Agilent Technologies

Running Rocky Mountain Basic from Board Test Basic

Rev B

Many of you have existing programs written in Rocky Mountain Basic (RM-Basic), or have found the wealth of RM-Basic program examples given in Agilent manuals, and wish to use them to do external instrumentation control using the Agilent 3070 Board Test Family. There is a minor problem running these programs however, since the Agilent 3070 uses a modified subset of RM-Basic called BT-Basic (Board Test Basic). A procedure which allows you to use RM-Basic programs while operating from BT-Basic programs, such as the Agilent 3070 testplan, is given below. Note the example passes variables in BOTH directions using pipes. Once set up this should be relatively transparent to the operator.

1. Make sure that RM-Basic is installed on your computer. The easiest way to do this is to type 'rmb' in a shell window. If the application is loaded an HP BASIC/UX window will appear on your screen. If you do not already have RM-Basic, it can be ordered under part number E2046A (license) and E2045A (media and documentation) option AAH (dat tape) on a series 700. For an older series 300, use part number E2040L (license) and E2040B (media and documentation) option AAH (dat tape).

2. Load the desired RM-Basic program on to your computer. You must then edit the RM-Basic program such that it becomes a subroutine. Note that editing and reading of RM-Basic programs must be done in the RMB environment. If you 'load' an RM-Basic program from the BT-Basic environment it will look like garbage. In this example the RM-Basic program was stored as BD_CONT.

3. Once the RMB program has been converted to a subroutine, a header must be added. This header will be "called" by the BT-Basic testplan. An example header with comments is given below.

```
10   !** Lines 10 thru 500 were added for this application.
20   !** The program was also renumbered.
30   !**
40   !** Any variables that are used in the RMB program must be dimensioned
50   !** in this header (as they already should be in the RMB subroutine).
60   !**
70        OPTION BASE 1
80        COM /Fault/ ,Power_l(8),Rats(4),Gain(12)
90        COM /Raid/ INTEGER Name(32)
100       COM Tx_data$[100], Rx_data$[140]
110       RESET 7      !** Insure instruments are reset and ready.
120       CLEAR 7
130       ABORT 7
140       WAIT 1
150       CLEAR SCREEN
```

```
160       PRINT
170       PRINT "RMB ready to run"        !** Appears in RMB window only.
180       PRINT
190       ASSIGN @Send_data TO "output-pipe"
200       ASSIGN @Recv_data TO "input-pipe"
210       !
220       LOOP
230            ENTER @Send_data;Command$  !** get data from BT-Basic
240            !** PRINT "Command$ = ";Command$
250            IF Command$="QUIT" THEN QUIT    !** ends RM-Basic session
260            ENTER @Send_data;Return_A$
270            ENTER @Send_data;Number
280            !** PRINT "========================================="
290            !** PRINT
300            !** PRINT "RMB Input = ";Command$; Return_A$; Number
310            !** PRINT
320            CALL Test(Command$,Return_A$,Number) !** call RM-Basic
330            !** PRINT "RMB Output;"
340            !** PRINT "-----------"
350            !** PRINT "    ";Command$
360            !** PRINT "    ";Return_A$
370            !** PRINT "    ";Number
380            !** PRINT
390            OUTPUT @Recv_data;Command$,Return_A$,Number !** to BT-Basic
400            !** WAIT 1
410       END LOOP
420       END
430       !
440       !
450       !
460       !
470       !
480  !** Original Rocky Mountain Basic Program.
490  !** If it is not already a subroutine, it must be converted to one.
500  !**
510  SUB Test(Command$,OPTIONAL Return_A$,Number)
```

4. Now it is simply a matter of adding a few changes to the BT-Basic testplan. It is suggested that you identify the needed RM-Basic program in the header of the testplan to avoid confusion later on. For example;

```
 !** Revision   3 Apr 96
!*************************************************************************
!**
!** This is a custom testplan, do not regenerate
!**
!** Used with Rocky Mountain Basic program BD_CONT
!**
!*************************************************************************
!
!   HP3070 STANDARD TESTMAIN                  Revision: "A700/B100:1(full)"
```

5. Next the sub Initialize_Constants subroutine needs a few modifications.

a. Add a global so any other subroutines can pass the desired variables.

```
!** In sub Initialize_Constants, add the following;
        global @Send_data, @Recv_data
        global @TS_8920   !** Step 5f below will use this line
```

b. Next create the pipes IF they do not already exist in the directory.

```
!** Execute this command once to create needed pipes in the directory.
    !**      exec '/etc/mknod output-pipe p'
    !**      exec '/etc/mknod  input-pipe p'
```

c. Open the pipes

```
!** Execute the following when nrun = 1 in order to open the pipes;
        assign @Send_data to "output-pipe"
        assign @Recv_data to  "input-pipe"
```

d. If RM-Basic windows exist, the program will not start correctly, so;

```
 !** Kill any existing RMB windows
        enter "ps -ef|grep BD_CONT|grep -v grep|cut -c 10-15|",,Err;Process$
        !** print "Error = ";Err
        if Err <> 101007 then           !** 101007 is BD_CONT not found
            execute "kill "&Process$

        end if
```

e. Start the RM-Basic program running (example program name is BD_CONT).

```
    !** create rmb window and run program

          execute "rmb BD_CONT &"

      !** wait for window to be established
          wait 3
```

f. The /dev file must know the address of the external instrument being controlled by the RM-Basic program, and the external instrument should be reset. Assuming an HP8920B at address 714 is being used;

```
!** From unix root, do; # /etc/mknod /dev/ts_8920 c 21 0x070e00
      !** It may also be necessary to; # chmod 666 /dev/ts_8920
      assign @TS_8920, Error to "/dev/ts_8920"; read,write,exclusive
      if Error <> 0 then
          print "From unix root; mknod as shown above must have been done"
          print "Try running the line 6 up from where program is paused."
          pause
      end if
      on error call HELP
      timeout @TS_8920, 12                !** set 12 second timeout on HPIB
      remote  @TS_8920                    !**
      output  @TS_8920,,Error; "*RST"     !** preset the HP8920B
      if Error = 100165 then
          print "Make sure the HP8920B is powered up and connected to HPIB."
          print "It is not responding for some reason."
          pause
      end if
      on error recover Error_Trap         !** reset on error
      !**
      !**          RM-Basic should be select code 9
      !**
```

6. It is possible that the operator will use the BREAK key to terminate the program. In this case it is desireable to also stop the RM-Basic program. This is easily done by adding the following to the 'Break_Trap:' routine in the testplan right before the 'off error' command.

```
!** kill any existing RM-Basic window running BD_CONT.
          enter "ps -ef|grep BD_CONT|grep -v grep|cut -c 10-15|",,Err;Process$
          if Err <> 101007 then            !** 101007 is BD_CONT not found
              execute "kill "&Process$

          end if
```

7. Finally we come to the actual use of the RM-Basic program itself. Usually you will have a subroutine, perhaps 'sub Analog_Functional_Tests', where you wish to run the RM-Basic program. A portion of that testplan subroutine is shown below;

```
!** Program board to reset, send 8920 RFG:AMPL -50, frequency 120
     Command$ = "RESET"
     Return_A$ = "RFG:AMPL -50"
     Number = 120
     call Setup_Board(Command$, Return_A$, Number)
```

8. Since control usually requires multiple calls to the RM-Basic program, it is easier to make it a seperate subroutine. For example;

```
  sub Setup_Board(Command$, Return_A$, Number)
     global @Send_data, @Recv_data
     !** Clear pipes.  Error 100044 indicates the pipe is empty.
         Clear:
         wait .1
         enter  @Send_data ,,Error; Unused
             if Error <> 100044 then goto Clear
         enter  @Recv_data ,,Error; Unused
             if Error <> 100044 then goto Clear
         Error = 0
     !** Output data to RM-Basic
         output @Send_data; Command$
         output @Send_data; Return_A$
         output @Send_data; Number
         !** print "Transmitting;"
         !** print "   |";Command$
         !** print "   |";Return_A$
         !** print "   |";Number
```

!** Now return data to testplan from RM-Basic. Try = 0 Next_try: enter @Recv_data,,Err; Command$,Return_A$,Number_$

if Err <> 0 then Try = Try + 1 !** print"Err = ";Err if Try > 50 then !** This is somewhat data dependent call Help pause goto Quit_1 end if wait .2 goto Next_try end if Quit_1: !** Returning variables all are strings and have an added !** carriage return which must be removed. Command$ = (Command$ [1;(len(Command$) -1)]) Return_A$ = (Return_A$[1;(len(Return_A$)-1)]) Number = val(Number_$ [1;(len(Number_$) -1)]) !** print "Receiving;" !** print " |";Command$!** print " |";Return_A$!** print " |";Number subend

9. Unfortunately the program may, in some strange circumstances, hang up. It can be particularly frustrating for operators, so a little program like that below is recommended.

```
sub HELP
       print using "@"     !** clear the screen
       beep | print | beep | print | print | print
       print
       print "A problem exists with the BT-Basic to BASIC/UX communications."
       print "If program can not clear itself this message remains on screen."
       print "Possibly there are more than one BASIC/UX windows present."
       print "These windows are probably below the window you are working in."
       print
       print "Put the cursor into any EXTRA (more than one) BASIC/UX window,"
       print "hit the break key, and then type quit."
       print "Make sure the remaining BASIC/UX window is running, NOT paused."
       print "Return cursor to the BT-BASIC window, type cont, and hit Enter."
       print
       print "Program may continue when error is freed, or you may have to"
       print "start over by hitting break in this window and then typing run."
       print
       print
       wait 3
   subend
```