PORT FE

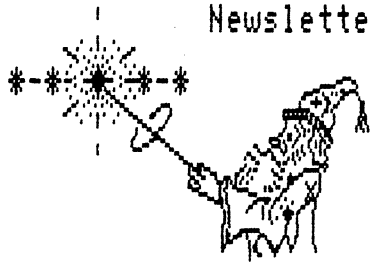
SORCERERS USERS' GROUP

(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.

February 1982 ISSUE

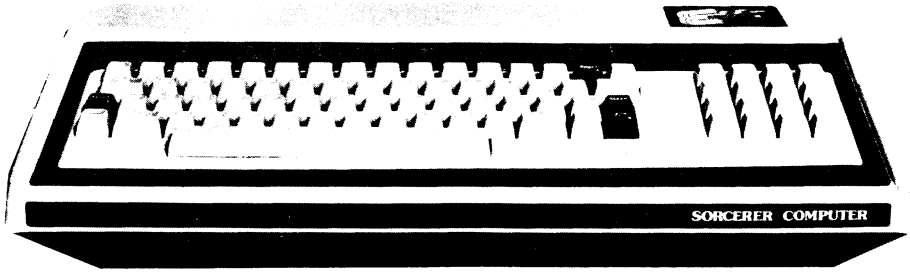
TABLE OF CONTENTS

MEETING PLACE

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

January	Thurs. 21st	February	Wed. 17th.	March	Thurs. 18th.
April	Wed. 14th.	May	Thurs. 13th.	June	Wed. 16th.

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



I'm still filling in for our new editor who is waiting for his S100 to be repaired by Mentzer Electronics. He is still waiting to try out his new Disk system. Last word that we heard is that it was 12th in line to be looked at. Hopefully he will have it back from Exidy soon.

In the last issue of PORT FE I mentioned about the problems with CP/M 2.2. I would like to explain briefly, that in the last issue I did not mention the name of the originator of the CP/M 2.2 . Well since the issue some new things have come to light. We tried to get in contact with the individual who made these changes and in fact did. The CP/M is sold under the license from Digital Research by Bruce Ratoff. This may have been the original version but it seems that Bruce Ratoff is the eastern representative for Digital Research .The CP/M was actually modified by some jerk in California who seems to be under the misguided conception that he's god's gift to the programing world. Well to make a long story short. He flatly told one of our members that if we didn't like the way he had reconfigured CP/M 2.2 that we should do it ourselves and hung up the phone on him. This type of attitude I did not expect from a group that was associated with Digital Research. Unfortunately I am not ready to release the updates and changes yet. They will probably be in the next issue.

Some of the readers who have ordered the Library tape may be or have experienced some difficulties in loading the programs. This has been corrected and was due to the high speed duplicator that had a bad write head. The program tapes unfortunately cannot be checked because of the time involved and the number of programs on the library tapes. The masters are checked though. For those that cannot load the tape we would ask that you return it/them and we shall send you a new one. Volume 3 should be ready in about one month's time and hopefully we shall have no more of the loading problems due to the high speed duplicator.

Visicalc has been looked at by one of our members (Walter Blady) and he has stated that in its present form is not compatible with the Sorcerer screen format. More to the point specifically is that it really requires a cursor positioning routine and so far even when implemented there is more to it than meets the eye. A source listing would be required to fully make it compatible.

Parallel port flustrations have had their toll on a few of us again, another one of those all nighters was fruitlessly enacted. Oh EXIDY in their great wisdom keep us up to all hours. How many times has this happened to Sorcerer owners I would hate to think about. Well in the light of what's been learned at the expense of some sleepless nights, we recommend that you all leave the parallel port alone for anything other than maybe slow input,joysticks, sound output and last but not least printer output. Data transfers at high speed are not for the parallel port.

Serial port data transfers in excess of 1200 baud, cannot be made either. We are looking into the hardware modifications that will allow transfers at higher baud rates.

Two new columns have been added to PORT FE. One will give some of you who want to learn insight on a few hardware projects. The other will give the 'C' programer something to work on or learn 'C' programming.

Due to the length of some of the articles appearing in PORT FE we will discuss the possibility of a bi - monthly issue at the next meeting. We would also like to have your comments on this matter as well. I hate to have to split up articles and I'm sure that many of you feel likewise. This would enable us to possibly more than double the content and at the same time we could include the complete longer articles as well. Think about it, we will not change PORT FE format if you the members wish to leave it the same.

H.A. Lautenbach

Everyone at some point in time has wished they could have a split screen like the Apple - with a text window at the bottom to output information and a graphic window above which is not disturbed by any print commands.

With the addition of this small text window routine, you will no longer have to worry about the screen inadvertantly scrolling just as the last Klingon is about to be destoyed or having to use fancy cursor controls to input information at a specific point on the display.

I have found it handy in making a pseudo Pet computer screen of 40*25 lines and as status lines in real-time Star Trek games.

Operation:

By calling the initialize routine at 330H, the input and output vectors in the monitor work area are patched to jump to the text window routine instead of directly to the monitor roms.

All input from the keyboard is fetched from the the flashing cursor keyboard sub-routine which is needed so no characters input will be displayed outside the set-up window.

Output is sent to the output character sub-routine. This displays the character in a specified window area defined by the user. It also looks after scrolling of the window, clearing the window and handling any control codes such as the cursor control characters. This will behave exactly as a normal Sorcerer screen would.

It will allow you to define any size 'window' on the screen (minimum 1 character wide) where all text will be output and scroiled. The routine will not affect any other part of the undefined screen area. Unlike the Apple in high-res mode, the window may be located anywhere, not just the bottom lines of the screen.

Installation:

The routine has been developed to be compatible with Rom-Pac Basic and as a subroutine in an all-machine language program. It will look transparent to the Sorcerer - it can't tell if a special controller is between the Basic and the monitor's character output routine.

This routine is too large to fit in zero page , it can't fit inside a 256 byte string and it is inadvisable to put in the programmable graphics area. The only other place was to install it either between the Basic program text and the top of memory or just before the start of the Basic program. The latter I chose because it is protected from being over-written by an expanding Basic program.

To install this routine, do the following steps:

1) Insert the Basic Rom-Pac , go to the monitor with 'BYE' and type in listing # 1 at 01D0 hex. Do NOT forget to include the three zeros at the end of the listing! These are the null bytes for Basic.

(----- OR -----)

Load the previously saved text window routine from tape, while in the monitor , then go to step 3.

2) Enter: '0149 = 40 03' --) This protects the machine language program so Basic won't put a program over top of it.
3) Type 'PP' then 'NEW' when in Basic. ---) The routine is now protected and installed below the Basic text area.
4) If the text window routine has already been saved on tape ,skip step # 4. Else type 'BYE' then 'SA WINDW 0140 0342' ---) this is the text window routine in a form which can be loaded from cassette for other programs needing it.
5) Return to Basic with 'PP'. Now type in the Basic program you want to use with the text window routine.

(----- OR -----)

If you have a Basic program on tape that you want the text window added to, do the following while in Basic:

'CLOAD NAME 1 0340' --) where NAME is the name of your Basic program.

6) Now you must program the text window for the desired size of window to have. POKE the following decimal locations with the data set by the comment beside the location.

473 (LEFT) = this is the leftmost cursor column that the window will begin in. If we poke in 10, then the left side of the text window will start 10 spaces from the left side of the screen.
474 (RIGHT) = this is the WIDTH of the window, NOT the cursor location of the right side of the screen. If 10 was poked in (LEFT) and 30 poked in (RIGHT) then the window will start 10 spaces from the left of the screen and end at cursor location 40 on the screen.
475 (TOP) = this is the very top line of the text window. 0 = the top line of screen and 30 = the very bottom line.
476 (BOTTOM) = this is the very bottom of the text window. 30 = the very bottom line of the screen.
477 = this must be initialized to the same number as (LEFT)
478 = this must be initialized to the same number as (TOP)
322 = poke this location with '255'. This stops Basic from interfering with the Text Window's auto carriage return.

To set the text window for a normal screen size of 64*30 lines, initialize the pointers to these values:

(LEFT) = 0 (RIGHT) = 64 (TOP) = 0
(BOTTOM) = 30 (477) = 0 (478) = 0
(322) = 64

7) The Basic program and the text window routine are now existing in memory unaware of each other. Thus we must link them together so they will be able to co-operate.
To link the text window to the input and output vectors (initialize routine), add these three statements after the pokes described above, in your program:

```
10 POKE260,48:POKE261,3:X=USR(0)
```

To move the cursor inside the window, insert this statement:

```
20 PRINT " ";
```

This completes the initialization of the text window. It is now firmly rooted to your Basic program. Before you run it , save the program on tape.

*** ATTENTION *** - this combination machine language-Basic program must be saved using the monitor and NOT with the CSAVE command. If it is CSAVEd, the program will not run properly when it is later loaded.

To save your program while still in Basic, do the following:

```
BYE
)DU 01B7 01B8

ADDR     0   1  2  3  4  5  6  7  8
01B0:    XX  XX XX XX XX XX XX YY ZZ

)SA NAME 0140 ZZYY
)PP
READY
```

The text window, the Basic program and the pointers protecting the routine are now on tape as one 'Basic' program.

To load the program back into Basic:

'CLOAD .NAME' ---) No need to load it in the monitor.

It can be 'CLOADed' while in Basic because the program was saved with special pointers. When loading, these overlayed the Basic work area automatically and protected the machine language routine.

Sample demonstration program:

Follow steps 1-8 for the setup of the Basic program and the machine language routine. Use listing # 3 for the Basic program described in step 5.

Wrap-Up

If you follow the above steps as indicated, you won't have any problems using the routine. Just remember to save the Basic program in the monitor or strange, cruel things may happen to your masterpiece!

Listing # 1

```
ADDR    0  1  2  3   4  5  6  7   8  9  A  B   C  D  E  F

01D0:   E6 11 DF 01  00 80 F0 80  F0 00 40 00  1E 00 00 F5
01E0:   D9 CD 12 02  D9 F1 C9 CD  EF 01 3E 20  CD C4 02 21
01F0:   DD 01 35 F0  3A DA 01 3D  32 DD 01 3A  DB 01 21 DE
0200:   01 BE D0 35  3A DE 01 CD  E8 02 21 D9  01 8E 32 D5
0210:   01 C9 FE 0C  28 27 FE 0D  28 47 FE 0A  28 49 FE 17
0220:   28 D9 FE 1A  28 41 FE 01  28 C5 FE 13  CA D6 02 FE
0230:   08 28 B4 FE  11 28 1E FE  20 D2 C4 02  C9 21 DD 01
0240:   46 3A DB 01  F5 CD 07 02  CD AE 02 06  00 F1 3C 21
0250:   DC 01 BE 38  EF 3A DB 01  32 DE 01 3E  00 32 DD 01
0260:   C9 AF 32 DD  01 18 9D 21  DE 01 34 7E  EB 21 DC 01
0270:   BE 38 91 EB  35 3A DB 01  F5 CD 07 02  2A D5 01 22
0280:   D7 01 21 DA  01 46 F1 3C  21 DC 01 BE  30 14 F5 C5
0290:   CD 07 02 C1  2A D7 01 48  06 00 EB 2A  D5 01 ED B0
02A0:   18 DA 06 00  CD AE 02 D2  04 02 21 DD  01 46 3E 20
02B0:   2A D5 01 5F  48 06 00 09  77 41 04 3A  DA 01 3D B8
02C0:   7B 30 ED C9  F5 CD 04 02  F1 21 DD 01  46 2A D5 01

ADDR    0  1  2  3   4  5  6  7   8  9  A  B   C  D  E  F

02D0:   48 06 00 09  77 41 21 DD  01 34 7E 21  DA 01 BE F5
02E0:   D4 61 02 F1  D2 67 02 C9  47 04 21 40  F0 11 40 00
02F0:   19 10 FD 22  D5 01 7D C9  E5 D5 C5 3A  DE 01 CD E8
0300:   02 3A DD 01  16 00 5F 19  3A D9 01 5F  19 7E F5 36
0310:   5F 11 50 00  D5 CD 18 E0  D1 B7 20 0E  1B 7A B3 20
0320:   F3 7E C1 C5  B8 28 E8 70  18 E7 C1 70  C1 D1 E1 C9
0330:   CD A2 E1 11  F8 02 CD 31  E6 11 DF 01  C3 15 E6 00
0340:   00 00
```

By Walter Blady

When I chose the Hayes Smartmodem over it's competitors, it wasn't without good reason. Insinuations in the manufacturer advertising of great STACKABLE (compatible) things to come, made me drool, so I considered the choice a sound one. Sure, the Smartmodem is only 300 (slow) baud, but hold it a minute, we're talking about the revolutionary STACKABLE. I might be able to stack something on it one day that will give me better than 300, or who knows what else - That new Smartclock they're designing - everyone knows they didn't need such a large container for a mere calendar/clock. There must be other goodies inside right? Wrong! It's finally here but look at this, just a plain ol' calendar/clock. Well that's not entirely true. The Smartclock is a very intelligent time piece that will control your system in real time. It also accept commands to do a variety of things in much the same manner as the Smartmodem.

But let's get back to their catch word 'STACKABLE' and the compatibility it suggests. It seems that word was nothing more than promotional hype. As a matter of fact, the only thing Smartmodem and Smartclock have in common is their ability to stack. A highly negligible feature to say the very least. What about the compatibility that allows them to share the same power supply, or the same teensy weensy serial port. Yes friends, if you buy this clock you'll have to plug in yet another adapter and find yourself one more (scarce) I/O port. Why, they don't even communicate with one another. Certainly these features wouldn't have been that difficult to build in.

I didn't mind so much that there was no provision for faster baud rates, or other surprise features - that was just wishful thinking on my part. To tell you the truth I was impressed with Smartclock and almost bought it on sight. But the lack of hard compatibility with Smartmodem stopped me. And from now on I'll strictly regard the word STACKABLE as just that. Nothing more.

Operation of 8 Bit Printers

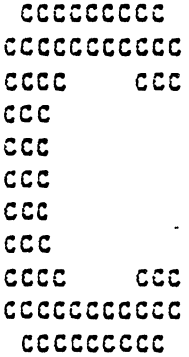
(Reprinted from the January issue SCUA)

Many printers need all eight bits of the data line to receive their character information. Exidy in its wisdom only permits seven bits to be conveniently used in the Centronics interface on the Sorcerer. There is a fairly simple modification which will eliminate this problem. All that you need to do on model I Sorcerers is to cut the trace going from pin 8 of 8F (74LS74) to the I0B6 line eg pin 17 on 2C (a 74LS241). Note that the trace from pin 8 of 8F to OUTPUT DATA AVAILABLE (pin 3 of jack J1) must be left intact. The I0B6 line is then connected to OUTPUT DATA ACCEPTED - pin 2 of jack J1. In the Sorcerer II exactly the same procedure of cutting one trace and putting in one connection is carried out. The 74LS74 in the Sorcerer II is I.C. 10F; the 74LS241 is I.C. 1D. The relevant part of the circuitry is illustrated in the lower right hand corner of schematic number 6 in the Technical Manual.

- Ben Williams (SCUA)

I would like to bring special attention to the above parallel port modification. In the last few months some of us have been trying to accomplish parallel to parallel data transfers at high speed and have run into some snags along the way. If anyone has accomplished this with the above modifications we would be pleased to hear from you. We shall also be trying some other modifications which involve the changing of the biasing to the 74LS74 in question. We feel that the handshake pulse width on the reset is lost.

H.A. Lautenbach



THE C PROGRAMMING LANGUAGE

By: Jacques Giraud

Are you sick of BASIC? Are you sick of slow languages? Are you sick of assembler? Well, I may have some good news for you then. The news is C, a structured (more about that later) compiled (fast) language. I have been running BDS C 1.44 on my Sorcerer for about six months now, and it is the only language that I use on a regular basis.

What is C?

"C is a general-purpose programming language which features economy of expression, modern control flow and data structures, and a rich set of operators. C is not a "very high level" language, nor a "big" one, and is not specialized to any particular area of application. But its absence of restrictions and its generality make it more convenient and effective for many tasks than supposedly more powerful languages."
The C Programming Language, Page IX. Kernighan & Ritchie.

First the bad points of C. C is a compiled language unlike BASIC. You create a source file with an editor and then run it through a compiler to get machine code (for CP/M it produces .COM file as the end result). If you make an error you have to correct the error and recompile again. C contains no run-time error checking. That means that you can run over the end of arrays, corrupt other variables and generally do some nasty things to your system. Since there is no run-time error checking, the code it produces is fast. BDS C lacks true floating point (it does have a library package that simulates it though), so it would not be good for designing accounting systems. (I have ordered Supersoft C which does contain full floating point, I will compare them in a later article). The syntax has been called a lot of things... in fact people generally divide into two categories the first time that they use C. The fanatics like myself and the people that curse the day C was ever invented. It is not a language for people who are just getting into computers.

Now for the good points. C is fast. C is concise. C lets you do stuff that you cannot even dream of doing in BASIC. C can, for the most part, replace assembler. It has a rich set of operators and control structures. It allows you to do things in a recursive manner. It has a preprocessor so that you can do macro substitution in the language. It has storage allocaters so that you can get storage on the fly. It also has a comprehensive stream directed I/O package. The print function takes a very detailed format so that you can print out hex, octal, left and right justified strings, floating point, regular and unsigned numbers. Full C has the following data types, short, long, int, float, double, char, struct, union, bit fields, statics and unsigned. BDS C is missing longs, statics, doubles, and bit fields. It also contains pointers to all of the above, so that you can manipulate them in some very interesting fashions. Enough talk let me show you some simple examples.

continued next page

Say we wanted a program to convert from upper case to lower case:-

```
#include  <bdscio.h>
#include  <dio.h>

main(argc, argv)
int argc;
char **argv;
{
     int c;                    /* conversion character */
     int getchar();            /* dio needs to see these */
     int putchar();

     dioinit(&argc, argv);
     while ((c = getchar()) != EOF)
          islower(c) ? putchar(c) : putchar(tolower(c));
     /*
      * could have done instead:

          if (islower(c))
               putchar(c);
          else
               putchar(tolower(c));
     *
     */
     dioflush();
}
```

Gee thats neat, but what the hell does it mean? I am getting to that. The first 2 statements are actually instructions to the preprocessor. It tells them to include the 2 files into this source file. Bdscio.h contains constants like EOF that are used very often in program. Dio.h contains the constants and variable that are used to do directed i/o. (Hang on for a minute, I will get to that). The angle brackets means that these are system .h (.h for header files) files, rather than ones that the user made up. The program gets under way with the main(argv, argv) function. Everything in C is a function that can have parameters and may or may not return a value. The main module is no exception. BDS C supports command line parameters, so when I have finished compiling the program. I would go:-

```
A>lower file1 file2 ... filen
```

and the number of arguments on the command line would be available in argc (argument count) and the actual commands would be argv. (argv[0] = lower, argv[1] = file1.. etc). Argv is a data type called a pointer to a pointer of characters, or a pointer to character pointer array. Don't worry if you don't understand it, pointer are the most difficult part about C.

The next lines declare the functions and variables that I am going to use in this function. I am going to use an int (16 bits long) to hold my character that I am working on, and I am going to be using the functions getchar(), and putchar() for i/o. In most programs the function definitions would be omitted but in this case, I want to use versions different from the standard ones, so I must declare them. Function in C always return ints unless you tell them otherwise. They can return anything except structures and unions (but they can return pointers to these). The next line initializes the directed i/o and then we finally get into the meat of the program, all 2 lines of it.

continued next page

The first thing that happens is that a character is got from the input stream and assigned to c. (c = getchar()) Some of you may be wondering why is c an int. Well it turns out that you want to some way of signalling EOF so they use a value like -1. Chars can only contain positive numbers, hence we have to use an int. (PDP11s can have negative chars). The next thing that happens in the program is that c is tested against the constant EOF. (while ((c = getchar() != EOF). The not equal operator in C is !=, the equals operator is ==. EOF is a constant that is defined in bdscio.h, so the preprocessor replaces the value for you with constant (I thinks its -1 but I am not sure, and there is very little need to know what it is, as long as you use EOF to represent it) In general defined constants are in upper case in C, and regular variables are in lower. It then does the next rather strange looking statement,

```
islower(c) ? putchar(c) : putchar(tolower(c));
```

Islower(c) is a function that returns true if c is a lower case character. The ? is like saying then in BASIC. The first clause is the one that it does if its true, so it prints the character. The clause after the : is executed if islower(c) is false. Tolower converts the given character to lower case, and putchar prints it.

Since the program is in a while loop it will continue to do this until it gets the end of the input. The statement dioflush() closes all the i/o channels that it may have used and then the program exits back to CP/M.

Remember I was mumbling about directed i/o. Well here is what it means. If I just typed A>lower CR it would wait until I hit a key and then it would be the conversion for me, if it was upper case. The standard input stream (called stdin) was the keyboard and the standard output stream (called stdout) was the console. I could if I wanted, make the input stream a file by doing this.

```
A>lower <file
```

That means take the contents of file and use it as input. The output would still appear at the console. I could also direct the output to another file by going:-

```
A>lower <file >output
```

and output would contain the output of the program. Whats more I have created a filter, something that reads the standard input and writes it to the standard output. The other thing you can do with directed i/o is to make one program and take the output of the first program and feed it into the second program. Say I had a program called pr which paginates text files and I wanted the input to be lower case only, I would go:

```
A>lower <file |pr
```

Lower would read file, and when it had finished doing its stuff it would give the output to pr as its input. A very powerful feature. That's what dioinit() and dioflush() set up for you.
Using C, you can create a wide variety of tools that do a lot of useful things, and in this column I hope to give examples of those tools in C.
If you want more information on the C, the standard text is:
The C Programming Language, Brian W. Kernighan & Dennis M. Ritchie, Prentice-Hall Software Series.
Another excellent text on tool building is Software Tools in Pascal or Software Tools in Ratfor (Ratfor is a Fortran preprocessor that looks identical to C). They are usually rubbing shoulder with the C Programming Language. They are both by Kernighan & Plaugher.

In Toronto, you can get them at the U of T book stores for about $20.00.

The following information has been received from our Sorcerer friends in the Netherlands, and lists various items available from members of the ESGC User Group. The prices quoted are in Guilder (Fl), the Dutch currency, and do not include shipping charges.
Available from: Gilbert Oegema ,Florisdonk 10,4707 VM Roosendaal,The Netherlands

**LIGHT-PEN for the Sorcerer. Price Fl. 70.- ($47.50 U.S.)
This light-pen plugs in the 'Parallel Interface' on the back of the Sorcerer, and comes together with 5 programs to teach you to use it inside a BASIC and ML-program, and one demo-program.
Avail. from: Gebr. Van Montfort,Smedestraat 13,6418 CR Heerlen,The Netherlands.

I/O PACK Price* Fl. 150.- ($99.95 U.S.)
An 'Eprom-Pack' with 24 I/O lines, user programmable. There is also room for 2 Eproms.

EPROM-PROGRAMMER Price: Fl. 300.- ($205.00 U.S.)
This Eprom-Programmer plugs (with 25 pole D-connector) in the I/O-Pack. It comes complete with all needed software, which can be put in the I/O-Pack.
Programmable are: 2716, 2516, 2732, 2532.
You can read out: the above + 2316, 2332, 2364.
The software enables the user to program per byte, block of bytes, or a whole Eprom at once. After programming an automatic check is made, in order to be sure everything is done correctly. All the switches are done by software!

EPROM-PACK Price:fl. 75.- ($49.95 U.S.)
This empty Eprom-Pack can be used to plug-in your own software in Eprom. You can also select, with a switch, between two blocks of 8K(when you use 2732's), without turning the Sorcerer off. This also saves the connector.

ROM-PACK BUS SYSTEM Price: Still unknown
With this buffered bus-system you can select by software, without turning the computer off, between 8 packs. Those packs can be exchanged during use. This means 64K ROM, Eprom and/or RAM is available. This system is delivered with a utility-Pack with routines to link the different packs by software e.g. in order to use a combined BASIC Word processor Development Pack....this system cannot be delivered for the moment.....

**PRINTER-INTERFACE for the EPSON MX80 Printers) also usable for the OKI u-80) Price: Fl. 115.- ($77.95 U.S.)
The two standard interfaces of the Sorcerer normally need their own cable. You cannot print graphics with the Centronics driver, but this is the most frequently used printer-driver. You can print graphics with the parallel-driver. This cable with incorporated interface, enables you to use both the drivers without exchanging the cables.

The program 'EXPAN' (Monitor Expansion) Price:Fl.25.- ($16.95 U.S.)
This is a very powerful Debugger/Disassembler. The 13 existing monitor commands are expanded with 23 new ones. The old commands ENTER, MOVE and SET are improved. Some of the new commands are: Breakpoint-manipulations (SR, RB, DR, DS, DB, CO, ER and EX9, Dump Ascii (DA), Hex-calculation (HE), Save previous file (SAF), etc....
Available from: Leo Gielen,Zeishof 18,6418 JJ Heerlen,The Netherlands

EPROM-ERASER Price: Fl. 110.- ($74.95 U.S.)
This Eprom-Eraser can, as more expensive models, erase 4 Eproms at a time in about 3/4 hour (2716). Of course, all types of Eproms can be erased with it. The UV-lamp is easy to find and to fit, by the user. The case is well built, the eraser is harmless at normal use.

continued next page.....

EXIDY SORCERER MONITOR ROM VERSION 1.1 B Price Fl.50.- ($33.95 U.S.)
This ROM is easy to exchange with the original one (version 1.0). All known bugs are fixed (as far as possible). The jump-addresses are still the same as in the 'old' one. For the do-it-yourself man a version is available which you can select with a switch between the old and the new ROM.

GAME INTERFACE/DA CONVERTER Price Fl. 140.- ($95.00 U.S.)
This system consists of four parts:
- Joystick interface
- Music interface
- Amplifier for the above
- 8 bits DA-converter, based on a DA-chip.

The music-interface is fully compatible with the 'Arrington Standard', so can be combined with his software.
Avail. from: Aad van Duijvenbode,Eliotplaats 7,3068 EE Rotterdam,The Netherlands.

EPROM for the character generator of the MICROLINE 80 printer
Price Fl. 55.- ($37.50 U.S.)

This Eprom, together with an output routine and a modified connection cable to the printer, enables you to print the 'Standard graphics' of the Sorcerer, and in another version the 'Greek Alphabet' and other useful technical signs.
For the Fl.55.- you get: the Eprom with its listing (don't forget to state which version you want!), an output routine (32 bytes), the outline for the cable, a list of the possible character-sets of the printer relevant with the set of the jumpers, the way to use the lot.

**CONNECTION CABLE between the Sorcerer and the MICROLINE 80 printer, which enables you to print graphics with the Word processing Pack, without any switch.
Price* Fl.100.- ($67.50 U.S.)

The aforementioned equipment / programs are listed showing the approximate dollar equivalent in U.S. currency. For any of those people that are interested you should make allowance for postage and duty + handling. Whether or not there has been sufficient postage allowed in the pricing is questionable for orders from abroad. We felt that this would be of interest to our members.

THE Z-80 PIO for the SORCERER

If you Don't own disks and you haven't got an S-100 then reading Port FE lately has been interesting, but of little practical use. To remedy this situation a special section,hopefully appearing regularly each month will be devoted to cassette users and what to do with a 50 pin connector besides plugging it into an S-100. During the months to come we will cover various ways to interface the following:

PIO's (parallel I/O)	SIO's (serial I/O)
Analog to digital conv.	Digital to analog conv.
Parallel to serial conv.	Serial to parallel conv.

The Z-80 PIO comes configured with two ports plus handshaking which can be programmed for Input, Output, or Bi-directional modes. I suggest you get Data sheets when you buy the chip. Any Z-80 handbook will likely have a section on PIO's which will help you understand the chip and it's programming. Presented here will be how to connect it to the 50 pin Bus and basic programming for output only. Next month we will go into it further. Once we get the 50 pin edge connector we will have to solder the bell wire to the pins we are using and that will be all the soldering necessary thanks to our experimenters peg strip.

continued next page

What you will need:

1 - 50 pin edge connector (Electrosonic)
1 - 5 Volt 1 amp. power supply (or pins 15 (+) and 1 on parallel port)
16 - L.E.D.S and 16-330 ohm resistors (Active)
1 - Z-80 PIO (Active)
1 - Experimenter's Peg strip (Active)
- Some solid Bell wire for hookup

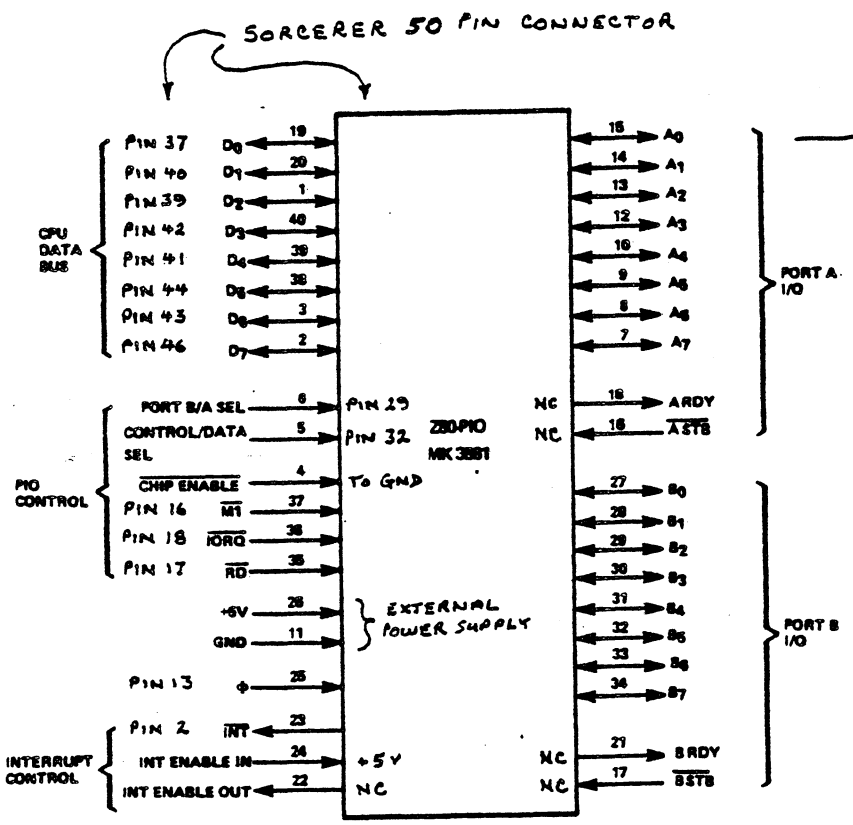
Follow the diagram and hook up as shown. a picture is worth a thousand words and this should present no problems. The why's and wherefores will be presented next month (the Hook). After hook-up is complete plug the connector in and enter this simple BASIC program:

```
10 A=0 : B=255
20 OUT 42,7: OUT 43,7: REM CLEAR INTERRUPTS
30 OUT 42,15 : OUT 43,15 : REM SET BOTH PORTS FOR OUTPUT
40 OUT 40,A : OUT 41,B : REM TWO OUTPUT PORT NUMBERS
50 A=A+1:IF A=256 THEN A=0
60 B=B-1: IF B<0 THEN B=255
70 GOTO 40
```

This program when run will count on the LED's in binary up to 255 on port A(40) and down to 0 on port B(41). The chip is programmed in lines 20 and 30, after which any call to the appropriate port will appear on the LED's. The bits of each port can be connected to diagram 2 to drive a 12 Volt relay or in short to act like the rem lines on your cassette interface. 16 devices can be controlled in this way or write some driving software and connect two more parallel devices. Any problems contact Brad Fowles at night at 1-416-584-2022.

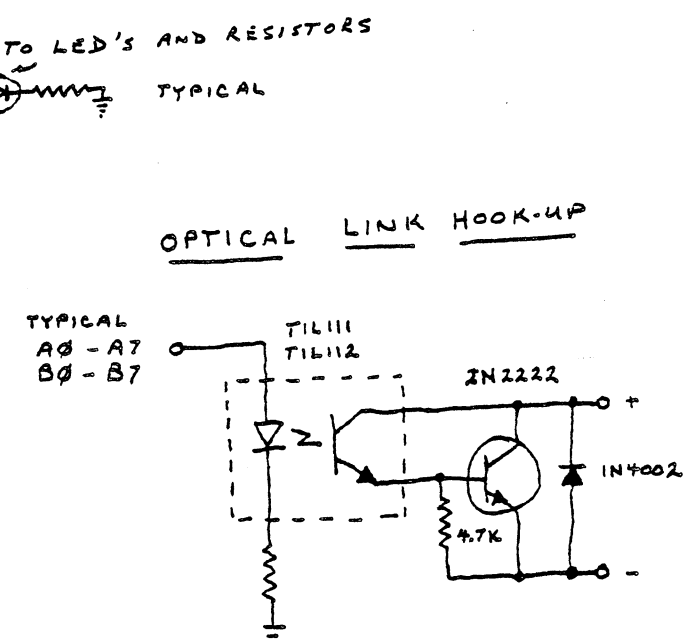
Any hardware types can easily convert this to a circuit board but wait until next month's installment and input will also be possible. Any and all input into this new section will be welcomed. You've got to do something while you're waiting for those tapes to load.

BRAD FOWLES



PIN 4 ⏚ (MUST BE CONNECTED)

DIAGRAM 1



NOTE: SAME HOOK-UP AS IS USED FOR CASSETTE MOTOR CONTROL.

DIAGRAM 2

SORCERER USERS' GROUP (TORONTO)

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 excecutive 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..... S100... Graph board.....
Disk system - Micropolis..... Discus.... Exidy.... other... Size.....
Other Equipment .....
.....

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