FOREWORD

The attached paper is now an interesting historical curiosity, but little more than that. It represents an experiment in carrying out realtime programming in a high-level language (Algol68), in user space, using the 'conceptual multiplexer' facility in the GEORGE 3/4 operating systems. In this era realtime programming would often have been done in assembler so we felt a great sense of liberation in having all the expressive power of Algol68 available to us. However, we knew that our application would not be portable, not so much because Algol68 could not be ported (it could, and it was) but because the underlying GEORGE operating system was solidly locked to ICL 1900 series hardware.

Very shortly after this paper was presented a tidal wave called "UNIX" spread over the academic computer science community. It was immediately obvious that the C language (though lacking some of Algol 68's expressiveness) would be a powerful system implementation language. The fact that the UNIX kernel was small, and programmed in C, and that the UNIX system calls were designed to allow processes in user mode to provide 'system' functionality (via close communication with the kernel) all came together to make multi-threading and multi-tasking relatively straightforward to implement. Moreover, provided UNIX itself survived as an operating system and proved capable of being ported to various hardware platforms (and over the past 30 years this has certainly occurred !) then real-time processes communicating with a UNIX kernel could also, themselves, become portable.

The program attached as Appendix 1 could be transliterated to C with little difficulty and could be used to drive multiple VDUs on a multiplexer (not that there are many 'dumb' VDUs or multiplexers around any more). The program text itself is interesting in several respects: the 'triple ref' technique is used to allow a **single** pointer to traverse a list and to effect insertions and deletions from that list. This technique can be transliterated to any language such as C, or C++, which distinguishes between different levels of reference and which allows access to intermediate levels of referencing via casts. Sadly, languages such as Basic, Pascal and Java do not qualify … Moreover, it is startling to see, in a non-functional language, the ability to provide functional arguments to procedures as written-out denotations of the required function (with no need to give a name to the function if this is not needed). Connoisseurs of program languages may well find other interesting snippets.

The attached paper was presented at a conference on "Applications of Algol 68" held at the University of Liverpool in 1975. This was in an era well before PDF, Word and Desktop Publishing, so only typewritten abstracts of papers were circulated at the time. Lack of resources for typesetting the various accepted papers meant that a formal Proceedings was never published.

COLOPHON

The rebuild of this final draft started with scanning in to Readiris Pro 9 OCR software from the original typescript. The recognition accuracy on the main paper was very good but relapsed into being abysmal for the program and sample output in the Appendices (which were a photocopy of faded output from a badly-adjusted lineprinter). UNIX *troff* was used to set up the correct typeface (Courier) and to get the line and page breaks as accurate as possible. Line diagrams within the paper were re-set using the *pic* pre-processor for *troff*.

The time taken to re-set this paper was 6 hours, the great majority of which was spent in 'rescuing' the program in the Appendix.

Development of a Real Time System in Algol 68.

K. C. Mander and D. F. Brailsford
Department of Mathematics
University of Nottingham
Nottingham NG7 2RD

For various reasons, many Algol 68 compilers do not directly implement the parallel processing operations defined in the Revised Algol 68 Report. It is still possible however, to perform parallel processing, multitasking and simulation provided that the implementation permits the creation of a master routine for the coordination and initiation of processes under its control. The package described here is intended for real time applications and runs in conjunction with the Algol 68R system; it extends and develops the original Algol 68RT package, which was designed for use with multiplexers at the Royal Radar Establishment, Malvern. The facilities provided, in addition to the synchronising operations, include an interface to an ICL Communications Processor enabling the abstract processes to be realised as the interaction of several teletypes or visual display units with a real time program providing a useful service.

Introduction

The real time system to be described was conceived as a convenient interface between the Algol 68R system developed at the Royal Radar Establishment, Malvern and the ICL 7900 Series Communications processors [1]. We would like to acknowledge the assistance of RRE in the provision of certain basic software which, by way of introduction, will now be briefly described. The software consists of a special loader to load program segments in a specific order together with a small segment which enables Algol 68R procedures to be executed in parallel, each with its own separate run-time stack area and which also allows control to transfer between procedures at user-specified points; this transfer is called coordination. The software is designed on the multi-tasking primitives proposed by Dijkstra [2].

Briefly,

```
{ SIGNAL s; }

up(s) Adds 1 to s and coordinates.

down(s)  If s=0, the coordinator suspends the
    procedure, with the result that some
    other procedure must run instead. The
    suspension is cancelled as
    soon as up(s) is obeyed anywhere.
    If s>0, the procedure remains unsuspended
    and 1 is subtracted from s the moment
    it continues. A coordination takes place,
    however, and may cause continuation to be
    postponed.
```

The procedures up and down are an extension of the Dijkstra V and P operations and the procedure 'launch next call', for example

```
launch next call; process1
```

establishes the next procedure call (i.e. process1) as an independent procedure to be run in parallel with others so launched. These procedures are referred to as processes and form the basic programming unit of the RRE software.

The ICL 7900 Series Communications Processors provide a multi-access service to ICL computers and the main concern in the development of a real time system in Algol 68 was the provision of a service for ICL 7071 teletypes and 7181 visual display units (vdus) other than that provided by ICL's Driver system. On a 1900 Series machine a typical configuration might be …

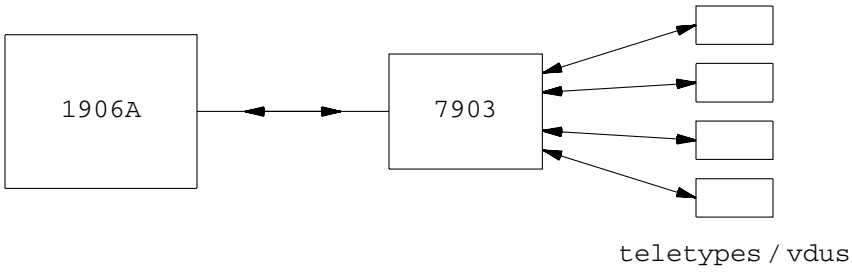


Figure 1

… with the whole of the 7900 processor available for the real time program to use. In practice this will provide far more devices than the average user would require and would also prevent the unwanted devices being used to provide any standard online service that the operating system may support. In the George 3/4 system, however, there is a facility for enabling a subset of the devices to be connected up to an in-core program providing some kind of specialist online service. The remaining devices on the communications processor continue to give the standard George 3 online service (MOP). It is the 'CONCEPTUAL' command which is used to associate a given name, for example, 'TERMINI' with a particular subset of the devices. The user can then make use of this subset without interfering with the rest of the service. Each device is associated with the conceptual by the 'ATTACH' command and the program is associated with the conceptual by the 'ONLINE' command within its job control. Thus schematically we have

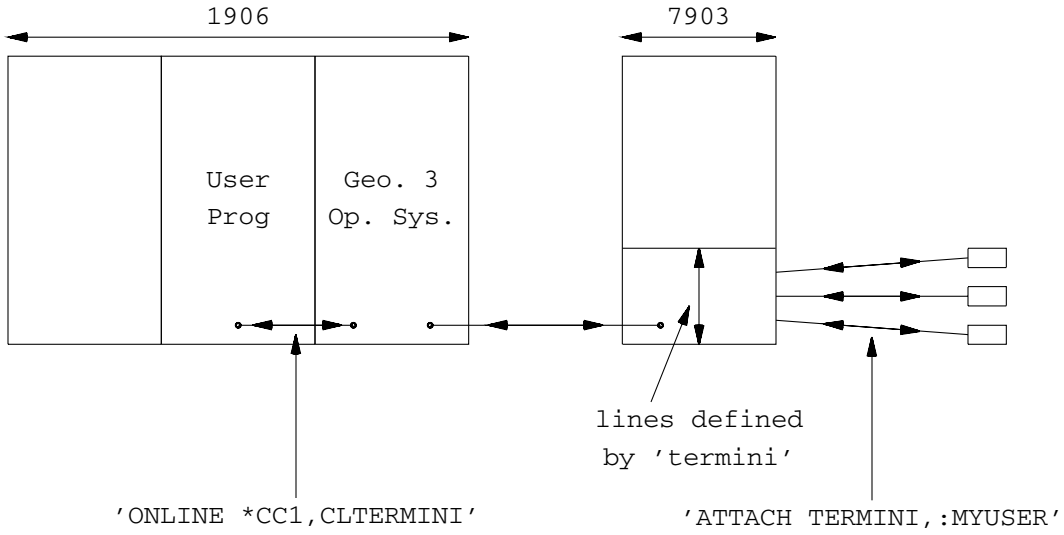


Figure 2

To each device attached there corresponds a distinct process within the user program which controls the transput to the device. Output to the devices is handled directly by the process concerned but input has to be handled by a separate process since the 7900 processor is a message buffering processor and there can be no

guarantee of the order in which the messages arrive from it.

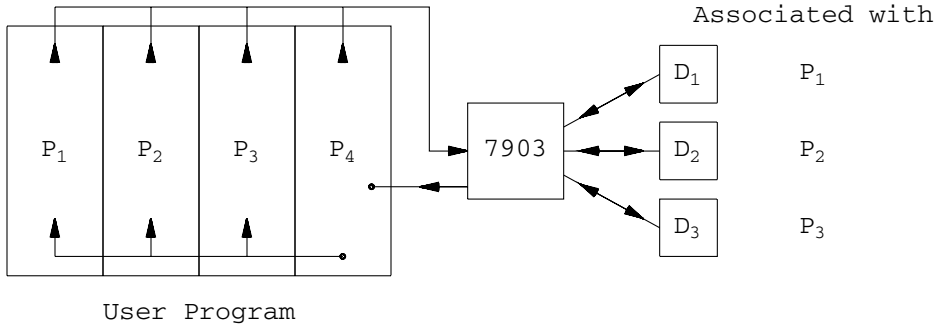


Figure 3

Having described the constraints introduced by the operating system we can describe the interface between the 7900 processor and the Algol 68R system.

There are few visible differences between the transput to file-store files and the transput to interactive devices. This is because most of the interface is hidden from the user; for example

```
outf(channel, $tl$, "hello")
```

would output the message "hello" on some device ('channel' is an appropriate CHARPUT variable). The actual transput is achieved by trapping the event of taking a newline and instead of making a transfer to a filestore file the transput is directed to the 7900 processor. Input is analogous to output, the program extract

```
INT i,j; inf(channel,$l3d,2d$,(i,j))
```

would serve to input i and j from a device. The call of newline is usually placed at the end of output formats and the beginning of input formats.

Unfortunately under the Algol 68R system formats are not re-entrant, that is to say that no two processes are allowed to execute the same format simultaneously. It is necessary, therefore, that each process expecting to use a format at the same time as some other process, should take its own copy. This can be done by having some global declaration of the format,

for example,

```
FORMAT global = $la, 2(2d), l 4(4a) $
```

and taking a copy within each process, for example,

```
FORMAT copy; copyformat(global,copy)
```

We are now able to construct device independent procedures for parallel execution. Such procedures will depend critically on transput being directed to the correct device and by declaring each procedure to have two REF CHARPUT parameters we can ensure that the correct transput is achieved. For example,

```
PROC myproc = (REF CHARPUT input,output)VOID:
BEGIN ... END
```

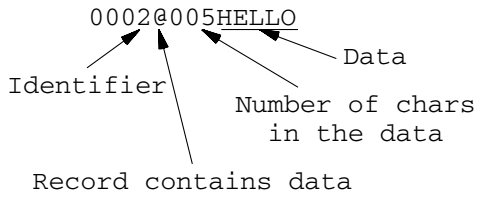
Within these procedures transput proceeds in the normal way. The following routine, for example, outputs the squares of input numbers.

```
PROC square = (REF CHARPUT input,output)VOID:
BEGIN
FORMAT fin,foutl,fout2;
copyformat($ l3v.3d $,fin);
copyformat($ tl $,foutl);
copyformat($ "the square of " 3v.3d, " is "
                6sv.6d l $,fout2);
        WHILE
        REAL r;
        outf(output,foutl,
                "type a number in the form xxx.xxx");
        inf(input,fin,r);
        outf(output,fout2,(r,r*r));
        r < 999.0
        DO SKIP
END
```

We are now able to associate procedures with devices. Each device has a unique identifier number, assigned to it by the operating system and denoting one of the lines of the conceptual multiplexer; devices are referred to by this number and it is supplied as the first parameter to the call of a procedure 'start iden', tbe second parameter being the procedure which is to be run on the device. As many devices can be started as have been previously defined and they are scanned for input by a procedure corresponding to the P_4 of Figure 3.

If the attached device is a vdu there is an extensive range of routines which can be used to provide lineprinter and backing store copies of the screen as well as facilities for clearing and racking up the screen. It is also possible to output strings of protected characters, that is, characters which cannot be overwritten by input from the vdu keyboard. Much use can be made of this facility for data collection purposes as will be seen later.

Throughout the testing of a program it could be inconvenient to attach devices and run in real time when, especially in the early stages of development, the program may contain some errors. It is possible, therefore, by setting the appropriate switch, to take messages from a filestore file instead of the communications processor, thereby obviating the need for real time. Similarly output for the processor is sent to a file. Records within the input file have to be in a format acceptable to the program i.e. as if they came from the processor and a small segment is provided to add the appropriate red-tape to messages. A typical record might be



With the exception of this internal transput, the program behaves in exactly the same way as it would if run in real time, without incurring the overheads and inconvenience which real time testing introduces. This approach has been used to develop a small illustrative application for this conference.

The application is a simple hotel reservation system. We have a small hotel with assorted double and single rooms most of which have bathrooms. We have two receptionists, one at the hotel and the second at some (possibly remote) other site. Each receptionist has a vdu into which the following commands may be input

```
ENQ       LISTALL
CANCEL    LISTFREE
DAY       LISTTAKEN
STOP
```

Depending on the command typed, different frames can be displayed on the screen and input requested. For example

DAY 15 sets the current day to 15. This also causes the removal of all old bookings i.e. those ookings which were only up to day 14 or efore

ENQ Causes the following frame to be output to the vdu

```
NUMBER OF SINGLE ROOMS                                  (  )
NUMBER OF DOUBLE ROOMS                                  (  )
BATHROOMSWITH EACH ROOM?                                (  )  Y = YES
FROM DAY (  ) TO DAY (  )
```

Items of data can then be input by either receptionist on behalf of a client in the spaces provided

```
NUMBER OF SINGLE ROOMS                                  (00)
NUMBER OF DOUBLE ROOMS                                  (01  )
BATHROOMSWITH EACH ROOM?                                (N)  Y = YES
FROM DAY (21) TO DAY (24)
```

LISTALL lists the bookings for each room, for example

ROOM	S/D	BATH	BEDS	OCCUPANCY FROM DAY 2
1	DOUBLE	WITH	0 1	V FROM DAY 2 TO 30
2	SINGLE	WITHOUT	1 0	V FROM DAY 4 TO 8
				T FROM DAY 9 TO 11

The program is written as one might write any other program with the addition of special transput procedure calls and the following calls which set up the individual processes.

```
define configuration(2,1);
vdu ( (1, 2) ) ;
start configuration;
start iden(1,receptionist);
start iden(2,receptionist);
scan configuration
```

The last three calls are of greatest interest, the two calls of 'start iden' create two processes each obeying the procedure 'receptionist' i.e. two separate areas of the run-time stack are set apart as work space for the procedures and certain information (for example the contents of registers, stack pointers etc.) is kept to enable the process to be restarted should control pass from it. The task of the procedure 'scan configuration' is to read messages from the communications processor; the messages read are of two types

(i) those of a supervisory nature i.e. containing information about the physical state of the system (for example, a device being inoperable) and

(ii) those containing data for the various processes within the program. Most messages are of the latter type and are passed to the appropriate process. 'Scan configuration' continues reading until both calls of the procedure receptionist have terminated (by both receptionists typing STOP).

A copy of the complete program follows at the end of this paper together with part of the hard copy output produced by the program.

The great advantage of the system is the ability to program real time applications in a high-level language; Algol 68 permits a freedom of expression which is impossible in an assembly language. This is, however, coupled with the penalty of a larger program and a slightly slower speed of execution. On a smaller computer (for example, one with less than 128K of core store) the program size may be of great importance and possible future developments include the provision of a system of overlays and the option of making one process control more than one device, hopefully reducing the overall size of the program.

We estimate that if the application presented were rewritten using ICL Driver with overlays it would occupy about 17K of core store (that being the size of the largest overlay) and would have taken 50 man-days to write; this compares with 30K and only 10 man-days in Algol 68.

References

1. ICL Data Communications and Interrogation Manual. ICL number 4328.

E. W. DIJKSTRA : Cooperating Sequential Processes In 'Programming Languages' edited by F. Genuys.

Acknowledgements

The authors would like to acknowledge the assistance of the Cripps Computing Centre at the University of Nottingham and Dr. D.P. Jenkins and Dr. A.J. Fox of the Roya1 Radar Establishment Malvern in the preparation of this paper. Support from the Science Research Council for K.C. Mander is also gratefully acknowledged.

APPENDIX 1—PROGRAM LISTING

```
HOTEL 'WITH' RTMATHSEG, VDUBASICSEG, 'FROM' RTLIB
'BEGIN'
'MODE' 'DATE' = 'STRUCT' ('INT' FROM, TO),
        'ENTRY' = 'STRUCT' ('INT' NUMBER, FROM, TO),
        'ROOM' = 'STRUCT' ('BOOL' SINGLE, BATH, 'INT' SBEDS, DBEDS,
                           'REF' 'BOOKING' B),
        'BOOKING' = 'STRUCT' ('BOOL' PENDING, TAKEN, 'DATES' DATE,
                             'REF' 'BOOKING' NEXT);
'REF' 'BOOKING' NOTBOOKED = 'NIL';
'INT' NROOMS;
   INF (STANDIN, $2D$, NROOMS);
[1:NROOMS] 'ROOM' ROOMS;
  FORMAT(STANDIN,$1B,B,D,D$);
  'FOR' I 'TO' NROOMS 'DO'
  'BEGIN'
        'REF' 'ROOM' R = ROOMS[I];
        IN (STANDIN,
           (SINGLE, 'OF' R, BATH 'OF' R, SBEDS 'OF' R, DBEDS 'OF' R));
        B 'OF' R := NOTBOOKED
        'END'

'MODE' 'ENQUIRY' = 'STRUCT' ('INT'SING, DOUB, 'BOOL' BATH);
'INT' DAY := 1, LASTDAY := 30;

'FORMAT' COMMANDLIST = $LC("ENQ", "LISTFREE", "LISTALL",
          "LISTTAKEN", "CANCEL", "DAY", "STOP", ""$,
        ROOMLIST = $1K 3V,2X B("SINGLE", "DOUBLE"), 2X
        B("WITH", "WITHOUT"), 22K 2(2X 2V)$,
        BOOKINGLIST = $ 35K A, 2X "FROM" 2X 3V, " TO " 3VL $,
        F1 = $2L10KT50K $,
        F2 = $ 3X T 2L $,
        F3 = $ L2D,2D,A,2D,2D $,
        F4 = $ 2L 10K "ROOM NUMBER " 3SV, " IS AVAILABLE FROM "
          3SV, " TO "3SV2L10K "SATISFACTORY ? "$,
        F5 = $2L 10K T 3X$,
        F6 = $ LA $,
        F7 = $2(2D)$,
        F8 = $ 2L T 2L $,
        F9 = $L "ROOM"2X"S/D"4X"BATH"25K"BEDS"35K
           "OCCUPANCY FROM DAY "3SVL$,
        F10 = $L3SV " OUT OF ", 3SV " ROOMS AVAILABLE"L$,
        F11 = $ 10K T X $,
        F12 = $ 2X T 2X $,
        F13 = $1K N(35)" " SAL$;
```

```
'PROC' LISTROOMS = ('PROC'('REF' 'BOOKING') 'BOOL' PB,
                    'REF' 'CHARPUT' OUTC):
'BEGIN'
  'C' LISTS THE ROOMS WITH PROPERTIES
        SPECIFIED BY PB
  'C'
        'FORMAT' RL, BL, HL, G1;
        COPYFORMAT(ROOMLIST,RL); COPYFORMAT(BOOKINGLIST,BL);
        COPYFORMAT(F9,HL); COPYFORMAT(F13,G1);
        STARTSCREEN(OUTC); OUTF(OUTC,HL,DAY);
        'FOR' I 'TO' NROOMS 'DO'
        'BEGIN'
          'REF''ROOM' R = ROOMS[I]; 'REF' 'BOOKING' RB := B 'OF' R;
          OUTF(OUTC,
                CLEARFORMAT(RL),
                (I,SINGLE'OF'R,BATH'OF'R
                SBEDS'OF'R,DBEDS'OF'R));
         'IF' RB 'IS' NOTBOOKED
         'THEN' OUTF(OUTC, CLEARFORMAT(BL),
                ("V",DAY,LASTDAY))
         'ELSE'
                'WHILE' RB 'ISNT' NOTBOOKED 'DO'
                'BEGIN'
                'IF'PB(RB)
                'THEN' OUTF(OUTC,CLEARFORMAT(BL),
                           ('IF' PENDING'OF'RB
                            'THEN' "P"
                            'ELSF' TAKEN'OF'RB
                            'THEN' "T"
                            'ELSE' "V"
                            'FI',
                            FROM'OF'DATE'OF'RB,TO'OF'DATE 'OF' RB))
                'ELSE' OUTF(OUTC,G1," ")
                'FI'
                RB := NEXT 'OF' RB
                'END'
        'FI';
        ( I=NROOMS''2
        | 'C' PRINT THE SCREEN ON LINEPRINTER
                AND CLEAR SCREEN
          'C'
          PRINTSCREEN(CURRENTVDU(OUTC));
```

```
          STARTSCREEN(OUTC);
          OUTF(OUTC,HL,DAY)
         )
        'END';
        PRINTSCREEN(CURRENTVDU(OUTC))
'END';

'PROC' INTTOCHARS = ('REF''VDU' ME,'INT' I, LINE, COL, NUMBER):
'BEGIN' 'INT' COPY := I;
        'FOR' J 'FROM' COL 'BY' -1 'TO' COL-NUMBER+1 'DO'
        (SCREEN 'OF' ME)[LINE,J] := 'REPR' (COPY'/:='10)
'END';

'PROC' FRAME1 = ('REF' 'CHARPUT' INC, OUTC, 'REF' 'ENQUIRY' ENQ,
                        'REF' 'ENTRY' ENT):
'BEGIN'
   'C' DISPLAYS THE FIRST "ENQUIRY" FRAME
   'C'
        'REF' 'VDU' ME = CURRENTVDU(OUTC);
        'CHAR' C; 'FORMAT' G1, G2, G3, G4, G5;
        COPYFORMAT(F1,G1);
        COPYFORMAT(F2,G2);
        COPYFORMAT(F3,G3);
        COPYFORMAT(F11,G4);
        COPYFORMAT(F12,G5);
        STARTSCREEN(OUTC);
        OUTF(OUTC,G1,"NUMBER OF SINGLE ROOMS");
   'C' SPECIFY A PLACE ON THE SCREEN FOR INPUT
   'C'
        INPUT FIELD(OUTC, PB, 2, 'FALSE);
        OUTF(OUTC,G1,"NUMBER OF DOUBLE ROOMS");
        INPUT FIELD(OUTC, PB, 2, 'FALSE);
        OUTF(OUTC,G1,"BATHROOMS WITH EACH ROOM?");
        INPUT FIELD(OUTC, PB, 1, 'FALSE);
        OUTF(OUTC, G2, "Y = YES");
        OUTF(OUTC, G4, "FROM DAY");
        INPUT FIELD(OUTC, PB, 2, 'FALSE);
        OUTF(OUTC, G5, "TO");
        INPUT FIELD(OUTC, PB, 2, 'FALSE);
   'C' POSITION CURSOR TO START OF SCREEN
   'C'
        SETCURSOR(OUTC,0, 0);
        INF(INC, G3,
           (SING 'OF' ENQ, DOUB 'OF' ENQ, C,
           FROM 'OF' ENT, TO 'OF' ENT));
```

```
        INTTOCHARS(ME, SING 'OF' ENQ,3,53,2);
        INTTOCHARS(ME, DOUB 'OF' ENQ,4,53,2);
        INTTOCHARS(ME, FROM 'OF' ENT,6,22,2);
        INTTOCHARS(ME, TO 'OF' ENT,6,34,2);
        BATH 'OF' ENQ := C=Y; (SCREEN 'OF' ME)[3,52]:=C;
        PRINTSCREEN(ME)
'END'

'PROC' FRAME2 = ('REF''CHARPUT' INC, OUTC, 'REF' 'DATES' D,
                        'REF' 'ENTRY' E) 'BOOL':
'BEGIN'
   'C' DISPLAYS THE SECOND "ENQUIRY" FRAME
   'C'
        'CHAR' C; 'FORMAT' G1, G2, G3, G4; 'BOOL' B;
        'REF' 'VDU' ME = CURRENTVDU(OUTC);
        COPYFORMAT(F4,G1);
        COPYFORMAT(F5,G2);
        COPYFORMAT(F8,G3);
        COPYFORMAT(F6,G4);
        STARTSCREEN(OUTC);
        OUTF(OUTC,G1,E);
        INPUT FIELD(OUTC, PB, 1, "FALSE");
        OUTF(OUTC,G2,"IF ""Y"" INSERT DATES");
        INPUT FIELD(OUTC, PB, 2, "FALSE");
        SETCURSOR(OUTC,0,0);
        INF(INC,G3,C); (SCREEN 'OF' ME)[4,28]:=C;
        (B := C = "Y"
        ! INF(INC,G4, D)
          INTTOCHARS(ME,FROM'OF'D, 5,35,2);
          INTTOCHARS(ME, TO 'OF'D, 5,45,2);
          FROM 'OF' E := FROM 'OF' D; TO 'OF' E := TO 'OF' D;
          PRINTSCREEN(ME)
        );
        B
'END'
```

```
'PROC' GARBAGE COLLECT = 'VOID':
'BEGIN'
   'C' DELETE ALL OLD BOOKINGS
   'C'
        'FOR' I 'TO' NROOMS 'DO'
        'BEGIN' 'REF' 'REF' 'BOOKING' RB = B 'OF' ROOMS[I];
                'IF' (RB 'ISNT' NOTBOOKED
                     ! DAY > TO 'OF' DATE 'OF' RB
                     ! 'FALSE'
                     )
                'THEN' RB := NEXT 'OF' RB
                'FI'
        'END'
'END';

'PROC' BOOKROOM = ('REF' 'ENTRY' ENT, 'BOOL' BOOK,
                        'REF' 'CHARPUT' OUTC):
'BEGIN'
   'C'  CONFIRMS BOOKING
        TRACER USES "TRIPLE REF" TECHNIQUE AND IS USED TO LOCATE
        THE CORRECT BOOKING IN THE CHAIN
   'C'
        'FORMAT' ERROR; COPYFORMAT(F8,ERROR);
        'BOOL' NOTMATCHED := 'TRUE';
        'REF' 'REF' 'BOOKING' TRACER := B 'OF' ROOMS[NUMBER 'OF' ENT];
        'WHILE'
        ( TRACER 'ISNT' NOTBOOKED
        ! NOTMATCHED := FROM 'OF' DATE 'OF' TRACER # FROM 'OF' ENT
        ! 'FALSE'
        )
        'DO' TRACER := NEXT 'OF' TRACER;
        'IF' (TRACER 'IS' NOTBOOKED)
            'OR' NOTMATCHED
        'THEN' OUTF(OUTC,ERROR,"BOOKING NOT FOUND")
        'ELSF' BOOK
        'THEN'  TAKEN 'OF' TRACER := 'TRUE';
                PENDING 'OF' TRACER := 'FALSE'
                DATE 'OF' TRACER := (FROM 'OF' ENT, TO 'OF' ENT)
        'ELSE'  'REF' 'REF' 'BOOKING' 'VAL' TRACER := NEXT 'OF' TRACER
        'FI'
'END'
```

```
'PROC' CANCEL ROOM = ('REF' 'CHARPUT' INC, OUTC):
'BEGIN'
        'ENTRY' ENT; 'FORMAT' G1, G2, G3;
        COPYFORMAT(F11,G1);
        COPYFORMAT(F12,G2);
        COPYFORMAT(F6,G3);
        OUTF(OUTC, G1, "ROOM NUMBER 2);
        INPUT FIELD(OUTC, PB, 2, 'FALSE');
        OUTF(OUTC, G3, "   FROM   ");
        INPUT FIELD(OUTC, PB, 2, 'FALSE');
        SET CURSOR(OUTC, 0, 0);
        NEWLINE(INC); INF(INC,G3,(NUMBER 'OF' ENT, FROM 'OF'ENT));
        BOOKROOM(ENT, 'FALSE', OUTC)
'END';

'PROC' SEEKROOM = ('PROC' ('REF' ROOM) 'BOOL' PB) 'INT':
'BEGIN'
   'C'  SEARCHES FOR A ROOM WITH PROPERTIES
        SPECIFIED BY PB
   'C'
        'BOOL' STOP := 'FALSE'; 'INT' RESULT := 0;
        'FOR' I 'TO' NROOMS 'WHILE' 'NOT' STOP 'DO'
        'BEGIN'
                'IF' STOP := PB(ROOMS[I])
                'THEN' RESULT := I
                'FI'
        'END';
RESULT
'END';
```

```
'PROC' ROOMFREE = ('REF' 'ROOM' R, 'REF' 'ENTRY' ENT) 'BOOL':
'BEGIN'
   'C' SEARCHES FOR A VACANCY IN A PARTICULAR ROOMS BOOKINGS
   'C'
        'REF' 'REF' 'BOOKING' TRACER := B 'OF' R;
        'BOOL' OK := TRACER 'IS' NOTBOOKED; 'BOOL' B:= OK;
        'WHILE'
        'IF' 'NOT' OK
        'THEN' TRACER 'ISNT' NOTBOOKED
        'ELSE' ('REF' 'REF' 'BOOKING' 'VAL' TRACER) :=
                'BOOKING' :=
                ('TRUE', 'FALSE', (FROM 'OF' ENT,
                ( B 'OF' R 'IS' NOTBOOKED ! LASTDAY | TO 'OF' ENT)),
                TRACER);
                ENT := (0, FROM 'OF' ENT,
                        (B ! LASTDAY ! FROM 'OF' DATE 'OF' TRACER));
                'FALSE'
        'FI'
        'DO' 'BEGIN'
                OK := TO 'OF' DATE 'OF' TRACER < FROM 'OF' ENT 'AND'
                        'IF' B := NEXT 'OF' TRACER 'IS' NOTBOOKED
                        'THEN' 'TRUE'
                        'ELSE' FROM 'OF' DATE 'OF' NEXT 'OF' TRACER > TO 'OF' ENT
                        'FI';
                TRACER := NEXT 'OF' TRACER
             'END';
OK
'END';

'PROC' FINDROOM = ('REF' 'ENTRY' ENT, 'BOOL' BATH, SINGLE) 'INT':
'BEGIN'
   'C' SEARCHES FOR A ROOM OF A PARTICULAR TYPE
       IF ONE IS NOT FOUND THE CONDITIONS BECOME
       PROGESSIVELY LESS STRINGENT
   'C'
        'INT' RESULT := 0;
        'IF' ( RESULT := SEEKROOM ( ('REF' 'ROOM' R) 'BOOL':
                                        'BEGIN'
                                        (SINGLE 'OF' R = SINGLE) 'AND'
                                        (BATH 'OF' R = BATH) 'AND'
                                        ROOMFREE(R,ENT)
                                        'END' )) = 0
        'THEN'
```

```
        'IF' ( RESULT := SEEKROOM(('REF' 'ROOM' R) 'BOOL':
                        'BEGIN'
                        (SINGLE 'OF' R = SINGLE) 'AND'
                        ROOMFREE(R, ENT)
                        'END' )) = 0
        'THEN' RESULT := SEEKROOM ('REF' 'ROOM' R) 'BOOL':
                                'BEGIN'
                                ROOMFREE(R, ENT)
                                'END')
        'FI'
        'FI'
RESULT
'END'

'PROC' PROCESSENQ = ('ENQUIRY' ENQ, 'REF' 'BOOL' OK,
                        'REF' 'ENTRY' ENT,
                        'REF' 'CHARPUT' INC. OUTC):
'BEGIN'
        'INT' U = SING 'OF' ENQ + DOUB 'OF' ENQ; 'INT' VALID ENTRIES := 0;
        [1:U] 'ENTRY' ENTRIES;
        'FORMAT' G1; COPYFORMAT(F10, G1);
        'TO' SING 'OF' ENQ 'DO'
        'BEGIN'
        'INT' I = FINDROOM(ENT, BATH 'OF' ENQ, 'TRUE');
        (I>0
        ! ENTRIES[VALIDENTRIES 'PLUS' 1] :=
          (I, FROM 'OF' ENT, TO 'OF' ENT)
        )
        'END'
        OUTF(OUTC, G1, (VALIDENTRIES, U));
        'FOR' I 'TO' VALIDENTRIES 'DO'
        'BEGIN'
                'REF' 'ENTRY' E = ENTRIES[I];
                'DATES' DA := (FROM 'OF' E, TO 'OF' E);
                BOOKROOM(E, FRAME2(INC,OUTC,DA,E), OUTC)
        'END'
'END';
```

```
'PROC' RECEPTIONIST = ('REF' 'CHARPUT' INC, OUTC):
'BEGIN' 'FORMAT' ROOM, ERROR;
   'C' THIS IS THE PROCEDURE TO BE LAUNCHED AS A
        SEPARATE PROCESS
   'C'
        COPYFORMAT(F8,ERROR); COPYFORMAT(COMMANDLIST, COMM);
        'INT' DUMMY; 'ENQUIRY' ENQ; 'BOOL' OK;
        'ENTRY' ENT; 'DATES' DA;
        'WHILE'
                STRATSCREEN(OUTC);
                INF(INC, COMM, DUMMY);
                'CASE' DUMMY
                'IN'
                FRAME1(INC, OUTC, ENQ, ENT);
                PROCESSENQ(ENQ, OK, ENT, INC, OUTC),

                LISTROOMS(('REF' 'BOOKING' B) 'BOOL':
                           ('NOT' (PENDING 'OF' B 'OR' TAKEN 'OF' B)),
                          OUTC),

                LISTROOMS(('REF' 'BOOKING' B) 'BOOL': ('TRUE'),OUTC),

                LISTROOMS(('REF' 'BOOKING' B) 'BOOL': (TAKEN 'OF' B), OUTC),

                CANCEL ROOM(INC, OUTC),

                INF(INC, $2D$, DAY);
                GARBAGE COLLECT,

                'SKIP'

                'OUT'

                OUTF(OUTC, ERROR, "REQUEST NOT RECOGNISED")

                'ESAC';
        DUMMY # 7
        'DO' 'SKIP'
'END'

        CANCEL ABANDON;
```

```
'C' DEFINE 2 DEVICES ON CHANNEL 0 ... 'C'
        DEFINE CONFIGURATION(2,0);

'C' ... BOTH ARE VDUS 'C'
        VDU((1,2));
'C' START THE WHOLE SYSTEM ... 'C'
        STARTCONFIGURATION;

'C' ... AND EACH DEVICE. 'C'
        STARTIDEN(1, RECEPTIONIST);
        STARTIDEN(2, RECEPTIONIST);

'C' SCAN FOR INPUT FROM THE DEVICES 'C'

        SCAN CONFIGURATION
'END'
'FINISH'
```

APPENDIX 2—SAMPLE OUTPUT

```
TERMINAL NUMBER  1              DATE: 02/04/75          TIME: 15.56.37

                NUMBER OF SINGLE ROOMS          (00)

                NUMBER OF DOUBLE ROOMS          (01)

                BATHROOMS WITH EACH ROOM?       (N)     Y = YES

                FROM DAY  (21)  TO  (24)
```

TERMINAL NUMBER 1 DATE: 02/04/75 TIME: 15.56.37

ROOM NUMBER 2 IS AVAILABLE FROM 21 TO 30

SATISFACTORY ? (Y)

IF "Y" INSERT DATES (21) TO (24)

TERMINAL NUMBER 1 DATE: 02/04/75 TIME: 15.57.26

ROOM	S/D	BATH	BEDS		OCCUPANCY FROM DAY 10				
1	DOUBLE	WITH	0	1	T	FROM	12	TO	15
2	DOUBLE	WITHOUT	0	1	T	FROM	21	TO	24
					P	FROM	25	TO	28
3	SINGLE	WITH	1	0	T	FROM	3	TO	10
4	SINGLE	WITH	1	0	T	FROM	3	TO	12
5	SINGLE	WITH	1	0	T	FROM	9	TO	11

TERMINAL NUMBER 1 DATE: 02/04/75 TIME: 15.57.27

ROOM	S/D	BATH	BEDS		OCCUPANCY	FROM	DAY	10	TO	30
6	SINGLE	WITH	1	0	V	FROM		10	TO	30
7	SINGLE	WITH	1	0	V	FROM		10	TO	30
8	SINGLE	WITH	1	0	V	FROM		10	TO	30
9	SINGLE	WITH	1	0	V	FROM		10	TO	30
10	SINGLE	WITH	1	0	V	FROM		10	TO	30