OTTAWA IBM PC USERS' GROUP

N E W S L E T T E R

November 16th, 1984 - Issue 84(2)

E X E C U T I V E

Position	Name	Res. Phone	Bus. Phone
President	Mike Luckham	592-1372	592-6500 x2034
Treasurer	Anne Moxley	592-4933	230-9096
Software Librarian	Mike Schupan	230-3755	~~592-5851 x281~~
Editor	Gord Hopkins	828-3834	726-3590
Meeting Facilities	Stu Moxley	592-4933	

Welcome to all our new members! It is nice to see our club growing and becoming more and more organized. For those new to the club, this is the second issue of the club newsletter, at least in this format, so you haven't missed much. In the months ahead we look forward to all your "new blood" stimulating additional club activities, so please feel free to speak up and offer your services in helping to set up special interest groups (SIGs), in providing tutorial experience to other members seeking assistance, or in providing other services or facilities that you feel may benefit the rest of the club.

I have been quite lucky in obtaining articles for this newsletter, but many of them had to come from outside sources. For upcoming issues, I would really appreciate hearing from all you "budding journalists". Relevant topics for submission include programming tips, tutorials on selected BASIC and/or DOS commands, technical discoveries, computer gossip, and finally, but not least, computer humour. If you can't think of anything, get your spouse, covivant, boy/girlfriend to share their perceptions about your "new love". I will accept submissions in either hard-copy or electronic form. For hard-copy, please send your items to:

Gordon Hopkins
17-D Forester Crescent
Nepean, Ontario K2H 8Y1

or bring them to the next meeting. If possible, I would prefer to receive your articles in electronic form. They can be formatted either as a WordStar document file or as a straight ASCII file (printable from DOS using the TYPE command). Those of you who wish to contribute in this manner should contact me (Gord Hopkins) at 828-3834 (evenings) to arrange copying of your files.

NEXT MEETING DATE IS: Wednesday, November 28th at 8:00 p.m.

MEETING LOCATION IS: CAMSELL HALL, Booth & Carling

UPCOMING EVENTS

For those interested, there is a large Computer Show coming up in Toronto, running from Monday, November 19th to Thursday, November 22. It is being held at the International Centre, out by the Toronto International Airport. The show is sponsored by the Canadian Information Processing Society (CIPS) and the keynote speaker at this event will be Dr. David Suzuki. It should be an excellent opportunity to see what is new on the market and to play with all the toys you dream of someday owning.

LAST MEETING: October 31st, 1984

We had an excellent turn out last meeting. There were close to 100 people in attendance; all with a burning hunger for library software. Poor Mike was inundated with requests and sold out our entire library stock by half-time. Many of you had to leave orders for this upcoming meeting. For the next time, we will try to stockpile enough back issues of the library diskettes to handle the hoards.

Our speaker for the evening was Dr. John Nash, President of Nash Information Services. He gave an excellent presentation on the "Trends and Hopes in Microcomputer Software". John was able to give us both a very objective evaluation of where software seems to be heading over the next decade as well as a personal "wish list" for software development. John and Mary, his wife and business partner, graciously made themselves available for discussions after the talk and were among the last to leave that evening. I would like to extend thanks, on behalf of the club, for their contribution to the success of our last meeting.

Nominations for executive positions were called for at the Halloween meeting in preparation for the elections to be held at the upcoming meeting in November. Please consider participating in this process because the stronger and larger our executive the more we can accomplish.

SOFTWARE LIBRARY NEWS

The reserve of public domain software that the club has been distributing on its monthly disks has started to grow again, thanks to the many recent contributions from club members. The software includes: PC-WRITE (a word processor), EPISTAT (a statistics package), a genealogy package, the PC Professor BASIC language tutorial, an Extended Batch Language facility, MVPFORTH (a FORTH Language Interpreter), and the Ultra-Utilities (similar to the Norton Utilities). Most of these offerings are distributed via the FreeWare concept. That is, they are public domain in terms of distribution, but the authors request a nominal donation if you find the software of some use to you. Our club encourages this type of activity and urges you to participate by sending in the appropriate donations if you make frequent use of these software offerings.

We will be ordering more software in the near future so if you would like some say as to what the club purchases, please contact Gord Hopkins at the next meeting, have a look through the catalog of software available, and make your requests known.

SOFTWARE EXCHANGE PROGRAM

I was approached at the last club meeting by Gerry Stuurop. He is interested in contacting other people in the club who have educational software for the IBM-PC and who would be willing to enter into a software exchange program. Gerry is specifically interested in software aimed at the 5-10 year old range. If anyone is interested, please contact Gerry at 592-6500 local 2000.

FOR SALE FROM GENERIC TRANSFORMS

1 - Borland Turbo Pascal c/w 8087 Support	$125	6 - Borland Sidekick, protected version	$75
2 - Borland Turbo Pascal 2.0	75	7 - Borland Sidekick, unprotected version	110
3 - Borland Turbo Toolbox	75	8 - Quaid Copywrite	65
4 - Borland Turbo Tutor	45	9 - Quaid Zerocopy	97
5 - Borland Pascal Gift Pack, items 2,3,4 in one pack	150	A - Quaid Explorer	97
N.B. items 4 & 5 to be released soon		B - GTx Loan Planner	35

Contact: Harry Gross, 824 Fleming Avenue, Ottawa, Ontario, K1G 2Z2 (613) 733-7989

SUBMITTED ARTICLES

Programming Tips for BASIC by Harry Gross

Saving Programs: Make the fist line of your program something like this:

```
10 'SAVE "A:MYPROG.BAS"   ' some remark, smart, dumb or otherwise
```

and you will be spared the hassle of saving a progras under a wrong name, or even worse, saving it under a name of another program already on disk. Just do an EDIT 10, remove the line number and remark symbol, and there you are. For program development, omit the remark symbol and the program will be saved on each run. The second remark symbol prevents a syntax error if there is anything on the screen.

Listing Programs: Make the second line of your program something like this:

```
20 'WIDTH "LPT1:", 128: LPRINT CHR$(15), CHR$(27)+"N"+CHR$(6), DATE$: LLIST
```

This, on an Epson Printer, will produce a listing in a condensed mode, 128 characters across. The first page will be headed by the date, and each page will have a six line skip over perforation at the bottom.

Error Traps: If you use the ON ERROR GOTO xxxxx type of routine, include the following statement somewhere in it.

```
..... KEY ON: KEY 9, STR$(ERL): KEY 10, STR$(ERN)
```

This will retain the error line and type of error on line 25 to remind you where to look for what you are looking for as the program is being debugged.

Taming a Wild ON ... GOTO ... Statement: Using a GOTO with a large number of arguments can be injurious to one's mental health, but a framework can be set up to keep things under control. Start at line 10000 with the following statement:

```
10000 ON A GOTO 10100, 10200, 103000, 10400, 10500, ...      etc.
10100 ' routine 1
10199 GOTO 30000
10200 ' routine 2
10299 GOTO 30000
10300 ' routine 3
10399 GOTO 30000
    etc.
30000 ' exit point
```

A sample routine might look like this:

```
10100 ' A=2, instruction for entry to this routine
10110      PRINT "Enter the last name, then first, separated by a comma."
10120      PRINT "Names, please?", NAME.LST$, NAME.FST$
10130          etc.
10199 GOTO 30000
```

Try to retain a single exit point, avoid jumping around inside the block, and retain unused entry points. They will be needed later. With the framework in place, the block can be expanded with code as needed, and the lines renumbered to fit the final arrangement.

Fixed Drive: IBM refers to their hard disk as a fixed drive because the media cannot be removed from the drive. I refer to my cat, Itsy Bitsy Mouser, as fixed because his drive has been removed.

Mainframe Computers: We in the microcomputer group tend to look down on mainframes as big, expensive, unfriendly dinosaurs. But, they do have some use, such as a source of scratch paper. Usually the beasties spew the stuff out in wads two to three inches thick. These tend to rest on somebody's desk for a few days on their way to the nearest circular file. Turn the wad over and run it through your printer. This is more than adequate for debugging purposes.

A $40 Modula 2 Compiler by Norris Weimer

M2M-PC, version 1.35 requires: IBM-PC DOS 2.0, 128K RAM
Modula Research Institute
950 N. University Ave.
Provo, Utah 84604
(801) 375-7402 US$40+2.50 handling, no credit cards

This is a $40 Modula 2 for the IBM-PC. Modula 2 is of course Wirth's successor to Pascal (and to Modula). It is an improvement upon Pascal, featuring separate compilation, better syntax, different I/O, and the ability to do low-level and concurrent programming. But it remains very similar to Pascal. The August Byte will feature the Modula 2 language.

This particular implementation is an adaption to the IBM-PC of a compiler Wirth wrote. It is a psuedo-compiler: it compiles to M-code (as UCSD Pascal compiles P-code), which is further interpreted when it is run. Thus, to use this system, you run the interpreter/monitor from DOS (interp). To compile, you give the interpreter the command 'modula'. Then to run your new program, you give the interpreter the filename of the object.

The documentation is a trial. It's not organized as well as it should be, and it should have had a bit more editing. It's not that it is wrong or incomplete exactly, but it gives the impression of being too heavy. That is, it presents too many facts without enough indication of what is important and what is needed to get started and what is for later. There are not enough examples, either in the documentation or on the diskettes.

For example, there is a remark in the manual (actually a footnote to the list of files on the distribution diskette) that a certain file must be on the disk used to boot the machine. The compiler will not work otherwise, but everything else works, so it isn't obvious what's wrong; you get the impression that the diskette is full. Little things like how to quit the interpreter are not in the documentation at all. The documentation isn't clear on what the limitations are either; there may be a 64K maximum on memory used.

The included documentation is for the compiler, so you will also need a copy of Wirth's book: Programming in Modula 2, second edition, for US$16.95 more.

In general, the compiler is useable, but Turbo Pascal is nicer to use and faster. The modula system does not include an editor, but it does use a secondary copy of the DOS command processor, so that you can use your favorite editor (or do any other DOS command) without unloading the interpreter (but you do have to wait for the editor to load). Also, it objects to tab characters in source code, and the errors found by the compiler are given by number only, so you have to look up the text of any error message in the manual.

The most awkward part is that the interpreter, the compiler, all the libraries you wish to use, and your source file must be on the default disk. Also, any files the compiler (or a utility program) writes, including temporary files, will also go on this disk. It doesn't actually say this anywhere, so I might be missing the trick here.

There is no debugging facility.

There was a review of this package in PC Magazine, 3 April 1984, page 183. Having seen the package and re-read the review, I find that the review was quite accurate. The package lacks the polish that would earn it an enthusiastic review, but it seems to be solid, and the price is excellent. (It is sold by a non-profit institution.)

The company has announced that its Modula 2 will also be available on the Macintosh in the fall.

The major advantage of Modula 2 over Pascal, in my opinion, is the facilities for separate compilation, which means that 'global' modules (such as graphic libraries, etc.) can be modified and re-compiled, and all programs written using that module will run properly without having to recompile them. This is a part of the language definition. You'll notice that such a specification goes beyond that of the language syntax and semantics - it deals with the way the operating system will work. It, in fact, requires that some kind of dynamic linking/loading is required at run time. This, however, conflicts with many operating systems, such as MS-DOS. MS-DOS likes programs to be pre-linked and stored in .EXE files, which can be loaded and executed. A Modula 2 compiler cannot store programs in .EXE files, because they are only linked to external modules at run-time (as per language requirements).

Thus, a Modula 2 compiler for an MS-DOS machine requires another 'environment' on top of MS-DOS on which to run, because the standard MS-DOS linkers, library maintenace facilities (LIB), debugger, loader, etc. are all essentially useless. All of these tools must be developed from scratch in order to provide a complete environment in which to develop and execute Modula 2 programs. Thus, the writing of the compiler itself is but a small part of getting a complete Modula 2 implementation running on an MS-DOS machine. This is certainly a factor in the products that have been seen to date. The Logitech compiler DID include all of these facilities needed, but the environment (which is essentially an operating system within an operating system) wasn't very polished. I have not used the product mentioned in this article, but I strongly suspect that these factors have played a role in the design of it as well.

My opinion is that in order to have an optimal Modula 2 compiler for the IBM PC (or any machine, for that matter) is that it should be written as an operating system-like product, rather than as a compiler. Modula 2 contains it's own 'system library' like C, so calls to MS-DOS, for example, are not required. For the IBM PC or any machine capable of running MS-DOS, I would suggest a Modula-2 operating system, containing all of the needed tools, with a 'link' to MS-DOS - probably by being able to read/write in MS-DOS file format, and also run MS-DOS programs and return to the Modula OS. A large task, but the only way to get a proper, complete implementation of Modula 2 on a PC (which is a perfect machine for this kind of task - I don't think Modula 2 is as well suited for, say, mainframe use).

Unix-based machines, I would think that a Modula shell, with all of the appropriate utilities, etc. would do the trick.

A Lotus 123 Question by Bill Storey

Help. I can't find the answer in two 123 manuals, and testing hasn't worked. I'm about to assume it can't be done: My spreadsheet has columns headed by month (Jan-Dec) and a TOTAL heading. I want to print on one sheet only, the data for months Jan thru June (columns B thru G) then alongside that, the TOTAL (column N).

I haven'thad any luck with this one. Yet if I print the entire WKS (col B thru N) then I get the TOTAL column on the second page of print but I don't want to print columns H thru M. And I just want one sheet.

I tried moving column N alongside column G and printing the result. The printout was exactly what I wanted, however two unwanted results: (a) Since column N (when moved alongside column G) replaced column H, I then lost the contents of column H forever; (b) the cells in column N were @SUM functions to total all entries in each row. Normally I get the correct values in column N but not so after the MOVE. The @SUM formulae are still in the cells but the answers are wrong.

A N S W E R S:

Gavin Godby:

I can think of one sneaky way to do it. Set up your left border to be the columns(B thru G) you want printed then set your data range to be the total column and go for it.

Stephen Samuel:

You could also just move column H over to, say, column Z where it won't get in the way.

How to Print Screen from a Basic Program

(C) Copyright 1982, Peter Norton, author of The Norton Utilities for the IBM PC -- Circulated via Public Domain Diskettes

When BASIC is operating on the IBM Personal Computer, the 'print-screen' function doesn't work. This article shows how your BASIC programs can perform a print screen operation, and gives you an opportunity to learn more about how the IBM PC works.

Before we explain it all, here is how to print-screen in BASIC. First, you must set up a print-screen routine by executing these instructions, once:

```
DEF SEG = 4095
POKE 0, 205   :   POKE 1, 5   :   POKE 2, 203
PRINT.SCREEN = 0
```

Then, your programs can print the screen contents at any time, by executing:

```
CALL PRINT.SCREEN
```

As long as the "DEF SEG" hasn't been changed, then the CALL statement will print the current screen display.

This bit of BASIC program doesn't make the "PrtSc" key work-- it lets your programs perform the print-screen function. Your program must decide when to print the screen, perhaps by acting on a special keystroke.

To understand how this works, we have to touch on some of the most fundamental parts of the IBM PC.

Built into the IBM PC, in the read-only-memory, or ROM, are many service routines that perform operations that are basic to the computer; among them is the print-screen operation. These services are activated by a mechanism known as a "software interrupt," and print-screen is requested with interrupt number 5.

The programs that service the keyboard are also found in the ROM. The ordinary keyboard service program always checks for the "PrtSc" key: when this key is pressed, the keyboard program performs an interrupt #5, and the screen is printed. The keyboard service routine uses interrupt #5, but doesn't own it exclusively-- any program that knows how to request interrupts can use it.

When BASIC is running, BASIC takes charge of the keyboard, and so the regular ROM keyboard program doesn't see the PrtSc key, and BASIC ignores this key. This prevents us from using the PrtSc key to print the screen.

So, the print-screen operation is not available in BASIC, ordinarily. But BASIC has the ability to accept and perform machine language subroutines. If we set up a program that will request interrupt 5, then we can CALL it, and print the screen. That is what the BASIC program above does.

Here's how it works. Our machine language program consists of only two instructions: one to request interrupt five, and the other to return to the caller. The "interrupt" instruction is two bytes long, and the "return" is one, for a total of three bytes. The three POKE statements above create those three bytes. The numeric values 205 and 5 are the interrupt #5 instruction, and the value 203 is the return instruction.

There is one messy little problem that occurs whenever machine-language programs are used in BASIC-- where to put them. There is no one clean simple way to deal with this problem; it has to be solved differently depending upon the size of the computer's memory, and upon what else a BASIC program might do with "DEF SEG" related statements.

The example above is set up for a computer with 64K of memory, and places the print-screen program in the last 16 bytes of memory. For another memory size, "I" expressed in kilobytes, or "K," this formula will do the same:

```
DEF SEG = I * 64 - 1
```

If I is 64K, then this formula works out to 4095, which is the number used in the example above.

your programs set the "DEF SEG" for any other reason, then it will have to be coordinated with its use for the PRINT.SCREEN program, or any other such machine language program. When PRINT.SCREEN is called, DEF SEG must be set to the same value used when the program was created with the POKE statements.

Printer Bit Image Graphics Demonstration by Dave Browning

This program demonstrates a portion of the Epson printer graphics capabilities when the printer is connected to an IBM PC.

This program has been tested for an Epson MX-100 which comes with the GraftraxPlus option standard. Although not tested, the program should also run as is for an Epson MX-80 with the Graftrax option. The program should also run as is for an Epson MX-70. (The MX-70 has only normal density bit image graphics mode, as opposed to dual selectable modes on the MX-80 with Graftrax or the MX-100, but this program does not utilize the higher density mode.)

The reason for the WIDTH "LPT1:",255 statement in line 120 (use this statement as is even if you have an 80 column printer), and for opening the printer "LPT1:" as file #1 in line 130 is to suppress the line feed that Basic and Basica automatically send after each carriage return (ASCII 13). Since Basic does not check context to distinguish between an actual carriage return and a character with the value 13 used for other purposes, it adds a line feed (ASCII 10) after every character 13 output with a PRINT, PRINT #, PRINT USING, PRINT # USING, or WRITE # statement. The WIDTH #1,255 statement suppresses the linefeed normally added to the end of lines also, so the program has to add it where needed (the CHR$(10) at the end of line 3000).

This subject is addressed in part in several places in the IBM Basic manual. See the OPEN and WIDTH statements as well as the print and write statements mentioned above. You may test this by removing the OPEN and WIDTH statements from the program and changing all PRINT #1 xxxxx statements to LPRINT xxxxx.

If this program still seems to be putting line feeds in strange places, you might check your printer switches for proper setting. Switch SW2-3 on the MX- 80 and MX-100 can be set to automatically add line feeds within the printer no matter what the computer does. See your printer manual for details of getting at and setting the proper switch.

After running the program as is, you may want to experiment with different settings of the parameters such as PRTLENGTH. For MX-100s and MX-80s with GraftraxPlus, changing the 75 in line 2020 to 76 will set the printer to high resolution graphics (twice as many dots per inch). Those with the MX-80 and MX-100 GraftraxPlus printer manual can now proceed with the excellent tutorials on bit graphics included in it. Since the manual does not have a lot of information on the IBM PC connection, this program was developed to show how to overcome the "automatic linefeed" types of problems.

One additional note: if you stop the program during execution with a control-break, the printer may be in the middle of graphics mode and be waiting for more graphics characters. If you then try to RUN the program (with or without changes) again, the printer may seem to do weird things until the buffer and the program are back in synch. Everything will be OK if the program runs to completion, or if you turn the printer power off momentarily before reRUNning the program.

```
100 '### PRINTER BIT IMAGE GRAPHICS DEMONSTRATION ###
105 '###    PROVIDED BY DAVE BROWNING 1/10/83    ###
110 'set up BASIC to suppress automatic line feeds
120 WIDTH "LPT1:",255
130 OPEN "LPT1:" AS #1
140 GOTO 1000
200 'subroutine to output one character to printer
210 PRINT #1, CHR$(X);
220 RETURN
1000 ' ### PRINTER TEST PROGRAM #####
1010 DEFINT A-Z       'runs faster using integers
1020 CHRSTART = 0     'first graphics character to print
1030 CHREND = 255     'last graphics character to print
```

```
1040 PRTLEN6TH = 50   'number of times to repeat the graphics char
1050 PRTLEN1 = PRTLEN6TH MOD 256    'printer expects two numbers-first remainder
1060 PRTLEN2 = PRTLEN6TH \ 256      'then number of "256's"
2000 FOR GRAFCHAR = CHRSTART TO CHREND     'repeats for graphics characters
2010 X = 27 : GOSUB 200          'sends <ESC> character
2020 X = 75 : GOSUB 200          'sends char K to shift printer to graphics
2040 X = PRTLEN1 : GOSUB 200     'sends n1 (# bit images modulo 256)
2060 X = PRTLEN2 : GOSUB 200     'sends n2 (# bit images times 256)(n2 max is 7)
2070 FOR I = 1 TO PRTLEN6TH      'send same character PRTLEN6TH times
2080   X = GRAFCHAR : GOSUB 200  'send the character
2090 NEXT I
3000 PRINT #1, "   X = ";X;CHR$(10);      'notes which graphics character was printed, and sends line feed.
3010 ' This flushes the printer buffer and returns the print
3020 ' head to the beginning of the next line.
3030 NEXT GRAFCHAR    'loop back for next character
3040 CLOSE #1         'resets the WIDTH statement to default
3050 END
```

Some Useful BASIC Function Definitions
by Harry Foxwell, SigSTAT chairman, Capital PC

Conspicuously missing from IBM Basic are predefined functions for computing the maximum and the minimum of two numbers, and a function for converting a character string from lowercase to uppercase.

Although it is fairly simple to code these functions with a few lines using IF statements, using predefined functions makes the overall program more compact and easily readable.

The maximum and minimum of two values can be computed using the following functions:

```
DEF FNMAX(X,Y) = X*ABS(X>=Y) + Y*ABS(X<Y)
DEF FNMIN(X,Y) = X*ABS(X<=Y) + Y*ABS(X>Y)
```

A function that converts a lowercase letter to its uppercase equivalent is:

```
DEF FNUCS$(A$) = CHR$(ASC(A$)+32*(A$>="a" AND A$<="z"))
```

Each of these function definitions depends upon the fact that the comparison operators (<, >, and =) yield a value of 0 when the comparison is FALSE, and yield -1 when the comparison is TRUE. These values, expressed simply as the original comparison, may be used in the same way as any other BASIC variable.

LATE NEWS FLASH >>>>>>

The guest speaker for the November has not yet been finalized, so we'll have to surprise you. Until then keep those disk drives whirring. We hope to see everyone out for the club elections and for the last meeting of the year. Remember there is no meeting in December because of Christmas. A Merry Christmas to everyone.

That's all for now!!!
HAPPY COMPUTING

Gordon Hopkins
Editor