PORT FE

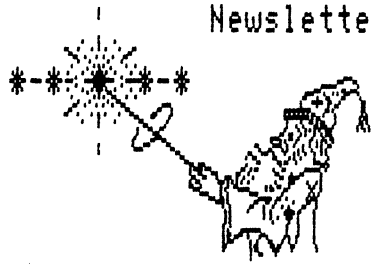
SORCERERS USERS' GROUP

(Toronto)

P.O. Box 1173 Sta. 'B'
Downsview, Ontario,
Canada. M3H 5V6

SORCERER

Newsletter



The Toronto Sorcerer Users' Group was founded in the Spring of 1979, a handful of willing and eager to learn members.

This newsletter shall at all times keep in mind the goal at its conception. To spread the seeds of knowledge.

Articles printed in this newsletter shall be free for all Sorcerer Users' groups to reprint or comment on as they see fit.

Articles submitted for this newsletter must be in no later than the beginning of the 1st of every month.

May 1982 ISSUE

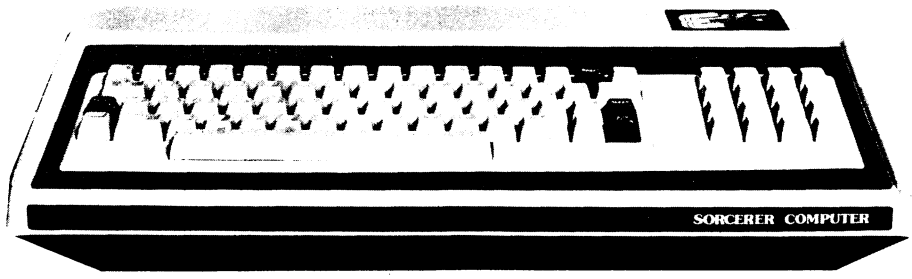
TABLE OF CONTENTS

MEETING PLACE

Location : Bathurst Heights Library - 7:00 PM 3170 Bathurst St.

May : Thurs. 13th. June : Wed. 16th.

One block north of Lawrence on the west side of Bathurst.



'C' LIBRARY ROUTINES

by Dereck Gomes

Welcome to the second installment of 'See C run'. For those of you who are not familiar with the 'C' programming language, you owe it to yourselves to check into it. It is fast becomming one of the most popular programming languages for microcomputers; and with good reason. It is a compiler oriented language with the versatility of much more expensive 'big machine' ones. However, this column is not here to extoll the virtues of 'C' but simply to provide users with a forum for library routines exchange. Read Kernighan & Ritchie's 'The C Programming Language' if you want to know more about 'C'.

This month, we have two routines dealing with scrolling up or down of any contiguous set of lines on the screen.

ROUTINE #1:

COMMENTS: This routine scrolls up a contiguous number of lines(lines) on the screen whose first line is (topline). (topline)'s range is 0 to 29, and (lines)'s range is 1 to 30.
CALLED ROUTINES: Movmem, and Setmem. (BDS C Package routines).
USAGE EXAMPLE: scrollup(0, 30); /* entire screen */

```
scrollup( topline, lines )
        int  topline, lines;
{
        char *to_spot, *from_spot, *last_line;
        int  count;

        if (topline < 0 || topline>29 || lines<1 || (topline +
                                        lines) > 30)
                return;
        if ( --lines < 0 )  lines = 0 ;
        count     = lines * 64 ;
        to_spot   = 0xf080 + ( 64 * topline );
        from_spot = to_spot + 64 ;
        last_line = to_spot + count;
        movmem( from_spot, to_spot, count );
        setmem( last_line, 64 , ' ' );
}
```

ROUTINE #2:

COMMENTS: This routine scrolls down a contiguous number of lines(lines) on the screen whose first line is (topline). (topline)'s range is 0 to 29, and (lines)'s range is 1 to 30.
CALLED ROUTINES: Movmem, and Setmem. (BDS C Package routines).
USAGE EXAMPLE: scrolldown(0, 30); /* entire screen */

```
scrolldown( topline, lines )

        int  topline, lines;
{
        char *to_spot, *from_spot, *first_line,
        int  count;

        if (topline < 0  || topline > 29 || lines < 1 ||
                        (topline + lines) > 30)
                return;
        if ( --lines < 0 )  lines = 0,
        count      = lines * 64 ;
        first_line = 0xf080 + ( 64 * topline );
        from_spot  = first_line;
        to_spot    = from_spot + 64 ;
        movmem( from_spot, to_spot, count );
        setmem( first_line, 64 , ' ' );
}
```

*** EXMON2 ***

Exidy Enhanced Monitor

Version 2.0

revised by W.S.Blady

This monitor includes the folowing features that are required by many popular application programmes.

1) Direct cursor positioning
2) Clear to end of line
3) Clear to end of screen
4) Clear entire line
5) Generate reverse ASCII set
6) Set highlight on
7) Set highlight off

Two printer drivers are included - Parallel and Serial. Both are used with the Exidy parallel and serial connectors. Each driver has two jump vectors - allow or disallow linefeeds. The serial driver has a reserved byte in the MWA 'waitn', for a delay value used after a Carriage Return. This is used for printers with slow carriages. The value must be poked in.

Two useful commands have been added to help those interested in fiddling with hex. code.

1) Fill memory with a byte
2) Search memory for a byte or word - location address is printed to the output device.

A fast keyboard status check routine has been added. When called, 0ffh is returned in 'A' and the ZERO flag reset when a key has been pressed. 00h is returned in 'A' and the ZERO flag set if no key has been pressed.

A more comprehensive jump table has been set up to include the most often used Exidy routines.

The top of screen may be defined. Any number of lines may be reserved at the top of screen, and scrolling or cursor movements will not interfere with this area. The only way to access this area is through direct cursor positioning.

Other additions are outlined below.

NEW SET OUTPUT COMMANDS:

```
SEt     O=V...VIDEO
          P...PARALLEL
          S...OUTAPE
NEW       N...CENTRONICS DRIVER, NO LINE FEEDS
NEW       F...CENTRONICS DRIVER, LINE FEEDS
NEW       D...SERIAL DRIVER, NO LINE FEEDS
NEW       L...SERIAL DRIVER, LINE FEEDS
```

NEW SET COMMANDS.

```
SEt     H=n.....SET TOP OF SCREEN (n must be in hex)
        G=R.....SET REVERSE ASCII
        G=N.....RESTORE STANDARD USER GRAPHICS
```

The top of screen can also be set by calling 'highsc' with the number of reserved lines desired, in register 'E'.

The cursor character may be changed by putting the value in register 'A' and calling 'cursor'. Or the value can be poked directly into the proper MWA location.

NEW SCREEN EDITING COMMANDS.

```
'ESC' = Ln Cn.....DIRECT CURSOR POSITION
'ESC'0C2.........CLEAR TO END LINE
'ESC'0C3.........CLEAR TO END SCREEN
'ESC'0C4.........CLEAR LINE
'ESC'0C5.........HIGHLIGHT ASCII ON
'ESC'0C6.........HIGHLIGHT ASCII OFF
'ESC'0C7.........LOAD REVERSE ASCII SET (destroys all user graphics)
'ESC'0C8.........RESTORE USER GRAPHICS
```

NEW COMMANDS:

```
FL      (fill)...(from address) (to address) (byte)
SR      (search).(from address) (to address) (byte/word)
```

NEW (MWA) STORAGE:

```
CRSFLG..ESCAPE SEQUENCE FLAGS (4 USED)
SCRTOP..OFFSET TO TOP OF SCREEN (n * 64)
WAITN...NULL VALUE FOR SERIAL PRINTERS WITH SLOW 'CR'
CRSCHR..CURSOR CHARACTER
```

Two original routines have been sacrificed to make room for these new features. They were considered by most to be of limited value.

1) Memory test (GONE)
2) Batch mode (GONE)

==========

CASSETTE DATA STORAGE FOR THE SORCERER

by Steve Dicker

For Sorcerer owners who haven't yet purchased disc drives, one of the biggest limitations you may encounter is the inability of the Sorcerer to write alphanumeric data to tape. Unfortunately, Exidy didn't include a command to do this in the Rom-Pac BASIC. There is a CLOAD* command, but using this to write alphanumeric data requires that the data first undergo a time-consuming transformation into numeric form (its ASCII equivalent, for example). This also means that extra memory space is consumed.

There is a method of writing data to the cassette that is relatively fast and simple to implement. I have been using it for some time and I've found it to be very reliable despite the fact that no CRC checking is performed. This method of cassette storage is based on the ability of the Sorcerer to choose the source of input and destination of output data. This may be done by using the Monitor SET command. For example, the command: SET O=S will direct succeeding data to be sent to the cassette output port until a new SET O= is issued.

The SET O and SET I commands merely change a pointer in the Sorcerer's Monitor Work Area. These pointers indicate where the appropriate input/output routine is located within the Monitor. Therefore, if you know the address of these pointers and if you know the address of the desired I/O routine you wish to use, you can POKE the necessary values in and control the flow of data from BASIC. Table 1 gives the required POKE addresses for 8, 16 and 32K machines and table 2 lists the values which must be POKED to select the desired I/O device.

Table 1 I/O POINTER ADDRESSES

MEM. SIZE	OUTPUT P1	OUTPUT P2	INPUT P1	INPUT P2
8K	8144	8145	8146	8147
16K	16336	16337	16338	16339
32K	32720	32721	32722	32723

Table 2 I/O ROUTINE ADDRESSES

DEVICE	INPUT FROM P1	INPUT FROM P2	OUTPUT TO P1	OUTPUT TO P2
Printer	-	-	147	233
Screen	-	-	27	224
Cassette	15	224	18	224
Keyboard	24	224	-	-

As an example of the use of these tables, assume that you wish to output to the cassette on a 32K machine. The appropriate POKE instructions would be:

POKE 32720,18:POKE 32721,224

Having done this, you may now send data to the tape using the standard BASIC PRINT statement. This data will be written to the tape in the same format as it would have been sent to the screen. In the same manner, you may direct the computer to input from the cassette and then utilize INPUT statements to read data in. The format in which the data is written to the tape is entirely up to the user. It's up to you to decide how the data file will be organized. The only requirement is that the PRINT/INPUT formats match.

Listing 1 shows an example program using this technique to write and read alphanumeric data. The file format I use (see fig. 1) is to write fifty nulls, followed by the file name, followed by a pause for a count of fifty, the data in the file (each entry followed by an APPROPRIATE pause when written) and the string "END" to mark the end of the file. When the file is read back in, I first look for a null, then the correct file name. The pause after the name ensures that I have time to verify that the name is correct without losing any of the succeeding information. Since all data are PRINTed to the tape, even numeric data are written as strings. Therefore, these must be transformed back into numeric data using the VAL function, as shown in the example program. The pause after each PRINT allows time for this type of processing between INPUTs.

The program may read a fixed number of entries from the tape (as it does here) or it may read until the END marker is found by an appropriate test. Thus the file length may be fixed or variable, as desired. One final point to note is that the program returns control to the keyboard if the first file found does not have the appropriate name. If you decide not to incorporate this feature into your programs, you may get caught in a situation where your program is looking for a file which does not exist (i.e. you entered the name incorrectly) and you can't regain program control.

When you run the example program you'll discover that data are printed on the screen as they are INPUT. This is a bonus feature of this storage method that I like. I can watch the data as it is read in and determine if the data is being read properly or how far along I am in the file. You'll also notice some of the shortcomings of this system. First, strings with colons(:), commas(,) and some other special characters are truncated at these characters when the data are read back in. Second, all data are preceded by a line feed (ASCII 10) when they are read back. The PRINT statement ends all output with a carriage return/line feed combination. (ASCII 13/10). However, INPUT reads until it finds a carriage return. The line feed is then left to be read as the first character of the next piece of data. Fortunately, both of these problems can be remedied through the use of a small machine language subroutine.

One final note about listing 1. You may find the subroutine starting at line 10000 useful in your own programs. This subroutine determines the proper POKE addresses of the I/O pointers for a Sorcerer with any memory size. The output pointer addresses are in 01 and 02 while those for the input pointer are stored in I1 and I2. The use of this subroutine allows your program to run properly without modification, should you ever expand the memory in your Sorcerer.

The machine language subroutine I mentioned above is given in listing 2. This is basically L.H. Daniels' program LINEIN ("TRS Text Formatter", Microcomputing, July 1980, p. 73), modified to run on the Sorcerer. You may wish to refer to it for a complete description of the operation of this subroutine.

When called, the subroutine accepts input from the current input device (i.e. keyboard, cassette, etc.) This input is used to build the string I$ and it is through this string that the BASIC program communicates with the subroutine. The subroutine allows characters such as ":; to appear in the string and it filters out all control characters. A carriage return is used to terminate input and return con-trol to BASIC. In addition, the subroutine allows the use of the RUB key without an automatic line feed occuring after a given number of key-strokes. Control X may be used to erase the entire line and restart input.

The subroutine is set up to LOAD and run at 0000. Listing 3 gives the BASIC instructions required to POKE the subroutine into memory. Should you wish to relocate the routine, you need only alter the address of subroutine BUILD (line 62, listing 2) and the address of STRING (line 33).

Listing 4 shows the modifications which should be made to the program in listing 1 to incorporate the machine language subroutine. The lines shown should be typed in after the original program has been loaded. Besides these changes, the subroutine at 11000 (from listing 3) should be added and line 2030 should be DELETED. When you run this modified program, try entering strings with commas and colons within them to verify that they may now be used.

When you begin to experiment with these routines in programs of your own you may find that the program in listing 5 comes in handy. Record the output from this program at the start of the tape on which you will record your programs as you develop them. The program writes the command:

POKE I1,24:POKEI2,224

onto tape twice (where I1 & I2 are the appropriate values for your memory size). Should you lose control of your program when it is expecting input from the cassette, play the tape and you should be able to regain control of your Sorcer-er. It may save you many frustrating hours of loading & reloading your program.

With a little experimentation you should be able to find many uses for the I/O technique described here. As I mentioned before, I have found it to be highly reliable so I haven't bothered cross-checking data as it is read in (for example, the Monitor LO command does this with CRC checking). You may like to experiment in this area yourself.

Of course, there is still the problem of cassette being a slow storage med-ium. You'll probably still find yourself dreaming of disc storage but at least you won't be having any nightmares about how you are going to store data in the meantime.

10 REM *** INITIALIZATION ***

Listing 1

```
20 CLEAR 200
30 DIM T$(5)
40 GOSUB 10000:REM - SET UP I/O POKE ADDRESS POINTERS
50 :
100 PRINT CHR$(12):PRINT
110 PRINT "CASSETTE I/O TEST PROGRAM"
120 PRINT
130 REM - GET TEST DATA
140 INPUT "GIVE ME A NUMBER";NU
150 FOR I=1 TO 5
160 :     PRINT:INPUT "ENTER ALPHANUMERIC DATA";T$(I)
170 NEXT I
180 :
190 GOSUB 1000:REM - SAVE THE DATA ON TAPE
200 :
210 REM - CLEAR ALL VARIABLES
220 NUM=0
230 FOR I=1 TO 5
240 :     T$(I)=""
250 NEXT I
260 :
270 PRINT "ALL VARIABLES HAVE BEEN CLEARED:":PRINT
280 PRINT "NU=",NU
290 FOR I=1 TO 5
300 :     PRINT "T$(";I;")= ",T$(I)
```

Fig.1 Data File Format

50 NULLS
FILE NAME
PAUSE FOR COUNT OF 30
DATA
PAUSE
DATA
PAUSE
.
.
DATA
PAUSE
"END"

```
310 NEXT I
320 PRINT:INPUT "PRESS RETURN TO CONTINUE";Z$
330 :
340 GOSUB 2000:REM - LOAD THE DATA FROM TAPE
350 :
360 REM - DISPLAY DATA LOADED FROM TAPE
370 PRINT "HERE IS THE DATA LOADED FROM TAPE:"
380 PRINT:PRINT "NU=";NU
390 FOR I=1 TO 5
400 :     PRINT "T$(";I;")= ";T$(I)
410 NEXT I
420 PRINT "*** TEST COMPLETED ***":PRINT
439 END
440 :
1000 REM - WRITE DATA TO CASSETTE
1001 :
1010 PRINT CHR$(12)
1020 INPUT "FILE NAME";FI$:       REM - GET A NAME FOR THE FILE
1030 IF FI$="" THEN 1020:         REM - NAME CAN'T BE A NULL
1040 PRINT:PRINT "PRESS PLAY AND RECORD ON CASSETTE"
1050 INPUT "THEN PRESS RETURN TO SAVE DATA";T$(0)
1060 PRINT:PRINT "WRITING DATA": REM - TELL OPERATOR WHAT'S DOING
1070 POKE 01,18:POKE 02,224:      REM - SEND OUTPUT TO CASSETTE
1080 FOR I=1 TO 50:               REM - WRITE 50 NULLS
1090 :     PRINT CHR$(10)
1100 NEXT I
1110 PRINT FI$:                   REM - WRITE FILE NAME
1120 FOR I=1 TO 50:NEXT I:        REM - PAUSE AFTER NAME
1130 PRINT NU:                    REM - WRITE NUMERIC DATA
1140 FOR I=1 TO 10:NEXT I:        REM - PAUSE AFTER DATA
1150 FOR I=1 TO 5:                REM - WRITE ALPHANUMERIC DATA
1160 :     PRINT T$(I)
1170 :     FOR J=1 TO 10:NEXT J:  REM - PAUSE AFTER DATA
1180 NEXT I
1190 PRINT "END":                 REM - WRITE END MARKER
1200 POKE 01,27:POKE 02,224:      REM - SEND OUTPUT TO SCREEN
1219 RETURN
1220 :
2000 REM - READ DATA FROM CASSETTE
2001 :
2010 PRINT CHR$(12)
2020 PRINT "FILE NAME",FI$:       REM - FILE TO SEARCH FOR?
2030 FI$=CHR$(10)+FI$:            REM - ADD LINE FEED TO NAME
2040 PRINT :PRINT "CUE TAPE TO FILE, PRESS PLAY AND"
2050 INPUT "PRESS RETURN TO LOAD DATA";T$(0)
2060 POKE I1,15:POKE I2,224:      REM - GET INPUT FROM TAPE
2070 PRINT "SEARCHING FOR ";CHR$(23);FI$
2080 INPUT T$(0):                 REM - SEARCH FOR LEADING NULLS
2090 IF LEN(T$(0))<>1 THEN 2080
2100 INPUT T$(0):                 REM - SEARCH FOR FILE NAME
2110 IF LEN(T$(0))=1 THEN 2100
2115 REM - TELL OPERATOR WHAT'S DOING
2120 PRINT "FOUND ";RIGHT$(T$(0),LEN(T$(0))-1)
2125 REM - IF WRONG FILE NAME, ABORT LOAD
2130 IF T$(0)<>FI$ THEN 2190
2140 PRINT "LOADING...":          REM - TELL OPERATOR WHAT'S DOING
2150 INPUT T$(0):                 REM - GET NU
2160 NU=VAL(RIGHT$(T$(0),LEN(T$(0))-1))
2170 FOR I=1 TO 5:                REM - GET ALPHANUMERIC DATA
2180 :     INPUT T$(I)
2190 NEXT I
2200 POKE I1,24:POKE I2,224:      REM - BACK TO KEYBOARD INPUT
2219 RETURN
```

```
10000 REM -- DETERMINE POKE ADDRESSES OF I/O POINTERS
10001 :
10010 REM - GET ADDRESS OF TOP OF RAM
10020 RT=PEEK(-4096)+256*PEEK(-4095)
10030 FS=RT-47:REM - CALCULATE START ADDRESS OF POINTERS
10040 O1=PS.O2=PS+1:I1=PS+2:I2=FS+3:REM - SET ADDR. POINTERS
10049 RETURN
```

```
0000                00000 ;         Listing 2 -- INPUT LINE ROUTINE
0000                00002 ;-------------------------------------------------------
0000                00003 ,  THIS ROUTINE WILL ACCEPT LINES WITH COMMAS AND
0000                00004 ;  COLONS IMBEDDED WITHIN THE LINE. IT ALSO ALLOWS
0000                00005 ;  EDITING  OF THE LINE WITHOUT LIMIT (IE. NO AUTOMATIC
0000                00006 ;  MATIC CARRIAGE RETURN
0000                00007 ;-------------------------------------------------------
0000                00009 ;       EXTERNAL REFERENCES
0000                00010 ;
      CDA0          00011 VARPTR EQU  #CDA0      ;VARIABLE SEARCH ROUTINE
      E009          00012 INPUT  EQU  #E009      ;USER INPUT ROUTINE
      E205          00013 CRLF   EQU  #E205      ;PRINT(CR),(LF)
      E01B          00014 VIDEO  EQU  #E01B      ;VIDEO OUTPUT ROUTINE
0000                00016 ;-------------------------------------------------------
0000                00017 ;       EQUATE TABLE
0000                00018 ,
      000D          00019 (CR)   EQU  #D         ;ASCII CODE FOR CARRIAGE RET
      007F          00020 (RUB)  EQU  #7F        ;  "     "     "  RUB
      0018          00021 CTRLX  EQU  #18        ;  "     "     "  CONTROL-X
      0008          00022 (DEL)  EQU  8          ;  "     "     "  DELETE
0000                00024 ;=======================================================
0000  F5            00026 LINEIN PUSH AF         ,SAVE 'EM
0001  C5            00027        PUSH BC
0002  D5            00028        PUSH DE
0003  E5            00029        PUSH HL
0004  01 80 49      00030        LD   BC,#4980   ;BC="I$"
0007  CD A0 CD      00031        CALL VARPTR     ,POINT DE TO LOCATION OF I$ DATA
000A  EB            00032        EX   DE,HL      ;PUT POINTER IN HL
000B  11 5B 00      00033        LD   DE,STRING  ;POINT DE TO I$ TEMP. STORAGE
000E  AF            00034        XOR  A          ,SET STRING LENGTH
000F  77            00035        LD   (HL),A     ;  TO 0
0010  23            00036        INC  HL         ;SET STRING LOC. PNTR TO
0011  23            00037        INC  HL         ,  TEMPORARY STORAGE
0012  73            00038        LD   (HL),E     ;  SPACE FOR I$
0013  23            00039        INC  HL
0014  72            00040        LD  (HL),D
0015  2B            00041        DEC  HL         ;POINT TO STRING LENGTH
0016  2B            00042        DEC  HL
0017  2B            00043        DEC  HL
0018  CD 09 E0      00044 REPEAT CALL INPUT      ;GET INPUT FROM KEYBRD OR TAPE
001B  28 FB         00045        JR   Z,REPEAT   ,WAIT FOR DATA
001D  FE 0D         00046        CP   (CR)       ;ON (CR), PRINT A
001F  20 08         00047        JR   NZ,RUB     ,  (CR),(LF) AND
0021  CD 05 E2      00048        CALL CRLF       ,  RETURN WITH I$
0024  E1            00049        POP  HL
0025  D1            00050        POP  DE
0026  C1            00051        POP  BC
0027  F1            00052        POP  AF
0028  C9            00053        RET
0029                00054        ;
0029                00055        ,
```

```
0029  FE 7F        00056 RUB    CP   (RUB)      ,ON (RUB), ERASE
002B  28 1C        00057        JR   Z,ONEBK    ,  ONE CHARACTER
002D  FE 18        00058        CP   CTRLX      ;ON CONTROL-X ERASE
002F  28 15        00059        JR   Z,POLYBK   ;  THE LINE
0031  FE 20        00060        CP   " "        ;IGNORE ANY OTHER
0033  38 E3        00061        JR   C,REPEAT   ,  CONTROL CHARACTERS
0035  CD 3C 00     00062        CALL BUILD      ;ADD CHARACTER TO THE LINE
0038  20 DE        00063        JR   NZ,REPEAT  ;GO BACK FOR ANOTHER
003A  18 0D        00064        JR   ONEBK      ;ERASE ONE IF AT END OF LINE
003C               00065        ;
003C               00066        ;
003C  12           00067 BUILD  LD   (DE),A     ;ADD A CHR TO THE LINE
003D  CD 1B E0     00068        CALL VIDEO      ;ECHO TO THE SCREEN
0040  13           00069        INC  DE         ,POINT TO NEXT AVAIL. STORAGE
0041  34           00070        INC  (HL)       ;INCREASE STRING LENDTH
0042  7E           00071        LD   A,(HL)     ,IF TOO LONG, SET Z FLAG
0043  FE 40        00072        CP   64         ,  (MAXIMUM LENGTH IS 64 CHR'S)
0045  C9           00073        RET
0046               00074        ;
0046               00075        ;
0046  46           00076 POLYBK LD   B,(HL)     ;SET B=NO. CHR'S IN I$ (ERASE
0047  18 02        00077        JR   ZEROCH     ;  THE ENTIRE STRING)
0049  06 01        00078 ONEBK  LD   B,1        ;DELETE A CHR
004B  7E           00079 ZEROCH LD   A,(HL)     ;CHECK IF LENGTH IS ZERO
004C  FE 00        00080        CP   0
004E  28 C8        00081        JR   Z,REPEAT   ;CAN'T DELETE IF NONE
0050  3E 08        00082 BKSPC  LD   A,(DEL)    ;DELETE THE CHR ON
0052  CD 1B E0     00083        CALL VIDEO      ;  THE SCREEN
0055  1B           00084        DEC  DE         ;MOVE STORAGE PNTR BACK
0056  35           00085        DEC  (HL)       ;DECREMENT STR. LENGTH
0057  10 F7        00086        DJNZ BKSPC      ;REPEAT "B" TIMES
0059  18 BD        00087        JR   REPEAT     ;GO BACK FOR INPUT
005B               00088 ;
005B               00089 ;      RESERVED STORAGE SPACE FOR I$
005B               00090 ;
005B  00           00091 STRING RSVR 65
009C               00092 ;
009C               00093 ;
009B               00094 END    EQU  $-1        ;ADDR. OF END OF STORAGE SPACE
```

SYMBOL TABLE

```
(DEL)    0008    (RUB)    007F    (CR)     000D    BKSPC    0050
BUILD    003C    CTRLX    0018    CRLF     E203    END      009B
INPUT    E009    LINEIN   0000    ONEBK    0049    POLYBK   0046
RUB      0029    REPEAT   0018    STRING   005B    VIDEO    E01B
VARPTR   CDA0    ZEROCH   004B
```

Listing 3

```
11000 FOR PK=0 TO 90
11010 READ PV
11020 POKE PK,PV
11030 NEXT PK
11040 RETURN
11050 DATA 254,197,213,229,1,128,73,205,160,205,235,17,91,0
11060 DATA 175,119,35,35,115,35,114,43,43,43,205,9,224,40,251
11070 DATA 254,13,32,8,205,5,226,225,209,193,241,201,254,127
11080 DATA 40,28,254,24,40,21,254,32,56,227,205,60,0,32,222
11090 DATA 24,13,18,205,27,224,19,52,126,254,64,201,70,24,2,6
11100 DATA 1,126,254,0,40,200,62,8,205
11110 DATA 27,224,27,53,16,247,24,189
```

Listing 4

```
50 GOSUB 11000:REM - LOAD "LINEIN" ROUTINE
60 POKE 260,0:POKE 261,0:REM - SET USER ROUTINE POINTER
160 :    PRINT:PRINT "ENTER ALPHANUMERIC DATA: ";
165 Q=USR(Q):T$(I)=I$:REM - CALL "LINEIN" TO GET STRING
1020 PRINT "FILE NAME. ";:Q=USR(Q):FI$=I$
1050 PRINT "THEN PRESS RETURN TO SAVE DATA":Q=USR(Q)
2020 PRINT "FILE NAME: ";:Q=USR(Q):FI$=I$
2050 PRINT "PRESS RETURN TO LOAD DATA":Q=USR(Q)
2070 PRINT "SEARCHING FOR ";FI$
2080 Q=USR(Q):                    REM - SEARCH FOR LEADING NULLS
2090 IF I$()"" THEN 2080
2100 Q=USR(Q):                    REM - SEARCH FOR FILE NAME
2110 IF I$="" THEN 2100
2120 PRINT "FOUND ";I$
2130 IF FI$()I$ THEN 2190
2150 Q=USR(Q):                    REM - GET NU
2160 NU=VAL(I$)
2180 .       Q=USR(Q):T$(I)=I$
READY
PRINT CHR$(19)
```

Listing 5

```
10 REM - SUBSTITUTE THE ACTUAL NUM. VALUES OF I1 & I2 IN CD$
20 REM - BELOW. THESE VALUES MAY BE OBTAINED FROM TABLE 1
30 REM - (INPUT COLUMN - P1 & P2 VALUES).
50 :
100 CD$="POKE I1,24:POKE I2,224"
110 PRINT "PRESS PLAY & RECORD ON THE TAPE RECORDER"
120 INPUT "AND PRESS RETURN";Z$
130 .
135 FOR NT=1 TO 2:   REM - WRITE IT TWICE
140 :    FOR I=1 TO LEN(CD$)
145 :         REM - WAIT FOR TRANSMIT BUFFER TO CLEAR
150 :         WAIT 253,1
155 :         REM - SEND CHR TO THE CASSETTE
160 :         OUT 252,ASC(MID$(CD$,I,1))
170 :    NEXT I
175 .    REM - WRITE A (CR) AT THE END OF THE LINE
180 :    WAIT 253,1
190 :    OUT 252,13
200 NEXT NT
210 :
220 PRINT "FINISHED"
299 END
```

========== ========== ==========

NEW MONITOR DEMO

At the next meeting we will be demonstrating the new monitor version II and all of the new features that it has. So for those of you who wish to see it in action come out to a meeting for a change. For those of you that have wanted to use Calcstar or any of the canned software that requires cursor positioning routines, this is the answer to your prayers.

The next meeting after June we will be letting you all know of the dates for subsequent meetings.

The Prez

Interfacacing the Sorcerer to a Z80 SIO

by Brad Fowles

Have you ever wished for a serial PORT that was not tied to the keyboard?

The purpose of this two part article is to give a brief overview of the Z80 SIO and interfacce it to a non S-100 Sorcerer (an S-100 card would be an easy dirivative).
Due to the complexity of this chip, it is assummed you would have the DATA SHEETS which ZILOG makes readily available where you buy the chip.

OVERVIEW

The Z80 SIO is a programmable DUAL channel device which provides formats for almost any mode of serial data communication. It is capable of handling asynchronous, synchronous bit oriented protocals ie. IBM Bi Sync, HLDC & SDLC. It is capable of data rates from 0-500K, and can be interrupt driven. With the exception of a simple decoder select chip and a variable clock rate generator it is virtually a stand alone device. A look at the block diagram, shows its hardware simplicity.

OTHER GENERAL FEATURES

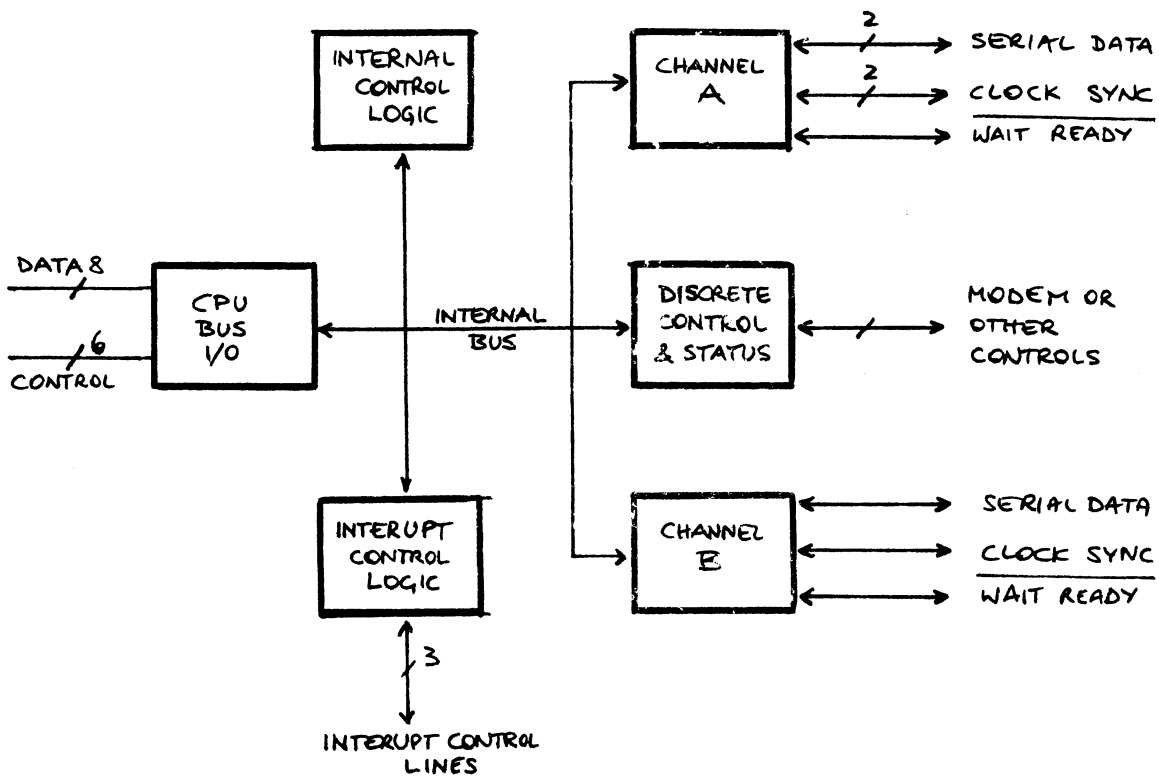
Asynchronous Operation

- 5,6,7 or 8 bit / character
- 1,1.5 or 2 stop bits
- even, odd or no parity
- x1,x16,x32 and x64 clock modes
- break generation & detection
- parity, overun & framing error detect

Binary Synchronous Operation

- internal or external character sync.
- 1 or 2 sync char. in separate reg.
- automatic sync. char. insertion
- CRC generation and detection
- all inputs fully TTL compatible

Next month the hardware connection to the Sorcerer and a simple terminal program which should provide (along with the data sheets) enough information for you to get more serial ports up and running for your particular application.



Membership Application Form Covering Jan. to Dec.1982

Membership to the group is not restricted to the TORONTO area. All persons willing to participate are invited to join.

As a member of the Sorcerer Users' Group (Toronto), I enclose the annual membership fee and agree to the following Terms.

1. That I will not, without the authorization of the board of directors, represent myself or take any action as agent, or representative or become spokesperson of the group.

2. That I will not use any software obtained from the SUGT library for any commercial purpose or financial gain. The library shall be available to me should I wish to obtain programs donated by other members. These programs shall not be distributed without the owners consent and/or the consent of the board of executive officers.

3. That I have the right to vote for the officers and directors of the organization at the annual general meeting.

4. That any breach of the above conditions and any other restrictions that the Officers of the Club may invoke in the future on my part may result in suspension or termination of my membership without refund.

Annual Membership Rates: (Jan - Dec)

Canadian - $15.00 Cdn - PLUS $6.00 Postage
U.S.& Foreign $15.00 (U.S Funds) PLUS $10.00 Postage

Payable to - SORCERER USERS' GROUP (TORONTO) - by Cheque or Money Orders.

The SUGT program library is available to all members in the following manner.

You may send $6.00 + $1.50 postage for each volume as they become available and we shall supply the cassette/s. Program cassettes shall be sent via Air Mail.
All issues of PORT FE shall be mailed first class, in the case of non local issues, they are mailed via Air Mail. Past issues of PORT FE are only available for the current calendar year. Contact the editor, he will advise the amount of payment for previous issues.

NAME(print):.....
ADDRESS:.....
CITY:.....
POSTAL CODE:.....
TELEPHONE. Res..... Bus.....

Payments enclosed (membership)..... Library tape/s.....Vol 1 or 2

Signature:.....

Please list the type of equipment you are using etc...
Sorcerer size: 8... 16... 32... 48... other..... 5100... Graph board.....
Disk system - Micropolis..... Discus.... Exidy.... other... Size.....
Other Equipment .....
.....

If you belong to any other Sorcerer Users' Group please list it below.
.....