RPGIV
Template Documentation

USER'S GUIDE
Version 2.0 Revised May 19, 2004

ProData Computer Services, Inc

2908 South 160th Street
Omaha, NE 68130
(402) 697-7575
(800) 228-6318
(402) 697-7576 Fax

Software License Agreement

This software system consists of computer software and documentation. It contains trade secrets and confidential information which are proprietary to ProData Computer Servces, Inc. ("ProData"). Its use or disclosure in whole or in part without the express written permission of ProData is prohibited.

Before You Start

The templates that are provided will cover many of your interactive programming needs but they may not cover them all. What they do provide is a structured programming base for standardizing your application programs. Templates are like any other tools if you use them for their designed purpose they will serve you well for many years. If you only use them from time to time the full benefits of structured template programming may not be achieved.

Versions & Releases

The templates are available on the iSeries 400 for operation system V4R2 and above. The templates have been developed utilizing RPG/ILE and CLP/ILE. Be sure you have the correct versions for your machine before you start.

How to get the most from templates

Templates can save a company hundreds of programming hours over the span of a single project. Our suggestion is to start out small by creating a few programs using the templates. Learn how they function and what the differences are between them. Create a template library on your system where the templates can be stored. This will allow your custom templates and the original templates to be stored in one standard library.

What about my standards?

If your standards are different than the ones used in the template programs what should you do? Create a library on your system to house the templates in the form they currently exist. Create your template library, and copy the original templates into that library. You can now modify the programs to reflect your shops standards and still get the benefit of using proven templates.

I don't need all of the templates subroutines all of the time do I?

The templates offer many options that you may not use in you current applications. If you don't want some of the functionality, remove those pieces when you create your program. Our suggestion is to leave the routines in the original programs so that in the future if you can take advantage of these built in functions.

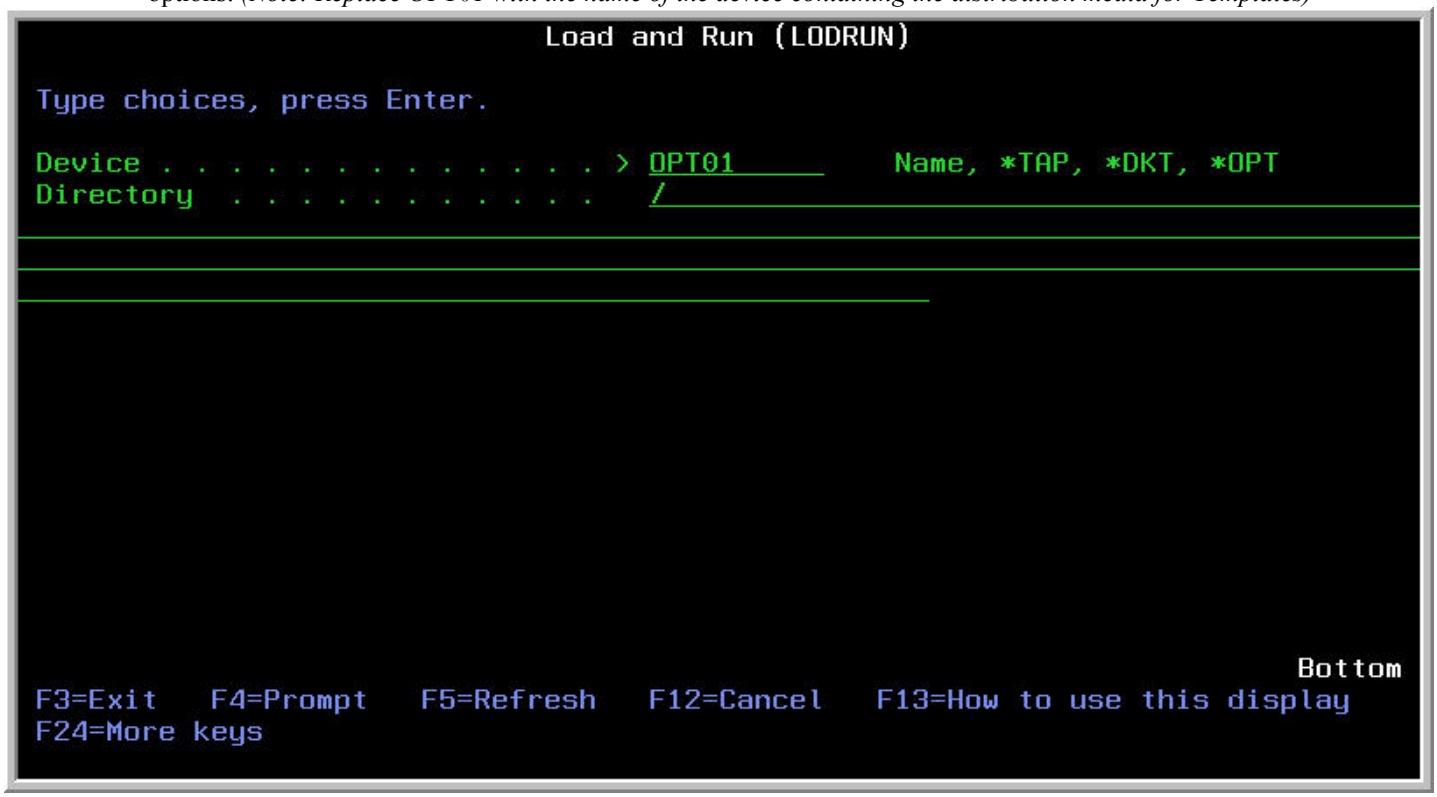
Installing, Saving, and Deleting Templates

Installing Templates

Use the Restore Library (RSTLIB) command to perform an initial install of the Templates. Follow the four-step installation process.

1. Sign on the iSeries/400 system as security officer (QSECOFR)

2. Install the Templates library with the Load Run command. Type LODRUN and press the F4 to prompt the following options: *(Note: Replace OPT01 with the name of the device containing the distribution media for Templates)*



Saving the Templates

Use the Save Library (SAVLIB) command to save the Templates. Follow the four-step save process.

1. Sign on the iSeries 400 system as security officer (QSECOFR)

2. Use the Initialize Tape (INZTAP) command to initialize a tape to hold saved version of the Templates.

3. Save the Templates with the SAVLIB Command. Type SAVLIB and press F4 to prompt the following options:
   (Note: Replace TAP01 with the name of the device containing the initialized tape)

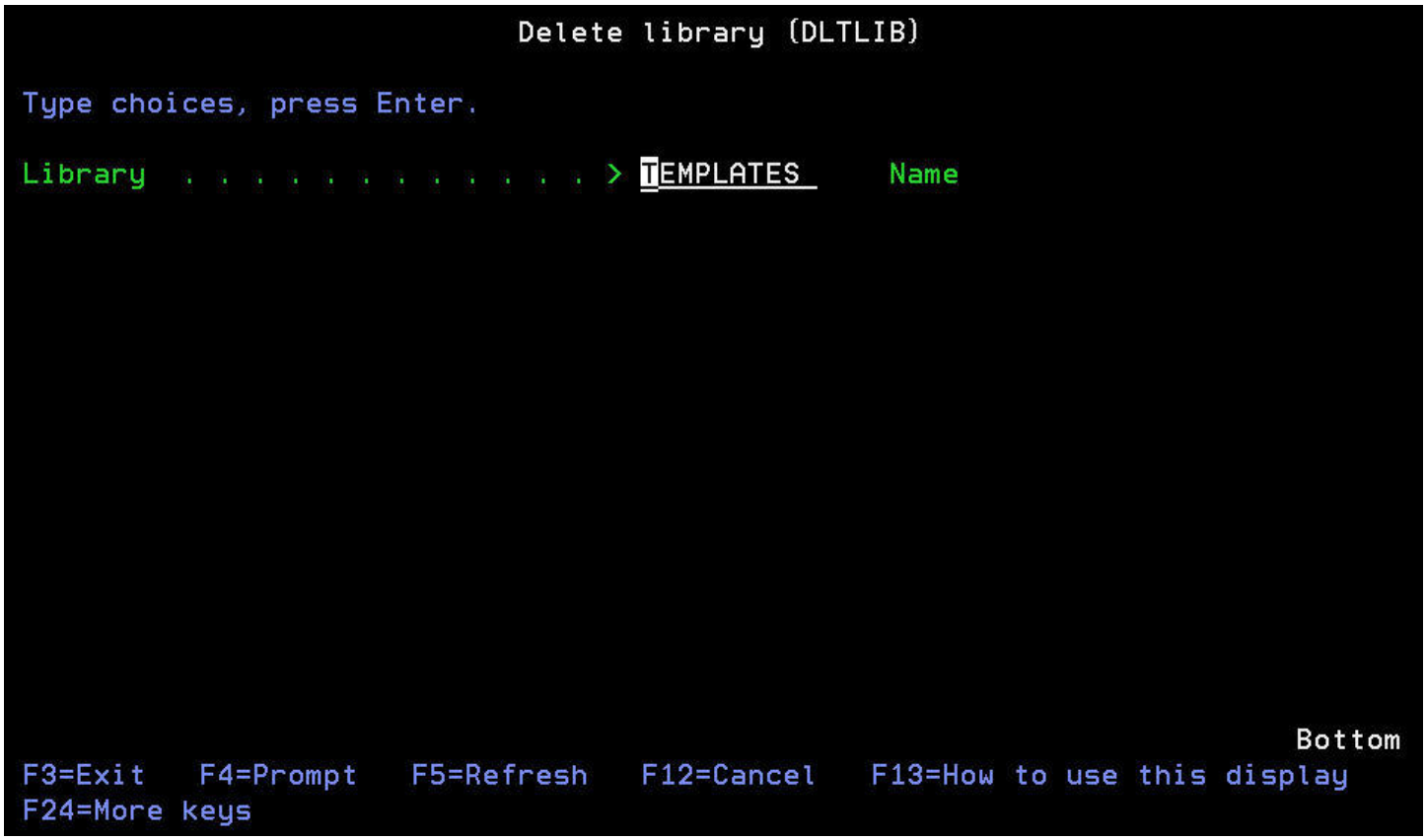
4. Sign off the AS/400 System.

Deleting the Templates

Use the Delete Library (DLTLIB) command to delete the Templates. Follow the Three-step process.

1. Sign on the AS/400 system as security officer (QSECOFR)

2. Delete the Templates with the DLTLIB command. Type DLTLIB and press F4 to prompt for the following options:

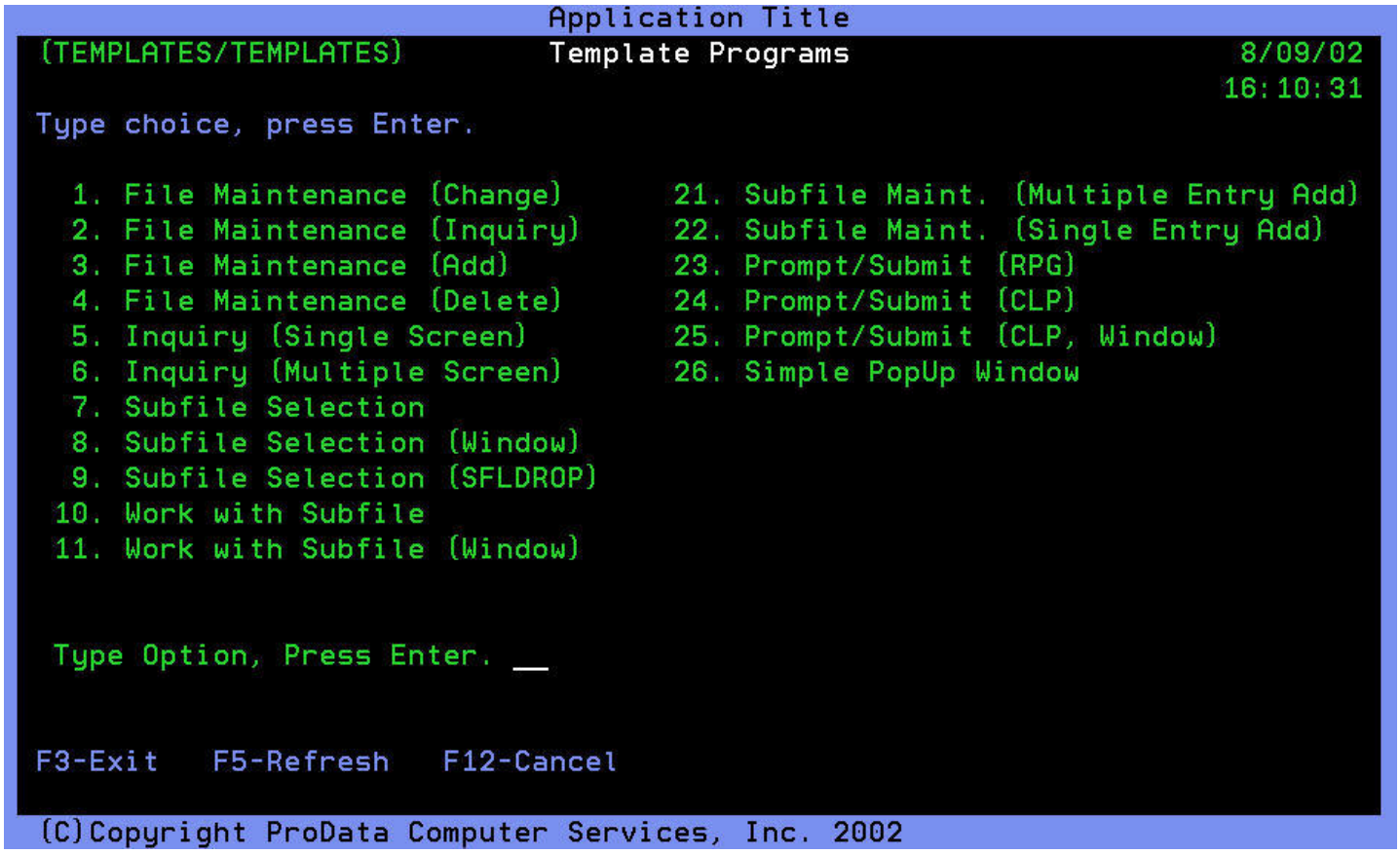


3. Sign off the AS/400 System.

Templates

The template programs that are provided are not just source code that can be copied. The objects are also provided to allow the execution of the templates. This allows the programmer to run the templates in order to determine if it is the desired one to copy. The template menu can be executed by calling the "TEMPLATES" program in the templates library. The template programs can be run by entering the option and pressing enter. The templates menu is displayed below.

Command
CALL TEMPLATES/TEMPLATES



File Maintenance (Change)

The file maintenance template program can run in 4 separate modes. The operation of the file maintenance program depends on the mode that is passed in on the entry parameters. The available modes are *ADD, *CHANGE, *DELETE, and *INQUIRY. The other parameters that can be passed in are the keys to the file that is being processed. This allows the maintenance program to position to the correct record for any of the available modes. Under the *CHANGE mode the file maintenance program allows the change of the record information contained in the file.

The file maintenance program also allows you to roll through the records in the file. The page up or page down keys allows the user to update several records without exiting out of change mode.

User edits can be placed in the file maintenance program to check the validity or requirements of fields that need to be populated. The coding for the user edits must be done by the programmer and are not contained in the template programs. An example of this is a field such as state code. The programmer would check the state code to ensure its validity.

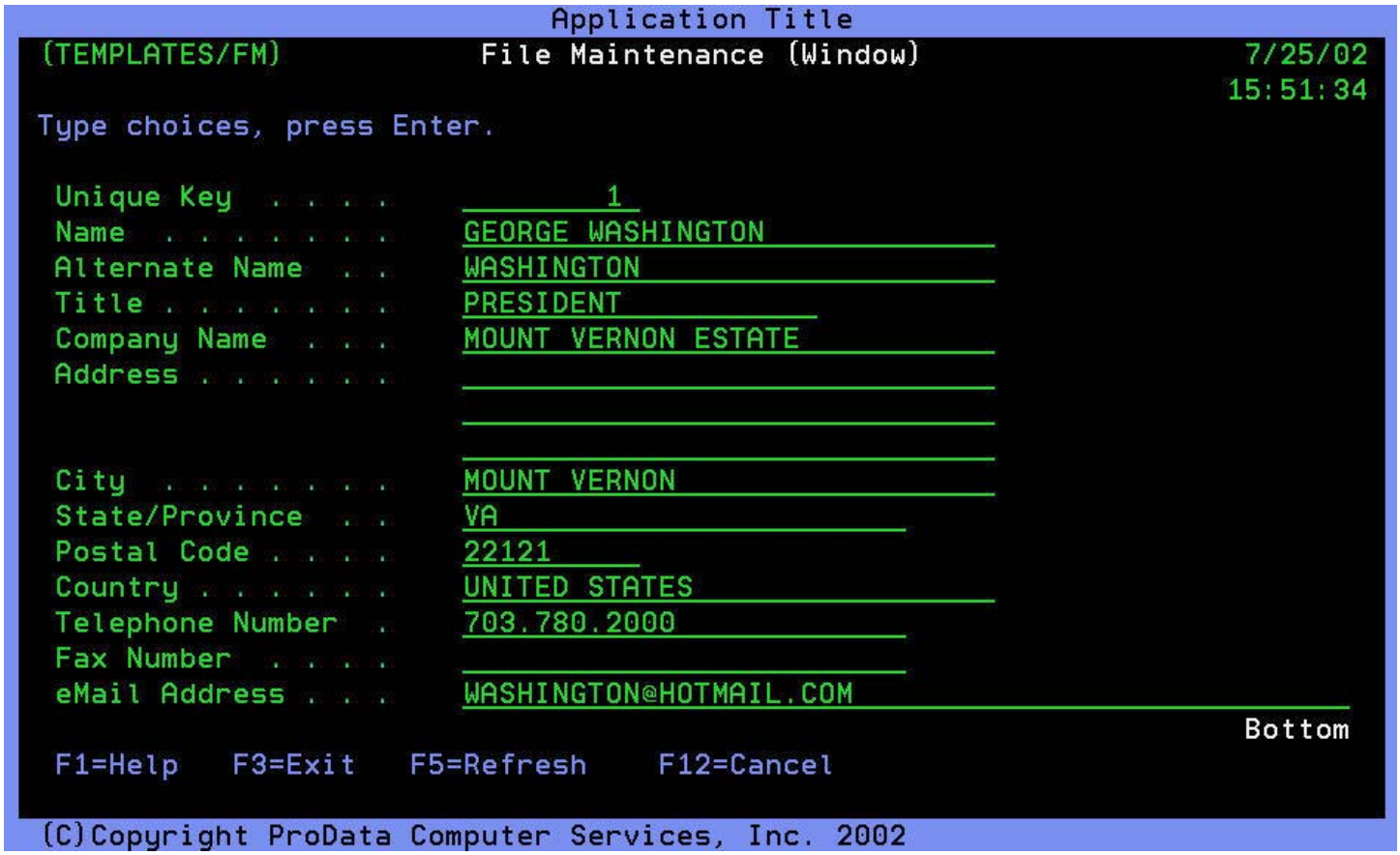
Command Keys **The command keys available in the File Maintenance program in *CHANGE mode, are described below.**

F1=Help — The help command key appears only for your help text implementation. The templates provide a simple window that appears when the Help key is pressed. You would have to modify the display file to include any help information you desire for your application.

F3=Exit — The exit command key exits the user from the program.

F5=Refresh — The refresh command key clears the screen and allows entry of a new key.

F12=Cancel — The cancel command key exits the user from the program.



File Maintenance (Inquiry)

The file maintenance inquiry program runs the same program as the file maintenance change program. The operation of the file maintenance program depends on the mode that is passed on the entry parameters. The available modes are *ADD, *CHANGE, *DELETE, and *INQUIRY. The other parameters that can be passed are the keys to the file being processed. This allows the maintenance program to position to the correct record for any of the available modes. Under the *INQUIRY mode the file maintenance program allows the user to display the file data.

The file maintenance program also allows you to roll through the records in the file. The page up or page down keys allows the user to display several records without exiting out of inquiry mode.

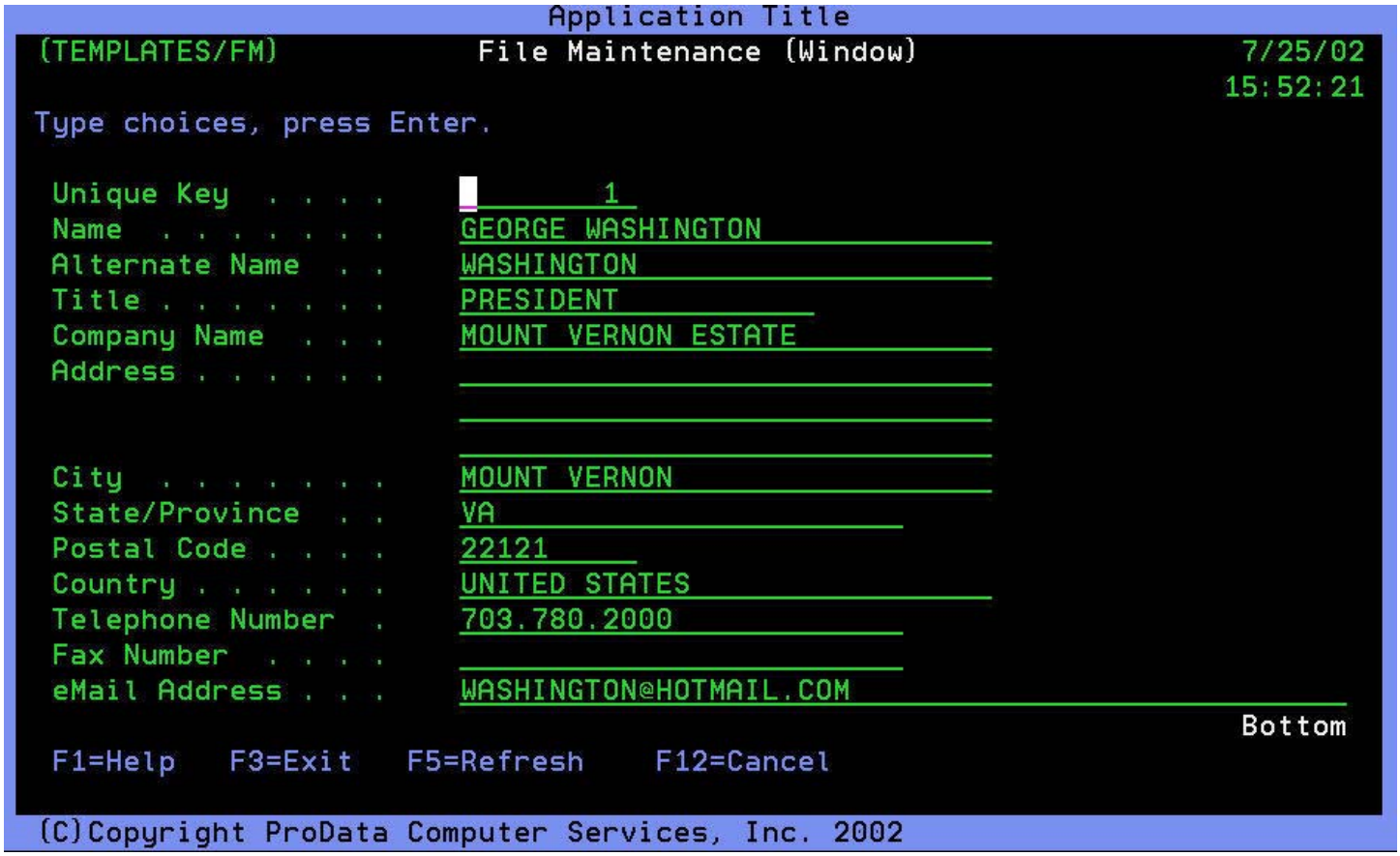
Command Keys **The command keys available in the File Maintenance program in *INQUIRY mode, are described below.**

F1=Help — The help command key appears only for your help text implementation. The templates provide a simple window that appears when the Help key is pressed. You would have to modify the display file to include any help information you desire for your application.

F3=Exit — The exit command key exits the user from the program.

F5=Refresh — The refresh command key clears the screen and allows entry of a new key.

F12=Cancel — The cancel command key exits the user from the program.



File Maintenance (Add)

The file maintenance add program runs the same program as the file maintenance change program. The operation of the file maintenance program depends on the mode that is passed on the entry parameters. The available modes are *ADD, *CHANGE, *DELETE, and *INQUIRY. The other parameters that can be passed are the keys to the file being processed. Under the *ADD mode the file maintenance program allows new records to be added to a file. If the record contains more fields than can be displayed on one screen the roll key can be press to displayed remaining fields.

If the roll keys are pressed in the *ADD mode the program will switch to *CHANGE mode and display the previous or next record.

User edits can be placed in the file maintenance program to check the validity or requirements of fields that need to be populated. The code for the user edits must be done by the programmer and are not contained in the template programs. An example of this is a field such as product number. The programmer would verify the validity of the product number to the product master file.

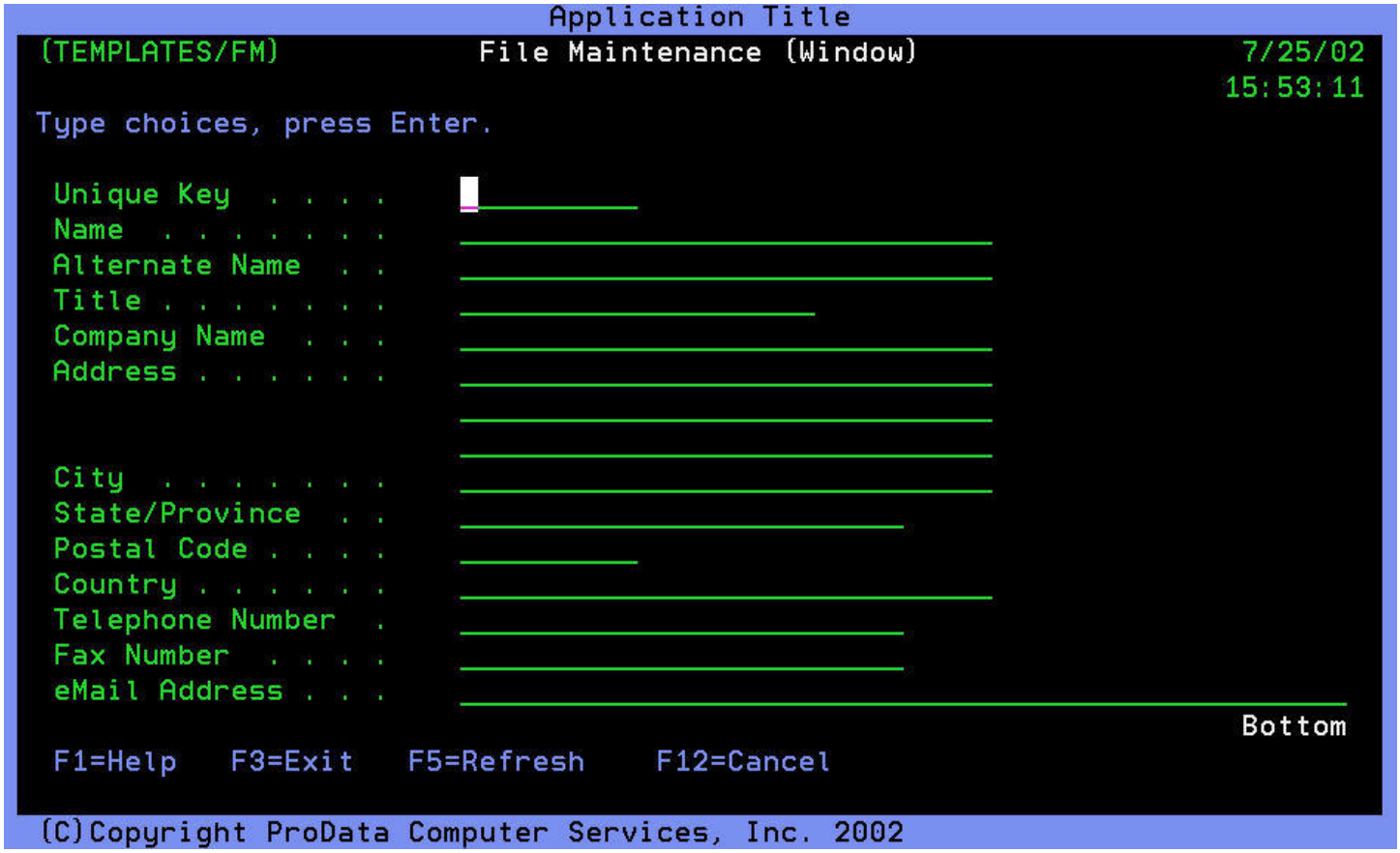
Command Keys **The command keys available in the File Maintenance program in *ADD mode, are described below.**

F1=Help — The help command key appears only for your help text implementation. The templates provide a simple window that appears when the Help key is pressed. You would have to modify the display file to include any help information you desire for your application.

F3=Exit — The exit command key exits the user from the program.

F5=Refresh — The refresh command key clears the screen and allows entry of a new key.

F12=Cancel — The cancel command key exits the user from the program.



File Maintenance (Delete)

The file maintenance add program runs through the same program as the file maintenance change program. The operation of the file maintenance program depends on the mode that is passed in on the entry parameters. The available modes are *ADD, *CHANGE, *DELETE, and *INQUIRY. The other parameters that can be passed in are the keys to the file that is being processed. This allows the maintenance to be positioned to the correct record at entry time. Under the *DELETE mode the file maintenance program allows the deletion of the record displayed. If the record contains more fields than can be displayed on one screen the roll key can be press to display remaining fields.

When the file maintenance program is running under the delete mode the record will be displayed with a message at the bottom of the screen instructing the user on how to perform the delete. The message that is displayed will appear as follows. "Press enter to delete record. Otherwise, press F12 to nullify." If the enter key is pressed the record will physically be deleted from the file. If F12 is pressed the program will exit the program.

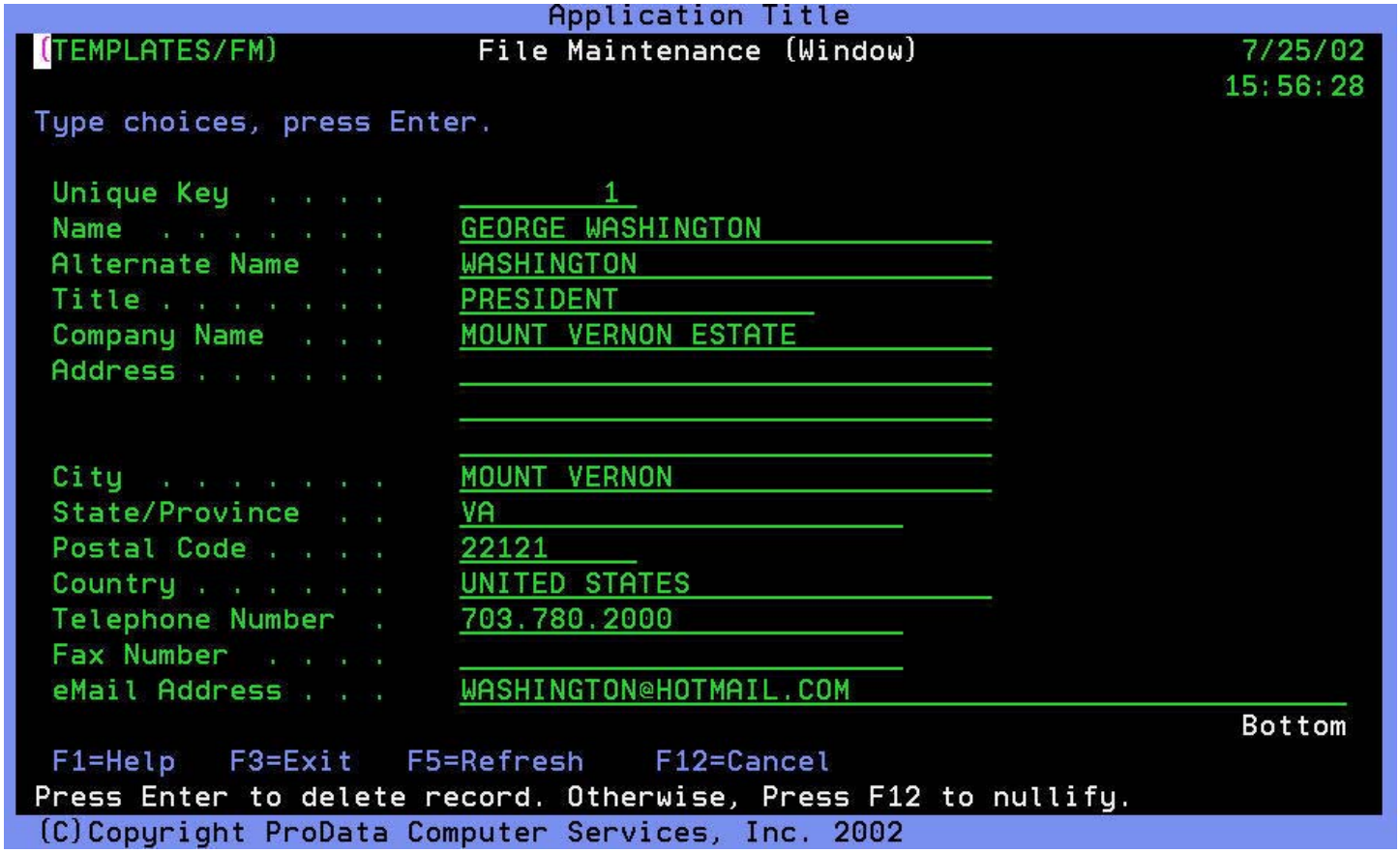
Command Keys **The command keys available in the File Maintenance program in *DELETE mode, are described below.**

F1=Help — The help command key appears only for your help text implementation. The templates provide a simple window that appears when the Help key is pressed. You would have to modify the display file to include any help information you desire for your application.

F3=Exit — The exit command key exits the user from the program.

F5=Refresh — The refresh command key clears the screen and allows entry of a new key.

F12=Cancel — The cancel command key exits the user from the program.



Inquiry (Single Screen)

The single screen inquiry program is used only for inquiry purposes. The entry parameters to the program are the keys to the record that is being displayed. If the parameters passed into the program are populated the record that corresponds to that key will be displayed. If the parameters are not populated the program will display the error "Record was not found". If the record being displayed contains more fields than can be displayed on one screen the roll key can be press to displayed the remaining fields.

The single screen inquiry program also allows you to roll through the records in the file. The roll forward or backwards allows the user to inquire several records without exiting the inquiry program.

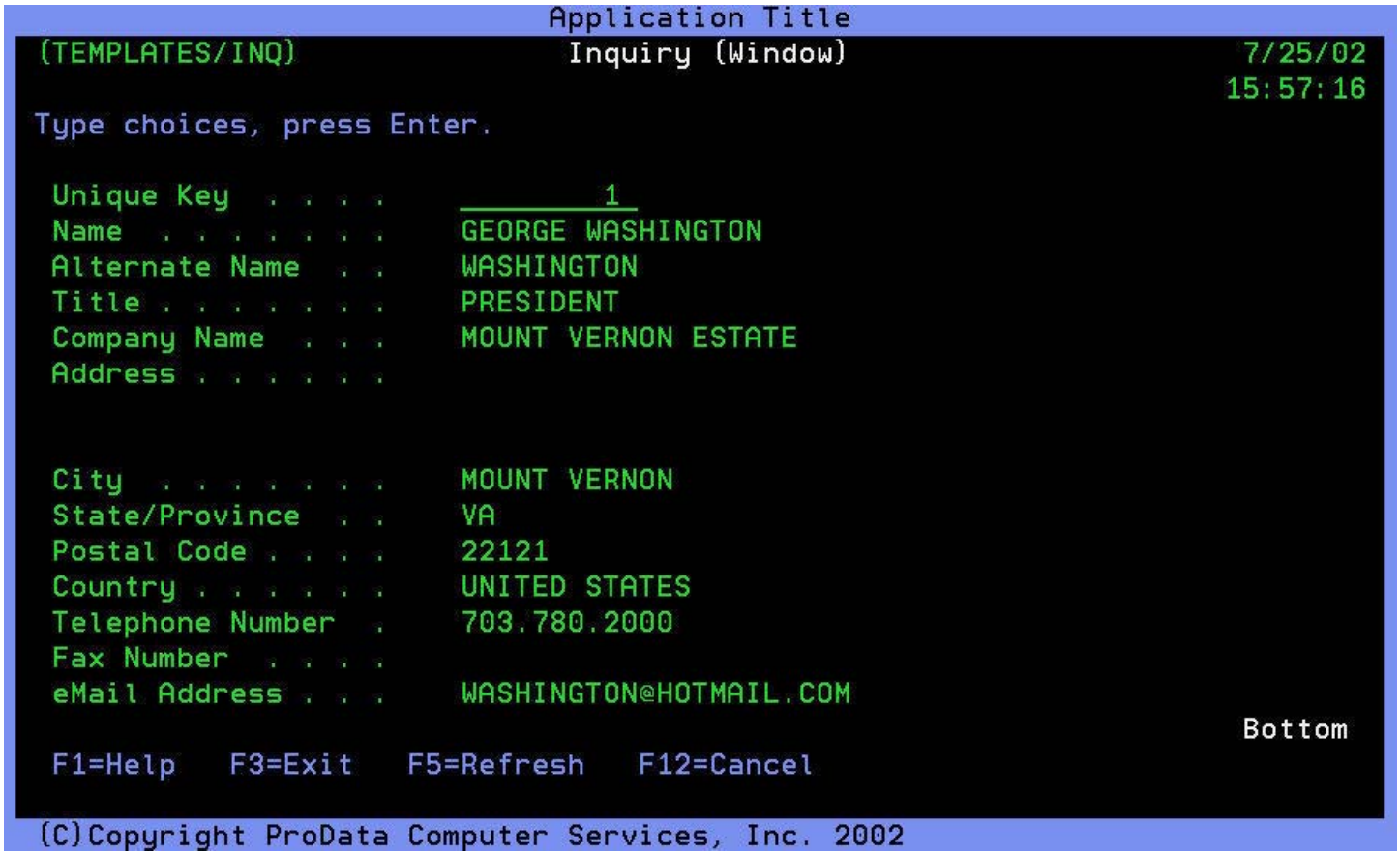
Command Keys **The command keys available in the Inquiry (Single Screen) program are described below.**

F1=Help — The help command key appears only for your help text implementation. The templates provide a simple window that appears when the Help key is pressed. You would have to modify the display file to include any help information you desire for your application.

F3=Exit — The exit command key exits the user from the program.

F5=Refresh — The refresh command key clears the screen and allows entry of a new key.

F12=Cancel — The cancel command key exits the user from the program.



Inquiry (Multiple Screen)

The multiple screen inquiry program is used only for inquiry purposes. The entry parametersto the program are the keys to the record that is being displayed. If the parameters passed into the program are populated the record that corresponds to that key will be displayed. If the parameters are not populated the program will display the error "Record was not found". If the record being displayed contains more fields than can be displayed on one screen the roll key can be press to display the remaining fields. The difference between the single and multiple screen inquiry is the fact that the second screen is displayed in a separate window. This program functions the same as the single screen inquiry only the appearance is different.

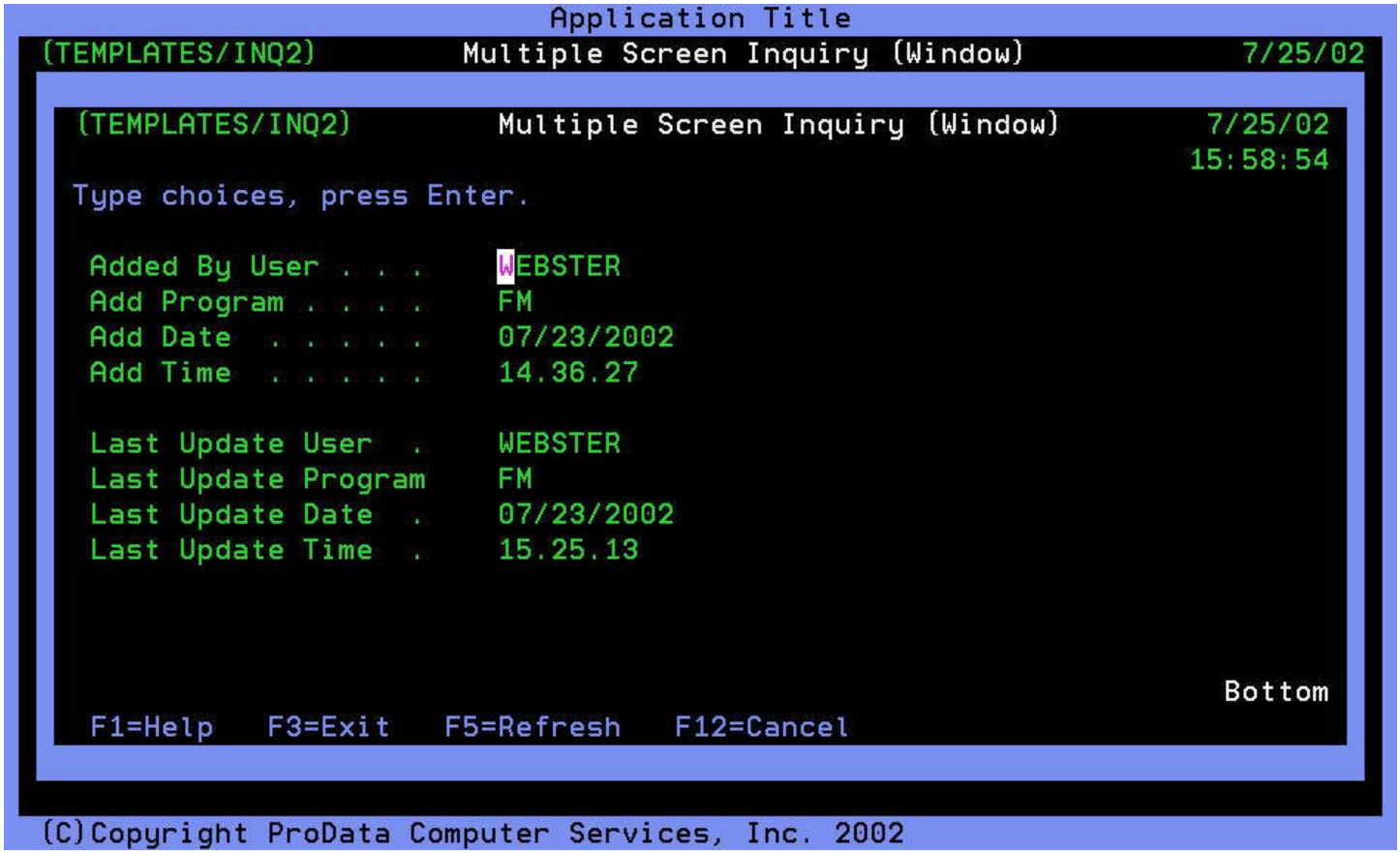
Command Keys **The command keys available in the Inquiry (Multiple Screen) program are described below.**

F1=Help — The help command key appears only for your help text implementation. The templates provide a simple window that appears when the Help key is pressed. You would have to modify the display file to include any help information you desire for your application.

F3=Exit — The exit command key exits the user from the program.

F5=Refresh — The refresh command key clears the screen and allows entry of a new key.

F12=Cancel — The cancel command key exits the user from the program.



Subfile Selection

The subfile selection screen can be used as a prompt screen for returning values to a program or as an informational prompt screen. When the record that is desired is selected, the program will return a value to the calling program that you wish to be returned. For example this program could be used as a prompt from a calling program to select the correct Unique Id. After selection this program will return, to the calling program, the Unique Id in the parameter fields.

Program Options **The options available in the Subfile Selection program are described below.**

1=Select — This option selects the record you have placed the option beside, then returns to the calling program, any fields you program to be returned.

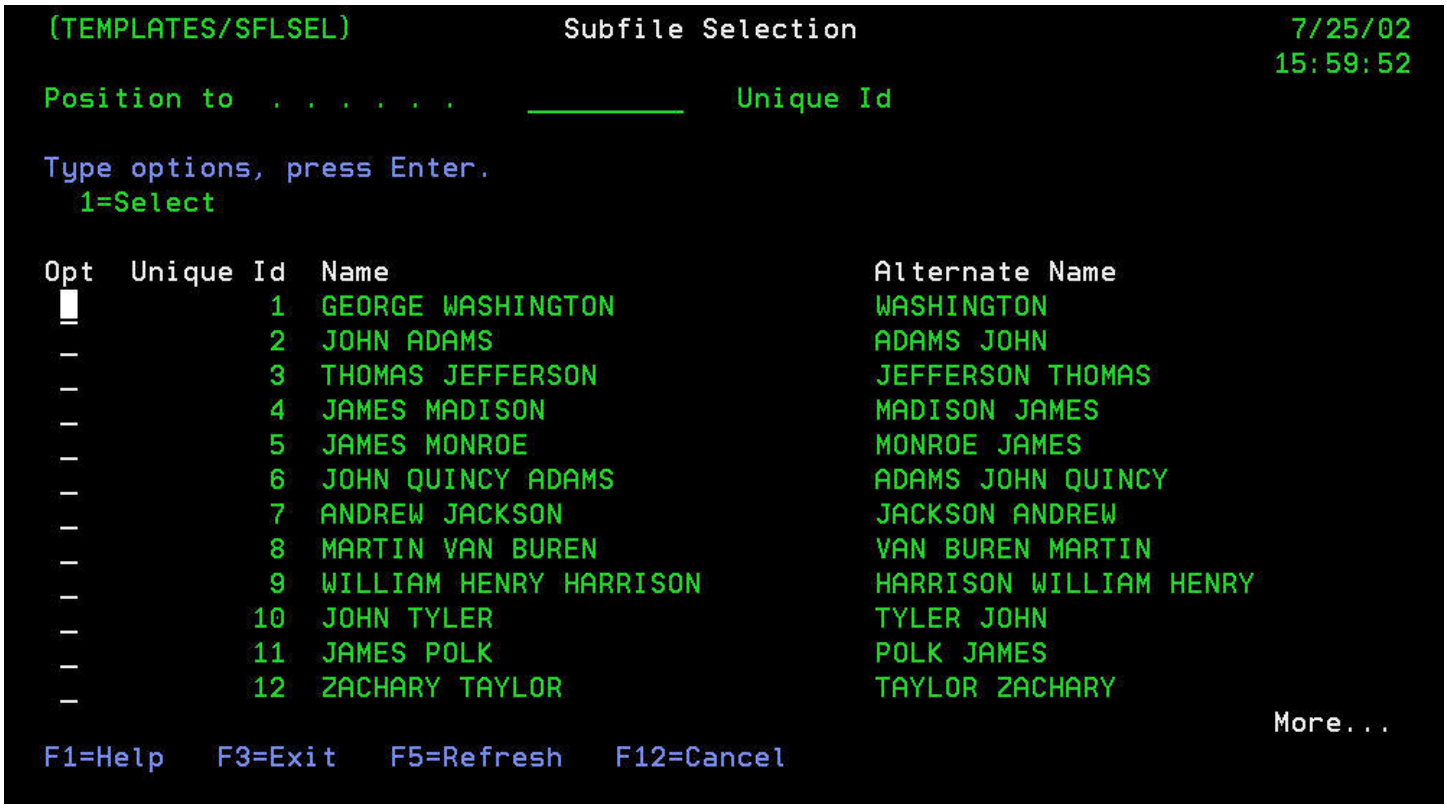
Command Keys **The command keys available in the Subfile Selection program are described below.**

F1=Help — The help command key appears only for your help text implementation. The templates provide a simple window that appears when the Help key is pressed. You would have to modify the display file to include any help information you desire for your application.

F3=Exit — The exit command key exits the user from the program.

F5=Refresh — The refresh command key clears the screen and allows entry of a new key.

F12=Cancel — The cancel command key exits the user from the program.



Subfile Selection (Window)

The subfile selection window can be used as a prompt screen for returning values to a program or as an informational prompt screen. When the desired record is selected the program will return a value to the calling program. For example this program could be used as a prompt from a calling program to select the correct Unique Id. After selection this program will return the Unique Id to the calling program.

Program Options **The options available in the Subfile Selection (Window) program are described below.**

1=Select — This option selects the record you have placed the option beside, then returns to the calling program, any fields you program to be returned.

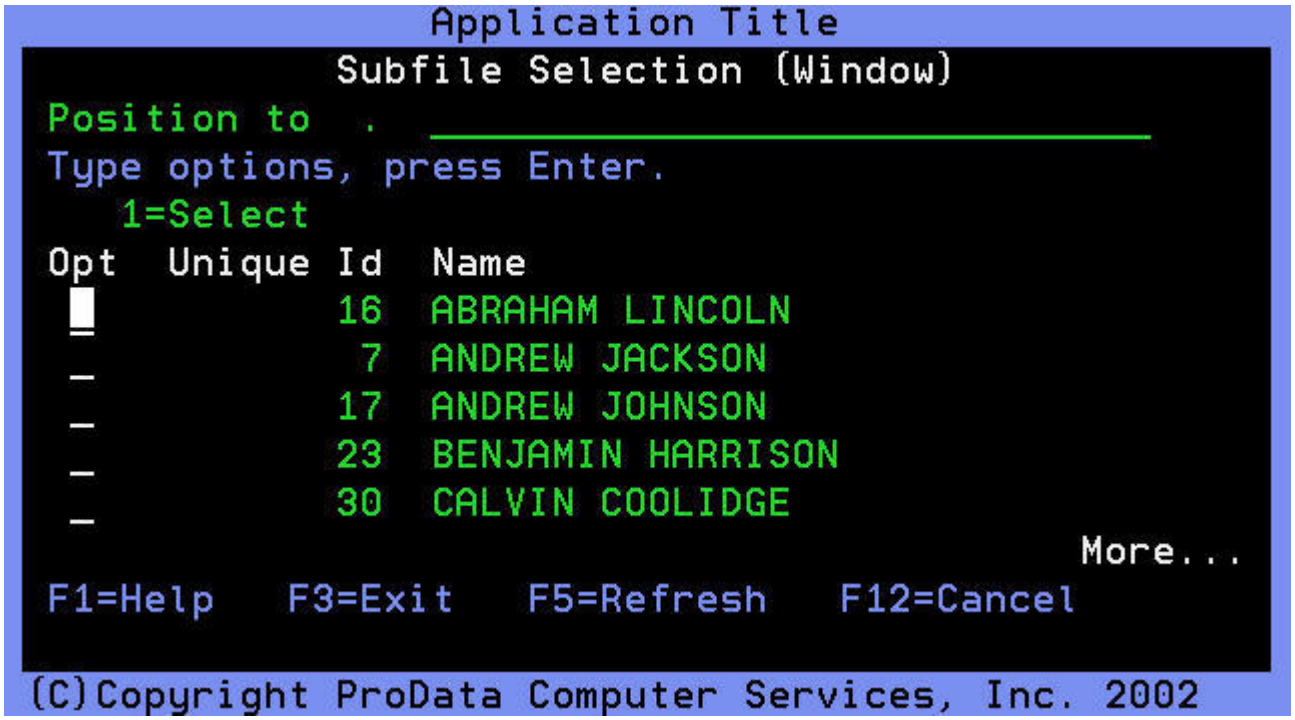
Command Keys **The command keys available in the Subfile Selection (Window) program are described below.**

F1=Help — The help command key appears only for your help text implementation. The templates provide a simple window that appears when the Help key is pressed. You would have to modify the display file to include any help information you desire for your application.

F3=Exit — The exit command key exits the user from the program.

F5=Refresh — The refresh command key clears the screen and allows entry of a new key.

F12=Cancel — The cancel command key exits the user from the program.



Work With Subfile

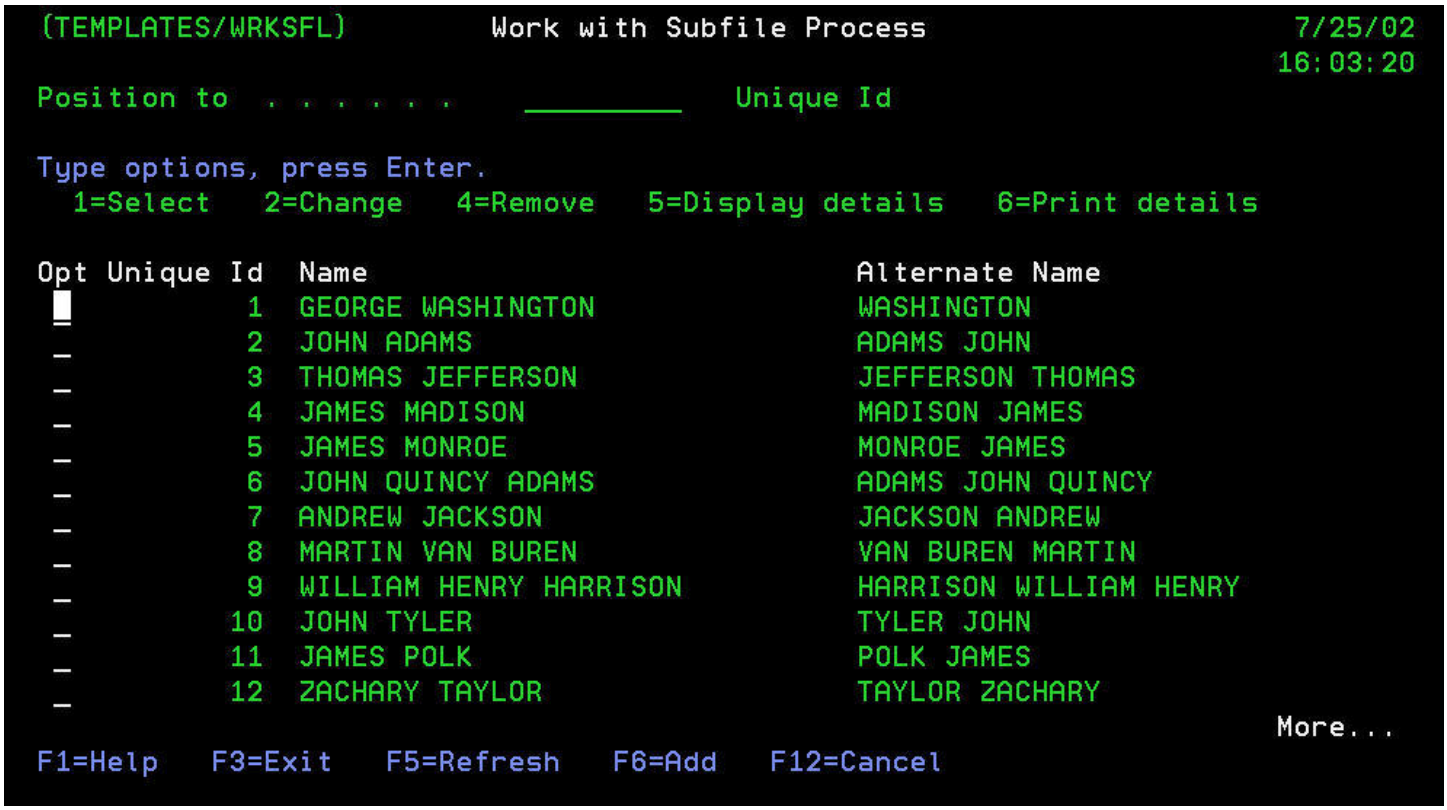
The Work with Subfile program can be used as a prompt screen for returning values to a program or as a interface to the File Maintenance & Inquiry programs.

Program Options **The options available in the Work with Subfile Process program are described below.**

1=Select	This option selects the record you have placed the option beside, then returns to the calling program, any fields you program to be returned.
2=Change	The change option will call the File Maintenance program with a parameter of *CHANGE and the key values for the record being processed.
4=Remove	The remove option will call the File Maintenance program with a parameter of *DELETE and the key values for the record being processed.
5=Display Details	The display details options will call the File Maintenance program with a parameter of *INQUIRY and the key values for the record being processed.
6=Print Details	The print details options will call your report program.

Command Keys **The command keys available in the Work with Subfile Process program are described below.**

F1=Help	The help command key appears only for your help text implementation. The templates provide a simple window that appears when the Help key is pressed. You would have to modify the display file to include any help information you desire for your application.
F3=Exit	The exit command key exits the user from the program.
F5=Refresh	The refresh command key clears the screen and allows entry of a new key.
F6=Add	The add command key call the File Maintenance program with a parameter of *ADD and the key values for the record being processed.
F12=Cancel	The cancel command key exits the user from the program.



Work With Subfile (Window)

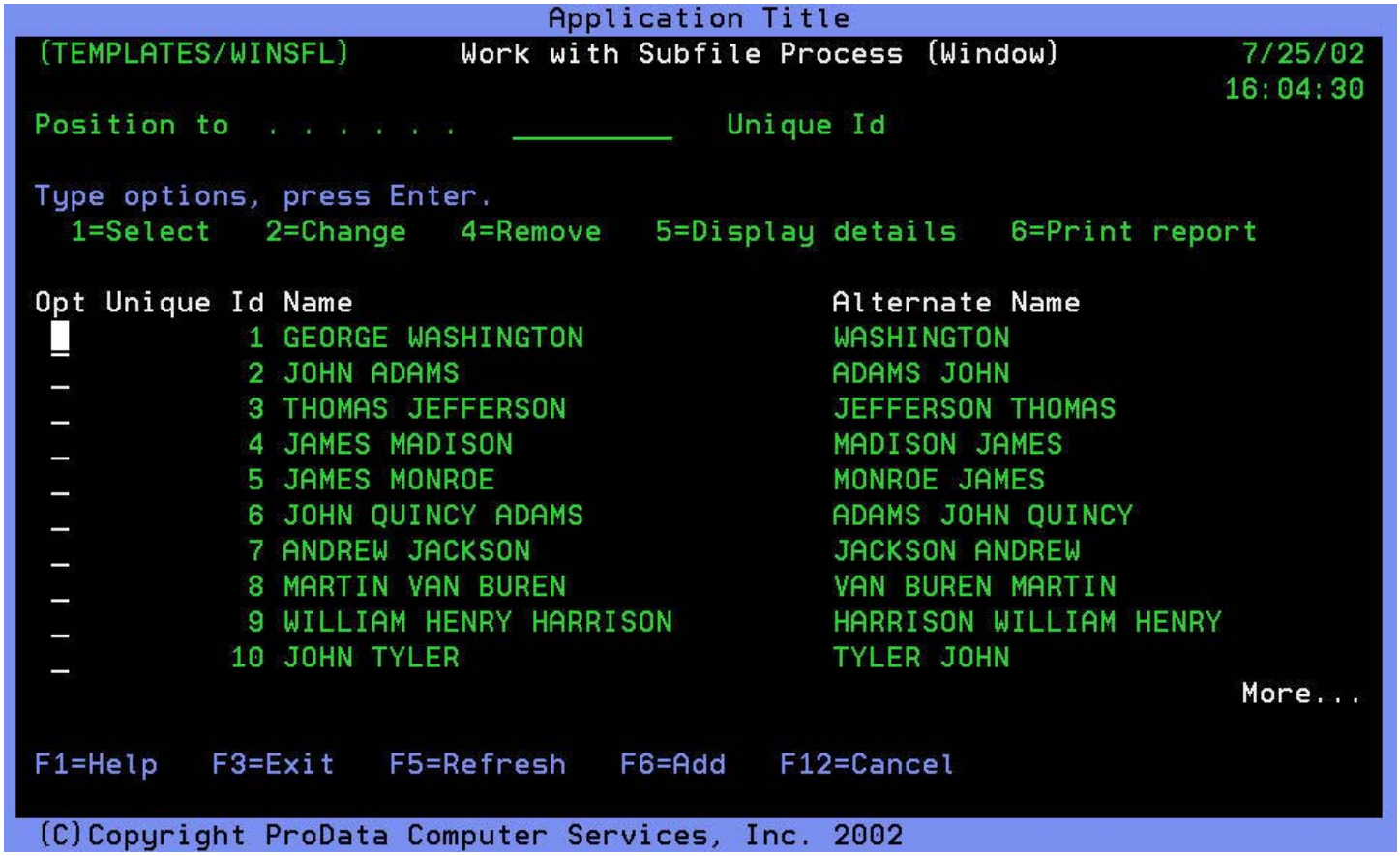
The subfile selection screen can be used as a prompt screen for returning values to a program or as a interface to the file maintenance and inquiry programs. The only difference between this program and the previously described work with subfile program is that the screen is wrapped with a window border.

Program Options **The options available in the Work with Subfile Process (Window) program are described below.**

1=Select	This option selects the record you have placed the option beside, then returns to the calling program, any fields you program to be returned.
2=Change	The change option will call the File Maintenance program with a parameter of *CHANGE and the key values for the record being processed.
4=Remove	The remove option will call the File Maintenance program with a parameter of *DELETE and the key values for the record being processed.
5=Display Details	The display details options will call the File Maintenance program with a parameter of *INQUIRY and the key values for the record being processed.
6=Print Details	The print details options will call your report program.

Command Keys **The command keys available in the Work with Subfile Process (Window) program are described below.**

F1=Help	The help command key appears only for your help text implementation. The templates provide a simple window that appears when the Help key is pressed. You would have to modify the display file to include any help information you desire for your application.
F3=Exit	The exit command key exits the user from the program.
F5=Refresh	The refresh command key clears the screen and allows entry of a new key.
F6=Add	The add command key call the File Maintenance program with a parameter of *ADD and the key values for the record being processed.
F12=Cancel	The cancel command key exits the user from the program.



Subfile Maintenance (Multiple Entry Add)

The Subfile Maintenance (Multiple Entry Add) screen is designed to allow maintenance to file records without the need of calling an additional maintenance program. The records can be updated directly from the initial subfile screen. The Subfile Maintenance (Multiple Entry Add) program allows the addition of new records by pressing a F6. The F6 key will display a blank subfile for the new record addition.

Program Options **The options available in the Subfile Maintenance (Multiple Entry Add) program are described below.**

4=Remove — The remove option will call the File Maintenance program with a parameter of *DELETE and the key values for the record being processed.

Command Keys **The command keys available in the Subfile Maintenance (Multiple Entry Add) program are described below.**

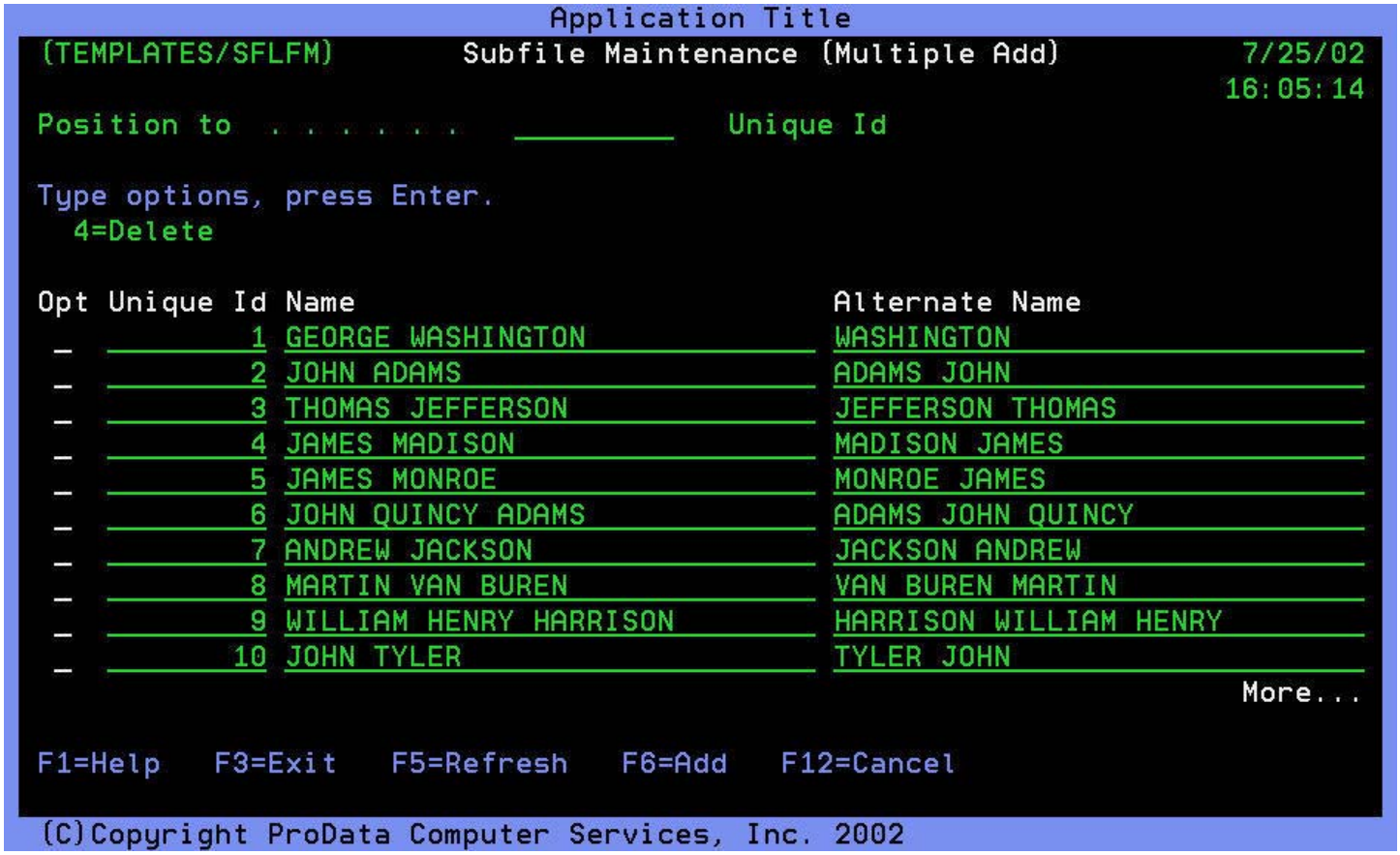
F1=Help — The help command key appears only for your help text implementation. The templates provide a simple window that appears when the Help key is pressed. You would have to modify the display file to include any help information you desire for your application.

F3=Exit — The exit command key exits the user from the program.

F5=Refresh — The refresh command key refreshes the contents of the screen.

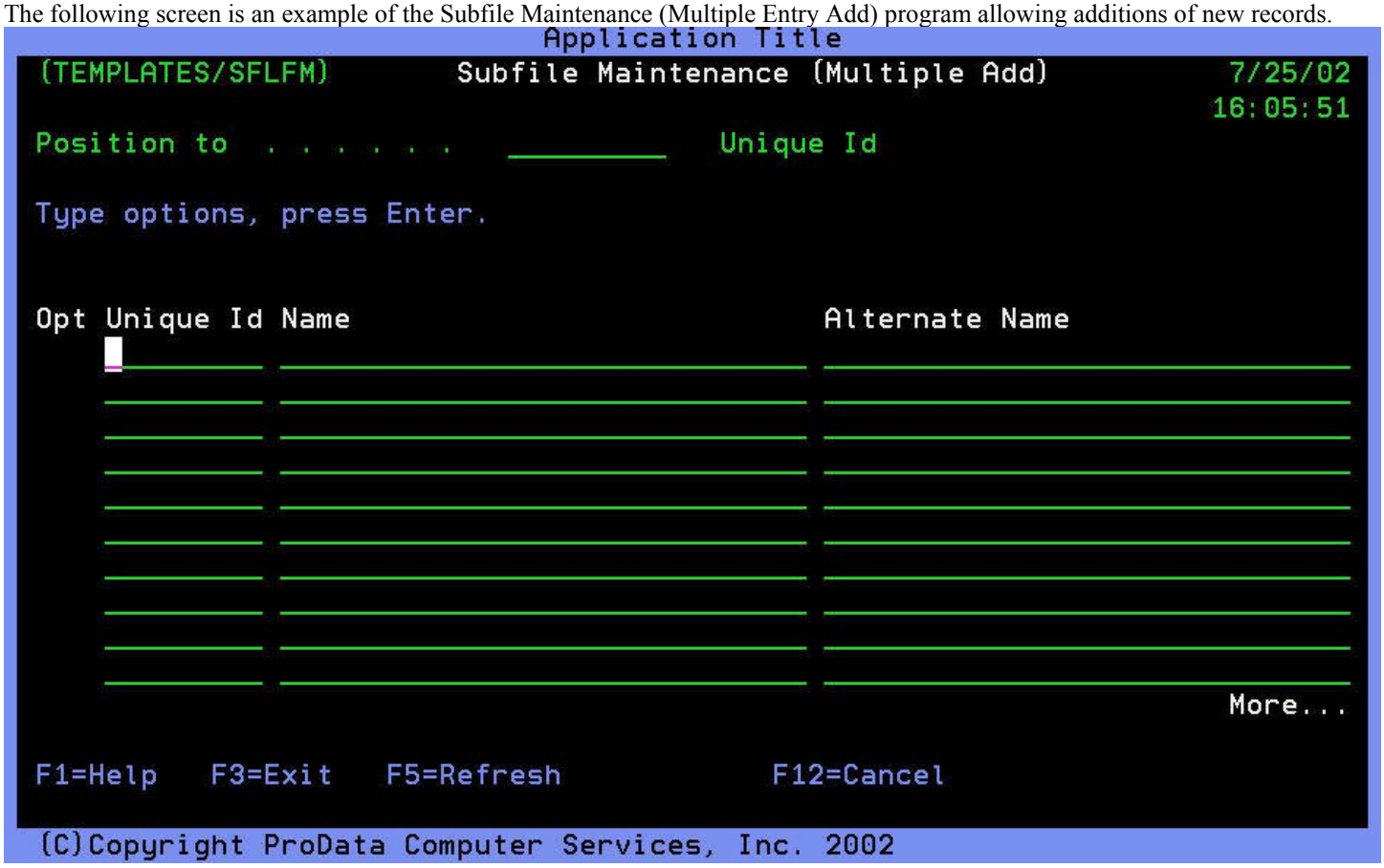
F6=Add — The add command key switches the program into a full screen add mode. A blank subfile is displayed, which will allow new record additions.

F12=Cancel — The cancel command key will cancel the Add or Delete action being processed. Otherwise, the cancel command key will exits the user from the program.



Subfile Maintenance (Multiple Entry Add) Continued…

The following screen is an example of the Subfile Maintenance (Multiple Entry Add) program allowing additions of new records.



Subfile Maintenance (Single Entry Add)

The Subfile Maintenance (Single Entry Add) screen is designed to allow maintenance to file records without the need of calling an additional maintenance program. The records can be updated directly from the initial subfile screen. The difference between the Multiple Entry Add and the Single Entry Add is the procedure used to add additional records. The Single Entry Add allows additions of new records from the initial subfile screen.

Program Options **The options available in the Subfile Maintenance (Single Entry Add) program are described below.**

4=Remove — The remove option will call the File Maintenance program with a parameter of *DELETE and the key values for the record being processed.

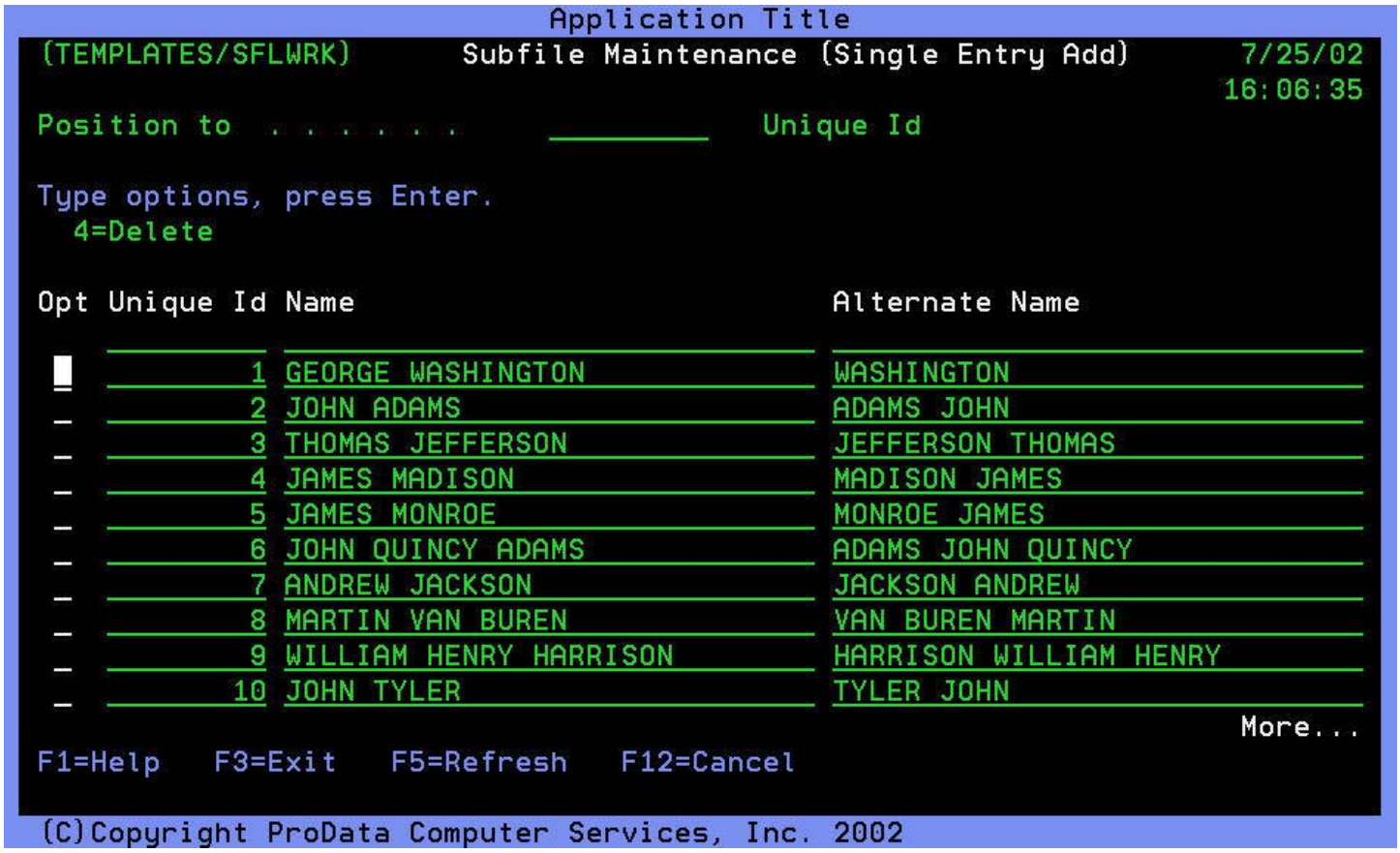
Command Keys **The command keys available in the Subfile Maintenance (Single Entry Add) program are described below.**

F1=Help — The help command key appears only for your help text implementation. The templates provide a simple window that appears when the Help key is pressed. You would have to modify the display file to include any help information you desire for your application.

F3=Exit — The exit command key exits the user from the program.

F5=Refresh — The refresh command key clears the screen and allows entry of a new key.

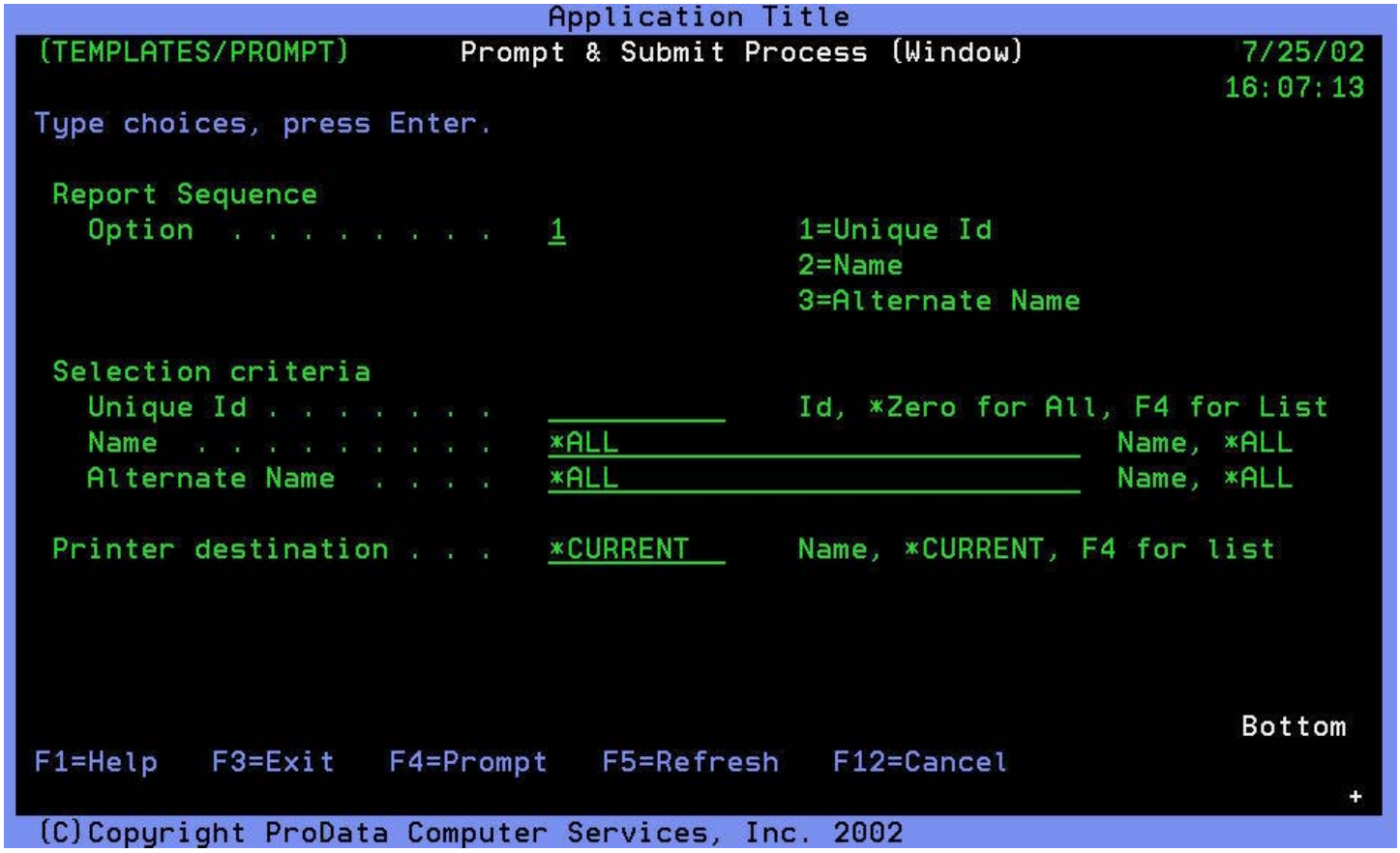
F12=Cancel — The cancel command key exits the user from the program.



Prompt & Submit Process (Window)

The Prompt & Submit Process (Window) program is an example of a method to prompt users for report or program run information.

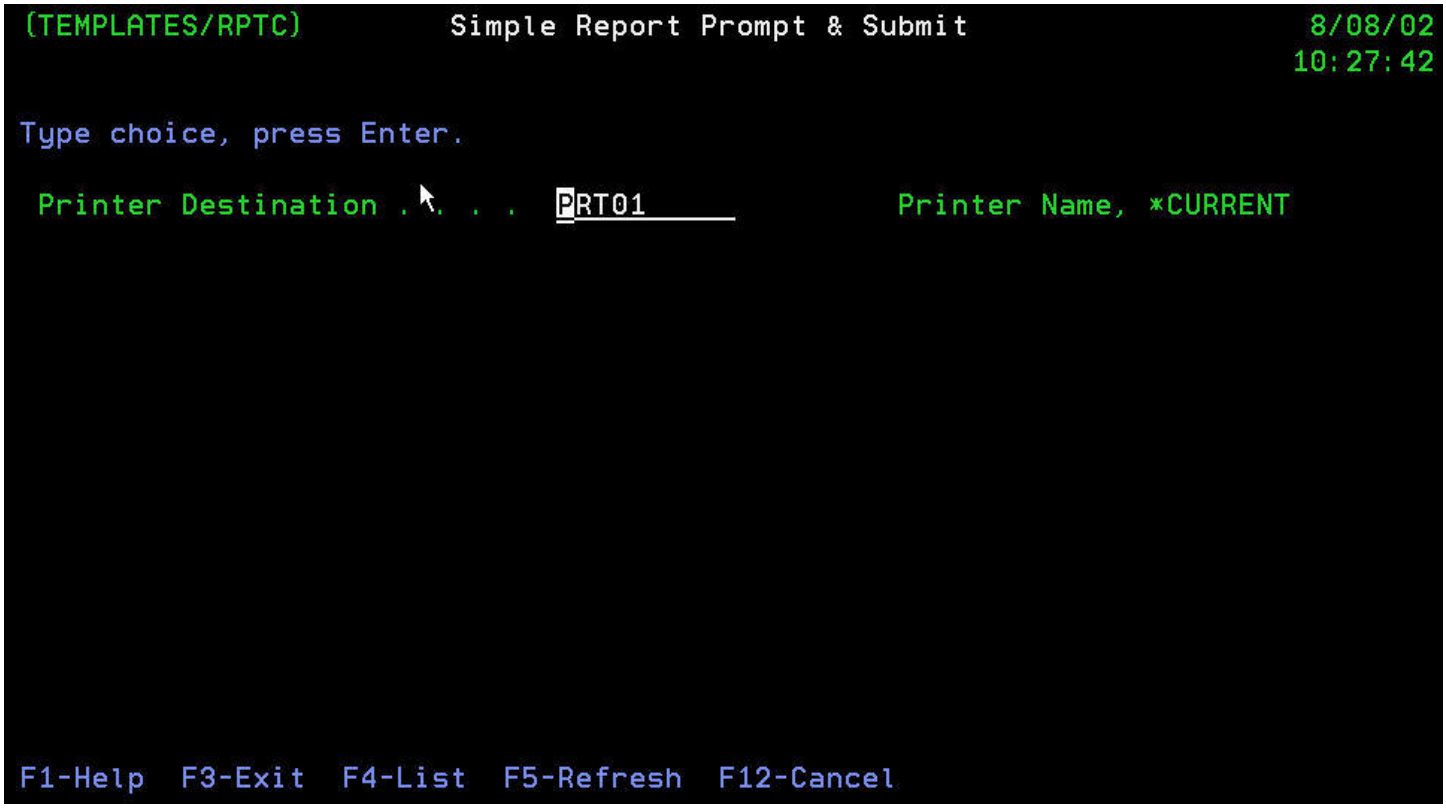
Command Keys	**The command keys available in the Prompt & Submit Process (Window) program are described below.**
F1=Help	The help command key appears only for your help text implementation. The templates provide a simple window that appears when the Help key is pressed. You would have to modify the display file to include any help information you desire for your application.
F3=Exit	The exit command key exits the user from the program.
F4=List	The list command key can be used to prompt and select valid information.
F5=Refresh	The refresh command key clears the screen and allows entry of a new key.
F12=Cancel	The cancel command key exits the user from the program.



Simple Report Prompt & Submit

The Simple Report Prompt & Submit program is an example of a method to prompt users for report or program run information.

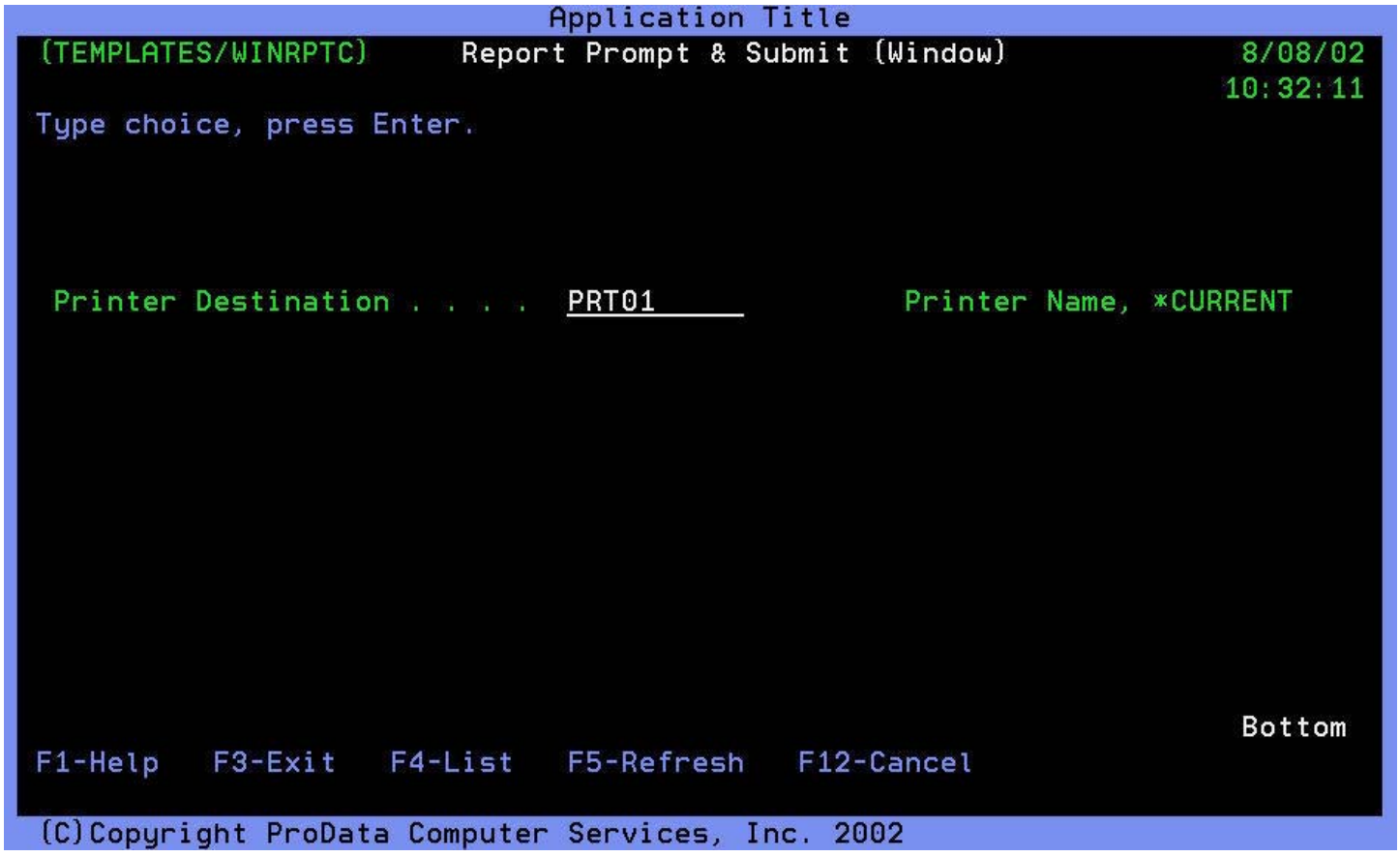
Command Keys	**The command keys available in the Simple Report Prompt & Submit program are described below.**
F1=Help	The help command key appears only for your help text implementation. The templates provide a simple window that appears when the Help key is pressed. You would have to modify the display file to include any help information you desire for your application.
F3=Exit	The exit command key exits the user from the program.
F4=List	The list command key can be used to prompt and select valid information.
F5=Refresh	The refresh command key clears the screen and allows entry of a new key.
F12=Cancel	The cancel command key exits the user from the program.



Report Prompt & Submit (Window)

The Report Prompt & Submit (Window) program is an example of a method to prompt users for report or program run information. The only difference between this prompt & submit program and the previous is this one is wrapped in a window boarder.

Command Keys	**The command keys available in the Report Prompt & Submit (Window) program are described below.**
F1=Help	The help command key appears only for your help text implementation. The templates provide a simple window that appears when the Help key is pressed. You would have to modify the display file to include any help information you desire for your application.
F3=Exit	The exit command key exits the user from the program.
F4=List	The list command key can be used to prompt and select valid information.
F5=Refresh	The refresh command key clears the screen and allows entry of a new key.
F12=Cancel	The cancel command key exits the user from the program.



Template Standards

Program Naming Standards

The template programs allow you to use any naming standards that you have developed for programs. It is highly recommended that some type of naming standards for programs be followed.

Program Headers

The header section provided in the template programs is used for information purposes only. Contained within the program header is the modification log. The modification log is used to track the genealogy of changes made to a program.

Header/Modification Logs

```
H copyright('(C)Copyright ProData Computer Services, Inc. 2002')
 *-------------------------------------------------------------------*
 *                                                                   *
 *  System Name...: Templates                                        *
 *  Program Name..: FM                                               *
 *  Program Type..: RPGLE                                            *
 *  Description...: File Maintenance (Window)                        *
 *  Creation Date.: 07/06/2002                                       *
 *                                                                   *
 *  (C)Copyright ProData Computer Services, Inc. 2002                *
 *  This software is licensed per individual machine.                *
 *  To order your copy: 1-800/228-6318                               *
 *                      sales@prodatacomputer.com                    *
 *-------------------------------------------------------------------*
 *              C r e a t i o n    C o m m a n d                     *
 *-------------------------------------------------------------------*
 *   crtbndrpg pgm(TEMPLATES/FM) srcfile(TEMPLATES/QILESRC) +        *
 *             srcmbr(*pgm) genlvl(10) text(*srcmbrtxt) +            *
 *             dftactgrp(*no) actgrp(*caller) bnddir(*none) +        *
 *             replace(*yes)                                         *
 *                                                                   *
 *-------------------------------------------------------------------*
 *           M o d i f i c a t i o n   J o u r n a l                 *
 *-------------------------------------------------------------------*
 * Date        Name             Description                          *
 * ----------  ---------------  ------------------------------------ *
 * 07/06/2002  Davey Webster    Initial Creation Of Program.         *
 *                                                                   *
 *                                                                   *
 *-------------------------------------------------------------------*
```

File Naming Standards

The template programs allow you to use any naming standards that you have developed for files. It is highly recommended that some type of naming standards for files be followed.

Display File Naming Standards

The template programs allow you to use any naming standards that you have developed for display file naming standards. It is highly recommended that some type of naming standards for display file naming be followed.

Record Formats

The standard record format for a screen is SCNREC#1. The breakdown of the record format naming convention described below.

SCNREC	The first part of the record format name identifies it as a screen record.
#1	Identifies this as screen format number 1.

If multiple screen formats are used the screen number will increment by one.
The following is an example of a display file with 3 screen record formats.

- SCNREC#1 - Screen record number 1
- SCNREC#2 - Screen record number 2
- SCNREC#3 - Screen record number 3

The following is an example of the screen record.

```
A*/ ------------------------------------------------------------ */
A*/ Screen Record #1                                             */
A*/ ------------------------------------------------------------ */
A          R SCNREC#1
A                                      WINDOW(MSGCTL#)
A                                      USRRSTDSP
A                                      RTNCSRLOC(&##_RECORD &##_FIELD)
A                                      CSRLOC(##_LINE#   ##_COLUMN#)
A                                      OVERLAY CHANGE(98)
A            ##_RECORD     10A  H
A            ##_FIELD      10A  H
A            ##_LINE#       3S 0H
A            ##_COLUMN#     3S 0H
A            #1_DBR         9S 0H
A            ##_PGMLIB     23A  O  1  1
A                                   1 31'Inquiry (Window)'
A                                      COLOR(WHT)
A                                   1 68DATE
A                                      EDTCDE(Y)
A                                   2 68TIME
A                                   3  1'Type choices, press Enter.'
A                                      COLOR(BLU)
A                                   5  2'Unique Key  . . . .'
A            #1_KEY         9S 0B  5 25
A N04                                  DSPATR(PR)
A  31                                  DSPATR(RI)
A  31                                  DSPATR(PC)
A                                   6  2'Name  . . . . . .'
A            #1_NAME       30A  O  6 25
A  04                                  DSPATR(ND)
```

```
A                                       7  2'Alternate Name  . .'
A            #1_ALTNAME     30A  O  7 25
A  04                                     DSPATR(ND)
A                                       8  2'Title . . . . . .'
A            #1_TITLE       20A  O  8 25
A  04                                     DSPATR(ND)
A                                       9  2'Company Name  . . .'
A            #1_COMPANY     30A  O  9 25
A  04                                     DSPATR(ND)
A                                      10  2'Address . . . . . .'
A            #1_ADDR1       30A  O 10 25
A  04                                     DSPATR(ND)
A            #1_ADDR2       30A  O 11 25
A  04                                     DSPATR(ND)
A            #1_ADDR3       30A  O 12 25
A  04                                     DSPATR(ND)
A                                      13  2'City  . . . . . .'
A            #1_CITY        30A  O 13 25
A  04                                     DSPATR(ND)
A                                      14  2'State/Province  . .'
A            #1_STATE       25A  O 14 25
A  04                                     DSPATR(ND)
A                                      15  2'Postal Code . . . .'
A            #1_POSTAL      10A  O 15 25
A  04                                     DSPATR(ND)
A                                      16  2'Country . . . . . .'
A            #1_COUNTRY     30A  O 16 25
A  04                                     DSPATR(ND)
A                                      17  2'Telephone Number  .'
A            #1_PHONE       25A  O 17 25
A  04                                     DSPATR(ND)
A                                      18  2'Fax Number  . . . .'
A            #1_FAX         25A  O 18 25
A  04                                     DSPATR(ND)
A                                      19  2'eMail Address . . .'
A            #1_EMAIL       50A  O 19 25
A  04                                     DSPATR(ND)
A                                      20 69'Bottom'
A                                         DSPATR(HI)
A                                      21  2'F1=Help'
A                                         COLOR(BLU)
A                                      21 12'F3=Exit'
A                                         COLOR(BLU)
A                                      21 22'F5=Refresh'
A                                         COLOR(BLU)
A                                      21 35'F12=Cancel'
A                                         COLOR(BLU)
```

Subfile Formats

The standard format for a subfile control record is SFLCTL#1. The standard format for a subfile record is SFLREC#1. The breakdown of the record format naming convention is described below.

Subfile Control Record

SFLCTL	The first part of the subfile control record name identifies it as a subfile control record.
#1	Identifies this as subfile control record number 1.

If multiple subfile control records exist the number will increment by one.
The following is an example of a display file with 3 subfile control record formats.
SFLCTL#1 - Subfile Control number 1
SFLCTL#2 - Subfile Control number 2
SFLCTL#3 - Subfile Control number 3

Subfile Record

SFLREC	The first part of the subfile record name identifies it as a subfile record.
#1	Identifies this as subfile record number 1.

If multiple subfile records exist the number will increment by one.
The following is an example of a display file with 3 subfile record formats.
SFLREC#1 - Subfile Record number 1
SFLREC#2 - Subfile Record number 2
SFLREC#3 - Subfile Record number 3

The subfile formats for the control records and the subfile records will be related to each other in the following fashion.

Subfile Control	Subfile Record
SFLCTL#1	SFLREC#1
SFLCTL#2	SFLREC#2
SFLCTL#3	SFLREC#3

The following is an example of the subfile record.

```
A*/ ------------------------------------------------------------ */
A*/ Subfile Record #1                                            */
A*/ ------------------------------------------------------------ */
A          R SFLREC#1                    SFL
A  40                                    SFLNXTCHG
A            #1_DBR          9S 0H
A            #1_SELECT       1A  B  9  2COLOR(WHT)
A  04                                     DSPATR(PR)
A  04                                     DSPATR(ND)
A  41                                     DSPATR(RI)
A            #1_KEY          9Y 0B  9  5CHECK(RB)
A N04                                     DSPATR(PR)
A  42                                     DSPATR(RI)
A                                         CHANGE(98)
A                                         EDTCDE(Z)
A            #1_NAME        30A  B  9 15CHANGE(98)
A  43                                     DSPATR(RI)
A            #1_ALTNAME     30A  B  9 46CHANGE(98)
```

The following is an example of the subfile control record.

```
A*/ ------------------------------------------------------------ */
A*/ Subfile Control Record #1                                    */
A*/ ------------------------------------------------------------ */
A          R SFLCTL#1                    SFLCTL(SFLREC#1)
A                                        SFLSIZ(0011)
A                                        SFLPAG(0010)
A                                        WINDOW(MSGCTL#)
A                                        USRRSTDSP
A                                        RTNCSRLOC(&##_RECORD &##_FIELD)
A                                        CSRLOC(##_LINE#   ##_COLUMN#)
A                                        OVERLAY
A  01 02                                 SFLDSP
A  01                                    SFLDSPCTL
A N01                                    SFLCLR
A  05                                    SFLEND(*MORE)
A                                        SFLCSRRRN(&#1_SFLCSR)
A                                        CHANGE(98)
A            ##_RECORD      10A  H
A            ##_FIELD       10A  H
A            #1_SFLCSR       5S 0H
A            #1_SFLREC       4S 0H      SFLRCDNBR
A            ##_LINE#        3S 0H
A            ##_COLUMN#      3S 0H
A            ##_PGMLIB      23A  O  1  1
A                                    1 25'Subfile Maintenance (Multiple Add)'
A                                       COLOR(WHT)
A                                    1 68DATE
A                                       EDTCDE(Y)
A                                    2 68TIME
A                                    3  1'Position to  . . . . . .'
A            #1_POSNTO       9Y 0B  3 28EDTCDE(Z)
A                                       CHECK(RB)
A                                    3 40'Unique Id'
A                                    5  1'Type options, press Enter.'
A                                       DSPATR(HI)
A                                       COLOR(BLU)
A N04                                6  3'4=Delete'
A                                    8  1'Opt'
A                                       DSPATR(HI)
A                                    8  5'Unique Id'
A                                       COLOR(WHT)
A                                    8 15'Name'
A                                       COLOR(WHT)
A                                    8 46'Alternate Name'
A                                       COLOR(WHT)
```

Message Formats

Only one message control record and message subfile record will be contained in each display file. The message control record also contains the window format information. The window information will only be included in the message control for window programs. The standard format for a message control record is MSGCTL#. The standard format for a message subfile record is MSGSFL#. The breakdown of the record format naming convention described below.

Message Control Record
MSGCTL# - Identifies the standard message control record.

Message Record
MSGSFL# - Identifies the standard message subfile record.

The following is an example of the standard message subfile record.

```
A*/ ------------------------------------------------------------ */
A*/ Message Subfile Record                                       */
A*/ ------------------------------------------------------------ */
A          R MSGSFL#                    SFL
A                                       SFLMSGRCD(22)
A            SFLMSGKEY                  SFLMSGKEY
A            PGMNAME                    SFLPGMQ
```

The following is an example of the standard message control record

```
A*/ ------------------------------------------------------------ */
A*/ Message Subfile Control Record                               */
A*/ ------------------------------------------------------------ */
A           R MSGCTL#                   SFLCTL(MSGSFL#)
A                                       WINDOW(*DFT 22 75 *NOMSGLIN)
A                                       WDWTITLE((*TEXT 'Application -
A                                       Title'))
A                                       WDWTITLE((*TEXT -
A                                       '(C)Copyright-
A                                        ProData Computer Services,-
A                                        Inc. 2002') -
A                                       *BOTTOM *LEFT)
A                                       FRCDTA OVERLAY
A                                       SFLDSPCTL
A                                       SFLDSP
A                                       SFLINZ
A  99                                   SFLEND
A                                       SFLSIZ(0002)
A                                       SFLPAG(0001)
A             PGMNAME                   SFLPGMQ
```

Subroutine Naming Standards

Subroutines within the template programs will start with the character "$". Listed below are several of the subroutine names that can be found through out the template programs. This is not a complete listing of all subroutines contained in the templates but is provided as an example of naming standards.

Subroutine	Description
$InzScn#1	Initialize screen number 1. If multiple display screens are present in your program, you will have additional initialize screen subroutines. Additional initialize screen subroutines would appear in your program as $InzScn#2, InzScn#3, etc…
$LoadScn#1	Load screen number 1. If multiple display screens are present in your program, you will have additional load screen subroutines. Additional load screen subroutines would appear in your program as $LoadScn#2, LoadScn#3, etc…
$DsplyScn#1	Display screen number 1. If multiple display screens are present in your program you will have additional display screen subroutines. Additional display screen subroutines would appear in your program as $DsplyScn#2, $DsplyScn#3, etc…
$ValidScn#1	Validate screen number 1. If multiple display screens are present in your program, you will have additional validate screen subroutines. Additional validate screen subroutines would appear in your program as $ValidScn#2, ValidScn#3, etc…
$UpDateScn#1	Update screen number 1. If multiple display screens are present in your program, you will have additional update screen subroutines. Additional update screen subroutines would appear in your program as $UpDdateScn#2, UpDateScn#3, etc…
$DeleteScn#1	Delete screen number 1. If multiple display screens are present in your program, you will have additional delete screen subroutines. Additional delete screen subroutines would appear in your program as $DeleteScn#2, DeleteScn#3, etc…
$InizSfl#1	Initialize subfile number 1. If multiple subfile screens are present in your program, you will have additional initialize subfile subroutines. Additional initialize subfile subroutines would appear in your program as $InizSfl#2, $InizSfl#3, etc…
$LoadSfl#1	Load subfile number 1. If multiple subfile screens are present in your program, you will have additional load subfile subroutines. Additional load subfile subroutines would appear in your program as $LoadSfl#2, $LoadSfl#3, etc…
$DsplySfl#1	Display subfile number 1. If multiple subfile screens are present in your program, you will have additional display subfile subroutines. Additional display subfile subroutines would appear in your program as $DsplySfl#2, $DsplySfl#3, etc…
$ValidSfl#1	Validate subfile number 1. If multiple subfile screens are present in your program, you will have additional validate subfile subroutines. Additional validate subfile subroutines would appear in your program as ValidSfl#2, $ValidSfl#3, etc…
$ProcSfl#1	Process subfile number 1. If multiple subfile screens are present in your program, you will have additional process subfile subroutines. Additional process subfile subroutines would appear in your program as $ProcSfl#2, $ProcSfl#3, etc…

$ValidAdd#1	Validate Add for subfile number 1. If multiple subfile screens are present in your program, you will have additional validate add subfile subroutines. Additional validate add subfile subroutines would appear in your program as $ValidAdd#2, $ValidAdd#3, etc…
$ProcAdd#1	Process Add for subfile number 1. If multiple subfile screens are present in your program, you will have additional process add subfile subroutines. Additional process add subfile subroutines would appear in your program as $ProcAdd#2, $ProcAdd#3, etc…
$UpdateSfl#1	Update subfile number 1. If multiple subfile screens are present in your program, you will have additional update subfile subroutines. Additional update subfile subroutines would appear in your program as $UpdateSfl#2, $UpdateSfl#3, etc…
$DeleteSfl#1	Delete subfile number 1. If multiple subfile screens are present in your program, you will have additional delete subfile subroutines. Additional delete subfile subroutines would appear in your program as $DeleteSfl#2, $DeleteSfl#3, etc…
$ProcKey#1	Process command keys for screen number 1. If multiple display or subfile screens are present in your program, you will have additional process key subroutines. Additional process key subroutines would appear in your program as $ProcKey#2, $ProcKey#3, etc…
$NextRec	Next record subroutine. Load the next record in the database being processed. Notify the user when the bottom of the list has been reached.
$PrevRec	Previous record subroutine. Load the previous record in the database being processed. Notify the use when the top of the list has been reached.
$PageUp	Page Up subroutine. Set the pointer position on the database file for the previous page of subfile records that are to be loaded.
$PrevScreen	Previous screen subroutine. Determines the navigation from screen to screen when a request has been made to return to the previous screen.
$Help	Help subroutine. Currently this is a shell subroutine. It is intended to process any help functionality you choose to add.
$List	List subroutine. The process of F4 lists for selected fields is contained within this subroutine.
$Reset	Reset subroutine. The reset subroutine is used to reset the initial values of the screen.
$ExitPgm	Exit program subroutine. A normal termination of the program is processed though this subroutine.
*InzSr	Initialize subroutine. The initialization subroutine allows you to process calculation specifications before 1P output. It is used to initialize values and setup miscellaneous structures.

Field Naming Standards

Field naming standards have been kept relatively simple. Program fields, display file fields and program work fields are the primary divisions in naming standards. Listed below are examples and descriptions of the 3 major field naming standards. In the RPG fields standards there are 2 more types of fields that are not used in the display files. Parameter fields are used for passing parameters to a program. Work fields are the second type of field that is specific to the RPG templates.

Display File Field Naming Conventions

Template Fields

Template Fields - The template fields that are used for controlling subfiles and other screen attributes start with the characters "##'. Template fields should not be removed from the programs when you edit the source files. The purpose of these fields, if not used initially, could be used later in the life of the program. The template fields are also referenced in the RPGLE program. Listed below are example template fields in a display file.

##_RECORD - Record format name
##_FIELD - Field name
##_COLUMN# - Column number of cursor
##_LINE# - Line number of cursor
##_PGMLIB - Program & Library name

The following is a section of a template display file where the template fields are used.

```
A*/ ------------------------------------------------------------ */
A*/ Screen Record #1                                             */
A*/ ------------------------------------------------------------ */
A          R SCNREC#1
A                                      WINDOW(MSGCTL#)
A                                      USRRSTDSP CHANGE(98)
A                                      RTNCSRLOC(&##_RECORD &##_FIELD)
A                                      CSRLOC(##_LINE#   ##_COLUMN#)
A                                      OVERLAY
A            ##_RECORD     10A  H
A            ##_FIELD      10A  H
A            ##_LINE#       3S 0H
A            ##_COLUMN#     3S 0H
A            #1_DBR         9S 0H
A            ##_PGMLIB     23A  O  1  1
```

User Defined Fields

User Defined Fields - User defined fields can be any naming convention you decide to use. The naming convention used by the templates places the characters "#1_" in front of the field name. An example of this would be a field in a file that was named "NAME" would be represented in the display file template as "#1_NAME". The numeric value after the pound sign relates to the screen that the field is related to. Listed below are example template fields in a display file.

#1_KEY - This is an example of a user defined field.
#1_NAME - This is an example of a user defined field.
#1_ALTNAME - This is an example of a user defined field.
#1_TITLE - This is an example of a user defined field.
#1_COMPANY - This is an example of a user defined field.

The following is a section of a template display file where the user defined fields are used.

```
A            #1_KEY          9S 0B  5 25
A N04                                 DSPATR(PR)
A  31                                 DSPATR(RI)
A  31                                 DSPATR(PC)
A                                   6  2'Name  . . . . . .'
A            #1_NAME        30A  O  6 25
A  04                                 DSPATR(ND)
A                                   7  2'Alternate Name  . .'
A            #1_ALTNAME     30A  O  7 25
A  04                                 DSPATR(ND)
A                                   8  2'Title . . . . . .'
A            #1_TITLE       20A  O  8 25
A  04                                 DSPATR(ND)
A                                   9  2'Company Name  . . .'
A            #1_COMPANY     30A  O  9 25
A  04                                 DSPATR(ND)
```

RPG Field Naming Conventions

Template Fields

Template Fields - The template fields that are used for controlling subfiles and other screen attributes start with the characters "##_'. Template fields should not be removed from the programs when you edit the source files. The purpose of these fields, if not used initially, could be used later in the life of the program. The template fields are also referenced in the Display File program. Listed below are example template fields in a RPGLE program.

The following is a section of a template RPGLE program where the template fields are used.

```
 *----------------------------------------------------------------
 * File Information Data Structure For Display File
 *----------------------------------------------------------------
d ##_Infds        ds
d  ##_Format            261    270
d  ##_CmdKey            369    369
d  ##_Line              382    382
d  ##_Column            383    383
```

User Defined Fields

User Defined Fields - User defined fields can be any naming convention you decide to use. The naming convention used by the templates places the characters "#1_" in front of the field name. An example of this would be a field in a file that was named "NAME" would be represented in the display file template as "#1_NAME". The numeric value after the pound sign relates to the screen that the field is related to. Listed below are example template fields in a RPGLE program.

#1_Name - This is an example of a user defined field.
#1_AltName - This is an example of a user defined field.
#1_Title - This is an example of a user defined field.
#1_Company - This is an example of a user defined field.

The following is a section of a template RPGLE program where the user defined fields are used.

```
 *----------------------------------------------------------------
 * $LoadScn#1 - Subr To Load Screen #1 From File
 *----------------------------------------------------------------
c     $LoadScn#1    BegSr

 * Retrieve Record From File
c                   clear                   DltFlag
c     kyEXAMPLEPF   chain(n)  EXAMPLEPF

 * Move File Fields To Display File
c                   if        %found(EXAMPLEPF)
c                   eval      #1_Dbr = pf_Record#
c                   eval      *in04 = *off
c                   eval      #1_Name    = Name
c                   eval      #1_AltName = AltName
c                   eval      #1_Title   = Title
c                   eval      #1_Company = Company
```

Internal Program Fields

Internal Program Fields - Internal program fields should not be modified when you copy the templates to create a new program. It is recommended that if these fields are not used to leave them in the code for future use. Examples of where these fields are used: Data structures, Informational Data Structures, Fields associated with API's and Navigational Flag controls. Navigational Flag controls are the only fields that we recommend you modify in the template programs regarding Internal Program Fields. List below are examples of internal program fields that reside in the template programs.

Field Name	Description
ControlFlag	Internal program field used for navigation of program processes.
DsplyScn#1	Internal program constant used by navigational flag process.
Mh_SndRmvDS	Data structure used for internal message handling API's QTMHSNDPM and QMHRMVPM.
MhSndPm	The parameter list used when calling the Send Program Message API (QMHSNDPM).

The following is a section of a template RPGLE program where the internal program fields are used.

```
* Send Message - Record not found in file &1.
c                   reset                   Mh_SndRmvDs
c                   eval      Mh_MsgData = pf_File
c                   eval      Mh_Msg# = 'ERR0001'
c                   call      'QMHSNDPM'    MhSndPm
c                   add       1             ErrorCount

 * If Record Is Loaded, & In Delete Mode, Process Delete
c                   if        Mode = DeleteMode and #1_Dbr <> *loval
c                   eval      ControlFlag = DeleteScn#1

c                   else
* Set Program/Display Variables
c                   eval      ControlFlag = DsplyScn#1
c                   endif
```

Parameter Field Naming Conventions

Parameter Fields - Parameter fields are used for *entry parameters into a RPGLE program. The parameter fields start with "p_".

Below is an example of parameter fields that reside one of the template programs.

```
 *----------------------------------------------------------------------
 * Parameters Passed Into Program
 *----------------------------------------------------------------------
c     *entry        plist
c                   parm                    p_Mode
c                   parm                    p_Key
```

Subfile Options

The template programs options in the subfile templates perform certain actions against a subfile record. Listed below are the options that currently exist in the templates.

Option	Description
1=Select	Selects the subfile record and changes the *entry parameters to the value of the selected subfile record. This option then returns to the calling program.
2=Change	Selects the subfile record and call the related maintenance program with the proper key fields and parameters for editing the corresponding record.
4=Remove	Selects the subfile records and prompts the delete confirmation screen displaying all records selected for deletion.
5=Display	Selects the subfile record and call the related maintenance program with the proper key fields and parameters for displaying the corresponding record.
6=Print	Selects the subfile record and call the related printing program.

```
Type options, press Enter.
  1=Select   2=Change   4=Remove   5=Display details   6=Print details
```

Command Keys

The template programs refer to command keys using the file information data structure for the display file being processed. Once a function or action key is pressed the program can determine what action, if any, should be taken. The following is a list of defined function/action keys.

Function Key	Description
F1	Used by Templates for processing HELP
F2	*(User definable function key)*
F3	Used by Templates for Exiting the program.
F4	Used by Templates for performing LIST Functions for fields.
F5	Used by Templates for Refreshing or Restoring initial values.
F6	Used by Templates to enter into ADD mode.
F7	*(User definable function key)*
F8	*(User definable function key)*
F9	*(User definable function key)*
F10	*(User definable function key)*
F11	*(User definable function key)*
F12	Used by Templates to navigate to Previous screen.
F13	*(User definable function key)*
F14	*(User definable function key)*
F15	*(User definable function key)*
F16	*(User definable function key)*
F17	*(User definable function key)*
F18	*(User definable function key)*
F19	*(User definable function key)*
F20	*(User definable function key)*
F21	*(User definable function key)*
F22	*(User definable function key)*
F23	*(User definable function key)*
F24	*(User definable function key)*
CLEAR	*(User definable function key)*
ENTER	Used by Templates to advance the current process.
HELP	Used by Templates for processing HELP
PageUp	Used by Templates to Roll to previous record or block of subfile records.
PageDown	Used by Templates to Roll to next record or block of subfile records.
PRINT	*(User definable function key)*
BackSpace	*(User definable function key)*

Subroutines

The following subroutines are some of the more common ones found in the template programs. Under each of these is a description of what that subroutine does in the template programs. Not all templates will contain all of these subroutines.

$DsplyScn#1 - Display Screen Number 1

The display screen subroutine will exist for each display screen you have in your program. If your program contains several screens you would have several display screen subroutines. The first screen would be associated with $DsplyScn#1 and the second would be $DsplyScn#2 and so on. The primary function of the display screen subroutines are listed below.

1. Display one or multiple messages in the message subfile.
2. Display and allow input from the display screen.
3. Remove any messages after control has been returned to the program.
4. Determine the action that should be taken by the program.
    a. If a command key has bee pressed the program will execute the command key processing subroutine.
    b. If a record has not been loaded and the screen has changed, the program will execute the load screen subroutine.
    c. If a record had been loaded and the screen has changed, the program will executed the validate screen subroutine.
    d. If something has occurred that didn't meet any of the above criteria, the program will load the navigational flag with the DsplyScn#1 value and redisplay the screen.
    e. If the program had all possible keys loaded at entry time, the exit subroutine will be ran. This allows the selection of a record from a subfile front-end program, with the return to the subfile program after completion of any maintenance.

The following is an example of the $DsplyScn#1 Subroutine.

```
 *-------------------------------------------------------------------
 * $DsplyScn#1 - Subr To Display Screen #1
 *-------------------------------------------------------------------
c     $DsplyScn#1   BegSr

 * If More Than One Message Has Been Sent, Set *IN99 *OFF
c                   if        MsgCount > 1
c                   eval      *in99 = *off
c                   else
c                   eval      *in99 = *on
c                   endif

 * Write Error Message Subfile Control Record
c                   write     MsgCtl#

 * Read/Write Screen Record #1
c                   exfmt     ScnRec#1

 * Remove Program Messages
c                   reset                     Mh_SndRmvDs
c                   call      'QMHRMVPM'      MhRmvPm
c                   clear                     MsgCount

 * Get Cursor Position
c                   eval      Line = ##_Line
c                   eval      Column = ##_Column
```

$DsplyScn#1 - Display Screen Number 1 Continuation…

```
c                   select
 * If Valid Command Key Process Command Key Routine
c                   when      *in99 = *on
c                   exsr      $ProcKey#1

 * If Record Is Not Loaded, & Screen Changed, Reload Screen #1
c                   when      #1_Dbr = *loval and #1_Key <> *zeros
c                   exsr      $Reset
c                   exsr      $LoadScn#1

 * If Record Is Loaded, & Screen Changed, Validate Screen #1
c                   when      #1_Dbr <> *loval and *in98 = *on
c                   exsr      $ValidScn#1

 * Otherwise, Return To Display Screen
c                   other
c                   eval      ControlFlag = DsplyScn#1

 * If # Of Parameters Passed=2, Exit Program
c                   if        %parameters >= 2
c                   exsr      $ExitPgm
c                   endif
c                   endsl

c                   EndSr
```

$LoadScn#1 - Load Screen Number 1

The load screen subroutine will exist for each display screen you have in your program. If your program contains several screens you would have several load screen subroutines. The first screen would be associated with $LoadScn#1 and the second would be $LoadScn#2 and so on. The primary functions of the load screen subroutine are listed below.

1. Retrieve the record from file without locking it.
2. If retrieval of the record was successful, populate the screen fields.
3. If retrieval of the record failed, send a message to notify the user record was not found.
4. If the record was found and the program mode is delete, set the navigational flag to DeleteScn#1.
5. If the record was found and the program mode is not delete, set the navigational flag to DsplyScn#1.

The following is an example of the $LoadScn#1 Subroutine.

```
 *-------------------------------------------------------------------
 * $LoadScn#1 - Subr To Load Screen #1 From File
 *-------------------------------------------------------------------
c     $LoadScn#1    BegSr

 * Retrieve Record From File
c                    clear                     DltFlag
c     kyEXAMPLEPF    chain(n)  EXAMPLEPF

 * Move File Fields To Display File
c                    if        %found(EXAMPLEPF)
c                    eval      #1_Dbr = pf_Record#
c                    eval      *in04 = *off
c                    eval      #1_Name    = Name
c                    eval      #1_AltName = AltName
c                    eval      #1_Title   = Title
c                    eval      #1_Company = Company
c                    eval      #1_Addr1   = Addr1
c                    eval      #1_Addr2   = Addr2
c                    eval      #1_Addr3   = Addr3
c                    eval      #1_City    = City
c                    eval      #1_State   = State
c                    eval      #1_Country = Country
c                    eval      #1_Postal  = Postal
c                    eval      #1_Phone   = Phone
c                    eval      #1_Fax     = Fax
c                    eval      #1_eMail   = eMail
c                    else
```

$LoadScn#1 - Load Screen Number 1 Continuation…

```
c                   eval      *in04 = *on
c                   eval      #1_Dbr = *loval
* Send Message - Record not found in file &1.
c                   reset                   Mh_SndRmvDs
c                   eval      Mh_MsgData = pf_File
c                   eval      Mh_Msg# = 'ERR0001'
c                   call      'QMHSNDPM'    MhSndPm
c                   add       1             ErrorCount
c                   endif

 * If Record Is Loaded, & In Delete Mode, Process Delete
c                   if        Mode = DeleteMode and #1_Dbr <> *loval
c                   eval      ControlFlag = DeleteScn#1

c                   else
 * Set Program/Display Variables
c                   eval      ControlFlag = DsplyScn#1
c                   endif

c                   EndSr
```

$ValidScn#1 - Validate Screen Number 1

The validate screen subroutine will exist for each display screen you have in your program. If your program contains several screens you would have several validate screen subroutines. The first screen would be associated with $ValidScn#1 and the second would be $ValidScn#2 and so on. The primary functions of the validate screen subroutine are listed below.

1. Set off all error indicators and clear the error count field.
2. If the program is in ADD mode, check to insure this record does not exist in the file. If the record does exist, send an error message that the entry already exists.
3. If the program is in change mode, make sure the record exists and is not locked. If the record does not exist, a message will be sent stating the record was not found. If the record is locked, the DSPRCDLCKC program is called. The DSPRCDLCKC program is described later in this document.
4. The next section of the validate subroutine is where you would place any of the edits you would like to check.
5. The final section of the validate screen subroutine will check the error count field. If it is equal to zero the update subroutine will be executed. If the error count is greater than zero, the navigational flag will be set to DsplyScn#1 and the program will redisplay the screen with any error messages that occurred.

The following is an example of the $ValidScn#1 Subroutine.

```
 *----------------------------------------------------------------------
 * $ValidScn#1 - Subr To Validate Screen #1
 *----------------------------------------------------------------------
c     $ValidScn#1   BegSr
c                   clear                   ErrorCount
c                   movea     '00000000'    *in(40)

c                   select
 * If In Add Mode, Make Sure The Record Doesn't Already Exist
c                   when      Mode = AddMode
c     kyEXAMPLEPF   chain     EXAMPLEPF
c                   if        %found(EXAMPLEPF)
c                   eval      *in31 = *on
c                   reset                   Mh_SndRmvDs
c                   eval      Mh_MsgData = pf_MbrName + pf_File +
c                               pf_Library + pf_Record + pf_Format +
c                               pf_Member + x'00000000' + pf_Format +
c                               pf_Member
c                   eval      Mh_Msg# = 'ERR0002'
c                   call      'QMHSNDPM'    MhSndPm
c                   add       1             ErrorCount
c                   endif
```

$ValidScn#1 - Validate Screen Number 1 Continuation…

```
c                   other

 * If In Change Mode, Make Sure The Record Exists, And Is Not Locked
c     kyEXAMPLEPF   chain(e)  EXAMPLEPF
c                   if        not %found(EXAMPLEPF)
c                   eval      *in31 = *on
c                   reset                 Mh_SndRmvDs
c                   eval      Mh_MsgData = pf_File
c                   eval      Mh_Msg# = 'ERR0001'
c                   call      'QMHSNDPM'  MhSndPm
c                   add       1           ErrorCount
c                   endif

 * Process Record Lock
c                   if        %error
c                   call      'DSPRCDLCKC'
c                   parm                  pf_File
c                   parm                  pf_Library
c                   parm                  PgmFileSts
c                   parm                  PgmMsgData
c                   eval      *in31 = *on
c                   add       1           ErrorCount
c                   endif
c                   endsl

 * Validate Name Field
c                   if        #1_Name = *blanks
c                   eval      *in41 = *on
c                   reset                 Mh_SndRmvDs
c                   eval      Mh_MsgData = 'Name field cannot be blanks'
c                   eval      Mh_Msg# = 'ERR0000'
c                   call      'QMHSNDPM'  MhSndPm
c                   add       1           ErrorCount
c                   endif

 * If No Errors Occurred, Process Update/Add Requests
c                   if        ErrorCount = *zero
c                   exsr      $UpdateScn#1
c                   exsr      $Reset
c                   else
c                   eval      *in40 = *on
c                   add       ErrorCount  MsgCount
c                   endif
c                   eval      ControlFlag = DsplyScn#1
c                   EndSr
```

$UpDateScn#1 - Update Screen Number 1

The update screen subroutine will exist for each display screen you have in your program. If your program contains several screens you would have several update screen subroutines. The first screen would be associated with $UpDateScn#1 and the second would be $UpDateScn#2 and so on. The primary functions of the update screen subroutine are listed below.

1. All screen fields are "eval"ed into their corresponding database fields.
2. If the program is in add mode, the key fields will be "eval"ed into the database fields and the record will be added. After the record has been added, a message will be sent notifying the user that a record was added.
3. If the program is in change mode, the record will be updated and a message will be sent notifying the users that a record was changed.
4. If the program had all possible keys loaded at entry time, the exit subroutine will execute, This allows the selection of a record from a subfile front-end with the return to the subfile after completion of the maintenance to the record.

The following is an example of the $UpDateScn#1 Subroutine.

```
 *------------------------------------------------------------------
 * $UpDateScn#1 - Subr To Update Changed/Added Records
 *------------------------------------------------------------------
c     $UpDateScn#1  BegSr

 * Move Display File Fields To The Record Fields
c                   eval      Name    = #1_Name
c                   eval      AltName = #1_AltName
c                   eval      Title   = #1_Title
c                   eval      Company = #1_Company
c                   eval      Addr1   = #1_Addr1
c                   eval      Addr2   = #1_Addr2
c                   eval      Addr3   = #1_Addr3
c                   eval      City    = #1_City
c                   eval      State   = #1_State
c                   eval      Country = #1_Country
c                   eval      Postal  = #1_Postal
c                   eval      Phone   = #1_Phone
c                   eval      Fax     = #1_Fax
c                   eval      eMail   = #1_eMail
c                   eval      LstUser = PgmUser
c                   eval      LstPgm  = PgmName
c                   time                    LstDate
c                   time                    LstTime
```

$UpDateScn#1 - Update Screen Number 1 Continuation…

```
* If In Add Mode, Move Key Fields To Record Fields
c                   if        Mode = AddMode
c                   clear                   DltFlag
c                   eval      Key  = #1_Key
c                   eval      AddUser = PgmUser
c                   eval      AddPgm  = PgmName
c                   time                    AddDate
c                   time                    AddTime
c                   write     EXAMPLE##
c                   eval      *in04 = *on

 * Send Message Record(s) Added To File
c                   reset                   Mh_SndRmvDs
c                   eval      bRecords = 1
c                   eval      Mh_MsgData = pf_File + pf_Library +
c                               pf_MbrName + cRecords
c                   eval      Mh_Msg# = 'INF0001'
c                   call      'QMHSNDPM'    MhSndPm
c                   add       1             ErrorCount

 * Else, Update Record
c                   else
c                   update    EXAMPLE##

 * Send Message Records Update In File
c                   reset                   Mh_SndRmvDs
c                   eval      bRecords = 1
c                   eval      Mh_MsgData = pf_File + pf_Library +
c                               pf_MbrName + cRecords
c                   eval      Mh_Msg# = 'INF0002'
c                   call      'QMHSNDPM'    MhSndPm
c                   add       1             ErrorCount

 * If # Of Parameters Passed=2, Exit Program
c                   if        %parameters >= 2
c                   exsr      $ExitPgm
c                   endif
c                   endif

c                   EndSr
```

$DeleteScn#1 - Delete Screen Number 1

The delete screen subroutine will exist for each display screen you have in your program. If your program contains several screens you would have several delete screen subroutines. The first screen would be associated with $DeleteScn#1 and the second would be $DeleteScn#2 and so on. The primary functions of the delete screen subroutine are listed below.

1. A message is sent to the user informing them to press enter to delete the record.
2. The message subfile is written to the screen, The delete confirmation screen is then excuted.
3. After control returns to the program, all messages are cleared.
4. If a command key was pressed, the process command keys subroutine is executed.
5. When the enter key is pressed the record is retrieve and checked for any locks. If the record was not found, a message will be sent to the screen. If the record was locked, the DSPRCDLCKC program will be called.
6. If the record was found and the record is not locked, it will be deleted.
7. The navigational flag will be se to $Reset and a message will be sent to notify the user that the record was deleted.
8. If the program had all possible keys loaded at entry time, the exit subroutine will be executed, This allows the selection of a record from a subfile front-end with the return to the subfile after completion of the maintenance to the record.

The following is an example of the $DeleteScn#1 Subroutine.

```
 *--------------------------------------------------------------------
 * $DeleteScn#1 - Subr To Process Deletes For Screen #1
 *--------------------------------------------------------------------
c     $DeleteScn#1  BegSr

 * Send Message To Press Enter To Delete.
c                   reset                   Mh_SndRmvDs
c                   eval      Mh_Msg# = 'INF0006'
c                   call      'QMHSNDPM'    MhSndPm
c                   eval      *in99 = *on

 * Write Error Message Subfile Control Record
c                   write     MSGCTL#

 * Write Screen Record #1
c                   write     ScnRec#1

* Write/Read Delete Confirmation Screen
c                   exfmt     Confirm#

 * Remove Program Messages
c                   reset                   Mh_SndRmvDs
c                   call      'QMHRMVPM'    mhRmvPm
c                   clear                   MsgCount

c                   select

 * If Valid Command Key Has Been Pressed, Process Command Key Routine
c                   when      *in99 = *on
c                   exsr      $ProcKey#1
```

$DeleteScn#1 - Delete Screen Number 1 Continuation…

```
* Otherwise Delete Requests
c                   other
 * Make Sure The Record Exists, And Is Not Locked
c     kyEXAMPLEPF   chain(e)  EXAMPLEPF
c                   if        not %found(EXAMPLEPF)
c                   eval      *IN31 = *ON
c                   reset                   Mh_SndRmvDs
c                   eval      Mh_MsgData = pf_File
c                   eval      Mh_Msg# = 'ERR0001'
c                   call      'QMHSNDPM'    MhSndPm
c                   add       1             ErrorCount
c                   endif

 * Process Record Lock
c                   if        %error
c                   call      'DSPRCDLCKC'
c                   parm                    PgmFile
c                   parm                    PgmFileSts
c                   parm                    PgmMsgData
c                   eval      *in31 = *on
c                   add       1             ErrorCount
c                   endif

 * Update Delete Flag With 'D' Record
c                   if        %found(EXAMPLEPF) and not %error
c                   eval      DltFlag = 'D'
c                   eval      LstUser = PgmUser
c                   eval      LstPgm  = PgmName
c                   time                    LstDate
c                   time                    LstTime
c                   update    EXAMPLE##
c                   endif

 * Reset Values, After Deletes Have Been Processed
c                   eval      ControlFlag = Reset
 * Send Message Records Flagged For Deletion
c                   reset                   Mh_SndRmvDs
c                   eval      bRecords = 1
c                   eval      Mh_MsgData = pf_File + pf_Library +
c                               pf_MbrName + cRecords
c                   eval      Mh_Msg# = 'INF0003'
c                   call      'QMHSNDPM'    MhSndPm
c                   add       1             ErrorCount
c                   endsl

c                   EndSr
```

$InizScn#1 - Initialize Screen Number 1

The initialize screen subroutine will exist for each display screen you have in your program. If your program contains several screens you would have several initialize screen subroutines. The first screen would be associated with $InizScn#1 and the second would be $InizScn#2 and so on. The primary functions of the initialize screen subroutine are listed below.

1. All screen fields and any work fields are cleared.
2. The navigational flag is set to $DsplyScn#1.

The following is an example of the $InizScn#1 Subroutine.

```
 *-------------------------------------------------------------------
 * $InizScn#1 - Subr To Initialize Screen #1 For Add's
 *-------------------------------------------------------------------
c     $InizScn#1    BegSr

c                   eval      *in04 = *off
 * Initialize Fields For Screen Record
c                   eval      #1_Dbr = *hival
c                   clear                   #1_Key
c                   clear                   #1_Name
c                   clear                   #1_AltName
c                   clear                   #1_Title
c                   clear                   #1_Company
c                   clear                   #1_Addr1
c                   clear                   #1_Addr2
c                   clear                   #1_Addr3
c                   clear                   #1_City
c                   clear                   #1_State
c                   clear                   #1_Country
c                   clear                   #1_Postal
c                   clear                   #1_Phone
c                   clear                   #1_Fax
c                   clear                   #1_eMail

 * Set Program/Display Variables
c                   eval      ControlFlag = DsplyScn#1

c                   EndSr
```

$Reset- Reset Subroutine

The reset routine is used to reset the program work fields. The position to fields for a subfile program will be reset and the navigational flag will be reset. If the program is in add mode the reset routine will execute the $InizScn#1 routine.

The following is an example of the $Reset Subroutine.

```
 *-------------------------------------------------------------------
 * $Reset - Subr To Reset Program
 *-------------------------------------------------------------------
c     $Reset        BegSr

c                   eval      *in04 = *on
c                   eval      #1_Dbr = *loval
c                   if        PgmSubr <> '*INIT'
c                   reset                   ##_Line#
c                   reset                   ##_Column#
c                   eval      ControlFlag = DsplyScn#1
c                   endif
c                   if        Mode = AddMode
c                   exsr      $InizScn#1
c                   endif

c                   EndSr
```

$ExitPgm - Exit ProgramSubroutine

The purpose of the exit program subroutine is to terminate program execution and return control to the calling program and/or menu. The exit routine will turn on indicator *INLR and perform the RETURN operation.

The following is an example of the $ExitPgm Subroutine.

```
 *-------------------------------------------------------------------
 * $ExitPgm - Subr To Exit/Terminate Program
 *-------------------------------------------------------------------
c     $ExitPgm      BegSr

c                   eval      *inlr = *on
c                   return

c                   EndSr
```

$List - List Subroutine

The list subroutine is placed in the templates to allow you to place F4 list prompts on fields you desire. If the field being processed is not coded for the F4 prompt, a message will be sent to the user informing them that the list function is not available for the field selected. If you would like to add a field and have the F4 prompt capability you only need to add a WHEN statement under the select in the list routine.

In the following example, when the F4 key is pressed on the #1_Key field, the program would call SFLWIN with the #1_Key as a parameter.

The following is an example of the $List Subroutine.

```
 *------------------------------------------------------------------------
 * $List  - Subr To Process F4-List Request
 *------------------------------------------------------------------------
c     $List         BegSr

c                   select
 * Perform F4-List For Unique Id
c                   when      ##_Field = '#1_KEY'
c                   call      'SFLWIN'                              99
c                   parm                    #1_Key

* Perform F4-List For

 * Otherwise, Send Message That List Not Defined
c                   other
c                   reset                   Mh_SndRmvDs
c                   eval      Mh_MsgData = ##_Field
c                   eval      Mh_Msg# = 'INF0008'
c                   call      'QMHSNDPM'    MhSndPm
c                   add       1             MsgCount
c                   endsl

c                   EndSr
```

*InzSr - Initialization Routine

The initialization routine is executed once every time your program is called. This routine will be the first routine to run in most template programs. The template programs use this subroutine to set initial values, translate incoming parameters, and define parameter lists.

The following is an example of what is contained in a *InzSr Subroutine. Not all templates will have the same information in the *InzSr. This is only an example of one templates initialization routine.

```
 *-------------------------------------------------------------------
 * *InzSr - Subr To Initialize Program
 *-------------------------------------------------------------------
c     *InzSr        BegSr

 * Build (Program/Library) Name
c                   eval      ##_PgmLib = '(' + %trim(PgmLib) + '/'
c                                         + %trim(PgmName) + ')'

 * Translate Any Parameters That Have Been Passed Into Program
c                   if        %parameters >= 1
c     Lower:Upper   xlate     p_Mode        Mode
c                   endif
c                   if        %parameters >= 2
c                   eval      #1_Key = p_Key
c                   eval      ControlFlag = 'LoadScn#1'
c                   endif

c                   select
 * If In Add Mode, Initialize Screen & Continue
c                   when      Mode  = AddMode
c                   eval      *in06 = *on
c                   when      Mode  = DeleteMode
c                   when      Mode  = ChangeMode
c                   other
c                   eval      Mode  = InquireMode
c                   eval      *in05 = *on
c                   endsl

 * Execute Reset Subroutine
c                   exsr      $Reset
c                   EndSr
```

Program Entry Parameters

Program Mode

The program mode informs the program being called of what maintenance capabilities should be available to the user when the application is ran. The valid program modes are *ADD, *CHANGE, *INQUIRY and *DELETE.

Application Program Interfaces

Application Program Interfaces or API's are provide by the IBM[tm] operating system for use in applications. Because they are provided by IBM[tm] the functionality will remain the same through out new versions of the operating system.

Send Program Messages (QMHSNDPM)

The send program messages API sends a message to the programs message queue.

Send Program Messages Parameters

The required parameters for the send program message is contained in the parameter list, MhSndPm. The parameter list is displayed below.

```
 * Parameter List For Send Program Messages API (QMHSNDPM)
c     MhSndPm       plist
c                   parm                    Mh_Msg#
c                   parm                    Mh_MsgFile
c                   parm                    Mh_MsgData
c                   parm                    Mh_MsgLen
c                   parm                    Mh_MsgType
c                   parm                    Mh_MsgPrev
c                   parm                    Mh_MsgPsc
c                   parm                    Mh_MsgKey
c                   parm                    Error
```

Field Name	Description
Mh_Msg#	Message ID. The identifying code for the predefined message being sent.
Mh_MsgFile	Message File. The name of the message file and the library in which it resides. The first 10 characters specify the file name, and the second 10 characters specify the library.
Mh_MsgData	Message Data. If a message identifier is specified, this parameter specifies the data to insert in the predefined message's substitution variables.
Mh_MsgLen	Message Length. The length of the replacement data or impromptu message text, in bytes.
Mh_MsgTyp e	Message Type. The type of the message. You must specify one of these values, *COMP, *DIAG, *ESCAPE, *INFO, *INQ, *NOTIFY, *RQS or *STATUS.
Mh_MsgPrev	Call Stack Entry. The call stack entry to send the message to, or the call stack entry to start counting from when using a value other than 0 for the Call stack counter parameter. The call stack entry you specify must be in the call stack.
Mh_MsgPsc	Call Stack Counter. A number identifying the location in the call stack of the call stack entry to whose message queue the message is to be sent. The number is relative to the call stack entry identified by the Call stack entry parameter. It indicates how many calls up the call stack the target entry is from the one identified by the Call stack entry parameter.
Error	Error Code Parameter

Send & Remove Program Messages Data Structures

```
 *----------------------------------------------------------------------
 * Internal Field Desc For Message Handler API's - QMHSNDPM & QMHRMVPM
 *----------------------------------------------------------------------
d Mh_SndRmvDs     ds
d  Mh_Msg#                       7a
d  Mh_MsgFile                   20a   inz('PGMMSGF   *LIBL     ')
d  Mh_MsgData                  256a
d  Mh_MsgLen                     9b 0 inz(256)
d  Mh_MsgType                   10a   inz('*INFO')
d  Mh_MsgPrev                   10a   inz('*')
d  Mh_MsgPsc                     9b 0 inz(0)
d  Mh_MsgKey                     4a
d  Mh_MsgRmv                    10a   inz('*ALL')
```

Remove Program Messages (QMHRMVPM)

The Remove program messages API removes messages from the program message queue.

Remove Program Messages Parameters

The required parameters for the remove program message is contained in the PLIST MhRmvPm. The PLIST is displayed below.

```
 * Parameter List For Remove Program Messages API (QMHRMVPM)
c     MhRmvPm       plist
c                   parm                    PgmName
c                   parm                    Mh_MsgPsc
c                   parm                    Mh_MsgKey
c                   parm                    Mh_MsgRmv
c                   parm                    Error
```

Field Name	Description
PgmName	Program Name. The program message queue that the message(s) will be removed from.
Mh_MsgPsc	Call Stack Counter. A number identifying the location in the call stack of the call stack entry to whose message queue the message(s) are to be removed. The number is relative to the call stack entry identified by the Call stack entry parameter. It indicates how many calls up the call stack the target entry is from the one identified by the Call stack entry parameter.
Mh_MsgKey	Message Key. This is the key to the sender's copy of the message in the sending program's message queue.
Mh_MsgRmv	Message Remove. Informs the API of what message should be removed from the program message queue. The templates pass "*ALL" in this parameter to remove all messages.
Error	Error Code Parameter

Navigational Flag Processing

Navigational flag processing is a programming technique used to structure the mainline of a program. The benefits of Navigational flag programming are listed below.

1) Mainline is kept to a minimum.
2) Maintenance to the program is very straight forward.
3) Novice programmers can pick up the flow of the program quickly.
4) Debugging of the program is simplified.
5) Program logic is straight forward and manageable.

Navigational Flag Processing Constants

The data structure associated with navigation flag processing is displayed below. All values that can be used with the ControlFlag field are defined in named constants. The named constants are used to control how the program will function in the mainline. The named constants were added to the templates to allow you to change the values without changing multiple lines in the template programs. An example of this would be if you wanted the display screen 1 named constant to contain 'DISPLYSCR1' instead of 'DSPLSCR1'. You could accomplish this by changing the named constant instead of changing every reference to the field in template program.

```
 *-------------------------------------------------------------------
 * Navigational Constants
 *-------------------------------------------------------------------
d DsplyScn#1      c                   'DsplyScn#1'
d LoadScn#1       c                   'LoadScn#1'
d ValidScn#1      c                   'ValidScn#1'
d UpDateScn#1     c                   'UpDateScn#1'
d DeleteScn#1     c                   'DeleteScn#1'
d InizScn#1       c                   'InizScn#1'
d Reset           c                   'Reset'
d ExitPgm         c                   'ExitPgm'
```

Program Main Line Routine

The main program loop for a template program is listed below. The program will check the value contained in the ControlFlag field if it equals the field that is being compared against the subroutine will be executed. For example if ControlFlag equal the value contained in DsplyScn#1 the subroutine $DsplyScn#1 will be executed.

```
 *------------------------------------------------------------------------
 * Program Main Line Routine
 *------------------------------------------------------------------------
c                   do        *hival
 *
c     ControlFlag   caseq     DsplyScn#1    $DsplyScn#1
c     ControlFlag   caseq     LoadScn#1     $LoadScn#1
c     ControlFlag   caseq     ValidScn#1    $ValidScn#1
c     ControlFlag   caseq     UpdateScn#1   $UpdateScn#1
c     ControlFlag   caseq     DeleteScn#1   $DeleteScn#1
c     ControlFlag   caseq     InizScn#1     $InizScn#1
c     ControlFlag   caseq     Reset         $Reset
c                   cas                     $ExitPgm
c                   endcs
 *
c                   enddo
```

Manipulation Of A Navigational Flag Program

The subroutine that will be executed is determined by the value in the ControlFlag field. For example your program is in the display screen subroutine and you wanted to validate any entries that a user has made. You would change the ControlFlag field to ValidScn#1. This would cause the navigational flag loop to execute the $ValidScn#1 subroutine. If errors existed in the validation routine you change the ControlFlag field to DsplyScn#1 to redisplay the screen with the errors that were found.

Translation Table Constants

The translation tables are used to convert alpha fields to upper case. The conversion allows consistency in entry parameters to ensure proper function of the program. Below are the data structures used in the conversion of the parameters to upper case.

```
 *---------------------------------------------------------------------
 * XLATE - Constants
 *---------------------------------------------------------------------
d Lower           c                   'abcdefghijklmnopqrstuvwxyz'
d Upper           c                   'ABCDEFGHIJKLMNOPQRSTUVWXYZ'
```

*ENTRY Parameters Translation

The parameter translations occur in *INZSR for all parameters passed to the program. Below is an example of the translation calcs. The XLATE operation code translates the values in the fields specified from lower case to upper case.

```
 * Translate Any Parameters That Have Been Passed Into Program
c                   if        %parameters >= 1
c     Lower:Upper   xlate     p_Mode        Mode
c                   endif
c                   if        %parameters >= 2
c                   eval      #1_Key = p_Key
c                   eval      ControlFlag = 'LoadScn#1'
c                   endif
```

DSPRCDLCK

The DSPRCDLCK program will display the following screen when a record lock occurs. This screen will inform you of the record #, library, file and the user that has the record locked.