# 7

# SUBFILES AND DATA QUEUES —A PERFECT COMBINATION

This next technique came to me when I was asked to create a subfile application that works like the Program Development Manager (PDM). If you've ever looked at PDM on the AS/400, you may have marveled at what a cool subfile application it is. PDM is extremely flexible. It allows you to position to any place in a subfile-like panel, page forward or backward from that position, change a record on any page of the subfile, and process all changed records only when the Enter key is pressed. On their own, each feature is simple to code in an RPG subfile program. But the real fun begins when you combine the features.

I worked for a software development house that wanted the IBM look and feel on all of its screens. The thinking was that users familiar with the AS/400 would be comfortable using the interactive screens in our software and would require less training. It seemed simple enough at first, but as you will soon see, incorporating all the features included with PDM into a subfile application is no small task. In fact, PDM isn't even a subfile application; its displays are written using the User

Interface Manager (UIM), the language used for many native OS/400 commands and all of the help panels on the AS/400.

See Figure 7.1 for a typical PDM screen.

```
                        Work with Objects Using PDM                 MYSYSTEM

Library . . . . .   VANDEVER         Position to . . . . . .
                                      Position to type  . . . . .

Type options, press Enter.
  2=Change       3=Copy        4=Delete      5=Display      7=Rename
  8=Display description        9=Save       10=Restore     11=Move ...

Opt  Object      Type        Attribute   Text
     QCLLESRC    *FILE       PF-SRC      ILE CL Source
     QCLSRC      *FILE       PF-SRC      Old CL source
     QDDSSRC     *FILE       PF-SRC      DDS Source
     QRPGLESRC   *FILE       PF-SRC      ILE RPG Source
     QTESTSRC    *FILE       PF-SRC      Internet Test Program Src



                                                                     Bottom
Parameters or command
===>
F3=Exit          F4=Prompt            F5=Refresh          F6=Create
F9=Retrieve      F10=Command entry    F23=More options    F24=More keys
```

*Figure 7.1: Example of a PDM screen.*

## THE DILEMMA

Well, what do you do if you're an RPG programmer who likes the look, feel, and flexibility of PDM, but doesn't know how to obtain them using UIM? Do you learn UIM? You could, but if you already know RPG, is learning a new language the most effective use of your time? This was the dilemma I faced.

I'm not against learning UIM, but I thought that there must be a more efficient way to get the same results using RPG and subfile processing. After some research, I recommended to my programming group that we use data queues to add the necessary flexibility to our subfile applications. Data queues are the way to get the features and flexibility you're looking for—without having to learn UIM.

## DATA QUEUES 101

Data queues are a type of system object (type *DTAQ) you can create and maintain using OS/400 commands and APIs. They're AS/400 objects that can be used to send and receive multiple record-like strings of data. Data may be sent to and received from a data queue from multiple programs, users, or jobs, making them an excellent mechanism for sharing data. They provide a fast means of asynchronous communication between two jobs because they use less system resources than database files, message queues, or data areas. Data queues have the ability to attach a sender ID to each entry being placed on the queue. The sender ID, an attribute of the data queue that's established when the queue is created contains the qualified job name and current user profile. Another advantage to using data queues is the ability to set the length of time a job will wait for an entry before continuing its processing. A negative wait parameter will tell the job to wait indefinitely for an entry before processing. A wait parameter of 0 to 99,999 will tell the job to wait that number of seconds before processing.

High-level language programs (HLLs) can send data to a data queue using the Send to a Data Queue (QSNDDTAQ) API and receive data using the Receive from a Data Queue (QRCVDTAQ) API. Data queues can be read in FIFO sequence, in LIFO sequence, or in keyed sequence. The technique I use for building PDM-like subfile applications requires a keyed data queue. Keyed data queues allow the programmer to specify a specific order in which entries on the data queue are received or to retrieve only data queue entries that meet a criterion. That is, the programmer can receive a data queue entry that's equal to (EQ), greater than (GT), greater than or equal to (GE), less than (LT), or less than or equal to (LE) a search key.

## WHY SHOULD I USE A DATA QUEUE IN A SUBFILE PROGRAM?

The reason behind using data queues in a subfile program stems from a combination of user requirements and an interest in selecting the most efficient solution for that combination. I wanted to provide the PDM look and feel by allowing users to position to anywhere in the subfile using the position-to field and to page

up or down from that new position. This is easily accomplished using a page-at-time subfile.

However, I also wanted the data that was changed on subfile records to be saved, regardless of where the user navigated in the subfile, until the user was ready to process them by pressing the Enter key. This is accomplished easily in a load-all or self-extending subfile, but not in a page-at-a-time subfile.

With page-at-a-time subfiles, you have to clear and build one new page of subfile records every time the user pages and uses a position-to field. Any previously changed records are not saved. I needed a way to combine the flexibility of the page-at-a-time subfile with the capability to process the changed records only when desired. I needed a place to store and reuse changed subfile records until the user was ready to process them. Using data queues to accomplish this task instead of data structures, files, or arrays frees you from some additional work.

During an interactive job, the data queue APIs can provide better response time and decrease the size of the program, as well as its process activation group (PAG). In turn, this can help overall system performance. In addition, data queues allow the programmer to do less work. When you receive an entry from a data queue using the QRCVDTAQ command, it's physically removed from the data queue. The programmer doesn't have to add code to deal with unnecessary entries. Here's what you do.

I use the data queue to store a replica of the changed subfile record. Each time a subfile record is changed and the Enter key or Page key is pressed, the changed subfile record is stored in the data queue. I do this because I build the subfile one page at a time and need to know which records were previously changed. Once records are no longer displayed on the screen, they're not part of the subfile. When the user positions through the subfile by scrolling or by keying something in the position-to field, the program will check to see if each record read from the data file exists in the data queue before loading it to the subfile. If the record exists in the data queue, it's written to the subfile, along with the earlier changes, and marked as changed. When the user is ready to process all the changed records by pressing the Enter key with nothing in the position-to field, the records

will be processed appropriately from the data queue. I have modified my original master file maintenance application using these principles. You can check out the CL, DDS, and RPG that make this work at the end of this chapter.

## THE DDS—THE SAME AS IT EVER WAS

Before I get to the RPG, I need to say a few things about the DDS and CL. For this application to have the necessary flexibility, the RPG program—and not OS/400—must completely control the subfile. Of course, you know what that means. For this to happen, the subfile page (SFLPAG) and subfile size (SFLSIZ) values must be equal. The subfile will never contain more than one page, but, as you'll see, the program will make it appear that the subfile contains much more than one page of data. You should otherwise recognize the DDS as our Master File Maintenance DDS and understand what it's doing. The complete DDS is included at the end of this chapter (see SFL011DF).

## CONTROL LANGUAGE—SO LITTLE CODE, SO MUCH CONTROL

It's important to note that even though entries are removed from a data queue after they're received, the space containing the entry isn't. Over the long run, performance will suffer because the data queue's size will increase. For this reason, I delete and re-create the data queue each time the program is called. Even if you build your data queues in QTEMP, as I do, it's best to delete and re-create them in case the user calls the program more than once before signing off. Program SFL011CL accomplishes this task. Again, you can find this program at the end of the chapter.

## ABRACADABRA! THE SUBFILE'S NEVER MORE THAN ONE PAGE

Now that the setup issues have been covered, let's perform some magic. Let's start with the RPG program, which is our Master File Maintenance program from

chapter 4, with a few additions thrown in. Rather than spending time rehashing the basic subfile techniques you've already mastered, I'll concentrate on how the program uses a data queue to make the subfile appear to be larger than it really is.

The program's first task is to load the subfile with one page of records (in this case, nine). This code is shown in Figure 7.2.

```
     *
     * Load data to subfile
     *
    C                   do        sflpag
    C                   read      sfl001lf                                90
    C                   if        *in90
    C                   leave
    C                   endif
     *
    C                   eval      option = *blanks
    C                   exsr      rcvque
    C                   eval      rrn1 = rrn1 + 1
    C                   if        rrn1 = 1
    C                   eval      savlnam = dblnam
    C                   eval      savfnam = dbfnam
    C                   endif
    C                   write     sfl1
    C                   eval      *in74 = *off
    C                   enddo
```

*Figure 7.2: Loading the subfile with one page of records.*

Notice that each time a record is read from the data file, the RCVQUE subroutine is executed. For the initial subfile load, this subroutine won't accomplish anything— (I will explain this later). However, after the initial load, the RCVQUE subroutine plays a vital part in the subfile load routine.

Once the subfile is initially loaded and displayed, the user can do several things. He can scroll through the subfile; add, change, display, or delete records; position to another place in the subfile; or exit the program.

The code listed in Figure 7.3 shows that no matter what the user decides to do, the ADDQUE subroutine is executed each time the Enter key or a valid function key (other than F3 or F12) is pressed. This subroutine uses the READC op code to

find changed records in the subfile and add them to the data queue using the QSNDDTAQ API.

```
     * If ENTER key is pressed and position-to non blank,
     * reposition the subfile to close to what was entered.
     *
    C                   when      (cfkey = enter) and (ptname  *blanks)
    C                   exsr      addque
    C     ptname        setll     sfl001lf
    C                   exsr      sflbld
    C                   clear                   ptname
     *
     * If ENTER key is pressed and position-to is blank,
     * process screen to interrogate options selected by user
     *
    C                   when      (cfkey = enter) and (ptname = *blanks)
    C                   exsr      addque
    C                   exsr      prcsfl
    C     savkey        setll     sfl001lf
    C                   exsr      sflbld
     *
     * Roll up - load the data Q's before loading subfile
     *
    C                   when      (cfkey = rollup) and (not *in90)
    C                   exsr      addque
    C                   exsr      sflbld
     *
     * User presses F6, throw the add screen, clear, and rebuild subfile
     *
    C                   when      cfkey = add
    C                   movel(p)  'Add   '      mode
    C                   exsr      addque
    C                   exsr      addrcd
    C     dblnam        setll     sfl001lf
    C                   exsr      sflbld
     *
     * Roll down - load the data Q's before loading the subfile.
     *
    C                   when      (cfkey = rolldn) and (not *in32)
    C                   exsr      addque
    C                   exsr      goback
    C                   exsr      sflbld
     *
    C                   when      *inkl
    C                   leave
```

*Figure 7.3: Each time a valid function key, other than F3 or F12, is pressed, the changed records are added to the data queue.*

Table 7.1 gives an explanation of the QSNDDTAQ parameters. The data queue entry will contain the option selected by the user and the key of the data file.

**Table 7.1: Required QSNDDTAQ API Parameters.**

| Parameter | Explanation |
|---|---|
| QUEUE | Name of the data queue |
| LIB | Library containing the data queue |
| LEN | Length of the data being written to the data queue |
| DATA | The actual data being written to the data queue |

Figure 7.4 shows the contents of the data queue when the user selects a 4 next to a record, then pages down to see another page. When he paged down, an entry was added to the data queue that consisted of the option (4) and the value in DBIDNM, which is the key to the data file, the key to the data queue, and a hidden field in the subfile.

```
                          Data Queue Display - TAA        2/02/00  12:14

Queue: SFL011DQ    Lib: QTEMP       Nbr of Entries:     1  Seq: *KEYED
Max Entry Length:   256   Key Length:     7   Force: *NO    Sender ID: *NO
        Text:
Entry: 00001     Enqueue Date: 02/02/00   Enqueue Time: 12:07:36
Posn  ....+....1....+....2....+....3....+....4....+....5....+....6....
   1  0000004                    40000004
  33  (DBIDNM as the key)       (OPTION followed by DBIDNM)
  97
 161
 225
```

*Figure 7.4: The contents of the data queue after the user places a 4 in the option field and presses the page-down key.*

The ADDQUE subroutine, shown in Figure 7.5, keeps track of all records changed through the subfile. For example, if the user decides to delete two records on the next page after selecting a 4 to delete a record on the current page, the ADDQUE subroutine sends the two changed records to the data queue before rebuilding the subfile in the page forward (SFLBLD) routine. Now there are three entries in the data queue, and nothing has been deleted. The same logic holds true if the user decides to position to another part of the subfile using the position-to field.

```
     *****************************************************************
     *  ADDQUE - Add subfile data to Data Queues
     *****************************************************************
     *
    C     addque        begsr
     *
     * Read the changed subfile records and write them to the data Q's
     * The first data queue is keyed by whatever the unique key of the file
     * is.  If no unique key in the file, use the relative record number.
     * This queue is used to save options selected on a specific subfile
     * line.  The second queue is keyed by option, and is used to process
     * like options together when the enter key is selected
     *
    C                   readc     sfl1
     *
    C                   dow       not %eof
     *
    C                   eval      len = qlen
     *
    C                   call      'QSNDDTAQ'
    C                   parm                    queue
    C                   parm                    lib
    C                   parm                    len
    C                   parm                    data
    C                   parm                    keyln
    C                   parm                    key
     *
    C                   readc     sfl1
    C                   enddo
     *
    C                   endsr
```

*Figure 7.5: This routine writes the changed records to the data queue.*

Now we can get to the details of the RCVQUE routine, as shown in Figure 7.6.

```
     *****************************************************************
     *  RCVQUE - Check DATAQUEUE before writing to subfile
     *****************************************************************
     *
    C     rcvque        begsr
     *
     * Read the data Q by the whatever the unique key from the
     * physical file to see if there is a saved option.  If so, display
     * the saved option when the subfile is displayed.
     *
```

*Figure 7.6: This routine removes entries from the data queue.*

```
C                   eval      order = 'EQ'
 *
C                   call      'QRCVDTAQ'
C                   parm                    queue
C                   parm                    lib
C                   parm                    len
C                   parm                    data
C                   parm                    wait
C                   parm                    order
C                   parm                    keyln
C                   parm                    key
C                   parm                    sndlen
C                   parm                    sndr
 *
C                   if        len > *zero
C                   eval      *in74 = *on
C                   endif
 *
C                   endsr
```

*Figure 7.6: This routine removes entries from the data queue (continued).*

This subroutine attempts to receive an entry from the keyed data queue using the same key as the record read from the database file (DBIDNM). The QRCVDTAQ API does this for you. The order is set to EQ (equal) so you will retrieve an entry only if there's one matching the record just read from the file. If the length is greater than 0 (`len > 0`), an entry was retrieved from the data queue. You then set on indicator 74, which conditions SFLNXTCHG in your DDS, to mark the record as changed when the subfile record is written. By doing this, subsequent READC operations will pick up the record the next time the page is processed.

Table 7.2 shows an explanation of the parameters for QRCVDTAQ.

If a matching entry exists in the data queue, the entry in the data queue—not the data from the database file—is written to the subfile. With this, the user can page and position through the subfile and store any changed records in the data queue. Whenever a record is read from the file, the data queue is checked to see if that record exists. If it does, it's displayed along with the previously selected option. If our user wanted to page up to see the first page after having selected two records for delete on the second page, he could do so. He would see a "4" in the

original record he selected for delete, and the data queue would now contain the two records from the second page.

If the user presses the Enter key and the position-to field is empty, the ADDQUE routine executes one last time to load any changes to the current page, and the PRCSFL routine is executed. The PRCSFL routine (shown in Figure 7.7) in this example is a little different than the one in my original Master File Maintenance program. This subroutine uses the RCVDTAQ API instead of READC to process all the changed records. Remember hat changed records will reside in the data queue, not the subfile, which never contains more than one page of data. By setting the key value, DBIDNM, to one and the order to GE (greater than or equal to), you're sure to retrieve all entries in the data queue. Figure 7.7 shows the RCVDTAQ API in action in the PRCSFL routine. This API will be run until the length (LEN) parameter is 0. That will happen when no more entries exist in the data queue.

*Table 7.2:*
*Required QRCVDTAQ API Parameters When Working with Keyed Data Queues.*

| Parameter | Explanation |
|---|---|
| QUEUE | Data queue name |
| LIB | Library containing the data queue |
| LEN | Length of entry received from the data queue |
| DATA | Data received from the data queue |
| WAIT | How long to wait for data (a negative number will cause the program to wait indefinitely) |
| ORDER | How to get the keyed data (EQ, GE, LT, etc.) |
| KEYLN | Length of the key to the data queue |
| KEY | The key field used to retrieve data |
| SNDLEN | Length of the sender ID information |
| SNDR | The sender ID information |

```
     * Receive data queue records until the queue is empty LEN = 0
     *
    C                   eval      dbidnm = 1
    C                   eval      order = 'GE'
     *
    C                   dou       len = *zero
     *
```

*Figure 7.7: How to process all the changed records from a subfile by going through the data queue.*

```
C                   call      'QRCVDTAQ'
C                   parm                    queue
C                   parm                    lib
C                   parm                    len
C                   parm                    data
C                   parm                    wait
C                   parm                    order
C                   parm                    keyln
C                   parm                    key
C                   parm                    sndlen
C                   parm                    sndr
 *
 *  If length is greater than zero, there was a record read.
 *  Process that record and receive from the second dataq to
 *  keep them in cinc.
 *
      C                   if        len > *zero
```

*Figure 7.7: How to process all the changed records from a subfile through the data queue (continued).*

Each time an entry is received, the data is run through a select routine to determine which function needs to be performed. In the case of this program, depending on the option taken, a display screen, an update screen, or a delete confirmation subfile will appear, just as it did in my earlier example.

Controlling the subfile within the RPG program and using data queues to store and retrieve changed subfile records allows you to create an extremely flexible subfile application that will furnish your users with everything they ever wanted in a subfile program. Besides, it's a great way to make a page-at-time subfile look like a lot like a load-all subfile.

## CODE EXAMPLES

The following code examples are used in this chapter.

## SFL011CL: CL Program to Create the Temporary Data Queue

```
/*============================================================*/
/*  To compile:                                               */
/*                                                            */
/*           CRTCLPGM PGM(XXX/SFL011CL) SRCFILE(XXX/QCLLESRC) */
/*                                                            */
/*============================================================*/
PGM

  DLTDTAQ    DTAQ(QTEMP/SFL011DQ)
  MONMSG     MSGID(CPF2105)
  CRTDTAQ    DTAQ(QTEMP/SFL011DQ) MAXLEN(256) SEQ(*KEYED) +
               KEYLEN(7)
  CALL       PGM(*LIBL/SFL011RG)

ENDPGM
```

## SFL011DF: DDS Using the Data Queue Technique

```
    A*
    A                                     DSPSIZ(24 80 *DS3)
    A                                     PRINT
    A                                     ERRSFL
    A                                     CA03
    A                                     CA12
    A*
    A          R SFL1                     SFL
    A*
    A  74                                 SFLNXTCHG
    A            DBIDNM    R        H      REFFLD(PFR/DBIDNM *LIBL/SFL001PF)
    A            OPTION         1A  B 10  3VALUES(‘ ’ ‘2’ ‘4’ ‘5’)
    A            DBLNAM    R        O 10  7REFFLD(PFR/DBLNAM *LIBL/SFL001PF)
    A            DBFNAM    R        O 10 31REFFLD(PFR/DBFNAM *LIBL/SFL001PF)
    A            DBMINI    R        O 10 55REFFLD(PFR/DBMINI *LIBL/SFL001PF)
    A            DBNNAM    R        O 10 60REFFLD(PFR/DBNNAM *LIBL/SFL001PF)
    A          R SF1CTL                   SFLCTL(SFL1)
    A*
    A                                     CF06
    A                                     SFLSIZ(0012)
    A                                     SFLPAG(0012)
    A                                     ROLLUP
    A                                     ROLLDOWN
    A                                     OVERLAY
    A N32                                 SFLDSP
    A N31                                 SFLDSPCTL
    A  31                                 SFLCLR
    A  90                                 SFLEND(*MORE)
    A            RRN1           4S 0H      SFLRCDNBR
    A                                 9  7‘Last Name’
    A                                     DSPATR(HI)
    A                                 9 31‘First Name’
```

*SFL011DF: DDS Using the Data Queue Technique (continued)*

```
A                                      DSPATR(HI)
A                                    9 55'MI'
A                                      DSPATR(HI)
A                                    9 60'Nick Name'
A                                      DSPATR(HI)
A                                    1  2'SFL011RG'
A                                    1 71DATE
A                                      EDTCDE(Y)
A                                    2 71TIME
A                                    1 24'Subfile Program with Update     '
A                                      DSPATR(HI)
A                                    4  2'Position to Last Name . . .'
A            PTNAME        20A  B  4 30CHECK(LC)
A                                    9  2'Opt'
A                                      DSPATR(HI)
A                                    6  2'Type options, press Enter.'
A                                      COLOR(BLU)
A                                    7  4'2=Change'
A                                      COLOR(BLU)
A                                    7 19'4=Delete'
A                                      COLOR(BLU)
A                                    7 34'5=Display'
A                                      COLOR(BLU)
A*
A          R PANEL1
A                                    1  2'SFL004RG'
A            MODE           6   O  2  2DSPATR(HI)
A                                    1 24'Subfile Program with Update     '
A                                      DSPATR(HI)
A                                    1 71DATE
A                                      EDTCDE(Y)
A                                    2 71TIME
A            DBIDNM    R        O  4 23REFFLD(PFR/DBIDNM *LIBL/SFL001PF)
A                                      DSPATR(HI)
A            DBFNAM    R        B  6 23REFFLD(PFR/DBFNAM *LIBL/SFL001PF)
A                                      CHECK(LC)
A            DBLNAM    R        B  8 23REFFLD(PFR/DBLNAM *LIBL/SFL001PF)
A                                      CHECK(LC)
A            DBMINI    R        B 10 23REFFLD(PFR/DBMINI *LIBL/SFL001PF)
A                                      CHECK(LC)
A            DBNNAM    R        B 12 23REFFLD(PFR/DBNNAM *LIBL/SFL001PF)
A                                      CHECK(LC)
A            DBADD1    R        B 14 23REFFLD(PFR/DBADD1 *LIBL/SFL001PF)
A                                      CHECK(LC)
A            DBADD2    R        B 16 23REFFLD(PFR/DBADD2 *LIBL/SFL001PF)
A                                      CHECK(LC)
A            DBADD3    R        B 18 23REFFLD(PFR/DBADD3 *LIBL/SFL001PF)
A                                      CHECK(LC)
A                                  23  2'F3=Exit'
A                                      COLOR(BLU)
A                                  23 12'F12=Cancel'
```

*SFL011DF: DDS Using the Data Queue Technique (continued)*

```
A                                       COLOR(BLU)
A                                    4  3'Customer Number . :'
A                                    6  3'First Name. . . . .'
A                                    8  3'Last Name . . . . .'
A                                   10  3'Middle Initial. . .'
A                                   12  3'Nick Name . . . . .'
A                                   14  3'Address Line 1. . .'
A                                   16  3'Address Line 2. . .'
A                                   18  3'Address Line 3. . .'
A          R PANEL2
A*
A                                    1  2'SFL004RG'
A            MODE           6   O  2  2DSPATR(HI)
A                                    1 24'Subfile Program with Update     '
A                                       DSPATR(HI)
A                                    1 71DATE
A                                       EDTCDE(Y)
A                                    2 71TIME
A            DBIDNM    R        O  4 2OREFFLD(PFR/DBIDNM *LIBL/SFL001PF)
A                                       DSPATR(HI)
A            DBFNAM    R        O  6 2OREFFLD(PFR/DBFNAM *LIBL/SFL001PF)
A                                       DSPATR(HI)
A            DBLNAM    R        O  8 2OREFFLD(PFR/DBLNAM *LIBL/SFL001PF)
A                                       DSPATR(HI)
A            DBMINI    R        O 10 2OREFFLD(PFR/DBMINI *LIBL/SFL001PF)
A                                       DSPATR(HI)
A            DBNNAM    R        O 12 2OREFFLD(PFR/DBNNAM *LIBL/SFL001PF)
A                                       DSPATR(HI)
A            DBADD1    R        O 14 2OREFFLD(PFR/DBADD1 *LIBL/SFL001PF)
A                                       DSPATR(HI)
A            DBADD2    R        O 16 2OREFFLD(PFR/DBADD2 *LIBL/SFL001PF)
A                                       DSPATR(HI)
A            DBADD3    R        O 18 2OREFFLD(PFR/DBADD3 *LIBL/SFL001PF)
A                                       DSPATR(HI)
A                                   23  2'F3=Exit'
A                                       COLOR(BLU)
A                                   23 12'F12=Cancel'
A                                       COLOR(BLU)
A                                    4  3'Customer Number:'
A                                    6  3'First Name . . :'
A                                    8  3'Last Name. . . :'
A                                   10  3'Middle Initial :'
A                                   12  3'Nick Name. . . :'
A                                   14  3'Address Line 1 :'
A                                   16  3'Address Line 2 :'
A                                   18  3'Address Line 3 :'
A*
A*
A          R WINDOW1                    SFL
A*
A            DBIDNM    R        H      REFFLD(PFR/DBIDNM *LIBL/SFL001PF)
```

### SFL011DF: DDS Using the Data Queue Technique (continued)

```
A            DBLNAM    R        O  6  2REFFLD(PFR/DBLNAM *LIBL/SFL001PF)
A            DBFNAM    R        O  6 26REFFLD(PFR/DBFNAM *LIBL/SFL001PF)
A*
A          R SF2CTL                    SFLCTL(WINDOW1)
A*
A                                      SFLDSP
A N41                                  SFLDSPCTL
A  41                                  SFLCLR
A N41                                  SFLEND(*MORE)
A                                      SFLSIZ(0009)
A                                      SFLPAG(0008)
A                                      WINDOW(4 10 16 52)
A            RRN2           4S 0H
A                                   5  2'Last Name'
A                                      DSPATR(HI)
A                                   5 26'First Name'
A                                      DSPATR(HI)
A                                   2  2'Press ENTER to confirm your choice-
A                                      s for delete.'
A                                      COLOR(BLU)
A                                   3  2'Press F12=Cancel to return to chan-
A                                      ge your choices.'
A                                      COLOR(BLU)
A*
A          R FKEY1
A*
A                                  23  2'F3=Exit'
A                                      COLOR(BLU)
A                                     +3'F6=Add'
A                                      COLOR(BLU)
A                                     +3'F12=Cancel'
A                                      COLOR(BLU)
A*
A          R FKEY2
A*
A                                  23  2'F3=Exit'
A                                      COLOR(BLU)
A                                     +3'F12=Cancel'
A                                      COLOR(BLU)
```

## SFL011RG: RPG Program Using the Data Queue Technique

```
     *
     *  To compile:
     *
     *              CRTRPGPGM PGM(XXX/SFL011RG) SRCFILE(XXX/QRPGLESRC)
     *
     *=============================================================================>
```

*SFL011RG: RPG Program Using the Data Queue Technique (continued)*

```
     Fsfl011df  cf   e             workstn
     F                                       sfile(sfl1:rrn1)
     F                                       sfile(window1:rrn2)
     F                                       infds(info)
     Fsfl001lf  if   e           k disk    rename(pfr:lfr)
     Fsfl001pf  uf a e           k disk
      *
      * Information data structure to hold attention indicator byte.
      *
     Dinfo             ds
     D cfkey                  369    369
      *
      * Constants and stand alone fields
      *
     Dexit             C                   const(X'33')
     Dcancel           C                   const(X'3C')
     Dadd              C                   const(X'36')
     Denter            C                   const(X'F1')
     Drollup           C                   const(X'F5')
     Drolldn           C                   const(X'F4')
     Dsflpag           C                   const(12)
     Dsflpag_plus_1    C                   const(13)
     Dqlen             C                   const(256)
     Ddisplay          C                   const('5')
     Dchange           C                   const('2')
     Ddelete           C                   const('4')

     Dlstrrn           S               4  0 inz(0)
     Dlstrrn2          S               4  0 inz(0)
     Dcount            S               4  0 inz(0)
     Dnew_id           S                    like(dbidnm)
     Dsavlnam          S                    like(dblnam)
     Dsavfnam          S                    like(dbfnam)
      *
      * Data Queue variables
      *
     Dlib              S              10    inz('QTEMP')
     Dqueue            S              10    inz('QUEUE1')
     Dlen              S               5  0 inz(256)
     Dkeyln            S               3  0 inz(7)
     Dwait             S               5  0 inz(0)
     Dsndlen           S               3  0 inz(0)
     Dorder            S               2    inz('EQ')
     Dsndr             S              10    inz('         ')
      *
      * Data structure to be loaded to data queue.
      *
     D data            DS
     D  option                        1
     D  dbidnm                 2      8s 0
     D    key                  2      8s 0
```

*SFL011RG: RPG Program Using the Data Queue Technique (continued)*

```
D  filler                 9    256    inz(*blanks)
 *
 *
D
 *
 ****************************************************************
 *  Main Routine
 ****************************************************************
 *
C     *loval        setll     sfl001lf
C                   exsr      sflbld
 *
C                   dou       cfkey = exit
 *
C                   write     fkey1
C                   exfmt     sf1ctl
 *
C                   select
 *
 * If ENTER key is pressed and position-to non blank,
 * reposition the subile to closet to what was entered.
 *
C                   when      (cfkey = enter) and (ptname  *blanks)
C                   exsr      addque
C     ptname        setll     sfl001lf
C                   exsr      sflbld
C                   clear                   ptname
 *
 * If ENTER key is pressed and position-to is blank,
 * process screen to interrogate options selected by user
 *
C                   when      (cfkey = enter) and (ptname = *blanks)
C                   exsr      addque
C                   exsr      prcsfl
C     savkey        setll     sfl001lf
C                   exsr      sflbld
 *
 * Roll up - load the data Q's before loading subfile
 *
C                   when      (cfkey = rollup) and (not *in90)
C                   exsr      addque
C                   exsr      sflbld
 *
 * User presses F6, throw the add screen, clear, and rebuild subfile
 *
C                   when      cfkey = add
C                   movel(p)  'Add   '      mode
C                   exsr      addque
C                   exsr      addrcd
C     dblnam        setll     sfl001lf
C                   exsr      sflbld
```

*SFL011RG: RPG Program Using the Data Queue Technique (continued)*

```
 *
 * Roll down - load the data Q's before loading the subfile.
 *
C                   when      (cfkey = rolldn) and (not *in32)
C                   exsr      addque
C                   exsr      goback
C                   exsr      sflbld
 *
C                   when      *inkl
C                   leave
 *
C                   endsl
 *
C                   enddo
 *
C                   eval      *inlr = *on
 *
 ****************************************************************
 *  ADDQUE - Add subfile data to Data Queues
 ****************************************************************
 *
C     addque        begsr
 *
 * Read the changed subfile records and write them to the data Q's
 * The first data queue is keyed by whatever the unique key of the file
 * is.  If no unique key in the file, use the relative record number. This
 * queue is used to save options selected on a specific subfile line.  The
 * second queue is keyed by option, and is used to process like options
 * together when the enter key is selected
 *
C                   readc     sfl1
 *
C                   dow       not %eof
 *
C                   eval      len = qlen
 *
C                   call      'QSNDDTAQ'
C                   parm                    queue
C                   parm                    lib
C                   parm                    len
C                   parm                    data
C                   parm                    keyln
C                   parm                    key
 *
C                   readc     sfl1
C                   enddo
 *
C                   endsr
 *
 ****************************************************************
 *  RCVQUE - Check DATAQUEUE before writing to subfile
```

*SFL011RG: RPG Program Using the Data Queue Technique (continued)*

```
     *****************************************************************
     *
    C     rcvque        begsr
     *
     * Read the data Q by the whatever the unique key from the
     * physical file to see if there is a saved option.  If so, display
     * the saved option when the subfile is displayed.
     *
    C                   eval      order = 'EQ'
     *
    C                   call      'QRCVDTAQ'
    C                   parm                    queue
    C                   parm                    lib
    C                   parm                    len
    C                   parm                    data
    C                   parm                    wait
    C                   parm                    order
    C                   parm                    keyln
    C                   parm                    key
    C                   parm                    sndlen
    C                   parm                    sndr
     *
    C                   if        len > *zero
    C                   eval      *in74 = *on
    C                   endif
     *
    C                   endsr
     *
     *****************************************************************
     *  PRCSFL - process the options taken in the subfile.
     *****************************************************************
     *
    C     prcsfl        begsr
     *
    C                   eval      *in41 = *on
    C                   write     sf2ctl
    C                   eval      *in41 = *off
    C                   eval      rrn2 = *zero
     *
     * Receive data queue records until the queue is empty LEN = 0
     *
    C                   eval      dbidnm = 1
    C                   eval      order = 'GE'
     *
    C                   dou       len = *zero
     *
    C                   call      'QRCVDTAQ'
    C                   parm                    queue
    C                   parm                    lib
    C                   parm                    len
    C                   parm                    data
```

*SFL011RG: RPG Program Using the Data Queue Technique (continued)*

```
C                   parm                    wait
C                   parm                    order
C                   parm                    keyln
C                   parm                    key
C                   parm                    sndlen
C                   parm                    sndr
 *
 *  If length is greater than zero, there was a record read.
 *  Process that record and receive from the second dataq to
 *  keep them in cinc.
 *
C                   if        len > *zero
 *
C                   select
 *
 *  process the edit program or subroutine
 *
C                   when      option = change
C                   movel(p)  'Update'      mode
C                   exsr      chgdtl
C                   if        (cfkey = exit) or (cfkey = cancel)
C                   leave
C                   endif
 *
 * when a 4 is entered write the record the the confirmation screen,
 * set on the SFLNXTCHG indicator to mark this record as changed,
 * and update the subfile.  I mark this record incase F12 is pressed
 * from the confirmation screen and the user wants to keep his
 * originally selected records
 *
C                   when      option = delete
C                   eval      rrn2 = rrn2 +1
C                   write     window1
 *
 *  process the display program or subroutine
 *
C                   when      option = display
C                   movel(p)  *blanks       mode
C     dbidnm        chain     sfl001pf
C                   exfmt     panel2
C                   if        (cfkey = exit) or (cfkey = cancel)
C                   leave
C                   endif
 *
C                   endsl
 *
C                   endif
 *
C                   enddo
 *
 *
```

*SFL011RG: RPG Program Using the Data Queue Technique (continued)*

```
     * If records were selected for delete (4), throw the subfile to
     * screen.  If enter is pressed execute the DLTRCD subroutine to
     * physically delete the records, clear, and rebuild the subfile
     * from the last deleted record (you can certainly position the
     * database file where ever you want)
     *
    C                   if        rrn2 > *zero
    C                   eval      lstrrn2 = rrn2
    C                   eval      rrn2 = 1
    C                   exfmt     sf2ctl
    C                   if        (cfkey  exit) and (cfkey  cancel)
    C                   exsr      dltrcd
    C     dblnam        setll     sfl001lf
    C                   endif
    C                   endif
     *
    C                   endsr
     *
     ****************************************************************
     *  SFLBLD - Build the List
     ****************************************************************
     *
    C     sflbld        begsr
     *
     *  Clear subfile
     *
    C                   eval      rrn1 = *zero
    C                   eval      *in31 = *on
    C                   write     sf1ctl
    C                   eval      *in31 = *off
     *
     * Load data to subfile
     *
    C                   do        sflpag
    C                   read      sfl001lf                               90
    C                   if        *in90
    C                   leave
    C                   endif
     *
    C                   eval      option = *blanks
    C                   exsr      rcvque
    C                   eval      rrn1 = rrn1 + 1
    C                   if        rrn1 = 1
    C                   eval      savlnam = dblnam
    C                   eval      savfnam = dbfnam
    C                   endif
    C                   write     sfl1
    C                   eval      *in74 = *off
    C                   enddo
     *
    C                   if        rrn1 = *zero
```

*SFL011RG: RPG Program Using the Data Queue Technique (continued)*

```
C                   eval      *in32 = *on
C                   endif
 *
C                   endsr
 *
 ****************************************************************
 *  GOBACK - page backward one page
 ****************************************************************
 *
C     goback        begsr
 *
C     savkey        setll     sfl001lf
 *
 * Re-position files for rolling backward.
 *
C                   do        sflpag_plus_1
C                   readp     sfl001lf
C                   if        %eof
C     *loval        setll     sfl001lf
C                   leave
C                   endif
 *
C                   enddo
 *
C                   endsr
 *
 ****************************************************************
 *  CHGDTL - allow user to change data
 ****************************************************************
 *
C     chgdtl        begsr
 *
 * chain to data file using selected subfile record
 *
C     dbidnm        chain     sfl001pf
 *
 * If the record is found (it better be), throw the change screen.
 * If F3 or F12 is pressed, do not update the data file
 *
C                   if        %found
C                   exfmt     panel1

C                   if        (cfkey  exit) and (cfkey  cancel)
C                   update    pfr
C                   endif

C                   endif

C                   endsr
 *
 ****************************************************************
```

*SFL011RG: RPG Program Using the Data Queue Technique (continued)*

```
     *  ADDRCD - allow user to add data
     *****************************************************************
     *
    C     addrcd        begsr
     *
     * set to last record in the the file to get the last ID number
     *
    C     *hival        setgt     sfl001pf
    C                   readp     sfl001pf
     *
     * set a new unique ID and throw the screen
     *
    C                   if        not %eof
    C                   eval      new_id = dbidnm + 1
    C                   clear                 pfr
    C                   eval      dbidnm = new_id
    C                   exfmt     panel1
     *
     * add a new record if the pressed key was not F3 or F12
     *
    C                   if        (cfkey  exit) and (cfkey  cancel)
    C                   write     pfr
    C                   endif

    C                   endif

    C                   endsr
     *
     *****************************************************************
     *  DLTRCD - delete records
     *****************************************************************
     *
    C     dltrcd        begsr
     *
     * read all the records in the confirmation subfile
     * and delete them from the data base file
     *
    C                   do        lstrrn2       count
    C     count         chain     window1
    C                   if        %found
    C     dbidnm        delete    pfr                                99
    C                   endif
    C                   enddo

    C                   endsr
     *
     *
     *
    C     savkey        klist
    C                   kfld                    savlnam
    C                   kfld                    savfnam
```