PORT FE

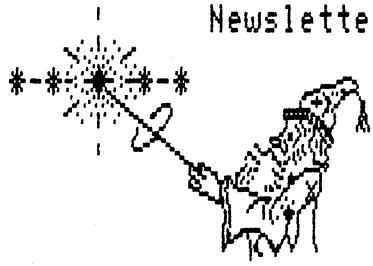
SORCERERS USERS' GROUP

(Toronto)

P.O. Box 1173 Sta. 'B'
Downsview, Ontario,
Canada. M3H 5Y6

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.

August 1982 ISSUE

TABLE OF CONTENTS

==========

==========

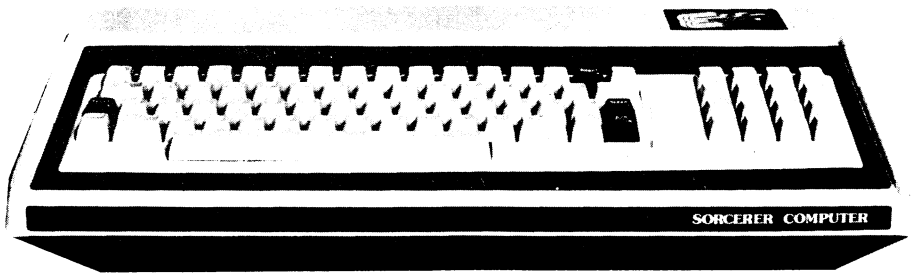
MEETING PLACE

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

----- Thur. Sept. 16 -----

Thur. Oct. 14 Thur. Nov. 18 Wed. Dec. 15

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



'C' LIBRARY ROUTINES by Dereck Gomes

This is our fourth installment of our 'C' library routines and, this month, we will be looking at two routines that provide us with inverse video of the normal ASCII character set. Routine #1 must first be used to create the inverse character set of ASCII characters which starts at F800H and put them in the standard and programmable graphics section, then routine #2 will refer to them simply by adding 128 to the ASCII code of the character requested. This is sufficient for most applications, but if you wish to use your graphics section for other uses, then you will have to use another method which will involve creating the inverse characters on the fly and keeping them in a graphics character cell for the duration of its need; a rather more cumbersome method. The method listed in these two routines is more elegant and faster, since you do not have to go through a routine to create it every time you want to use an inverse character; just ask for it.

Note, however, that if you use a routine that uses Exidy's CLEAR SCREEN routine then your graphics section will be re-written and you will have to invoke Invascii() again. It is best to use your own routine to clear the screen.

ROUTINE #1.

COMMENTS. This routine creates an inverse copy of the ASCII character set which resides at F800H and puts it in the graphics section for use by routine Inverse().
CALLED ROUTINES: None.
USAGE EXAMPLE: invascii();

```
invascii()              /* setup graphics with inverse of ASCII */
{
        char *graph;


        for (graph = 0xfc00; graph ; graph++)
        {
                *graph = ~ ( *( graph - 1024) ),
        }
}
```

ROUTINE #2:

COMMENTS: This routine changes a string of ASCII characters pointed to by (ptr) into its inverse video equivalent. Routine Invascii() is a prerequisite for this routine.
CALLED ROUTINES: None.
USAGE EXAMPLE. putdma(2, 3, inverse("this is a string"));

```
inverse( ptr )

        char *ptr;
{
        char *temp,
        temp = ptr;
        while ( *ptr )
        {
                *ptr = *ptr | 0x80;
                ptr++;
        }
        return( temp );
}
```

SIDE 3A — FORT FE Library Tape No.3 — August 1982

01	AVOID	Steer through an accelerating meteor shower, it's hairy!. Written by Stephen Cousins (ESC)	(CLOAD)
02	CHASE	Wipe out the robots in the maze before they wipe you out. Donated from Australia	( )LOG)
03	EVADE	Kill the enemy in the maze, then find your own way out. Donated from Australia	( )LOG)
04	FLITE	Similar program to the TRS80 Flight Simulator for Sorcerer.	( )LOG)
5a	LAZER	Two player game of 2 spaceships in conflict with each other.	(CLOAD)
5b	LAZER	Graphic characters for LAZER. This would be good in M/L!!! Donated from Australia	( )LO)
06	ELIZA	Logically speaking, have a conversation with a computer. Written by Dave Bristor	(CLOAD)
07	ZAP	Can you steer the robots into the highly electrified pylons. Written by R.G.Ruh	(CLOAD)
08	BKOUT	A new era for Breakout, utilizing the Sorcerer's graphics. Written by Ray Bannon	(CLOAD)
09	ROSE	Creates a random selection of graphic designs to the screen. Written by Ken Jackman, et al.	(CLOAD)
10	QUANT	This unique program is fast pixel manipulation at its best. Writer unknown (Pity you say)	( )LOG)
11a	DUMAP	An adventure game with graphics. Try to get the princess -	( )LO)
11b	KEYBD	and find as much treasure as you possibly can, then return -	( )LO)
11c	DUNG2	to the surface with the princess and some treasure to win. Written by Paul Taylor (PORT FE)	(CLOAD)
12a	DTASM	Now transfer programs between 2 Sorcerers via the parallel -	( )LO)
12b	DTCOM	port, this will explain how.(compl.) Documentation included. Written by Bob Stafford / David Woodberry	( )LO)

SIDE 3B — EUROPEAN SORCERER GEBRUIKERS GROUP (ESGG Tape No.1 - 1982)
(Dutch to English version)

01	INFO	Introduction to ESGG tape No.1	(CLOAD)
02	CONTS	Contents of this tape No.1	(CLOAD)
03	INFO1	Information about program CRC	( )LO)
04	CRC	Tuning the tape-recorder	(CLOAD)
05	INFO2	Information about program COPIE	(CLOAD)
06	COPIE	Program for copying tapes	(CLOAD)
07	INFO3	Information about program LINK	( )LO)
08	LINK	Linking two programs	(CLOAD)
09	INFO4	Information about program RENBR	(CLOAD)
10	RENBR	Renumber program	(CLOAD)
11	INFO5	Information about program EDIT	( )LO)
12	EDIT	Basic line editor	(CLOAD)
13	GRAPH	Making graphic characters	(CLOAD)
14	HISTA	Making histo- or bar-diagrams	(CLOAD)
15	CHRDS	Making diagrams	(CLOAD)
16	HELP	Calculations for conversions	(CLOAD)
17	NRNOT	Conversion to other number notations	(CLOAD)
18	DEHEB	Conversion of number notations	(CLOAD)
19	ENLAR	Enlarging characters	(CLOAD)
20	LITEP	Light-paper	(CLOAD)
21	LITPP	Another light-paper program	(CLOAD)
22	LOGTA	Logarithm-table	(CLOAD)

PRICE $8.00 U.S. + $1.50 Postage for each tape sold. Payable to the following:-
Sorcerer Users' Group (TORONTO) - by Cheque,Money Order.

FARTIAL TAPE NO.4

SIDE 4A THE FOLLOWING FROGRAM IS AVAILABLE FOR $6.00 U.S. Plus $1.50 Fostage

23	TICAR	Calculation of trade-in value of cars	(CLOAD)
24	TEX1	Kind of wordpack	(CLOAD)
25	CHKSM	Checksum to check your computer	(CLOAD)
26	CASH	Financial program	(CLOAD)
27	RENUM	Renumbering your program	(CLOAD)
28	CURAD	Cursor-control	(CLOAD)
29	ENLCH	Make-up of characters	(CLOAD)
30	PLOT1	Plotting	(CLOAD)
31	PLOT2	Plotting	(CLOAD)
32	SINUS	Plotting	(CLOAD)
33	CROSS	Plotting	(CLOAD)
34	FUTUR	Futuristic headlines	(CLOAD)
35	PRINT	Printer routine	(CLOAD)
36	LOTTO	Lottery aid	(CLOAD)
37	TEXT2	A kind of wordpack	(CLOAD)
38	POST1	Aid for corrispondence	(CLOAD)
39	GRAF8	Goniometric functions	(CLOAD)
40	MINE	Searching for minerals, a game	(CLOAD)
41	MINEI	Information belonging to MINE	(CLOAD)
42	DEFIG	Make-up figures	(CLOAD)
43	CALDR	Calender program for 1900 to 2100	(CLOAD)
44	SOLIT	Solitaire, a game with pawns	(CLOAD)
45	FIN3	Financial program	(CLOAD)

THE END OF ESGG-TAPE

[[]]]

If for any valid reason, some of these programs should be EXCLUDED from our library tape, please write to Toronto Sorcerers User's Group and the program in question will be deleted. For those of you to whom this may concern please accept our apology for including any such program. It is our sincere concern that we do not offend anyone. All programs listed are assumed to be public domain programs. Please send any comments to my attention. Thank you.

H.A. Lautenbach Fres.

[[]]]

EXMON2 and the WORD FROCESSOR FAC - From Norman Olson

The Word Processor Pac can be made compatible with EXMON2. We have found the nesessary addresses that need to be changed. For those of you that are running the PAC either relocated for disk operation or for the actual PAC itself. In the FAC the last FROM/EFROM is the only one affected. The following are the addresses that have to be changed.

WORD FROCESSOR PAC - Changes

add.	was/is			change to
DF21 - LD HL,	EC6E	------>	GRATBL:	ED4C
DF3F - LD HL,	ECBE	------>	CONTBL:	ED9C
DF48 - LD HL,	ED0E	------>	SHITBL:	EDEC
DF51 - LD HL,	ED5E	------>	SLOTBL:	EE3C
DF56 - LD HL,	EDAE	------>	UNSTBL:	EE8C

Thats all there's to it, and EXMON2 is compatible.

NOW THAT WE'RE ALL ASSEMBLED .... by Joseph R Power

Unless you happen to be using a Burroughs Medium System or an IBM 1401 (which are perverted enough to think in base 10), you've probably run into the problems of dealing with decimal numbers in binary machines. (Those machines have the reverse problem - handling binary numbers.) This article attempts to alleviate some of those problems. Once again we approach things from the considerations of space and speed - the only valid reasons for programming in assembly language.

The normal method of multiplying an integer by 10 on the Z80 is:

```
ADD   HL,HL      ;HL := NUM * 2
LD    D,H
LD    E,L        ;DE := NUM * 2
ADD   HL,HL      ;HL := NUM * 4
ADD   HL,HL      ;HL := NUM * 8
ADD   HL,DE      ;HL := NUM * 10
```

This takes 6 bytes and 52 cycles and will work for any integer less than 6554 (unsigned).

While the above routine is by far the most general and easy to understand, and while it works for the greatest number of cases, it is not the fastest IF the numbers being multiplied are much smaller. In particular, if the numbers are less than 52 the routine:

```
                 ; HL < 52
LD    A,L        ;A := NUM
ADD   A,A        ;A := NUM * 2
ADD   A,A        ;A := NUM * 4
ADD   A,L        ;A := NUM * 5
LD    L,A        ;HL := NUM * 5
ADD   HL,HL      ;HL := NUM * 10
```

is the same size but over 40% faster. The trick here is to realize that 10 is 5 * 2. Numbers less than 52 can be multiplied by 5 and still fit in one byte. This allows us to use the faster single byte additions.

One application where this technique can prove its mettle is in determining a cursor address given its row and column (where screen width is a multiple of 10). Since, for instance, most 80 column screens have 24 or 25 rows the routine would fit in quite nicely:

```
; ENTRY - L = ROW (0..24)
;         H = COLUMN (0..79)
;
; EXIT  - HL = CURSOR ADDRESS (SCREEN + [0..1999])
;
LD    E,H        ;PRESERVE COLUMN
LD    H,0
LD    D,H        ;DE := COLUMN
LD    A,L
```

```
ADD   A,A
ADD   A,A
ADD   A,L
LD    L,A
ADD   HL,HL       ;HL := ROW * 10
ADD   HL,HL
ADD   HL,HL
ADD   HL,HL       ;HL := ROW * 80
ADD   HL,DE       ;HL := ROW * 80 + COLUMN
LD    DE,SCREEN
ADD   HL,DE       ;HL := CURSOR ADDRESS
```

The described technique and the example given point out one of the major reasons people use assembly language instead of compilers for certain applications - in assembly language you can make certain assumptions that compilers cannot. You know what range of possible values a number can have whereas a compiler must anticipate the worst.

SIO INTERFACE

FIG. 2
CLOCK

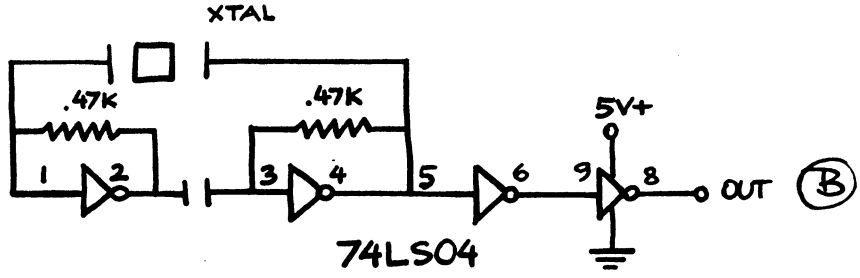
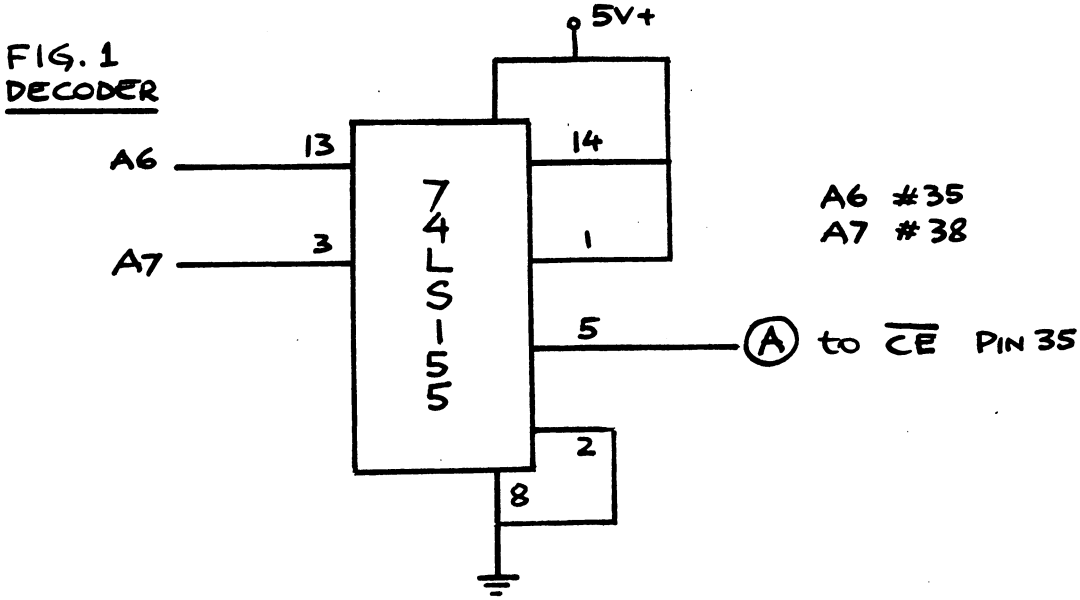


FIG. 1
DECODER



SOFTWARE for the EXIDY SORCERER

Software marked * comes with Sound and Joystick control

			Price U.S.$	Amount
GAMES				
8206	**DEFENDER** — Takes the form of the similar arcade game of the same name and is fast, furious action.	M/L	**22.95**	_____
8205	**MUNCH!** — Sorcerer version of the popular PAC MAN.	M/L	**22.95**	_____
8202	* **CIRCUS** — Arcade game by the author of Galaxians.	M/L	**19.95**	_____
8201	**SORCERER INVADERS** — The arcade Invaders at its best.	M/L	**16.95**	_____
8118	* **SUPER ASTEROIDS** — Real-time high resolution game.	M/L	**26.95**	_____
8117	* **GROTNIK WARS** — Sorcerer's answer to Star Raiders.	M/L	**22.95**	_____
8111	* **GALAXIANS** — Finest specimen of Galaxians anywhere.	M/L	**19.95**	_____
8105	**GOLF** — A skilful game with attractive graphics.	BASIC	**14.95**	_____
8103	* **SPIDER, ECHO and SOUND CARD** — Two captivating games.	M/L	**49.95**	_____
UTILITIES				
8208	**TAPE DOCTOR** — Put an end to tape errors forever.	M/L	**22.95**	_____
8207	**BASIC PRE-PROCESSOR** — Requires CP/M, Disk Editor & Disk BASIC	M/L	**19.95**	_____
8203	**ZAP80** — 'Secret Code Disassembler'	M/L	**22.95**	_____
8115	**SUPER GRAPHIC SCRATCH PAD**	BASIC & M/L	**19.95**	_____
8110	**ZETU** — Cassette-based Z80 Assembler	M/L	**29.95**	_____
8108	**ON-SCREEN EDITOR** — For Disk Extended BASIC "EXBASIC"	M/L	**29.95**	_____
8101	**TOOLKIT** — THE on-screen Editor for BASIC ROMPAC	M/L	**22.95**	_____
8005	**SWORD** — Sorcerer Word Processor	M/L	**22.95**	_____
EDUCATIONAL (** with Sound)				
8210	** **SPELLING TEST** — Allows you to set up a module tape with a voice track of the words to be spelt.	BASIC	**19.95**	_____
8209	**ARITHMETIC TUTORIAL** — Simulates the solving of Addition, Subtraction, Division, Multiplication and Times Tables. Printed report card.	BASIC	**24.95**	_____
8204	**BASIC TUTORIAL PKG.** —	M/L	**22.95**	_____
8119	** **TOUCH TYPE TUTOR** —	M/L	**17.95**	_____
8112	**MACHINE CODE TUTORIAL** — Eight exercises.	M/L	**24.95**	_____
7916	**KNOW YOUR SORCERER #1, 2, 3** — Ideal for new owners.	BASIC & M/L	**17.95**	_____

PROGRAMS ARE IN CASSETTE FORM UNLESS OTHERWISE STATED

POSTAGE & HANDLING

N. America only: Minimum **$1.50** (up to two items). Each additional item **75c** each. _____

Overseas: Inclusive of Registration or Insurance **$5.00** _____

Terms: Orders are payable in U.S. funds by certified cheque or money order.
C.O.D. orders are not possible. VISA or MASTERCARD accepted. **Total** _____

VISA No: _____ Expiry Date: _____

MASTERCARD No: _____ Expiry Date: _____

Name: _____ SIGNATURE _____

Address: _____

Order your software from
NORTHAMERICAN SOFTWARE
at their NEW MAILING ADDRESS:

Northamerican Software
P.O. Box 1376
Station B
Downsview, Ontario
Canada M3H 5V6

```
        TITLE ELAPSED TIME CALCULATOR DATE 04/04/1982
;=======================================================
;
;       AUTHOR:
;               HANK SZYSZKA
;               PO BOX 6
;               BEARSVILLE, NY
;               ZIP 12409
;               MODEM (914) 679-6559
;
;  ELAPSED TIME IS CALCULATED FROM CURRENT CLOCK
;  READING AND FIELD POINTED TO BY HL REGISTER PAIR.
;  DIGITS MUST BE STORED SEQUENTIALY IN NORMAL
;  ASSENDING ORDER. (HHMMSS) THIS SYSTEMS STORED
;  LOCATION IS 000A THRU 000F OF LOWER STORAGE.
;  THE BCBBS STORES THE TIME A USER LOGS ON AT THAT
;  ADDRESS FOR 'ON SYSTEM' TIME CALCULATIONS AND
;  INFORMATION TO USER. THE RESPONSE OF THIS PROGRAM
;  IF MUCH FASTER THAN A 'COMPILED' VERSION. CLOCK
;
;  BOARD IS A MOUNTAIN HARDWARE 100,000 DAY CLOCK.
;  ADDRESSED AT 128(DEC) 80(HEX).
;
;=======================================================
PORT:   EQU     128             ;BASE CLOCK ADDRESS
HPORT:  EQU     PORT+9          ;HOURS, TENS
BDOS:   EQU     0005            ;LINK TO CP/M BDOS FUNCTIONS
TBUF:   EQU     000AH           ;STORED TIME AREA
CR:     EQU     13
LF:     EQU     10
PAS:    EQU     09              ;PRINT UNTILL $
PAC:    EQU     02              ;PRINT A CHARACTER
;=======================================================
;
        ORG     0100H
;
;=======================================================
; GET CURRENT CLOCK TIME AND SAVE..
;=======================================================
START:  EQU     $                       ;START OF PROGRAM
        CALL    GETTM                   ;GET THE CURRENT TIME
        LD      HL,TBUF                 ;ADDR OF STARTING TIME
        LD      DE,STIME                ;START TIME BUFFER
        LD      BC,0006                 ;LENGTH TO MOVE
        LDIR                            ;BLOCK MOVE
;=======================================================
;  SAVE A COPY FOR PRINT OUT ORIGINAL IS DISTROYED
;=======================================================
        LD      HL,ETIME                ;CURRENT TIME BUFFER
        LD      DE,EDITB                ;EDIT BUFFER
        LD      BC,0006                 ;LENGTH TO MOVE
        LDIR                            ;BLOCK MOVE
;=======================================================
;  SET UP POINTERS FOR CALCULATIONS
;=======================================================
        LD      DE,ETIME+5              ;ENDING TIME
        LD      HL,STIME+5              ;STARTING TIME
;=======================================================
; SECONDS, UNITS POSITION
;=======================================================
        LD      A,(DE)  ;GET UNITS, SECONDS HIGH
        CP      (HL)    ;SEE IF OK FOR SUBTRACT
        JP      M,DB1   ;IF MINUS, BORROW NEEDED
        JR      NC,SB1  ;IF OK, SUBTRACT
DB1:    ADD     10      ;ADD TEN
        DAA             ;KEEP IT DECIMAL
        DEC     DE      ;GOT TO 10'S SECONDS
        PUSH    AF      ;SAVE A REG
        LD      A,(DE)  ;GET BORROW DIGIT
        SUB     1       ;REDUCE BY ONE
        DAA             ;KEEP DECIMAL
        LD      (DE),A  ;PUT BACK
        POP     AF      ;RESTORE A
        INC     DE      ;RETURN TO ORIGINAL DIGIT
SB1:    SUB     (HL)    ;ADJUST SECONDS
        DAA             ;DECIMAL ADJUST
        ADD     '0'     ;MAKE IT ASCII
        LD      (STR+7),A ;STORE RESULT
        DEC     HL      ;STEP TO THE
        DEC     DE      ;NEXT DIGIT
```

```
;=========================================================
; SECONDS, TENS POSITION
;=========================================================
         LD       A,(DE)    ,GET HIGH
         CP       (HL)      ,COMPARE TO LOW
         JP       M,DB2     ,SEE IF MINUS NUMBER
         JR       NC,SB2    ,IF OK, SUBTRACT
DB2.     ADD      6         ,ELSE ADD 6 (SECONDS TENS)
         DAA                ,KEEP IT DECIMAL
         DEC      DE        ,GOT TO 10'S SECONDS
         PUSH     AF        ,SAVE A REG
         LD       A,(DE)    ;GET BORROW DIGIT
         SUB      1         ;REDUCE BY ONE
         DAA                ,KEEP DECIMAL
         LD       (DE),A    ,PUT BACK
         POP      AF        ;RESTORE A
         INC      DE        ;RETURN TO ORIGINAL DIGIT
SB2.     SUB      (HL)      ,SUBTRACT DIGIT
         DAA                ;DECIMAL ADJUST
         ADD      '0'       ,MAKE IT ASCII
         LD       (STR+6),A ,STORE RESULT
         DEC      HL        ,STEP TO THE
         DEC      DE        ;NEXT DIGIT
;=========================================================
; MINUITS, UNITS POSITION
;=========================================================
         LD       A,(DE)    ;GET UNITS, SECONDS HIGH
         CP       (HL)      ;SEE IF OK FOR SUBTRACT
         JP       M,DB3     ,IF NEGATIVE, BORROW
         JR       NC,SB3    ,IF OK, SUBTRACT
DB3.     ADD      10        ,ADD TEN
         DAA                ,MAKE IT DECIMAL
         DEC      DE        ,GOT TO 10'S SECONDS
         PUSH     AF        ;SAVE A REG
         LD       A,(DE)    ,GET BORROW DIGIT
         SUB      1         ;REDUCE BY ONE
         DAA                ,KEEP DECIMAL
         LD       (DE),A    ;PUT BACK
         POP      AF        ,RESTORE A
         INC      DE        ;RETURN TO ORIGINAL DIGIT
SB3.     SUB      (HL)      ;ADJUST SECONDS
         DAA                ,DECIMAL ADJUST
         ADD      '0'       ,MAKE IT ASCII
         LD       (STR+4),A ,STORE ASCII CHAR
         DEC      HL        ;STEP TO THE
         DEC      DE        ;NEXT DIGIT
;=========================================================
; MINUITS, TENS POSITION
;=========================================================
         LD       A,(DE)    ,GET HIGH
         CP       (HL)      ;COMPARE TO LOW
         JP       M,DB4     ;BORROW IF NEGATIVE
         JR       NC,SB4    ,IF OK, SUBTRACT
DB4.     ADD      6         ;ELSE ADD 6 (SECONDS TENS)
         DAA                ;MAKE DECIMAL
         DEC      DE        ,GOT TO 10'S SECONDS
         PUSH     AF        ;SAVE A REG
         LD       A,(DE)    ,GET BORROW DIGIT
         SUB      1         ;REDUCE BY ONE
         DAA                ;KEEP DECIMAL
         LD       (DE),A    ,PUT BACK
         POP      AF        ,RESTORE A
         INC      DE        ;RETURN TO ORIGINAL DIGIT
SB4.     SUB      (HL)      ,SUBTRACT DIGIT
         DAA                ;DECIMAL ADJUST
         ADD      '0'       ;MAKE IT ASCII
         LD       (STR+3),A ;STORE IT
         DEC      HL        ,STEP TO THE
         DEC      DE        ;NEXT DIGIT
;=========================================================
; HOURS, UNITS POSITION
;=========================================================
         LD       A,(DE)    ;GET UNITS, SECONDS HIGH
         CP       (HL)      ;SEE IF OK FOR SUBTRACT
         JP       M,DB5     ,IS IT NEGATIVE?
         JR       NC,SB5    ,IF OK, SUBTRACT
DB5.     ADD      10        ,ADD TEN
         DAA                ;MAKE IT DECIMAL
         DEC      DE        ,GOT TO 10'S SECONDS
         PUSH     AF        ,SAVE A REG
```

```
        LD      A,(DE)    ,GET BORROW DIGIT
        SUB     1         ;REDUCE BY ONE
        DAA               ;KEEP DECIMAL
        LD      (DE),A    ,PUT BACK
        POP     AF        ,RESTORE A
        INC     DE        ,RETURN TO ORIGINAL DIGIT
SB5.    SUB     (HL)      ,ADJUST SECONDS
        DAA               ,DECIMAL ADJUST
        ADD     '0'       ,MAKE IT ASCII
        LD      (STR+1),A ,STORE IT
        DEC     HL        ,STEP TO THE
        DEC     DE        ;NEXT DIGIT
;=======================================================
; HOURS, TENS POSITION
;=======================================================
        LD      A,(DE)    ,GET HIGH
        CP      (HL)      ;COMPARE TO LOW
        SUB     (HL)      ,JUST SUBTRACT IT
        DAA               ;DECIMAL ADJUST
        ADD     '0'       ,MAKE IT ASCII
        LD      (STR),A   ;STORE FINAL
;=======================================================
;  NOW WE CAN EDIT AND DISPLAY THE RESULTS.
;=======================================================
        LD      B,6       ;BYTES TO EDIT
        LD      HL,EDITB  ;ADDR. EDIT BUFFER
TRL1.   LD      A,(HL)    ;GET A BYTE TO EDIT
        ADD     '0'       ;ADD ASCII BIAS
        LD      (HL),A    ;STORE IT BACK
        INC     HL        ,STEP ADDR. POINTER
        DJNZ    TRL1      ;LOOP FOR ALL 6
        LD      HL,EDITB  ,SET UP TO MOVE TO
        LD      DE,STR2   ;OUT-PUT FIELD.
        LDI               ,TRANSFER A BYTE (H-TEN)
        LDI               ;TWICE           (H-UNIT)
        INC     DE        ,SKIP ':' IN DESTINATION FIELD
        LDI               ;TRANSFER BYTE   (M-TEN)
        LDI               ,NEXT TOO        (M-UNITS)
        INC     DE        ,SKIP ':' IN DESTINATION FIELD
        LDI               ,TRANSFER BYTE   (S-TENS)
        LDI               ;AND NOW LAST    (S-UNITS)
;=======================================================
;  NOW, PRINT THE RESULT TO CRT, AND WE ARE DONE.
;=======================================================
        LD      DE,MSG2
        LD      C,PAS
        CALL    BDOS
        LD      DE,MSG1
        LD      C,PAS
        CALL    BDOS
        RET
;=======================================================
;  THIS ROUTINE READS THE CLOCK BOARD PORTS
;=======================================================
GETTM:  PUSH    BC        ,SAVE
        PUSH    HL        ,SAVE
        LD      B,6       ,NUMBER OF DIGITS
        LD      C,HPORT   ;ADDRESS OF CLOCK
        LD      HL,ETIME  ;STORAGE AREA FOR TIME
BEG2:   IN      A,(C)     ,REG-C CONTAINS PORT ADDRESS
        AND     15        ,STRIP HIGH BITS
        LD      (HL),A    ;SAVE DECIMAL VALUE
        INC     HL        ,STEP STORAGE ADDRESS
        DEC     C         ;DECREMENT PORT ADDRESS
        DJNZ    BEG2      ,LOOP FOR 6 PORTS
        POP     HL        ;RESTORE HL AND
        POP     BC        ;BC REGISTERS.
        RET               ;RETURN TO TASK
;=======================================================
;  TEMPORARY WORK AREA FOR CALCULATIONS
;=======================================================
STIME.  DB      0,0,0,0,0,0       ;COPY OF STORED TIME
EDITB.  DB      0,0,0,0,0,0       ,EDIT BUFFER
ETIME:  DB      1,3,2,3,4,2       ,CURRENT TIME
MSG1    DB      'You have been on the system for '
STR     DB      0,0,' ',0,0,':',0,0       ,DIFFERENCE
        DB      '$'
MSG2.   DB      'The correct time is now ------> '
STR2.   DB      0,0,' ',0,0,':',0,0       ;CURRENT TIME
        DB      CR,LF,'$'
```

The SIO Part III (final part) by: Brad Fowles

Now we will hook up the SIO to the Sorcerer and have a look at a generalized program illustrating one of the many configurations possible.

Apart from the SIO you will need a decode chip (74LS155) and a clock. If you can do with a baud rate of 2400 or less then the clock can come directly from the Sorcerer. For higher rates a separate clock circuit and crystal must be used. Note, system clock rate 02 must be 4.5 x the data rate. See Fig. 1 for a simple clock rate generator. There are two usable clock rates on the Sorcerer. One at 38.5 Khz. and one at 19.2 Khz. By programing bits 6 and 7 of register 4 you can obtain the following baud rates.

From 38.5 Khz.	use	x16	for	2400 baud
	use	x32	for	1200 baud
	use	x64	for	600 baud
From 19.2 Khz.	use	x16	for	1200 baud
	use	x32	for	600 baud
	use	x64	for	300 baud

These signals DO NOT come out the 50 Pin expansion bus, so pick one for your particular use and solder a wire to:

Sorcerer I	38.5 Khz.	pin 2	of	4D	74LS161
	19.2 Khz.	pin 14	of	4D	74LS161
Sorcerer II	38.5 Khz.	pin 2	of	15H	74LS161
	19.2 Khz.	pin 14	of	15H	74LS161

Do NOT confuse these signals with the system clock pin 13 of the expansion bus.

The DECODE chip appears in fig. 2 . NOTE: It is the same circuit we used in the PIO a few months back.

Now wire up the decode and, (if external clock is used), clock rate generator. Then wire the SIO as follows.

PIN	1	D1	TO #40	PIN	21	RESET	TO #1
	2	D3	TO #42		22	N/C (DCDB)	
	3	D5	TO #44		23	N/C (DTSB)	
	4	D7	TO #46		24	N/C (RTSB)	
	5	INT	TO #2		25	N/C (DTRB)	
	6	TO +5V No daisy chain			26	DATA OUT B	
	7	N/C	(IEO)		27	TO CLK #13	
	8	M1	TO #16		28	DATA IN B	
	9	+5V			29	N/C (SYNC B)	
	10	N/C	(W/RDY)		30	N/C (W/RDY B)	
	11	N/C	(SYNC)		31	GND TO #50	
	12	DATA IN A			32	RD TO #17	
	13	TO	CLK #13		33	C/D TO	A0 #29
	14	TO	CLK #13		34	B/A TO	A1 #32
	15	DATA OUT A			35	'A' DECODE CIRUIT	
	16	N/C	(DTRA)		36	IORQ TO	#18
	17	N/C	(RTSA)		37	D6 TO	#43
	18	N/C	(CTSA)		38	D4 TO	#41
	19	N/C	(DCDA)		39	D2 TO	#39
	20	0	TO #13		40	D0 TO	#37

NOTE 1. PIN Refers to SIO, # Refers to expansion bus.

NOTE 2: CLK Refers to either external clock rate generator or clock rate taken from the Sorcerer internally.

SIO PART III continued.

We will use Port # 80 thru # 83. Now we must program the SIO. The program which follows is an example for initializing the SIO only. It will configure one channel in its most basic form. All the N/C 's on the SIO are for modem and snyc controls. If you wish to use these consult the two previous articles, and program the chip accordingly.

Wait loops and storage of data are up to the programmer and system needs. Write in to the group or talk over at the meetings, any specific serial needs. The same goes for programming or hardware problems.

Remmember.----->	Port	80	is	Data	A
	Port	81	is	Control	A
	Port	82	is	Data	B
	Port	83	is	Control	B

SEE DIAGRAM ON PAGE 5

A listing for programming up channel 'B'

```
    LD C,#83          ;SET PORT B CONTROL IN C REGISTER
    LD A,#2
    OUT (C),A         ;POINTER SET TO 2B
    LD A,(interrupt vector)
    OUT (C),A         ;INTERRUPT VECTOR LOADED
    LD A,#4
    OUT (C),A         ;POINTER SET TO 4B
    LD A,#47
    OUT (C),A         ;EVEN PARITY, 1 STOP BIT, x 16 ASYNCRONOUS
    LD A,#5
    OUT (C),A         ;SET POINTER TO 5B
    LD A,#2A
    OUT (C),A         ;SEVEN BITS / XMIT. CHARACTER XMITTER
    LD A,#3
    OUT (C),A         ;7 BITS / RCV. CHARACTER, DCD AND CTS
                      ;ENABLE RCV + XMIT, RCV ENABLED
    LD A,#1
    OUT (C),A         ;SET POINTER TO 1B
    LD A,#17
    OUT (C),A         ;INTERRUPT ON EVERY CHAR.,STATUS
                      ;AFFECTS VECTOR EXTERNAL /STATUS
                      ;INTERRUPTS ENABLED
```

Channel 'B' is now set to send and receive asyncronous data.

It may seem confusing, but with the help of the data sheets supplied with the chip, you can add as many serial ports as you require.

The SIO is an amazingly versitile chip as I've said, an absolute must for any experimentor.

If any of you have any questions regarding SIO or PIO interfacing that these brief articles didn't cover or are not too clear on, please send $ 1.50 to cover mailing expenses.

Brad Fowles

To: PORT PE. c.o Brad Fowles
F.O. Box 1173 STN. 'B'
Downsview, Ont. Canada M3H 5V6

So concludes the articles on PIO and SIO interfacing.

Parallel Port Transfers :from David Woodberry of Australia.

Dear PORT FE Members!

In light of your comments on transfer of data at high speed using the parallel port which appeared in Feb 1982 PORT FE, I thought you might be interested in the following which was developed by myself for use in connection with a program to enable transfer between two Sorcerers with dis-similar disk systems connected via the parallel port.

The program transfers data at a speed comparable with that achieved with PIP on a single machine. Note that the operation of these routines does not require any hardware modification to the Sorcerer. I am also including for your convenience a copy on tape of the transfer program (ASM file) which has been used successfully on a number of occasions. The program includes a CRC check routine and error control in the rx-tx direction which will allow the receiving program to be restarted on a new disk if it runs out of space. File specification is PIP type with ? and * specs allowed for multi-file transmission.

DOCUMENTATION ON CABLE FOR TRANSFER PROGRAMS

The parallel port is used on both the send and receive computers and these are connected by a 20 wire cable as follows:

Send Machine Pin	Receive Machine Pin	Send Machine Pin	Receive Machine Pin
1	8	11	18
25-2	4	12	7
3	9	13	5
4	25-2	16	10
5	13	17	22
6	24	18	11
7	12	19	23
8	1	22	17
9	3	23	19
10	16	24	6

NOTE. Pins 14,15,20,21 are not used at either end. Pins 25 and 2 are commoned at either end and connected with a single cable to pin 4 of the other.
The connections are 'mirror-image' and may be read either way.

```
;This routine is the parallel driver for the sending side:-
SENDIT: PUSH    AF
SNDT1:  IN      A,(0FEH9        ;check to see if receiver ready
        BIT     6,A
        JR      Z,SNDT1-$
        POP     AF
        OUT     (0FFH),A        ,send it.
        RET
;This routine is the parallel driver for the receiving side:-
GETiT.  IN      A,(0FEH)        ;check to see if data available
        BIT     7,A
        JR      Z,GETiT-$
        IN      A,(0FFH)        ;get it
        PUSH    AF
        XOR     A               ;this bit is the data accepted
        OUT     (0FFH),A        ;signal from the receiver to
        LD      A,080H          ,the sender
        OUT     (0FFH),A
        POP     AF
        RET
```

The sending side routine is capable of sending 8 bits in parallel (contents of A register) to the receiving side (contents of A register). Because of the nature of the connection it is only possible to send 7 bits of data in the reverse direction. This can typically be used for control information and would need additions to the simple drivers shown here. Note further that since the connection between the two machines is symmetrical either can act as sender and receiver.

NOTE: with the program DT.COM the receiving side program should be started before the sending side program to avoid initialization problems.