SECTION XI

Communication with Non-BASIC Programs

A BASIC/3000 user can access an SPL/3000 procedure or a FORTRAN/3000 subprogram from a BASIC program with the CALL statement. As mentioned in Section I, he can enter MPE/3000 with the SYSTEM command or by pressing the break key. Another method for using the facilities of the MPE/3000 Operating System is with the SYSTEM statement. The SYSTEM statement enters an MPE system command dynamically into a BASIC/3000 program.

CALL Statement

The CALL statement calls for execution of an SPL/3000 procedure or a FORTRAN/3000 subprogram that exists in an MPE/3000 segmented procedure library (SL). Parameters may be passed to the called procedure or subprogram. The search for an external parameter begins in the group library. (See MPE/3000 Operating System, Reference Manual.

Care should be taken when using SPL or FORTRAN since these procedures are not controlled by the BASIC/3000 Interpreter and errors in them can propagate into the Interpreter causing indeterminate results.

Form

The forms of CALL are

CALL procedure name

CALL procedure name(actual parameter list)

** procedure name*

** procedure name(actual parameter list)*

The *procedure name* identifies the procedure or subprogram being called. The optional actual parameter list may contain numeric or string expressions, variables and array names followed by bound indicators (*) or (*,*). Anything that can be passed to a function (see Section VI) can be passed in a CALL statement.

The asterisk may be used interchangeably with CALL.

Explanation

CALL transfers control to the beginning of the specified procedure. If parameters are specified, they are passed by reference as address pointers rather than by value. If the parameter is an expression, a temporary variable is created by the Interpreter to contain the value of the expression.

In addition to the parameter addresses, BASIC/3000 passes the number of parameters and also code words describing the parameters (one code word for every three parameters). The code words may be used by the procedure to insure that the calling sequence is correct.

The parameter information used by the program or procedure being called is set up by the BASIC/3000 Interpreter when the CALL statement is executed. Depending on the called program, values may be returned to the BASIC program through the specified parameters. The formats of the parameter address table and of the parameter values is contained in Appendix F, Parameter Format. This information is useful primarily if the BASIC user is writing or modifying the called program. Otherwise, he only needs to know the type and the order of the parameters used in the called program in order to specify them in the CALL statement.

Examples

The first example calls the FORTRAN segment BOOLEAN containing three subroutines that perform Boolean operations. The first subroutine LAND performs a logical AND, the second LOR performs a logical OR, and the third LNOT performs a logical NOT. Each subroutine is called with a different CALL statement and the result is printed upon return to BASIC.

The second example calls the SPL procedure WHOM from a BASIC program. WHOM returns the user's name, group, accout, and home group from the identification codes entered by the user when he logs on. This information is derived by WHOM using the MPE/3000 system intrinsic WHO. WHO is an option variable procedure and, as such, cannot be called directly from BASIC/3000. This example illustrates, among other things, the interface with an intrinsic that cannot be referenced directly.

Each subroutine and procedure is listed following the BASIC program that calls it, and this list is followed by an SL (segmented procedure library) list requested after the segment is added to the SL.

1. Calling FORTRAN Subroutines

```
  10 INTEGER X,Y,Z
  20 LET X=1,Y=0
  30 *LOR(X,Y,Z)
  40 PRINT Z
>RUN
 1


  10 INTEGER X,Y,Z
  20 LET X=1,Y=0
  30 CALL LAND(X,Y,Z)
  40 PRINT Z
>RUN
 0


  10 INTEGER X,Y,Z
  20 LET X=1,Y=0
  30 CALL LNOT(Y,Z)
  40 PRINT Z
>RUN
-1
```

The FORTRAN segment containing subroutines LAND, LOR, and LNOT is listed below. The comment lines describe the parameter restrictions. Note that all three parameters should be integers.

The user should refer to the FORTRAN/3000 Reference Manual for instructions on writing a FORTRAN subprogram.

```
$CONTROL  USLINIT, NOLIST, SEGMENT=BOOLEAN
      SUBROUTINE  LAND(A,B,RESULT)
C         PERFORMS BOOLEAN-AND ON "A" AND "B", RETURNING RESULT IN
C         "RESULT".  "A" AND "B" MUST BE TYPE INTEGER SIMP. VAR. OR
C         EXPRESSION; "RESULT" MUST BE INTEGER SIMP. VAR.(SUBSCRIPTED
C         OR NOT).
        LOGICAL A,B,RESULT
        RESULT = A .AND. B
        RETURN
      END
$CONTROL  SEGMENT=BOOLEAN
      SUBROUTINE  LOR(A,B,RESULT)
C         PERFORMS BOOLEAN-OR ON "A" AND "B", WITH RESULT RETURNED IN
C         "RESULT".  PARAMETERS ARE SAME AS FOR "LAND".
        LOGICAL A,B,RESULT
        RESULT = A .OR. B
        RETURN
      END
$CONTROL  SEGMENT=BOOLEAN
      SUBROUTINE  LNOT(A,RESULT)
C         PERFORMS BOOLEAN-NOT ON "A", RETURNING RESULT IN "RESULT".
C         "A" AND "RESULT" ARE SAME AS FOR "LAND".
        LOGICAL A,RESULT
        RESULT = .NOT. A
        RETURN
      END
```

After the FORTRAN segment BOOLEAN is compiled, it is added to the SL (Segmented Library) using the Segmenter. An SL list is then requested that shows the segment BOOLEAN as the only segment in the SL of the user's group/account STUDENTS.CLASS:

```
:FORTRAN  BOOLEAN

PAGE 0001    HEWLETT-PACKARD 32102A.00.2    FORTRAN/3000      TUE, MAR 27,
             1973,  5:55 AM


00200000  $CONTROL  USLINIT, NOLIST, SEGMENT=BOOLEAN

END OF PROGRAM


 :SEGMENTER

 SEGMENTER SUBSYSTEM (1.2)
 -USL $OLDPASS
 -SL SL
 -ADDSL BOOLEAN
 -LISTSL

 SL FILE SL.STUDENTS.CLASS

 SEGMENT    0 BOOLEAN          LENGTH     30

    ENTRY POINTS     CHECK CAL STT  ADR
    LOR             141400    C     2      6
    LNOT            141000    C     1      0
    LAND            141400    C     3     15

    EXTERNALS         CHECK STT SEG

 1

 USED                   2200          AVAILABLE          375600

 -EXIT

 END OF PROGRAM
```

Notice, the -SL SL command works only if the SL exists. To build an SL enter the command:

BUILDSL SL[.group], records, extents

then enter the -SL SL command.

Library usage is described in the MPE/3000 Operating System, Reference Manual.

2. Calling an SPL Procedure

The SPL procedure WHOM is called by the BASIC program ZELDA. WHOM returns the user name, group, and account with which the user logged on, as well as the user's home group. The BASIC program uses this information plus the current time and date from the DAT$ function to print output as if from Zelda, the fortune telier.

The username, account, group, and home group are stored in string variables used as actual parameters in the call to WHOM.

The BASIC program:

```
ZELDA
  10 D$=DAT$(1,27)
  20 MAT READ M$
  30 CALL WHOM(U$,G$,A$,H$)
  40 REM
  50 REM
  60 PRINT LIN(2);"I'M ZELDA THE FORTUNE TELLER,"
  70 PRINT "I CAN TELL YOU A LOT."
  80 FOR I=1 TO 11
  90   IF POS(M$[I],D$[6;3]) THEN 110
 100 NEXT I
 110 PRINT LIN(1);"ON THIS ";M$[I];" ";DEB$(D$[10;2]);FNC$(D$[10;2]);","
 120 PRINT "IT'S ";DEB$(D$[20;5]);" O'CLOCK."
 130 PRINT LIN(1);"YOUR ACCOUNT NAME IS "'34;DEB$(A$);'34","
 140 PRINT "AND YOU'RE IN GROUP "'34;DEB$(G$);'34"."
 150 PRINT LIN(1);"YOUR SIGN-ON WAS "'34;DEB$(U$);'34","
 160 IF LEN(DEB$(H$)) THEN DO
 170   IF G$<>H$ THEN PRINT "AND "'34;DEB$(H$);'34" IS YOUR HOME."
 180   ELSE PRINT "AND YOU'RE IN YOUR HOME GROUP."
 190 DOEND
 200 ELSE PRINT "AND YOU HAVE NO HOME GROUP."
 210 PRINT LIN(1);"BUT ENOUGH SAID FOR NOW.";LIN(2)
 220 REM
 230 REM          ********************
 240 REM
 250 DEF FNC$(C$)
 255   IF C$="11" OR C$="12" OR C$="13" THEN RETURN "TH"
 260   IF C$[2]="1" THEN RETURN "ST"
 270   IF C$[2]="2" THEN RETURN "ND"
 280   IF C$[2]="3" THEN RETURN "RD"
 290   RETURN "TH"
 300 FNEND
 310 DIM D$[27],U$[8],A$[8],G$[8],H$[8],M$[12,9]
 320 INTEGER I
 330 DATA "JANUARY","FEBRUARY","MARCH","APRIL","MAY","JUNE","JULY"
 340 DATA "AUGUST","SEPTEMBER","OCTOBER","NOVEMBER","DECEMBER"
```

In these examples the user's name is JOHNDOE, his home group is STUDENTS, and his account is CLASS. When run under the group STUDENTS, the result is:

```
I'M ZELDA THE FORTUNE TELLER,
I CAN TELL YOU A LOT.

ON THIS MARCH 30TH,
IT'S 4:35 O'CLOCK.

YOUR ACCOUNT NAME IS "CLASS",
AND YOU'RE IN GROUP "STUDENTS".

YOUR SIGN-ON WAS "JOHNDOE",
AND YOU'RE IN YOUR HOME GROUP.

BUT ENOUGH SAID FOR NOW.
```

When ZELDA is run under the group PUB, the result is:

```
I'M ZELDA THE FORTUNE TELLER,
I CAN TELL YOU A LOT.

ON THIS MARCH 30TH,
IT'S 4:30 O'CLOCK.

YOUR ACCOUNT NAME IS "CLASS",
AND YOU'RE IN GROUP "PUB".

YOUR SIGN-ON WAS "JOHNDOE",
AND "STUDENTS" IS YOUR HOME.

BUT ENOUGH SAID FOR NOW.
```

The SPL procedure WHOM, compiled into segment WHOMSEG, is listed below:

```
$CONTROL    SEGMENT=WHOMSEG
$CONTROL    USLINIT, NOLIST, SUBPROGRAM
BEGIN
PROCEDURE WHOM(USERNAME,LOGONGROUP,LOGONACCT,HOMEGROUP);
          BYTE ARRAY USERNAME,LOGONGROUP,LOGONACCT,HOMEGROUP;
  BEGIN
    EQUATE CODES3=[6/1,5/1,5/1],      <<CODEWORD FOR THREE STRINGS>>
           CODE1=[6/1,10/0];       <<CODEWORD FOR ONE STRING>>
    INTEGER DELTAQ=Q-0;       <<DELTA-Q IN STACK MARKER>>
    INTEGER POINTER NUMBER;       <<WILL POINT TO # OF PARAMS PASSED.
                                    "NUMBER(1)" AND "NUMBER(2)" ARE
                                    CODEWORDS PASSED BY INTERPRETER  >>
    INTRINSIC WHO;
      <<>>
    @NUMBER:=@DELTAQ-DELTAQ+1;
    IF  NUMBER=4 AND NUMBER(1)=CODES3 AND NUMBER(2)=CODE1  THEN
      IF  USERNAME(-2)>=8 AND LOGONGROUP(-2)>=8 AND LOGONACCT(-2)>=8
          AND HOMEGROUP(-2)>=8      <<"...(-2)" IS PHYSICAL LENGTH>>
        THEN BEGIN
          WHO(,,,USERNAME,LOGONGROUP,LOGONACCT,HOMEGROUP);
          USERNAME(-1):=LOGONGROUP(-1):=LOGONACCT(-1):=HOMEGROUP(-1):=8;
                                      <<SET LOGICAL LENGTH OF STRINGS>>
        END;
    RETURN 0;              <<IN CASE OF ERROR IN CALLING SEQUENCE, LET
                             INTERPRETER CLEAN UP STACK                >>
  END;  <<PROCEDURE WHOM>>
END.
```

The WHO instrinsic called by WHOM is an option variable procedure provided by MPE/3000. It cannot be called directly from BASIC/3000, but can be accessed indirectly, as in this example, through an SPL procedure.

The SPL procedure verifies the calling sequence by first checking the number of parameters, then by looking at the type codes in the code words. This routine verifies that the physical length of the strings is at least large enough to contain the strings returned by WHO.

When called by WHOM, WHO returns the log-on user name, the account and group names, and the user's home group. WHOM passes this information to the BASIC calling program ZELDA after setting the logical length of the strings returned by WHO to 8.

The RETURN 0 (rather than RETURN) leaves the parameters in the stack in case there was some error in the calling sequence. Whenever there is a possibility of an error in the calling sequence, RETURN 0 should be used to exit from an SPL procedure.

The user should consult the SPL/3000 Reference Manual for instructions on writing an SPL procedure.

The file WHOMPROG contains the source of WHOM. WHOM is compiled into the segment WHOMSEG. After WHOM is compiled it is added to the SL (Segmented Library) using the Segmenter. It is then listed, showing entry points, external procedures, its length, and so forth.

The command :SPL WHOMPROG requests compilation of WHOMPROG:

```
:SPL  WHOMPROG

PAGE 0001    HP32100A.02.0


00200100  00000 0    $CONTROL    SEGMENT=WHOMSEG
00200200  00000 0    $CONTROL    USLINIT, NOLIST, SUBPROGRAM
 PRIMARY DB STORAGE=%000;    SECONDARY DB STORAGE=%00000
 NO. ERRORS=000;             NO. WARNINGS=000
 PROCESSOR TIME=0:00:02;     ELAPSED TIME=0:00:30

END OF PROGRAM
```

The command :SEGMENTER and subsequent commands add the segment WHOMSEG to the account SL called SL.PUB, and then lists the SL:

```
:SEGMENTER

SEGMENTER SUBSYSTEM (2.0)
-SL SL.PUB
-USL $OLDPASS
-ADDSL WHOMSEG
-LISTSL

SL FILE SL.PUB.CLASS

SEGMENT    0 WHOMSEG          LENGTH    100

    ENTRY POINTS     CHECK CAL STT  ADR
    WHOM               0    C    1     0

    EXTERNALS        CHECK STT SEG
    WHO                0     2   ?

1

USED                  1600          AVAILABLE              27200


-EXIT

END OF PROGRAM
```

For SL library usage, see the MPE/3000 Operating System, Reference Manual.

SYSTEM/RESUME Commands

The SYSTEM command is used to enter control of the MPE/3000 Operating System. The user's activity in BASIC/3000 is suspended and may be resumed with the RESUME command.

Form

The form of SYSTEM is:

SYSTEM

The form of RESUME is:

RESUME

Like all BASIC/3000 commands, the prompt for SYSTEM is >. The prompt for RESUME is : since RESUME is entered from MPE.

Explanation

When SYSTEM is typed, BASIC is suspended and the user enters MPE/3000. A colon (:) is output as a prompt signal and he may then type any MPE commands. The BREAK key may also be used to suspend BASIC. Usually there is no difference between the BREAK key and the SYSTEM command. However, when using BASIC at a terminal that does not have a BREAK key, the SYSTEM command is the only way to enter MPE without terminating BASIC.

When through with MPE, the user returns to the suspended BASIC operation by typing RESUME.

NOTE: The SYSTEM COMMAND has no effect when running a program in batch mode.

Example

```
>10 DIM A$(72)
>20 READ A1,A2,A3,A4,A5
>SYSTEM

:BUILD AA;REC=-72,,,ASCII;DISC=100
:RESUME
>30 PRINT #1,1;A1,A2,A3,A4,A5
>40 RESTORE #1
>50 LINPUT #1;A$
>60 PRINT A$
>70 DATA 10,20,30,40,50
>80 FILES AA
>RUN
 10              20              30              40              50
```

In this example, the user leaves the BASIC/3000 Interpreter to create an ASCII file. This can only be done with the MPE BUILD command. After creating the ASCII file, he returns to BASIC with RESUME and uses the ASCII file.

SYSTEM Statement

The SYSTEM statement provides a means to dynamically enter an MPE/3000 system command from a BASIC/3000 program. The command will be executed at the position in the program occupied by the SYSTEM statement.

Form

SYSTEM numeric variable,system command

The *system command* is specified as a string expression. The initial colon is not included in the command specification.

The *numeric variable* returns 0 if the command succeeds, or the MPE command error number if the command fails.

Explanation

When a system command is required within a BASIC program, the SYSTEM statement can be used to execute such a command during execution of the BASIC program.

Example

```
 10 REM..USE SYSTEM STATEMENT TO ENTER MPE COMMAND
 20 SYSTEM X,"FILE ABC;DEV=TAPE"
 30 IF X<>0 THEN 120
 40 FILES A
 50 DIM X[10]
 60 MAT READ X
 70 DATA 10,20,30,40,50,60,70,80,90,100
 80 MAT PRINT #1;X
 85 ASSIGN *,1
 90 SYSTEM Y,"STORE A;*ABC"
100 IF Y<>0 THEN 120
110 END
120 PRINT "SYSTEM STATEMENT FAILED"
>RUN
 FILES STORED = 1

 FILES NOT STORED = 0
```

The SYSTEM statement in line 20 causes execution of the MPE/3000 FILE command that assigns a file ABC to tape. In line 90 the SYSTEM statement causes execution of the MPE/3000 STORE command that stores the BASIC file A on the tape file ABC.