9820 USER DEFINABLE FUNCTION ASSEMBLY

The calculator's User Definable Functions Accessory (Figure 10A-10E) includes a 1024 bit plug in ROM with three 10 key overlays. Five keys are used for control and that leaves 25 keys available for definition if no other plug-in ROMS are in the machine. 15 keys if one other plug-in ROM is used and 5 if both other ROM slots are used. The uses of this accessory are described below.

In a program it frequently happens that some basic calculation is needed at several different places. It is clumsy, wasteful and error prone to duplicate the necessary statements each time they are needed. It is easier and more desirable to write them once and refer to the statements as the calculation is required. This capability is provided by subroutines and functions. Here we describe the basic subroutine and function capabilities of the calculator and how they are extended with the USER DEFINABLE FUNCTIONS accessory.

The calculator has basic subroutine capabilities provided by the GO SUB and RETURN keys. These keys allow one or more lines in the main program to be called as a subroutine by jumping to the first line with the GO SUB statement and returning to the main program by executing a RETURN statement. For example, it may be necessary to set the first ten R register to zeros at several places in the program. This job can be accomplished with the following program using a subroutine labeled "ZERO" as follows:

```
 0:  GO SUB "ZERO"  )
        :           |
20:  GO SUB "ZERO"  |      Program
        :           }      with three
35:  GO SUB "ZERO"  |      calls to "ZERO"
        :           )
```

```
40:  "ZERO"                                  }
41:  10 → Z                                  }  Subroutine
42:  Z - 1 → Z; 0→RZ; IF Z>0; GTO +0         }    zero
43:  RETURN                                  }
```

The calls to "ZERO" from lines 0, 20 and 35 cause the ten R registers to be cleared before returning to lines 1, 21, and 36, respectively. The usage of subroutine "ZERO" clearly saves space since the code in lines 41 through 43 need not be duplicated. In addition, as the program is segmented into subroutines it becomes easier to read and understand. If the subroutine is useful to others, it may be incorporated in their programs not only to save space but to save time writing their programs.

In the simple example, subroutine "ZERO" always does exactly the same job: setting the first ten R registers to zero. A more general subroutine would have the capability to set any ten consecutive R registers to zeros starting at R(J). To accomplish this, the subroutine must be altered and the value of the parameter J must be known by (or passed to) the subroutine. This value could be stored in the X register before calling the subroutine and the program could be changed as follows:

```
 0:  0→X; GO SUB "ZERO 1"
            :
20:  40→X; GO SUB "ZERO 1"
            :
35:  30→X; GO SUB "ZERO 1"
            :
40:  "ZERO 1"
41:  10 → Z
42:  Z - 1 → Z; 0 → R(X+Z), if Z>0; GTO + 0
43:  RETURN
```

The subroutine "ZERO 1" clears R registers 0-9, 40-49 and 30-39 in lines 0, 20 and 35, respectively. The programmer must be careful, however, since the subroutine uses both the X and Z registers. These registers must be saved if they contain valued information when the subroutine is called. This bookkeeping complicates using the subroutine and makes it less attractive and more conducive to errors. The problems become even worse as more parameters must be passed to the subroutine and as more working registers, such as Z, must be made available. These problems are circumvented by using advanced features found in the USER DEFINABLE FUNCTION ROM.

In addition the USER DEFINABLE ROM includes the concept of a function. A function differs from a subroutine in that the name of a function has a value associated with it. Therefore, function names can appear in any arithmetic expression to reference the value associated with the functions such as the names A, B, C, X, Y, Z, and R are used for registers. For example,

SIN, COS, LN and EXP

are functions which have values associated with their names and

$$\text{SIN (LN A)} - \text{COS (EXP B)} \rightarrow \text{X}$$

is a valid arithmetic statement containing several functions.

While some standard functions are built in to the calculator, it is desirable to be able to define other functions and have them work in the same manner that the functions sin, cos, ln, exp, etc. work. For example, if a solution of a problem required hyperbolic functions, it would be desirable to define the functions and write statements like

$$\text{SINH}(\text{A} + \text{B}) - \text{COSH } (\text{A} - \text{B}) \rightarrow \text{X}$$

4

The problems encountered in defining functions are similar to those of writing subroutines. Parameters of functions (arguments) must be known by or passed to the function and the working registers must be made available to the function so temporary results may be stored during the calculations. Defining functions differ from defining subroutines in that the value must be assigned to the function. The USER DEFINABLE FUNCTION block provides capabilities to solve these problems.

The option block has key arrangements as shown in Figures 10C-10E. Keys FA, FB, FC, FD and FE are assignable to any five subroutines or functions. GA through GJ and HA through HJ are also assignable in the absence of one or two other ROM blocks extending the capacity to 15 or 25 functions or subroutines. The remaining five keys facilitate the defining and calling these functions and subroutines.

Subroutines and functions that are defined with the USER DEFINABLE FUNCTION block are similar in structure to the main program: each routine is a list of one or more statements, numbered from zero, followed by an END. To define a simple subroutine to calculate the volume of a sphere and assign this subroutine to the FA key. First press

GTO FA EXECUTE

This places the machine in define subroutine mode related to key FA. Any other assignable key could be used in place of FA. Next, to define the subroutine for calculating the volume, STORE

```
0:  4/3*πZZZ→Z
1:  END
```

Storing the END returns the machine to the normal mode of operation. To use this subroutine to calculate the volume of a sphere with radius 5, press

5; FA EXECUTE

which is equivalent to

5→Z; GSB FA EXECUTE

The Z register is displayed. To call the subroutine from a program STORE

3: 5→Z; GSB FA

The five control keys (left keys of Figure 10C) extend these basic subroutine capabilities to include immediate execute as well as parameter passing and function subprograms. These keys are described below.

IMMEDIATE EXECUTE. The immediate execute key (displayed as IEX) is used in making the calculator respond immediately to the depression of a key without pressing EXECUTE. The IEX must be the first statement of the subroutine for the key to respond in this manner. When the key associated with such a subroutine is depressed, the routine is executed without pressing EXECUTE. For example, if the previous program was changed to

```
0:  IEX
1:  4/3*πZZZ→Z
2:  END
```

then merely press

5 FA

to invoke the routine. This specialized execution mode is valuable in simplifying keyboard operation to increase efficiency and productivity when moving similar calculations must be made from the keyboard.

CALL. To call a subroutine with parameters the CLL must be used. This key is used to indicate that a list of parameters will follow the subroutine name. Otherwise, the key is used exactly as GSB key. That is,

```
GSB FA              (no parameters)
CLL FA (A, 5, B+X)     (parameters)
```

The CLL statement should be the last statement of a line. The parameters need not be enclosed in parentheses.

PARAMETER. The P() or parameter key is used to access parameters that are being passed to subroutines and functions and is probably the most heavily used key of this ROM block. In addition to accessing parameters, the P() key may be used to create and access memory that is used temporarily as working registers while the subroutine is being executed. Accessing parameters and working registers is done with the P() key without affecting the A, B, C, X, Y, Z, or R registers.

The P() key is used exactly like the R() key but it references a sequence of parameters registers instead of the R registers. For example, if a subroutine FB is called with three parameters, P1 references the first parameter, P2 references the second, etc. That is,

```
CLL  FB (A,  5,  X-B)
         ↓   ↓    ↓
         P1  P2   P3
```

In this CLL, P1 references the A register, P2 and P3 reference memory locations where 5 and the value of X-B are stored temporarily during the execution of subroutine FB. The calculation of X-B is made and placed in a temporary location each time the CLL statement is executed before executing subroutine FB.

Temporary working registers may be created and accessed by using the P() key with subscripts higher in value than the number of parameters being passed. For example, subroutine FB had three parameters (P1, P2, P3). P4, P5, etc. could be used as working registers. Obviously, the number of such registers is limited since the calculator will run out of internal temporary storage eventually. An exact limit cannot be given since it is dependent on the availability of memory when the subroutine is initiated.

As the first example, consider rewriting subroutine "ZERO 1" to zero the specified ten R registers without destroying the value of the X or Z registers as the previous routine did. One parameter P1 must be passed replacing X and one working register P2 is used in place of Z. The necessary statements follow.

```
    PRESS   GTO FA EXECUTE
    STORE   0:  "ZERO 1 "
            1:  10 → P2
            2:  P2-1→P2; 0→R(P1 + P2); IF P2>0; GTO+0
            3:  END
```

Then, CLL of the form

```
            40→X; GO SUB "ZERO 1"
```

are replaced pressing

```
            CLL FA 4 0 STORE
```

which is displayed as

```
            20:  CLL ZERO 1 40
```

since the subroutine is started with the label "ZERO 1". The new routine operates as prescribed without destroying the values of either register X or Z freeing them for other purposes.

Another example is a routine to increment a register. The one parameter of this subroutine specifies the register to be incremented:

```
PRESS         GTO FC EXECUTE
STORE     0:  "INCR "
          1:  P1+1→P1
          2:  END
```

INCR may be called by

```
          10:  CLL INCR A
```

to increment the A register or

```
          20:  CLL INCR R(A+B)
```

to increment the R register specified by A+B. This example shows that a parameter may be used to return a result as well as access a value. Any number of parameters may be used in calling a subroutine.

DEFINE. A function differs from a subroutine in that it has a value associated with its name and, therefore, can be part of an expression. The DEF/→F key allows functions to be defined in the calculator. The key has two uses as its label indicates. First, it is used to place the machine in function definition mode DEF. Secondly, once the calculator is in function definition mode, it is used to assign a value to the function →F.

To place the calculator in <u>function</u> <u>definition</u> mode,

```
PRESS     DEF FA EXECUTE
```

This is analogous to placing the machine in subroutine definition mode; that is pressing

```
          GTO FA EXECUTE.
```

After placing the machine in function definition mode, the function is defined exactly as a subroutine with parameters except the →F allows a value to be assigned to the function.

As an example consider writing a function to define the hyperbolic sin function.

$$\text{Sinh } X = \frac{e^x - e^{-x}}{2}$$

as the FD key. First, to place the calculator in function definition mode,

```
        PRESS   DEF FD EXECUTE
```

To define the sinh function,

```
        STORE   0:  " SINH "
                1:  (EXP P1 - EXP(-P1) )/2→F
                2:  END
```

To use the function, the FD key is referenced just like the SIN key. For example,

```
        PRESS  FD ( 5 ) + FD ( 4 ) EXECUTE
```

which is displayed as

```
                SINH ( 5 )+ SINH ( 4 )
```

before EXECUTE is pressed since the definition begins with the label "SINH". Similarly,

```
                5:  SINH ( A+B ) / SINH ( A-B ) → A
```

can be stored as a program line. The machine truly behaves as if it had a "built in" capability to calculate hyperbolic sines.

As a second example, the maximum value function is programmed. This function has two parameters and is assigned the value of the larger of the two parameters. First,

```
        PRESS    DEF FE EXECUTE
and     STORE    0:  "MAX"
                 1:  P1 → F
                 2:  IF P2>P1; P2→F
                 3:  END
```

Notice that P1 is assumed to be the larger of the two parameters in line 1, and line 2 makes a correction if this is not the case. This function can be used to calculate and store the product of two maximum values as follows:

MAX (6, 9) MAX (-5, -4) → RA

or MAX (AB - C, 5) MAX (Z↑3, 5-A) → RC

Performing similar operations without using this function capability would require several registers to store intermediate results and would be very hard to read and understand in comparison.

SCRATCH. The SCR is used for several functions. Its primary use is to delete a user defined subroutine or function from memory to allow a key to be used for other programs or to increase the amount of memory available for the main program. To delete function FA,

PRESS SCR FA EXECUTE

To delete two (or more),

PRESS SCR FB, FC EXECUTE

Special functions of this key included recording and loading of programs. To record all programs in memory in the order stored,

PRESS GTO SCR; REC EXECUTE

To load these programs,

PRESS GTO SCR; LOD EXECUTE

To record one function or subroutine per one half card, place one subroutine or function in the machine and

PRESS GTO FA (OR OTHER KEY DEFINED); REC EXECUTE

Similarly, to configure a machine from a library created in this manner, order the functions and subroutines and PRESS

GTO FA; LOD EXECUTE

GTO FB; LOD EXECUTE

ETC.

To list function FA,

PRESS GTO FA LIST

The ability to configure the calculator in this manner makes it possible to customize the calculator from one problem to the next without reprogramming, entering and debugging the functions and subroutines needed. This ability combined with the capability of the calculator to modify the keyboard with a variety of plug-in ROMS allows versatility never before found in a calculator.

In summary, the USER DEFINABLE FUNCTIONS ROM for the calculator greatly extends the capabilities of the calculator. It has been shown how the block is used to write general purpose subroutines and functions. These routines communicate with the main program by parameter passing and allow working registers within the subroutine to be established and accessed. These features allow the user to define routines that do not require or destroy the content of the A, B, C, X, Y, Z and R registers. Therefore, the programmer is relieved of all the bookkeeping that is associated with calling a subroutine when parameters must be placed in specified registers; these registers usually have to be saved before storing parameters and restored after calling the subroutine. The programs written with this required bookkeeping become clumsy, obscure, hard to debug, and in general discourages the use of subroutines and functions.

Another advantage of the USER DEFINABLE FUNCTIONS ROM is its ability to define functions (subroutines that have a value associated with their names such as SIN and LN) that exactly imitate the behavior of the built in functions of the calculator. This allows the capabilities of the machine to be extended naturally when a problem that is based on different functions is encountered.

The option block also allows a library of general purpose subroutines and functions to be established and used easily. This ability greatly emancipates the programmer by allowing him to borrow something written by another with a minimum of effort.

In general, the USER DEFINABLE FUNCTIONS extend the capabilities of the calculator to make the machine easier and more natural to program. It may be the user's most valuable addition to the calculator.

Figure 10A

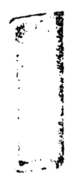
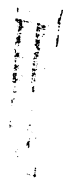


Figure 10B



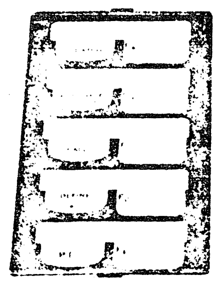


Figure 10C

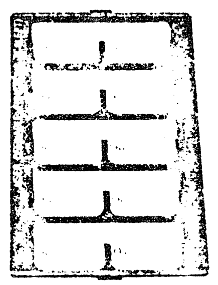


Figure 10D

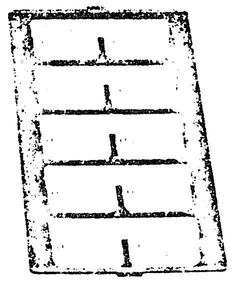


Figure 10E