CLARION TECHNICAL BULLETIN

Bulletin #116

Adding Hooks To Designer Models

Overview

This bulletin discusses how you can increase the control you have in your Designer-generated code by adding a few simple statements or "hooks" into Designer's model file. Sample model procedures are provided.

Note: The techniques covered in this bulletin deal primarily with the Batch 2008 NETWORK model file, but they may also be applied to the STANDARD model. This discussion, dealing with the FORM model procedure, may be used with any of the other model procedures.

Adding Hooks To Designer Models

The Designer utility program uses a "model" file to generate code. There are two model files: the NETWORK model file (for applications used on a network) and the STANDARD model file.

You can gain further control of your programs by adding a few simple statements to the model file. These statements allow code placed in an "Other" procedure to be executed. These statements are called "hooks." By placing your "hook" code in an "Other" procedure, you can change the way a procedure works without regenerating the entire program. (You can also call a "hook" from the Edit line in an Entry field.)

Assuming that Designer is using the NETWORK model file to generate code, here are two cases where you would want to create a hook in the NETWORK model:

- Case #1: **When deleting a record in one file requires deleting records in other files.** For example, suppose you want to delete a record for an invoice and all its associated detail file records simultaneously.

- Case #2: **When you want to retain the original value of a field so that you can perform calculations on an updated record.** For example, suppose you want to adjust a customer balance for an invoice change. You need to retain the initial total of the invoice and update the customer record by the difference between the "old" total and the "new" total.

To create the hooks for both of these cases, modify the model file using the "@SETUP" keyword wherever a hook is needed. To identify the hook, set a variable in the model file to the hook number. Then the @SETUP code is executed. The "hook" numbers are contained in an INCLUDE file called "HOOKEQU.CLA." Add this INCLUDE file to the include files in the GLOBAL model procedure.

Finally, add a Setup procedure to the FORM that needs the extra hooks. In our example, we would set the Setup procedure to:

```
INV_SETUP(H#);IF H# THEN GOTO PROC_EXIT.
```

The hook number is in H# (set by the model file) and is passed as an external to INV_SETUP. Passing the hook number as an external allows INV_SETUP to change the H# value. The IF after the call allows INV_SETUP to control whether the FORM should continue or return to the caller. If INV_SETUP wants to cause the FORM to return, then it simply leaves H# non-zero. If it wants to continue, it zeroes H#.

The example model procedure shown on the next page contains more hooks than you need to accommodate the cases listed above, but it demonstrates how to add more hooks if you need them. To summarize, the process is:

- Add a new equate to the HOOKEQU.CLA file.

- Add the call to @SETUP with H# set to the new hook value.

- Add a new OF to the Setup procedure's CASE statement.

Notice that OFs of unused hook values are commented out in the INV_SETUP procedure. This was done to keep the code as small as possible. If you want to use another existing hook, remember to un-comment the OF in the Setup procedure.


```
   Set up procedure specified in Designer for the FORM:
        INV_SETUP(H#);IF H# THEN GOTO PROC_EXIT.

   INCLUDE line for the GLOBAL model procedure:
        INCLUDE('HOOKEQU.CLA')

   The HOOKEQU.CLA include file:
        BEGIN    EQUATE(01)             !BEFORE THE OPEN(SCREEN)
        SETUP    EQUATE(02)             !NORMAL SETUP
        TOPMAIN  EQUATE(03)             !TOP OF THE MAIN LOOP
        BEFACC   EQUATE(04)             !BEFORE ACCEPT
        AFTACC   EQUATE(05)             !AFTER ACCEPT
        FLOOP    EQUATE(06)             !BEFORE THE FIELD EDIT LOOP
        TOPFLOOP EQUATE(07)             !TOP OF THE FIELD EDIT LOOP
        FIRSTFLD EQUATE(08)             !FIRST_FIELD EDIT
        PROCEXIT EQUATE(09)             !PROCEDURE EXIT
        LASTFLD  EQUATE(10)             !LAST_FIELD EDIT
        BEFPUT   EQUATE(11)             !BEFORE ADD/PUT/DELETE
        AFTPUT   EQUATE(12)             !AFTER ADD/PUT/DELETE
        NEXTFORM EQUATE(13)             !BEFORE CALLING @NEXTFORM
        DELFLD   EQUATE(14)             !DELETE_FIELD EDIT
        LASTCHNG EQUATE(15)             !IF RECORD HAS CHANGED

   FORM model procedure code:            (New code marked with !###)

 *FORM*************************************************************************
 @PROCNAME    PROCEDURE

SCREEN       SCREEN       PRE(SCR),@SCREENOPT
                          @PAINTS
                          @STRINGS
                          @VARIABLES
                          ENTRY,USE(?FIRST_FIELD)
                          @FIELDS
                          @PAUSE
                          ENTRY,USE(?LAST_FIELD)
                          PAUSE(''),USE(?DELETE_FIELD)
             .

SAVE_RECORD  GROUP;BYTE,DIM(SIZE(@PRE:RECORD)).
SAVE_MEMO    GROUP;BYTE,DIM(SIZE(@MEMO)).


  EJECT
  CODE
  H#=BEGIN        !###                        !SET HOOK TO BEGIN
  @SETUP          !###                        !CALL SET UP
  OPEN(SCREEN)                                !OPEN THE SCREEN
  SETCURSOR                                   !TURN OFF ANY CURSOR
  SAVE_RECORD = @PRE:RECORD                   !SAVE THE ORIGINAL

  SAVE_MEMO   = @MEMO                         !SAVE THE ORIGINAL

  H#=SETUP        !###                        !SET HOOK TO SETUP

  @SETUP                                      !CALL SETUP PROCEDURE
  DISPLAY                                     !DISPLAY THE FIELDS
  EXECUTE ACTION                              !SET THE CURRENT RECORD POINTER
    POINTER# = 0                              !  NO RECORD FOR ADD
    POINTER# = POINTER(@FILENAME)             !  CURRENT RECORD FOR CHANGE
  .
```

```
LOOP                                          !LOOP THRU ALL THE FIELDS
  H#=TOPMAIN    !###                          !SET HOOK TO TOPMAIN
  @SETUP        !###                          !CALL SET UP
  MEM:MESSAGE = CENTER(MEM:MESSAGE,SIZE(MEM:MESSAGE)) !DISPLAY ACTION MESSAGE
 @LOOKUPS                                    !DISPLAY FROM OTHER FILES
  @SHOW                                       !DISPLAY STRING VARIABLES
  @COMPUTE                                    !DISPLAY COMPUTED FIELDS
  @RESULT                                     !MOVE RESULTING VALUES
  ALERT                                       !RESET ALERTED KEYS
  ALERT(ACCEPT_KEY)                           !ALERT SCREEN ACCEPT KEY
  ALERT(REJECT_KEY)                           !ALERT SCREEN REJECT KEY
  @ALERT                                      !ALERT HOT KEY
  H#=BEFACC     !###                          !SET HOOK TO BEFACC
  @SETUP        !###                          !CALL SET UP
  ACCEPT                                      !READ A FIELD
  H#=AFTACC     !###                          !SET HOOK TO AFTACC
  @SETUP        !###                          !CALL SET UP
  @CHECKHOT                                   !ON HOT KEY, CALL PROCEDURE
  IF KEYCODE() = REJECT_KEY THEN RETURN.      !RETURN ON SCREEN REJECT KEY
  EXECUTE ACTION                              !SET ACTION MESSAGE
    MEM:MESSAGE = 'Record will be Added'      !
    MEM:MESSAGE = 'Record will be Changed'    !
    MEM:MESSAGE = 'Press Enter to Delete'     !
  .
  EDIT_RANGE# = FIELD()                       !SET ONE FIELD EDIT RANGE
  IF KEYCODE() = ACCEPT_KEY                   !ON SCREEN ACCEPT KEY
    UPDATE                                    !  MOVE ALL FIELDS FROM SCREEN
    EDIT_RANGE# = FIELDS()                    !  AND EDIT REMAINING FIELDS
  .                                           !
  H#=FLOOP
  @SETUP
  LOOP FIELD# = FIELD() TO EDIT_RANGE#        !EDIT FIELDS IN THE EDIT RANGE
    H#=TOPFLOOP !###                          !SET HOOK TO TOPFLOOP
    @SETUP      !###                          !CALL SET UP
    CASE FIELD#                               !JUMP TO FIELD EDIT ROUTINE
    OF ?FIRST_FIELD                           !FROM THE FIRST FIELD
      H#=FIRSTFLD !###                        !SET HOOK TO FIRSTFLD
      @SETUP      !###                        !CALL SET UP
      IF KEYCODE() = ESC_KEY THEN RETURN.     !  RETURN ON ESC KEY
      IF ACTION = 3 THEN SELECT(?DELETE_FIELD).!  OR CONFIRM FOR DELETE

    @EDITS                                    !EDIT ROUTINES GO HERE
    OF ?LAST_FIELD                            !FROM THE LAST FIELD
      H#=LASTFLD  !###                        !SET HOOK TO LASTFLD
      @SETUP      !###                        !CALL SET UP
      IF ACTION = 2                           !IF UPDATING RECORD
        HOLD(@FILENAME)                       !  HOLD FILE
        GET(@FILENAME,POINTER#)               !  RE-READ SAME RECORD
        IF ERRORCODE() = 35                   !  IF RECORD WAS DELETED
          ACTION = 1                          !    THEN ADD IT BACK
        ELSIF ¦                               !  IF IT HAS BEEN CHANGED
          @MEMO <> SAVE_MEMO OR ¦             !


          @PRE:RECORD <> SAVE_RECORD          !    BY ANOTHER STATION
          MEM:MESSAGE = 'CHANGED BY ANOTHER STATION' !INFORM USER
          SELECT(2)                           !  GO BACK TO FIELD 1
          BEEP                                !  SOUND ALARM
          RELEASE(@FILENAME)                  !  RELEASE FILE
          SAVE_RECORD = @PRE:RECORD           !  SAVE RECORD
          SAVE_MEMO = @MEMO                   !  SAVE MEMO
          DISPLAY                             !  DISPLAY THE FIELDS
```

```
          H#=LASTCHNG  !###                    !SET HOOK TO LASTCHNG
          @SETUP       !###                    !CALL SET UP
          BREAK                                !  AND CONTINUE
        .
        UPDATE                                 !UPDATE FROM SCREEN TO RECORD
        @RESULT                                !MOVE RESULTING VALUES
      .
      H#=BEFPUT        !###                    !SET HOOK TO BEFPUT
      @SETUP           !###                    !CALL SET UP
      EXECUTE ACTION                           !  UPDATE THE FILE
        ADD(@FILENAME)                         !    ADD NEW RECORD
        PUT(@FILENAME)                         !    CHANGE EXISTING RECORD
        DELETE(@FILENAME)                      !    DELETE EXISTING RECORD
      .
      IF ERROR() THEN STOP(ERROR()).           !  CHECK FOR UNEXPECTED ERROR
      H#=AFTPUT        !###                    !SET HOOK TO AFTPUT
      @SETUP           !###                    !CALL SET UP
      PUT(@FILENAME2)                          !  UPDATE SECONDARY FILES
      PUT(@FILENAME3)                          !  UPDATE SECONDARY FILES
      PUT(@FILENAME4)                          !  UPDATE SECONDARY FILES
      IF ACTION = 1 THEN POINTER# = POINTER(@FILENAME). !POINT TO RECORD
      SAVE_RECORD = @PRE:RECORD                !  NEW ORIGINAL
      SAVE_MEMO   = @MEMO                      !  NEW ORIGINAL
      @NEXTFORM                                !  CALL NEXT FORM PROCEDURE
      H#=NEXTFORM      !###                    !SET HOOK TO NEXTFORM
      @SETUP           !###                    !CALL SET UP
      ACTION = 0                               !  SET ACTION TO COMPLETE
      RETURN                                   !  AND RETURN TO CALLER

    OF ?DELETE_FIELD                           !FROM THE DELETE FIELD
      H#=DELFLD        !###                    !SET HOOK TO DELFLD
      @SETUP           !###                    !CALL SET UP
      IF KEYCODE() = ENTER_KEY                 !  ON ENTER KEY
      OR KEYCODE() = ACCEPT_KEY                !  OR CTRL-ENTER KEY
        SELECT(?LAST_FIELD)                    !    DELETE THE RECORD
      ELSE                                     !  OTHERWISE
        BEEP                                   !    BEEP AND ASK AGAIN
  . . . .

PROC_EXIT

  H#=PROCEXIT          !###                    !SET HOOK TO PROCEXIT
  @SETUP               !###                    !CALL SET UP
  RETURN
```



```
!************************************************************************
  The following is a separate procedure named as an "Other" in Designer
  and is not part of the model file.  This code actually does the
  hook processing. It cases off of the passed parameter to get to the
  correct hook code.
!************************************************************************

    The following is INV_SETUP:

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!
! Module:INVSETUP.CLA
!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

    INV_SETUP PROCEDURE(H)                      !SETUP PROCEDURE FOR INVOICE
    H EXTERNAL

      CODE

      CASE H
    !EDIT CODE CAN GO HERE - TO KEEP ALL THE CODE IN ONE PLACE
    !TELL DESIGNER THAT THE EDIT PROCEDURE FOR THE FIELD NEEDING THE EDIT
    !IS INV_SETUP(1000).
    ! OF 1000

    ! OF BEGIN
      OF SETUP
      OROF LASTCHNG
        OLDTOT$ = INV:TOTAL
    ! OF TOPMAIN
    ! OF BEFACC
    ! OF AFTACC
    ! OF FLOOP
    ! OF TOPFLOOP
    ! OF FIRSTFLD
    ! OF PROCEXIT
    ! OF LASTFLD
      OF BEFPUT !if we get here the record is the same
                !as when the procedure started and it
                !is in memory and held and ready to go
        CUS:NUMBER = INV:CUSTOMER          !SET CUSTOMER NUMBER
        HOLD(CUSTOMER)                     !GET THE CUSTOMER RECORD
        GET(CUSTOMER,CUS:CUST_KEY)         !WITH EXCLUSIVE ACCESS
        IF NOT ERROR()                     !IF THERE IS NO ERROR
          CASE ACTION
          OF 2 !REVISE                     !ON A CHANGE SUBTRACT BEFORE AMOUNT
            CUS:BALANCE -= OLDTOT$         !THEN FALL INTO ADD TO ADD IN NEW AMOUNT
          OROF 1 !ADD                      !ON AN ADD ADD THE AMOUNT
            CUS:BALANCE += INV:TOTAL
          OF 3 !DELETE                     !ON A DELETE SUBTRACT THE AMOUNT
            CUS:BALANCE -= INV:TOTAL
          .
          PUT(CUSTOMER)                    !PUT THE RECORD BACK
          IF ERROR()                       !IF THERE WAS AN ERROR
            RELEASE(CUSTOMER)              !RELEASE THE HOLD
            LOOP;STOP(ERROR()).            !THIS SHOULD NEVER HAPPEN
          .
        .
        IF ACTION = 3 !DELETING INVOICE    !ON A DELETE
                                           !GO GET RID OF TRANS FOR ORDER
```

```
        TRA:INVOICE = INV:NUMBER        !SET INVOICE NUMBER
        SET(TRA:INV_KEY,TRA:INV_KEY)
        IF NOT ERROR()                  !WHILE WE HAVE THE RIGHT RECORDS
          LOOP UNTIL EOF(TRANS)
            HOLD(TRANS)
            NEXT(TRANS)
            IF ERROR() OR TRA:INVOICE NOT = INV:NUMBER THEN BREAK.
            DELETE(TRANS)               !DELETE THEM

            IF ERROR() THEN RELEASE(TRANS).
          .
        .
      .
! OF AFTPUT
! OF NEXTFORM
! OF DELFLD
  .
  H = 0
  RETURN
```