FORTH IMPLEMENTATION NOTES - ACE SYSTEMS

TONY HILL
RR #2
HAMILTON, ONTARIO
L8N 2Z7

As mentioned in the last issue of IPSO FACTO, FORTH for the 1802 is now available and ready to go. In fact, ACE is now distributing copies for essentially the distribution cost. This article is to tell you a very little bit about FORTH, and to explain how to go about bringing it up on your system. Further articles are planned, but for now this should be enough to get you up and running.

FORTH is a programming environment which encorporates both a language, a compiler/interpreter, a "run time" package, an editor/assembler and an operating system. The language is to BASIC as a four function calculator is to a Hewlett-Packard programmable calculator. (This is actually a good analogy as FORTH and the HP calculator both work in reverse polish notation and can seem confusing until you come to appreciate the elegance of the system). FORTH is much faster than interpreted BASIC and offers a much simpler interface with the "real" machine. It would be impossible to describe FORTH in an article twice as long as this, so I think it will suffice to say that it is an interesting language which tends to become addicting. In view of the shortage of other good languages for the 1802, it becomes especially attractive.

Having said that, it's time to explain how to get this "wonderful" language up on your system. First of all, I should point out that there is a well organized group of FORTH users called the Forth Interest Group (or FIG). They have released "standard" FORTH implementations for a large number of systems, including most of the popular micros AND the 1802. I'll talk more about them and the 1802 approved version in future articles. For now it is sufficient to point out that they will sell you a general FORTH implementation manual and a source listing for 1802 systems. You will need both if you intend to get into FORTH seriously, but most especially the implementation manual. It lists the exact definition of all FORTH programming words and is the basic reference for FORTH systems. Note however that it does not teach you to program in FORTH, and at this point I have not read enough of the available books on FORTH to comment on a good basic learning text. The August 1980 issue of BYTE, which was dedicated to FORTH, is not a bad place to start though.

So, to get started, order the implementation manual and the source listing (or get photocopies from a friend, a practice encouraged by FIG) and load the code into your system. You can either type the whole 5-1/4 K in one byte at a time, or buy PROM's or tape from the club and transfer it onto your system that way. (See the end of this article for ordering information for all items mentoned). Then you must customize the I/O to match your system. FIG code includes some sample I/O routines but if you already have a monitor with your own routines callable by SCRT, I would recommend using those. Assuming you are going to do just that, here's how to patch them in. Example code is given for an 8k of RAM system.

FIG-FORTH code occupies memory from 005E to 153C in the basic version supplied by FIG. The user customizes this by adding any initialization code required for his system (usually in the space between 0000 and 005E) and some I/O interface routines. A tested method of doing this is explained below. The other customization required is to allocate RAM space for FORTH stacks and buffers. Usually the top two pages of RAM are used for this. The 16 bit addresses are then stored as follows:

MEMORY LOCATION	ADDRESS OF -	FUNCTION	EXAMPLE
006E/F	Top RAM page	USER variable area	1F00
0070/1	Top page - 1	Computation stack	1E00
0072/3	Top byte of top page - 1	Return stack	1EFF
0074/5	Half way up top page - 1	Terminal input buffer	1E80

INITIALIZATION CODE

Code to initialize SCRT registers, video cursor addresses, baud rates or any system dependant functions can be installed in memory at addresses 0000 to 005D. FORTH actually starts at 005E and so the space below that address is available to you. (Your initialization code should end with a BR 5E). The initialization code should set R3 as the program counter and ACE systems should also load the address of SCRT Call and Return routines into R4 and R5. R2 can be set as the X register if you like, but FORTH will reset it for itself. Note that FORTH itself does not use SCRT, R4 or R5. Sample code is provided at the end of this article.

I/O

There are four I/O routines that the user must supply. They are patched in by storing the start address of each of the routines at the following locations-

MEMORY LOCATION	FUNCTION	8K EXAMPLE
0543/4	character output routine	153D
0573/4	issue a carriage return	1550
055E/F	character input routine	1565
056C/D	test for break condition	157C

The I/O routines are entered with R3 as the PC. All I/O routines end with a SEP RC.

Code to interface to your monitor I/O routines may be added starting at location 153D. Change locations 007A/007B to the address of the next free byte after the end of that code. Also store this value at 007C/007D. In the example listing, this address would be 1585.

Registers 0,1,4,5,6,E,F are not used by FORTH and may be used as required. FORTH uses R7 and R8 as temporary registers only and so they are available for use as well. However on reentry into any I/O routine they may be changed from what they were left as.

R2 is a FORTH stack pointer and that stack may be used if it is cleaned up before exit from I/O routines. The R2 stack is a grow down in memory stack, and is left pointing to the next free byte. This usage is consistant with SCRT techniques. Note that R2 might not be set as the X register on entry to the I/O routines, so do so if you intend to use it as such.

Registers 9,A,B,C,D are reserved for use by FORTH and must be saved and restored if they are used by your monitor's I/O routines. Pushing them on the R2 stack is a good way of doing this.

OUTPUT ROUTINE

The character output routine is know as EMIT in FORTH. Data is passed to it as a byte pointed to by R9. EMIT should increment R9, load the byte then pointed to by R9 and pass it to an output device. It should then decrement R9 three times. See the example listing of EMIT that allows you to patch in the output routine in your monitor.

CARRIAGE RETURN

A routine must be provides to cause your output device to perform a carriage return and a line feed when called. There are no parameters passed to it. Note that if your output device automatically does a line feed when it receives a carriage return, you should modify the patch so that it does not send a line feed as well. An example is provided.

INPUT ROUTINE

The FORTH input routine is called KEY. It has no parameters passed to it. KEY should read a character from the keyboard, increment R9 three times and store the character read in at the memory location then pointed to by R9. It should then decrement R9, and store a 00 there. Again, see the example provided.

BREAK ROUTINE

The FORTH routine that checks for a break condition is called QTERM. It should increment R9 three time, store a 00 at the memory location pointed to by R9, decrement R9 and store a 00 if there is a no break or a 01 if there is a break condition. (If there is to be no break condition, then store a 00 all the time.) The example routine is a dummy break routine that can be used to get your system up initially.

ORDERING INFORMATION

FIG-FORTH information may be ordered from the following address-

FORTH INTEREST GROUP
P.O. BOX 1105
SAN CARLOS, CA.
94070

Prices are $15 in the USA and $18 anywhere else each for the listing or the installation manual. These figures are in U.S. dollars and FIG requires certified checks or money orders drawn on a U.S. bank; or a VISA or MASTERCHARGE number and the expiry date.

ACE is selling fig-FORTH code to its members on three 2716 EPROM's or ELF II format tape at $30 and $10 (Canadian or US) respectively. The intention of the EPROM's is to allow you to read them into your system and use your monitor to move the data into RAM starting at located a 0000. You do not have to have EPROM memory addressed at memory location 0000 and in fact it is not even particulary recommended. See the order page in this issue for our usual ordering information.

WHERE TO GET HELP

Anyone who has problems with getting FORTH up and running, or having any general questions can feel free to write me. I'll answer all letters, and even if I don't know the solutions to your problem I'll try to make suggestions. (I would appreciate a stamped and addressed envelope from any CANADIAN members that write).

I would also be interested in hearing from anyone now running FORTH who has any comments or tips about the 1802 implementation. Future IPSO FACTO articles will include information on how to get the FIG editor and RAM disk simulation running, as well as some neat little tricks and ideas you may find handy. Thanks to Ken Mantei for his FORTH notes, on which this article was based, and without which I would have had a hard time getting FORTH running.

Good Luck ;S (<-- a little FORTH "in joke")

```
;*************************************************************************
;*                                                                       *
;*  FORTH I/O CODE - FOR INTERFACE TO A RESIDENT MONITOR                 *
;*                                                                       *
;*                     THIS CODE IS INTENDED TO PROVIDE AN EXAMPLE OF HOW *
;*                     TO INTERFACE THE I/O ROUTINES IN YOUR RESIDENT    *
;*                     MONITOR TO FIG-FORTH.  CAREFUL STUDY OF THE REGISTER *
;*                     USAGE OF YOUR MONITOR IS NECESSARY TO INSURE THAT *
;*                     ANY OF THE RESERVED FORTH REGISTERS IT USES ARE   *
;*                     SAVED BEFORE THE MONITOR ROUTINES ARE CALLED.     *
;*                     MODIFY THIS CODE ACCORDINGLY.                     *
;*                                                                       *
;*************************************************************************
```

```
                                   ;**************************************************
                                   ;*  SAMPLE INITIALIZATION CODE                    *
                                   ;**************************************************

0000  F8 07   INIT:   LDI      START              ; SET R(3) AS THE PC
0002  A3              PLO      R3                 ;
0003  F8 00           LDI      #00                ;
0005  B3              PHI      R3                 ;
0006  D3              SEP      R3                 ;
0007  F8 XY   START:  LDI      CALL/256           ; SET UP SCRT REGISTERS
0009  B4              PHI      R4                 ;      R(4) AND R(5)
000A  F8 XY           LDI      RETURN/256         ;
000C  B5              PHI      R5                 ; ( ANY OTHER INITIALIZE
000D  F8 XZ           LDI      CALL               ;   CODE WOULD GO HERE
000F  A4              PLO      R4                 ;   TOO )
0010  F8 YZ           LDI      RETURN             ;
0012  A5              PLO      R5                 ;
0013  30 5E           BR       #5E                ; JUMP TO START OF FORTH
```

```
                                   ;**********************************************
                                   ;*  EMIT - CHARACTER OUTPUT ROUTINE           *
                                   ;**********************************************

                      .ORG     #153D

153D  19      EMIT:   INC      R9       ; SETUP R(9)

153E  E2              SEX      R2       ;
153F  9A              GHI      RA       ; EXAMPLE OF HOW TO SAVE A
1540  73              STXD              ;  RESERVED REGISTER IF USED
1541  8A              GLO      RA       ;  BY THE MONITOR OUTPUT ROUTINE
1542  73              STXD              ;

1543  09              LDN      R9       ; GET OUTPUT BYTE
1544  D4 WX ZY        +CALL    OUTPUT   ; CALL MONITOR OUTPUT ROUTINE

1547  60              IRX               ; EXAMPLE OF HOW TO RESTORE THE
1548  72              LDXA              ;  REGISTER SAVED AT THE START
1549  AA              PLO      RA       ;  OF THIS ROUTINE
154A  F0              LDX               ;
154B  BA              PHI      RA       ;

154C  29              DEC      R9       ;
154D  29              DEC      R9       ; CLEAN UP R(9) FOR FORTH
154E  29              DEC      R9       ;
154F  DC              SEP      RC       ; RETURN TO FORTH INTERPRETER
```

```
                                 ;****************************************************
                                 ;*  CR - CARRIAGE RETURN OUTPUT ROUTINE             *
                                 ;****************************************************

1550  E2           CR:    SEX     R2       ;
1551  9A                  GHI     RA       ; EXAMPLE OF HOW TO SAVE A
1552  73                  STXD             ;  RESERVED REGISTER IF USED
1553  8A                  GLO     RA       ;  BY THE MONITOR OUTPUT ROUTINE
1554  73                  STXD             ;

1555  F8 0D               LDI     #0D      ; LOAD A CARRIAGE RETURN
1557  D4 WX ZY            +CALL   OUTPUT   ; PASS IT TO MONITOR OUTPUT
155A  F8 0A               LDI     #0A      ; LOAD A LINE FEED
155C  D4 WX ZY            +CALL   OUTPUT   ; PASS IT TO MONITOR OUTPUT

155F  60                  IRX              ; EXAMPLE OF HOW TO RESTORE THE
1560  72                  LDXA             ;  REGISTER SAVED AT THE START
1561  AA                  PLO     RA       ;  OF THIS ROUTINE
1562  F0                  LDX              ;
1563  BA                  PHI     RA       ;

1564  DC                  SEP     RC       ; RETURN TO FORTH INTERPRETER


                                 ;*****************************************************
                                 ;* KEY - CHARACTER INPUT ROUTINE                     *
                                 ;*****************************************************

1565  19          KEY:    INC     R9       ; SET UP STORAGE AREA
1566  19                  INC     R9       ;
1567  19                  INC     R9       ;

1568  E2                  SEX     R2       ;
1569  9A                  GHI     RA       ; EXAMPLE OF HOW TO SAVE A
156A  73                  STXD             ;  RESERVED REGISTER IF USED
156B  8A                  GLO     RA       ;  BY THE MONITOR OUTPUT ROUTINE
156C  73                  STXD             ;

156D  D4 ZX WY            +CALL   INPUT    ; GET INPUT FROM MONITOR ROUTINE

1570  60                  IRX              ; EXAMPLE OF HOW TO RESTORE THE
1571  72                  LDXA             ;  REGISTER SAVED AT THE START
1572  AA                  PLO     RA       ;  OF THIS ROUTINE
1573  F0                  LDX              ;
1574  BA                  PHI     RA       ;

1575  9F                  GHI     RF       ; GET BYTE PASSED BACK FROM INPUT
1576  59                  STR     R9       ; SAVE IT
1577  29                  DEC     R9       ; CLEAN UP STORAGE AREA
1578  F8 00               LDI     #00      ;
157A  59                  STR     R9       ;
157B  DC                  SEP     RC       ;


                                 ;*******************************************
                                 ;* QTERM - BREAK CONDITION TEST ROUTINE    *
                                 ;*******************************************

157C  19         QTERM:   INC     R9       ; SAMPLE DUMMY BREAK ROUTINE
157D  19                  INC     R9       ;
157E  19                  INC     R9       ;
157F  F8 00               LDI     #00      ;
1581  59                  STR     R9       ;
1582  29                  DEC     R9       ;
1583  59                  STR     R9       ;
1584  DC                  SEP     RC       ;
```