SCIENCE AND ENGINEERING RESEARCH COUNCIL
RUTHERFORD APPLETON LABORATORY

COMPUTING DIVISION

DISTRIBUTED INTERACTIVE COMPUTING NOTE 660

PERQ
TECHNICAL NOTE 27

PERQ Interim Communications
Facilities II

Issued by
J M Loveluck

29 July 1982

DISTRIBUTION:

F R A Hopgood
R W Witty
K. Robinson
D A Duce
A S Williams
C Prosser
W P Sharpe
C P Wadsworth
I D Benest
E V C Fielding
T Watson
J M Loveluck
C J Webb
K J Fermor
A P Ferryman
L O Ford
J C Malone
J. Smith
RL Support/PERQ/Technical Notes File
RL Support/PERQ/General File

1. Introduction

This note describes progress, since the issue of DCS note 587, on interim communication utilities for the PERQ. Apart from two developments still to be completed (sections 3 and 4) the planned interim communication facilities for the PERQ are essentially comlete, and future effort on PERQ communications will be directed towards a suitable basic-block driver and link module for the TSBSP software.

2. Implementation of Kent Ring Handler under POS.

2.1 PERQ-PERQ FTP via RS232.

A user interface similar to that for PUFTP has been added to the Kent BSP software. The Single-Shot-Protocol (SSP) is used for transmission of requests to send/receive files. One machine must be put into 'poll' mode while the other machine transmits requests. The 'save' and 'restore' options have been implemented.

2.1.1 BUGS

A number of tests, involving variation of various time-outs, have revealed that the Kent software is not very reliable over the RS232. It seems probable that this unreliability is due to the ring protocol dependence of the software. In particular, for the RS232 communication there is no analogue of the ring node transmission status register used to signal the successful transmission of a basic block. This results in an inability to recover from certain error conditions.

2.1.2 Timing tests

Timing tests for PERQ-PERQ file transfer have been conducted, and the results are presented below. The Kent Pascal BSP software uses a 'slow' byte stream (T.E. Schutt 1981), for which four basic-blocks are sent in order to transfer one DATA block (NOT READY, READY, NODATA and DATA). Because the resulting BSP protocol overhead is quite high (40 bytes per DATA block) figures are presented for the total (including BSP overhead) and net (useful) data transfer rates. The tests were conducted on a file of 27676 bytes, 813 lines, and the file transfer was in the Kent 'line' mode, in which the BSP assigns each line of the file to a basic block. The tests were done with the RS232 line speed set at 4800 baud.

Net data transfer rate	:	1268 bits/sec
Total data transfer rate	:	2758 bits/sec

2.2 PERQ-PDP11 FTP via RS232

The Kent software (BBP, BSP, ICP, FTP and SSP) has also been implemented on the PDP11/70, with an RS232 driver at the minipacket level. More substantial modifications to the higher level software were required for the PDP11 than for the PERQ, due to the rather rigid adherence of the Unix VU Pascal to the ISO standard.

The same 'PUFTP' user interface as for PERQ-PERQ communication has been preserved, with the addition of a login and 'start up' mechanism similar to that of the PUFTP software. The PDP11 is always awakened in the 'poll' mode.

2.2.1 BUGS

Similar limitations on reliability to those mentioned for PERQ-PERQ transfers apply also to the case of PERQ-PDP11 communication via RS232.

2.3 PERQ-Cambridge Ring Interface.

A basic block driver for the GPIB/Polynet node interface has been written and tested. This driver has been interfaced to the Kent software (ICP, BSP and FTP). File transfers between PERQ and LSI11 have been succesfully carried out, in both directions, using this software.

2.3.1 BUGS

The major limitation at present is that of the (lack of) speed of data transfer to and from the ring interface (see below).

Other bugs detected include:

- Null lines in a file abort file transfer from PERQ to LSI11 with a protocol error.

- The 'close' sequence after file transfer is sometimes aborted for file transfers to the PERQ. This is probably a timing problem, with the PERQ busy closing a file when the close block arrives.

- Protocol errors are fairly frequent. Possibly due to a mismatch of time-outs between PERQ and LSI11.

2.3.2 PERQ Hardware/Firmware Limitations.

There are a number of limitations due to the current state of the PERQ hardware/firmware. It is hoped that these deficiencies will be mitigated by the planned reprogramming of the Z80 on the PERQ I/O board, and by the advent of the new I/O board, expected late 1982.

The major shortcoming is the very slow rate of data transfer through the Z80. Three Rivers state that the data transfer rate through the GPIB is somewhat in excess of 2 Kbytes/sec. Software overheads reduce this to around 1200 bytes/sec for data transfer to the ring (the ring protocol reduces this rate by at least an order of magnitude).

A second major restriction is the size (32 bytes) of the buffers in the Z80 used for GPIB I/O. Data transfer from the ring will cause flooding of the input buffer, unless special precautions are taken. To avoid this occurrence, special commands to the GPIB controller in the PERQ are required, which delay further data transfer until the current data byte has been read. The GPIB commands also pass through the Z80, and the net result is that the data rate for receiving data from the ring interface is less than half that for transmission.

Not all the facilities of the Texas TMS9914 GPIB Adaptor chip, which is the GPIB controller in the PERQ, have been implemented in the current Z80 firmware. In particular, commands related to polling of devices on the GPIB are unimplemented. In addition, although the BI (byte in) status bit of the TMS9914 interrupt status registers is passed through to the Z80, it appears that this is not the case for all the possible interrupts. The hardware GPIB/Polynet interface was designed to pass interrupts to the PERQ using the SRQ (Service Request) GPIB control line; although there is a corresponding interrupt bit in one of the TMS9914 interrupt status registers, it seems that the Z80 firmware does not enable this interrupt. This means that reception of minipackets at the Polynet node, and error conditions on transmission, cannot be signalled to the PERQ by interrupts, and software polling must be used instead.

For the reasons stated above, a genuinely interrupt-driven basic-block driver for the PERQ is not feasible at present. Only data-byte reception

will cause a PERQ processor interrupt, and this requires a preceding GPIB command. Nevertheless, an interrupt handler for the GPIB, which can deal with data from the tablet OR the ring interface, has been written, and can be used with the basic-block driver. An alternative version of the driver, which masks out GPIB interrupts, is also available. In fact, two versions of the interrupt handler have been written, one of which uses a buffer for temporary storage of received data. The interrupt handler could be more useful when the re- programmed Z80, incorporating GPIB polling facilities, becomes available, but this may well be overtaken by the implementation of UNIX on the PERQ, for which GPIB transfers will interface to the Accent kernel.

The interrupt handler procedures have been incorporated into an alternative version of the POS module IO_Private. A small modification to IO_Init is also required, in order to initialise a boolean variable 'ring' to its default value (false - the tablet is the default GPIB device). This means that all the IO modules must be re-compiled and a new bootfile made in order to use this alternative interrupt handler.

In timing tests it has been found that there is very little difference in data transmission rate between the three different versions (one with GPIB interrupts masked out, two versions of the interrupt handler, one with an input buffer) of the basic-block driver. This implies that the limiting factor is the rate of data transfer through the Z80, and that the interrupt latency of the PERQ processor is a small perturbation on this.

2.3.3 Timing Tests

Timing tests were conducted, under similar conditions to those described in section 2.1.2, with a file of 10474 bytes, 311 lines. Results are presented below for file transfers from PERQ to LSI11 and from LSI11 to PERQ.

	PERQ -> LSI11	LSI11 -> PERQ
Net data transfer	448 bits/sec	376 bits/sec
Total data transfer	980 bits/sec	818 bits/sec

3. Implementation of GPIB basic block driver under Unix on the Perq.

The latest version of the Accent kernel incorporates code for GPIB I/O, and some of the Spice Canvas package uses this code for tablet input. GPIB interrupts are handled by the ACCENT microcode, and input data from the GPIB is placed in a circular buffer. An output buffer is used for writing to the GPIB, but this buffer can hold only 8 bytes; it is not clear whether or not there is some fundamental reason for this.

Approprite procedures to read from and write to the ring interface registers have now been written, which use the ACCENT code for GPIB I/O. These procedures have been incorporated into the ring basic-block driver, and basic-block transmission between PERQ and LSI11 has been successfully performed in both directions. This work was done using the early Unix development system. File transfer was not possible because

file I/O was not included in the stream module for this system, but there is no reason to suppose that this should cause any difficulty, now that the functionality of the basic-block driver under PERQ Unix/ACCENT has been established.

4. Ring FTP to the PDP11s.

An FTP daemon for the PDP11 is being devised by Bill Sharp. This will allow file transfers from and to a fixed directory on the PDP11. Alternatively, it may be possible for a user to start up the daemon process in his/her directory. The intention is to use SSP to handle requests to send or receive files, in a similar manner to that used to implement the 'PUFTP interface' for PERQ-PERQ and PERQ-PDP11 FTP via RS232 (see sections 2.1, 2.2). When this PDP11 FTP daemon is implemented, it will be necessary to make some modifications to the PERQ software to take advantage of the facility.

References.

T.E. Schutt, 1981: 'Byte Streams on a Minicomputer', UKC Computing Laboratory Report No.6.