Appendix C: Serial/parallel ports and printing

Chapter Contents

Introduction

You can use LPRINT in an OPL program to send information (for printing or otherwise) to any of these devices:

3 A parallel port, PAR:A

A serial port, TTY:A

A file on the Psion

3 A file on an attached computer, e.g. on a PC or on an Apple Macintosh:
OPL does not provide access to the advanced page formatting and font control features of the Series 3c.

5 OPL provides more advanced formatting features and printing control using Printer OPX. See the 'Using OPXs on the Series 5' chapter for more details.

You can also read information from the serial port.

Using the parallel port on the Series 3c and Siena

In your OPL program, set up the port with the statement `LOPEN "PAR:A"`.

Provided the port is not already in use, the connection is now ready. LPRINT will send information down the parallel 3 Link lead for example, to an attached printer.

Example

```
PROC prints:
    OPEN "clients",A,a$
    LOPEN "PAR:A"
    PRINT "Printing..."
    DO
        IF LEN(A.a$)
            LPRINT A.a$
        ENDIF
        NEXT
    UNTIL EOF
    LPRINT CHR$(12); :LCLOSE
    PRINT "Finished" :GET
ENDP
```

Using the serial port

Section Contents

In your OPL program, set up the port with the statement `LOPEN "TTY:A"`.

Now LPRINT should send information down the serial link lead for example, to an attached printer. If it does not, the serial port settings are not correct.

Serial port settings

`LOPEN "TTY:A"` opens the serial port with the following default characteristics:

9600 baud
no parity
8 data bits
1 stop bit
RTS handshaking.

If your printer (or other device) **does** match these characteristics, the LOPEN statement sets the port up correctly, and subsequent LPRINT statements will print there successfully.

If your printer **does not** match these characteristics, you must use a procedure like the one listed below to change the characteristics of the serial port, before LPRINTs will print successfully to your printer.

Printers very often use DSR (DSR/DTR) handshaking, and you may need to set the port to use this.

Setting the serial port characteristics

Calling the procedure

The `rset:` procedure listed below provides a convenient way to set up the serial port.

Each time you use an `LOPEN "TTY:"` statement, follow it with a call to the `rset:` procedure. Otherwise the LOPEN will use the default characteristics.

Passing values to the procedure

Pass the procedure the values for the five port characteristics, like this:

```
rset:(baud%,parity%,data%,stop%,hand%,&0)
```

The final parameter, which should be `&0` here, is only used when reading from the port.

To find the value you need for each characteristic, use the tables below. You **must** give values to all five parameters, in the correct order.

Baud	50	75	110	134	150	300	600	1200	1800	2000	2400	3600	4800	7200	9600	19200
value	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16

Parity	NONE	EVEN	ODD
value	0	1	2

Data bits 5, 6, 7 or 8

Stop bits 2 or 1

Handshaking	ALL	NONE	XON	RTS	XON+RTS	DSR	XON+DSR	RTS+DSR
value	11	4	7	0	3	12	15	8

The `rset:` procedure:

```
PROC rset:(baud%,parity%,data%,stop%,hand%,terms)
    LOCAL frame%,srchar%(6),dummy%,err%
    frame%=data%-5
    IF stop%=2 :frame%=frame% OR 16 :ENDIF
    IF parity% :frame%=frame% OR 32 :ENDIF
    srchar%(1)=baud% OR (baud%*256)
    srchar%(2)=frame% OR (parity%*256)
    srchar%(3)=(hand% AND 255) OR $1100
    srchar%(4)=$13
    POKEL ADDR(srchar%(5)),terms
    err%=IOW(-1,7,srchar%(1),dummy%)
    IF err% :RAISE err% :ENDIF
ENDP
```

Take care to type this program in exactly as it appears here.

Example of calling the procedure

```
PROC test:
    PRINT "Testing port settings"
    LOPEN "TTY:A"
    LOADM "rset"
    rset:(8,0,8,1,0,&0)
    LPRINT "Port OK" :LPRINT
    PRINT "Finished" :GET
    LCLOSE
ENDP
```

`rset:(8,0,8,1,0,&0)` sets 1200 Baud, no parity, 8 data bits, 1 stop bit, and RTS/CTS handshaking.

Advanced use

The section of the `rset:` procedure which actually sets the port is this:

```
srchar%(1)=baud% OR (baud%*256)
srchar%(2)=frame% OR (parity%*256)
srchar%(3)=(hand% AND 255) OR $1100
srchar%(4)=$13
POKEL ADDR(srchar%(5)),terms
err%=IOW(-1,7,srchar%(1),dummy%)
IF err% :RAISE err% :ENDIF
```

The elements of the array `srchar%` contain the values specifying the port characteristics. If you want to write a shorter procedure, you could work out what these values need to be for a particular setup you want, assign these values to the elements of the array, and then use the IOW function (followed by the error check) **exactly** as above.

Reading from the serial port

If you need to read from the serial port, you must also pass a parameter specifying terminating mask for the read function. If `terms` is not supplied, the read operation terminates only after reading exactly the number of bytes requested. In practice, however, you may not know exactly how many bytes to expect and you would therefore request a large maximum number of bytes. If the sender sends less than this number of bytes altogether, the read will never complete.

The extra parameter, `terms` , allows you to specify that one or more characters should be treated as terminating characters. The terminating character itself is read into your buffer too, allowing your program to act differently depending on its value.

The 32 bits of `terms` each represent the corresponding ASCII character that should terminate the read. This allows any of the ASCII characters 1 to 31 to terminate the read.

For example, to terminate the read when Control-Z (i.e. ASCII 26) is received, set bit 26 of `terms`. To terminate on Control-Z or <CR> or <LF> which allows text to be read a line at a time or until end of file set the bits 26, 10 and 13. In binary, this is:

0000 0100 0000 0000 0010 0100 0000 0000

Converting to a long integer gives &04002400 and this is the value to be passed in `terms` for this case.

Clearly `terms` cannot be used for binary data which may include a terminating character by chance. You can sometimes get around this problem by using `terms` and having the sender transmit a leading non-binary header specifying the exact number of full-binary data following. You could then reset the serial characteristics not to use `terms` , read the binary data, and so forth.

Example reading from serial port

This example assumes that each line sent has maximum length 255 characters and is terminated by a <CR> and that Control-Z signals the end of all the data.

```
PROC testread:
    LOCAL ret%,pbuf%,pbuf1%,buf$(255),end%,len%
    PRINT "Test reading from serial port"
    LOPEN "TTY:A"
    LOADM "rset"                    REM receive at 2400 without h/shake
    rset:(11,0,8,1,0,&04002000)     REM Control-Z or CR
    pbuf%=ADDR(buf$)                REM could be pbuf% on Series 3c
    DO                  REM read max 255 bytes, after leading count byte
        len%=255
        pbuf1%=pbuf%+1
        ret%=IOW(-1,1,^pbuf1%,len%)
        POKEB pbuf%,len%            REM len% = length actually read
                                    REM including terminator char
        end%=LOC(buf$,CHR$(26)) REM non-zero for Control-Z
        IF ret%<0 and ret%<>-43
            BEEP 3,500
            PRINT
            PRINT "Serial read error: ";ERR$(ret%)
        ENDIF
        IF ret%<>-43                REM if received with terminator
            POKEB pbuf%,len%-1      REM remove terminator
            PRINT buf$                  REM echo with CRLF
        ELSE
            PRINT buf$;                 REM echo without CRLF
        ENDIF
    UNTIL end%
    PRINT "End of session" :PAUSE -30 :KEY
ENDP
```

Note that passing -1 as the first argument to I/O keywords means that the LOPEN handle is to be used. Also, OPL strings have a leading byte giving the length of the rest of the string, so the data is read beyond this byte. The byte is then poked to the length which was read.

Printing to a file

Section Contents

Printing to a file on the Psion

In your OPL program, specify the destination file with an LOPEN statement like this:

5 `LOPEN "D:/PRINT/MEMO"`

3 `LOPEN "B:/PRINT/MEMO.TXT"`

3 Printing to a file on a PC or Apple Macintosh

As if you were going to transfer a file:

- Physically connect the Psion and the other computer.
- Select the 'Remote link' option in the System screen and press Enter.
- Run the server program (supplied with 3 Link if you are using a Series 3c or Siena, or with the Series 5 itself) on the other computer.

In your OPL program, specify the destination file with an LOPEN statement.

For example, to a PC:

```
LOPEN "REM:/C:/BACKUP/PRINTOUT/MEMO.TXT"
```

Any subsequent LPRINT would go to the file `MEMO.TXT` in the directory `/BACKUP/PRINTOUT` on the PC's drive `C:`.

With a Macintosh, you might use a file specification like this:

```
LOPEN "REM:HD40:MY BACKUP:PRINTED:MEMO5"
```

An LPRINT would now go to the file `MEMO5` in the `PRINTED` folder, itself in the `MY BACKUP` folder on the hard drive `HD40`. Note that colons are used to separate the various parts of the file specification.