LA-UR-88-3372

CONF 881111--1

LA-UR--88-3372

DE89 002289

TITLE MENU: A FORTH MENU COMPILER

AUTHOR(S) Kenneth B. Butterfield

SUBMITTED TO Tenth Annual FORML Cofnerence
Pacific Grove, California
November 25-27, 1988

Los Alamos Los Alamos National Laboratory
Los Alamos, New Mexico 87545

MASTER

FORM NO 836 R4
ST NO 2629 5/81

DISTRIBUTION OF THIS DOCUMENT IS UNLIMITED

MENU: A FORTH MENU COMPILER

Kenneth B. Butterfield
Los Alamos National Laboratory
P.O. Box 1663, MS J562
Los Alamos, NM 87545

ABSTRACT

Menu-driven command interpreters are an effective means of controlling portable instrumentation where input may be limited to a hex keypad and output to a few lines on a liquid crystal display. The MENU compiler uses the name fields from the words comprising the menu options as prompt strings. It produces compact program code, thus conserving memory that is often limited in ROM-based systems. Input for menu function selection is vectored to allow switching between input sources. The host system consists of the eight-bit Motorola M68HC11 processor and Max-FORTH (FORTH-83).

Using menus to select program control has proven to be both economical and straightforward. Programming for portable instrumentation has, as one of its main constraints, minimum facilities for interaction with the operator. Other limitations might include small memory size and the need to interface directly with hardware. Frequently a large portion of the program memory space is devoted to driving detectors and timers, which can leave little space for the user interface. At Los Alamos, the instruments we have designed and built have included a hex keypad and an LCD display, with lights and switches for communication with the user. While there are no large CRT displays and ASCII keyboards, or mice, to provide the windowing and point and click interactions currently in vogue, we have found a simple menu to be very useful.

Our MENU compiler (see Appendix A) is basically a fancy CASE statement that includes prompting for input by listing the available options and checking the operator's choice for validity. It is optimized to save memory space by using the name fields of the option words for prompt strings. A FORTH system that allows long names is essential to adequately prompt the user. (A prompt string composed of only the first three letters of a name would be ridiculous.) Range checking the operator's choice is included, even though it complicates the program, because portable instrumentation must be immune to operator error.

MENU has the usual three time frames associated with FORTH compilers, and the compile time frame for the compiler itself is straight forward. The only complication occurs because MaxFORTH hides the head of SP@. The time frame associated with compiling a menu provides the first unusual feature. In this time frame, the <BUILDS code is activated to put FORTH into the state normally associated with the colon (:) compiler. From this point on, tokens from the input string are converted into compilation addresses and stored, as usual, in the dictionary. This frame ends when

the semicolon (;) compiles a reference to EXIT. The last time frame is the run time of a menu. At this time the DOES> code from the menu compiler (1) determines from the list of compilation addresses the number of options, (2) lists the name fields of the options, (3) gets an input value, and (4) executes the selected word. The EXIT command is used by RANGE-CHECK to determine the end of the option list. The LIST-OPTION command uses a compilation address to point to the head of a word and prints the name field. Once a selection has been made, the compilation address passes to EXECUTE in the normal manner. If an invalid selection is made, program execution passes out of the menu with no further action taken.

One useful provision in MENU is the ability to vector the input to a variety of sources. We have used serial ports, key pads, and switches, either individually or in parallel, for input. Hardware interaction such as the timing out of a clock can be used to simulate input from the user as well. For example, the timeout can select 'Stop_Data_Collection' in a menu that also allows printing the time remaining and other data collection statistics. This type of menu permits monitoring of the collection process while interrupt driven data routines operate in a "background" mode. The input routine's main requirement is to return a value between 0 and N, where N is the number of options. The word GETKEY in the example (see Appendix A) shows one method for converting normal ASCII codes into the required range. GETKEY is a very simple word that would also map @, A...I into the range 0...9. The example also details the method for defining the input vector.

Converting MENU to other FORTH systems may reveal several system dependencies. The most likely dependency will probably be in the choice of words used to print the name fields of the options and the menu itself. This choice depends on the structure of the dictionary header associated with a word. MaxFORTH is a target compiler system that can produce headerless code. As part of this feature, the heads produced by MaxFORTH contain one extra pointer field (the parameter field address pointer) that modifies the arithmetic used in moving from the compilation address to the name field address. Other system dependencies will probably include name changes: Many systems use CREATE directly, rather than <BUILDS. The word compiled by a semicolon may be called semicolon S (;S) instead of EXIT. And the names for words used to move within a head may vary from one system to another.

Appendix B shows MENU in action. Note that the name of the menu and the names of the options available in the menu are displayed as part of the overall operator prompt. The name of the option selected is printed to help document program path selection. These names are long because they serve as part of the user interface and need to be clear and descriptive. Another aspect of using menus is illustrated in Print_Stack, which includes the loop termination as one of the available options. One point not illustrated in the example, but worth mentioning, is that a menu can serve as one option in another menu making it easy to build a tree structure of control operations.

Thus far, we have found the menu compiler presented in this paper to be an excellent user interface for portable instrumentation. It is easy to use and conserves memory in systems that may have limited resources. MENU is probably not as user friendly as a full-windowing, pop-down user interface, but it certainly retains the simplicity exhibited by the FORTH language.

APPENDIX A: MENU WORDS

```
VARIABLE MENU_VECTOR
HEX

: GETKEY KEY F AND ;

: KEY_INPUT
   [ ' GETKEY CFA ] LITERAL
   MENU_VECTOR !
;

KEY_INPUT

VOCABULARY MENU
MENU DEFINITIONS

: LIST_OPTION ( CFA  ... / List option in a menu
   2- NFA SPACE ID. ;

: RANGE_CHECK ( ADD ... MAX ADD   / Determine upper limit
   DUP
   BEGIN
      DUP @
      [ ' EXIT CFA ] LITERAL
      = NOT
   WHILE 2+
   REPEAT
   SWAP
;

: LIST_OPTIONS ( MAX ADD ... MAX ADD / List options )
   2DUP
   DO I OVER - 2/ CR .
      I @ LIST_OPTION
   2 +LOOP
;

: DO_N ( MAX ADD N ...   EXECUTE NTH WORD IN MENU
  2* + SWAP
  OVER - 0> ( over range check )
  IF
     @
     DUP LIST_OPTION CR
     EXECUTE
  ELSE DROP THEN
;

: PROMPT ." Press key to select " ;

: MENU_INPUT ( Get a value from 0 to n -- vectored)
   MENU_VECTOR @ EXECUTE
;
```

```
FORTH DEFINITIONS
: MENU ( DEFINES A MENU )
       ( USE:
       (    MENU NAME W1 W2 W3 ... WX ; )
       (    when NAME is executed the ID's of
       (       W1 ... WX  are displayed,
       (       a key is read and the corresponding
       (       word is executed )
   [ MENU ] ( MENU VOCABULARY is hidden after this definition )
   <BUILDS         ( the following compiles   SPS ] )
      [ EC48 , ]   ( SPS is headerless in MAX-FORTH )
      [COMPILE] ]  ( ] IS IMMEDIATE in MAX-FORTH )
   DOES>
      DUP 6 - NFA ( find name field - system dependent )
      CR ID. 4 SPACES PROMPT
      RANGE_CHECK LIST_OPTIONS
      MENU_INPUT
      DUP 0< NOT ( under range check )
      IF DO_N ELSE DROP THEN
;
```

APPENDIX B: Example of MENU definition

```
: Base_10  BASE @ DECIMAL .8 BASE ! ;
: Base_16  BASE @ HEX .8 BASE ! ;
: Base_2   BASE @ 2 BASE ! .8 BASE ! ;
: Base_8   BASE @ 8 BASE ! .8 BASE ! ;
: Exit_Menu  DROP -1 ;

MENU Print_Bases
   Base_2
   Base_8
   Base_10
   Base_16
   Exit_Menu
;

: BASES
   KEY_INPUT
   BEGIN
      0
      Print_Bases ( top 2 numbers = base, 0 )
   UNTIL
;


( SAMPLE INTERACTION )

12 55 76 123 C8

BASES
Print_Bases    Press key to select
0  Base_2
1  Base_8
2  Base_10
3  Base_16
4  Exit_Menu Base_2

10000
0
11001000
100100011
1110110
1010101
10010
```

```
Print_Bases    Press key to select
0  Base_2
1  Base_8
2  Base_10
3  Base_16
4  Exit_Menu Base_8

20
0
310
443
166
125
22
Print_Bases    Press key to select
0  Base_2
1  Base_8
2  Base_10
3  Base_16
4  Exit_Menu Exit_Menu
OK
```