher MP, Parliamentary Under-Secretary of State for Industry.

## NOTE FROM THE EDITOR

Here at last is the second issue of the newsletter. It is the same mixture of announcements, lists and longer technical articles. Two of the latter result from talks given at the September Technical Meeting. It is hoped that the list of User Group members will be helpful - this will be kept up to date in subsequent issues.

Please keep the contributions coming. Those describing practical experiences with occam will be particularly welcome. The editor is:

Martin Bolton
Dept of Electrical and Electronic Engineering
University of Bristol
Queen's Building
University Walk
BRISTOL BS8 1TR

## OCCAM GETS A BRITISH COMPUTER SOCIETY AWARD

The British Computer Society have made their 1984 Technical Award to INMOS for OCCAM. The President of the BCS, Dr Ewan Page, read the following citation at a dinner held at the Dorchester Hotel, Park Lane, London on 22nd October.

"OCCAM was designed to enable users to program future systems consisting of hundreds or thousands of interconnected computing elements. OCCAM adds the idea of a process, which represents a computer running a program, and interprocess communication, which represents input and output between computers, to the essential concepts of present day sequential languages. The result is a simple and elegant language which can be used to improve the design and implementation of current microprocessor applications."

David May of INMOS, the chief architect of OCCAM, received the award from Mr John Butcher, who was deputising for Mr Norman Tebbit, Secretary of State for Trade and Industry.

At a meeting held at the Royal Society on 3rd December, David gave a technical presentation including a demonstration of OCCAM programs running on transputer prototypes.

MDP

## OCCAM USER GROUP TECHNICAL MEETING

Report by Michael Poole

The first technical meeting of the occam user group was held at the Watershed, Bristol on 21st September 1984. This group is the first special interest group of INMOS customers to be established and is open to anyone with an interest in occam, the programming language for the transputer.

About 100 people attended, from universities, polytechnics, government research and industry. A small number of visitors from Europe were present.

The meeting took the form of nine lectures by members, each lasting 20-30 minutes. The speakers discussed their experience with the occam language both as an aid to design and programming of forthcoming real systems and as a theoretical tool for discussing the properties of systems of concurrent processes and their cooperation.

OCCAM has been designed to facilitate formal reasoning about programs and deriving from this the ability to transform programs from one form to another to optimise their performance on different hardware. Bill Roscoe from Oxford University discussed the differnt ways of formally specifying the meaning of programs, and specifically his own work on algebraic semantics. Richard Bornat from Queen Mary College discussed a general problem of a process wishing to send output to another process, but with the option to proceed on other work if the other cannot receive the output.

Roland Backhouse from Essex University and John Ainscough from Brunel University reported on their experiences in the university teaching environment. Essex have used occam as a vehicle for teaching the concepts and practice of concurrent programming, and Roland stressed the importance of providing students with substantial supporting software for display management, etc. so they can concentrate their efforts on the concurrency. John had given a student the task of writing a complete occam compiler as a final year project, and it was impressive how someone with little relevant previous experience was able to create such a compiler which compiles five times as many lines per minute as the standard compiler for Pascal on VAX/VMS.

Don Fay from Queen's University Belfast discussed his experience with one of occam's intellectual parents (Hoare's Communicating Sequential Processes, CSP for short) and more recently with occam. He was concerned with the way each language tends to influence the way one considers the coding of a problem, and was very appreciative of the way occam had clarified things for him. He found several kindred spirits in the audience, however, when he mentioned his difficulty in getting occam programs to stop gracefully.

The people with real applications in mind were Mike Reeve from Imperial College, London and two people from RSRE

Malvern. Mike Reeve is a member of the SERC funded ALICE project which is concerned to build a novel hardware solution to the problem of making high speed logical inferences from a database of facts. He described how the project had originally aproached INMOS to see if they could use our CAD tools for special purpose VLSI design but had been persuaded by Iann Barron to use transputers instead. They are now in detailed discussion with ICL about the design and construction of a prototype.

Sue Peeling of RSRE discussed the coding of some simple sorting algorithms in occam and their consequent execution time on a transputer. David Broomhead of RSRE considered some aspects of special pupose solutions to particular logical inference problems. (It was intriguing to speculate what his example concerning recognition of animals at the zoo hid in the way of a real Ministry of Defence application).

The meeting was brought to a close by David May of INMOS who ran through the variety of ways potential customers are thinking of using arrays of transputers. If his predictions are anywhere near true there is little doubt that INMOS will be able to sell as many as they can make (as long as the price is right). It emerged that many potential users really want to be able to compute with real (floating point) numbers, and so David indicated that he was working hard on design in this area.

David also mentioned his latest work extending the occam language, and the availability of the occam programming system and the occam portakit. The meeting closed with an impromptu demonstration of a prototype transputer processor (the S43) running an occam program displaying aliens jumping around a visual display screen] This was the first time such a demonstration has been given in public.

The general reaction from people who attended the meeting was favourable and the general feeling was that future meetings should be longer and have adequate bar facilities for extended exchange of ideas. Only one thing went wrong - British Rail managed to lose the line between Paddington and Reading that morning and some people therefore arrived rather late.

## <u>THE NEXT USER GROUP TECHNICAL MEETING</u>

29th March 1985

The second one-day technical meeting of the OCCAM User Group will be held at Oxford University on 29th March 1985.

There will be speakers from Oxford University Programming Research Group, from INMOS and from other OCCAM users.

Further details of the meeting and an invitation to attend will be sent to all members in February.

## UPDATE ON THE OCCAM PROGRAMMING SYSTEM

For a fuller description of these products see the Summer 1984 newsletter.

OPS VAX/VMS                    available now

First release implements an integer subset of occam, and omits file folding and separate compilation. These will be included in the second release, as will 68000 and 8086 code generation.

Subsequent releases - available free to previous purchasers - will implement full occam, which will provide multilength arithmetic, structures and IEEE real arithmetic. In addition, the full folding mechanisms will be supplied.

OPS VAX/Unix                    available Q1 1985

Similar facilities to OPS VAX/VMS

INMOS Workstation                    available Q1 1985

A 68000-based workstation with Winchester, floppy disk, megabyte of memory and a terminal together with the OPS software.

Related Products:

OCCAM Evaluation Kit                    available now

A low cost introduction to occam, available for a number of small computers.

OCCAM Portable Compiler kit                    available now

See next item.

OPS/IBM PC                    available Q2 1985

Implementation of the OPS on an IBM PC or IBM PC XT, under MS-DOS.

Transputer Development System                    Q3 1985

Software tools for IMS T424 support. Available as an upgrade to OPS customers, when the cost of the OPS is deducted from the TDS price

Full details from:

Philip Mattos
INMOS Limited
Whitefriars, Lewins Mead
BRISTOL BS1 2NP

## THE OCCAM PORTAKIT

If you have been waiting patiently for an opportunity to try out occam on your local computer, for which no compiler has yet been available, you may be interested in a new software product from INMOS, the OCCAM portakit. This kit consists of a magnetic tape and accompanying manuals. All files on the tape are ASCII text files and include a compiler from occam into an interpretable intermediate code. All you have to do to be able to compile and run occam programs is to write an interpreter for this intermediate code which will run on your operating system.

To make this job as easy as possible the kit includes example interpreters written in widely available languages such as Pascal, Fortran and BCPL as well as a formal definition written in occam itself. Having written such an interpreter and interpreted the suite of pre-compiled test programs provided, you are in a position to interpret the compiler itself in order to compile your own occam programs. The output from these runs will be the interpretable versions of your programs which you can then interpret with your interpreter.

The compiler was itself written in occam and compiled through itself. The source is included in the kit so that if you are really adventurous you may consider modifying its code-generation process to generate code for your target architecture.

The kit is designed in the first instance for 32-bit machines with uniformly addressable address space of about half a kilobyte. (If you want to try it out on other machines then it will be at your own risk). Full details are available from all INMOS sales offices.

MDP

## OCCAM COMPILER AVAILABLE

An occam compiler is available, which runs on Prime computers an generates Prime machine code. The compiler implements the full language and there are a few extensions, notably for separate compilation and recursive processes. For further information, contact Dr. A.J.Fisher, Department of Computer Studies, The University, Hull HU7 7RX.

## PORTAKIT INTERPRETER WRITTEN IN C

A portakit interpreter written in C is available from Michael Harrison, Dept of Computer Science, University of York, to whom all enquiries should be made.

## FFT PROGRAM OFFERED

Owen Ransen of INMOS has written a complete 8-sample FFT in occam. This has has been run and tested on the VAX OPS. So if you want to know how a real FFT can be done, write to Owen for a copy of his listing.

## A DISTRIBUTED MAXIMA ALGORITHM

Richard Taylor, INMOS

There exists a network of transputers where each transputer contains a number. After the procedure 'maxima' has been executed in each transputer, they will all contain the same number which is the largest number in any transputer in the network.

```
PROC maxima (CHAN L1o, L1i,              --link 1
                  L2o, L2i,              --link 2
                  L3o, L3i,              --link 3
                  L4o, L4i,              --link 4
             VALUE maxpath,              --maximum path length
             VAR max)  =                   --local  input  and
result

SEQ i = [0 FOR maxpath]
  VAR max1, max2, max3, max4 :
  SEQ
    PAR
      L1o ! max
      L2o ! max
      L3o ! max
      L4o ! max
      L1i ? max1
      L2i ? max2
      L3i ? max3
      L4i ? max4
   maximum (max, max1, max2, max3, max4) :
```

The procedure 'maxima' takes as parameters the channels which are the four links of a transputer (which it is assumed are all connected to other transputers), 'maxpath' which is the longest path, counted in a number of links, between any two transputers in the network and 'max' which is the local number. The procedure 'maximum' makes its first parameter the maximum of its five parameters.

The procedure outputs 'max' on all its links and inputs the maxima of all these numbers. After this has been done 'maxpath' times, 'max' in each transputer will be the same.

## A BIDIRECTIONAL PROTOCOL FOR OCCAM

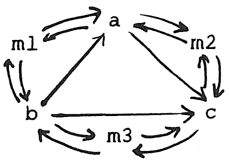
Richard Bornat, Dept of Computer Science and Statistics
Queen Mary College

OCCAM is designed for efficient implementation on a simple machine. Its communication mechanisms are particularly restricted to fit this straitjacket. By requiring one-to-one channel pairing, message passing becomes almost as easy to implement as assignment. By requiring guarded commands (ALTs) to have receive guards only, non-deterministic input becomes hardly more difficult.

It is well known that bidirectional guarded commands allow some added programming flexibility. A buffer in a pipeline of processes is most easily programmed with a bidirectional guarded command, offering simultaneously to receive a new value from a producer or send a stored value to a consumer. With unidirectional guarded commands the consumer and buffer must use some sort of signal-ready protocol, which militates against the use of channels as abstract interfaces to processes. The buffer is only one example: rings of processes are another, server processes which attempt to handle multiple streams of requests with minimal latency are a third. Although there are often 'ways round' the problem it would be nice if it didn't exist.

It is possible to construct an occam-like system which allows bidirectional guarded commands, and it is possible to implement it in occam itself provided that in the occam-like system also there is only one-to-one 'channel' connection. Processes in the system which execute a guarded command must also go through a protocol of message queries and replies to achieve synchronisation with a partner. The maximum number of occam messages per guard of the bidirectional command is 6, so the cost is significant but perhaps affordable, if the application demands it.

In the occam-like system each 'channel' is represented by five occam channels, one connecting two processes and the other four connecting them to a 'matcher' process which supervises the protocol:



The protocol itself is a development of one proposed by Buckley and Silberschatz for a bidirectional implementation of CSP. Processes ready to communicate identify themselves to a mátcher with OFR messages and withdraw with CNCL.

Processes are numbered arbitrarily but uniquely: this numbering is used by matchers to control communication and to avoid deadlock and livelock cycles. A matcher which has two ready messages sends the higher-numbered one a RDY message: if it replies CMT it is committed to the pairing and the matcher sends RQ to the partner. If the partner replies YES the matcher has made a pairing and tells the original process so with MCH.

The description of the protocol is a couple of state diagrams, one for processes and one for matchers. In this description processes-numbers and channels are treated the same. In the occam implementation they are of course distinct.

A process executing a guarded command maintains two data-structures which describe the state of execution of a guarded command. $M_i$ addresses the matcher named in enabled guard i, $g_i$ the state of communication with that matcher:

| | | | |
|---|---|---|---|
| $g_i$ | = | blank | normally |
| | | offered | when OFR(p) sent |
| | | ready | when RDY($m_i$) received |
| | | #x | when RQ($m_i$,x) received |

It is in one of four states:

| | |
|---|---|
| E | executing, outside any guarded command |
| G | in a guarded command but not committed to any offer |
| C | committed to a particular RDY message |
| P | paired with a particular offer by YES or MCH |

Because of the use of synchronised message-passing and CNCL messages a process can be sent no messages in E state. The state-table below refers to four predicates on the contents of array g:

b: Ei: g =blank
y: Ei: g = x
r: Ei: g =ready
o: Ei: g ˜=blank

| state | message received | change in state | change in data | message sent |
|---|---|---|---|---|
| G↑y | - | P | $g_i$ ,q:=blank,x | m !YES |
| G↑b | - | G | $g_i$ :=offered | m !OFR(p) |
| G↑r | - | Č | c:=blank | m !CMT |
| P↑o | - | P | $g_i$ :=blank | m !CNCL(p) |
| P↑˜o | - | E | - | - |
| GvCvP | RQ(m ,x) | - | $g_i$ :=#x | - |

| | | | | |
|---|---|---|---|---|
| GvCvP | RDY($m_i$) | - | $g_i$:=ready | - |
| C | NO | G | $g_c$:=offered | - |
| C | MCH(x) | P | $g_c$,q:=blank,x | - |

The effect of the algorithm, if it terminates, is to assign to 'q' the address of the sucessful process partner. This value can then be used to send or receive data as appropriate.

A matcher keeps record of offers from two processes $P_1$ and $P_2$. It counts in a variable j the number of offers received and not yet withdrawn. It has three states:

| | |
|---|---|
| I | idle, waiting for OFR messages |
| W0 | waiting for reply to RDY |
| W1 | waiting for reply to RQ |

The state table requires a single predicate:

o: $P_1 > P_2$

| state | message received | change in state | change in data | message sent |
|---|---|---|---|---|
| I↑j=2↑o | - | W0 | - | P !RDY(m) |
| I↑j=2↑~o | - | I | $P_1$,$P_2$:=$P_2$,$P_1$ | - |
| IvW0vW1 | OFR(x) | - | j, $P_{j+1}$:=j+1,x | - |
| W0↑j=2↑o | CMT | W1 | - | $P_2$!RQ(m,$P_1$) |
| W0↑(j=1v~o) | CMT | I | - | P !NO |
| IvW0 | CNCL($P_1$) | I | i,$P_1$:=i-1,$P_2$ | - |
| IvW0 | CNCL($P_2$) | - | j:=j-1 | - |
| W1 | CNCL($P_2$) | I | j:=j-1 | $P_1$ !NO |
| W1 | YES | I | j:=0 | $P_1$ !MCH($P_2$) |

The protocol can be proved to be deadlock- and livelock-free, and requires an average of 4.5 messages per guard. It can be impemented in about 100 lines of occam, similarly deadlock- and livelock-free.

## EXPERIENCES USING CSP THROUGH TO OCCAM (OEK)

Don Fay, Queen's University, Belfast

(This is a summary of the talk given at the September 1984 Technical Meeting)

1. PDP-8 to ICL 1900 link for CSP. Problems centred around physical and logical allocation of channels using the link. Some particular problems caused by having to use the ICL standard interface and its inherent DMA. No significant example programmed on this system.

2. Digital differential analyser in OCCAM. This technique can be used to generate functions as solutions of differential equations. It is particularly suitable with fixed point arithmetic where incremental changes of functions are required.

OCCAM processes are used to implement an array of integrators together with a "plugboard" process. The plugboard represents the interconnections of the integrators. A further process is needed to fan out a control signal. At the time of the talk there was a problem getting all processes to synchronise their termination.

3. Image display hardware scheme

A brief summary was given of the paper which is published in Microprocessors and Microsystems, Vol. 8, No. 1, Jan/Feb 1984.

4. FFT

An O(1), wavefront array implementation of the FFT was described in outline. (Log2n) X 2 levels of hardware would be required. There is potential for considerable parallelism within each level, although there is no need for this to be implemented in separate hardware. The OCCAM channels are used as pipes, with all data transfers in a forward direction - no control or data is sent in a reverse deirection.

2 5. Using OCCAM to describe an existing bounded buffer imlementation

Using OCCAM to describe an existing algorithm led to clumsy OCCAM code. The moral seems to be to redesign from scratch using OCCAM concepts. Particular problems occur when the algorithm described includes interrupts.

------

For further details, please write to Mr Fay. (address in member list)

# FIVE ESSAYS ON OCCAM

P.H.Welch, GEC Avionics, Rochester

Occam is a language for designing and implementing concurrent systems. It is characterised by the simplicity and power of its concepts together with an efficient execution model. Nevertheless, there are areas of the language where some enhancements, balanced by some further simplifications, may be worthwhile. These essays discuss several possible amendments aimed at increasing the clarity, security and flexibility of the language without damaging its performance. They also seek to encourage a functional style of systems' design as opposed to the procedural mechanism derived from traditional sequential programming languages and computer architectures.

## 1. Channel parameters

Specify which channel parameters of a PROC are for input and which are output. This is vital piece of the semantics of a PROC which needs needs to be known simply in order to produce syntactically legal code. Currently, this information is available from comments (unreliable), identifier names (also unreliable) or by examining the PROC body (ok, if you are a compiler, but breaking an important principle of information hiding).

For example :-

```
PROC nos --> (CHAN out) =
  -- out! 0;  1;  2;  3;  . . .
  ...

PROC pair.sum (CHAN in) --> (CHAN out) =
  -- in? x0;  x1;  x2;  x3;  . . .
  -- out! (x0 + x1);  (x1 + x2);  (x2 + x3);  . . .
  ...
```

The additional information contained in the specification enables the compiler to produce more useful error diagnostics. Consider the following PROC, where we have neutral channel names and no comments :-

```
PROC integrate (CHAN sally) --> (CHAN fred) =
  ...
```

An attempt to 'sally!' or 'fred?' in its body can be rejected straight away. With current notation, such an inconsistency would not be found until 'integrate' was applied and, then, its cause would not be pinpointed.

A similar syntax is used to instantiate these PROCs :-

```
PROC squares --> (CHAN sue) =
  CHAN a, b:
  PAR
    nos --> a
    integrate (a) --> b
    pair.sum (b) --> sue:
```

It is now transparent, just from the specification lines of 'nos', 'integrate' and 'pair.sum', that the network is legally connected. Currently, illegal networks are only detectable by having access to the bodies of their component processes. This runs counter to the principles of information hiding (which are likely to be particularly relevant to the construction of large parallel systems).

## 2.Functional system building

Conventional notation for system building in Occam has a procedural style. Connections have to be made through explicitly declared channels and the programmer finds himself needing to draw pictures to give himself a chance of getting them correct. With the separation of input from output channel parameters (Modification 1), a freer functional style of system building can be adopted. Some lower-level details (like explicit channels and PAR directives) can be omitted. Program semantics, transformation and derivation becomes easier since we are able to concentrate more on the code (algebra) and less on the pictures. For example :-

```
PROC squares --> (CHAN sue) =
  pair.sum (integrate (nos)) --> sue:
```

The compiler has to work harder but there is no run-time penalty. The semantic clarity of the functionally expressed networks is such that we might be able to stop drawing so many networks to aid our design. Indeed, network diagrams might be rendered as obsolete a technique as sequential flow-diagrams and for the same reason - we have a better means of expression.

## 3. Input channel buffers

When writing at a low-level (i.e. "?", "!", ":="), many variables need to be declared just to catch input data before processing. Often, the original input values do not need to be retained beyond this processing.

To save the programmer the need to declare such local variables, associate with each channel a "buffer" variable of the same name. To indicate the structure of the algorithm, let an input process always "guard" another (indented) process.

For example :-

```
PROC succ (CHAN in) --> (CHAN out) =
  WHILE TRUE
    in?
      out! in + 1:
```

The concept of input buffers and input operations guarding another process frees the programmer from controlling explicit variables. Updatable variables are a concept derived from the von-Neuman sequential machine and their existence constitutes a temptation to the programmer to continue with previous modes of thought. Programming without variables or assignment leads to clearer algorithms that are free from side-effects. In the case of low-level Occam cycles, these concepts lead to a more expressive language and a more efficient implementation. The language may be simplified by the removal of variables, ANY and the assignment process.

4. Synchronisation control

Allow "negative" channels with an opposite hand-shake protocol to permit output guards. Allow "free" channels with no synchronisation. Such channels complement conventional ("positive") channels, have simple and direct implementations and model common real-world objects (e.g. a particular free channel, TIME, already exists in the language).

We show that free channels may be simulated using negative channels and negative channels by positive channels. However, for reasons of clarity and performance, it might be an idea to support these ideas directly within the language.

5. Types

Introduce strong typing, packages, polymorphism and higher order processes. For a language which claims to be a language of "information", Occam makes us express that information at a very low level - namely words and word vectors. This is quite forgiveable in the short (and medium?) term since Occam tackles a problem of computing (concurrency and non-determinism) that is much less well understood than types. Eventually, though, for designing and implementing very large systems, we are likely to need the extra security, ease of writing and clarity provided by some form of (strong) typing.

Polymorphic and higher order types are possible, without spoiling the simple and fast execution model of Occam. The compiler will have to work harder but it might be worthwhile. The language is looking quite functional.

## SUMMARY

The proposed amendments have been motivated by the desire to increase the confidence of the system designer (customer, tester, maintainer). The method has been to increase our powers of expression for the specification ((1), (4) and (5)) and implementation ((2), (3), (4) and (5)) of objects. A major benefit is that the specification, by capturing more information, becomes more independent (and, thus, separable) from the implementation. This is very important to the human trying to design (or understand the design of) a system containing such objects. To use an object, he need only understand what the object does, not how it does it. Having the specification in a formal notation (as opposed to natural language documentation) decreases ambiguity and increases the support the compiler can give for checking consistency within the design.

A final thought: the choice of SEQ/PAR, VALUE/CHAN or PACKAGE/PROC are different reflections of the same issue. Is this really just an implementation detail? Is it possible to conceive of a further simplification to the language that frees the programmer from this decision, leaving this for separate target configuration instructions given to the compiler and from which the program itself is independent?

(The full paper is available from the author, whose address appears in the member list)

## OCCAM ACTIVITIES AT LOUGHBOROUGH

R.P.Stallard, Dept of Computer Studies
University of Technology, Loughborough

Work on multiprocessor systems at Loughborough University in the Department of Computer Studies began in 1978 with a dual processor Interdata system. Since then a four processor Texas Instruments 990/10 shared memory system has been installed. Much of the multiprocessor research at Loughborough has centred on the efficient implementation of numerical algorithms on different types of parallel computer architecture. The only programming languages used have been extended versions of FORTRAN permitting parallel execution and synchronization between processors. These languages suffer from all the pitfalls of botching a sequential language to run in parallel.

For a variety of reasons I decided to write an occam compiler for the Department's VAX/UNIX system in the language 'C'. Initially a system to execute on the VAX was written but execution on the actual parallel system is envisaged.

Objectives of the compiler design were principally for flexibility and for speed of object execution. Flexibility was achieved by using the UNIX compiler and lexical analyser utilities (yacc and lex) and by favouring straightforward coding without space/time optimizations. A dynamic stack based mechanism is used for space allocation when calling procedures (as opposed to using textual subsitituton). Process descriptors are also allocated on the stack, this method reduces run time space requirement and permits relatively easy language extension for dynamic space allocation features, but has the extra cost (on a single processor) of managing a single shared stack between multiple allocator and deallocator processes. The internal occam pocess scheduler is kept as simple as possible to minimize context switch time. The basic scheduling policy is for active runnable processes to explicitly wake up suspended processes rather than scanning dormant processes repeatedly.

Another aim was to provide occam as just another language under UNIX. All standard UNIX utilities are able to manipulate occam source text so that the standard text editors (vi, ed) can be used. Occam programs can use redirected input/output and pipelines just like other UNIX programs.

Although compiler validation is never entirely possible the current version does not suffer from any known bugs and has passed the code generation tests kindly supplied by INMOS (these tests did uncover some rather obscure bugs that were totally unrelated to the purpose of the tests). A research student has started to use the occam compiler for programs that model the execution of systolic arrays.

The compiler has taken about five man-months of effort to write, supporting the view that occam is a comparatively simple language to implement. The only language definition document was the INMOS occam booklet.

There are, however, some local language restrictions. The implemented. These are slices, prioritized ALT and the whole of the configuration section. Slices will be implemented although there are difficulties in ensuring correct usage as slice sizes are permitted to be variable.

Several local language extensions have been added (although normal INMOS proto-occam can still be used). These are currently:-

Variable PAR Replicator Counts

The space allocation scheme permits dynamic creation of processes without any additional repercussions, more than 10,000 processes have been created and run in a single program.

Recursive Procedure Calls.

The stack procedure call mechanism enables recursive calls to be treated just like ordinary calls to procedures.

Multiple Occam Source File Compilation and Provision of Libraries.

A program can be split into several separate source files and separately compiled. Libraries of commonly used procedures can be referenced explicitly rather than building special procedure names into the compilation stage.

It is also possible to include subroutine libraries written in 'C' (and via 'C' other languages). This has been done by declaring procedures to be 'external' (defined in another file) or and accessible to other files).

Normal 'PROC' declarations are local to the source file. Full cross file parameter type checking is enforced.

The object program generated by the compiler is in fact standard 'C' language source and the UNIX 'C' compiler is used to generate object and link in the run time support routines (also written in 'C'). The standard 'curses' package is used for cursor movement commands (provided by the standard occam library) so that run time execution should be terminal type independent. Implementation on other UNIX 4.1/4.2 systems should be straightforward, but different word lengths and program size limits could cause problems. The current slowest stage in occam compilation is the 'C' compiler stage because a lot of 'C' is produced for each occam source line. A future facility could provide direct compilation from occam to VAX assembler.

The run time support facilities enable 'CTRL C' to be used at any time to interupt program execution and inspect the status of the processes. Facilities currently include setting spy and break points in the object (given the required source line number) to trace program execution, controlling process, stack and clock tracing, the ability to abort or restart program executon and checking that subscripts are valid on all channels and variable vectors.

Future work at Loughborough on occam will hopefully include formal analysis programs to explore livenesses, data dependencies and potential program transformation of occam programs.

## THE DIAGRAMMATIC REPRESENTATION OF OCCAM PROGRAMS

Alistair Munro, Computer Centre, University of Bristol

### Introduction

SADIST is a means for representing the structure of a concurrent system. It recognises two components: processes, which may be nested hierarchically, and channels, that form an irregular interconnection between them. It presents the system to the user as a set of diagrams in which processes are represented by boxes and channels by lines between them.

The motivation is to make the interconnection structure visible, and to show the designer where cycles exist. Cycles of communication between processes are the source of many problems in concurrent systems, and faulty design can lead to deadly embrace deadlocks. They may intersect, or be nested within one another, and they are difficult to detect in a source program.

The sections of the paper describe the details of the SADIST representation of the programming language occam, and the implementation of the method on the PERQ single user system.

## SADIST

A system is presented diagrammatically as a hierarchical structure of processes linked by communications channels. A diagram is composed of boxes that represent processes, and lines that represent channels; other information about the system is included as text, labelling the boxes and lines, (Fig 1). The frame labelled "current level" contains processes with communications channels. Some of these terminate outside the frame, and represent channels entering from the level above.

A hierarchical structure of diagrams is called a model. The current level of Fig 1*contains "nested models", representing the sub-trees. The processes in the current level are the roots of the nested models, and access to them occurs through partly-concealed frames surrounding the current one. The current level is the root node of a process in the level above, which is represented by a similar, partly-concealed frame.

The hierarchy can be understood by relating the model structure to that of an occam program. The model contains two networks. First, it is a tree of nodes representing the processes of the program. A second irregular network of channels joins the terminal nodes.

## Process Tree

The tree of processes consists of terminal nodes and non-terminal nodes. A diagram may contain a mixture of both.

### Terminal nodes

Most of the terminal nodes are the primitive processes of the program.

* Fig 1 is on page 30

Where a process identfier is encountered it is treated as a terminal node too, although it is really the attachment of a subtree if considered in terms of the passing of control. Declarations have to be treated in two ways: VAR and CHAN statements are terminal nodes.

### Non-terminal nodes

Most non-terminal nodes are constructs. SEQ and PAR nodes contain nested models that may be further constructs or primitives. Conditionals and guarded processes are considered to be non-terminal nodes within IF and ALT nodes respectively. A PROC declaration is the root of a separate sub-tree and is therefore non-terminal node too.

## Channel Network

Channels have a source and a destination. Most of them will be contained within the model. There are a few external channels that are connected to the world outside the model. These may include those that have no source or destination because of programming errors.

Lines are attached to specific edges of a box depending on their function. Inputs enter a box from the left; outputs emerge from the right. Lines representing inputs in guards enter the top edge of a box, and are referred to as "control" channels.

The ends of channels are input and output primitives, that is, terminal nodes of the tree, if they are contained within the model. Each pair of output/input nodes will share a non-terminal node on the path back to the root of the model. The line is extended back up through each of the node diagrams until the shared node is reached and the final join is made at this point. External and partially connected channels are brought up through the tree to the nearest root node. This may be the root of a PROC declaration or of the system.

Channels may be shared, by an error of programming, and this is shown by forks or joins in the affected lines.

## Representation of Replicators

SEQ, PAR, IF and ALT constructs may be replicated if there is some regularity in the component processes and their interconnection. This is represented by creating two boxes within the root node of the replicator model. One of these is the first process; the second is the last in the chain. Channels that interconnect them are drawn once only.

## Naming Scheme

The text labeling is derived from the occam source. The naming of processes is determined by their position in the tree. Channel naming is based on the relation positions of lines on a specific box edge, numbering them clockwise round the box.

The processes in a diagram are not ordered in any way. Their positioning may be chosen to suit their interconnection structure and minimise the confusion of crossing lines, but has no significance as far as the order of execution of processes is concerned. This should not be important in a concurrent system in any case.

## SADIST Implementation

The SADIST programs are:

a compiler that translates occam source into the internal network representation.

a generator that translates the network database into an occam program.

a generator that translates the network database into an intermediate graphics language for subsequent processing to produce hard-copy.

an editor that manipulates the internal database by modifying the positions of objects on the diagrams interactively.

## Requirements of the Target System

Briefly, the target system must support:

virtual memory, since the code sections are large, and the data sections must be unlimited in size.

interactive graphics, with the ability to service events from pointer devices and character keyboards, (GKS level 2c equivalent), and display segmentation with segment dragging or low-level access to the display hardware to allow double-buffering to be implemented.

tools for program development, specifically a compiler that support multiple program modules with associated linker and run-time libraries.

The hardware that the system will run on should be adequately configured so that it does not impose any inherent limitations on the performance of the program.

A PERQ became available which fulfilled the basic needs, as follows:

POS operating system, supporting a segmented memory management system.

Access to low level features of the hardware: line and raster-op primitives, pointer and character input.

Pascal compiler

The machine has 1 MByte of physical memory, a character keyboard, and a digitising tablet.

The PERQ has proved adequate to support the project so far. It is not ideal from the programmer's point of view: in terms of high-level graphics packages or transparent virtual memory for instance it is a "naked machine" and the user must be able to implement many of the low-level facilities himself. One of the benefits of the PERQ, under POS, was that most of the original program modules could be used without substantial change; modification of the compiler directives was all that was needed in most cases.

Continuing Development

Initial impressions of the use of the system suggest that the diagrams have limited value in isolation. Many are trivial, and only a few contain useful information about the communications structure. Output that gives an overview, or is more selective about what is displayed is an obvious enhancement.

occam is the language for the Transputer and one application of SADIST is to redesign and implement itself on a a suitably configured Transputer system.

## <u>OCCAM PROGRAM LIBRARY</u>

Contributions for the above are invited of any programs which may be of interest to other users. A brief (one sheet) description of what the program does, which system it runs on plus name and address of author, should be sent to the address below. Program listings are not required since the details will be published in the newsletter and those interested can contact the author directly. However, the librarian has access to Vax, Sage and Sirius systems so is able to transfer programs between these systems if necessary.

Mrs S M Peeling
RSRE
St Andrews Road
Malvern
Worcestershire
WR14 3PS

Tel: Malvern (06845) 2733 Extn 2228

## BIBLIOGRAPHY UPDATE

compiled by INMOS and the Editor

Papers about occam and the Transputer by INMOS

Richard Taylor, "Signal Processing with occam and the transputer". (Available from INMOS)

Pete Wilson, "The transputer - a general purpose multiprocessing component". (Available from INMOS)

Pete Wilson, "Digital signal processing with the IMS T424 transputer". (Available fron INMOS)

Paul Walker, "Using transputers for optimising cost/performance". (Examples of how transputers could be used for the FFT) (Available from INMOS)

Paul Walker, "Estimating transputer performance". (Available from INMOS)

P.Wilson, "Design station captures complete system design/implementation cycle,"

Midcon '83 Conference Record (Chicago, Sept 13-15, 1983) Los Angeles: Electronic Conventions Inc., 1983, pp 1.5/1-6.

P.Wilson, "Thirty-two bit micro supports multiprocessing,"

Computer Design , Vol. 23, No. 6, June 1984, pp 143-150.

Philip Mattos, "The Transputer",

New Electronics, Aug 14, 1984, pp 43-45.

S.Brain, "Writing parallel programs in occam,"

Electronic Product Design , Vol. 5, No. 9, Sept 1984, pp

47-50,53,54.

Pete Wilson, "Programming for Multiprocessor Designs",

Electronic Imaging, October 1984, pp 59-62.

R.Taylor, "Graphics with the transputer,"

Computer Graphics 84 Conference , Oct 9, 1984.

IMS T424 Transputer Reference Manual , Nov 1984.

Papers on occam and the Transputer authored outside INMOS

C.Gross, "Programming of parallel processors in occam,"

Electronique Industrielle , No. 49, March 15, 1983, pp 57-60. (In French)

Dick Pountain, "OCCAM Occult - programming in parallel with occam",

Personal Computer World , July 1983, pp 136-141.

P.Palerma, "The transputer: the European revolution in micro architecture,"

Elettronica Oggi (Italy), No. 1, Jan 1984, pp 65,66. (In Italian)

D.Q.M.Fay,"Comparison of CSP and the programming language occam",

Australian computer science communications ,Vol 6, No. 1, Feb 1984, pp 13-1 to 13-10.

T.Durham, "Inmos: a final frontier?"

Systems (S.Africa), Vol. 14, No. 4, April 1984, pp 22-24.

D.Q.M.Fay, "OCCAM manual gives programming guidance to users at various levels"

Microprocessors and Microsystems , Vol 8, No. 5, June 1984.

"'Transputer' a component for fifth generation computers,"

Elektronika (Netherlands), Vol. 32, No. 12, June 29, 1984, pp 13,15,17,19. (In Dutch)

B Jane Curry, "OCCAM solves classical operating system problems", Microprocessors and Microsystems , Vol 8, No. 6, July/August 1984, pp280-283.

Dick Pountain, "The transputer and its special language, oc-cam,"

Byte , Vol. 9, No. 8, Aug 1984, pp 361,362,364,366.

Anthony Skjullum, "OCCAM: A parallel processing language from the U.K."

Computer Language , November 1984, pp 55-60.

Rory Johnston, "Inmos paves the way for leap,"

Computer Weekly , Dec 6, 1984, p 24.

John McCrone, "Inmos tackles the marketing of Transputer,"

Computing , Dec 6, 1984, p 24.

John McCrone, "Imperial College plan may get a commercial deal,"

Computing , Dec 13, 1984, pp 26,27.

Geraint Jones and Bill Roscoe, "Find the median of nine numbers - quickly". Unpublished.

J.G.Harp, J.B.G.Roberts and J.S.Ward, "Signal Processing with transputer (traps)",

Computer Physics Communications (in press).

D.S.Broomhead, J.G.Harp, J.G.M.McWhirter, K.J.Palmer, J.G.B.Roberts, "A practical comparison of the systolic and wavefront array processing architectures",
2 Proc. of the IEEE International Conf. on Acoustics, Speech and Signal Processing (Tampa, FL, March 1985)

Useful background material

E.Shapiro, "A concurrent Prolog Bibliography".

E.Shapiro, "Systolic Programming: A paradigm of Parallel Processing"

## BACK NUMBERS

Copies of Issue No. 1 of the Occam User Group Newsletter are available while stocks last on application to the secretary at INMOS. This issue included 6 pages of bibliography on CSP, OCCAM and the Transputer.

## OCCAM USER GROUP - List of members as at 12th December 1984

John Ainscough, Dept of Engineering and Management Systems, Brunel University, Kingston Lane, Uxbridge, Middlesex.

S Akiyama, Electro-technical Laboratory, Umezono Sakura-Mura Tukuba, Ibaragi-Ken, Japan.

J Atkinson, CAP Scientific Ltd, 20-26 Lambs Conduit Street, London.

Dr L.V.Atkinson, Department of Computer Science, University of Sheffield, SHEFFIELD S10 2TN.

Dr Jean Bacon, The Hatfield Polytechnic, Computer Science Div., PO Box 109, College Lane, Hatfield HERTS AL10 9AB.

Roland Backhouse,Dept of Computer Science, University of Essex, Wivenhoe Park, Colchester,Essex CO4 3SQ.

Mike Barton, E & EE Department, University of Bristol, University Walk, Bristol BS8 1TR.

Mark Beech, E & EE Department,University of Bristol, University Walk, Bristol BS8 1TR.

Mitch Beedie, Electronic Design, Hayden Publishing Co. Inc, Avalon House, Cranston Road, East Grinstead, West Sussex.

Dave Berry, Computer Science Dept, JCMB Kings Buildings, West Mains Road, Edinburgh.

David Bevan, GEC HRC CSRL, East Lane, Wembley, Middx HA9 7PP.

Dr P Blackledge, GEC Telecommunications Ltd, Telephone Works, PO Box 53, Coventry CV3 1HJ.

Ulrich Bollinger,Bollinger Datentechnik GmbH,An der Fuhr 15,5030 Hurth 6, West Germany.

Martin Bolton, E & EE Department, University of Bristol, University Walk, Bristol.

Richard Bornat, Queen Mary College, Mile End Road, LONDON.

Steve Bowran, CAP Scientific Ltd, 20-26 Lambs Conduit Street, LONDON WC1N 3LF.

J.Brankin Esq, British Telecom, Royston House, 34 Upper Queen St, Belfast.

Dr Graham Brooks, Dept of Computer Science, University of Sheffield, Hicks Building, Sheffield.

David Broomhead, RSRE, St Andrews Road, Great Malvern, Worcs WR14 3PS.

Geoffrey Burn, GEC Hirst Research Centre, East Lane, Wembley, Middx HA9 7PP.

M Burton Esq, Texas Instruments Ltd, MIS45, Manton Lane, Bedford.

Khan Busby, Thorn EMI, Central Research Labs, Trevor Road, Hayes, Middlesex UB3 1HH.

B Cantwell, Dept R14.1.1, British Telecom Research, Laboratories, Martlesham Heath, Ipswich IP5 7RE.

Professor C R Cavonius, Inst.F Arbeitsphysiol, Ardeystr 67, D-4600 Dortmund 1, West Germany.

C.F.Chan, Dept of Elec.Eng.Science, University of Essex, Colchester, Essex.

Mr A Cheese, 159 Windermere Drive, Warndon, Worcester WR4 9JF.

Alan Clark, Electrical Engineering Dept, Leicester Polytechnic, PO Box 143, Leicester LE1 9BH.

D.A.Clarke, IBM (UK) Laboratories Ltd, Hursley Park, Winchester, Hants SO21 2JN.

D.M.Cleal Esq, Central Computer &, Telecommunications Agency, 157-161 Millbank, London.

Adrian Cockcroft, Cambridge Consultants Ltd, Science Park, Milton Road, Cambridge CB4 4DW.

Andrew J Cole, Computer Based Learning Unit, Leeds University, Leeds, West Yorkshire LS2 9JT.

Murray Cole, Dept of Computer Science, University of Edinburgh, JCMB Kings Buildings, Mayfield Road, Edinburgh EH9 3SZ.

Conrad Cork, Leicester Polytechnic, PO Box 143, Leicester LE1 9BH.

P Cornwell Esq, Texas Instruments Ltd, MIS45, Manton Lane, Bedford.

Michael Coyle, British Telecom, Royston House, 34 Upper Queen St, Belfast BT16FD.

M.K.Crowe, Paisley College of Technology, High Street, Paisley, Renfrewshire PA1 2BE.

B Jane Curry, Computer Centre, Chelsea College, University of London, Pulton Place, London SW6 5PR.

Mr C Dilloway, Highcroft, Gunhouse Lane, Stroud, Glos GL5 2EB.

Gunter Dotzel, Modulaware, E-Rommelstr 59, D-8520 Emangen, WEST GERMANY.

P.H.Duffin Esq, Central Computer &, Telecommunications Agency, 157-161 Millbank, London.

Peter Eckelmann, INMOS GmbH, Danziger Sr.2, 8057 Eching, WEST GERMANY.

M Eglin, Systems Dept, British Aerospace, Brough, N.Humberside.

Keith M Farrington, CAP (Reading) Ltd, Trafalgar House, Richfield Avenue, Reading, Berks RG1 8QA.

D.Q.M.Fay, Queens University (Belfast), Belfast, Northern Ireland BT7 1NN.

L.Fernando, UMIST, Wright Robinson Hall, Altrincham Street, Manchester.

Paul Fertig, Programming Research Group, Oxford University, 8-11 Keble Road, OXFORD.

Ian Firth, Dept of Computer Studies, Loughborough University, Loughborough, Leics LE11 3TU.

Dr A.J.Fisher, University of Hull, Dept of Computer Studies, Hull HU6 7RX.

David Freestone Esq, British Telecom, Research Laboratories, R14.2.4, Martlesham Heath, Ipswich IP5 7RE.

Dr M.S.Gate, BAC Dynamics Group, Manor Road, Hatfield, Herts AL10 9LL.

B.W.J.Gooding, BG Software Design Ltd, Holly Cottage, Ashford Hill, Newbury, Berks RG15 8BQ.

Jan Gottschick, Gottschick Computer, Kreuznacher Str. 40, 01000 Berlin 33, GERMANY.

Dr David T Gray, Queens University, Dept of Computer Science, Belfast, BT7 1NN.

M Griffiths, MWG Systems, 106B Lexham Gardens, London W8 6JQ.

Patrick A.V.Hall, Cirrus Computers Ltd, Waterman House, 101-107 Chertsey Road, Woking, Surrey, GU21 5BL.

Gordon Harp, RSRE, St Andrews Road, Great Malvern, Worcs WR14 3PS.

Dr Neville R.Harris, Distributed Systems Group, Department of Computer Science, Engineering School, Trinity College, DUBLIN.

Matthew Hennessy, University of Edinburgh, Kings Building, Mayfield Road, Edinburgh.

Professor C.A.R.Hoare, Oxford University, Computing Laboratory, 8-11 Keble Road, Oxford OX1 3QD.

Ian Horton, Dept of Computer Science, Exeter University, Physics Tower, Stocker Road, Exeter EX4 4QL.

Wolfgang Hromada, Othmar Lackner, Elektr.Bauelemente, Landstr.Hauptstr.37, A-1030 Wien, AUSTRIA.

J Hughes, Cirrus Computers Ltd, Waterman House, Chertsey Road, Woking, Surrey GU21 5BL.

Dr M.E.C.Hull, Department of Computing Science, University of Ulster, Shore Road, Newtown-Abbey, Co.Antrim BT37 0QB.

Mr D.G.N.Hunter, Standard Telecommunications, Laboratories Ltd, London Road, Harlow, Essex.

David Hurford, Gresham CAP, Osborneway, Hook, Hampshire.

Peter Ilieve, Ferranti Infographics Ltd, Bell Square, Brucefield, Livingston, West Lothian EH54 9BY.

Jeremy Jacob, Programming Research Group, Oxford University, 8-11 Keble Road, OXFORD.

Mr T.A.James, Thorn EMI Central, Research Labs, Trevor Road, Hayes, Middlesex.

Mr Paul G Jenkins, University College of Swansea, Singleton Park, Swansea SA2 8PP.

Dewi I Jones, School of Electronic Eng Science, University College of N Wales, Dean St, Bangor, Gwynedd.

Geraint Jones, Programming Research Group, Oxford University, 8-11 Keble Road, OXFORD.

Ian Jones, Thorn EMI Electronics Ltd, 120 Blyth Road, Middlesex UB3 1DL.

Dr J.M.Kerridge, Dept of Computer Studies, Sheffield City Polytechnic, Pond St, SHEFFIELD S1 1WB.

Nigel Kingswood, E.E. Department, University of Bristol, University Walk, Bristol.

Brian Lambert, TMC Ltd, Swindon Road, Malmesbury, Wiltshire SN16 9NA.

David Lamkin, Acorn Computers, Fulbourn Road, Cherry Hinton, Cambridge.

Dr A.E.Lawrence, Microprocessor Unit, Oxford University, Computing Service, 13 Banbury Road, Oxford.

John R Lawton, CAP Scientific Ltd, 20-26 Lambs Conduit Street, Holborn, London WC1N 3LF.

Andrew Lees, Telectronics, Bake Manor, Trerulefoot, Saltash, Cornwall.

Mr McPhee, British Telecom Research, R14.2.4, Martlesham Heath, Ipswich, IP5 7RE.

Mr R.F.Maddock, IBM UK Laboratories Ltd, Hursley Park, WINCHESTER SO21 2JN.

Gordon Manson, Sheffield University, Hicks Building, Hounsfield Road, Sheffield, S3 7RH.

F Marzano, I.N.F.N - Dipartimento Di Fisica, Piazzale A.Moro,2, I-00185-ROMA, Italy.

Peter Milligan, Queens University of Belfast, College Park, Belfast, N IRELAND, BT7 1NN.
Mr P Mills, Quintek, The College, College Road, Westbury-on-Trym, Bristol BS9 3EJ.
Neil Milner, GEC Avionics, Airport Works, Rochester, Kent, ME1 2XX.
Dr Heinz Muehlenbein, GMD, PO 1240, Sankt Augustin 1, D-5205, W GERMANY.
Alistair Munro, Computer Centre, Bristol University, University Walk, Bristol.
T Muntean, IMAG, Laboratoire de Genie , Informatique de Grenoble, BP-68, St Martin D'Heres F38402, FRANCE.
Kei Nakada, Personal Media Corporation, 8-1-11 Nishi-gotanda, Shinagawa-ku, Tokyo, Japan.
Chris Nettleton Esq, Systems Designers Ltd, 1 Pembroke Broadway, Camberley GU15 3XH.
J.R.Newport, CAP Scientific Ltd, 20-26 Lambs Conduit Street, Holborn, London WC1N 3LF.
Prof J.V.Oldfield, Dept of Elect and Computer Eng, Syracuse University, Link Hall, Syracuse, NY 13210 USA.
Dan Oestreicher, Cirrus Computers Ltd, Waterman House, Chertsey Road, Woking, Surrey GU21 5BL.
Paul Otto, Dept of Computer Science, University College, Gower Street, LONDON WC1E 6BT.
Mr K.J.Palmer, RSRE, St Andrews Road, Great Malvern, Worcs WR14 3PS.
Mr S Patel, International Computers Ltd, Wenlock Way, West Gorton, Manchester.
C.Pearson, CAP Scientific Ltd, 20-26 Lambs Conduit Street, London.
Mr R.Peel, University of Surrey, Dept of Electrical Engineering, Guildford, Surrey.
Mrs S.M.Peeling, RSRE, St Andrews Road, Great Malvern, Worcs WR14 3PS.
M.J.Pickett, CAP (Reading) Ltd, Trafalgar House, Richfield Avenue, Reading, Berks, RG1 8QA.
Richard Porch, Industrial Artist, 19 Market Road, Canton, Cardiff CF5 1QE.
Ben Potter Esq, ICL, Regent House, 87 London St, Reading.
Dick Pountain, Byte Magazine, 8 Rousden St, London, NW1.
A Pudner, British Aerospace, Millbanke Way, Bracknell, Berkshire.
Dr P W Rautenbach, Standard Telecommunications, Laboratories Ltd, London Road, Harlow, Essex.
Mike Reeve Esq, Dept of Computing, Imperial College of Science and Technology, 180 Queens Gate, LONDON.
B Rickman, Marconi Defence Systems Ltd, The Grove, Warren Lane, Stanmore, Middx.
Dr J.B.G.Roberts, RSRE, St Andrews Road, Malvern, Worcs.
Bill Roscoe Esq, Oxford University, Programming Research Group, 8-11 Keble Road, Oxford OX1 3QD.
A.J.Rose, GEC Avionics Ltd, Elstree Way, Boreham Wood, Herts WD6 1RX.
K Rouhi, 27 Stonor Road, LONDON W14 8RZ.

C.G.Rowden, Dept of Elec.Eng.Science, University of Essex, Colchester, Essex.

Mr D Roweth, Edinburgh University, Physics Dept, Kings Building, Mayfield Road, Edinburgh.

Ralph Rudolph, Messerschmitt-Bolkow-Blohm GmbH, Kreetslag 10, D-2403 Hamburg 95, Depart TE441, W GERMANY.

D.A.Rumball, Centre for Remote Sensing, Dept of Physics, Imperial College, London SW7.

Mr Gunter Sachse, Institut de Recherches, Robert Bosch S.A., B.B. 18, CH 1027 Lonay, SWITZERLAND.

Dan Sahlin, The Royal Inst. of Techn, S-10044 Stockholm, Sweden.

Rafael Sala, Intersoftware S.A., Pau CASALS 24, Barcelona 08021, SPAIN.

Dr Patricia Samwell, Department of Computing, Polytechnic of North London, Holloway Road, London, N7.

Dr Mike Sanderson, Essex University, Dept of Computer Science, Wivenhoe Park, Colchester.

R.P.Saunders, Trend Communications Ltd, Knaves Beech Estate, Loudwater, High Wycombe, Bucks HP10 9QZ.

Mr N.R.Saville, Algorithmic Systems Engineering Limited, 28 Valentine Road, Kings Heath, Birmingham.

G.H.W.M. Schellen, Zuringveld 34, 5467 KG, Veghel, Holland.

Edgar Scherer, Uni Saarbrucken, 66 Saarbrucken, Im Stadtwald, Rechenzentrum, Bau36, 6600 W GERMANY.

Michael R.C.Seaman, Torus Systems Ltd, Science Park, Milton Road, Cambridge CB4 4BH.

Dr John A.Sharp, University College of Swansea, Singleton Park, Swansea SA2 8PP.

Dr A.J.Simmons, Smith Associates Consulting, System Engineers Ltd, 45-47 High Street, Cobham.

Mr R.P.Stallard, Loughborough University, Ashby Road, Loughborough, Leics LE1T 3TU.

G Staniewicz, Weirlord Ltd, Victory House, Somers Road North, Portsmouth, Hants PO1 1PJ.

Dr B.R.Stonebridge, Bristol University, Computer Science Department, University Walk, BRISTOL BS8 1TW.

Charles Stormon, Syracuse University, 111 Link Hall, Syracuse, N.Y. 13210, USA.

A.G.Tagg, Oxford Polytechnic, Headington, Oxford, OX3.

D.P.Thomas, MOD (PE), RM117A D2, AWRE, Aldermaston, Reading, Berks RG7 4PR.

Dr G.Topping, Dept of Computing, North Staffs Polytechic, Blackheath Lane, Stafford.

Dr. H.R.A.Townsend, National Hosp, Queen Square, LONDON, WC1.

Mr P.Townsend, International Computers Ltd, Wenlock Way, West Gorton, Manchester.

Gunther Trautzl, Astek Elektronik GmbH, Carl-Zeiss-Str.3, 2085 Quickborn, WEST GERMANY.

Mr Simon Turner, Plessey Electronic Systems Ltd, Roke Manor,. Romsey, Hants SO5 02N.

Mr M Van Harmelen, Dept of Computer Science, University of

Manchester, Manchester M13 9PL.
Osmo Vikman, Labsystems OY, Pulttitie 9, SF-00810 Helsinki 81, Finland.
Dr J.S.Ward, MOD (PE), RSRE, St Andrews Road, Great Malvern, Worcs WR14 3PS.
Dr D.M.Watson, Thorn EMI Central, Research Labs, Trevor Road, Hayes, Middlesex.
Mr H.C.Webber, MOD (PE), RSRE, St Andrews Road, Great Malvern, Worcs WR14 3PS.
Dr P.H.Welch, Training Division,GEC Avionics Limited, Airport Works, Rochester, Kent ME1 2XX.
Tim Werner, DKF7 Heidelberg Inst. 08, 6900 Heidelberg, Im Nevenheimer Feld, W GERMANY.
R.W.Wilcock, Wilcock Software Services, 41 Berwick Road, Wood Green, LONDON N22 5QB.
Mr H.N.Williams, GEC Avionics Ltd, Airborne Display Division, Airport Works, Rochester, KENT.
Dr Shirley A Williams, University of Reading, Computer Science Dept, Whiteknights Park, Reading RG6 2AY.
Dr Burkard Wordenweber, Shape Data Ltd, 2 All Saints Passage, Cambridge, CB2 3LS.
E.J. Zaluska, Department of Electronics and Information Engineering, University of Southampton, SOUTHAMPTON SO9 5NH.
H Zedan Esq, Bristol University, Electrical & Electonic Enginering Dept, University Walk, Bristol.

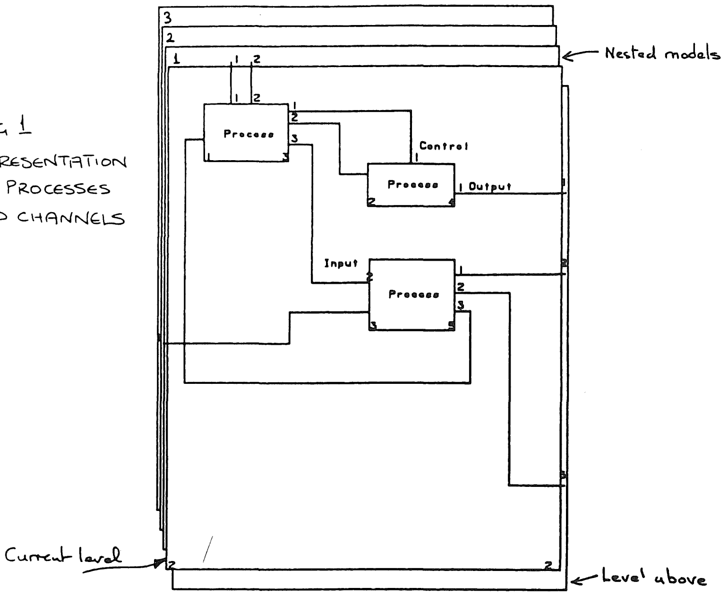


FIG 1
REPRESENTATION OF PROCESSES AND CHANNELS

See p18.

continued from back cover

## Program Exchange

The User Group does not provide a library but maintains a catalogue and, via the newsletter, allows members to publicise programs that they are willing to make available.

**Contributors** :- Please send a one page description to the coordinator of what your program does, its form (source/compiled etc), its hardware/operating system dependence, the exchange medium (type, format etc) and the name and address of the provider. It is advised that appropriate disclaimers be included.

**Requestors** :- Please make your request to the provider and not to the User Group. The User Group can itself provide no support for such programs nor can it accept any responsibility for problems that might arise due to their use.

Program exchange coordinator:

Mrs Sue Peeling
Royal Signals and Radar Establishment
St Andrews Road
GREAT MALVERN
Worcs WR14 3PS Tel: Ø6845 2733 (x2228)

## User Group Committee

In addition to the individuals mentioned above the following are members of the informal committee and would be willing to answer any queries about the group's activities.

Mr Gordon Harp (Chairman)
Royal Signals and Radar Establishment
St Andrews Road
GREAT MALVERN
Worcs WR14 3PS Tel: Ø6845 2733 (x2824)

Mr Chris Nettleton
System Designers Limited
1 Pembroke Broadway
Camberley, Surrey GU15 3XH Tel:Ø276 62244

Dr Geraint Jones
Programming Research Group
University of Oxford
8-11 Keble Road
OXFORD OX1 3QD Tel: Ø865 54141

# occam™ user group

·O·U·G·

The User Group is an informal organisation run by its own members. Its primary concern is the occam programming language, developed by INMOS Ltd. By virtue of its special relevance to occam, the INMOS transputer hardware is also included in the Group's area of interest.

The main aim of the User Group is to act as a forum for the interchange of information among existing and potential users of these products and as a channel for communication with INMOS. These aims will be met by organising meetings, issuing a newsletter, and supporting the exchange of programs between members.

Membership is free upon submission of the enclosed application slip. The User Group is mainly dependent upon its own members to contribute to meetings, to provide material for the newsletter and to make their occam programs available to other members.

## Occam Newsletter

This is the main vehicle for communication between members and is sent out free of charge. It is issued approximately twice yearly in June and December. Members are encouraged to submit short descriptions of their interest in and intended uses of occam. Text may be retyped, but diagrams should be suitable for reproduction. Please submit articles, letters, comments, enquiries on any occam or transputer-related subjects to the Editor:

Dr Martin Bolton
Department of Electrical and Electronic Engineering
University of Bristol
Queens Building, University Walk
BRISTOL BS8 1TR. Telephone: 0272 24161

## Meetings

These are held aproximately twice yearly in September and March. Apart from any necessary business they will include informal presentations by members and by INMOS. If you are prepared to give a presentation or act as host to a future meeting, please inform the User Group Secretary:

Dr Michael Poole
Software Support
INMOS Limited
Whitefriars, Lewins Mead
BRISTOL BS1 2NP. Telephone: 0272 290861

continued inside