North Texas PC NEWS

February 1983

Vol 2 North Texas IBM Personal Computer Users Group No. 2

New Meeting Time

Time flies when you're having fun!

Members voted at the last meeting to begin our monthly meeting thirty minutes earlier. It seems that two hours is just not long enough to cover all the things we'd like. Beginning this month we'll start the regular session at 9:30. Everything is squared away now and we should have ample space for all. We'll be meeting in the Business School auditorium, 1st floor of the Cox School of Business in the Fincher building at SMU.

Agenda

For the next meeting, the general order of business will be:

1. Club Business
2. Presentation - (Not determined at Press time.)
3. Question and Answer Period (as time allows)
4. Vendor Announcements
5. Club "Professional" Members
6. Specialty Group Agendas for Next Hour (Each SIG Chairman makes 1-minute announcement)

Chris

Special Interest Programs

Programmers

Programmers Group members Mike Hauser and Mike Wiles gave a presentation on driving the PC speaker at the January meeting. Discussion included the hardware block diagram, using the programmable timer, and dual frequency sound reproduction. A procedure for sampling the PC cassette input to produce intelligent speech was also presented.

During the general discussion portion of the meeting, Microsoft was reported to have in production evaluation support for the 8087 Numeric Data Coprocessor chip. The 8087 will be important for high-volume number crunching tasks such as graphics.

February meeting topic:

HARD DISK EXPERIENCES

Suggestions and volunteers for topics for the structured portion of the Programmers' meetings are solicited. Call group chairman Neil Bennett.

Dick Gall

Business Applications

We'll discuss a new software package called 1-2-3, an integrated spread sheet - database - word processing - graphics package. A free copy of the tutorial version will be available to anyone who brings a blank diskette to the February meeting.

Alan High

Next Meeting February 19, 1983

Auditorium, Cox School of Business, Fincher Building
(1/4 Mile South of Heroy Building
SMU Campus
9:30 TO 12:00

North Texas PC NEWS

Published monthly by members of North Texas IBM Personal Computer Users Group for their use. Members each receive a free subscription; for others, price of the NEWS is $1.00 per copy. Advertising is accepted; contact Editor for rates. Members are requested to notify the editorial staff in writing of address changes. Send all editorial correspondence to the Editor, PC NEWS, 2025 Rockcreek Dr. Arlington, TX 76010.

Editor	John Pribyl	(817)275-4109
Programming Editor:	Tom Prickett	(214)387-3667

Deadlines:
All material for publication in PC NEWS (articles and ads) must be received by the NEWS staff no later than the first Saturday of the month in which publication is desired.

L E T T E R S

We welcome letters to the Editor, pro and con, about your club, the PC News, or other related activities. Mail your letters to Editor, North Texas PC News, 2025 Rockcreek Dr, Arlington, TX 76010.

Needs help with printer...

I am seeking information on "how to - " interface a Trendata printer to my new IBM PC. The Trendata printer is an IBM with an ASCII board. I have had it working with my ALTAIR S-100 bus system and would like to connect it to my new system.

If you or any of your members could offer any suggestions I surely would appreciate it. Am including a schematic of the set-up I have been using with the ALTAIR.

I would also like to know more about your club and newsletter.

Robert W. Kelley
5806 Mt Terminal Dr.
Waco, Texas 76710

(If you can help, please see me for the schematic mentioned by Robert in his letter. A complimentary copy of PC NEWS is on its way! Ed.)

North Texas IBM Personal Computer Users Group

A non-profit, independent group, not associated with IBM Corporation. The Group meets on the third Saturday of the month at 9:30 AM in the Heroy Bldg on the SMU Campus.

Officers:

President	Alan Elliott	(214)941-8475
Program Chairman	Chris Morgan	(214)446-0484
Treasurer	Bill Hood	(214)350-9794

Special Interest Groups:

Beginners	Mike Durbin	(214)271-8779
Business Applications	Alan High	(214)385-0553
Disk of the Month	Will Janoschka	(214)231-6449
Programmers	Neil Bennett	(214)238-7650

Special Group

Bylaws Committee	Fred Williams	(214)245-4319

Dues: Professional Membership - $36.00 year
Regular Membership - $24.00 year
Student membership - $12.00 year

Payable in January, dues are prorated for balance of year when applicant joins after January. Dues may be mailed to the editor of PC NEWS.

E d i t o r i a l

Our discussion of member address listings/labels at the last meeting revealed several viewpoints. Some members do not want their addresses arbitrarily passed along. Others feel that access to the complete name, address, and phone number should come with the membership. Then there are those who think that our Users Group should profit if anyone is to benefit from our mailing list.

A possible solution: Sell the list but, before doing so, allow each member to choose whether or not his or her address is to be included. For the sale price we provide to the purchaser one set of address labels. As for the price for each set of labels, it seems appropriate to charge at least $30 (half the $60 price of a full page PC NEWS ad). Any member purchasing a set of labels would get a $4 rebate (1/3 of the yearly "premium" paid by professional members) for each set of labels bought.

The subject should be discussed during the main session of our meeting, the pros and cons aired, and a vote taken whether or not to sell. Then, if response is yea, another vote should be taken on the price to charge.

John

BEGINNERS' CORNER

By Mike Durbin

For months now I have waited patiently for an attendee at the Beginner's group session to ask a technical question. Finally, at the January meeting, it happened. Near the end of the meeting an individual in the back of the room very innocently asked if I could explain the DOS FORMAT command, what it was, why it was required, and what exactly it did. Well, like a barrel of oil penned under ground for several million years and suddenly set free by the driller's bit I commenced to gush and didn't shut up until my audience had started to wilt several hours later. In the gushing process, however, I learned several important facts, such as (1) although only one person had expressed an interest in knowing more about the DOS FORMAT command, just about everyone in the room had an interest (though unexpressed) in the answer to the question; (2) people new to personal computers (i.e. the so-called beginners) do have a real interest in finding out how a computer does its thing, even if they don't openly exhibit such an interest; and (3) my heretofore reluctance to get technical with folks new to computers was, for the most part, a bad idea! To reflect my new knowledge base on this matter this article will be technical. Howsomever, don't panic! It won't be that technical.

The most basic fact about a modern computer is that, when functioning properly, it is binary. This simply means that the instructions which the computer is executing, the data on which the computer is operating, and the internal control information which allows the computer to keep track of what it is doing can all be thought of as sets of binary states. A state is binary if it can only assume one of two values; in a computer the two possible values of a binary state are the binary numbers 0 and 1. In reality, of course, these two numbers are our symbolic representation for differing voltage levels in the machine. A sequence of binary numbers such as '01101001' is one such set of binary states. If this binary sequence is being used as a machine instruction, then it will have a different meaning than it would if it were being used to represent program data. If it does represent program data, then it may represent either numeric or character information. Exactly what a particular sequence of binary digits (referred to as bits) represents is not important to the present discussion. A sequence of eight bits (such as the binary number above) is referred to as a byte. And now let us proceed to the storing and addressing of computer information.

All Gaul is divided into three parts and all computer information (both program instructions and program data) is stored in bytes, so speaketh a wise man some many years ago. So how do I get to one of these byte things. Well, if you are a computer, you do it with an address. Every byte of binary data stored in a computer has a unique address. And, wouldn't you know it, this address is another sequence of those binary digits 0 and 1. In the IBM PC such an address is a sequence of twenty (that's a bunch) binary digits. Such a sequence, when interpreted as a positive binary number, can uniquely address any one of 1,048,576 locations. Now your own computer may only contain memory chips to store 65,536 (i.e. 64K) bytes of information, but this limitation is strictly circumstantial (any number of memory board manufacturers will be happy to uncircumstantialize you on this matter). It is important to understand at this point that these computer-addressable bytes of data we are speaking of are internal to the computer, i.e. they include none of the data which is stored on your magnetic tapes or diskettes (more on these things later). This fact becomes of paramount importance when you reach down and turn your computer off because, as you already know (perhaps painfully), this is when all that computer-addressable data gets flushed back into the universal energy pool (like erased, man). And, because this internal storeage of bytes of data is dependent on the flow of electrons through the computer, and, since we don't want to leave our computers on all the time, we have, to save the day, external storage devices, to which we will now turn our wearying attention.

Cassette tape recorders and disk drive devices are used to store bytes of computer information on a permanent basis (i.e. without the requirement for constantly present electricity). On a cassette recorder byte addressability is of no concern; information is simply stored sequentially on the tape, one byte of information after another. The bytes are grouped (or blocked) for error-handling purposes and in order to create identifiable files of data. Each such file is given a unique file name and any subsequent retrieval and use of the bytes of binary information contained in a file are effected through use of this file name. Data storage on diskettes via a disk drive device is not quite so simple. If you wish to know more on this subject, come to the Beginners' group at the next meeting. Our discussion subject will be data storage on diskettes.

I'll be C'ing you

by Neil Bennett

Nothing is worse than a reformed smoker, except maybe someone who has used assembler code on about a dozen different machines over some 20 years and has now decided that there is a better way.

I have said that SIG doings should be written up in the newsletter so everyone can be in on them (and form a permanent record), so I am putting finger to keyboard following my own advice and hoping that others will do likewise.

The way to get the absolute best performance out of your micro is to write programs in assembler. Sometimes the only way to make peripherals sing and dance in time to the music is to drive them from an assembler routine. However, if you are writing a large program which is going to be around for a few years, assembler may not be the best possible choice. The micro market is changing so rapidly that a few years is an eternity. (Look back a few years; the changes in the future will be even bigger). There will still be machines that can run your assembler code, but there will be other machines that you would like to have your program running on as well. You could rewrite your code in assembler for the new machines, but that gets old quickly.

At this point I feel I should confess my bias. I am writing software for the market place and my income depends on appeal of the program and the size of the potential audience. As a counter-example, I would not be tempted to write a program to calculate the eclipses of the sun for the next 500 years. If the program was for a PC and needed 512KB of memory and the 8087 chip, then I would grow thin off the royalties.

I want to be able to write a system for the present breed of microcomputer and then, when bigger better faster machines come along, be able to crank the handle and put the system on a new machine. I don't want to spend 4 months building the system today and then another 4 months rebuilding it for a new machine.

I can't prove it, but I think the "C" language is about as perfect for my needs as I am going to find. There is just one reference book on C, and that's by Kernigan and Ritchie (known as K&R). The language went through a long incubation period within Bell Labs before surfacing in the outside world. The UNIX operating system is written about 94% in C, and the 6% of assembler is needed to talk to peripherals. There is a short article on C in a recent Softworld. It is about a year ago that I noticed the first C compilers appearing on the micro market for 16 bit machines. I purchased the Telecon C compiler last August and settled down to write my programs in C, with the idea that I could write my program once and it would go onto all sorts of machines (K&R list these machines having C systems:- PDP-11 (the original implementation), Honeywell 6000, IBM 370 and Interdata 8/32 - and that book was written in 1978!). There are C compilers for lots of other machines such as the 8080, Z80 and of course the 8086 (which we all know and love in it's 8-bit form as the 8088).

I write code in a simple style. Even so, I found I was constrained by the very simple code structures in C; until I tried doing things their way. I now find I can put a C program together easily, and it is far more readable than my assembler code ever was. The very simple structures force you to write simple functions. (If the function you are writing won't fit on the screen, it is probably too long.) There was an unexpected bonus with C. When I rewrote a large program (HHB) that I had written in 6502 assembler for the APPLE, I found there was all sorts of near-duplicated code. Each time I had written a similar function in assembler, I had done it in a slightly different way. Because the assembler source was so big I was not able to see the near-duplications. In C, you have to work at it to write code to do similar tasks in more than one way and because the overall system was less lines of source, it was easy to spot the duplications and build one function to replace several.

C allows you to concentrate on what the program is to do rather than the mechanics of how it is to do it. Now, I don't know which registers are being used; and what's more, I don't care. But being an old assembler buff I could not resist the temptation to look at the assembler code generated by the C compiler. I wish I hadn't. I estimated that the code generated was about 3 times bigger than I could do by hand. The only explanation possible was that the

▶

I'll be C'ing you ... 2

C compiler treated the 8086 as just a 16-bit wide Z80; that this compiler was originally written for an 8-bit machine and upgraded to exploit the IBM PC market. However, the programs worked and I pushed on. The original HHB program ran on a 48KB APPLE and I started getting worried when my IBM PC C-based system passed 64KB with more to go. I have 128KB so I could keep on building; the reality of the market place is that there is a bigger market for a program that will run in a 64KB machine rather than a program which needs 128KB. At this point I started thinking there must be better C compilers out there. I also started thinking I could build my own C compiler more easily than translate the C-based HHB program into assembler by hand. (Particularly, if I wanted to make a few changes to HHB in the tuning stages.) So, I held my nose and pressed on to finish the C-based HHB and not look at how big the code space was getting.

As I write this (1/13), my own C compiler for the PC is about 70% complete; the design is about 95% complete and there is lots of tedious clerical work left to generate assembler code for each operation to deal with about 5 flavors of operands. My compiler generates one instruction to increment a counter. The Telecon system, at worst, generates 5 instructions. I also plan to backend the system with a peep-hole optimizer. This is an exotic name for a simple concept. The assembler code will be buffered to about 30 lines before going out to disk. Simple heuristics can be used to find curable inefficiencies in this "peep-hole". All that looks like a mouthful. Here's an example. If the instruction about to go out to disk is a jump, and its target is in the peep-hole, then the jump can be made into a jump short (because the target is within 127 bytes). Another example; to cope with all situations the compiler has to generate code for an "if" like this:- je cc1; jmp cc2; cc1: do some code; cc2: common paths again. If the optimizer finds the cc2 label in the peep-hole, then the code can be simplified to :- jne cc2; cc1: do some code; cc2: etc.

I have used the same register conventions as the Telecon system. This will allow me to build a mixed program with some functions compiled by the Telecon compiler and other functions compiled by my own C compiler. Such an approach has other advantages in proving out functions one by one.

The C Compiler itself is written in C. Can you imagine a BASIC system written in BASIC? It's easy to build a C system in C and there are some definite advantages to doing it that way.

Here's the first. I am building my C compiler using the Telecon system. When I am all through I can use my own C compiler to compile itself. That will result in my C compiler being much smaller, but still generating the same optimized code. Because it generates fewer instructions, it will also run faster.

Here's the second. You can "migrate" the compiler onto another machine. We will both get confused if I don't use a concrete example. The obvious machine that I will want to move to is the to-be-announced APPLE 68000-based LISA. (There was even a picture of it in TIME.) Most C compilers have a distinct code generation stage; this is part of the inefficiency - the assembler code is generated at the very last. To migrate to the 68000, I would change the C system on the 8086 to generate 68000 code. I would then have a 8086 based system I could use to compile C programs to run on the 68000, i.e. a cross compiler. I could then compile the new C compiler on the 8086 cross compiler. This would give me a C compiler which would run on the 68000 and it would generate 68000 code. Sounds simple? This is the migration path that was probably used to move from the Z80 to the 8086. It works, but if the original C compiler was written especially for the Z80, the migrated 8086 system might not be very good. The best way to get a well matched C compiler for the 8086 is to start from scratch with that machine in mind.

Those are a few of the arguments for the assembler buffs. If you write in BASIC why should you consider C? In a recent Creative Computing, Will Fastie compares the execution times of BASIC and C (and others). The infamous Sieve benchmark takes 2020 seconds using the BASIC interpreter and the Lattice C takes 11. That's a ratio of about 184:1!. Obviously that's not comparing like with like (an interpreter vs. a compile and assemble); I suggest you read the article and also the original Sieve benchmark article and it's followup.

What happened to the option of seeing what other C compilers were out there? This turns out to be

▶

I'll be C'ing you --- 3

rather expensive. The Lattice C compiler (or others) may be what I want. I have heard good things about that system and I have also heard bad things. To sample it would cost me $500 and if it is just what I want that would be dandy. If it is not suitable, it would be a rather expensive experiment.

So there you have it. I would love to talk to any people interested in walking through the design of the C compiler (particularly the code generation) or acting as beta-test sites for me and feeding back fault reports. But be warned! It is not full C by any means. It is a subset which I find adequate for what I want to do.

An sicks munce ago I cooden efen spel C.

Neil

REFERENCES

The C programming Language, by Brian W. Kernighan and Dennis M. Ritchie. Published by Prentice-Hall.

The C programming Language, SOFTWORLD November, 1982 Vol 1, No. 3 page 6. Available at SOFTWARES, North East Corner of Spring Valley Rd and Coit in Richardson. Also at other Computer stores.

Telecon Systems, 1155 Meridian Avenue, Suite 218, San Jose, CA 95125. See their ad. in January, 83 BYTE page 218.

TIME January 3rd, 1983, page 26. Also look at the photo on page 23. Do you hear footsteps behind you?

Creative Computing, February 1983, ibm images by Will Fastie, page 278.

A High-Level Language Benchmark, by Jim Gilbreath, BYTE September 1981, page 180.

Eratosthenes Revisited, by Jim Gilbreath & Gary Gilbreath, BYTE January 1983, page 283.

Lattice C. Lifeboat Associates, 1651 Third Avenue, New York, N.Y. 10028. Also see PC magazine, September 1982 (The PC Product Guide issue), Page 214. I counted 7 C compilers in this section.

DISK OF THE MONTH

by Dick Gall

Issues to be available at the February meeting:

1. "BEST OF 1982"
2. JANUARY 1983 (a sellout at 2 previous meetings)
3. FEBRUARY 1983 (see below)

Price: $5.00. Media: DSDD 5" diskettes formatted as single-sided for compatibility with most all PC's. Public domain software only.

FEBRUARY DISK HIGHLIGHTS

Contributed by Bob Van Wyk, the following items originate from the Houston Club:

‡‡ PAC-GAL. A game in both standard and compiled basic. Uses keyboard cursor arrow keys. Variable playing speed. You lose if you don't move!

‡‡ VCBACKU & EWBACKUP. ASCII text instructions for making backups of VISICALC and EASYWRITER.

3 Electronic Design programs, submitted by J.C. Hoisington. Originally published in the October 1982 issue of ELECTRONIC DESIGN. AMPDES, CAPNET, and VOLTREG.BAS.

BYTEMAZE, bits of public domain programming from Neil Bennett. An assembly language implementation of BYTE MAGAZINE'S December '81 article "HOW TO BUILD A MAZE". Includes assembly source code and assembled segments that can be linked into an executable program using LINK. Builds a maze on the PC screen, solves it instantly and leisurely shows how the solution was found by threading through every path until the MAZE start is linked with the end.

CLOCK. Another use for the tool for modern times. Complete with chimes on the quarter hour. Contributed by Claude Head.

Plus more info on customizing WORDSTAR, making WORDSTAR files printable, and documentation for LD.COM (list directory, from January).

DISKETTE FULL NOT TRUE for March. PROGRAM AND INFORMATION CONTRIBUTIONS NEEDED. Call disk-of-the-month Chairman Will Janoscnka at 721-8449 or Dick Gall at 234-8888.

PC COMM

by Chris Jacobs

This month we cover Compuserve, one of the available videotext services. To use Compuserve, you will need to equip your PC as a communicating terminal (see previous articles). You will also need a subscription to the service. Access to Compuserve and Dow Jones News Retrieval can be obtained as a package at an excellent price by buying Radio Shack's TRS-80 Videotex/Compuserve Information Service Terminal Package for approximately $19.95. This package gives you the forms to apply for both Compuserve and Dow Jones. Normally Compuserve has a $22.00 subscription fee and Dow Jones has a $50.00 fee. In addition, both services also include some free hours of connect-time. When you use this package to start the services, both waive the normal fees so your only starting costs are for the package itself. Be certain to buy the package numbered 26-2224 otherwise you will buy software that you could not use on the IBM PC.

For your subscription to Compuserve, you will receive:

1. account number and password
2. user's guide
3. free subscription to Today Magazine
4. free subscription to Update, Compuserve's monthly newsletter

You will pay an hourly rate for connect-time based on:

1. Monday-Friday 5AM-6PM $22.00/hr
2. Monday-Friday 6PM-5AM $ 5.00/hr
3. Weekends/Holidays $ 5.00/hr

The above figures are based on use of Compuserve's own communication network, and if you use TYMNET it costs $2.00/hr more.

Service type 1.
Communications.

1. Citizen's Band Radio Simulator
   has all the channels of CB, with handles, private channels to talk with others without anyone eavesdropping, and the ability to have conferences with groups of users.
2. Electronic mail
3. Bulletin boards
   from wide-open buying and selling to special interest groups
   such as gardeners, pilots, gamers, programmers, photographers and more.

Service type 2.
Transactions.

1. Comp-u-Card
   electronic buying service for more than 30,000 items -- from calculators to refrigerators.

Service type 3.
Information

1. Commodity News Service
2. Archer Commodity Report
3. Standard and Poor's Information File
4. Small Business Report
5. Raylux Investor's Report
6. Better Homes and Gardens
7. Popular Science Magazine online
8. Newspapers

Service type 4.
Games

1. Decwars--multiple player war in space
2. Megawar--advanced version of Decwars
3. Backgammon
4. Concentration
5. Chess
6. Adventure
7. Mugwump
8. Roulette
9. Maze
10. Othello

As you can see from this small list, Compuserve has many areas of interest to many people. Compuserve is not hard to use and dialing it up takes little effort. With a small outlay, you can connect your PC to a world composed of business information, stock reports, special-interest groups, and even talk (via computer) to a dozen people at one time, all located in different states.

Give it a try, and you will find at least one area of Compuserve that will catch your fancy.

Next month, we cover Dow Jones News Retrieval.

Chris Jacobs

Programming Topics

Print Spoolers and Disk Emulators

Well, there you are - you finally jumped over the fence and are now standing in the field instead of the back yard. As far as you can see, the field goes on endlessly. What are you going to do now that you are here? And what does this have to do with computers? I'm talking about that add-on memory card you just slipped into your IBM PC! Sure, it only has 64k now, but it won't be long before you fill it up with chips. And then another card or two, or maybe one of the new generation cards that has more than 1/2 megabyte, a clock/calendar with battery backup and several input/output adapters all sitting in one slot . . .

Now that you have all this memory, how can you use it? The bad news is that you almost can't. Basic can't use it; WordStar can't; VisiCalc can; Multi- Plan can't; Personal Editor can; EasyWriter can't; neither can Pokeman, Donkey, or BPI! It looks like the "can'ts" out number the "cans". It will probably be quite some time before most applications can make use of memory amounts greater than 64K or 128K. You probably require the memory for one or two applications, but it just sits there the rest of the time.

So what can you really do with it? The ideal situation would be to somehow use this excess memory capacity to bring better overall performance and ease of use to the system, which would result in higher productivity for you. The good news is that there are software products on the market today which will do this - "Print Spoolers" and "Disk Emulators". You might have seen these advertised in the magazines or have received one packaged with an expansion card.

This is the first article in a series which will provide information to help you understand what they do, what to look for when comparing the various products, and ways to creatively use their features to obtain higher productivity in your working environment. Although there are hardware devices which will perform the same or similar functions, the emphasis here will be on the disk emulators and print spoolers which achieve these results through software. Both of these products are similar in that they make use of the extra "RAM" memory space in your computer.

First, let's talk about print spoolers. The name is derived from the data processing term "spool", which stands for Simultaneous Peripheral Operations OnLine. This refers to a process in which storage is used as a holding area to reduce the processing delays when transferring data between slow speed input/output devices and high speed processors.

The type of print spooler program that you can purchase for your PC uses the extra memory area to temporarily hold print data, like a water tank. It directs the print data from your applications to this area, fooling the application into thinking that the data has been output to the printer. Your application does not have to wait for the actual printing, which is a slow mechanical process. Since it is not "print bound", or running at printer speed, it will complete the job faster and allow you to begin other work. Meanwhile, operating in the background, the spooler program sends the data from the print holding area to the printer without causing interference to the application.

This holding area is frequently called a print "buffer". If the print buffer is large enough, all of the data from the application will fit in it. In this case the application will complete quickly, and you will be able to do other work on the computer while the printing is being performed. In a situation where all of the data does not fit, then the application will have to wait - but only until there is room in the buffer and not for all of the data to be printed. This process occurs automatically and without any changes to your programs.

Let's compare the time differences between operating with and without a print spooler. For a hypothetical situation, assume that you must complete the editing and printing of a couple of documents using your favorite word processing program, Slow-Typer. Since you have already keyed in the documents, all you need to do to finish the job is to perform about four minutes worth of editing on each document before printing it. The printing will take 16 minutes per document on your dot matrix printer.

Without a print spooler, a time chart of the task might look like the following: ▶

Programming Topics

(continued)

```
Task     Time              Time Scale

--------- ---- 1...5...10...15...20...25...30...35...40

1st Edit    4  ----
1st Print  16      ----------------
2nd Edit    4                      ----
2nd Print  16                          ----------------
```

In the above chart, note that you are idle during the 16 minutes that the first document is printing. You need to perform the editing of the second document, however, but must wait until the first has completed printing before you begin. Then during the 4 minutes that you are editing the second document, the printer is idle, waiting for you! You are tied up for 24 minutes and your printer and computer are tied up for 40 minutes.

With a print spooler, the time chart would look like the following:

```
Task     Time              Time Scale

--------- ---- 1...5...10...15...20...25...30...35...40

1st Edit    4  ----
1st Format  2      --
1st Print  16      ----------------
2nd Edit    4        ----
2nd Format  2            --
2nd Print  16                    ----------------
```

In this chart, it still takes 4 minutes to edit a document and 16 minutes to print it. But after the editing, there is only about a two minute wait. This is the time that it takes your Slow-Typer program to format the document as it is printing it. Of course the spooler is capturing the data, so that Slow-Typer never waits for the actual printing. After this brief wait, you can edit the second document and then print it. In this case you are free after 10 minutes instead of 24 minutes. The printer is free after 36 minutes instead of 40. And the computer can be used for other tasks after 12 minutes instead of 40.

Your productivity is doubled and you become less dependent on the speed of your printer. The best part of all of this is that it happens automatically - once the spooler is installed and operating on your system, you really don't need to be aware of it. And now you can start printing everything in the slower "enhanced" printing mode for a better print quality. Your trusty old Slow-Typer program and slow printer suddenly become more than adequate for your needs.

The next article in this series will introduce disk emulators. You will see another way to rejuvenate that old Slow-Typer program and make it more responsive in spite of the fact that you are now working with large files. Some of the features of the various products will also be discussed.

Michael Wiles

Clear Screen Routine

(Reprinted with permission from the Dallas IBM Personal Computer Club Newsletter #5. Thanks to S.Berlin, and the Dallas Club. Ed.)

The following contribution is from the Raleigh "Ripcord" Newsletter. This is a simple but excellent routine illustrating many useful points. The purpose of the routine is to create a command named "CLEAR.COM" to be used for clearing the screen from DOS. I think that this little routine is absolutely elegant in its approach and its simplicity. I give a real "tip of my hat" to Bob Gibson who provided it.

```
1000 REM: GENCLEAR.BAS - Program to build a DOS COM file
1010 REM: Author:  R. A. Gibson  Written:  18 July 1982
1020 REM: Enter and run the program, which builds a file
1030 REM: on the default drive.  The program reads the
1040 REM: data, and uses a checksum to verify that the
1050 REM: has been entered correctly.  If any of the
1060 REM: value (including the checksum) are not
correct,
1070 REM: the message "Data in Error" is displayed.
1075 B$ = SPACE$(17)
1080 CHECKSUM = 0
1090 FOR I = 1 TO 17
1100   READ B
1110   CHECKSUM = CHECKSUM + B
1120   MID$(B$,I,1) = CHR$(B)
1130 NEXT I
1040 READ B
1150 IF CHECKSUM <> B THEN PRINT "Data in Error":END
1160 OPEN "CLEAR.COM" AS #1 LEN=17
1170 FIELD #1,17 AS A$
1180 LSET A$ = B$
1190 PUT #1
1200 CLOSE
1210 DATA 30,51,192,80,184,6,0,183,7,51,201,186,79,24
1220 DATA 205,16,203,1698
```

Programming Topics

(continued)

Clear Screen Routine (continued)

The assembly language program which is being represented by the data statements in the above BASIC program, is as follows:

```
                .xlist
                page 62,132
                page
                .list
cseg            segment para public 'code'
                assume  cs:cseg,ds:cseg
clear_screen    proc    far     ;Clear screen command
                org     100h    ;DOS Program Prefix
                push    ds      ;Save the Segment Pointer
                xor     ax,ax   ;Program Prefix Offset
                push    ax      ;'return' start of prefix
                mov     ax,06h  ;Function Number - Scroll
                mov     bh,07h  ;Attribute Byte
                xor     cx,cx   ;Upper Left Corner
                mov     dx,184fh;Lower Right Corner
                int     10h     ;Video I/O Interrupt
                ret             ;Return to DOS
clear_screen    endp
cseg            ends
                end
```

Once the file (CLEAR.COM) has been built, all you need do is to type CLEAR from the DOS prompt.

Tips on Producing User-Friendly Software (Friend or Foe?)

One of the new "buzz" words we're beginning to hear in the computer business is "friendly." Machines are friendly. Software is "user-friendly." We may be on the verge of a great outpouring of friendliness from computer designers and software authors.

This is the first of several articles I will write for PC News this year on the subject of making your software more "friendly." While I often use the word "friendly" myself, I'm going to make an effort not to use it much in these articles. It lends a human attribute to an inanimate thing. If you don't mind, I'll use the word "usable."

"Usability" is equated in the trade with "ease of use." Software, for example, is said to be usable if it's easy to use. Actually there's more to it than that. To be usable, it also has to make the user's job easier. A teaching tool is usable if it reduces the effort to learn something, reduces the effort to do something, and forgives human errors.

Now, in most cases, the user could care less what your program does on the inside of the machine or how it does it. (An exception to this is the program that takes longer than necessary to do a task or fails to keep the user informed when delays are inevitable.)

Besides the normal output of a program, the user sees on his screen messages from the program asking for information. This requires him to enter something at the keyboard, which may call for another message, then another entry, and so on. This dialog between person and machine is the single most important factor in the user's perception of friendliness or usability. Messages can be worded, even timed, in such a way as to relax the user and let him perform his task with the least frustration. Or they can raise his blood pressure, tie a knot in his stomach, and seduce him into making mistakes.

Take the example of a program that uses both INKEY$ and INPUT statements. The INPUT statement requires the user to press the Enter key to send what he typed in. But most INKEY$ routines are coded to "grab" the keyed character and act on it. There is probably no evidence that one of these methods is any better than the other for keyboard entry. But the program that MIXES the two is just causing trouble for the user. Easywriter 1.0 was this way.

One problem of mixing the use of INPUT with INKEY$ is that the user has no way of telling if he needs to press the Enter key. Suppose he has developed a rhythm of entering data followed by the Enter key. If he does that

▶

Programming Topics (continued)

with your INKEY$ routine, his Enter will be buffered for the next INPUT statement, which may be the wrong response for it. He runs a risk of clobbering a file or losing a lot of time to get back to where he was. Worse, he may become totally lost.

Do you understand the feeling, Mr. Programmer, of being completely lost in an undocumented computer procedure when your precious diskette is sitting in there waiting to be written on? The feeling is acute anxiety, and it can get very bad for some people. It's very important that you keep your end user in mind when you design your dialogs.

There are several ways to present the computer's half of a dialog. The program can ask questions that the user has to answer. The program can put up a "form" on the screen, and the user fills in the blanks. The program can present a menu of choices, which the user selects with a key entry. The Function keys (the ten keys on the left of the keyboard labeled F1-F10) can be used for input. The program might provide a list of commands that the user can type in to do certain tasks. The program could even have a language with its own syntax that the user has to learn.

Each of these techniques has its own advantages and disadvantages. Future articles will cover each of these dialog techniques in more detail. Perhaps in a programming class I can cover some technical material that would be hard to read in a newsletter.

There has to be a reason for a dialog of any kind. Your dialogs should have the following objectives:

1. Keep to a minimum the number of actions the user has to take to run the program, ▶

Pgm. Ed. Note -

An important usable feature of word processors is the ability to communicate these "words" to the outside world. Two of the articles this month were received via telephone line, output files from different word processors. One of the files had to be post-processed before it was sent. The other looked awfully strange when it got to the other end. The ability to output to a standard DOS file would be a very desirable feature, kind of a least common denominator. Most of the word processors seem to provide a way in, but no easy way out. Could it be that these friendly software developers are friendly only to the extent of their best interests?

Tom Prickett

2. Don't require the user to memorize a lot of stuff.
3. Let the user have some flexibility; don't box him in.

Several things you want to keep in mind when planning your dialogs are to design them around tasks the user has to perform and to design for tolerance of user errors. You might also recall the old adage "Never give the operator a task that the machine can do for itself." And remember that programs improve with age; input never does. You must ALWAYS check keyboard and diskette input to prevent errors from getting into the system.

With regard to the last point above, you might consider that one reason programs get better as they get older is due to the feedback the author gets from his users. Any good programmer will fix a bug he finds himself, but many bugs are found by the users rather than the author. It would be a good idea to try to build in a mechanism for getting user feedback. Andrew Flugelman's FREEWARE(tm) provides this. As a result, Mr. Flugelman has made numerous improvements to his first program, PC-TALK.

Munson Hinman

PC Update

The following news items are reprinted with permission from Computer Industry Update published by Industry Market Reports, Los Altos, CA.

- - - - - - - -

Lazor Systems of Sunnyvale, CA introduced the Retailer point-of-sale software package which includes a retail cash register for use on the PC. A register with 16K bytes of memory, an RS-232 interface and software is $4,950.

- - - - - - - -

Pure Data Ltd. of Markham, Ontario in Canada announced the PDI 464 combination multifunction board with six separate functions including expansion memory, two serial ports, disk emulator and a print spooler. A 256K byte version is $1,095.

- - - - - - - -

Quadram of Norcross, GA announced three multifunction boards which combine memory with an RS-232C serial port, RAM disk simulation and printer spooling. The board is available with memory capacities from 64K bytes to 512K bytes at prices from $475 to $1,295. The firm is also offering the Quadboard and Quadboard II with 256K bytes of memory, system clock, spooler, RAM disk, RS232 serial port and either a parallel or serial port. Prices range from $595 to $995.

- - - - - - - -

Techland Systems of Mount Vernon, NY announced the Blue Lynx which allows the Personal Computer to emulate the IBM 3270 series controllers, display devices and printers. The card is $690.

Resources II

by Carrington Dixon

After my first article, I had Autumn Revolution called to my attention. Revolution is certainly a resource that every PC owner should be aware of. It is, however, somewhat different from the magazines that I discussed in the first article and probably, in that respect, deserves the separate treatment that it is getting here.

Autumn Revolution is a magazine and it is a users group. It is one of the several national users groups that have formed around the IBM PC. A recent issue of their monthly newsletter lists ten local chapters (none of them in Texas). A software and technical library is available to members. Until recently a technical hotline was available, but this service was discontinued as it failed to pay its own way and was a financial burden on the organization.

The software library is extensive. I counted over forty entries in the partial listing in the current newsletter. Most, if not all, of these seem to be programs that are unique to Autumn Revolution rather than software that is commercially available elsewhere. There are a number of business applications and a variety of games. Prices vary from $12 to $140. There is a software diskette available for each issue of the newsletter ($15) that contains all of the programs and VisiCalc templates discussed in that issue. There is an annual diskette that contains all the programs etc., for the whole year of newsletters (that's the $140 item).

The technical library consists mostly of commercially available books. This makes it less special than the software library. In the Dallas area there are so many places where you can buy these books off the shelf that there is little need to order by mail. It is a worthwhile service for those living outside the big cities, though.

The most visible feature of Autumn Revolution is their "monthly" newsletter, Revolution. (Monthly is in quotes as they seem to have published only seven issues their first year). Revolution reminds me of the early issues of Personal Computer Age. It has the same semi-technical bend, the same shortage of illustrations or pictures. Do not let the term "technical" scare anyone off. Yes, you will find things like the System Memory Map in Revolution, but you will also find articles which deal with the basics. The latest issue contains articles on expanding memory from both the programmer's and the user's view. For the business application user, there is a series of articles on analyzing securities with VisiCalc. For the program developer there is an article on Thread, a new language developed for the PC which was described as "like FORTH without the Reverse Polish Notation".

Judged solely as a magazine, Revolution is expensive for what you get. If you join the users group, it may be cost-effective. It would be a major undertaking to judge the worth of all the software that they have made available - I doubt that any one person could. If you think you might use the software library, then they are certainly worth a try.

Carrington

- - - - - - - -

Autumn Revolution
P.O. Box 55329
Tulsa, OK 74155

1-year membership $30.00
2-year membership $55.00

PC NEWS Financial Report

August through December 1992

INCOME:	
Advertising Income	$ 255.00
Cash from Group	544.00
Total Income	$ 799.00
EXPENSES:	
Postage	$ 185.39
Office Supplies	61.42
Copies	14.05
Printing	460.50
Art Supplies	13.01
Telephone Calls	6.14
Total Expenses	$ 740.51
BALANCE : Jan 1993:	58.49
	$ 799.00

SHOP & SWAP

(Four lines free each month to members, otherwise cost is 15 cents per word. Ads will run three consecutive months unless notified earlier to cancel. Ads must be renewed after three months if continuation is desired. Mail ads to the Editor.)

CUSTOM PROGRAMMING services and consultation. Call Chris Morgan at 739-5895.

FOR SALE: Printer interface for Olivetti typewriter (ET-121). $350 installed. Call John Williams at (214) 699-7472 or (214) 987-7479.

INSTRUCTION COURSES for IBM PC (Architecture, DOS, BASIC, WORDSTAR, VISICALC, dBase II); Private custom programming services (free estimates). Contact Mike Durbin at 840-9344.

WANTED: Advertisers for PC NEWS. Rates are reasonable, exposure is great! Circulation approximately 120 PC users/owners. Contact J.P. Pribyl, at phone 817/275-4109 for particulars.

Street Electronics' ECHO-GP VOICE Synthesizer. $289. COMPUTER GOLD Microsystems, 2417 Coleshire Drive, Plano, TX 75075. We accept Visa or M/C. Phone (214) 596-2598. Southwest dealer for Street Electronics