The 68xxx Machines

Vol. 1, Issue 3, April 1991 Price: $2.75

The 68xxx Machines is published and copyright (C) 1991 by Catham House Company, RD#1 Box 375, Wyoming, DE 19934. Ph. (302) 492-8511. The editor is James H. DeStafeno. One year USA subscription is $12.50. Canada and Mexico $14.25. All others (surface) $17.25. All major credit cards accepted. Our low prices reflect a 10% cash discount. Please add 10% to credit card orders. Any size display advertising is accepted. The half page rate is $10/issue. Write for other size/duration rates.

Readers are encouraged to contribute letters, articles, programming information and other material related to computers with the 68xx(x) processors; excepting Macs and Amigas. Please send material to the above address. Thank you for your support.

The Editor's Thoughts

By Jim DeStafeno

Well... Due to Ron Anderson's treat, its even a tighter squeeze this month, but its well worth it. We have had lots of questions about REXSDOS and SK*DOS; what are they, what are they like, where can they be gotten, etc. Ron has done a yoemans job discussing them. He even tossed in OS-9/68000 for good measure. Guess Ron is one of four or five people that have written this article. He has long term in depth first hand experience with the three operating systems. This makes him uniquely qualified to undertake the task. I count us luck to have such a man on staff.

And speaking of "good guys", van der Poel finishes his three part series this month. I'm not a "C" programmer, yet I've been able to enjoy his efforts. I hope I've convinced him to share with us more of the ways he has used to solve vexing programming problems.

Still havn't heard from IMS on the MM/1 or Frank Hogg Lab.s on their machine(s).

This Issue:

Advertiser's Index

Microware has just announced a translater program that converts source code of their 09/68K BASIC to their 'C' compiler source code. That not only means a Basic program can be compiled to M/L, but when written on one computer it will run on all the computers the Microware 'C' language runs on. I feel this is a major pargramming advancement.

Lastly, if you've written a short program, have a new piece of software you'd like to evaluate or have other information our readers might be interested in, let us know. We are always interested in printing articles from new writers. And, as always, your comments and criticisms are welcome.

So, kick back and let Ron and Bob take you for a ride; which would be hard if not impossible, to get anywhere else.

Classified Ads

- WANTED SS-50 equipment. SWT CPU card, also Gimix PIO #28 (30 pin) FDC. Alen E. Gordon, MD / 160 NW 176 St / Miami, FL 33169 / (305) 653-8000.

- SALE Complete, ready to plug in; all hardware, software, manuals and cables; super fast 20MB hard disk and 35/40 track, double sided floppy disk drive; both in one case, for CoCo I, II or III. Works with both BASIC and OS-9. HD is partitioned. Used sparingly; $525. Jim DeStafeno / Rd 1, Box 375 / Wyoming, DE 19934 / (302) 492-8511.

- SALE Tandy/TRS-80 Model 100 laptop computer. Built in 300 baud modem. BASIC, text editor, communications software built in ROM. Has 32K RAM. Includes modem and printer cables and all manuals. Good condition. $175 including shipping. Phone Randy Krippner: (414) 853-7044.

o o o

Turn that old computer equipment into cash with a *68xxx* classified ad. For Sale ads are $5.00 per 50 character line. Wanted ads are just $2.50 per 50 character line.

Selections In C

Part 3 of 3

By Bob van der Poel

Last issue, in Part 2 of this series, we created an array of pointers to functions which let us create a cursor with different video attributes. This month we will take this concept one step further and set up an array which contains not only function pointers, but also a set of unique values paired to each function. This type of table is very useful in the main input parser of a program.

My text editor VED has a main loop which simply waits for a keypress and then branches to an appropriate routine: if the keypress is a character then we go to the edit() routine, if it is a control we execute the appropriate function. This could be done with a massive SWITCH..CASE, but the method presented in the following fragments is much easier. First off, a structure is needed which will hold both the keyvalues and the corresponding functions:

```
    struct jumpent{
       char key;
       int (*fn)();
    };
```

Next, let's have a look at the idle loop:

```
    mainloop()
    {

       /* let the compiler know which
          routines we'll be using */

       extern int addmacro(), append(),
find(),
                 block(), delete();

       register int k;

       static struct jumpent cmds[]={
          'a'-0x60, addmacro,
          'g'-0x60, append,
          'f'-0x60, find,
          'b'-0x60, block
          'd'-0x60, delete,
          0,0;
       }

       for(;;){
          k=curkey(x,y,*curpos,0);
          if(k>=' ' edit(k);
            else   if(dojump(k,cmds))
doerror("Unknown function");
       }
    }
```

Of course, VEDs list of functions is much longer. The main jump table has over 60 entries. Both the structure and the loop are more complex, but this example will suffice for our needs here.

What we have done is to set up an array with entries matching each possible keypress and the corresponding function. In this example, if CTRL-A is pressed we want to call the function addmacro(), for CTRL-D we want delete(), etc. All we need now is a function, dojump(), which will examine each entry in the array looking for a match and, if one is found, call the corresponding function.

```
    dojump(c,tbl)
    char c;
    struct jumpent tbl[];

    {
       for( ; tbl->key; tbl++){
          if(tbl->key==c){
             (*tbl->fn)();
             return 0;
          }
       }
       return -1;
    }
```

This function receives the key to look for the base address of the table. It loops through the table until a match or the end of the table is found. If the character 'c' is found in the table the corresponding function is called and a 0 is returned. If a match is not found a -1 is returned so that the caller can alert the user that an unknown or illegal option was selected.

The idea of jump table selection can be expanded for your own needs. For example, there is no reason for the match characters to remain type char. With a simple modification they could easily be changed to integers, floats, or even strings. It is also possible to change things so that parameters are passed to the functions, and with some trickery the functions could even return values.

By using pointers to functions, we can create compact, fast C programs. Due to space limitations, there is only space to show code fragments and simple examples, but with this base you should be able to expand this technique and adapt the concept to your own programs. If you have any comments on this series or suggestions for future articles please drop me a note here at the "The 68xxx Machines" or directly to me at PO Box 355, Porthill, ID, 83853.

REX, SK*DOS and OS-9/68000

By Ron Anderson

This is a "special report" in response to, "What is REX, MONK, SK*DOS and OS-9/68XXX?" (But lets call OS-9/68000 OS-K) The focus is on REXDOS and MONK because they are the newest operating systems available for Peripheral Technology computers. We've been talking about SK*DOS for several months, and OS-K has been mentioned as well. REXDOS was developed several years ago by Dan Farnsworth for his own use. It and MONK, a simple startup boot / debugging 68000 program, (normally called a monitor program), along with OS-K and SK*DOS are now available for the Peripheral Technology 68000 computers.

>>> GETTING REX UP <<<

At work I have a number of PT68K-2 (PT= Peripheral Technology) computers and an older -1A. I have a PT68K-2 at home too. The company presently doesn't have any of the new PT68K-4 systems, though our next addition will most likely be one of them.

When Dan Farnsworth asked me to check out his REX and MONK combination, I reminded him I didn't have a -4 system. "That's OK, REX should run on a -2 just as well. We'll have a chance to see if there are any glitches." So I agreed to the test.

After two weeks of sessions, some lasting until 2 AM, I had it all working. Don't get me wrong, it wasn't difficult, but I had a lot of stuff I wanted to port over to REX. If the stuff couldn't be ported, I wasn't much interested in using REX for anything.

First job was to get my system configured. I have some old 40 track standard DSDD drives. (Dan tells me they are obsolete. However, the company probably has 20 or 30 of them on various older systems used for development work). The system I choose to use for the test has a Monochrome video board and monitor, a 20 MEG hard disk, an 80 track floppy and two 40 track floppies. I disconnected the hard drive for the preliminary tests.

I found I could boot from the supplied system disk. I immediately duplicated it per the instructions in the manual.

Next I enabled the 40 track drives. Soon, I noticed the floppy MAKDISK utility formated with no sector interleave (i.e. I saw sequentially numbered sectors on each track). REX not only reads a track at a time, but no interleave means it reads a track in one revolution of the disk drive. Wow! I formatted an 80 track disk that way. It turned out to be really fast reading and writing.

Next I tried to format a 40 track disk, but got a fatal error. A call to Dan with the problem rewarded me a few days later with a new MAKDISK utility. It seems MAKDISK didn't read the drive steprate from the Drive Information Table (DIT) correctly. It tried to step the 40 track drives too fast.

With all working cleanly I tried Dan's editor, EDDI, and found the documentation didn't apply to the video monitor and IBM keyboard configuration I was using. Another call to Dan got me the correct information for commands.

As soon as EDDI was running, I decided to work on the Whimsical compiler. After a short conference with its author, John Spray (who works with me), I modified the necessary system interface code. With a little debugging we (John with my help), had a cross compiler running; (running under SK*DOS on one machine while compiling code for PAT to run under REX). I soon had PAT, my own editor running under REX.

When long time computer users switch systems, they (speaking for myself too) try to make the new system look just like the one they are used to. For example, FLEX had a LIST utility that displays a text file to the screen stopping every 24 lines until the user hits ESC or space.

When I got into MS-DOS I found TYPE lists a file with no stops. Rather then TYPE, I typed LIST and got a NO SUCH FILE error so often I made a batch file called LIST.BAT. It simply told the computer to type the file and pipe the output to MORE. MORE stops listing every 24 lines and prompts the user to press a key to continue.

A week later we (again John with my help), had Whimsical running under REX and capable of compiling itself; making an exact copy of itself able to run under REX. Along the way, I backed all my files off the hard disk and reformatted it with the REX format utility. I had a slight problem getting REX to boot from the hard disk. A call to Dan got me the answer the next night by phone. All works fine now. I've even modified my pet home made utilities so they work under REX as well.

>>> WHERE DID REX COME FROM ? <<<

Lets look at a little history. Dan, like myself, was a long time user of computers based on the 6800 and 6809 processors. The main operating system available for those processors was FLEX, supplied with the hardware of South West Technical Products Co. FLEX was written and supplied by Technical Systems Consultants, originally located in West Lafayette Indiana; presently in Chapel Hill North Carolina.

TSC as they are known, have long since dropped support of FLEX (which at least operationally, I understand, was a copy of an operating system developed by DEC for their PDP series of computers). TSC has gone on to bigger and better things in the form of Uniflex, a multi-user, multi-tasking operating system developed first for the 6809 and later extended to run on a 68000 based computer.

Dan and lots of others including myself, liked FLEX a great deal. We always felt we had something better than CP/M used by the machines with the 8080 and Z-80 Intel processors. In retrospect, some of that might have been the "my dad's car is better than your dad's car" syndrome.

Anyway it was agreed the 6809 is vastly easier to program in assembler (all we had at first) than it was with the Intel processors. Therefore... it was not unreasonable for Dan to write an operating system for the 68000 hardware very similar in "look and feel" to FLEX and the earlier DEC operating system.

In fact, he has done more than that. He also wrote an extensive ROM monitor program called MONK to run with REX.

I must point out both REX and SK*DOS have maintained complete text file compatibility with the old FLEX. In addition, SK*DOS has compatibility with binary program files as well. That was a natural desire since the early development of software for the 68000 systems had to be done on 6809 systems using cross assemblers. A cross assembler is an assember program that runs on one machine but generates code for a different processor or operating system. (In the process of getting a lot of familiar software tools transferred over to a new processor or operating system, having a cross assembler or compiler is of great value, as is disk compatibility.)

While old FLEX binary file format is not quite compatible with REX files, it is easy to write a program to convert FLEX files to REX compatible files. (Dan has already done so). I won't get into the binary file structure or the differences in great detail. They are fairly trivial, Dan having left the "short record" form of FLEX out, probably for simplicity and size. (A cross assembler or compiler that runs under FLEX and produces code for SK*DOS could be easily modified to produce code for REX.

Normally magazines don't want comparison articles written because there is usually a winner and a loser. That doesn't help advertising revenue. ["68xxx" says let the chips fall where they may. Ed] In this case, I think I can make a good case for choosing any of the three operating systems. I've mentioned OS-K briefly previously, but not in great depth. There are numerous factors to consider in choosing your operating system. Let's look at each in turn.

>>> A LOOK AT REXDOS <<<

There are several good reasons you would want to consider REXDOS. First of all it is now supplied with all Peripheral Technology PT68K-4 computers. These computers can use high density floppy drives (the ones used by AT and newer style IBM clone computers). That means you can install a 1.44 megabyte 3.5 inch drive and/or a 1.2 megabyte 5.25 inch drive.

They read and write more quickly than conventional double density drives because they pack the data more closely. One of the main features of REXDOS is the floppy disk file handling. REXDOS (May I use REX for short?) handles disk files by reading and writing a whole track at one time.

Because of the whole track approach, the sectors are physically on the drive in the order in which they must be read. A whole track can be read in one revolution of the disk. By timing test, disk operations (read and write) are about 2.5 times faster using REX than the times of SK*DOS.

I'd better expand on that. The test consisted of copying a very long file from hard disk to floppy. Dan's separate read and write tests indicate reads from a floppy are twice as fast and writes are seven times as fast. Actually it is not quite that simple because REX reads a whole track even if only one sector is needed, so REX is a bit slower with very small files, but considerably faster for large files. I am told a -4 with the high density drives reads and writes files about 2.5 times faster then my old -2. That approaches the disk operation speed of IBM clone.

As for terminals, REX will run a serial terminal, a monochrome (Hercules) board and monitor and/or any color monitor from CGA to VGA. The PT uses third party graphics adaptor boards. They can be bought inexpensively form Peripheral Technology or your own computer "goodie" source. Graphics adaptors vary somewhat. Some use "different" hardware which they make compatible by means of a onboard BIOS ROM. REX requires a board with "standard" hardware. Check with Peripheral Technology for approved boards.

Even with these features, REX is a very simple and small single user operating system. Even so, it is rather difficult to add other co-resident programs. Perhaps the average user wouldn't want or need to do that, at least not for a while.

REX gets along fine with software written for loading at absolute addresses. That is, it doesn't require position independent code. Of course, programs written in position independent code work fine under REX too. Operating systems calls take more bytes of code than with SK*DOS.

In general there are fewer and less capable OS calls than in SK*DOS. Most of those who would be capable of writing assembler code interfacing with the operating system would be able to code the missing routines rather easily. As Dan put it, the operating system features supplied are generally those required for the operating system. Where functions might be useful to a user, Dan has made entry points available to the user. The usual OS calls exist for such things as opening a file to read or write, closing a file, etc. There are calls to output a string of characters terminated by $04 (a leftover from FLEX in these days when C uses $00 and most others follow). I have converted several Utility programs from SK*DOS to REX versions with little difficulty.

The original copy of this discussion contained one strong negative reaction to REX due to the lack of a built-in multiple directory feature. One night I was discussing this with Dan and he suggested multiple small partitions for the hard disk. My immediate reaction was I would then have directories of fixed sizes, which struck me as a disadvantage.

However, after kicking the idea around, I soon had a scheme that could change partitions leaving the working drive number alone. I won't go into detail here since the subject is complex enough to deserve a whole article at another time. By the time I was finished, I had 15 partitions on my hard disk (aside from the drive 0, the system disk partition). Thirteen of those are 30 cylinders or just under 1 Megabyte. Two are 60 cylinders, just under 2 Megabytes.

By the time I was finished, I had figured out how to give partitions names, and change REX's Drive Information Table by overlaying different starting sectors. This allowed switching directories by name (CD TEXT would switch to that directory). I was also able to make commands to find the name of the present directory, (PD shows the Present Directory name).

OS9/68000 SOFTWARE

QUICK ED - Screen editor and text formatter $275.00
A high quality documentation tool and program editor ideally suited to laser printer users. Uses function and cursor keys on any terminal, configurable per user. Microjustifies mixed proportional text. Automatic table of contents generation and user-definable macros and commands. Drives any printer. Ideal for multi-user systems. Available on a 30-day try before-you-buy basis.

FLEXELINT V4.00 - The C source code checker $495.00
Flexelint finds quirks, idiosyncracies, glitches and bugs in C programs. 60 options control checking by symbol name or error number. Checks include intermodule inconsistencies, definition and usage of variables, structures, unions and arrays, indentation, case fall-through, type conversions, printf and scanf format string inconsistencies, and suspicious semi-colons. A must for all serious C programmers.

IMP - Intelligent Make Program $250.00
IMP does everything you wished Microware's Make would do, and a great deal more. It is well-behaved, consistent, and extremely flexible. It has a built-in C-like preprocessor and has comprehensive debugging facilities. Rules can be user-defined, and make files for jobs other than assembly-language or C compilation are easily constructed.

DISASM_OS9 - OS-9/68K Disassembler $250.00
This high-speed, three-pass 68000 disassembler can also handle the 68010 and 68020. It intelligently decodes module headers and produces symbol information that can be repeatedly edited and passed through the disassembler allowing iterative disassembly. The system libraries are read to supply symbols.

WINDOWS - C Source Code Windowing Library $250.00
This C source code library package supports multiple overlapping windows displayed on one character-based terminal screen. It supports window headers and footers, and pop-up windows. Windows may be moved, panned, written to while off-screen, etc.

PROFILE - User State Program Profiler $270.00
Designed to profile user-state programs. Profile effectively samples a traced execution building statistical information as it goes. It reads symbol table modules to give a function-by-function account of the time spent during execution. The user may "zoom in" on a function to find a smaller range of addresses where time is being spent.

PAN UTILITIES - C Source Code Utility Set $250.00
Forty useful utilities are supplied in this C source code package. Included are utilities to move files, find files, patch disks, undelete, cross-reference C programs, set and remove tabs, and spell-check documents.

PC9 - MS-DOS to OS-9 Windowing System $350.00
PC9 allows an MS-DOS computer to be used as a terminal to multiple processes on a remote OS-9 system linked by a single serial cable. Each OS-9 process is displayed through a resizable, moveable window on the PC screen. Terminal emulation facilities support uMACS and other screen editors and provide a programmable PC keyboard. Access to PC disk drives is also available through the OS-9 unified I/O system, giving disk capability to ROM based OS-9 systems. A hot key switches between DOS and OS-9 displays.

*** delmar co ***

Middletown Shopping Center - PO Box 78 - Middletown, DE 19709
302-378-2555 FAX 302-378-2556

OS-9 is a Trade Mark of Microware Systems Corp. Flexelint is a Trade Mark of Gimpel Software. MS-DOS is a Trade Mark of Microsoft.

SYSTEM IV COMPUTER

THE **SYSTEM IV** is a high performance computer system based on the Motorola 68000 microprocessor operating at a clock speed of 16 MHz and has been designed to provide maximum flexibility and versatility. Microware's Professional OS9/68000 operating system is included with the **SYSTEM IV** providing an efficient multi-user and multi-tasking environment. This provides the user with a PC for home use, small business applications and a viable low-cost solution for many industrial control applications (embedded systems). Special requirements (such as midi, sound, A-D/D-A, net-working, etc.) are easily handled with readily available low-cost PC/XT boards which can plug into the **SYSTEM IV** expansion slots. And, as user requirements change or improved special function boards become available, they may be added or replaced at the user's option. Thus, when software requiring multi-media or other new capability becomes a reality, the user will be able to add that capability easily and have the latest technology at his disposal.

TO ACCESS THE LARGEST SOFTWARE BASE available, an MS-DOS board, the ALT86, will be available shortly as a low-cost option. This board has a V30 (8086) microprocessor running at 10 MHz, includes 1 Meg of 0-wait state RAM, uses the Chips and Technology BIOS, has a socket for an 8087 math co-processor and plugs into one of the **SYSTEM IV** expansion slots. Additionally, an OS9/6809 software emulator/interpreter will be available soon. The emulator/interpreter will permit running most COCO OS9/6809 software on the **SYSTEM IV**.

OTHER OPERATING SYSTEMS may be installed. These include CPM, UNIFLEX, MINIX, STARDOS, REXDOS and most any other operating system capable of running on the 68000 microprocessor chip.

THE DESIGN OF THE SYSTEM IV is derived from previously successful designs and uses components that have been tested and proven in other systems. SYSTEM IV's uniqueness stems from the ability of its designer and manufacturer, Peripheral Technology, to provide well designed, reliable hardware at a low cost. Further, only the functions necessary to the basic operation have been designed into the mother board. Seven PC/XT compatible expansion slots allow an unrestricted selection of standard PC/XT accessory boards by the user. The user is not locked into any preconceived notions of what is best.

THE MOTHER BOARD is a 4 layer XT size board which holds the microprocessor, sockets for up to 4 MBytes of 0-wait state RAM, a battery backed-up clock, 4 serial ports, 2 parallel ports, a high density (37C65) floppy disk controller, 7 PC/XT compatible expansion slots, a memory expansion connector to allow an additional 6 MBytes of 0-wait state DRAM, keyboard connector and the necessary system support chips.

THE TERMINAL SYSTEM includes the mother board with 1 MByte of on-board DRAM, a high density floppy disk drive (3 1/2" or 5 1/4"), 4 serial port connectors, a parallel printer port connector, a 200 watt power supply, mini-PC style case capable of holding 5 half-height drives and Professional OS9/68000. This configuration requires the use of an external terminal(s).

THE CONSOLE SYSTEM adds a VGA (800 x 600 x 16) graphics board and an AT style keyboard and provides full graphics capability at the console. Terminals may be added.

THE SYSTEM IV comes with a one (1) year parts and labor warranty.

TERMINAL System	$ 999.00
CONSOLE System	$1,149.00

OPTIONS

3 MByte additional DRAM		$120.00
Hard Disk Controller and driver		$ 69.00
40 MByte Hard Disk		$295.00
20 MByte Hard Disk		$240.00
Additional 5 1/4" or 3 1/2" HD Floppy Drive		$ 92.00
AT Style keyboard and 800 x 600 x 16 VGA Card and driver		$159.00
For 1024 x 768 x 256 VGA Card w/1 Meg of Memory in place of standard VGA card	add	$170.00
Mono Display Card in place of VGA card	deduct	$ 50.00

Prices subject to change without notice.

Special monitor prices when ordered with the SYSTEM IV.

See the PERIPHERAL TECHNOLOGY AD for kits.

*** delmar co ***

Middletown Shopping Center - PO Box 78 - Middletown, DE 19709
302-378-2555 FAX 302-378-2556

Finally I could make a new directory or remove one, or get a list of all the active directories.

The fixed size for each is not really a major disadvantage. A major ADVANTAGE of the scheme is each partition has its own directory sectors. The directories tend to be short or at least shorter than if all the files on the disk were in one large one, so disk access is faster overall.

The directory complication only arises if you are using a hard disk. If you plan to use only floppy drives, you can use a different disk for each kind of computer activity. Each disk becomes a new directory.

Another initial disadvantage, REX was missing the ability to redirect input and output as SK*DOS and OS-K allow. The scheme of using device drivers and device numbers is not implemented.

The substitute for output to printers, the ability to run two different printers on different ports is very good. The code which runs the printers is essentially a device driver, though a bit simpler and less flexible in nature than that of the other two OSs.

SK*DOS has the input and output redirection to files via the mechanism of the < and > operators on a command line. I've written a couple of utilities to handle those operations. Old FLEX had an I and an O redirection utility that did just what you think they might suggest. I've written them for REX as well. No doubt Dan would have done so if he had needed them for anything.

They are not very complicated. Using them (they are public domain and will be available from Dan, Peripheral Technology or me), is quite simple. Suppose you want to assemble a file and run the listing to another file rather than the printer. Normally you would assemble the file: ASMK FILE +BGS

To shift the output to a file called FILE.LST you would use the O utility like this: O FILE.LST ASMK FILE +BGS

The output file specification must follow the O which must be first. O opens the output file, substitutes some code to write to that file in place of the terminal. Essentially it overwrites the OUTCH vector in REX. When control is returned to REX, the input line pointer is right at the A of ASMK, so it assembles the file. The output goes to FILE.LST. At the end of the assembly, ASMK or any other program does a JMP WARMS. Warm start of REX closes all open files, so FILE.LST is closed. It works fine. In fact, when you use the P for output to a printer, the printer driver does pretty much what O does.

Dan has developed a pseudo disk directory system called SUBCAT which allows different directory files. They all remain in one large directory per drive whether or not you use SUBCAT. (This is true of SK*DOS as well. There is basically one large directory, but files are tagged with a directory code).

Using SUBCAT is a simple process. You can sort files into directories. Then you can use the features of SUBCAT, which presents you a screen directory listing from which you may choose a file by bumping the cursor down through the list. Then you may choose one of several options to edit the chosen file, assemble it, delete it, copy it to another drive or disk, view it on the screen, etc.

One very nice feature, you can insert up to 36 characters as a comment to go along with each directory entry. (We've all at one time or another, done a directory of our system disk and wondered what several of the command files were or what they did.) A disadvantage of the system is you are never "in" a directory as such. Newly created files are not automatically placed in the current directory. You must remember to "load" them, probably an operation which would shortly become second nature. Once you have "loaded" a file to the directory, its name stays in the directory regardless of how many times you edit and reassemble it. (The name stays in the directory even if you delete the file without going through SUBCAT, i.e. directly with the delete utility).

If you like a menu style of operation as opposed to using a command line, (i.e. choosing from a list of possibilities rather than simply telling the computer what to do), you will like this utility a great deal. On the other hand, if you like command line style, you probably will avoid it.

You might begin using REX with SUBCAT, then a you become familiar with the system switch to using the command line directly.

If you are a real hacker like me, you probably will have an overwhelming reason to want to have REX. Dan has made it public domain. If you want to develop software or utilities to do something special, you can peek at the source code and figure out how it works so you can attack the problem.

I ventured into such a project just a couple of nights ago. I was trying both a video (Monochrome) terminal with an IBM clone keyboard and a serial terminal. I was checking whether everything worked correctly in switching back and forth between them. I needed two different versions of my editor PAT, for no other reason than I needed a different terminal configuration file for the serial terminal.

After digging in the MONK source I found MONK places the address of the keyboard handler in a trap vector at memory address $74. Also byte $76 became $1B when the terminal was live and $1C when the monochrome was live. I used that fact to detect which is active and load the appropriate PAT version, all transparent to the user. The point is, it was easy when I could look at the operting system source code.

Dan or Peripheral Technology will supply all the source code for REX and MONK. I think the price is $15. With it you can program your own MONK ROMs. If you prefer to buy the ROMs also, the charge is $20.

This discussion wouldn't be complete without a mention of one very nice feature of REX. It has a built-in terminal emulator which operates when you use it with an IBM keyboard and monitor. It accepts a substantial subset of the commands for a Televideo 925 terminal.

Part of the reason it was so easy to get PAT running under REX was the terminal emulation. I simply selected a configuration file I already had for a Televideo terminal and modified it a little. Dan claims the IBM keyboard is "fully decoded". However I found the cursor keypad arrows to emit the same codes as ^H, ^J ^L, and ^M. That is easy to get around. When you get a key and it is one of those, you can look at a variable, which is non-zero if the control key is being pressed.

When the arrow keys are used

alone, the control key is up. Of course when you type ^L, the control key is down. I found I still couldn't distinguish between the left arrow and the backspace key since both emit code $08 and the control key is not down for either of them.

Dan has been very quick to fix things and to add things I have requested. What is left to clean up is in the area of differences in the operation of very similar utilities, or perhaps in skimpy documentation. I am in the process of trying to help Dan with the documentation. I've been so busy playing with REX I haven't made a lot of progress. However I now know enough about REX to be able to do an intelligent job (I hope).

REX is the simplest of the three operating systems and as such might be an easy one to use while learning about the 68000 processor. Enough software is presently available for it to be useful. Dan has his editor EDDI, the assembler ASMK, the directory system SUBCAT, as well as his spelling checker SPELLB for sale.

I have PAT running under REX, subject to a little more testing. I've been using it for several weeks with no new bugs found. The Whimsical compiler is running fine. Thanks to John Spray, I also have a cross compiler version running which I can use to compile programs on the SK*DOS machine that can be read and run on the REX machine. In addition, Dan is working on a BASIC interpreter, and there is a good possibility of a C compiler in the future.

>>> A LOOK AT SK*DOS <<<

SK*DOS was written by Peter Stark, STAR-K Software. He is a long time computer hobbyist, writer of computer articles, teacher of computer science, and friend (I hope still).

Peter wrote SK*DOS several years ago. It had been shipped with Peripheral Technology hardware until just recently. Athough SK*DOS is also based on FLEX, it has capabilities beyond those of FLEX. It is written wholly in position independent code, therefore even all the utilities are position independent. This makes it easier to stack programs in memory without interference and gives greater overall flexibility.

Whimsical, a language similar to Pascal that compiles to M/L, which runs under SK*DOS also generates position independent code. However, SK*DOS's "C" compiler generates absolute position code; which is a little inconvenient.

SK*DOS can run with a serial terminal, monochrome (Hercules) board and monitor, or with a CGA board and monitor. I understand Peter is working on EGA/VGA compatibility presently. SK*DOS is available for the -4 hardware. It supports high density disk drives; both 3.5" and 5.25". A revised version for the older -2 computers can also support high density drives via a floppy controller board that plugs into the I/O bus. It is supplied by Peripheral Technology.

SK*DOS can be purchased from Peripheral Technology or from Peter Stark. Contact either for details on the current version.

One reason why you might want to try SK*DOS is the built-in multiple directories. In SK*DOS you are always in a "current directory" on your working drive. If you edit a file, or create one, it is automatically done in the current directory. You can use directories on your system drive too, but I don't. In fact you can simply ignore multiple directories if you like.

Another reason for using SK*DOS it has more software. RBASIC from Bob Jones in Canada runs very well. The "C" compiler available from Computer Systems Consultants in Georga works well, as does ASM, their assembler. ASM is a necessary part of the "C" compiler.

Dan Farnsworth has SK*DOS versions of EDDI, SPELLB, his assembler ASMK, and the catalog program SUBCAT. I have my editor PAT and a simple text formatter called JUST. There are several other utility programs written by users, available through STAR-K's bulletin board. I have several I can supply to anyone who wants them. More about that in a later issue of this newsletter. (Most all of them will also be available for REX.)

I ought to mention batch files. In SK*DOS the first filename on the command line is assumed to be a command (.COM) file. If that file is not found, SK*DOS searches the system drive for a file of the same name with the extension .BAT.

If one is found, it is used as an input command file. That is, it substitutes input from that file for command lines entered on the terminal. The switch from command file to batch file is mode automaticlly. Long ago, I wrote a batch file called FORMAT3.BAT. It calls the format utility. Prompts for format are answered by redirecting the input to a file that has all the correct response answers. SK*DOS input files have the extension .PIP so my input file is FMT3.PIP and the batch file is a single line that looks like: FORMAT 3 <0.FORMAT3.PIP

```
  FORMAT3.PIP contains:
    40
    dddblank
    1
    y
```

These are the necessary answers to the prompts, being careful to have CRs only where they are needed. I use a default disk name of BLANK and a default number of 1. Later you can use DISKNAME to rename the disk, give it a number and date.

SK*DOS is a more grown up version of the FLEX style operating system then REX. There are more features available, it is more modular and more flexible. This should allow for easier updates and improvements. On the other hand, the sourse code for SK*DOS is not available. Also SK*DOS has slower floppy disk read / write operations.

>>> A LOOK AT OS-K <<<

I have used OS-K on a 68020 system, a 68008 system and very briefly on a CoCo (6809). It is a high capability operating system from Microware that supports multiple users and multi-tasking. Its I/O is interrupt driven. As an example, printing can be done (not via a spooler) while editing another file, and a compiler running in the background.

I found the 68020 system to be so fast it wasn't practical to try to do multiple tasks on it other than printing a file or a listing while doing something else. I would start the C compiler and go to edit something else only to find the compiler done before I had written three or four words.

Of course it shines with two or three users on the same computer.

From one or more users, the requests cause the requested programs to load into memory. Each user runs them from there from his own variable space. In the event a user attempts to load a program that is already in memory because a prior user is running it (Multiple users) the OS notes the program is already in memory and just sets up a variable for the new user. Both users use the same program, which is in memory only one time.

I found I could load PAT and run two different terminals as two users with no problems. Each user has his own variable space so when my time slice comes, all my variables are in use, and when yours time slice comes, all your variables are in use (including the variables from the two different terminal configuration files). The result, your terminal can work with different commands than mine. Essentially we each have our own program counter during our time slice, which makes my running of the program independent of your running of the program. (Note, this is totally different then what MS-DOS machines call networking, but appears to the user to be much the same as is done on UNIX systems. Ed)

OS-K has many features that make it a mini UNIX system. It has multiple directories and sub-directories with file path specifications equivalent to those in UNIX; similar to MS-DOS. It uses device descriptors and drivers that are completely modular.

OS-K is easily adapted to different hardware configurations. Some time ago there was a 6809 version that ran on everything from the SWTPc to the Radio Shack Color Computer. (OS-K is very I/O and/or memory intensive. As such a hard disk is manitory for any serious work.)

It is well supported by its creator, Microware. There are many competing software packages available for it including PLuS, a compiled language from Windrush Micro Systems in England; Microware's BASIC-09, a mix of BASIC and Pascal; a very complete C compiler from Microware; a very complete (and expensive) Pascal from OmegaSoft, several word processors and editors, a 4th generation databas, an assembler, and other language compilers including (I think) Fortran.

Granite Software Systems supplies software to do directories, and read and write text files back and forth between OS-K and MS-DOS formatted disks.

OS-K has a couple of drawbacks. First it is EXPENSIVE. Second it is not easy to configure for any given system. I found I had to read instructions long and carefully to get it set up for my system. By the time I needed to do that operation again, I had forgotten how and had to go through the same learning process once again.

Few people have mastered writing device descriptors and drivers for it. The documentation is large and fairly complete, but not well organized; therefore difficult to follow. (I once had to skim through about 80 pages of the manual to figure out how to write the header for an assembler program). In my opinion, if you want a full blown operating system in which to get deeply involved you would like (Love. Ed) OS-K.

However, if you are a computer USER (not interested in programming) and want a wide variety of any one application like desktop publishing, spreadsheets, word processors with on-line spelling checker, on-line thesaurus, etc. you don't want any of these operating systems. In fact, you don't want one of these 68000 computers; not at least for the present.

None of the software presently available for any of these applications is equal to the high level capabilities you can get for a Mac, an IBM desktop or clone. However, if you want to learn about computers and how they run, operating systems, developing useful software utilities, or maybe even more ambitious projects in the realm of graphics programming or word processing, you will like one (or maybe even all three) of the above OSes.

If anyone is interested, I have written some utilities to interchange text files between SK*DOS and OS-K disks. Running under SK-DOS, these allow you to do a directory of an OS-K disk and then to copy files from one format to the other. I may well soon have REX versions too.

>>> ALL OF THE ABOVE ? <<<

It would be nice to be able to run any or all of the above operating systems on the same computer. Unfortunately presently you need to change monitor ROMs to do so. Monk is included with the PT hardware to allow the booting of REX or OS-K. STAR-K's HUMBUG monitor can boot SK*DOS or OS-K. I've been asking Dan Farnsworth to add the capability to MONK to boot SK*DOS too. I don't think SK*DOS uses much code from Humbug, if any.

Dan is presently working on the same capabilities for REX. Its not much fun to change the ROMS to switch back and forth between systems. I sweat over damaging pins and making a set of ROMs unusable. Maybe Peripheral Tech could do an adaptor board to hold both sets of ROMs. They could be installed at the same address with an enable voltage supplied by a physical switch. That way, the user could switch ROMs and run either operating system.

Unfortunately the three systems require different hard disk formats. SK*DOS will let you partition the hard disk so you can run OS-K on one part and SK*DOS on the other. Either can be booted from the SK*DOS power-up monitor prompt. Right now your "main" system can be run from the hard disk and a couple of secondary systems from floppies. As the old saying goes, "ya pays yer money and ya takes yer choice".

Could I make one last point? If you are thinking of one of these systems at least consider a hard disk. If you look hard you can find an ST-225 drive without controller for around $185. The controller from PT is $79 for a total of $264.

If you go with a floppy drive you will soon find you need at least 2 floppy drives and those will cost you just about $200. Of course with the hard disk you will still need one floppy drive to transfer disk supplied files to your hard disk.

You could use a floppy and a RAMDISK. Of course RAMDISK needs more memory then without one. One megabyte of RAM can be gotten for under $100. (I remember when I paid $129 for 16 Kbytes of memory). However, I am rather wary of RAMDISK. Maybe I'm getting too old and absent minded but I tend to turn the computer off having forgotten to copy any new files from RAMDISK to a physical disk. I am also afraid of power failures. Again, it is a personal preference.

Anyway, I hope you enjoy using one or more of these operating systems. Happy computing!