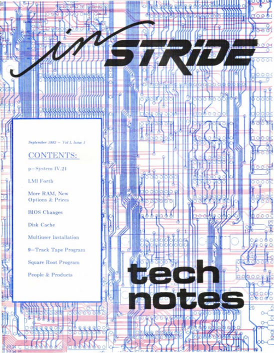
in STRIDE



September 1985 – Vol I, Issue 1

CONTENTS:

tech notes

New Tech Format For In Stride

by Verlene Joyce Bonham

Welcome to the *In Stride Tech Notes!* Stride has launched this new magazine, a daughter to *In Stride* , especially to support our users. It will concentrate on technical issues while the parent magazine *In Stride* will address product and marketing concerns.

As a subscriber, you get both magazines. *In Stride* will be published every third month (November is the next issue) while the *Tech Notes* will be published eight times a year, that is, every month that the parent isn't. This combo gives us a better way to publish actual programs, letters and fixes for user support (Yes! We have been listening!) and still provide monthly product information.

Also, the six week preparation time for the slick color format meant real scheduling problems for a company that moves as fast as Stride. With our nifty little laser printer, we can push the Tech Notes through press in only about a week –– the information will be more timely and, hopefully, more accurate.

The new format is less costly to print, so the subscription price has been lowered accordingly. (See the back for pricing). Those of you who are owners and get it free probably don't care, but as we are getting more and more real *paid* subscribers this change allows us to offer the most information at a reasonable cost.

Despite the howls from our third party software and hardware vendors, we've decided that the Tech Notes are *not* the place for ads. We'll still offer them in the *In Stride*. (Warning: November ad space is full already and we're wc-king on February). If you're a vendor and you want to brag a little bit, send us an article on some technical aspect of your system. It won't cost you anything either!

From the comments above, you can already tell that the Tech Notes are going to be have a much more "down home" style. Hopefully, the notes will cut through some of the marketing puff too prevalent in our industry and get down to issues that concern our users. If talking about bugs and optimized code scares you, there will still be plenty of reviews and how–to notes –– just skip over the other stuff.

If you don't like something speak up. We'd like to see more user letters, notes and tricky programming routines from you. It's best to send it in softcopy; Stride p–System, CP/M–68K or Unix cpio diskette format are best. That way, errors won't creep into the copy when someone retypes it.

The split in the new format should please those of you yelling for more technical support and still let us strut our stuff in blazing color once in a while.

In this attempt to add more "beef" to our publication, we can only quote the fella in the current hamburger chain commercial:

"We're confident that this meets with your approval ..." □

New Fast p–System Interpreter

Available in the new p–System IV.21 release (see below) is a new interpreter. The SYSTEM.INTERP file has optimized code for increased system speed, especially applications that do many string operations.

This is a special interpreter, not the standard SofTech release, and is customized only for Stride computers. It will not work on any version of the p–System prior to IV.21 so don't try to mix and match.

The new interpreter increases throughput from 10% to 25% depending on the application. □

Version IV.21 p–System Now Available

This issue devotes a lot of space to the details of the new IV.21 p– System release. That makes it only fair to tell you how to get it.

Historically, Stride offered versions IV.12A, IV.13 and IV.20 of the p–System operating system. Several important operational differences occured between IV.13 and IV.20. The IV.21 version has several *new* features and bug fixes, but code files are essentially compatible with IV.20.

In addition to the new p–System, Stride has updated the 400 Series BIOS and included several optimization features to speed up applications.

Stride continues to support our earlier computer line and a full release of IV.21 will be available for the Sage.

This new release will be shipped standard with every new machine starting October 15.

To **upgrade** to the p–System Pascal compiler, you have to provide proof of a prior purchase of the Program Development Kit. The PO# on your order is best.

Prices given are in USA dollars, international folks must check with their dealers for pricing.

There are really three upgrades: p–System IV.21 for the Stride 400 Series, p–System IV.21 for the Sage II and IV and the 400 Series BIOS upgrade.

It is a good idea to purchase the full p–System upgrade and not just the BIOS upgrade. The all–important IV.21 machine license will be needed if you intend to order diagnostics and other support programs from Stride.

p–System IV.21 – 400 Series

Part number SF0159 is the run–time package which retails for $99 and consists of the SYSTEM, UTILITY, and BUILD diskettes.

SF0160 includes the above plus the Pascal compiler and 68000 assembler on a fourth diskette called DEV for $150. (This requires prior purchase of the Program Development Kit).

p–System IV.21 – Sage II & IV

SF0161 is the run–time package which retails for $99 and consists of the SAGE, SGUTIL, and SGBLD diskettes.

SF0162 includes the above plus the Pascal compiler and 68000 assembler on a fourth diskette called SGDEV for $150. (This requires prior purchase of the Program Development Kit).

BIOS Update – 400 Series

DK0100 contains a new 400 Series BIOS, MU.BIOS, UTIL.CODE and MU.UTIL.CODE for $25.□

Forth & CP/M–68K

by Ray Duncan

You have a program to develop. And a deadline to meet. You need a prototyping environment that makes your productivity soar. Do you turn to the Forth programming language?

Chances are that you don't ... yet. Forth has long been recognized for its extremely efficient program development environment –– a compiler/interpreter with an integral editor, assembler and other tools. But until recently, the Forth programmer had to give up all of his other development tools because Forth insisted on being an operating system in addition to a language.

Laboratory Microsystems has specialized in bringing Forth to operating systems that are industry–standard. We first brought a Z–80 Forth into the CP/M world. Then a 8086 Forth into the MS–DOS environment. Now, we have harnessed the power of 68000 Forth and put it within the CP/M operating system.

Exploiting CP/M–68K

How does Forth work with CP/M–68K? Simple. Forth appears as a CP/M ".68K" file, which is an executable code file to CP/M. When you invoke Forth (by typing its name), CP/M loads and runs the file, and you are in the Forth language environment.

Forth itself performs all I/O through CP/M–68K functions. For instance, console I/O is done through the BDOS (Basic disk I/O, a part of CP/M) function that performs direct console character read and write.

Programs written in Forth are traditionally stored in Forth "screens." A screen in Forth is a unit of source code that consists of 16 lines of 64 characters per line. LMI Forth stores screens into normal CP/M– 68K files –– they can be copied, deleted, etc., just as other CP/M–68K files. In the past, one reason for using Forth as an operating system was to gain a performance advantage over "general purpose" operating systems in reading and writing disk files. Stride has made this argument obsolete! Using a Sage II we recorded the amount of time needed to load a file from a 40–track floppy disk and a RAM disk. The results are impressive –– the RAM disk loads only 30% faster than the floppy disk!

Using the Power of Forth

LMI Forth is a highly optimized version of 83–Standard Forth. The Forth Interest Group (fig) publishes standards of transportability for the Forth language, their latest standard is Forth–83. LMI doesn't stop with the 83 Standard, though, but adds its own proprietary Forth nucleus, assembler, editor and CP/M–68K interface. Editing is done with a special Forth screen editor that responds to WordStar(tm) control codes. All of your favorite editing functions are there, in a "what you see is what you get" format.

One of Forth's unique features is the ease with which assembly language may be mixed with high–level code. In LMI Forth, a full Motorola 68000 assembler is on–line at all times. To ease the transition into Forth, this assembler uses the Motorola mnemonics instead of a derived set.

Part of a Family

Forth users on the Stride are no longer an island with LMI Forth. We make Forth–83 compilers for PC–DOS, MS–DOC, CP/M–80, CP/M–86, CP/M 68K, ProDOS and UNIX. And high–level Forth programs developed with one will run with all. Many companies use LMI Forth to bridge between home computers, PC's and supermicros like the Stride.

For extremely large programs, LMI has developed a 32–bit version of Forth–83 that exploits the full addressing range of the M68000 processor. It allows you to write Forth programs up to the limit of your RAM space, yet it runs only 5–15% slower that the 16–bit version of LMI Forth.

Finally, for the most sophisticated users of Forth, there is the LMI Forth Meta Compiler. This program takes an

```
Screen #53

( Sage II BIOS calls from 68000 LMI Forth          08/18/85 )

10000 BEGIN-MOD  ASM68K  END-MOD

FORTH DEFINITIONS DECIMAL

CODE SBIOS       ( param_area  fcn_code ––– param_area )
        (SP)+ D0 MOVE.          ( get function code into D0 )
        (SP) D5 MOVE.           ( get param_area, leave copy)
        0 d(BP< D5) A0 LEA.     ( convert to physical addr )
        14 # TRAP.              ( call Sage BIOS )
        NEXT.                   ( reinvoke inner interpreter)
END-CODE

FORGET-MOD       ( get rid of assembler module )
-->
```

```
Screen #54

( get time from Sage BIOS                          08/18/85 )

CREATE TIMEVAL 4 ALLOT

: @TIME          ( ––– time_in_60ths )
TIMEVAL 6                        ( put params on stack )
SBIOS                            ( call Sage 'read clock')
2@                               ( get time value in 60ths )








```

application developed in LMI Forth and compiles it for a different processor. With it, you could use the Stride as a development system for IBM PC programs or for ROMed applications running on an 8051 microcomputer. Currently, the Meta Compiler can generate code for the 68000, 8086/88, 8080, Z–80, 8051 and 6502 microprocessors; and new processors are added each year.

A Forth Example Using the Stride BIOS

Here's an example of how Forth can be used with assembly language to extend and customize the language for the application at hand. Suppose you need to read the current time from the Stride clock under CP/M –– trouble! CP/M–68K doesn't even know the clock exists, and has no way to access it in a portable fashion.

The builders of the Stride installed a very powerful BIOS (Basic I/O System) in the System ROM of the computer. To access it, you need to set the D0 and A0 registers of the 68000 and invoke TRAP 14. In screen 53, shown on the next page, the word SBIOS takes the BIOS function number and the address of a buffer, performs the call and returns to high–level Forth. As you can see, a "word" in Forth is like a procedure or function in another language.

Using the new Forth word is as simple as using any of the built–in words. In screen 54, the word @TIME (pronounced "fetch time") uses the Stride BIOS word to read the value of the system clock in 60ths of a second. Any other functions of the Stride BIOS may be invoked simply by preparing a buffer of suitable length and using the SBIOS word.

More Information

If you are interested in CP/M–68K Forth or have questions about it, feel free to call Laboratory Microsystems at (213) 306–7412. Our customer support bulletin board, at (213) 306–3530, carries news and information about LMI's latest products and includes a comprehensive database of both Forth and non–Forth public–domain programs. It is available between the hours of 6 p.m. and 8 a.m. Monday through Friday and all day Saturday and Sunday.□

Program Changes

Various small changes have been made to the following programs of the Stride p–System IV.21 release.

All MISCINFO files

If you have your own SYSTEM.MISCINFO file, you will need to make the following changes before running on IV.21:

```
HAS EXTERNAL SOCKET POOL            is FALSE
CODE POOL SIZE                      is 16383
SEGMENT ALIGNMENT                   is     4
MAX NUMBER OF USER SERIAL VOLS      is     6
FIRST SUBSIDIARY VOL NUMBER         is    26
SOCKET POOL SIZE               stays at    0
```

TOPSYS.CODE

A new file TOPSYS4.2 replaces the old one. It has more features to allow file transfer on systems with only one floppy. (Our thanks to C. Emery of TDI Canada for the changes.)

TERMINAL.CODE.

A new terminal selection is "STRIDE". It now accommodates new system units and patches SYSTEM.MISCINFO for code pool and socket pool options.

STRIDE.CODE & SAGE.CODE

STRIDE.CODE is a new program for the 400 Series that figures out the memory size, disk size and BIOS version of the machine. It is set to display on boot.

SAGE.CODE is the Sage II and IV version of STRIDE.CODE .□

My terminal's setup and the machine's serial port setup are different. But it works! How come?

Settings such as 7 data bits with 2 stop bits will work as 8 data bits and 1 stop bit under most conditions –– the number of total bits is equal. You must, however, match setups to always have correct operation.

More Ram, New Options & Prices

Effective September 15, the standard (minimum) RAM for all Stride 400 Series systems is now **512K** bytes. Previously the minimum was 256K bytes.

The new systems will feature 256K bit DRAM chips soldered in either 512K, 1M or 2M byte units.

Bigger Winchester

The Stride 440's now have an option for a 67M byte Winchester. The popular 440 can now expand from the "floppy only" version announced this summer to this large storage configuration.

The 20M byte hard drive option has been dropped from the 460 line. The smallest Winchester offerred on the 460 is now 33M bytes.

Omninet An Option

The Omninet Local Area Network is now an option as a chip set kit. All systems have sockets for the network chips to make field installation easy. The new price for the LAN option is a very reasonable $595 (retail).

New Price

Prices on the new systems without the Omninet chips, and with 512K bytes of RAM (256K bytes **more** of RAM) will raise by only $95. This is a considerable savings when compared to the prior cost of a 512K system.□

Here are some useful tricks using the Stride (Wyse Wy–50) terminals

If you send the terminal incorrect code and locked it up, clear it by typing the SETUP key (no shift) three times. This is also a fast way to reset the function keys.

If you type ESC while in SETUP mode, all terminal parameters revert to the Wyse factory defaults.

Major Enhancements In The IV.21 p—System Release

The IV.21 version of the p—System has been released by Stride Micro for *both* Sage and 400 Series machines. See the earlier article on order information.

In the discussion below, changes that occurred between versions IV.13 and version IV.21 are also re—summarised.

Applications that work under IV.20 need no changes to work under IV.21 unless you desire to take advantage of some of the new features.

External Code Pools

The IV.21 release (not IV.20) provides for management of multiple external code pools, based on the amount of available memory. Code space and data space are still restricted to 64K bytes. However, multiple code and data segments can now be resident in memory.

The operating system automatically adjusts this area for varying amounts of machine memory provided the appropriate fields have been set in the SYSTEM.MISCINFO file to the maximum size allowed. Stride distributes several different terminal versions of SYSTEM.MISCINFO in the subvolume MISC: with the maximum size (16,383K bytes) set. Refer to the later article on *Memory Assignments* for details on how to define the extra code and data pool area.

The routines in the new unit DATASEGM are needed to access the external pool area.

Performance

The associate time (time necessary to start programs) for small memory machines (128K or less) is now much faster in IV.21. (Note: Stride does not officially support less than a 128K system, with typical memory needed per user being 154K.)

Changes have been made to provide a vastly improved boot time, faster textfile I/O, program invocation improvements and improvements to the PME for the 68000 processors.

STARTUP Program

STARTUP is a new facility which allows you to run a set of commands at startup. The possibilities include most of the commands of the filer and p—System command line. You can redefine the root volume (useful when loading a RAM disk) and also transfer the files over. Programs can be run one after another, subvolumes mounted, etc.

This is a very useful facility for personalizing your own system's setup.

S(et Command

S(et is a new command in IV.2x releases. It is called from the main (command) p—System menu line. It allows you to set the following items:

```
Date
Time
Workfile name
Library text file name
Compiler name
Assembler name
Editor name
Prefix volume
```

Note that you can still set the date and prefix by using the filer. The standard Stride DATE program is also available to set the clock.

The workfile name is now designated using S(et. The filer is no longer used for this purpose. Also, the SYSTEM.WRK form of the workfile is no longer valid.

The S(et command displays information about the current configuration. This information includes what real number package is installed, whether or not the print spooler is installed and the size of the code pool.

Editor

With versions IV.1x, the sequence for updating a file was: Q(uit, W(rite, $, <cr> and R(eturn. Then you could continue editing. With IV.2x, this has been simplified. You can now save your file by typing **W** for W(rite. The file is saved under its original name. You do not leave the filer and your position in the file is kept.

With IV.1 when Q(uit U(pdate was used to exit the editor, the file *SYSTEM.WRK.TEXT was created. With IV.21, workfiles are treated differently. Q(uit U(pdate writes your file to disk with the same name that it had originally. See the comments on the next page under *Workfiles* .

You can now use Q(uit W(rite to write your text file to the PRINTER: or any other serial volume.

Besides T(ok and L(it , there are now two additional modes of searching in F(ind and R(eplace. They are C(ase and I(d. C(ase enables the editor to search for a string but ignore whether the letters in that string are uppercase or lowercase. I(d allows the editor to search for a string as it appears to the Pascal compiler: the case of the letters is ignored, underscores are ignored and only the first 8 characters are significant.

A new line has been added to the S(et, E(nvironment display. It indicates whether or not changes have been made since the file was last saved on disk.

A warning is given if you attempt to exit the editor without saving your changes.

Filer

The filer no longer supports G(et, S(ave, W(hat, or N(ew. These four commands were involved in manipulating the workfile. The workfile is now handled differently. See the paragraphs under *Workfiles* .

The L(ist command now displays the files on a disk in two columns instead of one.

The E(xtended list command now displays the time that a file was last updated. This only happens, however, if the system time was set when the file was updated. The S(et command, the STARTUP program or the Stride DATE program may be used to set the system time.

Print Spooler

The print spooler now includes the ability to use wildcards in specifying the files to be printed and the ability to disable special character handling when printing data files.

Version Identifier

When a p—System program starts to execute, the version number of that program is displayed in the following format:

```
[nR.m.x]
```

In this format, the letters 'n', 'm' and 'x' denote numbers, while the uppercase 'R' stands for "Release" and is just used for separating the letters 'n' and 'm'. The letter 'n' is the major release number, 'm' is the interim release number and 'x' is the bug fix release number.

Compilers and Assemblers

For IV.2x releases, the compiler and assemblers create a code file with the same name as the text file when a simple <cr> is typed at the output file prompt. Previously, in versions IV.1x, the file SYSTEM.WRK.CODE was produced.

All compilers and assemblers now handle syntax errors in the same manner as the Pascal compiler. Each compiler or assembler has its own error file. In the case of the Pascal compiler, this is still SYSTEM.SYNTAX. When a syntax error is encountered, the compilation or assembly pauses. You are asked if you wish to continue or go to the editor. The editor automatically edits the file being processed *even if it is an include file*. If the errors file is present, the error message is displayed on top of the screen. Otherwise, the error number is displayed.

The Pascal compiler has a new compiler option: $L^. When using the $L+ and $L– toggles to control a compiled listing, the $L^ command causes the listing to be switched to the state prior to the previous $L+ or $L– command.

Debugger

Several new features have been added to the p–System debugger. (This is available on the program development disk.)

A command now displays the state of the stack.

Another command now allows you to look at disk memory as you could already examine main memory.

The environment list command has been improved.

Single stepping has been improved.

A command allows you to look at a code segment within the code pool.

Workfiles

Workfiles are handled differently from previous versions of the p–System. Now, the S(et command, not the filer, is used to designate the workfile. Also, SYSTEM.WRK.TEXT and SYSTEM.WRK.CODE workfiles do not exist in IV.2x. The following summarizes how workfiles are now used:

Use the S(et command to designate a workfile name. Both the .TEXT and the .CODE file that begin with that name are considered to be workfiles (if they exist).

Start the editor. The name of the workfile will be displayed. If you simply press <cr>, the workfile will be edited. However, you can backspace over the name and type in another name if you wish. (Or you can add characters to the name displayed.)

Leave the editor by using Q(uit, U(update. The output file has the same name as the original file had. (SYSTEM.WRK.TEXT is not created.) If a code file with corresponding name exists on the same disk, that code file will be removed. Note that simply using Q(uit U(pdate does not mean that the output file is the workfile; the S(et command must be used to define the workfile.

Start a compiler or assembler by pressing **C** or **A** . The name of the workfile is displayed as the file to be compiler or assembled. As with the editor, if you simply press <cr>, that file is processed. However, you can alter the name displayed and the new name will be the file compiled. The prompt for the code file is then displayed. If you simply type <cr>, the output code file is given the name that corresponds to the text file. You can enter a different code file name if you wish.

If you start the compiler by typing **R** for R(un, instead of **C** for C(omp, the workfile is always compiled and the output code file is always given the name that corresponds to the input text file. (The prompts for the file to be compiled and the output code file are not displayed.)

Pascal Language Enhancements

The *UCSD Pascal Handbook* describes the Pascal language under the p–System. There are some enhancements to the language which were made after the last publication of the book. Refer to the *Program Development Reference Manual* for details on Pascal enhancements.

Version IV.21 contains these new enhancements:

```
Conformant Arrays
Interface Conformant Arrays
Functional and Procedural Parameters
SIZEOF Intrinsic Enhancement
Exit Code
Improved Selective Uses
Improved Code Generation
```

A Conformant Array is a Pascal type which allows you to pass different sized arrays as the same parameter to a routine (during separate calls to that routine). An Interface Conformant Array is a similar type which allows you to pass variables of different types as well as sizes in the same parameter to a routine (during separate calls to that routine). Problems were found with the implementation of Conformant arrays under IV.20, which have been fixed in the new IV.21 release.

The Function and Procedural Parameters enhancement allows you to pass functions and procedures as parameters to routines. For example, you can pass one procedure as a parameter to another procedure.

The SIZEOF intrinsic allows you to determine the size of any variable or Pascal type. This intrinsic can now return the size of a structure in units other than bytes. It also can return the size of a particular field within a record.

Exit code may be specified for each procedure in the same manner that termination code is specified for UNITS. This feature is very useful for error recovery code.

The "selective uses" feature is now much easier to use. You need only to specify the identifiers that you explicitly use. In IV.1 you had to specify not only the identifiers that you explicitly used but also the names of sub–structures that you used.

The code generated by the Pascal compiler is slightly better than that generated by previous versions of the compiler. If you recompile an old program the new code file is usually smaller than the old code file.

Processes

Concurrent processes in Pascal use memory on the Heap for their own stack. These individual stacks are used by processes in the same way that the main p–System stack is used by non–concurrent code. (For example, variables are stored there). With IV.21 this Heap space is released when a process terminates. This was not the case with IV.1 . In addition, processes can be terminated by other processes (see the Error Handling Unit) and are automatically terminated by and EXIT(PROGRAM) statement.

Closing Files

Often it is desirable to open an existing file (using RESET) and also create a new file (using REWRITE)

(Continued)

both of which have the same name. During the time both files are open, the old file is a "permanent file" and the new file is a "temporary" file.

Previously, the order in which the two files were closed did not matter. The CLOSE (with a LOCK or a CRUNCH) on the new temporary file would change its status to a closed permanent file. At the same time, the old file would be removed.

With IV.21, you must first CLOSE the old file before closing the new file. This order is now important since the old file is "locked" during the time it is open and therefore cannot be removed by other operating system commands.

This difference between version IV.13 and IV.2x may require your application to be changed and recompiled.

Real Time CLOCKUNIT

A new unit called CLOCKUNIT supports setting and reading a real–time clock. The original STRIDE time and date unit, TAD_UNIT in the TOOLS.CODE library has not been changed. New programs written with portability in mind to/from Stride machines should use the CLOCKUNIT routine.

A user program called SETCLOCK was released by SofTech but is not distributed by Stride. The DATE program is still the user program for setting the system time and date.

System Units

Changes have been made to the interface sections of the following units. The changes are fully described in the *Program Development Reference Manual.*

```
KERNEL
SCREENOPS
ERRORHANDLER
WILD
DIRINFO
SYSINFO
FILEINFO
```

Any IV.13 programs using KERNEL should be recompiled.

Two new routines are included in SCREENOPS to handle reading long strings. This does not affect any programs which currently use SCREENOPS.

ERRORHANDLER has many new features. It now allows you to: handle execution errors; cancel processes; and translate I/O and execution error numbers into English text. These additions should not affect any programs which currently use ERRORHANDLER.

The file management units, WILD, DIRINFO, SYSINFO, and FILEINFO, contain additional routines. The WILD unit will also allow your program to use two new "wildcards" (*) and (+). These additions should not affect any programs which currently use these units.

TRANSFER is a new unit which allows your programs to move files in the same way that the filer's T(ranfer command does.

ATTRIBUTES is a new unit to set/query the various attributes of a file such as type, date etc.

DATASEGM allows a program to manage a data area (the extra CODE and DATE pool) outside of the stack/heap area. In addition, data can be passed between programs using this mechanism. □

Reading Foreign UNIX Floppies

It is sometimes necessary to read foreign floppies under UNIX systems. Some foreign UNIX systems start writing **cpio** information on the second track. The Stride **cpio** program expects data to start on the first track.

You can use the **dd** program to take care of differences such as this. For example:

```
dd if=/dev/rmt0/  skip=8 |cpio -i[options]
```

This will read a foreign **cpio** formatted diskette which was written with 512 byte sectors starting at the second track.□

UNIX cflow Status

The **cflow** *program which generates C program flow graphs is currently incompatible with the configuration of the C compiler and* **lint**. *The conflict has been resolved for the next UNIX release.*

400 Series BIOS Changes

A new BIOS, MU.BIOS, UTIL.CODE and MU.UTIL.CODE are available in the October release, either combined with the IV.21 release or separately with the BIOS update disk.

*****

None of these changes require any recompilation of your programs.

*****

Modifications to support new vendors of tape drives were made to the QIC02 code.

Various serial channel driver changes were made, mostly to provide better compatibility between 400 Series and Sage communications routines.

The Winchester driver and buffer handling were modified for better Liaison network operation and the Disk Cache feature.

An minor change was made to the Clock Circuit driver to deal with a trivial problem which only occured when setting the date on the last day of March, May, July, October or December.

An on–line configuration change to the attach numbers now works correctly.□

Manual Typos

Appendix A on pages 233 and 234 of the Liaison Operating System manual, lists the Execution Errors numbers incorrectly. Each error number is one too large.

On page 618–620 of Volume II of the Stride Owner's manual, the offsets numbering for the serial channel data format is incorrect. Each offset group shows one byte. The sequence for the groups should be 0–9.

p–System IV.21 Memory Assignments

```
   1      2          3               4                5             6          7
 ---------------------------------------------------------------------------------
|DATA  | CODE | EXTENDED      | LIAISON       | Modula-2      | INTER-   | SYSTEM     |
|AREA  | POOL | CODE and      | SOCKET POOL   | GAP           | PRETER   | STACK      |
|      |      | DATA POOLS    |               |               |          |            |
|64K   | 64K  | (optional)    | (optional)    | (optional)    | 13K std. | 256 bytes|
|std.  | std. |               | 10K Liaison   | 512K needed   |          |            |
|      |      |               | 32K DKserver  | to run the    |          |            |
|      |      |               |               | Scenic M2     |          |            |
|      |      |               |               | compiler.     |          |            |
 ---------------------------------------------------------------------------------
```

With the IV.21 p–System release, the memory needed per user can be assigned by defining the size of each of the areas shown in the diagram at right.

DATA AREA

The default data area is set to 64K bytes and cannot be changed by the user. Using the new DATASEGM unit, however, you can use part of the CODE POOL area as a data area. It is possible with this technique to pass information between programs.

CODE POOLS

The CODE POOL size in the SYSTEM.MISCINFO files determines this. The MISCINFOs released on the STRIDE update have already been changed to allow the maximum size of the total code pool area. This is 16,383K bytes (including the standard 64K code pool). The amount of memory left over after the SOCKET POOL and Modula–2 GAP are assigned is given to the CODE POOL. Normally, you will not have to change anything on the standard release to get extra code pool room.

SOCKET POOL

The SOCKET POOL is only needed for Liaison network operation. The "HAS EXTERNAL SOCKET POOL" field in SYSTEM.MISCINFO must be set TRUE. The field "SIZE OF EXTERNAL SOCKET POOL" must be set to 10 (K bytes) for a network user. If a diskserver will be installed as that user, the size must be increased to 32 (K bytes). The easiest way to make these changes may be to run the program TERMINAL.CODE.

Modula–2 Gap

The "Modula–2 GAP" is not normally accessed by the p–System. Currently, this area is only used by the Scenic Modula–2 system. WORD 3 of the USER OPERATING SYSTEM INFORMATION field in the system configuration must be set to the total operating system size (ALL AREAS) in Hex to K bytes. This word is set using UTIL for single user operation and MU.UTIL for multiuser operation.

Interpreter & Stack

The space assigned to the Interpreter and Stack areas can not be changed by the user.

Multiuser Considerations

Each user in the multiuser system is setup just as a single user is. The multiuser BIOS is put at the very top of memory above all of the users and above any RAM disks.

On the Stride 400 Series release disks, the MU.BUILD program now has a simplified way to define all of the memory areas for a multiuser installation.□

Check Your Floppy Setups

For proper operation on the Mitsubishi floppy drives, the MOTOR ON DELAY FACTOR should be set to 9600H with UTIL.CODE. The Mitsubishi drive has a round LED light and a push/pop latch.

Regardless of vendor, this value is recommended for **all** *Sage and 400 Series drives.*

New MU.BUILD Simplifies Multiuser Installation

Installation of a Stride multiuser system is easier with the aid of a new program called MU.BUILD. An early version of this program was offered for the 400 Series but not for the Sage. The IV.21 p–System release includes MU.BUILD for the 400 Series and MU.SGBUILD for the Sage on the respective BUILD diskettes.

The BUILD program has been updated to allow most of the functions of MU.UTIL and WFORMAT but in a friendly manner.

The **hard disk** assignment helps you calculate in blocks, the size and number of partitions you can have for your disk. You now have full control of the name and type of partitions. (WFORMAT requires that you create a WINFO file with the editor and do your own calculations.) BUILD will create the WINFO file and load it for you.

The **memory** option helps you calculate, in K bytes, the memory size for each user. (MU.UTIL requires Hex arithmetic). If more than 512K bytes of memory is available beyond the users area, the new Disk Cache option will be enabled.

The **serial ports** option lets you assign the ports to the users as they are numbered on the back panel.

You can now easily install such features as Foreground and Background users, a second printer and the Liaison network.

MU.BUILD will load the correct boot and system files for you on the p–System partitions. You can also assign partitions to other operating systems, but their files will not be loaded.

When it is done, a bootable multiuser system has been installed. Fine tuning can take place at this point using MU.UTIL to modify the MU.CONFIG file that was built by MU.BUILD.

Step–by–step documentation for using MU.BUILD is also included in the new release.□

9–Track Tapes A "HIT" For Stride

By Judy Olson and Mark Borgerson

Virtually all mainframes and minis can read and dump data and text files on half–inch, 9–track reel–to–reel tape. The Stride microcomputer now has that capability with a software and hardware package developed by a group in Corvallis, Oregon. The Stride HITS (Half–Inch Tape Subsystem) provides an easy means for transporting massive volumes of data between the Stride and any mainframe or minicomputer, as well as access to large archives of data already stored on magnetic tape.

The HIT Team

Jackie Gordon of Viz.Ability, Corvallis, Oregon, is a researcher in atmospheric optics. For her contract with the Army Research Office she needed a tape drive to access a large volume of data that only existed on half–inch, reel–to–reel magnetic tape. Viz.Ability already owned a Sage II and had bought a Stride 460 for its large Winchester disk and because the VMEbus allowed the addition of nonproprietary sub–systems. Jackie planned on using her Stride 460 to process her data, but first needed a means of transferring it from the archive tapes onto the Winchester disk.

Lou Gordon (another partner of Viz.Ability and associate professor at Oregon State University) attempted to locate a tape subsystem for the Stride. After a number of phone calls and attending the Stride Faire in February, 1985, he was able to locate a number of people who were also interested in a tape drive system, but no one who could provide it.

Lou then talked to Mark Borgerson (of 8–Bit Systems, Corvallis, Oregon), a friend and hardware/software consultant. Mark had built his first computer in 1977 and has written 68000 assembly language programs for a number of computer systems. He felt he could handle any of the problems involved with developing a tape drive system; the cost, however, could not be justified for a single person working on a single project. But if a package were put together for resale . . . .

With the addition of another friend, Stride dealer Jeff Stander (of Maritime Infosystems, Ltd.), the Stride HITS team was formed. Development work would be done under Viz.Ability with Mark handling the hardware interface and programming. Marketing, distribution and support would be through Maritime Infosystems, Ltd.

The HITS Product

The Stride HITS hardware consists of a Stride 460 or 440 (the system is not recommended for the 420), a Kennedy tape drive, model 6809 or 9600, and the HITS kit (three Motorola boards, two of which are connected as a single piggyback module, and three connecting cables). Four compatible drives can be daisy–chained in the system.

The software consists of three main components, a control unit, a utility program and a hard disk backup program. They presently run only in the p–System environment.

The **Tape Control Unit** is a package of Assembly routines which activate the tape drive. It can be linked with Pascal or FORTRAN code under the p–System. Source code for the Tape Utility Program will be included as an example program using the Tape Control Unit.

The **Tape Utility Program** is menu driven. It has file–to–tape and tape–to–file capability for transferring tape files to hard disk. It also can be used to investigate a tape of unknown format by printing the data on the screen in either ASCII or HEX. It has block–by–block and file–by–file spacing capabilities, forward and backward.

The **Tape Backup Program** (also menu driven) allows the user to read and write in a tape–to–disk or disk–to–tape manner. Whereas the Tape Utility Program copies single files, the Tape Backup Program copies whole volumes. It also has the catalog ability necessary for archiving purposes. Of the three software pieces, the Tape Backup program has the most built in safe–guards.

Besides the Motorola boards and cables, Tape Control Unit, Tape Backup program, Tape Utility program and source code, a Stride HITS buyer will also receive a manual and automatic notification of updates. For the first year after purchase, a buyer will receive free any software update applying to the purchaser's original operating system, upon request. After the first year there will be an item–by–item charge. Additional HITS custom programming is available by special arrangement with the HITS group.

Further plans include integrating the Stride HITS into the single user BIOS to allow access to the tape drive as a Stride device. The current HITS version runs only under the p–System operating system whereas the BIOS is common to all operating systems. Adaptations to additional Stride operating systems are also anticipated.

TECHNICALITIES

The just–released Kennedy 9600 tape drive is a perfect match for the Stride. It is compact and modern, with both streamer and stop/start modes at 100 and 45 IPS respectively. It can use either 1600 or 800 BPI tapes and has full diagnostic features. Its front load feature handles all reel sizes up to 10 1/2" and it is reasonably priced. Three or four 9600s can be installed in the vertical space occupied by an older vacuum column drive. The Kennedy model 6809 provides a less expensive and slower option, at 1600 BPI only.

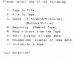
The tape controller hardware consists of two subsystems, the VMEbus I/O adapter and the MT (MagneticTape) adapter. The I/O adapter plugs into the VMEbus on the Stride and decodes the VMEbus short address mode signals. It also handles the assignment of vectored interrupts for the processor. The I/O adapter decodes a segment of the VMEbus short address space. It then passes on to the MT adapter address bits, data bits and timing signals. These signals are exchanged over a ribbon cable which plugs into a connector on each unit. The MT adapter also receives its +5V power from the I/O adapter via this cable. There is a manufacturer's warranty on both boards.

The MT adapter (two 1/2 VME size, piggyback cards) further decodes the address space selected by the I/O adapter to enable the selection of a block of contiguous registers at consecutive odd addresses. These registers are used to control operation of the tape deck and also to read data and status from the tape deck. Control, status and data signals are exchanged between the MT adapter and the tape deck over two ribbon cables.

Mark Borgerson encountered some problems in integrating the magnetic tape system into the Stride 460 microcomputer. These required some hardware modifications of the I/O adapter to interface it with the Stride. Much of the development effort was spent in solving problems involving floating address bits and VME timing sequences.

After these problems were solved, Mark wrote assembly–language procedures to communicate with the tape deck. These are linkable with p–System Pascal or FORTRAN programs and can be used to read and write data on 1/2–inch magnetic tapes in a number of different formats. The proper decoding of the blocks read from the tapes is a function of the host Pascal or FORTRAN p–System program. The prototype assembly language procedures did not modify the interrupt structure of the Stride BIOS and used only polled I/O to communicate with the tape deck. Even with these constraints, data could be written to tape at more than 1M byte per minute using the Kennedy model 9600 tape deck running in start–stop mode at 45 inches per second. With further optimization of the tape read and write routines, data transfer rates can be doubled or tripled for disk backup purposes, particularly with faster operating systems.

Software Operation



```
         Tape Utility Program  V1.0

Please select one of the following:

   T: Tape to File.
   F: File To Tape.
   S: Space  (F-forward/R-reverse)
             (B-block/F-file).
   B: Beginning  (Rewind Tape).
   R: Read a block from the Tape.
   A: ASCII display of tape data.
   H: Hexadecimal display of tape data.
   I: Initialize a tape.

Your Selection?
```

The Tape Utility program allows the user to determine the characteristics of newly–arrived tapes, display the data on the tape in either hexadecimal byte or ASCII character format and to move single disk files to and from the tape.

This last capability has been of particular value to Viz.Ability which uses tapes of atmospheric data with files more than 7M bytes long.

The model 9600 Kennedy tape drive used with the HITS program offers 45 IPS start/stop and 100 IPS streaming performance along with automatic tape loading.

A tape file is read into a disk file in a large partition of the hard disk on the Stride 460. Pascal programs can then access the data using standard data file manipulation techniques.

The Tape Utility is a menu–driven program which allows the user to use the HIT system with only a few keystrokes at the terminal. The menu for the preliminary version of the program is shown to the left. This menu is subject to change as the program is updated.

The **Tape to File** menu selection prompts the user for the name of the disk file to which the tape should be written. If a file by that name already exists, the user is asked if the file should be overwritten. The user is also asked if the tape file has a descriptive header block. If the response is positive, this block will be read and displayed on the screen, but not written to the disk file.

File to Tapewill transfer a disk file selected by the user to the magnetic tape. The file is written at the End of Data (EOD) position on the tape. The EOD position consists of two consecutive End of File (EOF) blocks. The EOD mark is then written after the new file. The data from the disk file is preceded by a short header block which names the disk file and specifies the time and date of the transfer to tape. This is the block that is normally ignored when the file is transferred back to disk. There must be a Write Ring in place on the tape or the program will notify you that the tape is write–protected.

The **File to Tape** and **Tape to File** commands are intended to allow the user to manipulate files which are too large to be conveniently stored on floppy disk. The requirement that new files be appended to the data already on the tape makes these commands only marginally useful for short data and program files.

The **Space** command actually requires the user to enter two more characters before the tape deck is activated. The second character (F or R) specifies the direction to move the tape. The third character (F or B) specifies whether the tape should move to an EOF mark or just move a single tape block. For example, the characters **SFF** would cause the tape to move forward to the next file mark.

The **Beginning** command is the computer controlled equivalent of the Rewind button. The tape deck remains On Line and ready to read the first block on the tape.

The **Read** command reads the next block on the tape into an internal buffer.

(Continued)

The **ASCII display** command displays the data in the tape buffer in ASCII character format. The data is displayed 20 lines at a time. A line is defined as either 80 consecutive characters or a series of characters followed by a Carriage Return character. After 20 lines are displayed, the user can either hit the Escape key to return to the main menu, or any other key to display the next 20 lines. Control characters are not displayed.

The **Hexadecimal display** command displays the data in the tape buffer as a 4 byte hexadecimal address followed by 16 bytes in hexadecimal format, then 16 ASCII characters (control characters are replaced by periods).

Initializing a tape simply means writing two consecutive EOF marks at the beginning of the tape. Thus, any future File to Tape commands will start writing at the beginning of the tape, overwriting any previous data on the tape. Since this command can potentially destroy data on the tape, the user must verify the command to initialize the tape. A Write Ring must also be in place on the tape.

Product Options

The following product information shows the options available on the HITS package.

All options include the following software: Tape Control Unit, Tape Utility Program and Tape Backup Program.

Stride HITS

$4,950	I/O Adapter Board Magnetic Tape Adapter board All necessary cables
$7,650	With Kennedy model 6809 1600 BPI PE only 100 IPS Streaming 12.5 IPS True Start/Stop
$9,100/ $8,950	With Kennedy model 9600 800 BPI NRZI or 1600 BPI PE 100 IPS Streaming 45 IPS True Start/Stop

The package with the Kennedy model 9600 is available at the introductory price of only $8,950 until December 31, 1985.

HITS is distributed by Maritime Infosystems, Ltd., 6660 Reservoir Road, Corvallis, OR. Contact J. Stander (503) 929-2552 □

400 Series DISK CACHE Improves Winchester Hard Disk Throughput

The Disk Cache is a new feature of the Stride 400 Series BIOS and MU.BIOS available with the p-System IV.21 release or the BIOS update disk.

The Disk Cache is an area of memory (usually 512K or more) set aside by the user for use by the hard disk driver. Information to and from the disk is "cached" in this memory area. Obviously, the cache cannot hold the entire contents of the disk. Tracks are stored as they are read and written with the "least recently used" tracks being replaced by the new tracks.

READ Operation

When a track is read from the disk, it is also cached. If the track requested is already in the cache, it is read from the cache, not from the disk.

Therefore, the first read to a track is at regular disk speeds. However, the second read of that same track will take place at memory access speed, not the slower hard disk access speed.

Applications which frequently access the same information will find that the Disk Cache will significantly improve throughput as it is much faster to read memory than disk!

WRITE Operation

When a track is written to the disk, it is also written to the cache. As cache memory can be lost due to a power failure, machine reset, etc., a write operation is always done to the disk, insuring the integrity of the files. This means that a write operation is actually a little SLOWER than non-cache operation because the extra step of writing the cache takes a little more time.

RAM DISK vs DISK CACHE

Disk Cache is very similar in operation to RAM disk. In fact, you install it in almost exactly the same way. However, it provides several important advantages over RAM disk:

It is safer. Disk Cache operation insures that data is saved on the hard disk. RAM disk data will vanish if power is lost or the system is rebooted.

Boot time is faster. Files and data do not hatve to be loaded to the RAM disk. This eliminates having to plan which files to put into RAM disk.

More data can be accessed. RAM disk only works for the files and data stored on it. Disk Cache works for all data on all the hard disks.

Disk Cache Installation

The Disk Cache is installed using a new command in UTIL UTIL.CODE and MU.UTIL.CODE under the RAM disk option. Stride recommends that you back up your old system files before installing any of the new files on your system.

The new version of the multiuser installation program MU.BUILD will automatically assign any leftover memory to the Disk Cache if the amount leftover is greater than or equal to 512K bytes.

The same Disk Cache will handle multiple drives and all partitions that are made accessible to the operating system by the SYSTEM.BIOS or MU.BIOS configuration.□

What is <ETX> ?

<ETX> is a key command used to "close" an operation in the p-System, usually in the Editor. On the 400 Series and the Sage, Stride uses a CTRL/C. (Hold the Ctrl key down while typing "C").

To redefine the <ETX> command in the SYSTEM.MISCINFO file, use the SETUP.CODE program.

Square Root Speed

by G. A. Gallup

For a number of years my students and I have been developing and running a package of computer programs for performing molecular electronic structure calculations. We have tested and run these on a number of different computer systems. Through these comparisons, we discovered that the running time of one particular program in our set varied considerably more from machine to machine than the basic machine speeds would warrant. Further investigation showed that a computer in which the square root function was implemented in hardware gave the fastest time for our program.

This particular program segment must calculate the distances between pairs of several thousand points in three dimensional space. This requires the three dimensional Pythagorean theorem and all those square roots –– in double precision, too.

I have now converted these programs to a Stride 440. Previous experience led me to examine how rapidly square roots were handled in the program where this is crucial. Timing tests showed that the SVS FORTRAN library routine for DSQRT takes, on the average, 1.8 milliseconds to execute.

I felt this was rather sluggish and thought that it might be possible to get more speed with a different type of function subroutine. While examining the possibilities, I recalled the old–fashioned "long division" method for calculating square roots that we were all taught in grade school and considered that an assembly routine using binary arithmetic could implement this method entirely within the registers of the 68000.

The eight long word data registers available on the 68000 are indeed sufficient for the task. The assembly listings show single and double precision square root routines that in the worst cases take only 40% of the time of the SVS library routines. The single precision version is normally better than that.

(Continued)

```
* Listing 1. Subroutine for square roots by the long division
*           method in the registers of the MC68000.

        globl  SQRT68          * MUST Capitalize for FORTRAN compiler

* FORTRAN usage gives REAL*4 square root of REAL*4 number A:
*           X=SQRT68(A)
*************************************************************
SQRT68: movea.l 4(sp),a0       * address of argument
        move.l  (sp)+,(sp)     * Clean up stack
        movem.l d2-d7,-(sp)    * save d2-d7 on stack
        move.l  (a0),d0        * get argument, test for zero
        beq     wrpup          * if zero, nothing to be done
        bpl     do             * test for neg,if pos, go ahead
        moveq   #9,d0
        lea     msg,a0
        move.l  a0,d1          * send error message
        trap    #2
        moveq   #0,d0
        bra     wrpup
do:     move.l  d0,d2          * must do square root
        moveq   #9,d7
        rol.l   d7,d2
        andi.l  #255,d2
        andi.l  #8388607,d0
        addi.l  #8388608,d0
        moveq   #0,d1
        addi.l  #8388608,d1
        addi.l  #127,d2
        lsr.l   #1,d2
        bcc     do1
        lsl.l   #1,d0
do1:    move.l  d1,d3
        moveq   #23,d6
        sub.l   d1,d0
        beq     asmbl
        lsl.l   #1,d0
        lsr.l   #1,d3
loop:   move.l  d1,d4
        lsl.l   #1,d4
        add.l   d3,d4
        lsl.l   #1,d4
        lsl.l   #1,d0
        add.l   d3,d1
        sub.l   d4,d0
        beq     asmbl
        bpl     ok
        add.l   d4,d0
        ssb.l   d3,d1
ok:     lsr.l   #1,d3
        subq.w  #1,d6
        bne     loop
asmbl:  moveq   #9,d7
        lsl.l   d7,d1
        or.l    d2,d1
        ror.l   d7,d1
        exg     d1,d0
wrpup:  movem.l (sp)+,d2-d7           * restore registers
        rts
        .data
msg:    dc.b    'Negative argument '
        dc.b    'in SQRT68 ',10,13,'$ '
        .end
```

I have named the single precision routine SQRT68 and the double precision DSQR68 to eliminate interaction with the library routines. They are used in a FORTRAN program in the standard way:

```
X = DSQR68(A)   or   X = SQRT68(A).
```

The IEEE standard floating point format (see Appendix C of the SVS FORTRAN Reference manual) is used in these, of course, since they must match the other SVS FORTRAN usage.

To use these routines they must be assembled and then linked into a FORTRAN program. If SQRT68 is in the file SQRT68.S then the command:

```
A>AS68 -P -L SQRT68.S >SQRT68.LST
```

will produce the file SQRT68.O and a listing in SQRT68.LST. A similar command will assemble DSQR68.S and produce the files DSQR68.O and DSQR68.LST.

The submit file FPF77.SUB on the FORTRAN distribution disk contains a sample linkage step for SVS FORTRAN. The simplest way to use the square root routines is to change the appropriate line of this submit file to

```
A>LO68 -T500 -O $1.68K S.O $1.O
 SQRT68.O DSQR68.O SVSFPLIB OPTOFF CLIB.
```

Of course, probably only one of these is needed for any given program.

This is the version for hardware having an FPU. If the system does not have an FPU, the SVSFPLIB file is not appropriate.

A library with these routines can be made with the archive program AR68 or they can be added to CLIB. *CAUTION:* This last possibility will produce a very nonstandard CLIB.

I cannot claim that these routines are as fully optimized as they might be, however, they represent a significant saving in time for the square root function. After division, square rooting is the next most common numerical procedure in typical calculations.

I would like to thank my colleague, Henry Baumgarten, for useful discussions on this problem.□

Ed.Note: Professor Gallup has kindly donated these routines to the public domain. Other Stride implementors who wish to utilize them in their own programs are welcome to do so. Gordon Gallup may be reached at Department of Chemistry, University of Nebraska, Lincoln, Nebraska 68588

```
* Listing 2. Double precision square roots routine using the long
*            division method in the M68000 registers (July 14,'85)

         globl  DSQR68           * Must be capitalized to be linked

* Fortran usage:  Produces REAL*8 square root of REAL*8 number, A.
*            The numbers are in standard IEEE floating point format.
*
*       IMPLICIT REAL*8(A-H,O-Z)
*        . . . .
*       X=DSQR68(A)
*        . . . .
*************************************************************
DSQR68: movea.l 4(sp),a0         * Retrieve passed parameter
        move.l  (sp)+,(sp)       * Justify stack
        movem.l d3-d7,-(sp)      * Preserve environment
        move.l  4(a0),d0         * Obtain LSLW of argument
        move.l  (a0),d1          * Obtain MSLW of argument
        bmi     err1             * If neg. send message and exit
        bne     do               * If not zero go to process
        tst.l   d0               * Test for standard zero
        bne     err2             * If not,display message and exit
        bra     wrpup            * If real zero, process is done
do:     move.l  #$7ff00000,d7    * Mask for exponent
        and.l   d1,d7            * Obtain exponent
        sub.l   d7,d1            * Strip off exponent
        bset    #20,d1           * Replace blind one
        addi.l  #$3ff00000,d7    * Double exponent displacement
        lsr.l   #1,d7            * Divide by two
        bclr    #19,d7           * Determine if exponent was odd
        beq     do1              * If odd, double mantissa by
        lsl.l   d0               *  shifting left
        roxl.l  d1
do1:    lsl.l   d0               * Shift left again
        roxl.l  d1
        clr.l   d2               * Clear answer mantissa
        clr.l   d3               *   area
        move.l  #$200000,d3      * Initial mantissa times two
        sub.l   #$200000,d1      * 1st approx. to sq. rt. is one
        bne     do2              * If zero, branch to assembly area
        tst.l   d0
        beq     asmbl
do2:    move.l  #19,d5           * Set loop counter for MSLW
loop11: bset    d5,d3            * Set trial bit in answer mantissa
        sub.l   d3,d1            *  and subtract from argument
        bpl     ok1              * Check for sign

* If neg then
        add.l   d3,d1            * Restore original argument
        bclr    d5,d3            * Clear trial bit
        bra     loop12           * Branch to shift
* else
ok1:    bclr    d5,d3            * Cancel trial bit and put
        addq.b  #1,d5            *  in correct place one
        bset    d5,d3            *  position to the
        subq.b  #1,d5            *  left.
* endif

loop12: lsl.l   #1,d0            * Shift current mantissa one to
        roxl.l  #1,d1            *  the left
        dbra    d5,loop11        * Loop on d5
```

(Continued)

```
* Listing 2 - continued  The following section bridges over the
*               LW break between two portions of word
*
        move.l  #$80000000,d2       * Set trial bit
        sub.l   d2,d0               *  and subtract
        subx.l  d3,d1
        bpl     ok2                 * Test for sign.
* if neg then
        add.l   d2,d0               * Restore values of trial mantissa
        addx.l  d3,d1
        moveq   #0,d2               * Clear bit
        bra     bshft               * branch to shift
* else
ok2:    moveq   #0,d2               * Clear trial bit and
        bset    #0,d3               *  put in correct position
* endif

bshft:  lsl.l   #1,d0               * Shift trial mantissa once
        roxl.l  #1,d1               *  to the left
* This section does second LW of answer
        move.l  #30,d5              * Set counter for LSLW loop
loop21: bset    d5,d2               * Set trial bit
        sub.l   d2,d0               *  and subtract
        subx.l  d3,d1
        bpl     ok3                 * Test for negative result
* if neg then
        add.l   d2,d0               * Restore current mantissa
        addx.l  d3,d1
        bclr    d5,d2               * Clear trial bit
        bra     loop22              *  and branch to shift
* else
ok3:    bclr    d5,d2               * Cancel trial bit and
        addq.b  #1,d5               *  put it in correct
        bset    d5,d2               *  position one place
        subq.b  #1,d5               *  to the left
* endif

loop22: lsl.l   #1,d0               * Shift current mantissa left
        roxl.l  #1,d1
        dbra    d5,loop21           * Loop on d5
asmbl:  lsr.l   #1,d3               * Shift answer mantissa one
        roxr.l  #1,d2               *  to the right
        bclr    #20,d3              * Cancel blind one in mantissa
        move.l  d7,d0               * Assemble mantissa and exponent
        add.l   d3,d0               *  in (d0,d1)
        move.l  d2,d1
exrup:  movem.l (sp)+,d3-d7         * Restore environment
        rts
err1:   move.l  #emsg1,d1
        bra     errout
err2:   move.l  #emsg2,d1
errout: move.l  #9,d0
        trap    #2
        clr.l   d0
        clr.l   d1
        bra     exrup
        .data
emsg1:  dc.b    'Negative argument in DSQR68',13,10,'$'
emsg2:  dc.b    'Non normalized number in DSQR68',13,10,'$'
        .end
```

People & Products

Eric Smith of **Scenic** announces a new office location for his firm: Scenic Computer Systems, 12314 Scenic Dr., Edmonds, WA 98020. (206) 742-6677. Eric is currently beta testing his **native code Modula-2 Compiler** and is also a source of the popular ASE editor.

Mike Caro of **Poker** fame reports on **TurtleGraphics** and the **FPU:** "TurtleGraphics runs about 30% faster with the FPU, depending on procedures used. This is because so many real parameters are present."

October 1, **Ed Hayden** of **BOS** plans to announce new USA versions of all the BOS business products. A new **Sales Order Processing** package will also be announced. Ed says his "packed pricing" makes it very attractive to buy a set of BOS applications.

Dennis Graham of Micro Research and Development says that **Word7** has now been translated into French, German, Norwegien and Israeli! (303) 594-6727.

We hope **Jeff Stander** and friends have finally recovered from the champagne party they threw to celebrate finishing the HITS tape program. (See earlier article.)

According to *Sahara Soft*, the newsletter for **Mirage** users, Mirage now supports **CP/M-68K** as an application, allowing multiple (but still separate) users. Sahara Software Ltd., telephone (01 627 1733).

Yep, it's true. The p-System is changing hands. SofTech assets are in the process of being transferred to the new owners now. An official announcement will occur as soon as the lawyers tidy up. Right, **E.W.?**

After the latest MODUS conference in Menlo Park, maybe the Modula-2 compiler drought is over. About 70 folks showed up, with every third (maybe every second) attendee a compiler implementor!

Dick Karpinski, MODUS newsletter editor, moans the lack of **Modula-2** articles. It seems the Modula-2 folks would rather write code than copy for his magazine. Give Dick a break and send some over E-mail. Telemail: RKarpinski, Compuserve: 70215,1277, Bitnet: dick@ucsfcca uucp:ucbvax!ucfcgl!cca.ucsf!dick.

BN

Editor: Verlene Joyce Bonham

In Stride Tech Notes is a publication of Stride Micro, issued eight times yearly. Subscriptions are $12 for one year and include the *In Stride* parent magazine which is published quarterly, for a total of 12 issues per year.

Tech Notes back issues are also available for $0.50 and *In Stride* back issues for $2.00 as supply lasts.

Purchase of a Stride computer includes a one–year subscription upon receipt by Stride Micro, Reno, of a fully completed owner's registration card.

Requests for subscriptions, reprint permission, ad rates, bulk orders or submission of prospective articles should be sent to the *In Stride* Editor at Stride Micro's Reno address.

Postmaster: Change of address notices should be sent to:

***In Stride* Editor**
Stride Micro
P.O. Box 30016
Reno, NV 90520 –0016
(702) 322–6868 (8a.m.–5 p.m. PST)
TWX: 910–395–6073

Stride Micro Eastern Division
112–116 Washington Street
Marblehead, MA 01945
(617) 639–0750

Stride Micro Southern Division
13760 Noel Road
Suite 300
Dallas, TX 75240
(214) 392–7070

Stride™, *"Performance By Design"*™ *and NOD*™ *are trademarks of Stride Micro.*

STRIDE
MICRO