INTERACTIVE COMPUTER SYSTEMS, INC.

6403 DIMARCO ROAD • TAMPA, FLORIDA 33614
(813) 884-5270

OmniFORTH NEWSLETTER

Interactive Computer Systems, Inc. provides this newsletter for OmniFORTH users. It is our way of giving you more information about OmniFORTH and announcing new products. We also wish to stimulate interest in FORTH and solicit letters, comments, and suggestions to improve OmniFORTH.

New Products Now Available: . . . . . .

a) Cross Reference. . . . . Price $25
Prints a sorted cross reference of your FORTH application. Identifies each word that you create and where it is used. External words that are part of FORTH are also shown. Excellent for tracing word usage, documentation, and permits rapid redefinition of a base word and all words that used it. Distributed as source listings and system documentation. (OmniFORTH users get source disk.)

b) Video Editor. . . . . Price $25
Lets users with Video Display Terminals edit and input data into OmniFORTH screens. The Video Editor is an extension to the EDITOR that provides cursor control functions over the entire screen. The new Video Editor is supplied with SOROC cursor routines that may be modified for your terminal. Comes with source listings and user documentation. (OmniFORTH users get source on disk.)

c) OmniFORTH 2.3 Source. . . . . Price $495
Includes 8080 Assembler source listings and disk. Requires A.M. Ashley's PDS (Program Development System) for the North Star Minidisk. Customer must sign Software License Agreement prior to shipment.

New Software to be Released:. . . . . .

a) OmniFORTH supporting FORTH-79 standards.
(Available after March 1981 for $95.
Current OmniFORTH users get update for $30)

b) OmniFORTH for the Radio Shack TRS-80.
(March 1981 with FORTH-79 standards for $95)

c) Floating Point Arithmetic Routines.

d) OmniFORTH Compiler for the 8080 / Z80.

e) Word Processor.

f) Data Base Management System.

Disk Access Using RWDSK

OmniFORTH 2.3 was released without examples on how to access the North Star disk. This section deals with RWDSK (an undocumented OmniFORTH word) that lets you access disk. RWDSK is a system dependent word used by OmniFORTH to interface with the North Star disk routine DCOM. The definition of RWDSK is:

```
addr  sec  #sec  r/w  drns&drv  --  err
```

RWDSK uses the same parameters required by DCOM. addr is the start of RAM address. sec is the starting disk number (0-349 for single & double and 0-699 for quad). #sec is the number of sectors to transfer. r/w is the command (0 = write, 1 = read, 2 = verify). drns&drv is the density and drive byte where density is the most significant bit (0 = single 1 = double/quad) and drive is 1 thru 4. drns&drv can be 1 2 3 4 for single density and HEX 81 82 83 84 for double/quad density. The execution of RWDSK leaves the DCOM error on the stack so you can check if arguments were illegal.

. . . . . .

SCR # 60 CD Copy Disk Example using RWDSK

This example will show you how to use the OmniFORTH word RWDSK to copy disks. It can be used to backup OmniFORTH disks without returning to DOS.

a) Define the TOP of memory above LIMIT that can be used for a data buffer. Our OmniFORTH system has LIMIT at 8000 HEX and we had memory up to D000 that could be used for the buffer.

```
HEX  D000  CONSTANT  TOP
```

b) Define a constant for the last sector to be moved.

```
DECIMAL  350  CONSTANT  LAST
```

Note that single and double density disks will use 350 for LAST while quad density will use 700.

c) Now allocate variables.

```
0 VARIABLE  DR      ( drive to be read from )
0 VARIABLE  DW      ( drive to be written to )
0 VARIABLE  #SEC    ( # of sectors to transfer)
```

d) Create a word that increments the sector variable by the number of sectors to be transfered and determines if the last sector has been reached.

```
: NEXT-SEC  #SEC @  SEC +!     ( increment SEC )
   LAST  SEC @  #SEC @  +  ( look ahead LAST )
   - DUP 0< IF #SEC +!   ( partial #SEC xfer )
           ELSE DROP THEN  ( full #SEC xfer )
   #SEC @  0=  ;   ( true if #SEC = 0 on LAST)
```

e) Now create words to READ and WRITE disk using the OmniFORTH RWDSK interface to North Star DCOM.

```
: WRITE  LIMIT  SEC @  #SEC @
   0  DW @  RWDSK  DISK-ERROR ! ;

: READ   LIMIT  SEC @  #SEC @
   1  DR @  RWDSK  DISK-ERROR ! ;
```

f) Initialize SEC to zero and compute the number of sectors to be transferred #SEC = (TOP - LIMIT)/ (256(DENSITY +1)). Note that #SEC is computed to use memory from LIMIT to one less than TOP.

```
: INIT-SEC  0 SEC !                ( zero SEC )
   TOP  LIMIT  -                ( TOP - LIMIT )
   256 DENSITY @ 1+ *        ( 256(DENSITY +1))
   /  #SEC !  ;              ( set #SEC to fit )
```

g) Define a word to copy until #SEC = 0.

```
: COPY  INIT-SEC
   BEGIN  READ  WRITE  NEXT-SEC  UNTIL
   ." DONE " ;
```

h) Create a word to compute density and drive.

```
: D&D  DUP  128  DENSITY @  *  +  ;
```

i) Finally, define CD to copy disks using the value on top of the stack as the destination or to-drive and the value below as the source or from-drive.

```
: CD  SWAP  D&D  DR !  ." COPY DISK FROM " .
            D&D  DW !  ." TO "  .  CR
      ." RETURN TO CONTINUE "
      KEY 13 = IF COPY ELSE ." OMIT " THEN ;
```

Note that CD will print the drive numbers and ask that you press RETURN to continue. If any other key is pressed, the copy will not execute and the message "OMIT" will print.

. . . . . .

OmniFORTH Disk Error Recovery

OmniFORTH will return to North Star DOS whenever a hard disk error occurs. The following example will give you a routine that returns control to OmniFORTH.

. . . . . .

SCR # 61 North Star DOS Hard Disk Error Patch

Review your North Star DOS System Software Manual section G for the DOS entry point to HDERR, the hard disk error routine. Our release had HDERR located at 2019 HEX loaded with a C3 (JMP) followed by a two byte address (locations 201A and 201B). We will be creating a FORTH HDERR routine to ABORT the disk request and return control to you.

a) The FORTH HDERR routine prints the error message before executing ABORT.

```
: HDERR  CR  ." HARD DISK ERROR"  ABORT  ;
```

b) Set up assembly code to enter HDERR.

```
ASSEMBLER  HERE
```

Where ASSEMBLER establishes the vocabulary, and HERE pushes the address of the next available dictionary location onto the stack. Note that it is not necessary to use the word CODE to create a word header or dictionary entry for the routine.

c) Create code that will Jump to OmniFORTH HDERR.

```
' HDERR  I LXI  NEXT JMP
```

A FORTH word can be entered from assembly code by placing the parameter field address in the interpreter pointer and executing NEXT. The 8080/280 OmniFORTH interpreter pointer is the BC register pair and is referred to as I and I' in the ASSEMBLER.

d) Poke HERE into memory location 201A.

```
201A  !
```

This completes the hard disk error patch. Any attempt to access disk that causes a Jump to DOS HDERR will now execute the OmniFORTH HDERR.

. . . . . .

Using Interrupts in OmniFORTH

OmniFORTH 2.3 did not include documentation on how to use interrupts. The remainder of this newsletter illustrates a few examples that will guide you in coding interrupt tasks in both ASSEMBLER and in high level FORTH. Even though examples are given that apply to the North Star computer, we hope that you can learn from them and apply similar techniques to your own system.

Before proceeding, the OmniFORTH ASSEMBLER should be read and understood. You should also review how your particular system and CPU handles interrupts. This discussion assumes that you already understand something about interrupts and FORTH in general.

Only 8080 or Z80 mode 0 interrupts are shown in the following examples. We hope that once you understand how the mode 0 interrupt processing works, you can advance to the other modes supported by your CPU.

The initial interrupt response routine must be written in machine code because registers that were being used when the interrupt occurs must be saved. The remaining interrupt task can be done in assembly (which has the advantage of execution speed) or you may switch to FORTH for the ease of high level code. Finally, the restoring of registers and return to normal processing will be coded in assembly.

. . . . . .

SCR # 62 North Star Disk Read/Write Interrupt Patch

It should be noted that the North Star disk system (and most other disk systems) cannot be interupted while making disk transfers.

This patch will enable your OmniFORTH to access the North Star disk when using interrupts. It is a new definition of R/W that disables interrupts before accessing the disk and enables interrupts after the disk transfer has been completed.

a) Code routines to disable and enable interrupts.

```
CODE :DI  DI  NEXT JMP

CODE :EI  EI  NEXT JMP
```

b) New FORTH word to read / write disk with interrupt disabling and enabling.

```
HEX
: %R/W    SWAP   T&SCALC
      >R  SEC @  DENSITY @  0=  1+  R>
      DENSITY @  IF 80 ELSE 0 THEN
      DRIVE  @   OR   :DI  RWDSK  :EI
      DISK-ERROR ! ;
```

This new definition of R/W requires the same parameters (addr blk f) on the stack for the address of the buffer, block number, and flag to write f=0 or read f=1.

c) Define the Patch new R/W to old R/W word.

```
: PATCH-R/W
      ' %R/W  CFA  ' R/W  !
      ' ;S    CFA  ' R/W 2+ ! ;
```

d) Execute the patch to install new R/W.

```
PATCH-R/W
```

This will patch the new %R/W code field into R/W and store the ;S return to complete the patch.

. . . . . .

SCR # 63 North Star Real Time Clock Counter

This example illustrates how the North Star real time clock interrupt can cause a counter INTC to be incremented. It runs as an independent task to FORTH where FORTH can read INTC at anytime to test it.

a) Create a FORTH variable to hold the count.

```
HEX   0 VARIABLE INTC
```

b) Invoke the assembler and store the entry point on the stack.

```
ASSEMBLER   HERE
```

After the response code is complete, we will code a JUMP to HERE at the interrupt vector location used by the clock.

c) Save any registers that will be used or altered by the interrupt response code.

```
H PUSH   PSW PUSH
```

d) The interrupt task, increment the variable INTC.

```
INTC LHLD   H INX   INTC SHLD
```

e) Reset the real time clock flag (system dependent).

```
50 A MVI   6 OUT
```

f) Restore the saved registers, enable the interrupt, and return from the interrupt.

```
PSW POP   H POP   EI   RETI
```

g) Poke a JMP HERE at the interrupt vector address 0.

```
C3 0 C!   1 !
```

This completes the interrupt code and installs a jump to HERE (the beginning of our routine) in the interrupt vector (locations 0-2).

h) The following routine is used to start the clock.

```
: ARM-CLOCK  C0  6  P!  :EI  ;
```

CAUTION: Remember that the North Star disk cannot be interrupted. Don't forget to patch in the new R/W disk word displayed on screen 62 before attempting disk access.

. . . . . .

SCR # 64 FORTH High Level Response Code

This example is simplified and is used only to illustrate the use of high level FORTH in the interrupt code.

a) Create a code routine that will restore the registers and return from the interrupt.

```
CODE :RETI
     DI  PSW POP  I POP  W POP  H POP  EI RETI
```

b) Write the high level FORTH interrupt response code (you should try to keep it as short as possible).

```
: INT1  ." INTERRUPT 1 "  CR  :RETI ;
```

Note that the semicolon ; will never execute in INT1 because :RETI will return to the interrupted process by executing the RETI instruction.

c) Evoke the assembler, save all the registers that will be used in your response, and then enable interrupts.

```
ASSEMBLER HERE
   H PUSH   W PUSH   I PUSH   PSW PUSH   EI
```

d) Cause execution of INT1 by loading the PFA (parameter field address) of INT1 into the interpreter pointer and then Jump to NEXT.

```
' INT1 I LXI   NEXT JMP
```

e) Poke a JMP HERE into the interrupt vector at memory location 08H. Note that HERE was pushed onto the stack in step c) above and points to the beginning of the assembly code to save program registers.

```
C3 8 C!   9 !
```

f) Code a retart to enable testing interrupt.

```
CODE :RST1  1 RST  NEXT  JMP
```

The RST restart intruction, or software interrupt, performs the same action as a call, but Jumps to only one of eight memory locations.

Inst.	Vector	Inst.	Vector
0 RST	00H	1 RST	08H
2 RST	10H	3 RST	18H
4 RST	20H	5 RST	28H
6 RST	30H	7 RST	38H

. . . . . .

SCR # 65 Nonmaskable Interrupt to ABORT

This can be a highly useful routine if you have access to the NMI interrupt line. It allows you to make an ABORT button that can interrupt runaway FORTH routines. This routine will return control to you without requiring you to reload your application.

a) The interrupt response code in this example is high level FORTH and prints "NMI" on the console before executing ABORT.

```
: NMI  CR  ." NMI "  ABORT ;
```

b) Create initial interrupt response code.

```
ASSEMBLER HERE
     EI   ' NMI  I LXI   NEXT JMP
```

c) Poke the JMP HERE at the NMI vector location 66H.

```
C3 66 C!   67 !
```

Note that this example does not require saving program registers, the environment, or returning from the interrupt. ABORT creates a new environment for OmniFORTH by clearing the stack and returning control to the operators terminal.

. . . . . :S