CROSSTALK Journal of Hewlett-Packard Technical Computer User Groups
MAR./APRIL 1985

HP General Manager to speak

Daring to expose himself to critical questioning by desktop computer users, Mr Bruce Marsh, General Manager at Hewlett-Packard Australia P/L, has agreed to be guest speaker at their next meeting.

This Annual General Meeting of the H.P.D.C.U.G.V. will be held at 4 pm. sharp on Tuesday, 16th April, 1985 at HP, Blackburn (NOT CSIRO, Syndal as normal) and should be one not to miss. ALL ARE WELCOME, members and others alike.

The retiring committee will take advantage of the opportunity to hold a swift election of office-bearers for 1985/86. Nominations are invited and will be received any time up to the election.

Some regular meeting features will be included, such as a "HELP" session, where problems (programming or equipment-oriented) may be raised and assistance organised, and presentation of technical brief technical offerings ("SNIPPETS").

I hope to see you there. Refreshments will be offered. If you have any queries or nominations, please contact me on (03) 859 6643.

— Chris Simpson
(President, HPDCUGV)

Co-ordinator's Comments

Welcome to the March/April 1985 issue of Crosstalk. So far the year has started off a little slow after the Christmas break, but as you can see things are hotting up with this issue roughly on time.

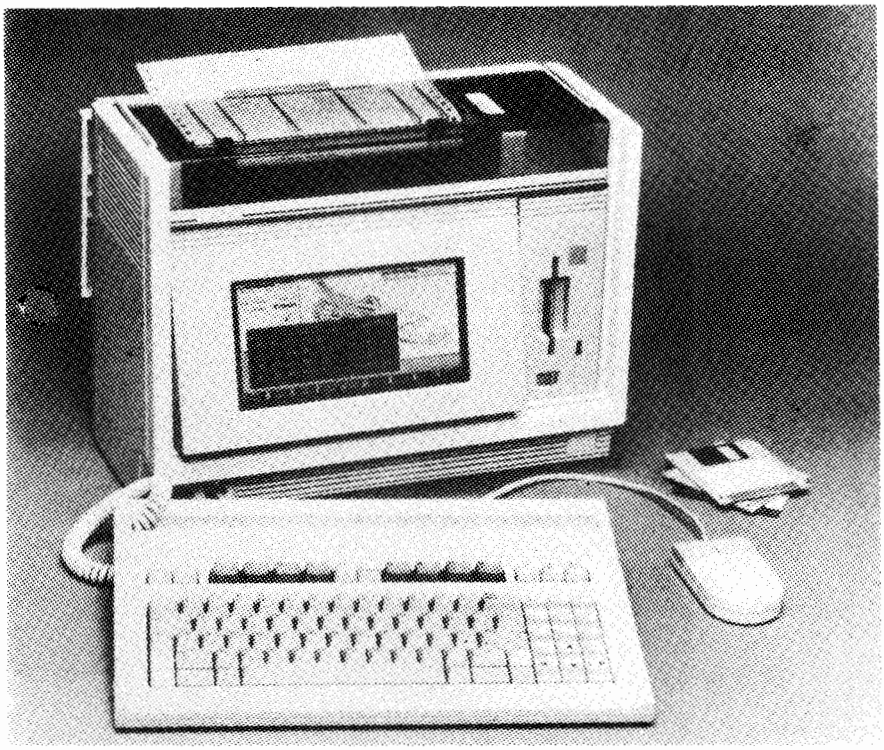
It is the commitment to get Crosstalk out every two months which, I hope will encourage readers and members to contribute. Without the contributions Crosstalk will be rather thin.

The editorial committee seems to be working well and I am sure that the members will support the committee in finding material.

Please send contributions to Norm Kay, Chris Simpson or myself. These will be in print as soon as possible. The deadline for the May/June issue is the first week of April. The deadline for the July/August issue is the first week of June.

I expect this year to go well for Crosstalk and that the knowledge shared and distributed will help all users to better understand, use and enjoy their systems.

John Green,
HP Melbourne



CONTENTS

HP Computer Museum
www.hpmuseum.net

For research and education purposes only.

Introducing the HP Integral PC

(Dick Siegel/PCD)

The power of a UNIX operating system in a personal package

The UNIX operating system makes the Integral PC a true multitasking computer. It can run numerous applications at the same time. It can do multiple processes — like printing files or monitoring instruments — while the user does other computer-aided work such as spreadsheet analysis. And to keep all of this activity organized, the user can view the different applications through the Integral PC's multiple windowing implementation, called HP Windows. The Integral PC comes with a remarkable 512K bytes of memory, expandable up to 1.5M bytes in the standard transportable configuration. With dual bus expanders, the RAM can be boosted to 5.5M bytes. A Motorola 68000 (16/32 bit) 8Mhz microprocessor makes this a very powerful application engine. In addition, an HP-proprietary 16-bit GPU drives the 255 × 512 bit-mapped flat-panel Electroluminescent (EL) display to provide outstanding graphics capability.

The power of the UNIX operating system made easy

Built into the Integral PC is 256K bytes of ROM, which includes the kernel of the UNIX operating system, a friendly user interface (Personal Applications Manager), and the window manager (HP Windows). These are all available within seconds after the machine is switched on. The Integral PC's Personal Applications Manager (PAM) is similar to the PAM that runs on the HP Touchscreen and The Portable personal computers. In the case of the Integral PC, PAM has been customized for the multitasking environment of HP-UX 2.1. Both novices and experts will appreciate the way that PAM simplifies the use of this powerful operating system.

The Integral PC also comes with a helpful tutorial disc and an easy-to-understand manual to further enhance the easy-to-use aspects of the Integral PC.

All together now: transportable and integrated

The Integral PC is a complete, ready-to-go transportable personal computer. Everything you need for full computing power including a built-in ThinkJet printer, a 90-key full-travel keyboard with numeric pad, a 9-inch diagonal 80 × 24-default window EL display, a single 3½-inch double-sided, double-density disc drive (710K bytes formatted), an HP-IB interface, two HP-HIL (Human Interface Link) front ports for the keyboard and the optional HP Mouse, and two plug-in ports for extra memory and optional interfaces, such as RS-232, Current Loop, GPIO, BCD and HP-IL (HP-Interface Loop). Included with the system is a set of six software discs, including a tutorial disc, utilities disc, HP-UX commands disc, standard applications disc, diagnostic disc, and a system disc.

Personal productivity software

The Integral PC will have a number of personal productivity software packages available within the introduction time frame, including: Multiplan®, a leading spreadsheet analysis tool; MemoMaker, a friendly easy-to-use memo and report writer; Picture Perfect, a business graphics presentation tool; Unify, a relational data base manager; Datacom, a character-mode terminal emulator with file transfer; VT/E, a VT100 full-terminal emulator; Personal Card File, a Rolodex-like file-card manager; and many other packages from both the MS™-DOS and UNIX software universe.

Disc drives

A large number of disc drives is fully supported by the HP Integral PC. These include essentially all of the HP-IB disc drives currently in production. Included are both single- and double-sided disc drives, 3½- or 5¼-inch floppies, as well as the new HP 7941 and 7945 Winchester discs. The full support of these disc drives guarantees a full range of choices in providing the most functional configuration for any use to which the HP Integral PC may be put. The recommended disc drives are: HP 9122S or D to provide an additional one or two channels of 3½-inch microfloppy capability, or an HP 9133D to provide 15M bytes of Winchester capability and an additional 3½-inch drive. The HP 9133D or 9134D hard disc is recommended for HP-UX system development, where the additional storage capacity and multi-volume capability are useful.

Printers

A number of printers are supported for use with the HP Integral PC, but the built-in HP ThinkJet offers such terrific performance, including screen-dump graphics, that only occasionally may an external printer be needed. The HP LaserJet printer will probably be the most used because of its silent, high-speed, letter-quality output. Other printers include the letter-quality HP 2602A daisy wheel and the high speed dot-matrix HP 2932 and 2634 printers.

Graphics plotters

The HP 7470 and 7475 plotters, as well as the new HP 7550 are fully supported by the HP Integral PC.

HP 9845B, 9878A, and 9825B/T now obsolete

(Cathleen Hendershot/FSD)

Due to low customer demand and the increasing popularity of the current HP 9000 Series 200 and 500 products, Fort Collins Systems Division announces the obsolescence of the HP 9845B, 9878A I/O Expander, and 9825B/T. These products will be removed from the Corporate Price List on May 1, 1985.

US orders for these products will be accepted until May 31, 1985. International orders will be accepted until July 31, 1985. Final shipments of these products will be on September 15, 1985. Support life will be through September 15, 1995.

Application software for the HP Integral PC

(David Bufford/PSD)

The HP Integral PC provides software compatibility with other computers running version 2.1 or 2.2 of the HP-UX operating system. Currently this includes the HP 9000 Models 217, 220, and 236. The following software is available from HP for these computers:

P/N	Product
45413G 45413H	Calculator (Multi-user)
45420G 45420H	MemoMaker (Multi-user)
45473G 45473H	Multiplan (Multi-user)
45524G 45524H	MicroTrak (Multi-user)

In addition, the following products are available for the HP Integral PC only:

P/N	Product
82815J	Datacom
82860J	HP-UX BASIC
82857J	HP-UX C

The HP Integral PC – a powerful multitasking controller

(Bill Hodges/PCD)

The HP Integral Personal Computer upholds the HP Series 80 and Series 200 heritage by supporting both HP and non-HP instruments. As a low-cost instrument controller, the HP Integral PC is richly endowed with I/O capabilities. Standard features such as a built-in HP-IB (IEEE 488) interface bus (allows up to 14 peripherals to be connected on the same bus) and two expansion ports for additional/optional memory and interfaces enhance the value of the HP Integral PC in laboratory and control-engineering environments.

An optional bus expander will be available to provide an additional five expansion ports for extra interfaces, memory, or modems.

In order to continue to support a substantial installed base of HP Series 80 customers, the HP Integral PC will support I/O under HP-UX Technical BASIC. This BASIC is largely compatible with HP Series 80 BASIC and contains all of the I/O drivers included in the HP Series 80 I/O ROM including: HP-IB, Serial, GPIO, BCD, and HP-IL. Under the HP-UX "C" programming compiler, such I/O enhancements as Real Time Extensions and the Device Independent Library (DIL) have also been added to support the HP Integral PC as a data collector and controller. The Real Time Extensions include: real time priority, reliable software

NEW PRODUCTS

signals, shared memory, memory lock, sync file system, interval timer, and time of day. Because the Integral PC's HP-UX operating system allows multitasking, real time priority allows the processes to schedule themselves at a real time priority without interruption from other lower priority processes. For example, data collection can be set at a higher priority level than plotting a chart (another process) or even calculating a large Multiplan® spreadsheet. Under HP-UX Technical BASIC, real time priority can be used by calling up routines written in "C". The DIL again is aimed at giving users the ability to programmatically control their instruments in conjunction with HP-UX using "C". The current set of DIL I/O libraries to be supported are HP-IB, Serial, GPIO, and BCD device-control interfaces.

Languages and tools available for HP Touchscreen personal computer

(David Newcomer/PSD)

With the announcement of the Lattice C Compiler on January 1 and the recent announcement of several software tools for the HP Touchscreen personal computer, HP now offers a wide selection of languages and programmer's tools. This is our current offering:

P/N	Product	Description
45445D	BASIC by Microsoft	Series 100/BASIC interpreter
45446D	Compiled BASIC by Microsoft	Compiler for BASIC by Microsoft
45450D	GW-BASIC by Microsoft	an advanced BASIC interpreter
45447D	Pascal by Microsoft	the popular Pascal compiler
45448A	COBOL by Microsoft	for business applications
45449A	FORTRAN by Microsoft	business, scientific, personal use
45452D	Lattice C Compiler	New! For serious programmers
45310A	BASIC Programmer's Library	practical routines for BASIC users
45443A	Forms Master	powerful forms design tool
92248BA	Cross Reference Utility	for BASIC by Microsoft users
45435A	Touchscreen Programmers Tools	Assembler and four manuals
45311A	Icon Design System	graphics programming tool

REVISED HP 1000 A-SERIES DATA BOOK

(Ted Proske/DSD)

A revised HP 1000 A-Series data book (P/N 5953-8761). This revised data book incorporates the new 256K-byte RAM ECC (error correcting code) memory products, new terminals, and discs, corrects all known errors in the previous data book (P/N 5953-8712), and deletes all coverage of L-Series products. Because L-Series coverage has been omitted from P/N 5953-8761, P/N 5953-8712 should be kept by anyone needing data on HP 1000 L-Series products.

HP 92077A RTE-A data sheet update

(Ted Proske/DSD)

With expanding use of RTE-A, some customers have encountered problems with configuring large systems. To help customers better understand in advance the large-system support capabilities of RTE-A, we've published an update for the HP 92077A RTE-A data sheet in the HP 1000 Software Technical Data book, Volume 1 (P/N 5953-8710). As soon as it becomes available, the RTE-A update **must** be included in the HP 1000 Software Technical Data book, clipped to pages 2-11.

HP 1000 Series A600+ and A700 gain 256K bytes RAM

(Dan Meitus/DSD)

Last February, Data Systems Division (DSD) introduced HP's first 256K-byte RAM product for the HP 1000 A900 computer. Now, DSD is making available these high-density 256K-byte RAMs for five new memory products for the HP 1000 Series A600+ and A700 computers. With these five new products, we now have a complete line of 256K-byte RAM products for the A-Series family.

New HP 1000 products

P/N	SPU	Description	Replaces
HP 12110A	A600+	512K-byte ECC memory controller	N/A
HP 12110B	A600+	1M-byte ECC memory controller	N/A
HP 12111A	A600+/ A700	512K-byte ECC memory array	12104A
HP 12111B	A600+/ A700	1M-byte ECC memory array	12104B
HP 12111C	A600+/ A700	1M-byte ECC memory array	12104C

DESKTOP FORUM

DESKTOP TITBITS

A few problems and their solutions for desktops

PRODUCT NUMBER: HP-87A/86 REVID: A
Problem: If a multi-line function is called from a 'Disp' or 'Print' statement, i.e. 20 DISP FNS, and the multi-line function contains a 'Read' statement, the machine may lock up or may generate erroneous error messages like Error 21: ROM Missing.

Work Around Solution: Do not call multi-line functions which contain 'Read' statements from 'Disp' or 'Print' statements.

PRODUCT NUMBER: HP-85A REVID: B
Problem: Operating system will not accept a calculator mode statement with the 'Rename' statement followed by the '@' (at sign) and any other statement or command.

Work Around Solution: Do not use 'Rename' in multistatement command. Note: This problem will not occur if there is a mass storage ROM in the system as the mass storage ROM also defines the 'Rename' statement and permits statements and commands to follow using the '@' (at sign) Syntax.

PRODUCT NUMBER: HP-87A/86 REVID: A
Problem: If the colon is forgotten in the MSUS portion of the 'Initialize' command, e.g. Initialize "B", "D701", the current default drive will be initialized. The MSUS is ignored. (Note: This problem also exists in the 85 Mass Storage ROM).

Work Around Solution: Don't do this.

PRODUCT NUMBER: HP-87A/86 REVID: A
Problem: When using an HP-87 Assembler ROM, a source code file longer than 64K-bytes will show up in a catalog as having 257 bytes/rec.

Work Around Solution: This will not affect loading and storing of source files.

PRODUCT NUMBER: HP-85A REVID: B
Problem: Operating system will not accept a calculator mode statement with the 'Rename' statement followed by the '@' (at sign) and any other statement or command.

Work Around Solution: Do not use 'Rename' in multistatement command. Note: This problem will not occur if there is a mass storage ROM in the system as the mass storage ROM also defines the 'Rename' statement and permits statements and commands to follow using the '@' (at sign) Syntax.

PRODUCT NUMBER: HP-85A REVID: B
Problem: If a single line function definition starts in the middle of a multistatement line it will generate erroneous line numbers if it causes an error or is traced, e.g.

```
10 X=999999 @ DEF FNA = 2
20 DISP FNA/X
30 END

TRACEALL [ENDLINE] [RUN]
TRACE LINE 10 X=999999
TRACE LINE 10 TO 20
.000002000002
TRACE LINE 899 TO 3D

10 X=99999 @ DEF FNA = 2/0
20 DISP FNA/X
30 END

[RUN]
    WARNING 8 ON LINE 899: /ZERO
    1 000001E494
```

Work Around Solution: Do not use 'DEF FN' in a multistatement line.

PRODUCT NUMBER: HP-85A REVID: B
Problem: While in calculator mode the statement SIN(SIN(SIN(SIN(SIN(SIN(.3))))) will give an 'ERROR 88: BAD STMT'. Any of the TRIG functions will do so. However A = (Same as above) will work correctly.

Work Around Solution: Always assign statements of this nature to a variable in calculator mode.

9825 to 86B data transfer

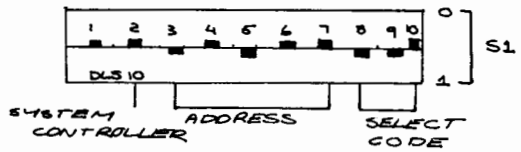
PROJECT:
To establish means of taking data files from HP 9825 Desktop Computer and passing the data into A. HP86B Desktop Computer and storing the data on 4.6 Meg. Winchester.

PRELIM:
The structures of the two data bases involved are entirely different so the data is passed in groups of strings into temporary files in the HP 86B system.

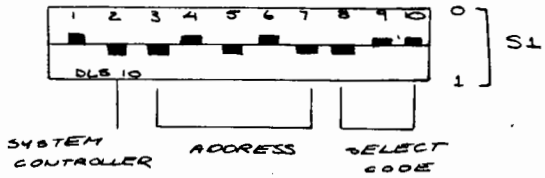
From here the required translation of data can be performed without further need of the link between the two computer systems.

EQUIP:
9825. HP-IB Interface 98034A Select Code 7 HP86B, 4.6 Mbyte Winchester HP 9133A, HP-IB Interface 82937A.*

* The 82937A Interface has to be opened and the Pattern on switch S, set as below:-



The normal pattern for S1 when used in the HP85 is as below:-



PROCEDURE: Link both HP-IB Interfaces together.
Key following programs (or adaptions of) into respective computers.
Press 'RUN' on HP-86B
Press 'RUN' on HP9825
Data will be transmitted.

HP9825

```
0 : dim D$[25,
  146]
1 : trk 1; ldf 0,
  D$
2 : for L=1 to 25
3 : ltf(D$[L])→X
4 : fmt 1,f6.0,c
5 : wrt 720.1,X,
  D$[L,3], next L
6 : dsp "done!"
*13315
```

HP-86B

```
10 DIM A$(25)[146],X$[200]
20 FOR L=1 TO 25
30 ENTER 9 ; X$
32 PRINT X$
40 A$(L)=X$
50 NEXT L
60 DISP "DONE"
70 END
```

— John Hedger

TWO SNIPPETS (SERIES 200 BASIC)

1. COMMAS INTO NUMBERS

Function to put commas into a number to separate 000's SERIES 200 BASIC

E.g.: to convert 12345678.9999 to 12,345,678.9999

N.B. It's dead easy on Series 80 by using IMAGE DDDCDDDC . . . (C for comma) but there's no such IMAGE facility in 200 BASIC that I've found yet!

Example of use, where the number in X is to be printed RIGHT-justified in a field 16 characters wide and shown rounded to 2 dec. places: PRINT . . .:FNN$(X,16.2)

```
230  !.....................................................................
240 Number:    DEF FNN$(N,I) ! ....... Returns N with COMMAs (Image=I=fw.dp) ...
250            DIM R$[40],Z$[40]
260          L=INT(I)
270          I$="#,"&VAL$(L)&"D"
280          IF L<>I THEN I$=I$&"."&CHR$(48+10*FRACT(I))&"D"
290          OUTPUT Z$ USING I$;N
300          Z$=TRIM$(Z$)
310          IF Z$[1,1]="." THEN Z$="0"&Z$
320          Z$=REV$(Z$)
330          X=POS(Z$,".")
340          R$=Z$[1,X]
350 Num1:    Z$=Z$[X+1]
360          IF LEN(Z$)<=3 THEN Num2
370           R$=R$&Z$[1,3]&","
380           X=3
390           GOTO Num1
400 Num2:    R$[1,L]=R$&Z$
410          RETURN REV$(R$[1,L])
420    FNEND
430  !.....................................................................
```

2. BEWARE OF "FRACT" FUNCTION

FRACT is brutally honest when dealing with REAL-precision numbers. No rounding is performed and you get exactly how it is stored, which is always approximate, as it is stored in binary format, not decimal.

So, if you get the result '1' for the calculation:
PRINT INT (10*FRACT(N)) — where N = 16.2 say, the reason is that the fractional part of 16.2 is something like .1999 999 999 996 (and calculations are done to a very high precision).

MORAL: Beware FRACT! Use **rounding** to obviate the problem — either explicitly, e.g.: PROUND function), or implicitly, e.g. CHR$ (..+FRACT(...)) where the CHR$ function does the rounding.

— CHRIS SIMPSON (03 859 6643)

AND ANTENNAE

A spectacular presentation was made by Gordon Roberts of Telecom at the HPDCUGV meeting on 21st February at CSIRO, Syndal. Twenty-three people watched in awe as Gordon, complete with a colour video tape especially prepared for the occasion, explained how he has used the 3-D graphics capabilities of his HP9000 desktop (the big one) to assist with microwave antenna mast location and design. The video saved a lot of heavy cartage of computer gear and yet brought home to us the incredible speed and quality of colour graphics available on the 9000, and Gordon's wizardry at the art. His phone number is (03) 606 6063.

The meeting was a busy one with offerings of a 9825 tape-based Text-Editor program (on cartridge) by Peter Hendy (03) 878 3333, and of a snippet (to be published). John Michell declared his need for a "System Monitor" required for the HP86B Assembler package. John is on (03) 870 1125. Also advice was sought about capturing print-out from another computer — John was swamped with suggestions! Graeme Rodwell asked for help from anyone familiar with RS232 Block Mode and also Checksums: (03) 509 2444.

By invitation, HP's turn came next. John Klein explained briefly the newest of the new HP restructures and how it affects us — it all sounds quite promising — thankyou John. And Ian Clarke brought along the brand new "Integral PC" computer, having boned up ready to give a splendid demonstration of this remarkable portable UNIX-based machine.

Thanks go to CSIRO and Rory Cox again for the venue. As usual, horse-whips were required to vacate the premises!

The next meeting will be another spectacular and open to all comers, including a controversial talk by HP's General Manager Bruce Marsh, more new equipment, more snippets, info and help, and an AGM will be squeezed in too: Tuesday, 16th April, 1985 at 4.00 pm. sharp at HP this time (and drinks are on the house)!

— Chris Simpson, HPDCUGV President, (03) 859 6643

CONNECT

CONNECT is a general-purpose multi-user terminal-emulator for the HP1000 A-series user. Using the HP12040B MUX, it provides a logical I/O switch between any terminal and a remote system. In addition, it can transfer files between systems, providing a complete intersystem link capability.

CONNECT features

* Any terminal on the system can connect to any remote system via modem or hardwired link.
* Terminal and remote may use different baud rates and pacing methods (XON/XOFF,ENQ/ACK, response-string recognition, etc)
* Multiple concurrent sessions.
* Extensive interactive contextual HELP command.
* Automatic link configuration depending on either remote CPU link and/or local user ID.
* Transparent file transfer utility with error checking allows any file to be moved between HP1000 and remote system.
* PLUS many other features which make CONNECT the logical choice for remote system links.

Call us today!

OR

Call our dealers:

QLD:Peter Cox (CES)
Phone (07) 356-8066
NSW:Arthur Carlton (IAS)
Phone (049) 436-158
WA :Matt Saunders (COMSYS)
Phone (09) 381-7877

TUSC
Computer Systems

901 Whitehorse Road,
Box Hill. 3128. Victoria.
Phone : (03) 898-0636
Telex : AA36524 (DCSMMC)

A SORT ROUTINE USING VIRTUAL MEMORY

(By Robert Hanna, STC)

While working on a program for shop floor machine control, a sort routine was needed that was able to handle large files and also to be able sort by different fields, compared to the usual comparing the first column, then the second, and so on.

Initially the sort routine was written to operate in standard memory and operated on characters but the standard memory allocation proved to be insufficient, so an alternative had to be found.

This sort routine operates using virtual memory, and therefore can only operate on integers (a VMA restriction). The maximum file length is 10,000 lines of 66 integers (or 10,000 lines of 132 characters). This requires character data to be converted to integer, and so, to do this, two additional routines were written, 'prepare-input' and 'prepare-output', for converting the character data to and from integer form. To enable the data to be sorted according to the required fields, these two routines also rearrange the line into the desired sequence for sorting, and arrange it back to the original sequence after the sort.

One further routine, 'set-up-sequence', was written to enable 'prepare-input' and 'prepare-output' to carry out the required string manipulations, producing an array containing the original and new field positions for manipulation control.

The sort routine and its supporting routines are written as subroutines and as such may be put in a library and called by any program that must carry out a sort. The program listed below is mainly intended as an example of how the subroutines are called, but may be used as is or modified to suit your application.

Note, this routine only sorts in ascending order, if a descending order is required then all that needs to be done is that the sorted file be written out in reverse order.

```
FTN7X,S
$files 2,2
$EMA /data_block/

*{+-----------------------------------------------------------------+
*|Name :  SORT                                                      |
*|Type :  Program                                                   |
*|Language :  FTN77                                                 |
*+-----------------------------------------------------------------+
*| ABSTRACT :                                                       |
*|-----------                                                       |
*|     Sorts large files,according to up to 10 field specifications |
*|     specified in the run string.                                 |
*+-----------------------------------------------------------------+
*| CALLING SEQUENCE :                                               |
*|------------------                                                |
*|         sort,I/P file namr,O/P file namr [,field specifications] |
*|           Up to 10 fields may be specified and should be of the  |
*|           form                                                   |
*|                    X:Y                                           |
*|           where X is the start column and Y is the finish column |
*|                 of the given field                               |
*|                                                                  |
*|         eg.  sort,aaa,bbb,6:12,1:5,30:35,.....                   |
*+-----------------------------------------------------------------+
*| PRE-REQUISITES :                                                 |
*|----------------                                                  |
*|     when loading,specify VMA size and working area.              |
*+-----------------------------------------------------------------
*| Create date :       1:00 PM MON  ., 6 JUNE., 1983                |
*| Programmer  :       Robert Hanna                                 |
*| Last change :       <830606.1600>                                |
*+-----------------------------------------------------------------+}


******************************************************************
      program SORT (4,250)
     + ,sorts files RH<830606.1600>
******************************************************************

      implicit none

      character*132 input_buffer
      character*132 output_buffer
      character*20 input_file

        character*20 output_file
        character*8 buffer

        integer input_unit
        integer output_unit
        integer rcpar
        integer index
        integer length
        integer string_length
        integer position
        integer error
        integer M
        integer N

       integer S
       integer seq(10,2)
       integer sequence(21,4)
       integer sort_array(66,10000)

       logical ifbrk

       common /data_block/ sort_array
       common /seq_block/ sequence

       data input_unit/10/
       data output_unit/20/

*==== Get input file from runstring
       IF (rcpar(1,input_file) .EQ. 0)
      + STOP 'sort,<input_file>,<output_file>[,<field specifications>]'

*==== Get output file from runstring
       IF (rcpar(2,output_file) .EQ. 0)
      + STOP 'sort,<input_file>,<output_file>[,<field specifications>]'

*==== Get field specifications from runstring
       DO S=3,12
            IF (rcpar(S,buffer) .EQ. 0) THEN
                 GO TO 30
            ELSE
                 position=index(buffer,':')
                 length=index(buffer,' ')-1
                 READ (buffer(1:position-1),*) seq(S-2,1)
                 READ (buffer(position+1:length),*) seq(S-2,2)
            END IF
       END DO

 30    CALL set_up_sequence (seq,132)

       OPEN (input_unit,file=input_file,use='NONEXCLUSIVE',iostat=error,
      + err=99)

       OPEN (output_unit,file=output_file,iostat=error,err=99)

       N=0

 40    READ (input_unit,'(a132)',iostat=error,err=99,end=60) input_buffer

*==== Break flag
       IF (ifbrk()) THEN
            WRITE (1,*) char(27)//'&dB--- OPERATOR BREAK ---'//char(27)//
      +      '&d@'
            GO TO 99
       END IF

       N=N+1
       CALL prepare_input (input_buffer,N)
       GO TO 40

 60    CALL quicksort (N,66)

 50    DO M=1,N
            CALL prepare_output (output_buffer,M)
            WRITE (output_unit,'(a132)',iostat=error,err=99)
      +      output_buffer
       END DO

 99    IF (error .NE. 0) WRITE (1,'("Error = ",i6)') error

       CLOSE (input_unit)
       CLOSE (output_unit)

       END


$EMA /data_block/
*{+------------------------------------------------------------------+
*|Name :  prepare_input                                              |
*|Type :  Subroutine                                                 |
*|Language :  FTN77                                                  |
*+------------------------------------------------------------------+

*| ABSTRACT :   Prepares data for use by the subroutine 'quicksort' by   |
*|-----------       i) arranging the line in required sequence for sorting |
*|                 ii) puts the line in 'sort_array' in VMA          |
*|              See abstract for subroutine 'quicksort'.             |
*|              See abstract for subroutine 'set_up_sequence'.       |
*+------------------------------------------------------------------+
*| CALLING SEQUENCE : CALL prepare_input (input_buffer,row)          |
*|------------------   where input_buffer= line just read in from i/p file |
*|                         row=position in sort_array for input_buffer |
*+------------------------------------------------------------------+
*| PRE-REQUISITES : Specify VMA size and work area.                  |
*|----------------                                                   |
*+------------------------------------------------------------------
*| Create date :       1:00 PM  MON., 6 JUNE , 1983                  |
*| Programmer  :      Robert Hanna                                   |
*| Last change :      <830606.1600>                                  |
*+------------------------------------------------------------------+}
******************************************************************
      SUBROUTINE prepare_input (input_buffer,row)
     + ,prepares data for sortq RH<830606.1600>
******************************************************************

      implicit none

      integer J
      integer row
      integer column
      integer length
      integer sequence(21,4)
      integer buffer_integer(66)
      integer sort_array(66,10000)

      character*(*) input_buffer
      character*132 arrange_buffer
      character*132 buffer_character

      equivalence (buffer_character,buffer_integer)

      common /data_block/ sort_array
      common /seq_block/ sequence
```

```
      length=len(input_buffer)

x==== Convert data to correct sequence for sorting priority

      DO J=1,21
          IF (sequence(J,1) .NE. 0)
     +     arrange_buffer(sequence(J,3):sequence(J,4))=
     +     input_buffer(sequence(J,1):sequence(J,2))
      END DO

      input_buffer=arrange_buffer

x==== Convert data to numerics for storage in VM
      buffer_character(1:length)=input_buffer

      DO column=1,66
          sort_array(column,row)=buffer_integer(column)
      END DO

      RETURN
      end ! prepare_input


$EMA /data_block/
x{+-------------------------------------------------------------------------+
x!Name :  prepare_output                                                    !
x!Type :  Subroutine                                                        !
x!Language :  FTN77                                                         !
x+-------------------------------------------------------------------------+
x! ABSTRACT :   Takes data from the sort_array in VM and                    !
x!-----------          i) puts the data back in character form              !
x!                    ii) puts the line back in original sequence           !
x!               See abstract for the subroutine 'set_up_sequence'.        !
x+-------------------------------------------------------------------------+
x! CALLING SEQUENCE : CALL prepare_output (input_buffer,row)                !
x!------------------  where output_buffer= line to be read out to o/p file!
x!                           row=position in sort_array for output_buffer  !
x+-------------------------------------------------------------------------+
x! PRE-REQUISITES : Specify VMA size and work area.                         !
x!-----------------                                                        !
x+-------------------------------------------------------------------------+
x! Create date :       1:00 PM  MON., 6 JUNE , 1983                        !
x! Programmer  :      Robert Hanna                                         !
x! Last change :      <830606.1600>                                        !
x+-------------------------------------------------------------------------+>
x************************************************************************
      SUBROUTINE prepare_output (output_buffer,row)
     + ,prepares data for output after quicksort RH<830606.1600>
x************************************************************************

      implicit none

      integer J
      integer row
      integer column
      integer length
      integer sequence(21,4)
      integer buffer_integer(66)
      integer sort_array(66,10000)

      character*(*) output_buffer
      character*132 buffer_character
      character*132 rearrange_buffer

      equivalence (buffer_character,buffer_integer)

      common /data_block/ sort_array
      common /seq_block/ sequence

      length = len(output_buffer)

x==== Convert sorted data back to characters
      DO column=1,length/2
          buffer_integer(column)=sort_array(column,row)
      END DO

      output_buffer=buffer_character(1:length)

x==== Convert sorted data back to original sequence
      DO J=1,21
          IF (sequence(J,1) .NE. 0)
     +     rearrange_buffer(sequence(J,1):sequence(J,2))=
     +     output_buffer(sequence(J,3):sequence(J,4))
      END DO

      output_buffer=rearrange_buffer

      RETURN
      END  !   prepare_output


$EMA /data_block/
x{+-------------------------------------------------------------------------+
x!Name :  set_up_sequence                                                   !
x!Type :  Subroutine                                                        !
x!Language :  FTN77                                                         !
x+-------------------------------------------------------------------------+
x! ABSTRACT :   Takes an array seq(10,2) containing the sort field spec-   !
x!-----------   ifications and creates a usable array for prepare_input    !
x! and prepare_output subroutines. The array seq must be of the form :     !
x!             seq(X,1) is the start of the X field                        !
x!             seq(X,2) is the finish of the X field                       !
x+-------------------------------------------------------------------------+
x! CALLING SEQUENCE : CALL set_up_sequence (seq,length)                     !
x!------------------  where seq=array,(10,2),containing sort field specs.  !
x!                          length=length of input_buffer in main prog.    !
x+-------------------------------------------------------------------------+
x! PRE-REQUISITES : Specify VMA size and work area.                         !
x!-----------------                                                        !
x+-------------------------------------------------------------------------+
x! Create date :       1:00 PM  MON., 6 JUNE , 1983                        !
x! Programmer  :      Robert Hanna                                         !
x! Last change :      <830606.1600>                                        !
x+-------------------------------------------------------------------------+>
x************************************************************************
      SUBROUTINE set_up_sequence (seq,length)
     + ,sets up sequence array for use by arrange and rearrange
     + RH<830606.1600>
x************************************************************************

      implicit none

      integer C
      integer D
      integer row
      integer column
      integer length
      integer sequence(21,4)
      integer seq(10,2)
      integer sort_array(66,10000)

      common /data_block/ sort_array
      common /seq_block/ sequence

x==== Make sure 'sort_array' and 'sort_seq_array' are clear for setting
x==== up sequence for sorting priority
      DO C=1,21
          sort_array(1,C)=0

          DO D=1,4
              sequence(C,D)=0
          END DO
      END DO

x==== 'sequence' is set up
      DO C=1,10
          sequence(C,1)=seq(C,1)
          sequence(C,2)=seq(C,2)
      END DO

      DO row=2,20,2
          sort_array(1,row)=sequence((row/2),1)
          sort_array(1,(row+1))=sequence((row/2),2)
      END DO

      CALL quicksort (21,1)

      DO row=2,20,2
          IF (.NOT. ((sort_array(1,row).EQ.0) .OR.
     +     (sort_array(1,row) .EQ. (sort_array(1,(row-1))+1)))) THEN
              sequence((10+row/2),1)=sort_array(1,row-1)+1
              sequence((10+row/2),2)=sort_array(1,row)-1
          END IF
      END DO

      sequence(21,1)=sort_array(1,21)+1
      sequence(21,2)=length

      C=1

      DO row=1,21
          IF (sequence(row,1) .NE. 0) THEN
              sequence(row,3)=C
              sequence(row,4)=C+sequence(row,2)-
     +         sequence(row,1)
              C=sequence(row,4)+1
          END IF
      END DO

      RETURN
      END  ! set_up_sequence
$EMA /data_block/
x{+-------------------------------------------------------------------------+
x!Name :  quicksort                                                         !
x!Type :  Subroutine                                                        !
x!Language :  FTN77                                                         !
x+-------------------------------------------------------------------------+
x! ABSTRACT :  This subroutine sorts an array, that is in VMA, of size up  !
x!-----------  to (66,10000) in ascending order. The area of 'sort_array'   !
x! that is actually used can be specified and only this area is sorted,    !
x! eg (32,100), 100 rows by 32 columns.                                    !
x! NOTE : dimensions are (column,row)                                       !
x+-------------------------------------------------------------------------+
x! CALLING SEQUENCE : CALL quicksort (row,column)                           !
x!------------------  where row and column are the dimensions of section   !
x!                        of sort_array actually being used                !
x+-------------------------------------------------------------------------+
x! PRE-REQUISITES : Secify VMA size and work area.                          !
x!-----------------                                                        !
x+-------------------------------------------------------------------------+
x! Create date :       1:00 PM  MON., 6 JUNE , 1983                        !
x! Programmer  :      Robert Hanna                                         !
x! Last change :      <830606.1600>                                        !
x+-------------------------------------------------------------------------+>
x************************************************************************
      SUBROUTINE quicksort (row,column)
     + ,sorts data array RH<830606.1600>

x     The algorithm for this quicksort subroutine was translated
x     from the "Non-recursive Version of Quicksort" Program 2.11
x     on P80 of "Algorithms+Data Structures=Programs" by Niklaus
x     Wirth
x************************************************************************

      implicit none

      integer row
      integer column
      integer C
      integer D
      integer I
      integer J
      integer L
      integer R
      integer S
      integer B(200,2)
      integer comparater(66)
      integer exchange_buffer(66)
      integer sort_array(66,10000)
      logical less_than
      logical greater_than

      common /data_block/ sort_array

      S=1
      B(1,1)=1
      B(1,2)=row
      DO WHILE (S .NE. 0)
         L=B(S,1)
         R=B(S,2)
         S=S-1
         DO WHILE (L .LT. R)
            I=L
            J=R
            D=(L+R)/2
            DO C=1,column
                comparater(C)=sort_array(C,D)
            END DO
            DO WHILE (I .LE. J)

x=============== This loop is done for sort_array(-,I) less than comparater
                DO WHILE (less_than(comparater,I,column))
                    I=I+1
                END DO
```

```
!=============== This loop is done for sort_array(-,J) greater or equal to
!=============== comparater
                DO WHILE (greater_than(comparater,J,column))
                   J=J-1
                END DO
                IF (I .LT. J) THEN
                   DO C=1,column
                      exchange_buffer(C)=sort_array(C,I)
                      sort_array(C,I)=sort_array(C,J)
                      sort_array(C,J)=exchange_buffer(C)
                   END DO
                END IF
                IF (I .LE. J) THEN
                   I=I+1
                   J=J-1
                END IF
             END DO
                IF (I .LT. R) THEN
                   S=S+1
                   B(S,1)=I
                   B(S,2)=R
                END IF
             R=J
          END DO
       END DO

       return
       end ! quick_sort

$EMA /data_block/
!*****************************************************************************
      LOGICAL FUNCTION less_than (comparater,I,column)
     + ,sort_array LT comparater RH<830606.1600>
!*****************************************************************************

      implicit none

      common /data_block/ sort_array

      integer column
      integer C
      integer I
      integer comparater(column,1)
      integer sort_array(66,10000)

      less_than=.false.

      DO C=1,column
          IF (sort_array(C,I) .LT. comparater(C,1)) THEN
              less_than=.true.
              GO TO 210
          ELSE
              IF (sort_array(C,I) .NE. comparater(C,1)) GO TO 210
          END IF
      END DO

 210  return
      end  !  less_than

$EMA /data_block/
!*****************************************************************************
      LOGICAL FUNCTION greater_than (comparater,J,column)
     + ,sort_array GT comparater RH<830606.1600>
!*****************************************************************************

      implicit none

      common /data_block/ sort_array

      integer column
      integer C
      integer J
      integer comparater(column,1)
      integer sort_array(66,10000)

      greater_than=.false.

      DO C=1,column
          IF (sort_array(C,J) .GT. comparater(C,1)) THEN
              greater_than=.true.
              GO TO 210
          ELSE
              IF (sort_array(C,J) .NE. comparater(C,1)) GO TO 210
          END IF
      END DO

 210  return
      end  !  greater_than
```

HP 1000 New File System Time Stamps

One of the many nice features of the new file system on the HP 1000 line of computers is time stamping of files. You can now do things like:-

- backup all the files updated since the beginning of August 1984.
- backup and delete all files not accessed since June 1984.
- List all the files called test.ftn on the system and when they were last updated.

Now imagine what happens if one day you forget to set the time. Every file accessed or updated has its appropriate time stamps set to a date in 1983! Just as bad is the day someone enters the year as 1994.

What you need is a program that makes sure you don't forget to set the time or set it to something silly. Well, here it is.

TIME manages the system time on an HP1000 with a hierarchical file system (i.e. RTE-A and RTE-6 Rev. C.83 or later).

It is run as follows:

TIME,(lu),(mode),(clock),(grace)

Where the characters (and) enclosing a word denote the name of a parameter which is specified at run time, the characters (and) are never entered when the program is run. Parameters not needed for a particular mode may be omitted.

The basic idea of the program is to ensure that the time stamps on a system remain logically consistent. This involves ensuring the time is set to something sensible between the time that the system is booted and when users start using files. At the same time it is important to ensure that once set the system time can't be altered except by the system manager.

The program has several modes:

(mode) = s Set Mode

To ensure that the time is correctly set before users get onto the system, TIME will normally be run at the very beginning of the welcome file. This ensures that no users or programs can run until the time is set.

TIME maintains a file/system/time which contains the last time known to the program. This is known as the last recorded time. TIME keeps asking for the time until it gets a date and time later than the last recorded date and time but no more than (grace) days after this.

It is sometimes desirable to be able to bypass this checking. If the response to "Is this correct (y/n)?" matches the password on the second line of /system/time this checking is not performed. This feature is deliberately not advertised by the prompts.

IF TIME detects that the system has just been booted (i.e. system time at default value) it updates system time to the last recorded time before asking the operator to enter the time. This is needed because it is actually possible to get a break mode prompt and "of,time", should this be necessary.

For example:

```
CI.01> TIME,1,S,12

 Please set date and time.

Year [ 1984 ]
Month [    7 ]
Day [   10 ]
Hour [   12 ]
Minute [   15 ] 16
Second [   58 ] 40
AM/PM [ AM ]

Tue Jul 10, 1984 12:16:40 am

Is this correct (y/n) ? Y

CI.01>
```

Use 12 hour format.
Defaults are shown in square brackets, pressing carriage return selects the default.
Time will not be set until confirmed as correct.

(mode) = a Automatic Mode

If it is impractical to have the system inoperative from boot-up until the time is set, auto mode can be used at the beginning of the welcome file and set mode after the system is got going. Auto mode simply sets the time to the last recorded time, as long as this is later than the current time.

(mode) = u Update Mode

Obviously the closer the last recorded time is to the time that a file was last accessed the more accurate TIME can keep the file time stamps. Update mode updates the last recorded time from the system time. It should be run in this mode as part of your backup procedure and preferably on a regular basis. One way to achieve this is to run it in the system manager's hello file (write access to /system/time is needed to alter the time). If you are paranoid you can time schedule TIME it only uses rmpar parameters.

As protection from users altering the system time (e.g. with settm()) update mode checks that the current time is later than the last recorded time. If this is not so the current time is updated to the last recorded time rather than the reverse.

(lu) is the lu to use for operator dialogue. If nothing sensible is given 1 is used.

(clock) is 12 for 12 hour clock (i.e. AM and PM) or 24 for 24 hour clock. It is only used by set mode.

(grace) is used to limit the time difference between the last recorded time and the new time entered in set mode. This is to prevent the embarrassing situation of having files that were last updated next year. If omitted or unreasonable (i.e. greater than 365) a default of 6 is used.

To alter either the last recorded time or the current time write permission is needed on /system/time. The protection mode on this file is the basic mechanism used to determine who gets to change the time.

If no parameters are given TIME just prints the current date and time. No other action is performed and anyone can use this mode.

To set up a system to use TIME perform the following steps:

1) Compile and load/link the program TIME.
2) Place a copy of the object code for time on /programs/ and for RTE-6 also on a system fmgr lu so it can be run from the welcome file.
3) For RTE-6 systems ensure that the owner of /system is "." i.e. it was created by a program (e.g. ci) running out of session. On RTE-A anyone can own /system. Protection for /system should be "rw/" (or "rw/r").
4) Create a two line file called /system/time. On the first line enter YYMMDD.HHMMSS where YY=year, MM=month, DD=day, HH=hour, MM=minute, SS=second. For example, 9 a.m. on 1st January, 1970 would be 700101.09000 (no — the program won't stop working on 1st January, 2000. A value for year in the range 0 to 144 is interpreted as 2000 to 2144). On the second line enter the password used to bypass time validity checks. The password must begin with a 'Y' or 'y'. The file should have "rw/" protection if you want this to remain a secret.

I am quite happy to make the source available to anyone who wants it, through their account SE, but it should be clearly understood that there is no commitment from HP to support this program. Any bugs you find you can keep.

— David Triggs

ODD SPOT

Did you see Computerworld's recent article (3/12/84) about new HP gear and especially the NEW PORTABLE Battery driven Disc drive for $5.600.00 or so? (Actually they meant the 9114A drive which costs $1.371 inc. tax).

Connecting your P.C. to the HP 1000

(Keith Haberle, C.S.I.R.O. Chemphysics)

With the increasing number of PC's in the workplace there has been a lot of interest shown in the ability to communicate with various mainframes.

The mainframes of interest to us are the HP1000 and those on the CSIRONET network. Problems that we have encountered so far with file transfers to and from CSIRONET have strengthened our resolve to restrict our efforts to communicating with the HP, and rely on existing proven links between the HP and CSIRONET for any CSIRONET connections.

First we need to establish on what we envisage as being the basic workstation to connect to the HP, we see the following components as essential:

1. Microcomputer with dual floppy drives
2. V.D.U.
3. Printer
4. Various compilers and utilities, e.g. Word Processors.

Extras may well include graphics capabilities, either on-screen, plotter, or both.

The printer is essential for stand alone computing and should be able to be used as a remote printer to the HP when operating on that system. Either as a direct L.U. or by some means of spooling to the micro discs.

First and foremost with any communications package is the ability to make the workstation act as a dumb terminal. This should of course be as transparent as possible to the user, i.e. if the system uses linefeed as a terminator so should the communications package if at all possible. There are any number of communication packages that can perform this task O.K. The next requirement of the package is file transfer, and this is a major stumbling block.

Consider getting files from the HP to the micro, the simplest method for transfer is to set up a process that writes records (usually variable length, terminated by CR/LF) directly to the micro's connecting port, i.e. ST, file, 1.

Things are a little bit sticky here because this has to be set up while in terminal mode, but without actually performing the CR to start up the transfer, you leave this command pending, go back into the communications package menu, and set up the micro for data capture, only now can a CR be sent to the HP. Once the CR is seen at the HP end the data will come streaming down the line until the file is exhausted.

There are a number of commercially available communications packages which use this means of file transfer. We have one of these and it has been tested on a number of micros with reasonable results. The first and most obvious disadvantage is that it includes an awkward and complex series of commands, some on the HP, some on the micro, to initiate the process. this can be lived with and obviously has by a large number of people. It's major disadvantage is that there is no error checking whatsoever. Therefore the file has to be checked manually, after transfer, for the possibility of corrupted or even missing records, and we have seen both of these problems occurring.

Not withstanding these error possibilities, the file has to be edited locally anyway due to the junk records that appear at the beginning and end of file transfer. These occur because the system is not automatic and simply sets up a receiving buffer to grab everything that appears on the line, e.g. the prompt sent when the ST is complete. A further problem here is that if you forget to turn off capture mode at this point, all the subsequent conversation that occurs in terminal mode will be added to the capture buffer until such time as the capture mode is turned off.

To send files to the HP is essentially the reverse process, with the same sort of inherent problems, e.g. ST, 1, file and of course local editing of the file once on the HP would be needed.

The transmission errors mentioned earlier are found to be highly dependent upon both baud rate and system load.

The conclusion about file transfers with this type of package is that although it works to a certain degree, it is very primitive. One aspect about file transfers that should not be ignored is the need to obtain a directory listing of the micro drives without having to exit the communications package. Not all available packages have this facility and are therefore unsuitable.

As mentioned before, the local printing of files would be a most desirable feature. This appears to be offered on some packages as a toggle to print while listing to the screen, we are yet to see one on which this actually works. Being able to treat the printer as a distinct LU on the HP system or spooling print files to local disc would be highly desirable.

So far we have as basic requirements:

1. Terminal emulation
2. Reliable file transfer
3. Local directory listing

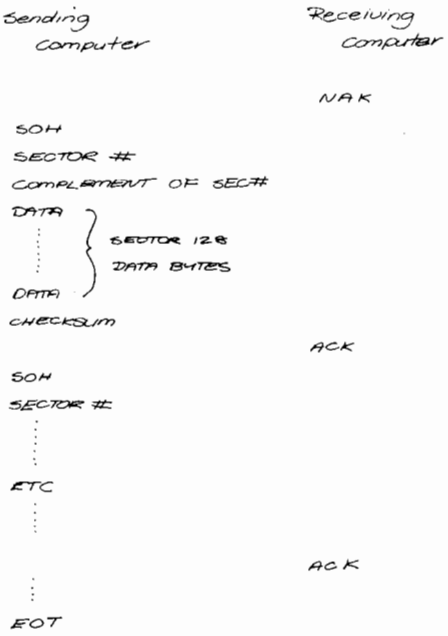
Remote file print handling.

We have looked at and tried a number of packages, each offer some of the above, some claim all, but none actually fulfill at the requirements. Each is highly sensitive to baud rate and this is the area of major concern. The only answer being to implement some form of file transfer protocol. On investigation, one such protocol which seems to be fast becoming the industry standard for micro's, Z80 based CP/M machines in particular, is the Ward Christensen Protocol.

This involves sending the file sector by sector, i.e. 128 bytes at a time. Each sector is preceded by 3 bytes containing respectively

SOH (01 Hex) start of header
the sector number and
its complement
then follows the 128 byte record and finally a checksum, which is the sum of the sector bytes modulo 256.

WARD CHRISTENSEN PROTOCOL
Normal transmission



By definition, the receiving machine has to get into synch with the sender by first sending a NAK. The first sector is then transmitted and is re-transmitted if after a prescribed interval as ACK is not sent by the receiving computer, a certain number of re-tries is usually allowed before aborting the transmission.

Clearly then for micro to mainframe file transfer the micro has to force the mainframe to send a NAK, so we need a program on the mainframe which can be started by the micro, which will then break up the sectors sent into variable length records and also calculate and check the checksum.

Note here that for the transfer we are talking of fixed records of length 132 bytes, the last record is made up to this by padding out the sector with the SUB character (1A Hex). This is standard in both CPIM and MS DOS.

Once the code is written for the HP it is irrelevant which processor it is talking to, but the common part on the micro is a different story. Portability is most important if it is to be at all useful (especially to us since we have micros with 280 8086, 6809, etc.). This code cannot realistically be written in Z80 then transcribed to 8086 or 6809 or whatever. But we do need to be able to handle bit manipulation and to address I/O ports directly. Higher level languages like FORTRAN are not suitable.

We have looked at C and it seems to satisfy all the above criteria although it is rather hard to get into, another and possibly easier to use language would be PASCAL, in particular TURBO PASCAL which seems also to satisfy the above, is available for a wide range of machines and is also a lot easier to get into.

Of all the packages available that also use Ward Christensen protocol, XMODEM, MODEM 7, etc. or one of its derivatives is fairly close to what we want, but as is usual, not exactly what we desire. The source code for a number of these is readily available and I feel that one of these with modifications would suit our needs and save us the immense amount of time and energy of having to start from scratch.

Interex/HP 1000 users conference scheduled on April 9-12

(Pam Tower/SMC)

Interex, the International Association of Hewlett-Packard Computer Users, and the European HP 1000 users group are proud to present their biennial computer conference in Europe. The conference, organized by the Belgian users group, will be held at the Conference Center of the University of Antwerp, Belgium, on April 9-12, 1985. You are invited to take part in this unique exchange of technical expertise. Come and share experiences and learn of new products and applications.

The theme of the conference is "Systems and Applications", and paper topics include operating systems, graphics, CAD/CAM, data base management, data communications, and instrumentation. There will also be a vendor exhibit.

STOP PRESS!

The next issue of Crosstalk is very light for articles at this stage.

I would ask all members and other interested parties to put pen to paper and make the next issue a big fat one.

Articles can be sent to me at Hewlett-Packard:

John Green,
Applications Engineering Organisation,
Hewlett-Packard Australia Limited
31-41 Joseph Street,
Blackburn, 3130.
Phone (03) 895 2661.

PUZZLE PLACE

1. A CUBIC CALENDAR

a) Number two cubes (a single digit on each face) such that any date from the 1st to the 31st may be represented.

b) Place on **lowercase** letter on each face of three cubes so that the first three letters of each of the twelve months may be represented.

c) As for a), but representing the last two digits of any year to 2010.

Voila! An attractive (manual!) calendar.

Mark Michell,
99 Hotham Street,
Collingwood, 3066

SPECIFICATIONS FOR SUBMISSION OF ARTICLES AND ADVERTISEMENTS

All material for Crosstalk should be sent to one of the addresses listed at right, from where it will be forwarded to the co-ordinator for publication. Publication dates are subject to receipt of sufficient material. For specific details contact Glenda Patterson on (03) 895 2576.

ARTICLES: Articles should be typed with any diagrams and program listings in camera-ready form (Author's name, address and phone number should be included).

ADVERTISEMENTS: Display ads. should be in camera-ready artwork form. The printer may be instructed to layout ordinary typeface ads.

CURRENT ADVERTISING RATES:

Full page — $250
Half page — $125
Column/cm — $4

There is a 20% discount on these rates for regular advertisers. Classified ads. are free for user group members, and $10 each for non-members.

Advertisers will be billed upon receipt of ad. The user groups reserve the right to change rates, limit space availability and reject advertising which is deemed inappropriate.

ADDRESSES FOR SUBMISSION OF ARTICLES AND ADVERTISEMENTS:

HP Technical Computer Users Group, N.S.W.
Box 3060 GPO,
Sydney, 2001.
N.S.W.

Norm Kay,
HP1000 Users Group (Vic.)
C.S.I.R.O.
Box 160, Clayton, 3168, Vic.

H.P.D.C.U.G.V.
Mr Bernie O'Shannessy,
Arlec,
30-32 Lexton Road,
Box Hill, 3128, Vic.

HP Desktop Users Group (N.S.W.)
Dr. R. W. Harris,
C/o C.S.I.R.O. Division of Mineral Physics,
PMB 7, Sutherland, N.S.W. 2232.

COMING EVENTS

11th March:	'RTE-A Prog. & System Manager' Course, HP Sydney
13th March:	HP1000 Technical Computer Users Group Annual General Meeting, 4 pm. at HP Melbourne
18th March:	'RTE-6VM Sess. Mon.' Course, HP Sydney 'Introduction to HP1000' Course, HP Melbourne 'MS DOS on HP150' Course, HP Melbourne 'Assembler on HP150' Course, HP Melbourne 'DSN Link on HP150' Course, HP Melbourne
25th March:	'RTE-6VM Sess. Mon.' Course, HP Sydney 'HP9000 UX Intro.' Course, HP Sydney 'Getting Started on HP150' Course, HP Melbourne 'Wordstar on HP150' Course, HP Melbourne 'Lotus 123 on HP150' Course, HP Melbourne 'Condor on HP150' Course, HP Melbourne 'Visicalc on HP150' Course, HP Melbourne
1st April:	'HP9000 UX Sys. Admin.' Course, HP Sydney
8th April:	'HP9000 Ser. 200 UX Sys. Admin.' Course, HP Sydney
15th April:	'RTE-6VM Sys. Mgr.' Course, HP Sydney
16th April:	'H.P.D.C.U.G.V. Annual Meeting with guest speaker, Mr Bruce Marsh, General Manager, HP — from 4.00 pm. to 6.30 pm. at HP, Blackburn.
22nd April:	'Intro. to HP Terminals' Course, HP Sydney.
29th April:	'HP9000 UX Intro.' Course, HP Melbourne 'RTE-A Prog. & Sys. Mgr.' Course, HP Melbourne 'Getting Started on HP150' Course, HP Melbourne 'Wordstar on HP150' Course, HP Melbourne 'Lotus 123 on HP150' Course, HP Melbourne 'Condor on HP150' Course, HP Melbourne 'Visicalc on HP150' Course, HP Melbourne
6th May:	'Pascal' Course, HP Sydney 'Series 200 Bas. Op. & Prog.' Course, HP Melbourne 'RTE-A Prog. & Sys. Mgr.' Course, HP Melbourne
13th May:	'Fortran 77' Course, HP Sydney 'RTE-6VM Sess. Mon.' Course, HP Melbourne 'MS DOS on HP150' Course, HP Melbourne 'Assembler on HP150' Course, HP Melbourne 'DSN Link on HP150' Course, HP Melbourne
20th May:	'HP9000 UX Sys. Admin.' Course, HP Melbourne 'RTE-6VM Sess. Mon.' Course, HP Melbourne
27th May:	'RTE-A Prog. & Sys. Mgr.' Course, HP Melbourne 'Getting Started on HP150' Course, HP Melbourne 'Wordstar on HP150' Course, HP Melbourne 'Lotus 123 on HP150' Course, HP Melbourne 'Condor on HP150' Course, HP Melbourne 'Visicalc on HP150' Course, HP Melbourne
30th May:	'H.P.D.C.U.G.V. Meeting, 4.00 — 6.30 pm. at CSIRO, Syndal.

NOTE: If you have requirements for training on any HP products not covered in our formal training programme, please contact Audrey May on (03) 895 2661.

CLASSIFIED ADVERTISEMENTS

PUZZLE SOLUTION

Answers to last issue's puzzle —

1. Twinkle, Twinkle, little star.
2. Birds of a feather flock together.
3. Look before you leap.
4. Beauty is only skin deep.

RTE INTERNALS

If anyone is interested in attending an RTE internals course, please contact Brett Hutchinson, Systems Engineer, Melbourne. (03) 895 2661.