FOCAL

FOCAL has been available for the 6502 for quite awhile now and offers some advantages that make it an attractive alternative to BASIC. The fact that an assembly-listing is available makes it especially beneficial to those of us who are interested in delving into the inner workings of a high-level language and perhaps modify it and/or extend to suit our whims. FOCAL includes provisions for adding to the command language and makes interfacing to machine language functions a piece of cake. BASIC offers none of this.

FOCAL is available from two sources at this time; ARESCO (P.O. Box 43, Audubon, Pa 19407) and 6502 PROGRAM EXCHANGE (2920 Moana, Reno, NV 89509). They both offer FOCAL for about the same price, however the Program Exchange has developed a library of FOCAL programs including StarTrek, so I would highly recommend that you get their flyer and see whats available (I think it costs 50c). Also they have an excellent 104 page user manual which is available for $12.00. I just received it in time to mention it in this issue and can recommend it as an effective means for becoming familiarized with FOCAL operations.

Up to this point, the biggest single disadvantage of FOCAL has been that there was no built-in way of saving and loading FOCAL programs using cassette or disc. Well, I have found a way to accomplish this and if you'll be patient I'll impart the knowledge to you.....(by the way, the absolute memory locations hold true only for the Version 3D (and possibly FCL-65E) other implementations will have to know where their particular pointers are).....

SIMPLE!!!All you have to do is to save the pointers PBADR ($31,32) and VARBEG ($3E,3F) and the data that is referenced indirectly between them. For instance: PBADR points to $360A and VARBEG points to $390F. Your storage device driver program should dump all data from $360A to $390F and also the pointers themselves which must be reinitialized when you re-load that particular program. How else is FOCAL supposed to know where that program is???

No, I haven't actually written a cassette friver for FOCAL (I use disc) but don't see any problem at all doing just that.. But, wait a minute...before we all go off on our own and write our own version of the ultimate FOCAL cassette handler, let's figure out some sort of a "standard". I thing it's important to be able to work with named records instead of our regular ID number. All we really need to do is extend the ID portion of the KIM cassette format to include a fixed number of ASCII characters (say 8) and include an area for the pointer information that we need. It's necessary that we have some proposals by the next issue so we can get started on our driver software. As far as the command extension to FOCAL is concerned, let's reserve the letters "K" for KEEP (which will save the program on cassette) and "L" for LOAD (which will load a program from cassette into memory).

We may want to use a binary recording format for increased speed and could probably "lift" some of the code from the cassette driver presented in issue #7/8 (written by John Oliver).

More next time. Got any ideas about FOCAL that you'd like to share?

****

LANGUAGE LAB:

focal

At this point in time, FOCAL is the most documented of the high level languages which run on our beloved 6502. Having a complete source listing is definitely invaluable.

This openness on the part of the implementor has made it so easy to fidget around with FOCALs internala and even fix a problem or two.

One of the things that did sort of annoy me was the almost 1 character delay encountered when typing in FOCAL program text from a hard-copy terminal. (I have the Aresco version).

As it turns out, thanks to the source listing, I found that FOCALs author did some elaborate arm waving to prevent KIM from echoing the character which has been input to the TTY port. No small feat, I might add, since KIM echoes the tty input in hardware (not software!).

(If you're wondering how - FOCAL makes the terminal think that the character getting echoed is a RUBOUT character - which the terminal ignores).

Anyhow, I don't quite know why FOCAL bothers to do this - the character ends up getting echoed in software anyway. (There is a function which does enable to echo to be shut off completely).

Make the following changes to FOCAL. This patch was found in the FOCAL User Manual ($12.00 from the 6502 Program Exchange) and was apparently an update for FCL-65E.

```
34AA   84  A5       OUT    STY  SAVYR
                           ; save "Y"
34AC   20  A0  1E          JSR  OUTCH
34AF   A4  A5              LDY  SAVYR
                           ;restore "Y"
34B1   18                  CLC
                           ; indicate success
34B2   60                  RTS
                           ; return


34B3   E6  76        IN    INC  HASH
                           ;bump random seed
34B5   2C  40  17          BIT  SAD
                           ; test input port
34B8   30  F9              BMI  IN
                           ; loop 'til start bit
34BA   A5  6B              LDA  ECHFLG
                           ; get echo flag
34BC   D0  03              BNE  NOECH
                           ; branch for no echo
34BE   4C  5A  1E          JMP  GETCH
                           ; get character with
                             echo

34C1   AD  42  17  NOECH   LDA  SBD
                           ; get port status
34C4   29  FE              AND  #FE
                           ; turn off bit
34C6   8D  42  17          STA  SBD
34C9   20  5A  1E          JSR  GETCH
34CC   48                  PHA
                           ; save character
34CD   AD  42  17          LDA  SBD
                           ; get port status
34D0   09  01              ORA  #01
                           ; turn on bit
34D2   8D  42  17          STA  SBD
                           ; make echo a rubout
34D5   A9  00              LDA  #0
                           ; get a null character
34D6   20  A0  1E          JSR  OUTCH
                           ; echo it
34D9   68                  PLA
                           ; restore input char.
34DA   18                  CLC
                           ; indicate success
34DB   60                  RTS
                           ; return
```



```
28F2   EA EA EA  was 20 02 29
35B4   B3        was A5
```

Faster typists will really notice a difference.

A really neat feature of FOCAL is the fact that you can add specialized functions.

Function calls consist of four (or fewer) letters beginning with the letter "F" and followed by a parenthetical expression which may contain an argument to be passed to the function.

There are a number of functions which are included in FOCAL, such as:

- FINT - returns the integer portion of a number
- FABS - returns the absolute value of a number
- FMEM - allows one to examine or deposit into a memory location.

FOCAL decides which function is being called by performing a "HASHING" of the function name and searching for that value in a function dipatch table. Using hash codes simplifies the lookup table design structure quite a bit. It may even speed things up a bit also.

If you wish to install your own functions, the hash code for the particular function name and the function address must be installed in the extra space provided in the lookup table.

Figuring out the hash code for your function is not so easy, however, unless you use FOCAL itself to do the computation.

In version 3D, place a BRK or JMP KIM at location $29EF. Then execute the following command:

```
SET X = F???(1)
```

where F??? is your new function name (FADC for example) and (1) is there because you need a parameter of some sort.

Program control will then be returned to KIM, or wherever your BRK vector pointed, and the hash code will be found in location $0065 as well as the Accumulator and the "X" register.

Several readers are preparing articles on FOCAL additions and modifications, so we have alot to look forward to in this section.

I just saw the latest Dr. Dobbs Journal at the newstand (computer store newstand, that is) and noticed that they published a rather large FOCAL program. (I don't recall the issue number).

Do YOU have any FOCAL articles or programs that you'd like to see in print? Then send 'em in.

I highly recommend the $12.00 FOCAL USER MANUAL from the 6502 Program Exchange to those who are learning to program in this language as well as those who are just curious and perhaps want to see how FOCAL compares to BASIC.

At the present time, FOCAL for the 6502 is available from two sources. Write to them for pricing and availablity.

ARESCO
PO BOX 43
Audobon, PA
19407

6502 Program Exchange
2920 Moana
Reno, Nevada 89509

focal

Lots of neat mods are in store for FOCAL. We're going to add a cassette save & load facility, a Basic-like data statement, output to KIM's seven segment display, the ability to handle arrays of strings, an improved print command, a machine language subroutine call and a few minor fixits and speed-up mods.

Before we do all this, however, we need some room. The present size of the Aresco V3D is about 6K so let's stretch it out to an even 8K and give ourselves a little breathing room. If you examine the listings (love them listings!), you'll notice that the user program must start right after Focal because of line number 00.00 at $35EB.

(One problem: all these mods pertain to V3D which is distributed by Aresco and not necessarily to FCL-65E which is distributed by 6502 Program Exchange. The symbolic addressing info might pertain to FCL-65E but since I don't have a listing of FCL-65E, I can't be sure. FCL-65E might be an updated version of V3D but I can't be sure).

Extend V3D FOCAL to 8K by moving $35EB through $360A to $3FE0-$3FFF. This moves the line 0.0 startup message to the top of the 8K block that will be used by FOCAL. Some zero page pointers must also be changed to allow for the above mod.

```
change:  TEXTBEG $002F   FROM $EB TO $E0
                 $0030   FROM $35 TO $3F
         PBADR   $0031   FROM $09 TO $FE
                 $0032   FROM $36 TO $3F
         VARBEG  $003E   FROM $0A TO $FF
                 $003F   FROM $36 TO $3F
         VARST   $0040   FROM $0A TO $FF
                 $0041   FROM $36 TO $3F
         VAREND  $0042   FROM $0A TO $FF
                 $0043   FROM $36 TO $3F
         PDLIST  $0053   NO CHANGE
                 $0054   FROM $3F TO $5F
```

This is the FOCAL pushdown stack and should be set to some convenient page up out of the way of FOCAL programs. $5F assumes a 16K system.

Another thing that must be improved is the way FOCAL sets up zero-page. Actually, it doesn't. I really can't understand why the implementers overlooked this problem. Oh well...it's easy to fix. At $3F00 add the following zero page intialization routine. $3F00 will become the new cold start address.

```
         ZPAGE  = $0
         ZSTORE = $3F10
         LENGTH = $BF      ; NUMBER OF BYTES
         STARTF = $2000
         *=$3F00
CSTART   LDX #0          ; INITS THE LOOP COUNTER
ZLOOP    LDA ZSTORE,X    ; START MOVING DATA
         STA ZPAGE,X
         INX
         CPX #LENGTH+1
         BNE ZLOOP
         JMP STARTF      ;   PAGE IS SET UP
                         ; GO TO FOCAL
```

Ok now, we have stretched out FOCAL to 8K and added a Z page initialization routine. What next? We'll start adding mods from $35EB-$3EFF.

More next time.....

focal

FOCAL MODS

....speed it up
a little...

from Bernhard Mulder
Mozart Str 1
6744 Kandel
West Germany

We change the procedure EATCR (and EATCR1) which is called by the findline, which in turn is called from the GOTO, IF, ON, DO command routines.

We assume that the carriage return char is in memory and avoid the call of the routine GETC, where switches are tested which will never be set, when we caome from EATCR (start the following routine at $26D0 in the Aresco version 3D and $26DD in the "6502 Program Exchange" FCL-65E (V3D).

```
C6  2A    ECR1  DEC  TXTP     ;EATCP1
A4  2A    EACR  LDY  TXTP     ;EATCR
A9  0D          LDA  #0D      ;load CR which we are
                               looking for
D0  01          BNE  TST1
C8        LABL  INY           ;next character in line
D1  28    TST1  CMP  (TXTA),Y;C.R. found already?
D0  F8          BNE  LABL     ;branch if noy
B1  28          LDA  (TXTA),Y;store away for others
85  2B          STA  CHAR
C8              INY
98              TYA           ;calculate address
                               part CR.
18              CLC
65  28          ADC  TXTA
85  28          STA  TXTA
85  33          STA  TXA2
A5  29          LDA  TXT1
69  00          ADC  #00
85  29          STA  TXT1
85  34          STA  TA21
A9  00          LDA  #00
85  2A          STA  TXTP
85  35          STA  TXP2
60        ENCR  RTS
```

Make the following changes to Aresco V3D

```
208D 20 D0 26 (was 20 D7 26)
21FF 20 D0 26 (was 20 D7 26)
22E1 20 D0 26 (was 20 D7 26)
2752 20 D2 26 (was 20 D0 26)
```

or make the following changes to the Program Exchange FCL-65E

```
208D 20 DD 26 (was 20 E4 26)
21FF 20 DD 26 (was 20 E4 26)
22E3 20 DD 26 (was 20 E4 26)
275F 20 DF 26 (was 20 DD 26)
```

Those of you with ROR instructions in your CPU can eliminate the ROR simulator in FOCAL with the following code.

Start at $3291 for the Aresco version 3D
Start at $3293 for the Program Exchange FCL-65E

```
7E 89 00  ROR1  ROR  EP4,X ;need not simulate ROR
E8              INX
D0 FA           BNE  ROR1
60              RTS
```

Plenty more mods in store for FOCAL. Until next issue.

focal

First of all, I want to thank Dave and Don Marsh from the 6502 Program Exchange (2920 Moana Ln., Reno NV 89509) for providing me with the source listing for their version of FCL-65E. The listing has been invaluable in getting all the mods set up for both versions of FOCAL (one from Aresco and the other from the Program Exchange).

By the way, both versions must be suitably modified as per issue #14 in order to use the modifications that will be presented. Program Exchange FCL-65E users need to move the start up message at line 00.00 to the top of the 8K block by moving the data at $35D4 through $35F3 to start at $3FE0.

In trying to cooordinate these mods across two versions of FOCAL, I've run across a zero-page usage problem in the Program Exchange versions. This version uses about 50 bytes of zero-page for terminal I/O. (According to the exchange, this was done to make FOCAL more portable between different machines). Anyhow, the long and short of it is - these I/O routines will have to be moved back into FOCAL to allow freer use of zero page.

Once the line 0.0 has been moved up to the top of the 8K block, the I/O routines from $00A9-$00DC can be moved to start at location $35D4. Of course, the internal references to OUT and IN will have to be changed to reflect the changes.

FOCAL ENHANCEMENT PACKAGE

The 'NOTES' is now distributing a very useful FOCAL enhancement package that will let you save and load complete FOCAL programs on cassette as well as lines or groups of lines and/or program variables. Commands may also be executed directly from cassette. The package was written by Joe Woodard. For ordering info, see the cassette software ad in this issue.

ADDING A CASSETTE INTERFACE AND A USER FUNCTION TO 6502 PROGRAM EXCHANGE'S FOCAL 65-E

by William C. Clements, Jr.
1489 51ST Ave East
Tuscaloosa Al 35404

The FOCAL language is really a good alternative to BASIC, at least for KIM users. Of course, it doesn't have everything. The features I missed the most were a cassette interface and the ability to execute a machine-language routine within a FOCAL program. This article shows how to add tape Load and Keep commands and how to implement a "user" function similar to that of TINY BASIC. The modifications apply to FOCAL-65 (V3D) for the KIM-1 as supplied by the 6502 Program Exchange.

Listing 1 gives the code needed to add the cassette interface commands. I began it directly after the FOCAL interpreter, because I had moved the RAM allocation for program text and variables to another area. It can go anywhere in memory that you wish, with simple relocation and listing the addresses of routines KEEP and LOAD in FOCAL's command dispatch tables. The cassette Load command enters the regular KIM monitor at $1873, and the Keep command uses a Hypertape routine in my system; it's almost a necessity to use a cassette dump routine faster than KIM's, since the memory required to store the FOCAL program statements in ASCII form can be large.

The tape operations could have been done using the existing I-O handlers provided in FOCAL, but I preferred to use conventional commands. The form of the commands is L xx to load a file having hexadecimal i.d. "xx" and K xx to record a file with i.d. "xx" onto tape.

Readers who have program control of their tape recorders might want to use these commands inside a FOCAL program to manipulate tape files. I can only use them in the immediate execution mode, since I have to push buttons on the recorder. The KIM tape routines exit to loc. zero, which my code sets up with a jump instructin. Hitting the G key on the TTY after either tape operation is through will get you back into FOCAL. All starting and ending address for the tape files are automatically set by the routines, including the final address after loading a file.

The Keep routine uses Hypertape stored in my system at loc. $C400; the jump at location TAPOUT will need to be fixed by the user to suit his own system. The jump in loc. zero restarts FOCAL at its cold start, as that's the only way I can use it. If you want to get back into the middle of an executing FOCAL program, the jump at location JMPFOC and the data bytes at locations ADLOW and ADHIGH will have to be changed.

The "user" function works like the one in TINY BASIC; it allows user-supplied machine code to be executed as a FOCAL function. The FOCAL code to invoke it is S FUSR (a_1,a_2,a_3,a_4), where the a's are the four arguments. The first, a_1, must always be present because it is the address to which the program will jump to begin the user's code. a_2,a_3,and a_4 are optional. a_2, if present, will be evaluated and the least significant eight bits stored in the accumulator before executing the user's code. a_3 and a_4, if present, are similarly evaluated and placed in the X and Y registers, respectively. Thus up to three bytes may be transmitted directly from the FOCAL program to the machine code (more of course can be transmitted in either direction by using FOCAL's version of PEEK and POKE, the FMEM function). The arguments can be constants, simple variables, or any other legal FOCAL expressions, and as such have decimal values.

As examples, the statement S FUSR(8192,0,16,10) will cause a jump to location $2000 with zero in A, $10 in X, and $0A in Y. The statements

```
1.1 S A=100
1.2 S B=13
1.3 S FUSR(625-(A+B),,B,)
```

would jump to $200 with $0D in X. Note that there are always three commas, as FOCAL uses them to tell which argument is which. If you want the variable FUSR itself to have a numerical value after its execution such as FRAN or FABS do, you can have your machine code put that value into the floating accumulator FAC1 (locs. $80-83 - see p. 7 of the 6502 Program Exchange's listing of FOCAL 65-E). Your machine code must transfer control to loc. FPOPJ (in FOCAL) when it is ready to re-enter FOCAL, and it will return to the point in your FOCAL program where FUSR was invoked. Listing 2 gives the machine code needed for adding FUSR to FOCAL.

The changes required within the tables of the FOCAL interpreter to make it recognize K. L, and FUSR and to execute the codes in Listings 1 and 2 are given now. The format follows that of the original listing of FOCAL.

ARESCO	PROGR. EXCH		
$350B	$34F4	18	BYTE HFUSR
3527	3510	36	HBYTE FUSR
3543	352C	41	BYTE FUSR
3557	3540	4B	ASCII 'K'
3558	3541	4C	ASCII 'L'
356B	3554	36	HBYTE KEEP
356C	3555	36	HBYTE LOAD
357D	3566	04	BYTE KEEP
357E	3567	33	BYTE LOAD

```
0010  2000              ;CASSETTE INTERFACE AND USER FUNCTION MODS
0020  2000              ;FOR FOCAL FROM W. CLEMENTS 1979
0030  2000
0040  2000              ;KIM LOCATIONS
0050  2000              PACK   =$1FAC
0060  2000              ID     =$17F9
0070  2000              PREG   =$F1
0080  2000              INL    =$F8
0090  2000              SAL    =$17F5
0100  2000              SAH    =$17F6
0110  2000              EAL    =$17F7
0120  2000              EAH    =$17F8
0130  2000              LOADT  =$1873
0140  2000              VEB    =$17EC
0150  2000              HYPER  =$0200          ;ADDRESS OF HYPERTAPE ROUTINE
0160  2000
0170  2000              ;FOCAL LOCATIONS
0180  2000              FOCAL  =$2000
0190  2000              GSPNOR =$29A3          ;($29B1 IN ARESCO VERSION)
0200  2000              PBADR  =$31
0210  2000              VARBEG =$3E
0220  2000              INTGER =$2F85          ;($2F93 IN ARESCO VERSION)
0230  2000              M1     =$B1
0240  2000              CHAR   =$2B
0250  2000              NXIARG =$2F7B          ;($2F89 IN ARESCO VERSION)
0260  2000
0270  2000              ;MOD LOCATIONS
0280  2000                     *=$0
0290  0000              JMPCOM *=*+2           ;JUMP VECTOR IN ZERO PAGE
0300  0002              NARGS  *=*+1           ;NUMBER OF ARGS IN USR
0310  0003              SAVA   *=*+1
0320  0004              SAVX   *=*+1
0330  0005
```

```
2000                ;THIS ROUTINE MAKES KIMS LEDS ANOTHER
2000                ;OUTPUT DEVICE FOR FOCAL
2000                ;WRITTEN BY BERNHARD MULDER
2000                ;           MOZARTSTR 1
2000                ;           6744 KANDEL
2000                ;           WEST GERMANY
2000
2000                ;THE STARTING ADDRESS FOR THIS ROUTINE
2000                ;MUST BE PLACED INTO FOCALS OUTPUT
2000                ;DEVICE TABLE SO FOCAL KNOWS WHERE IT IS.
2000                ;
2000                ;IN THE ARESCO VERSION, PUT THE HIGH ORDER
2000                ;BYTE OF THE STARTING ADDRESS FOR THE LED
2000                ;OUTPUT ROUTINE IN LOCATION $35BB AND THE
2000                ;LOW ORDER ADDRESS IN $35CO. IN THE 6502
2000                ;PROGRAM EXCHANGE VERSION OF FCL-65E, THIS
2000                ;WOULD CORRESPOND TO LOCATIONS $35A4 AND
2000                ;$35A9 RESPECTIVELY.
2000
2000                ;TO USE THE LEDS AS AN OUTPUT DEVICE,
2000                ;SIMPLY OUTPUT TO DEVICE #2 BY
2000                ;EXECUTING 'S FODV (2)'. ALL SUBSEQUENT
2000                ;OUTPUT WILL BE SENT TO THE DISPLAYS.
2000                ;
2000                PADD    =$1741
2000                SBD     =$1742
2000                SAD     =$1740
2000
2000                ;ZERO PAGE LOCATIONS
2000                        *=$A9
00A9                SPE1    *=*+1           ;THESE TWO CELLS CONTROL THE DISPLAY
00AA                SPE2    *=*+1           ;SPEED OF KIMS 7 SEGMENT DISPLAYS
00AB                SPE3    *=*+1
00AC                SPE4    *=*+1
00AD
00AD                ;START THIS ROUTINE RIGHT AFTER THE CASSETTE
00AD                ;AND USER FUNCTION MODS
00AD
00AD                        *=$36B0
36B0                        .OFF 2000
36B0
36B0  A8            SEVS    TAY
36B1  A2 01                 LDX #$1
36B3  BD 0A 37      SEV1    LDA DISP,X
36B6  9D 09 37              STA DIS,X
36B9  E8                    INX
36BA  E0 06                 CPX #$6
36BC  D0 F5                 BNE SEV1
36BE  B9 10 37              LDA TABB,Y
36C1  8D 0F 37              STA DISP+5      ;***BEWARE***NON-ROMABLE CODE HERE!!!
36C4  A2 01         ENRY    LDX #$1
36C6  B5 A9         ENR1    LDA SPE1,X
36C8  95 AB                 STA SPE3,X
36CA  CA                    DEX
36CB  10 F9                 BPL ENR1
36CD  20 E3 36      ENR3    JSR ZEI1        ;DISPLAY
36D0  C6 AB                 DEC SPE3
36D2  D0 F9                 BNE ENR3
36D4  C6 AC                 DEC SPE4
36D6  D0 F5                 BNE ENR3
36D8  18                    CLC             ;FINISH WITH NO ERRORS
36D9  60                    RTS
36DA  A0 00         ZEI2    LDY #$0
36DC  8C 41 17              STY PADD
36DF  8C 42 17              STY SBD
36E2  60                    RTS
```

```
36E3  20 DA 36    ZEI1   JSR ZEI2          ;ALL DISPLAYS OFF
36E6  A9 7F              LDA #$7F
36E8  8D 41 17           STA PADD
36EB  A2 09              LDX #$9
36ED  8E 42 17    ZEI3   STX SBD
36F0  B9 0A 37           LDA DISP,Y
36F3  8D 40 17           STA SAD
36F6  98                 TYA
36F7  48                 PHA
36F8  A0 7F              LDY #$7F
36FA  88          ZEI4   DEY
36FB  D0 FD              BNE ZEI4          ;DELAY ABOUT 500 CYCLES
36FD  68                 PLA
36FE  A8                 TAY
36FF  C8                 INY
3700  E8                 INX
3701  E8                 INX
3702  C0 06              CPY #$6           ;ALL DIGITS FINISHED ?
3704  D0 E7              BNE ZEI3
3706  20 DA 36           JSR ZEI2          ;ALL OUT
3709  60          DIS    RTS               ;DIS=DISP-1
370A
370A  00          DISP   .BYTE 0,0,0,0,0,0
370B  00
370C  00
370D  00
370E  00
370F  00
3710
3710              ;HERES THE TABLE WITH THE SEVEN SEGMENT
3710              ;CODES.  ASCII CODE=TABLE INDEX
3710
3710  00          TABB   .BYTE 0,0,0,0,0,0,0,0
3711  00
3712  00
3713  00
3714  00
3715  00
3716  00
3717  00
```

FOCAL LED OUTPUT

HEX DUMP STARTS HERE

TO SAVE SPACE... HERE'S A HEX DUMP OF THE SEVEN SEGMENT CODE TABLE.

```
3710  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
3720  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
3730  88 8A A2 00 B6 00 C6 82 B9 8F 64 C6 8C C0 98 D2
3740  BF 86 DB CF E6 ED FD 87 FF EF C1 C5 00 C1 00 D3
3750  00 F7 FC D8 DE F9 F1 BD F6 84 9E F0 B8 B7 D4 DC
3760  F3 E7 D0 ED F8 9C EA BE EE C9 00 00 00 00 00 00
3770  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
3780  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

```
0340  0005
0350  0005                 ;FOCAL MODS START HERE
0360  0005                       *=$35EB
0370  35EB                       .OFF 2000
0380  35EB  20 A3 29    SUB    JSR GSPNOR       ;GET NEXT BLANK CHAR
0390  35EE  20 AC 1F           JSR PACK         ;CONVERT TO HEX AND STORE
0400  35F1  20 A3 29           JSR GSPNOR       ;REPEAT FOR
0410  35F4  20 AC 1F           JSR PACK         ;NEXT DIGIT
0420  35F7  A5 F8              LDA INL
0430  35F9  8D F9 17           STA ID           ;SET TAPE ID
0440  35FC  A9 4C              LDA #$4C         ;SETUP JUMP LOCATION
0450  35FE  85 00              STA JMPCOM       ;IN ZERO PAGE
0460  3600  A9 00              LDA #0           ;CLEAR STATUS REG
0470  3602  85 F1              STA PREG RTS     ;AND RETURN
0480  3604
0490  3604  20 EB 35    KEEP   JSR SUB          ;SET ID ETC
0500  3607  A5 31              LDA PBADR        ;SET KIM
0510  3609  8D F5 17           STA SAL          ;TAPE REGISTERS
0520  360C  A5 32              LDA PBADR+1
0530  360E  8D F6 17           STA SAH
0540  3611  A5 3E              LDA VARBEG
0550  3613  8D F7 17           STA EAL
0560  3616  A5 3F              LDA VARBEG+1
0570  3618  8D F8 17           STA EAH
0580  361B  A9 00        ADRLOW LDA #<FOCAL
0590  361D  85 01              STA JMPCOM+1     ;MAKE JUMP INSTR. A
0600  361F  A9 20        ADRHI  LDA #>FOCAL     ;RETURN TO COLDSTART
0610  3621  85 02              STA JMPCOM+2
0620  3623  4C 00 02    TAPOUT JMP HYPER
0630  3626
0640  3626  AD ED 17    ENLOAD LDA VEB+1        ;SET ADDRESS AT END OF
0650  3629  85 3E              STA VARBEG       ;PROGRAM TEXT
0660  362B  AD EE 17           LDA VEB+2
0670  362E  85 3F              STA VARBEG+1
0680  3630  4C 00 20    JMPFOC JMP FOCAL        ;RETURN TO FOCAL
0690  3633
0700  3633  20 EB 35    LOAD   JSR SUB          ;SET ID ETC
0710  3636  A9 26              LDA #<ENLOAD     ;MAKE JUMP POINT TO
0720  3638  85 01              STA JMPCOM+1     ;THE REST OF THE TAPE
0730  363A  A9 36              LDA #>ENLOAD     ;LOAD ROUTINE
0740  363C  85 02              STA JMPCOM+2
0750  363E  4C 73 18           JMP LOADT        ;READ THE CASSETTE
0760  3641
0770  3641                 ;NOW FOR THE 'USR' FUNCTION
0780  3641
0790  3641  A9 4C        FUSR   LDA #$4C         ;SET UP JUMP LOC.
0800  3643  85 00              STA JMPCOM
0810  3645  20 85 2F           JSR INTGER       ;FET FIRST ARG. IN FAC1
0820  3648  85 01              STA JMPCOM+1     ;REARRANGE LOW AND HIGH ORDER
0830  364A  A5 82              LDA M1+1         ;BYTES INTO JUMP LOCATION
0840  364C  85 02              STA JMPCOM+2     ;THAT WILL EXECUTE USER CODE
0850  364E  A9 00              LDA #0           ;ZERO THE ARG. COUNTER
0860  3650  85 02              STA NARGS
0870  3652  20 7A 36           JSR USRARG       ;EVALUATE AND SAVE HOWEVER MANY
0880  3655  84 03              STY SAVA         ;ARGUMENTS ARE LEFT
0890  3657  20 7A 36           JSR USRARG
0900  365A  84 04              STY SAVX
0910  365C  20 7A 36           JSR USRARG
0920  365F  A5 02              LDA NARGS
0930  3661  F0 0E              BEQ JMPUSR       ;JUMP TO USER'S CODE IF NO MORE ARGS
0940  3663  C9 01              CMP #$1
0950  3665  F0 08              BEQ STAC         ;SET 'A'=ARG, IF ONE ARG LEFT
0960  3667  C9 02              CMP #$2
0970  3669  F0 09              BEQ STACX        ;SET 'A'=ARG1, 'X'=ARG2 IF TWO LEFT
0980  366B  C9 03              CMP #$3
0990  366D  F0 05              BEQ STACX        ;SET 'A'=ARG1, 'X'=ARG2, 'Y'=ARG3
1000  366F  A5 03        STAC   LDA SAVA         ;ARG1 IN 'A'
1010  3671  4C 00 00    JMPUSR JMP JMPCOM       ;GO DO USER'S CODE
1020  3674  A5 04        STACX  LDA SAVX
1030  3676  AA                   TAX
1040  3677  4C 6F 36           JMP STAC
1050  367A
1060  367A  A5 2B        USRARG LDA CHAR         ;GET CURRENT CHARACTER
1070  367C  C9 2C              CMP #',          ;ANOTHER ARGUMENT?
1080  367E  F0 04              BEQ GETARG       ;YES, GO GET IT
1090  3680  C9 29              CMP #')          ;END OF STATEMENT?
1100  3682  F0 06              BEQ RET          ;YES, RETURN NOW
1110  3684
1120  3684  20 7B 2F    GETARG JSR NXIARG       ;EVALUATE NEXT ARG.
1130  3687  A8                   TAY
1140  3688  E6 02              INC NARGS        ;COUNT ARGS PAST FIRST
1150  368A  60          RET    RTS              ;RETURN
1160  368B
1170  368B                       .END
```

BUGS

In Issue #16, two boo-boos were found by sharp readers. I really goofed the Focal cassette interface on page 15. In line 0150 of the listing, HYPER should be addressed to $C400 (as in paragraph 4, page 15) not $0200. Also in that same listing, line 290 should read JMPCOM ***+3 (not JMPCOM ***+2). That and the missing RTS instruction after line 470 through the whole thing off. Here's a hex dump of the corrected program.

```
35EB  20 A3 29 20 AC
35F0  1F 20 A3 29 20 AC 1F A5 F8 8D F9 17 A9 4C 85 00

3600  A9 00 85 F1 60 20 EB 35 A5 31 8D F5 17 A5 32 8D
3610  F6 17 A5 3E 8D F7 17 A5 3F 8D F8 17 A9 00 85 01
3620  A9 20 85 02 4C 00 02 AD ED 17 85 3E AD EE 17 85
3630  3F 4C 00 20 20 EB 35 A9 27 85 01 A9 36 85 02 4C
3640  73 18 A9 4C 85 00 20 85 2F 85 01 A5 82 85 02 A9
3650  00 85 03 20 7B 36 84 04 20 7B 36 84 05 20 7B 36
3660  A5 03 F0 0E C9 01 F0 08 C9 02 F0 09 C9 03 F0 05
3670  A5 04 4C 00 00 A5 05 AA 4C 70 36 A5 2B C9 2C F0
3680  04 C9 29 F0 06 20 7B 2F A8 E6 03 60 36 36 04 00
*
```