FORTH NOTEBOOK

by C. H. Ting . PHD

OFFETE ENTERPRISES, INC.

1985

FORTH NOTEBOOK

by C. H. Ting , PHD

OFFETE ENTERPRISES, INC.

1985

FIRST EDITION, SEPTEMBER 1983 (FIRST PRINTING)

MAY 1985 (SECOND PRINTING)

****

****

Printed in the United States of America

by

Offete Enterprises, Inc.

1306 SOUTH "B" STREET

SAN MATEO, CALIFORNIA 94402

TEL: (415) 574-8250

PREFACE

My experience in the last few year in teaching FORTH to people of different backgrounds was that Starting FORTH was quite sufficient as a textbook, introducing people to the basics of FORTH. However, to more experienced programmers, the materials in it are not enough. Though there are numerous examples and exercises, most of them are brief and the scope of the examples is also limited. I was constantly asked to provide real applications besides teaching examples. Another shortcoming is that it does not deal with the internals of the FORTH operations at the machine code level. There is a wide gap between Starting FORTH and the source code as shown in the fig-FORTH Model and Installation Manual.

We've seen a number of books on FORTH appearing in the last year. They are welcome news. However, most of them did not exceed Starting FORTH either in the scope of their treatments or in the amount of examples towards more realistic applications. Starting FORTH probably will remain the bible of FORTH to beginners for many years to come.

There are two areas where FORTH books are of urgent need: one is in presenting program design with examples of moderate complexity, and the other is to explore the FORTH computer in a deeper level, dealing with real CPU's and instruction sets. What I wanted to present in this Notebook are some of my personal efforts in these two directions.

In this Notebook, I collected many programs which I used for teaching purposes. Many interesting games were translated from BASIC into FORTH. These translations offer some insight into the natures of these two different languages. I am very interested in the computerization of the ancient oriental GO game. A few programs in this field are included. My profession requires the utilization of digital image processors. A number of programs dealing different aspects of image processing were used to demonstrate that FORTH is a more natural language for this type of applications. I included them here as working examples of practical applications. They are of very limited use to people who do not have the same image processor as mine. Nevertheless, they do illustrate many commonly used techniques in handling digital images.

Recently, I was involved in a bit slice microprocessor project and developed a microassembler to assemble microcodes. It is a very simple solution to a rather complicated problem. I simply connot resist the temtation to publish it here. Two other programs of similar nature were included for the same reason. One was developed when I was with Yang Ming Medical College in Taipei, to identify bacteria according to the results of a set of tests. Another one was to explore the technique of continuous Fourier transform I proposed a number of years ago.

I taught a number of short courses on FORTH and used FORTH screens to prepare viewgraphs. These courses were generally organized into eight sessions, treating subjects ranging from data and return stacks to text and inner interpreters. These viewgraphs are also collected in this Notebook for those who might want to use them as teaching aids.

In my book Systems Guide to fig-FORTH, I avoided the very unpleasant task of explaining the nucleus words which were in 6502 machine codes as published in the fig-FORTH Model and Installation Manual. I worked out these words using PDP-11 codes with some commentary. This section, hopefully, will be a worthy appendix to the Systems Guide.

I am greatly indebted to the members of the Taiwan FIG Chapter who encouraged me to put together this Notebook. The very informal format adopted here was decided when I was participating in their Summer Workshop, July 21 to 25, 1983. It is a response to their immediate needs which I believe are shared by many young FORTH communities. Applications programs dealing with real world problems sometimes are more interesting than textbook examples which were designed to sell a viewpoint or a language.

Thanks are due to Mr. Anson Averell who carefully read through the manuscript and made numerous suggestions and corrections.

All the programs presented in this Notebook were fully debugged as far as I could test them. However, I cannot guarantee that they are bug-free. Users beware! Like all responsible software maufacturers, this is the proper place to state that I shall not be liable for errors contained herein or for the furnishing, performance, or use of the material contained herein.

San Mateo, California
September, 1983

Chen-hanson Ting

CONTENTS

FIGURES

DIALECTIC VARIANCES

Almost all the programs presented in this notebook were written in the earlier version of poly-FORTH, developed by FORTH, Inc., released in Nov. 1979. Names of many instructions in this version of FORTH are different from the other dialects such as fig-FORTH, FORTH-79 Standard, and the latest version of poly-FORTH, i.e., poly-FORTH II.

For the convenience of readers who are more familiar with fig-FORTH or FORTH-79, I had prepared a short list on the variance between these three dialects. This list is by no means complete nor even exhaustive. It contains only the instructions I used very often in this Notebook. I felt obliged to warn the reader of these variances, so that proper modification can be made upon transporting programs to other FORTH systems.

DIALECTIC VARIANCES

poly-FORTH	fig-FORTH	FORTH-79
>IN	IN	>IN
ABORT" xxx"	ABORT	ABORT
AGAIN	REPEAT	REPEAT
---	AGAIN	AGAIN
BLANK	BLANKS	---
EMPTY	COLD	COLD
END	UNTIL	UNTIL
EXIT	;S	EXIT
BEGIN-IF-AGAIN	BEGIN-WHILE-REPEAT	BEGIN-WHILE-REPEAT
FLUSH	FLUSH	SAVE-BUFFERS
MINUS	MINUS	NEGATE

DIALECTIC VARIANCES

poly-FORTH	fig-FORTH	FORTH-79
MOVE	CMOVE	CMOVE
---	STATE	STATE
THEN	ENDIF	THEN
VARIABLE	n VARIABLE	VARIABLE
WORD (c -- addr)	WORD (c ---)	WORD (c ---)
[']	'	'

PROGRAM TRANSPORTABILITY

Most of the words are common to all FORTH systems and the FORTH programs are transportable between different systems. However, care must be exercised because of the differences in a few words. I would not expect readers to take the programs and blindly type them in their FORTH computer without looking at the contents carefully. It is probably better if one would question the way a word was defined and experiment with improvements and enhancement. It is alway a pleasure to shave off a few words in a definition, or to use a different approach to achieve the same goal while speeding up the execution.

THREE CHARACTER NAMES

Poly-FORTH had been criticized vehemently for retaining only the first three characters and the character length of word names. In using this naming convention for a number of years, I don't feel it is a major limitation as long as I have the freedom in choosing names. It is not difficult to find unique names for all the words in an application package. However, there were occasions that my tongue slipped and seemingly different names were not identified by the system as such. Strange things happen depending on the loading sequence of screens and the weather. Many hours were spent before realizing that there was a conflict in names. These things were part of the problems one would have to solve in the debugging processes.

The only instances that the three character names became a limitation were when I had to writen programs comforming to other people's naming schemes. Two examples in this Notebook fell into this category. In the programs on image processing, I tried to stick to the naming conventions used by De Anza, the IP manufacturer. Most register mnemonics were 6 characters long and the distinguishable characters were often in the fifth or the sixth character. In the microassembler program, many of the microcode fields definitions and operator mnemonics were indistinguishable in the first three characters. I had to doctor these names so that significant charaters were positioned in the first three charaters, making names not quite natural to those familiar with the original AMD literature. Only in these agonizing hours, I missed the 31 charater names in fig-FORTH.

GAMES IN FORTH

A good friend of mine, Mr. Li-wuu Wang, moved to the San Francisco Bay area from Los Angles. He was looking for a job. He asked me what he should do in the meantime. His major is in computer sciences. I suggested to him that he might find FORTH interesting and useful, and offered him my LSI-11 computer if he wanted to play with this language. After reading Starting FORTH and working out most of the exercises in the earlier chapters, he asked me what he should do with this language. I showed him a book, BASIC COMPUTER GAMES, edited by David H. Ahl, and asked him if he would translate or rework some of these games in FORTH. His agreed to try.

Mr. Wang spent many weekend on my computer, and literally glued his nose to the CRT screen. Most of the games collected here were his handiwork. Unfortunately, he landed a job with a local disk controller manufacturer and had to bid farewell to my computer. Otherwise, he might have just finished translating the whole book of games by now.

We can make some observations in comparing the same game written in FORTH with that in BASIC. First, the string processing capability of BASIC is quite extensive and it can handle conversation with the user very easily. Doing the same things in FORTH, one has to invoke system words like WORD and NUMBER. Second, in most of the games, integer arithmetic is sufficient and FORTH works well. If it requires anything at or above the level of square root, we have to go back and start building tools in FORTH.

Lastly, lengths of programs in FORTH are approximately the same as those in BASIC. The degree of readability of code is also the same in either language. Therefore, comparing individual programs, FORTH does not have much advantage over BASIC, as far as programming alone is concerned. However, many of the games require the same type of tools. If we build these tools once, they can be used in other programs. The real advantages in FORTH become obvious only by viewing all the games together.

In a few games, I also throw in a second version which deviates from the literal translation to take advantage of the features unique to FORTH. I hope these second versions will shed some light for the beginners and encourage them to dig deeper into the FORTH treasure box.

ANIMAL

This game was adapted from Basic Computer Games, p. 4, Ed. D. H. Ahl, attributed originally to A. Luehrman at Dartmouth College. This program is very interesting because animals are added to the program as the game is played and it can be adapted to other areas of interests.

ORIGIN	The starting block to store animal information.
K	A running index pointing to the line of message stored on disk.
KEY, EMIT	Standard terminal I/O instructions.
LINE	From the line number, return the address of message in disk buffer.
LAST	Fetch the total number of lines stored in the first cell in Block ORIGIN.

QUESTIONS AND ANSWERS

.QA	Print the message by its line number.
.ANIMAL	Print all the animals in the file.
YES	Wait for a key stroke. Return true if 'Y' is pressed.
QUESTIONS	Ask questions and wait on keyboard. If 'Y' is keyed, exit the loop and store the line number in K.
QUERY	Wait for a text line input from the keyboard.
INPUT	Ask the user to type in a question that would distinguish two animals.
NEW-ANIMAL	Get the new animal's name and add it to the file.
ANSWER	Get a YES/NO answer from the user and update the line of question, in which the first two bytes point to two lines for YES/NO branching.

FINAL LOOP

EXPAND	Move the K'th line to the end of file.
GUESS	Ask the user if the computer guessed the right animal. 'Y' indicates end of game. Otherwise, go the the routine to add a new animal to the file.
ANIMAL	The main game loop.

```
  135 LIST

( ANIMALS, CHT, 23-NOV-82)
140 CONSTANT ORIGIN
VARIABLE K
: KEY     0 'S 1 EXPECT ;
: EMIT    'S 1 TYPE DROP ;
: LINE    ( N --- A )    16 /MOD   ORIGIN + BLOCK
          SWAP 64 * +    ;
: LAST    ( --- N )      ORIGIN BLOCK @ ;
: .QA     ( N --- )      LINE 2+ 62 -TRAILING TYPE ;
: .ANIMALS ( --- )       CR ." ANIMALS I ALREADY KNOW ARE:"
          CR   0   LAST 0 DO
          I LINE @   0= IF   I .QA 3 SPACES   1+ THEN
          DUP 5 =    IF CR DROP 0 THEN   LOOP    DROP   ;
: YES    ( --- F )       KEY 89 = ;



  136 LIST

( ANIMALS, II, CHT, 23-NOV-82)
: QUESTIONS ( --- )
          1   BEGIN   DUP LINE @   IF CR
          DUP .QA   LINE   ." ?"   YES + C@   AGAIN   K !   ;
: QUERY   S0 @ 64 EXPECT   0 >IN !   ;
: INPUT   CR ." PLEASE TYPE IN A QUESTION THAT WOULD DISTINGUISH
 A "  CR   HERE COUNT TYPE   ."  FROM A "   K @ .QA   ."  :"
          QUERY ;
: NEW-ANIMAL   CR ." THE ANIMAL YOU WERE THINKING WAS A :"
          QUERY 1 WORD 1+   LAST 1+ LINE   0 OVER !
          2+ HERE C@  MOVE   UPDATE   ;
: ANSWER   CR ." FOR A "   HERE COUNT TYPE
          ."  THE ANSWER WOULD BE :"
          1 WORD 1+   K @ LINE 2+   HERE C@ MOVE   UPDATE
          LAST DUP 1+ YES 0= IF SWAP THEN   256 * +   K @ LINE
          !   2 0 LINE +!      UPDATE   ;


  137 LIST

( ANIMALS, III, CHT, 23-NOV-82)
: EXPAND   K @ LINE   LAST LINE   64 MOVE   ;
: GUESS    CR ." IS IT A "   K @ .QA   ." ?"
           YES CR   ABORT" --- TYPE 'ANIMAL' TO TRY AGAIN."
           NEW-ANIMAL   EXPAND   INPUT   ANSWER   ;
: ANIMAL   BEGIN   CR CR
           ." ARE YOU THINKING OF AN ANIMAL?"
           YES IF   QUESTIONS   GUESS
           ELSE   .ANIMALS   ( FLUSH )   THEN
           CR CR   0 END   ;
EXIT
```

PRINT THE FILE

PRINT Print the contents of the entire file. The first two bytes are pointers to other lines, and they are printed separately. Because of these two bytes, the file cannot be listed using the regular LIST command.

SOME COMMENTS

When I first took on the task of translating BASIC games into FORTH, I was very optimistic in that I could write the same game in much shorter codes, and really make the FORTH programs outshine their BASIC brothers. After a number of tries, I was not so sure of myself. The string commands in BASIC are very efficient in doing user interfacing. In comparison, I have to use QUERY, WORD, NUMBER, etc, to do the same thing.

The FORTH programs are not much better than the BASIC programs in length or in clarity. I am sure the FORTH programs will run much faster, but speed is not a very important concern in these games. I had to be satisfied in that I had done some examples to compare these two languages, without proving that FORTH is superior than BASIC.

```
  138 LIST

( ANIMALS, IV, CHT, 23-NOV-82)
: PRINT   LAST 0 DO   CR   I LINE DUP  C@ 3 U.R   1+ C@ 3 U.R
          I .QA   LOOP   ;
EXIT
```

```
  139 LIST
```

```
  140 LIST


DOES IT SWIM
DOES IT HAVE SCALES
DOES IT LIKE PEANUTS

IS IT STREAMLINED
FISH
```

CALENDAR

This program is adopted from BASIC COMPUTER GAMES, p. 37, attributed to Geoffrey Chase of Abbey, Portsmouth, RI. To print a year's calendar, you must specify the day of 1 Jan and the year.

MONTH>DAY Given the month (0-11), it returns the days in that month by looking up the table MONTH>DAYS above.

MONTHS A super string containing the names of the months.

STARS Print out a string of '*'.

TITLE Print the header of a month specified on stack.

SKIP-DATE Skip N columns in printing the first day of month.

WEEKS Print days of a month in the 7 column format.

NEXT-MONTH From day-of-year and month, calculate the day of 1st-of-month for next month.

MONTHS Given 1st-day-of-year, print out a 12 month calendar.

HEAD Print the header for the year calendar.
?LEAP Return true if processing a leap year.
PLUS-ONE Assign 29 days to Febuary in the leap year.
CALENDAR The final word doing all the processings to print out a year calendar. The day of 1 January and the year in AD must be given on the data stack.

96 LIST

```
( CALENDER )
CREATE  MONTH>DAYS  31 , 28 , 31 , 30 , 31 , 30 ,
                    31 , 31 , 30 , 31 , 30 , 31 ,

: MONTH>DAY    ( MONTH ---)    2 * MONTH>DAYS + @ ;

: MONTHS    ." JANUARY  FEBRUARY  MARCH    APRIL     MAY       JU
NE      JULY     AUGUST SEPTEMBER OCTOBER  NOVERBER DECEMBER" ;

: .MONTH    9 *   [']  MONTHS + 3 +   9   TYPE ;

: STARS    ( N --- )    0 DO 42 EMIT LOOP ;
```

97 LIST

```
( CALENDER )
: TITLE    ( MONTH --- ) 3 CRS  27 STARS  .MONTH  36 STARS CR CR
   8 SPACES ." S"  8 SPACES ." M"  8 SPACES ." T"  8 SPACES
   ." W"  8 SPACES ." T"  8 SPACES ." F"  8 SPACES ." S" CR CR
   72 STARS  CR CR ;

: SKIP-DATE    ( N --- )    DUP ?DUP   IF   9 * SPACES THEN ;

: WEEKS    ( D-Y D/M --- D-Y)  1+ 1 DO   I 9 U.R   DUP I + 7 MOD
                0= IF   CR CR   THEN   LOOP ;

: NEXT-MONTH    ( MONTH --- DAY)    MONTH>DAY   +   7 MOD ;
```

98 LIST

```
( CALENDAR     con't )
: MONTHS     ( 1ST-DAY-OF-YEAR ---, 0 FOR SUNDAY)
            12 0 DO I   TITLE   SKIP-DATE
            I MONTH>DAY WEEKS   I   NEXT-MONTH   LOOP DROP ;

: HEAD    ( Y ---)   20 CRS   25 SPACES   .   ." CALENDAR "   3 CRS ;
: ?LEAP   ( Y --- F)    4 MOD   0= ;
: PLUS-ONE    ( F ---)    IF 29 ELSE 28 THEN   2 MONTH>DAYS + ! ;
: CALENDAR    ( 1ST-DAY YEAR --- )
             DUP HEAD   ?LEAP   PLUS-ONE   MONTHS CR ;

6 1983 CALENDAR
```

CALENDAR, 2ND VERSION

The BASIC CALENDAR works fine. However, you have to look up which day is the Jan. 1 and tell the computer before it does the printing. This version can determine that day automatically.

JULIAN	The Julian day of the first day in the year. This Julian calendar starts at Jan. 1, 1949.
4YEARS	Total days in four years, including a leap year.
DAY	An array of the first day of each month, offset from Jan. 1.
DAY0	A copy of DAY, modified for leap year if the year to be processed is a leap year.
YEAR	Calculate the Julian day of Jan. 1 and prepare DAY0.
EMIT	Standard EMIT.
STARS	Print a string of stars.

PRINT ONE MONTH

HEADER	Print the header for the month identified on stack.
BLANKS	Skip columns according to the day of the first day in this month.
.DAYS	Print the days in a month in the column format.
MONTH	Print one month calendar.

PRINT CALENDAR

Give the year to CALENDAR and it will print the year's calendar, taking care of leap years and days all by itself.