occam™ user group · newsletter

No. 3 Summer 1985

CONTENTS

奥坎剃刀

勿增任何非必要之物

Translation:

"Occam's Razor

Do not add any inessential material"

(See page 7)

EDITORIAL NOTE

We continue to welcome items for inclusion in the newsletter. They should not normally exceed about 2000 words; longer offerings may require editing. Submission in machine-readable form can provide great savings in retyping - the Editor (address on back cover) will be pleased to give further details.

Martin Bolton (Editor), Michael Barton (Assistant Editor)

SECOND TECHNICAL MEETING OF THE occam USER GROUP

Michael Poole, Inmos

The second technical meeting of the group was held at St Catherine's College, Oxford on 29th March 1985. The College is one of few in Oxford which has ample parking space for visitors' cars, but compensates for this by being very difficult to get to around the one-way traffic system. Nearly 150 people attended the meeting split fairly evenly between academic and industrial organisations. We were particularly pleased to welcome visitors from the USA and from Germany.

Professor Tony Hoare of Oxford University opened the meeting by describing some of the history of "Communicating Sequential Processes" showing how many of the ideas now in occam originated in his work from 1974 onwards, first widely publicised in his well-known CACM paper published in 1978. He explained that he omitted named channels (in favour of named processes) from that paper, not because he did not like them, but because he was worried about introducing too many new ideas at the same time. He now supports the occam model in this area.

Alan Burns from the University of Bradford, who teaches occam both in the University and commercially for The Instruction Set Ltd, gave an account of his experience teaching occam. He finds that concurrency comes easily to students who have not had to struggle with previous inadequate language features for parallel systems. The one most important attribute of occam for teaching purposes is that it is a small language and there is no difficulty in getting the whole language over in a short course.

Alistair Munro from the University of Bristol showed us some of his computer-generated pictures showing the structure of processes expressed in occam. This system was described in the previous OUG newsletter.

Ian Page of Oxford University spoke of his ambition to invent a fully dynamic animated graphical version of the Adventure style of personal computer game. This desire has motivated his research into the problems of distributing the various stages of the "Graphics pipeline" on to a network of processors (transputers?). He mentioned the idea of a "square word" whose bits represent a square set of raster dots in a picture, rather than the much less appropriate linear words used in most current systems. Occam would seem to be a very useful tool in aiding the discussion of design problem in this kind of system.

Russell Wayman, a software project leader at Inmos, then gave an exposition of the programming style that has been adopted in writing the occam programming system, the integrated editor and compiler for occam itself. The ability to structure a large body of software as a collection of nearly autonomous process communicating in a limited number of different ways with each other has been a great benefit of using occam as the implementation language for this software. He emphasised the need for a language to describe the protocol of message interactions.

After lunch Jon Kerridge of Sheffield Polytechnic gave an entertaining account of his experience implementing the occam Portakit on a Unix system. He managed to get the system compiling the first example program 12 days after first loading the tape. He used the Fortran example interpreter and mentioned some of the problems which arise because no two Fortran systems are the same. The talk led to a plea for someone to write an optimised compiler generating the same PIS code, but avoiding the significant workspace requirements and interpretive execution time penalty of the system as distributed.

Peter Welch, who at the time of the talk was an industrial fellow at Marconi Avionics, but who is now back at the University of Kent, then gave a talk on a functional style of programming, and his views on ways in which occam could be enhanced to facilitate this. He achieved this with dual overhead projectors - the provision of which had been a mystery until he spoke. This talk was related to his article in the previous newsletter.

We then had a complete change - Dave Simpson of SCICON gave an introduction to some of the problems of testing a newly designed and constructed circuit board based around transputers. This brought out the difficult problems which are facing engineers now that more and more is getting inside increasingly complex chips in which it is no longer possible to insert logic probes into data paths, etc. A need for a "Link Breakout Unit" which could be inserted into a link between transputers to enable the traffic to be monitored in some way emerged.

The meeting closed with a position statement from Peter Cavill, Director of the Microcomputer group at Inmos on the progress of Transputer silicon. He mentioned the difficulties that had been experienced in getting the manufacturing process implemented at Colorado Springs and the decision to transfer development to Newport. He expressed confidence that significant numbers of working chips would be available for customer evaluation later this year, even if the memory size might have to be reduced for a short while. He also discussed longer term plans for silicon systems which is a new business area intended to capitalise on Inmos silicon design skills for the production of semi-custom VLSI devices of various kinds.

The meeting was well received by most of those present, but the plea for more informal discussion time was made again. We hope that the next meeting which will extend over two days will meet people's needs in this area.

I should like to close by thanking Geraint Jones for making the initial contact with the College, the College for their excellent facilities, Gordon Harp for chairing the meeting, and my colleagues Janice Seymour and Kathy Yarnold for managing all the administrative work so effectively.

occam PORTAKIT IMPLEMENTATIONS

John Wright, Inmos

This list contains the names of those people and organisations who have returned implementor registration cards and have agreed that their names may be published.

I have added the makes and names of the computers and operating systems which each implementor was considering.

We have produced a document describing an occam runtime support systems. This document will be of interest to anyone designing a native code implementation of the portakit. It is available from Software Support at Inmos.

Steve Taylor, Dept of Applied Maths, Weizmann Institute, Rehovot, Israel

Peter Horan, Div. Computing and Maths., Deakin University, Geelong, Victoria 3217, Australia
  Gould PS2000   Unix

T Muntean, Laboratoire de Genie Informatique de Grenoble, Institut IMAG, BP68, 68402 St-Martin-D'Heres, France
  VAX   Unix
  BULL-Sems SPS7/SM90

Mr Donguy Epshom, 13 Rue du Chatellier, BP426, 29275 Brest Cedex, France
  Hewlett Packard HP9000

Thierry Gagnebin, LAMI-EPFL, Av. de Cour 37, CH-1007 Lausanne, Switzerland

Werner Hett, Ingenieurschule HTL, Quellgasse 12, CH-2500 Biel 3, Switzerland
  Data General   AOS/VS

Paul Kussmaul, Hewlett Packard Gmbh, Herrenberger Str. 110, 7030 Boblinger, W Germany

HP 9000	HPUX

Dr Schroder, TU Hamburg-Harburg, Harburger Schloss-Str 20, D-2100 Hamburg 90, Germany

Prime 9950	Primos

Craig Davidson, Gryphon Systems, 922 Grange Hall Rd, Cardiff, CA 92007, USA

IBM pc	
Apple Macintosh	

Robert Gustafson, Simulation Specialists Inc., 609 W Stratford Pl., Chicago, USA

Wicat	MCS

Charles Askew, Dept of Physics, Univ. of Southampton, Highfield, SO9 5NH, England

PERQ	PNX

G P Otto, Dept of Computer Science, UCL, Gower Street, London WC1E 6BT, England

VAX	UNIX

Geraint Jones, Oxford University P.R.G., 8-11 Keble Road, Oxford, OX1 3QD, England

HLH Orion	UNIX

F J Maddix, Bristol Poly, CSM Dept., Coldharbour Lane, Frenchay, Bristol, England

Prime	Primos
VAX	VMS

R M A Peel, Dept of E & E Engineering, Univ. of Surrey, Guildford, GU2 5XH, England

DEC VAX 11/750	UNIX
Prime 750/9950	Primos/UNIX
RCA 1802	

Bob Bird, Dept of Computer Science, Univ. of Kent, Canterbury, CT2 7NF, England

DEC VAX	UNIX

J M Kerridge, Dept of Computer Studies, Sheffield City Poly., Sheffield, S1 1WB, England

Perkin Elmer PE3210	UNIX
IBM pc	PCDOS
IBM 4341	VMCMS

Alan Mycroft, Computer Lab, Cambridge University, Cambridge, England

IBM 3081	MVS

B Jane Curry, Computer Centre, Chelsea College, London, England

Perkin Elmer PE3205	UNIX

Peter Lynch, Prime Computer Ltd, The Merton Centre, Bedford, MK40 2PN, England

Prime 50 Series	

Nick Jeffery, ICL, Westfields West Av., Kidsgrove, Stoke-on-Trent ST7 1TL, England

ICL 2900 series	VME/b

Cyrus Hazari, IT Research Centre, University of Bristol, Bristol BS8 1TR, England

Apollo	Aegis

occam PRODUCTS FROM Inmos

Chris Followell, Inmos

We offer a range of support products for the occam language. For customers who wish to evaluate the benefits of parallel programming we offer the occam Evaluation Kit. This is an interpretive implementation of a subset of the occam language based on the UCSD P system. It is available from our distributors for a range of computers and costs £175.

For the development of full occam programs we have available the occam programming system for VAX computers running the VMS operating system. This is available for £1500.

We also have available the occam portakit. This package contains details of how to create your own interpretive version of occam for your own host computer. It is available from our distributors for £250.

The Transputer will be made available in the 4th quarter of this year on a range of Evaluation boards, priced from £1500. The first board will comprise a 32bit Transputer, some fast RAM and 2 serial ports for communication with the Host computer. Full details are available from Inmos on request.

In order to develop code for the Transputer we will be offering a Transputer Development System. This will comprise a standalone computer system (C.P.U., 1MegaByte of memory and 15 MegaBytes of Winchester mass storage.) running the development software. This will comprise an occam compiler, transputer code generator, debug software, performance estimator and memory interface configuration program. It will sell for £8,500.

We will be offering the same software package for the VAX/VMS system. This will cost £3000 or will be offered to existing users of the O.P.S. as an upgrade at £1500.

Finally the Transputer itself will be available from the 1st quarter of 1986 in production volumes at a competitive price.

occam AT PEKING UNIVERSITY

M.H. Rogers, Department of Computer Science
University of Bristol

Earlier this year I visited the Department of Computer Science at Peking University, People's Republic of China, and had discussions with Professor Cho-Chun Hsu and his colleagues who are planning to use occam as a hardware description language for database applications. Readers expecting to see occam program listings in Chinese will be disappointed; the Department is currently using the evaluation kit and the portakit and has no immediate plans to implement a Chinese character set version. However, a Chinese translation of Occam's Razor, kindly supplied by the Department, appears on the front cover of this issue. (Thanks to Dr W. Fong of Bristol for redrawing the script.)

AN occam COMPILER

John Ainscough* and Alistair Brightman
Brunel University, Uxbridge, Middlesex, England.

(* now at Manchester Polytechnic)

ABSTRACT

The paper describes an occam Programming System designed as a final-year project for the Special Engineering Programme, Brunel University. The system compiles programs written in occam. It incorporates a single-pass, recursive descent type compiler, written in PASCAL, which produces p-code for a hypothetical stack machine.

An interpreter, also written in PASCAL, runs the p-code. It simulates concurrent execution of up to 64 processes by switching from one to another after every occam statement.

There has been much interest recently in the introduction of the occam programming language, and in particular its simplicity and ease of use. The purpose of this paper is to demonstrate that these features relate not only to the language but to its implementation.

1. Introduction

The compiler itself is written in PASCAL and compiles code for a P machine interpreter which is again written in PASCAL. The language that is compiled is due to Inmos (1), and has some minor variations from that used in Proto occam.

2. Lexical Analyser and Parser

The lexical analyser is complex. This is because of the textual significance of the end of source line, and the use of indentation to identify the structure, and the large number of special character sequences.

The occam parser was constructed by producing a set of syntax diagrams for occam, then using the rules described in Wirth (2) the diagrams were converted into a PASCAL program. A procedure TESTENDLINE was included to enable the lexical analyser to compute the level of indentation for the current line which is held in a global variable IND. This must be checked after every invocation of TESTENDLINE and before every component of a construct.

The data structures of the compiler are based on those of Ben-Ari's modified version of PASCAL-S (3).

3. The P-Machine

The p-machine is a modified version of the machine of PASCAL-S with p-code and data stores, S; process and channel tables; and a top of stack pointer T; an instruction register I.R.; program counter PC; and a base address register B. The static link is held in a DISPLAY table.

3.1 Machine architecture for parallel processing

Additional facilities must be provided within the machine if parallel processing is to be simulated, some of these features are derived from Ben-Ari's p-machine (3).

The machine must be able to retain the state vector of several processes, this is achieved by a process table PTAB. Each PTAB entry contains the pc, s, b and display. The NPR field contains the number of parallel sub-processes activated by the process attached to this entry.

The allocation of PTAB entries to parallel processes occurs at run-time. The asynchronous nature of the parallel construct in occam means that processes can terminate in any order, so the PTAB entries actually in use at any time are not necessarily contiguous. To utilise the array effectively, the array FREELIST contains a list of those entries which are free. When a component process of a parallel command is activated, the number of the next available PTAB entry is taken from the free-list and the pointer FP is incremented. On termination a parallel process decrements FP and puts the number of its PTAB entry into the free-list for re-allocation.

A dynamic data structure is required to keep a record of the processes which are ready to run. This is done by the LAST and NEXT fields of PTAB, which are used to link active processes into a circular, doubly-linked list. Parallel execution of a number of processes is simulated by switching from one PTAB entry to the next in the chain, at appropriate intervals, and using registers from that entry to fetch and execute instructions.

Channels are represented within the p-machine as an array of records CTAB, containing fields for data-transfer and handshaking between processes. This is not used as a stack, because it is assumed that channels correspond to physical links, the addresses of which are known at compile time.

3.2 Scheduling of Parallel Processes

Scheduling the execution of a number of processes on a single processor can be implemented in some of several ways, the simplest being to switch from one process to another at the end of every machine cycle (i.e. instruction-level concurrency). Another strategy is to run a process until it cannot proceed, then choose a different one.

In this implementation statement-level concurrency was implemented by simply adding an instruction to switch process to the p-code instruction that terminates ordinary sequential processes. A switch is also made as the last instruction of the SUSPEND procedure, so that the chosen process is one from the active list. Changing from executing one process to another is done by the procedure SWITCH, which normally just finds the PTAB index of the next process in the active list, and assigns this to the current process pointer.

The algorithm has been refined slightly in line with work done by Roper and Barter (4) who have implemented a CSP-type language. They noted that the use of a PASCAL 'read' statement to input from a terminal stops all the other processes unnecessarily if no data is ready. Therefore, input from the terminal should be delayed until no other process can execute, in which case any extra delay is immaterial. A queue of processes waiting to execute a PASCAL 'read' is maintained within PTAB using the link fields. The procedure SWITCH chooses the first process in the queue allowing the 'read' to execute if no other process can proceed. Processes in a 'wait' state (i.e. alternative constructs waiting for a guard to become ready and wait primitives) are considered unable to proceed - the number of such processes is NWAIT, which is incremeted at the start of a wait or alternative and decremented when it has executed.

The execution cycle of the p-machine is repeated until the end of the program or until a run-time error occurs. The status of the processor is indicated by the processor status register PS. During operation, PS = RUN, but when an error is detected, a p-machine instruction can modify this to signify which error has occurred and stops the processor. The source line being executed when the error was detected is found from a "map" of the p-code in the array MAP and is printed with the error message.

4. P-Code and Code Generation

The design of the p-code instructions and the code-generation parts of the compiler were implemented together. Many of the features in this section are derived from Ben-Ari.

The wait primitive is implemented as a busy wait similar to a WHILE loop. The WHILE, SEQ and IF are straightforward. The sequential construct with a replicator is very similar to the PASCAL "for" statement and the WHILE is identical to its PASCAL counterpart. The IF is implemented as a sequence of expression evaluations and conditional jumps past the component processes. A 'computed jump' instruction was added to the p-machine, used to jump to the next construct on completion of a component process.

4.1 Scheduling

There are two procedures local to the interpreter. SUSPEND and ACTIVATE are used to suspend and activate parallel processes. Activation of a process involves taking the PTAB entry for an existing process and adding this to the list of active processes after the last entry. The process may then be chosen for execution at some time by the scheduling algorithm; which is a simple "round robin" scheduler. A suspended process is simply removed from the list. If the process is to be terminated, its PTAB entry can be added to the free-list, for possible re-allocation.

The activation record for a parallel process consists of:

STATIC LINK

PARALLEL LINK

REPLICATOR INDEX (if necessary)

The static link performs exactly the same function as for any other process. However, it points to a location in the stack space of the parent process. The "Parallel link" is a record of the PTAB index of the process which activated this process. This is required to inform the parent process when the child process has terminated. The replicator index is only allocated if required when it holds the value of the replicator for this process - each parallel process in a PAR construct with a replicator must be supplied with its own copy of the replicator value with which it is associated. The compiler takes account of this when declaring the replicator index: if the construct is a PAR, then its address is recoded as being the base address for the current data segment plus a two location offset, otherwise, the index is allocated a location in the previous data segment, as normal.

4.2 Alternative Processes

The alternative construct was quite complex to implement because of the complicated syntax and the number of possible way of evaluating a guard.

The evaluation of a guard leaves three data items on the stack. These are: -

1) The address of the code to be executed should the guard succeed

2) The address of the channel

3) The state of the guard

An instruction at the end of the construct will use this data to decide whether or not to execute the option.

The instruction chooses which guard to check first on a random basis, and will select the first ready guard it encounters.

If a guard is chosen, its record replaces the first one. The stack is then cleared of all other records and the top of the stack now will indicate a guard is ready by being set to true and the next two locations will contain the data needed to execute the guarded process. When an ALT is used as an option, this ensures that it behaves just like any other, leaving a single record on the stack. The last instruction of the construct examines the top of the stack, if a process is chosen the address is fetched from the stack, and executes a jump to that location. If no guard is chosen execution jumps to the beginning of the construct and the cycle repeats.

The first instruction executed when an input guard is chosen performs the input operation. Then the main process itself is executed. After this the computed jump instruction clauses a jump to the exit address previously stores on the stack.

4.3 Input and Output primitives

Input and output between p-machine processes use the channel table CTAB. Each record has a data register which holds one word value, and two "handshaking" registers. The handshaking is used to activate and suspend processes.

There are two pre-defined channels KEYBOARD and SCREEN, which use standard Pascal 'read' and 'write' procedures. The output buffer is compiled by checking for ASCII line feed characters in the data, and issuing a 'writeln'.

5. System Testing and Performance

The testing of the compiler was accomplished using a suite of ten occam programs that were supplied by Inmos and modified to the occam definition used here. These tested most aspects of the execution including allocation of variable storage, expression evaluation, communications and constructs. Also they use most of the valid forms of all the syntactic entities except slices.

A problem encountered as a result of running these test programs was the slowness of execution of the ALT construct particularly when used with a replicator.

The computer compiles code at approximately 2000 lines per minute when listing the occam code to a file and this reduces to 1500 lines/minute when listing directly to the screen on a VAX 11/730.

Internal debugging facilities are included in the compiler and the interpreter.

6. Conclusions

The objective of the project, to produce a working, portable occam programming system, has been achieved. A simple and fast compiler has been written, which compiles all the features of the language except constant expressions prioritised constructs and PLACED processes. Channels are currently defined globally. The system successfully compiled a modified suite of test programs supplied by Inmos which exercises almost all the features of the language.

Acknowledgement

The authors would like to acknowledge the support given to this project by Dave May and his colleagues at Inmos.

(1) occam Definition. Due to David May (Inmos) November 1983.

(2) Wirth, N. 'Algorithms + Data Structures = Programs' Prentice-Hall International (1976)

(3) Ben-Ari, M. 'Principles of Concurrent Programming' Prentice-Hall International (1982)

(4) Roper, T.J. and Barter, C.J. 'A Communicating Sequential Programming Language and Implementation' Software - Practice and Experience (v11) 1215-1234 (1981)

Commercial Application Packages

Cliff Dilloway - Independent Consultant

The purpose of this note is to outline a technique for implementing a wide ranging commercial application package in occam. The intention is to be practical and to throw up ideas, the author has no intention of taking the proposition any further. Commercial applications packages are systems used in business for accounting, stock control and the like.

The notion is based on two observations that have not been disputed and a reflection based on their truth. Every commercial implementation of a system is different from every other commercial implementation. Secondly, no commercial implementation does any "thing" that is unique. The reflection derived from those two statements is that there must be a (large) set of things which in some combination or other will meet the requirements of every commercial implementation. (1)

Whatever these "things" are they have an interesting property in common with a process in occam. Every "thing" must be able to co-exist with every other "thing" just as every process must be able to co-exist with every other process. The interesting notion is that if all "things" can be implemented in occam we have the makings of a universal system. There are two problems, the purely technical one of finding a means of ensuring that the right "things" get done and the practical problem of ascertaining what all the "things" are. Our interests are in the second problem.

There are two pressures that are leading to the identification of "things". The first is the process of selection of commercial applications packages for purchase. A book (2) intended to assist in ascertaining the requirements of a commercial implementation lists all the possible "macro-things" that might be required. As a first book of this nature it does not go into the detail required for "things" but it is comprehensive and a good starting point for further research.

The second pressure is really the converse of the book. Taking as a group all the commercial applications packages for a particular type of system there is a distinct tendency for each of them to be enhanced to the point where they are all able to carry out the same functions. These functions are "macro-things". Market place pressures are identifying what needs to be done in a commercial system.

If it were possible to implement a "thing" system it would help overcome one of the greatest difficulties in commercial applications. Businessmen do not in general know what they want a computer to do for them and in any case commercial pressures are such that requirements can change quite rapidly. A "thing" system would be able to cope with anything that needed to be done providing the control mechanism we have not discussed can be provided.

It will be interesting to observe whether commercial systems develop on these lines.

References

(1) Applications Packages. Cliff Dilloway "Software World" Vol. 13
No. 2 1982

(2) The Software Sifter. Philip Franket and Ann Grass. Collier Macmillan 1984.

SOME EXPERIENCES OF occam IMPLEMENTATIONS

H Zedan and C Hazari, IT Research Centre, University of Bristol

Summary

This article attempts to present a methodological technique for testing some occam implementations. An attempt is made to rectify deficiencies in two current implementations

(a) by the introduction of a new primitive

(b) via alteration to the Portakit implementation.

1. [The following is by no means a complete assessment of the current occam implementations. However , it is more toward a report of some deficiencies in the implementation].

An implementation of a programming language should conform to the same rules which the language itself satisfies. We shall demonstrate that some of the current occam implementations do not conform to the most basic algebraic rules of occam. The two implementations in mind are the OPS occam (running on VAX under UNIX) and the Portakit implementation (running on Apollo under AEGIS) . The basic algebraic rules which we are interested in are those concerning PAR and ALT:

(R1) PAR is commutative. I.e PAR (a , b) = PAR (b ,a).

(R2) ALT is commutative. I.e ALT (g1 s1 , g2 s2) = ALT(g2 s2, g1 s1). Moreover, in the case where both g1 and g2 are

Consider the following simple occam PROCs:

```
P1:: ALT                    P2:: c1!'1'              P3:: c2!'2'
       c1?x
         screen!x;-3
       c2?y
         SKIP

P1':: ALT
        c2?y
          SKIP
        c1?x
          screen!x;-3
```

Note that P1 and P1' are (algebraically) equivalent.

Also consider the following programs:

```
(a) PAR        (b)  PAR      (c)  PAR
      P1              P3            P3
      P2              P1            P1'
      P3              P2            P2
```

On running these programs on OPS occam we get the following surprising results:

```
(a) 1 DEADLOCK  (b) DEADLOCK  (c) 2 DEADLOCK
```

Another striking drawback in this implementation is illustrated in the following example. Consider

```
P4::  SEQ                  P5:: screen!'x';-3
        screen!'1';-3
        screen!'2';-3

PAR           PAR
  P4            P5
  P5            P4
```

gave 12x and x12 respectively.(Note that a sensible implementation should give DEADLOCK in both cases!.)

On the other hand ,when experimenting with the Portakit implementation similar problems were found. For example, let

```
P6::  screen!'1'           P7::  STOP

PAR           PAR
  P6            P7
  P7            P6
```

gave stop and 1 respectively. In here the scheduler starts with the last component of the PAR, and if it encounters a STOP process then the entire program aborts. Also, because of the scheduler a passable guard in ALT is picked accordingly. For example consider

```
P8:: ALT
       c1?x
         screen!x
       c2?y
         screen!y
```

then

```
(a)' PAR               (b)' PAR           (c)' PAR
       P2                     P3                 P8
       P8                     P8                 P3
       P3                     P2                 P2
```

will give: 1 Deadlock, 2 Deadlock, and 1 Deadlock respectively.

We conclude that these simple examples are among many (not included here for the lack of space) which show that both implementations do not conform to occam rules. However, it was not only intended to report some faults in the implementations but also to give a methodological technique for testing an implementation of a language. For simulation purposes,for instance, the above mentioned implementations are obviously not suitable.

2. In the literature there are two kinds of nondeterminism. The two types do differ in the way nondeterminism is resolved , and have a different effect on the way the process terminates. The first kind, which is known as 'local nondeterminism', is the 'old' notion which was introduced by Dijkstra [1]. In this type, the process P, which can communicate with any of the processes P, P,...,P, for example, decides 'on its own', i.e locally,which one to communicate with. In the second type, which is known as 'global nondeterminism '[2],the other process's willingness to communicate is taken into the process decision. In the light of [2],we suggest having a new primitive which enables a clean termination of a process which depends on the communication guard of another: Upon termination process P signals an END signal which will be sensed by all processes communicating with it. Thus if this signal is sensed by a process then its decision will be either to abort itself or to wait for the communication may take place in the future. For example (a)',(b)' and (c)' above will terminate properly if P8 takes the new form:

```
P8:: ALT
       c1?x
         SEQ
           screen!x
           END
       c2?y
         SEQ
           screen!y
           END
```

3. Tampering with Portakit. The current implementation of the Portakit interpreter aborts the program being executed if a 'STOP' is encountered in a process inside a PAR. This should not occur. The compiler generates an 'error' instruction (PIS code) in response to a 'STOP'. A simple solution is to force a 'start_next_process' when an 'error' is encountered. In the 'start_next_process',if the active process queue is not empty, the process which called 'error' is list is empty,and the number of processes waiting for synchronization, n,is zero, then the interpreter can be stopped, otherwise either deadlock has occurred or a process has been abandoned. (n may be deduced from the number of unsuccessful inword and outword calls.)

The primitive introduced in the previous section can bring about distributed termination by arranging for an abandoned process to self-abort. The Portakit interpreter could support distributed termination, but in the general case ,it should distinguish deadlock and process abandoned.

In the case of example 1,after the ALT has communicated with P2, say, it will force a start_next_process. P2 is then active and successfully terminates with an endp instruction. This forces a start_next_process. The active list is empty but the process count is non-zero. Arrange for that part of start_next_process to be executed according to a process count of 1. This will pick up execution at the next instruction in the parent process.

[1] E W Dijkstra--"A discipline of programming"--Prentice Hall 1976.

[2] N Francez, C A R Hoare--"Semantics of nondeterminism, concurrency and communication"--In "Mathematical Foundations of Computer Science 1978", Lecture Notes in Computer Science, Vol 64.

<u>An occam Interpreter for Unix</u>

Malcolm K. Crowe and Martin E. Corr, Paisley College of Technology

1. Introduction

The purpose of this note is to announce the completion of an occam interpreter for the UNIX operating system. The present implementation is on a VAX, but it is written in C and should be fairly portable. It is available free of charge for educational and research purposes. All features of the occam language have been implemented, including named processes, tables, and replicated constructs; but excluding vector and configuration operations.

The interpreter uses a single Unix process for the occam program, with the various component processes of the program being scheduled and executed in a timeshared fashion. The interpreter also allows the programmer to follow the progress of the program in terms of the processes within it. Since the programming of real time and concurrent applications is much more difficult than conventional ones, this facility is a very useful one.

The interpreter enforces all the run-time checking required by the occam reference manual, including the use of variables in the PAR construct. This ensures that parallel processes are genuinely independent and can be shared among different transputers. Future work at Paisley will implement the extended features not currently supported, and will allow the monitoring of such aspects as data flow, parallelism and local storage limits for a given distribution of an occam program over different processing elements.

2. Structure of the Interpreter

The interpreter consists of a translator based on Yacc which translates the given occam program into a tree-like data structure, and an interpreter for this execution tree.

The Yacc grammar for occam may be of general interest. One feature is that changes in indentation level in the occam source are converted by the lexical analyser into BEGIN/END pairs. The only departure from the language as specified in the reference manual is that at the end of the declaration of a named process, the colon must appear on a line by itself, at the correct indentation level.

The translator builds the execution tree, assigning tree addresses for all variables, channels, etc, declared by the program. These addresses consist of a (depth, offset) pair: the depth is the static depth, i.e. the depth of each constituent process of a construct being one more than the construct itself, and the offset is the offset within either the constant table of the process, or the variable storage of the activation of that process.

Constants and descriptors for references are held in the constant table. To assist in error checking for use of variables by parallel processes, a use structure is built for any variable, or channel, that is potentially used by more than one of a set of parallel processes. This is necessary for tables, or replicated processes, since it is usually not possible to check at compile time whether the proposed use is legal. Shared variables or channels are then accessed via their use structure, which is part of the activation of each parallel construct intermediate between the declaring process and the accessing process.

3. The Execution Phase

The execution tree contains the static information about all processes. Each process is a node of the tree, with primitive processes at the leaves, and constructs as internal nodes. When a process is started, an activation record for it is created, including space for its local storage, and this activation is placed on the ready list. Selection of a ready process from this list is performed at random.

All references to variables and channels are converted to storage location offsets for the tree. Hence, at execution time, when the variables/channels are being used, the memory allocated for them is referenced directly, instead of using names, for which searches must be made.

A channel is implemented as a pair of activation pointers. An activation waiting to do I/O waits in the channel until the matching operation on this channel is selected. Then the I/O takes place, and the activations are terminated.

Whenever an activation terminates, its parent process is rescheduled. If it is a SEQ construct, it will schedule the next of its offspring if any; if a PAR, it will terminate if it has no more children.

4. Running the Interpreter

Command format is:- occam [-Y] [-T] [-d] [-D] [<filename]

This command compiles and runs an occam program. Occam programs may be either typed straight into the interpreter "interactively", or, preferably, read from files. "filename specifies such a source file. The protocol is for these files to have ".oc" suffix, although this is not enforced.

-Y: Trace the path of the parser through the grammar. i..e. it produces a description of the movement of the parser state machine through the states describing the grammar.

-T: This option will:-

1. print out the symbol table

2. print out a description of the program directly from the execution tree on completion of the program parse and before execution. This can be useful because it describes exactly the input program in terms of the arrangement of all the processes, static depths, dynamic storage within and the filling-in of the contents of named processes when called.

-d dD: These flags provide the means for run time checking of the progress of the program. As processes within the program are scheduled, executed, and terminated, these flags produce descriptions of these events. With the -D flag, complete process descriptions are given for the process involved in any run time event. This flag tends to be quite verbose and should only be used in very detailed examination of the execution is required.

With the -d flag, only the run time events are described, with just a reference to the process in question being produced. Thus a more concise description of execution is produced; when used with the

-T flag, the progress of the occam program may be easily followed.

5. Error Reporting

The Parser's error reporting is quite basic, and leaves room for improvement. When a syntax error occurs, the line at which the error was detected is displayed. Depending on the error, some information concerning its type may be produced.

Errors such as illegal use of shared variables and process deadlock are trapped with suitable messages being produced. Use of the run time flags provides for easier isolation of the problem.

6. Terminal I/O

This is achieved using a pre-declared channel called "TTY".

```
e.g. VAR x:
       SEQ
         TTY? x
         TTY! x + 1
```

i.e. read x from terminal and output its value +1 back to the terminal. Both processes will wait until the desired operation may be performed.
Note that the value read in by the process

```
TTY? x
```

is in ASCII format and must be converted appropriately.
It is important to note that the TTY channel is a non-standard facility: it is provided for ease of use of the interpreter.

7. References

1. Inmos Ltd [1984] - occam Programming Manual. (Prentice-Hall).

2. Stephen C. Johnston [1978] - Yacc: Yet Another Compiler Compiler. (Unix Programmer's Manual, Seventh Edition, Volume 2B)

3. D.May [1983] - "occam" (Sigplan Notices, Vol. 18, No.4)

4. D.Q.M. Fay [1984] - "Experiences Using Inmos Proto-occam" (Sigplan Notices, Vol. 19, No.9)

<u>Experience with the occam Portakit</u>

Jon Kerridge, Sheffield City Polytechnic, Pond Street, Sheffield, S1 1WB

Introduction

Experience with the porting of the occam portakit is described. A brief overview of the portakit occam system is given including details of changes which have been incorporated into the language. The transportation of the software to a UNIX system is described together with the problems incurred. The results of the exercise are then given. The intended use of the portakit system is then presented and finally some conclusions are drawn.

Overview of the Portakit

The software is supplied on a tape together with some documentation comprising an "Implementor's Guide" and an "occam Compiler Guide". Also enclosed was a single sheet giving the tape format and blocking information.

The Implementor's Guide contained the syntax and semantics of the portakit machine, which is understood to be very close to the transputer itself. A description of how files and input/output are related to the host operating system is also given. A suggested implementation strategy, especially if one of the supplied interpreters could not be used, is described.

The software included on the tape was some test files to help debug an interpreter, some example interpreters written in a variety of languages, the object of the compiler and some files containing the source of the occam compiler. The test files generally included three files for each test. A ".occ" file which contains the occam source coding. A ".pis" file which contained the diagnostic listing generated by the Pascal interpreter. For some of the tests the diagnostic file was not included. It should also be noted that the different interpreters supplied produce slightly different diagnostic output.

Language Modifications

The major change to the language has been the incorporation of "hard channels" so that communication with the host environment can be easily undertaken. Each device or handler has associated with it a specific channel number. (screen is 1, keyboard is 2). The channel number is incorporated into the channel declaration by "CHAN name AT n". Each file has two channels associated with it, one to send data to the handler, the other to receive data from the handler. Associated with all the handlers are a set of predefined control codes.

Other language modifications include a process called STOP which starts but never stops. The semantics of IF have been changed so that the process stops if none of the conditionals is true rather than terminating. The determination of time is now found by means of a channel TIME rather than the variable NOW. All keywords of the language must be in upper-case and case is differentiated in identifiers.

Transportation of the Software

The tape was written in ANSI format at 1600 bpi. This format was not amenable to being read by the UNIX TAR command. The tape could be read by the DD command however. In this case a program had to be written which inserted an "end of line" character at the end of each record. Using DD there were eight records which had to be skipped between the end of one file and the start of the next. The name of a file could be extracted five records after the end of the previous file.

It had been decided to use the Fortran 77 based interpreter as no 'e' version was available. This was done because the Pascal on our Unix system was interpreted. We found some errors in the Fortran system which included equivalence not working. The supplied interpreter used Z and Q formats which were not available on the Unix fortran.

Some problems were encountered with the portakit software itself. Generally non-standard usage was well commented except that the use of Q format was not noted in the subroutine INFILE. The interpreter allocated Logical Unit Numbers 11, 12 and 13 to disc files and the Unix fortran system only assigns LUN's up to 10. The implementation assumes the ability to dynamically create the diagnostic file at one point to generate error messages, when a file is "opened" which cannot be accessed.

Results

The interpreter was fully tested and working in twelve days of very part-time effort. Unfortunately, it only compiled at ten lines per minute. On discussion with other people it was found that this was particularly slow but not too unrepresentative of high level language implementations. Inmos have a version written in VAX assembler which runs at about 250 lines per minute on a 780.

Intended Use

The hard channels are easy to extend and thus a version is being created which will run on a set of Z80 micros connected to a Cambridge Ring. Thus two hard channels will be defined to interface to the ring. A similar project is being undertaken on a network of IBM PC's using a Corvus network.

Conclusions

The supplied interpreters are only models of what can be done. The intention would be to use the high-level language versions as a basis for producing an assembler language version. As a means of getting a more flexible implementation of occam which is not tied to a particular implementation such as the occam evaluation kit, the portakit provides a viable route.

NEW BOOKS

Programming in 'occam' - an introductory text

Occam, as the Programming Manual announces in its first sentence, is a new programming language: a language, therefore, to which there are as yet few published introductions. It is a tribute to the simplicity of the language that the fifty short pages of the Inmos reference manual serve it so well. What that manual does not address, however, is the radically different style which is appropriate to occam programs.

The occam programmer is encouraged to think of process creating and synchronization as being as cheap as the other primitive operations of the language. Even implementations on a conventional single processor machine need make process creation and scheduling no more expensive than procedure invocation. This startling scale of costs gives much greater freedom of expression, and leads naturally to a coding style with which few programmers are yet familiar.

"Programming in occam" is a tutorial introduction to parallel programming, based on the lecture notes that I have been using at the Programming Research Group for the past two years. It consists of three more of less distinguishable sections: an introduction to occam, a catalogue of common program fragments, and a number of complete occam programs.

The summary of the language, being only ten pages long, does not claim to stand in place of the reference manual. It describes the core of the language, and should be sufficient to make the presentation in the rest of the monograph self contained. The description is informal without imprecision, and I trust is the more readable for that.

There follows a section in which the reader is introduced to the arcana of concurrent programming: buffers and synchronizing signals, pipelines and division of labour. Like the joints of carpentry, it is as well for the uninitiated to become familiar with these early, but their value is seen in their subsequent application to larger constructions.

The largest part of the monograph describes a number of actual occam programs of different types. There is an interrupt handler, the discussion of which deals with 'real time' programming in occam. There are examples of highly parallel programs, such as might be run on special purpose multiprocessor machines, or which might be used to simulate such machines. An implementation of an adaptive Huffman coding algorithm, on the other hand, demonstrates the use of parallelism to simplify the programmer's task in a less exotic environment.

An appendix contains the complete text of each of the programs discussed in the monograph. Of course, code is not the most palatable form of literature, so these programs are necessarily compromises between realism and readability. The longest is some hundred and fifty lines long, which I trust is long enough to give a true flavour of the language in use, without being too long to be useful to the reader.

"Programming in occam" (80pp + 25pp appendix) is published by Oxford University Computing Laboratory as one of a series of monographs on topics in computation. Copies can be obtained from the Programming Research Group (Technical Monographs), 8-11 Keble Road, Oxford OX1 3QD, price £2.50.

Geraint Jones
21st May 1985

Communicating Sequential Processes - C.A.R. Hoare

(Prentice/Hall International, pp256, £27.95)

It was during the first meeting of the occam user group, in Bristol, that someone in the audience asked "Excuse me, but what is this CSP that everyone keeps mentioning?". Quite. As one of the mathematicians who was seduced into computing science by the Dijkstra & Hoare double-act, it had not occurred to me that there could be someone who had not read the 1978 Communications of the ACM article in which Tony Hoare set out his recipe for a parallel-programming language. This recipe is one whose flavour comes out in occam, which is the reason so many people at oug meetings talk about Communicating Sequential Processes.

CSP is widely known in the academic computing science community, both as a tool and as an object of research. Now at last it has become more accessible outside the research lab, with the publication of a book of the same name. In this book, Professor Hoare sets out what I might call a Grand Unified Theory of computing. Although the subject of the book is a theoretical programming language - CSP - the book is about almost all that is interesting in computing science: about programs and specifications, what it is for a program to satisfy a specification; about concurrency and communication; about synchronization and the sharing of resources; about non-determinism, divergence, deadlock, and their control.

"This is a book," claims the preface, "for the aspiring programmer; the programmer who aspires to a greater understanding and skill in the practice of an intellectually demanding profession." If I may venture to interpret this claim, I would suggest that when Tony Hoare says "programmer" he is talking of coders, designers, computer scientists, and mathematicians, and this is a book addressed to them all. It is suggested that this is a text book suitable for use in a university computing science course, and it does contain material that has been used in academic courses and in Oxford's courses for industry. This is not to deny that it is an eminently readable introduction to the theoretical background of occam and other parallel programming languages.

The first thing that strikes the reader is that the programs in this book look more like mathematical formulae than like the programs to which he is used: the author, of course, holds that programs are mathematical formulae. A reader who ventures to cross this small notational barrier will find he is rewarded. The pursuit of elegance is not an end in itself, but a means to making rigorous programming more tractable. Experience suggests that exposure to this material can cause CSP addiction, and that the addict will find CSP a useful way of organizing his thought about all sorts of computing problems.

Geraint Jones
22nd June 1985

BIBLIOGRAPHY UPDATE

compiled by Inmos and the Editor

Papers about occam and the Transputer by Inmos

M.Harrison, P.Wilson, "Transputer development using the occam Programming System," Wescon '83 Proceedings.

P.Eckelmann, "occam - Die Sprache fuer Multiprozessorsysteme," Elektronik Industrie Vol. 15, No. 4, 1984, pp 56-62.

P.Eckelmann, "occam und Transputer: Im Einsatz fuer parallele Signalverarbeitung," Elektronik Industrie Vol. 15, No. 5, 1984, pp 63,64,68.

P.Cavill, E.Milani, "Transputer Systems," Elettronica Oggi No. 10, October 1984, pp 81,82,84,86,88. (In Italian)

R.Taylor, "Signal processing with occam and the Transputer," IEE Proceedings Part F Vol. 131, No. 6, October 1984, pp 610-614.

P.Eckelmann, "Methodisches Programmieren in occam," Elektronik No. 21, Oct 19, 1984, pp 218-223.

P.Eckelmann, "Multiprozessorsysteme als eine logische Konsequenz," Markt & Technik No. 46, 1984, pp 67-73.

P.Mattos, "The transputer," IEE Colloquium on New Microprocessors (Digest No. 97) Nov 20, 1984, pp 3/1-3.

C.Mansfield, A.Baker, "Intercommunication between transputer processors," Computing the Magazine Jan 31, 1985, p 18.

D.May, "occam," Computer Bulletin Vol. 1. Part 1, March 1985, p 14.

P.Eckelmann, "Portakit," Elektronik Industrie No. 3, 1985, pp 34-41.

P.Walker, "The transputer," Byte Vol. 19, No. 5, May 1985, pp 219,220,222,224,225,227,230,232,235.

C.Whitby-Strevens, "The transputer architecture," 12th International Symposium on Computer Architecture (Boston, June 17-19, 1985)

C.Whitby-Strevens, "RISC and the I1 instruction set for the transputer," ibid.

I.Barron, Invited talk, VLSI 85 (Tokyo, Aug 26-28, 1985) IFIP.

Papers on occam and the Transputer authored outside Inmos

D.Q.M.Fay, "An implementation of the Fast Fourier Transform on the transputer," Computer Science and Informatics (India), Vol. 14, No. 2, 1984.

"Transputer - a new microcomputer building component," Vyber Inf. Organ. Vypocetni Tech. (Czechoslovakia), No. 3, 1984, pp 341-345. (In Czech)

D.Benchley, "occam's edge," PC World March 1984, pp 60-65.

F.D.Greco, "RISC, transputers and mass storage," Program. J. Vol. 2, No. 4, 1984, pp 13-16.

D.Q.M.Fay, "Experiences using Inmos proto-occam," SIGPLAN Notices Vol. 19, No. 9, Sept 1984, pp 5-11.

W.Hromoda, "Die Sprache des Transputers," Elektronikschau (Austria), No. 10, 1984, pp 48-52.

D.A.Reed, "The performance of multimicrocomputer networks supporting dynamic workloads," IEEE Transactions on Computers, Vol. C-33, No. 11, Nov 1984, pp 1045-1048.

M.Stevens, "Transputers: lego for fifth generation machines," Computer Bulletin Ser. 2, No. 42, Dec 1984, pp 21-23.

A.Cockroft, "You've never had it so fast," CCL Interface Winter 1984.

D.Q.M.Fay, "occam - a language for the 1980s and beyond," BCS (Belfast Branch) 1985 Handbook.

Fujitsu Ltd., "Prolog-based expert system for logic design," Proceedings of the Intern. Conf. on Fifth Generation Computer Systems 1985.

M.D.Harrison, "Monitoring a target network to support subsequent host simulation," Journal of Microcomputer Applications Vol. 8, No. 1, Jan 1985, pp 75-85.

H.Ebert, "Ein Transputer kommt selten allein," Elektronik Industrie No. 1, 1985, pp 80-88.

"10 Views from the top - Niklaus Wirth," Electronic Design Vol. 33, No. 1, Jan 10, 1985, pp 266,267.

T.Durham,"The transputer route to supercomputing," Computing the Magazine February 10, 1985, pp 4,5.

G.Jones, Programming in "occam", Oxford University Computing Laboratory Programming Research Group, Technical Monograph PRG-43, March 1985.

D.Fay, "Interrupts and the hardware software rendezvous," Microprocessors and Microsystems Vol. 9, No. 2, March 1985, pp 57-63.

R.Chapman, T.S.Durrani, T.Willey, "Design strategies for implementing systolic and wavefront arrays using occam," Proc. of the IEEE Int. Conf. on Acoustics, Speech and Sig. Proc. (Tampa, FL, March 1985)

R.Dettmer, "occam and the transputer," Electronics and Power Vol. 31, No. 4, April 1985, pp 283-287.

M.J.P.Bolton, D.A.Cowling, "Real-time flight simulation with transputers," 16th Annual Modeling and Simulation Conference (Pittsburgh, April 25,26, 1985)

B.Heal, "Multiprocessor solution in occam to an NP-complete problem," Microprocessors and Microsystems Vol. 9, No. 4, May 1985, pp 162-170.

M.McLean, "Breaking the trend of microprocessor evolution," Electronics Times No. 320, June 27, 1985, pp 34,35.

T.Mano, F.Maruyama, K.Hayashi, T.Kakuda, N.Kawato, T.Uehara, "occam to CMOS: experimental logic design support system," IFIP 7th International Symposium on Computer Hardware Description Languages(Tokyo, Aug 29-31, 1985)

F.Vajda, "Critical issues in the application of a transputer in a concurrent system," Euromicro 85 (Brussels, Sept 3-5, 1985)

oooOOOooo

OCCAM USER GROUP

THIRD TECHNICAL MEETING

The next technical meeting of the group will be held at the Universiy of Kent, Canterbury, from midday Monday 23rd September to afternoon Tuesday 24th September 1985. A charge of £30 per person will be made.

For details see the enclosed green sheet or contact Michael Poole or Kathy Yarnold at INMOS.

occam USER GROUP UPDATE

Members who joined or moved between 12 Dec 1984 and 7 June 1985.

G Adshead, International Computers Ltd, Wentlock Way, West Gorton, MANCHESTER, M12 5DR
Greg Aharonian, Software Manager, Rosetta Software Services, PO Box 404, Belmont, Mass, 02178, USA
John Ainscough, Dept of E&E Engineering , J Dalton Faculty of Technology, Manchester Polytechnic, Chester Street, Manchester
Simon Anthony, Smith Associates, 45-47 High Street, Cobham, Surrey, KT11 3DP
Professor Arvind, Lab for Computer Science, Masschusetts Institute of Tech, 545 Technology Square, Cambridge, Massachusetts 02139, USA
Charles Askew Esq, University of Southampton, SO9 5NH
Prof D Aspinall, Dept of Computation, UMIST, Sackville Street, MANCHESTER, M60 1QD
M Atiquzzaman, B22 Main Bldg, Dept of E.E.&E, UMIST, PO Box 88, Manchester, M60 1QD
DLA Barber, Department of Trade & Industry, Millbank Tower, Millbank, London, SW1P 4QU
Prof D.W.Barron, Dept of Computer Studies, The University, Southampton, SO9 5NH
David Bell, Department of Geography, University College London, 26 Bedford Way, LONDON, WC1
Mike Bell, CCL Design & Development, Cambridge Consultants Ltd, Science Park, Milton Road, Cambridge, CB4 4DW
Robert L Belleville, Apple Computer, 20525 Mariani Avenue, Cupertino, California 95014, USA
Dr Klaus Berkling, CASE Centre, Syracuse University, Hinds Hall, Syracuse, New York 13210, USA
Wolfgang Bernard, Sabine Bernard Electronik, Edenkobener Str. 2, 6800 Mannheim 31, West Germany
Michael Biscoe-Taylor, Smith Associates, 45-47 High Street, Cobham, Surrey, KT11 3DP
Dr T Buckley, Computer Studies Dept, Leeds University, Leeds, CS2 9JT
Dr Alan Burns, University of Bradford, Bradford, West Yorkshire, BD7 1DP
Prof RM Burstall, Dept of Artificial Intelligence, Hope Park Square, Meadow Lane, Edinburgh, EH8 9NW
Dr P C Capon, Dept of Computer Science, University of Manchester, M13 9PL
C.Y.Chan, Dept of Computer Studies, University Leeds, Leeds 2
S.K.R Chan, Dept Electrical & Electronic Eng, University College of Swansea, Swansea, SA2 8PP
Li Chang, Institute of Computing, Technology Academia, Sinica, PO Box 2704,1, Beijing, CHINA
Dr Chris Chatwin, Dept of Mechanical Engineering, James Watt Building, University of Glasgow, GLASGOW, G12 8QQ
Guy Chemla, Inmos SARL, Immeuble Monaco, 7 rue Le Corbusier, SILIC 219, 94518 Rungis Cedex, FRANCE
Peter Clayton Esq, Rhodes University, Dept Computer Science, PO Box 94, Grahamstown 6140, South Africa
Alex D'Agapeyeff, 60 Wildwood Road, LONDON, NW11

Prof E L Dagless, University of Bristol, Dept Electrical Eng., University Walk, BRISTOL, BS8 1TR

Michael Dalton, Icon Holographics, 133 Norring Hill Gate, London, W11 3LB

John Darlington, Imperial College, South Kensington, LONDON, SW7 2AZ

Prof A.C.Davies, The City University, Northampton Square, London, EC1V 0HB

Michael Davison, ARE Portsdown, XCC2.1 Room 512, Block 3, Portsdown, Portsmouth HANTS, PO6 4AA

S Vedat Demiralp, Dept of Computer Science, University of Kent, Canterbury, Kent, CT2 7NS

Bernard Donguy, EPSHOM, 13 rue du Chateflier, BP426, 29275 Brest Cedex, France

G Donnan, Univ of Ulster at Jordanstown, Dept of Computing Science, Shore Road, Newtownabbey, Co Antrim BT37 0QB, N.IRELAND

George Edge, PA Technology, PAT Centre, Back Lane, Melbourne, Royston, HERTS

Terry Ellison, SCICON Ltd, Wavendon Tower, Milton Keynes, Bucks, MK17 8LX

M.Fernando, Logica Ltd, Cobham, Surrey

Prof Nissim Francez, Technion, Israel Inst of Tech, Computer Science Department, Haifa, ISRAEL

Michael Gehret, Silcherstr. 15, D-8944 Groenenbach, West Germany

Kevin Greene, CASE Centre, Syracuse University, Hinds Hall, Syracuse, New York 13210, USA

Bengel Guenther, Brown Boveri & Cie AG, Neustradter Str. 62, D-6800 Mannheim, WEST GERMANY

Robert Harris, CORD Designs Ltd, 89 Eastworth Road, Chertsey, SURREY, KT16 8DX

Stephen Hart, Icon Holographics, 133 Notting Hill Gate, London, W11 3LB

JC Harwell Esq, Harwell Data Processing, 7 Swallow Street, LONDON

B.W.Heal, Portsmouth Polytechnic, School of Information Science, Mercantile House, Hampshire Terrace, Portsmouth, PO1 2EG

Dr A Hey, Dept of Physics, University of Southampton, Southampton, Hants, SO9 5NH

Richard Hiett, Data General Ltd, Hounslow House, 723-734 London Road, Hounslow, Middlesex, TW3 1PD

Dr O.R.Hinton, University of Kent at Canterbury, The Electronics Laboratories, The University, Canterbury, KENT, CT2 7NT

Hsu, Cho-Chun, Computer Science Dept, Peking University, Beijing, CHINA

Yohoshua Imber Ph D, Dept of Elect Eng, Faculty of Engineering, Tel Aviv University, ISRAEL

Mr Paul G Jenkins, University College Cardiff, Dept Mineral Exploitation, Newport road, Cardiff, CF2 1TA

Chris Jesshope, Dept of Electronics, & Information Eng., The University, Southampton, SO9 5NH

Dr Mike Kemp, Logica Technical Centre, Betjeman House, 104 Hills Road, Cambridge, CB2 1LQ

Falk-Dietrich Kuebler, Tannenweg 7, D-5102 Wuerselen, West Germany

Dr A.E.Lawrence, 87 Early Road, Witney, Oxon, OX8 6ET

J.W.Lockton, Micro Consultants Ltd, 31 Turnpike Road, Shaw, Newbury, Berks, RG13 2HA

Stewart McKechnie, Sigmex Ltd, Sigmex House, North Heath Lane, Horsham, W.Sussex

Andrew Mackie, CAP (Reading) Ltd, Trafalgar House, Richfield Avenue, Reading, Berks, RG1 8QA

Lt Cdr P.E.Marshall RN, Defence ADP Training, Centre, Blandford Camp, Blandford Forum, Dorset, DT11 8SP
Dr P.Martin, Thorn EMI Central Research, Laboratories, Dawley Road, Hayes, Middlesex, UB3 1HH
Dr H Mehlo, Carl Zeiss, PO Box 1369, D7082 Oberkochen, West Germany
Martin Meindl, Wurmstr.13, D-8000 Munchen 50, West Germany
P.E.Mitchell, PA Technology, Cambridge Laboratory, Melbourn, Royston, Herts, SG8 6DP
Dr W.R.Moore, Dept of Electronics and , Information Engineering, The University, Southampton, SO9 5NH
Klaus Moritzen, Schwalbenweg 12, D-8520 Erlangen , W Germany
Naoyuki Motomura, Yaskawa Electric Mfg Co Ltd, Yahata Nishi, Kitakyushu, 806 Japan
Al Mudrow Esq, CSA, 950 North University Avenue, Provo, UTAH, 84604, USA
Dr Peter Muller, Dept Photogrammetry & Surveying, University College, GowerStreet, LONDON, WC1 6BT
Dr A Mycroft, ADR, Computer Lab, Cambridge University, Corn Exchange St, Cambridge, CB2 3QG
Kei Nakada, Personal Media Corporation, 8-1-11 Nishi-gotanda, Shinagawa-ku, Tokyo, Japan
Chris Nettleton Esq, Systems Designers Scientific, Pembroke House, Pembroke Broadway, Camberley GU15 3XD
Dan Oestreicher, 3 West London Studios, Fulham Road, LONDON, SW6
Rikio Onai, Institute for New Generation, Computer Technology, Mita Kokusai Building 21F, 4-18 Mita 1-chome, Minato-ku, Tokyo 108 JAPAN
DJ Otway, GEC Hirst Research Centre, East Lane, Wembley, HA9 7PP
Dr Derek J Paddon, Dept of Electrical Engineeering, Keio University, 14-1, Hiyoshi, 3 Chome, Kohokuku, Yokohama 223, Japan
Dr Ian Page, Oxford University, Programming Research Group, 8-11 Keble Road, Oxford, OX1 3QD
Professor Dr Y Paker, Polytechnic of Central London, Fac of Engineering and Science, 115 New Cavendish Street, London, W1M 8JS
Prof David Park, Dept of Computer Science, Warwick University, Coventry, CV4 7AL
Ralph A Phraner, 516 Shrader Street, San Francisco, California 94117, USA
G.D. Plotkin, Dept of Computer Science, University of Edinburgh, JCMB, The Kings Buildings, Mayfield Road, Edinburgh EH9 3JZ
Patrick Pope, Dept of Physics, University of Southampton, Southampton, Hants, SO9 5NH
Prof IC Pyle, University of York, Heslington, YO1 5DD
Prof B Randall, Newcastle University, Claremont Tower, Claremont Road, Newcastle-on-Tyne, NE17 1RU
B Reynolds, Micro Focus, 58 Acacia Road, London, NW8 6AG
Mike Richardson, CCL Design & Development, Cambridge Consultants Ltd, Science Park, Milton Road, Cambridge, CB4 4DW
Harald Richter, MPI f. Plasmaphysik, Abt. Informatik, 8046 Garching, W Germany
Doug Robertson, Sigmex Ltd, Sigma House, North Heath Lane, Horsham, W.Sussex
Lawford Russell, CAP, 233 High Holborn, LONDON, WC1
Andy Rutter, The Instruction Set Ltd, 17-18 Bedford Square, LONDON, WC1B 3JA

Dr Patricia Samwell, Centre for Information Eng., The City University, Northampton Square, London, EC1V 0HB
Mr N.R.Saville, Algorithmic Systems, Engineering Limited, 5 Valentine Court, Kings Heath, Birmingham, B14 7AN
Heinz Schleuttler, CASE Centre, Syracuse University, Hinds Hall, Syracuse, New York 13210, USA
Abraham Seidmann Ph D, Dept of Ind Eng, Faculty of Engineering, Tel Aviv University, ISRAEL
Prof CH Sequin, Univ of California, Computer Science Div, Berkeley, California, 94720, USA
D Shorter, Systems Designers Ltd, 57 High Street, Frimley, Surrey
Thomas Sigl, Heckenrosenweg 28, 85 Nurnburg 60, WEST GERMANY
Dave Simpson, SCICON Ltd, Wavendon Tower, Milton Keynes, Bucks, MK17 8LX
R Sleep, University of East Anglia, Norwich, NR4 7TJ
Dr David J Stanley, Logica UK Ltd, Cobham Park, Downside Road, Cobham, Surrey
Dr Gardiner Stiles, CASE Centre, Syracuse University, Hinds Hall, Syracuse, New York 13210, USA
Dr John M Taylor, Hewlett Packard Limited, Filton Road, Stoke Gifford, BRISTOL
M Tedd Esq, Caemawr, Eglwysfach, Machynlleth, Powys, Wales
Mrs Shimei Tian, 2061 Cornell Road, £212, Cleveland, Ohio 44106, USA
Raglan Tribe, Info Technology Dept, British Aerospace plc, Dynamics Group, Bristol Division, SPC 267, PO Box 5 Filton, Bristol BS12 7QW
DA Turner, University of Kent, Canterbury, Kent, CP2 7NS
A.M.Tyrrell, St Peters College, The University of Aston, College Road, Saltley, Birmingham, B8 3TE
Mr I Walker, Dept Elec & Electronic Eng, University of Aston, in Birmingham, Gosta Green, Birmingham, B4 7ET
Dr P.H.Welch, Computing Laboratory, University of Kent, Canterbury, Kent, CP2 7NS
Timm Werner, DKF7 Heidelberg Inst. 08, 6900 Heidelberg, Im Nevenheimer Feld, W Germany
Stewart Wilson, Institut Laue Langevin, 156X Centre de Tri, 38042 Grenoble, Cedex, FRANCE
Bernd Wolff, Schlossstr. 30, D 5300, Bonn 1, West Germany
Dr J.C.P.Woodcock, Dept of Electronic, and Electrical Engineering, University of Surrey, Guildford, Surrey, GU2 5XH

BACK NUMBERS

Copies of Issues 1 and 2 of the occam User Group Newsletter are available while stocks last on application to the secretary at Inmos. Issue 1 included 6 pages of bibliography on CSP, occam and the Transputer. Issue 2 included a complete list of members, and a bibliography update.

---

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:

Mr Hugh Webber,
RSRE, 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)
RSRE, St Andrews Road,
GREAT MALVERN,
Worcs WR14 3PS Tel: Ø6845 2733 (x2824)

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

Mr Chris Nettleton,
System Designers Scientific,
Pembroke House, Pembroke Broadway,
CAMBERLEY, Surrey GU15 3XD Tel:Ø276 6862ØØ

Mr Simon Turner,
Plessey Electronic Systems Research Ltd,
Roke Manor, ROMSEY,
Hants SO5 ØZN Tel Ø794 515222(x2219)

Dr Peter Welch,
Computing Laboratory, The University,
CANTERBURY, Kent CP2 7NS Tel Ø227 66822 (x629)

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 an enrolment form. 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 User Group 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: Ø272 24161 (x412)

Technical 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: Ø272 29Ø861