# Soft Coded Functions Guide

## Introduction

The first programs I wrote were on the S/3 Mod 15D. The largest program you could present to the system was 16K, unless you took advantage of the Shared Virtual Area (SVA) which would allow you to max out a program at 48K. Obviously with only 16K to play with, programs were generally limited to a single function, and typically complex processes would have to be built with a job stream of small programs, sequentially staged to execute against a common set of files.

The S/38 had few limits in terms of program size and IBM did encourage monolith code development. I only attempted about 3 applications before I realized the model was all wrong. Yes, loading everything into main memory at once does mean that execution is fast (assuming your monstrous program does not get paged out, which they invariably did). But the trade-off in terms of maintainability simply wasn't worth marginal performance gains.

I began looking for a better model for programs where functionality could be driven by database entries instead of "hard-coded" instructions. I made a conscious attempted to strip common functions out of programs and externalize as many of those functions as I could. From that point forward, all my applications were modular in nature, limited in function and database driven.

I borrowed an idea from Bob Cozzi to place program function keys into a database file and have a program react to the instruction on file, rather than react to the function key itself. I was able to externalize functions, separating function definitions from the program. I extended the process to include external program options and designed database files to contain program options and function keys, turning them into macro instructions.

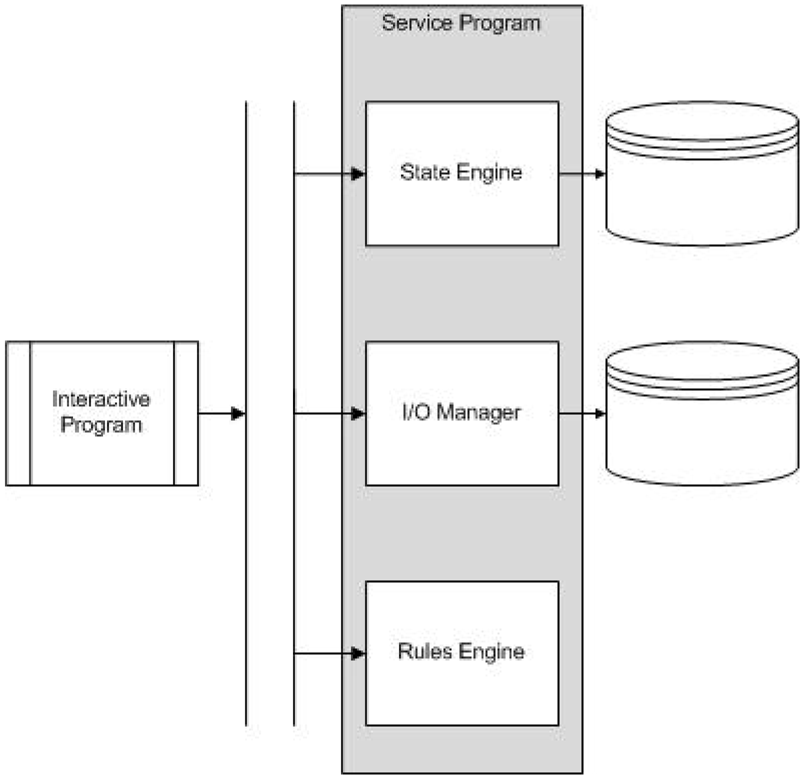
At the time the AS/400 was introduced, the payback to this methodology became clear. Converting from the S/38, where F1 was an exit key to the AS/400, where F3 was the exit key, the programs designed with soft-coded functions did not require any changes, only the database file was updated—F1 entries in the function key database were simply updated to read F3, instead of F1. No programs or displays were changed in order to implement the new system standard of F3 as the exit function.

Shortly after the AS/400 arrived I incorporated a gate-level application security system into the code. This, in effect, made application security an external function, controlled by a database file. I also introduced external program help, by incorporating a generic help processor into the soft-coded function key processor. (The concept of externalized help processes took shape after a COMMON discussion with Carson Soule.)

As the IBM mid-range platform evolved, I began updating what I had come to think of as the soft-coded application model. I began developing a method of separating presentation from data. This was more of an evolutionary step, rather than a revolutionary step. Having created a mechanism for external options, function keys, application security, and help, web or not, the next logical step was to separate the presentation of the data from the data. With the advent of ILE, this became the SOFTCODE service program.

## Design Overview

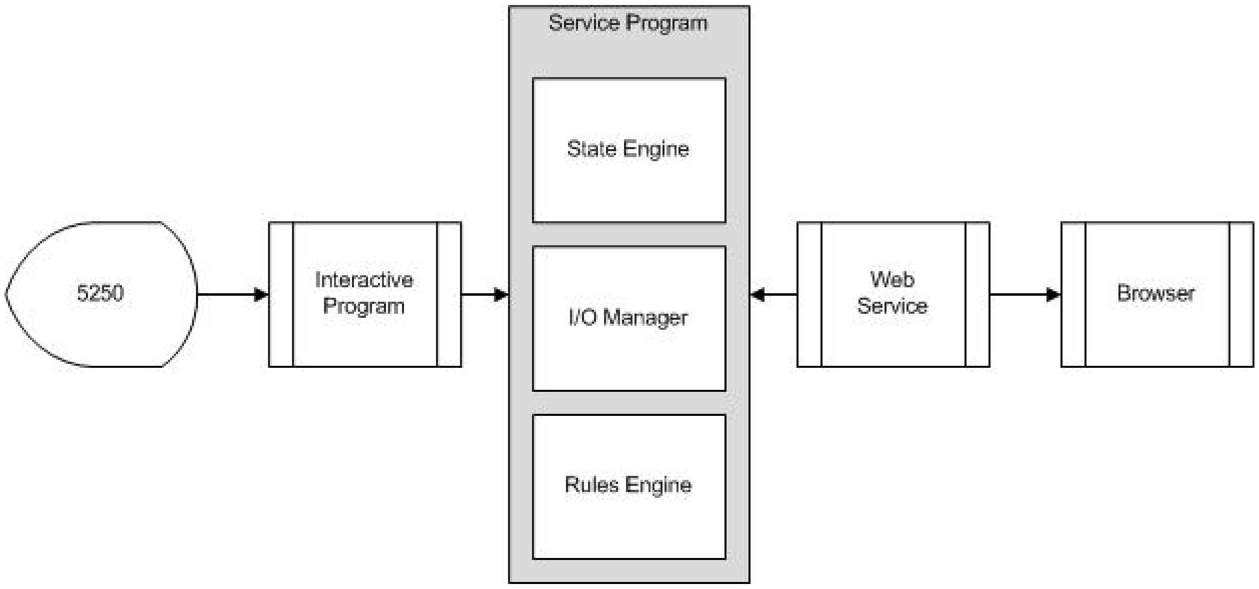
The program design concepts used in the Soft Code system are not necessarily unique in theory, but rare in implementation. Though written primarily in RPG, the development model varies greatly from the early models of top-down procedural codes familiar to most long-time RPG developers. It is far more similar to the modular, event driven concepts that are found in the stateless environment of CGI development, or web processes. In fact the service program and modules more closely resemble a Java bean than traditional RPG application development.



In the Soft Code environment, programs are generally constructed without containing a database file. Instead the program is bound to a service program or file manager module which contains to data to be acted on. Typically an interactive program only manages the presentation of the data and the data management itself is left to the I/O manager.

This separation of presentation from data (and business rules) allows the RPG application to be relatively small. Keeping the code compact makes maintenance easier and faster. And since time is money, it also is cheaper to maintain throughout the Software Development Life Cycle (SDLC).

There is also an interesting by-product of this very modular style development; re-usable functions. Since the data management is separate from the data presentation, web applications and traditional 5250-style applications can share the same data and data services.



There is no real need to replicate rules or I/O functions in order to serve up information. It has the added benefit of applying the same rules to data from the web side, or the 5250 side of the equation. Since the information is presented from a common source, the information displayed on a 5250 screen will match exactly the information presented through the browser—since the data source is the same. This type of application structure produces reusable code and fits into the idea of creating SOA type applications.

At the highest conceptual level, applications should simply provide services. The popular SOA (Service Oriented Architecture) concept describes the goal fairly well. Because ultimately, isn't that what a business, any business, is about—providing services to a customer? Of course, without further definition, SOA is just another TLA (Three Letter Acronym).

The W3C defines SOA as:

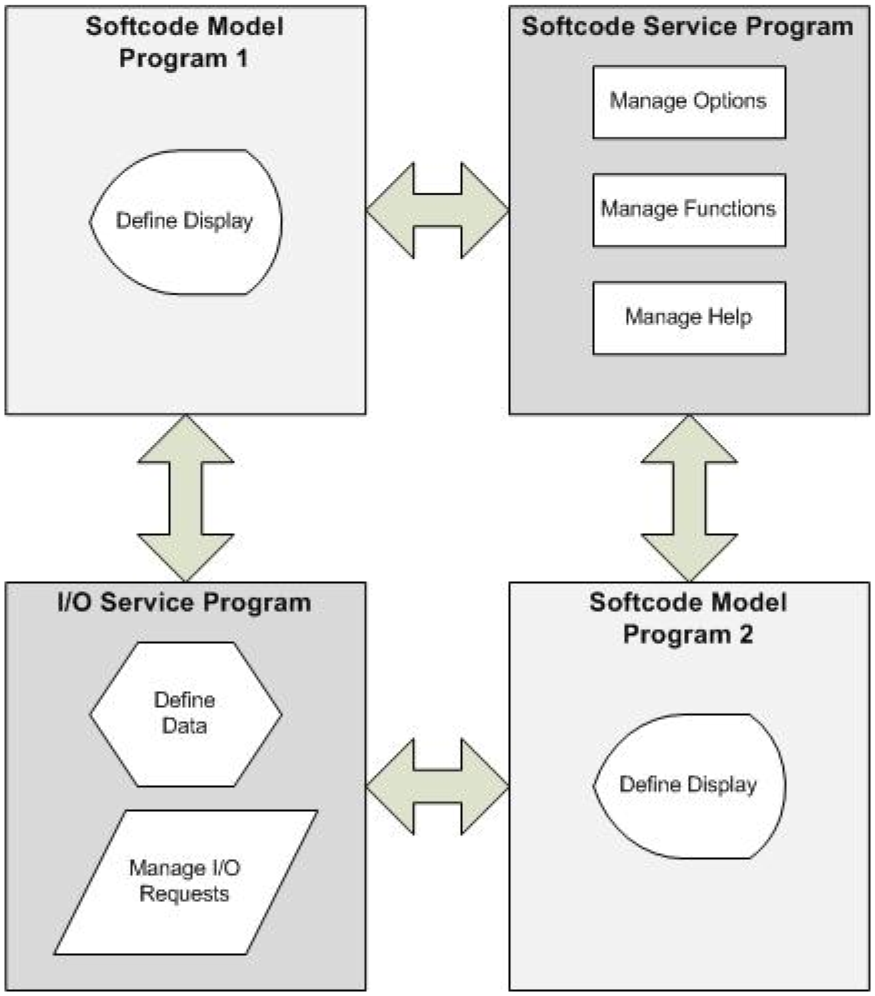
*"A set of components which can be invoked, and whose interface descriptions can be published and discovered."*

Quite frankly, with apologies to the W3C, that definition falls short of the mark. In the first place, components don't always appear as part of a set. Secondly, by W3C definition, SOA is only comprised of implemented and deployed components, rather than the model on which they were built, (the ARCHITECTURE, if you please). Architecture implies a style, deployed or not. A more practical definition of SOA might be:

*The standards, practices, and models, that enable application functionality to be provided and consumed as services published to satisfy the requirements of the service consumer. Services may be invoked, published and discovered, regardless of implementation through a single interface, based on a common (standard) form.*

The current IBM Power Systems platform offers the opportunity to move forward toward SOA without the expense of redeveloping legacy applications—embracing the soft-coded application model rather than traditional top-down monolithic structures can:

1) Leverage existing applications
2) Help provide web services
3) Require minimal training of technical personnel



With the introduction of the Integrated Language Environment (ILE) IBM offered a model of how to build models aimed at providing services from a common interface. Service programs are very much like Java classes. A Java class may be composed of a number of methods, bound together. If a class is imported to an application, the methods of the class are directly available to the application.

Services programs may be composed of a single language, such as RPG, or constructed of modules of many different languages, C++, COBOL, CL, etc. The components of the service program are referred to as procedures and resemble methods. Like a class, once a program is bound to the service program, the application will have access to the procedures contained in the service program. System i tools, such as IBM's WDSC (Websphere Development Studio Client), will allow service programs to be exposed as web services.

## *SOFT CODED FUNCTIONS*

### Soft coded functions: Intro

New interactive application models should be designed with ILE in mind. Binding to the SOFTCODE service program to manage function key use and program options leaves a programmer free to create an application program independent of command key assignments, or program options. With the support programs in place, the Soft Coded Function keys (SCF), and Soft Coded Program options (SCP) allow for the development of concise data-centric application programs.

As with any application development effort, keep in mind a few rules. Whether using Soft Code design templates or not, do not write programs that attempt to perform too many functions.

Rule of thumb for RPG programs:

a) A program under 650 lines is ideal.

a) A program of 700-900 lines is acceptable.

b) A program of 1,200 lines is unwieldy.

c) A program of 1,500 lines is approaching unmanageable.

d) A program of 1,700 lines is unmanageable.

e) A program of 2,000 lines could be considered epic.

f) A program over 2,000 lines is something to terrify rookie programmers with.

## SCF/SCP Commands

Since function keys and program options are contained in a database rather than in the program, a number of the applications within the Soft Code system have command interfaces created to invoke SOFTCODE processes from a command line. The following commands may be entered to invoke Soft Code processes.

```
CPYSFTDTA: Copy soft code data
EDTPNLKEY: Edit panel keys
EDTPNLOPT: Edit panel options
WRKPNLKEY: Work with panel keys
WRKPNLOPT: Work with panel options
WRKSECGTE: Work with security gate
WRKSFTUSR: Work with Soft Code Users
```

This set of commands provides an easy to remember management suite for the applications designed to use the soft coded utilities. If they seem familiar, they should. The command syntax mimics the IBM syntax of verb and object.

This suite of commands allows a developer to create and maintain application security, externally defined panel options, and externally defined function key events.

## SCF/SCP Database Files

The program function keys and macro instructions are contained in the data base file, SCFUNCPF. The program options and option macros are contained in the database file, SCOPTNPF. By removing the function keys and options from the program, the operations performed by the options and function keys may be maintained independently of the program itself.

Function key assignments and program options may be removed from the application without changing the program code. Conversely adding new functions or program options can be performed without changing the program code. (Unless a new parameter list happens to be introduced.)

Because the function editor manages a program’s options and functions, commands, such as DSPMSG, or WRKACTJOB may be added to a program without any change to existing program code. The Soft Code editor is coded to recognize a command (a string starting with an ampersand ‘&’) and execute the command. The program does not have to contain a routine to execute a command, the function editor with provide the capability. In effect, the database files supporting the Soft Code provide a method of building macro instructions for applications to share.

Another benefit of moving program functions and objects from the actual program (and display file) is the text of the function is stored with the function. The text displayed for the function or option is contained in the database along with the macro instruction. Thus a program can be designed to be multi-lingual. On one application the F3 (Exit) function may read ‘Exit’ and in another application the same function may read ‘Salida’ (Spanish), or ‘Ausgang’ (German).

A Soft Code application does not really react to a function key, or an option. The editor looks up the function key or panel option and returns the ‘function’ to the requesting program. This increases the flexibility of the application and allows it to be compatible with other system i environments. For example a program may be coded to exit when the macro instruction returned is ‘EXIT’. The macro ‘EXIT’ may be assigned to F3, to be consistent with AS/400 mode panels, or F7 (End Job) so it is familiar to users used to working in the S/36 environment.

## Macro Instruction Format

The macro instruction field of the soft coded program options and function keys is 45 bytes long. This instruction may contain several different types of operations. And in Soft Code, the macro instruction may take several different formats.

An ampersand '&' in byte 1 identifies a CL command. The Soft coded function editor will pass this command on to the generic command executive program to be processed.

```
    000000000111111111122222222223333333333444444
    123456789012345678901234567890123456789012345

    &WRKACTJOB
    |
    |____________Command to process
```

An asterisk (*) in byte 1 signals a reserved function. Code has been included in the soft-coded function editor to process the macro instruction, it will not be returned to the requesting program.

```
    000000000111111111122222222223333333333444444
    123456789012345678901234567890123456789012345

    *HELP
    |
    |____________Soft coded internal process
```

With no special characters imbedded in the instruction, the macro instruction will be returned to the requesting program for execution. In this case, the format is left to the requesting program to interpret. A standard format for macro instructions to drive subroutines is evident in the process of command keys and other common program operations.

```
EXSR @EditKeyPressed
SELECT;
WHEN SUBOP = 'CALL';
   EXSR @CALLS;
WHEN FUNCT = 'EXIT';
   QUIT();
WHEN FUNCT = 'RESET';
   EXSR @RESET;
WHEN FUNCT = 'PROMPT';
   EXSR @PROMPT;
WHEN FUNCT = 'MORE';
   DisplayKeys(cmdkey: z$key1: z$key2: M);
WHEN FUNCT = 'MOREOPT';
   DisplayOptions(option: z$opt1: z$opt2: O);
ENDSL
```

The sample code is designed to examine the macro (FUNCT) and perform the appropriate subroutine; such as display more keys (DisplayKeys), or more program options (DisplayOptions), or to exit the program (@EXIT).

## Complex Macro Instruction

```
0000000001111111111222222222233333333334444444
1234567890123456789012345678901234567890123445
<--->< -------->< -------->< ---------------->
CALL SC0335RP  PLIST1    *RESERVED          V
|    |         |                            |
|    |         |                            |___Action Code
|    |         |
|    |         |
|    |         |
|    |         |________________________________PARM list
|    |__________________________________________Program name
|_______________________________________________Operation code

 1-05 operation code
 6-15 program
16-25 program data structure name
26-44 RESERVED
45-45 Action code:   A = Add
                     C = Change
                     D = Delete
                     V = View (display only)
                     S = Sort
```

A file, SCMACRPF, has been created to be used as an externally defined data structure which can be used in a program to define the macro. Using the externally defined macro provides consistency in naming the various parts of the complex macro format.

## Editing Macro Instructions

Example:

The following D specs contain several externally defined data structures commonly associated with the SOFTCODE development process.

```
      *================================================================
     D CF             E DS                  EXTNAME(SCKEYSPF)
     D PGMDS          ESDS                  EXTNAME(SCPSTSPF)
     D DSPDS          E DS                  EXTNAME(SCDSPFPF)
     D MACDS          E DS                  EXTNAME(SCFUNCPF) INZ
     D OPTDS          E DS                  EXTNAME(SCOPTNPF) INZ
     D FUNCT          E DS                  EXTNAME(SCMACRPF) INZ
     D GATEPR         E DS                  EXTNAME(SCGATEPF)
```

Though it is not a recommend practice, the macro instruction may be defined internally using RPG I-specs.  An example of the the program defined data structure re-defines the file defined function macro (field name, FMACRO,) into its various components; the op-code, subprogram, calling parameter list, and action.

```
     I            DS
     I                                         1  45 FUNCT
     I                                         1   5 SUBOP
     I                                         6  15 SUBPGM
     I                                        16  25 CALLPM
     I                                        45  45 SUBACT
```

This subroutine is an example of using the format of the complex macro instruction to format dynamic calls based on the information provided in the macro instruction.

```
        BEGSR @CALLS;
           EXSR @SETPM;
           MONITOR;
           SELECT;
              WHEN CALLPM = 'PLIST1';
                 CALLP WithParms(p$gate: p$catg: p$mode: p$rtn);
              OTHER;
                 CALLP NoParms();
           ENDSL;
           ON-ERROR;
              P$ERR = 'MIS0012';
           ENDMON;
           EXSR @RETPM;
        ENDSR;
```

A display of the soft coded command keys (SCF) for a program shows entries for internal macro instructions, including one formatted for the @CALLS subroutine.

```
SCROYA1    SC0020RP      iSoftwerks Incorporated          SYSNAME   4/02/10
SCROY      634042        SOFT FUNCTION EDITOR             SC0020S1 08:12:45

  APPLICATION: SC0320RP    PANEL:            AUTHORITY LEVEL: 000

 Key ID     Macro Function, program call or command       Function text
 F21        &CALL QUSCMDLN                                Command line
 F23        MOREOPTS                                      More options
 F24        MOREKEYS                                      More keys
 F3         EXIT                                          Exit
 F4         PROMPT                                        Prompt
 F5         RESET                                         Refresh
 F9         CALL SC0335RP  PLIST1                       A Add item







                                                                            +
                           F3=Exit
```

FIG. 1

An Add item request has been entered for the application SC0335RP. The macro has been coded to instruct the requesting program to call the program using the parameter list; PLIST1.

```
  F6     CALL SC0335RP  PLIST1                      A Add item
  |      |    |         |                           | |
  |      |    |         |                           | |_Text
  |      |    |         |                           |___Action code
  |      |    |         |_______________________________Parameter list
  |      |    |_________________________________________Program
  |      |______________________________________________Op code
  |_____________________________________________________Function key
```

The request to call SC0335RP has been coded to signal the requesting application to use the parameter list, PLIST1, to invoke the program.

## *Soft Code Support Procedures*

The SOFTCODE service program contains a number of procedures to manage the maintenance and functionality of Soft Code application development. The procedures listed on this page show the range of functions defined within the service program.

### SOFTCODE Service Program Exports

| | | |
|---|---|---|
| CHECKDEFHEADER | SETGATECAT | SETTABCODE |
| CLEARDEFHDR | SETGATECURSOR | SETTABLEID |
| CLOSEDEFHEADER | SETGATELEVEL | UPDATETABCODE |
| DELETEDEFHDR | SETGATENAME | ACTIVATEUSER |
| FINDDEFHDR | UPDATEGATE | CHECKUSERCTL |
| GETDEFINITION | ADDOPTION | CHECKUSERDATA |
| GETNEWTCODE | CHECKOPTNDATA | CLEARUSERREC |
| INSERTDEFHDR | CLEAROPTNREC | CLEARUSRCTL |
| NEXTDEFHDR | CLOSEOPTIONCURSOR | CLOSEUSERCURSOR |
| SETDEFCODE | DLTOPTION | CLOSEUSRCTLCSR |
| SETDEFDESC | GETNXTOPTION | DELETEUSER |
| SETDEFINECURSOR | GETPGMOPTION | DELETEUSRCTL |
| SETDEFNAME | OPTIONFOUND | DLTALLUSRCTL |
| UPDATEDEFHDR | SETOPTIONID | EXPIREUSER |
| ADDFUNCTION | SETOPTIONLVL | GETNEXTUSER |
| CHECKFUNCDATA | SETOPTMACRO | GETNEXTUSRCTL |
| CLEARFUNCREC | SETOPTNACTN | GETUSERDATA |
| CLOSEFUNCTIONCURSOR | SETOPTNCALL | GETUSRCTL |
| DLTFUNCTION | SETOPTNCURSOR | INSERTUSER |
| FUNCTIONFOUND | SETOPTNPARM | INSERTUSRCTL |
| GETNXTFUNCTION | SETOPTNPGMID | SETSUBSTITUTE |
| GETPGMFUNCTION | SETOPTNPNLID | SETUSERCAT |
| SETFUNCACTN | SETOPTNTEXT | SETUSERCTL |
| SETFUNCCALL | UPDOPTION | SETUSERCURSOR |
| SETFUNCCURSOR | CHECKTABLEENTRY | SETUSEREDTE |
| SETFUNCKEYID | CLEARTABCODE | SETUSEREMAIL |
| SETFUNCLEVEL | CLOSEDETAILCURSOR | SETUSERLEVEL |
| SETFUNCPARM | DELETEDETAIL | SETUSERNAME |
| SETFUNCPGMID | DELETETABCODE | SETUSERXDTE |
| SETFUNCPNLID | DETAILENTRIES | SETUSRCTLCSR |
| SETFUNCTEXT | FINDTABCODE | UPDATEUSER |
| SETPGMMACRO | GETTABABBR | UPDATEUSRCTL |
| UPDFUNCTION | GETTABCODE | USERFOUND |
| CLEARGATEREC | GETTABDATA | USRCTLFOUND |
| CLOSEGATECURSOR | GETTABDESC | CHECKOBJECT |
| DELETEGATE | INSERTTABCODE | DISPLAYKEYS |
| GATEFOUND | NEXTTABCODE | DISPLAYOPTIONS |
| GETGATEDATA | SETCODABBR | GETFUNCTION |
| GETNEXTGATE | SETCODDESC | GETKEYTEXT |
| INSERTGATE | SETDETAILCURSOR | GETOPTION |
| | | GETOPTTEXT |
| | | SQLERROR |

## Defining a function key

A single module is required for the SCF is FUNCTIONKEYS a procedure designed to load mnemonic variables with the hex code pattern for function keys. This provides a straight forward method of command function coding without the use of indicators. It also has the added advantage of easily interpreted code.

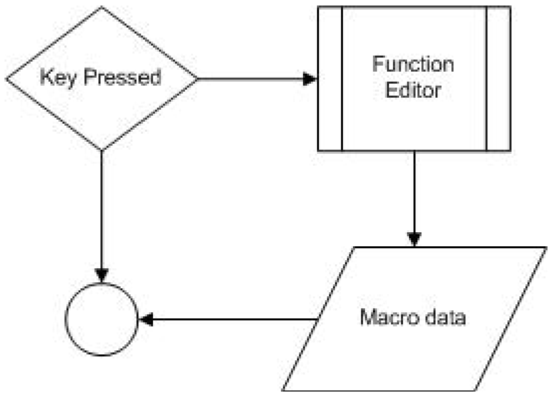
1) Define the key structure with the externally defined file SCKEYSPF
2) Define the AID byte for the last key pressed using the data structure DSPDS

```
********************************************************************
FSC0190DF CF   E             WORKSTN
F                                     SFILE(SC0190S1:RRNSI)
F                                     INFDS(DSPDS)
 *==================================================================

D SC0190RP       PI
D  p$usrp                     10a   Const Options(*nopass)

D FunctionKey   E DS                  EXTNAME(SCKEYSPF) qualified
D PGMDS         ESDS                  EXTNAME(SCPSTSPF)
D DSPDS         E DS                  EXTNAME(SCDSPFPF)
D MACDS         E DS                  EXTNAME(SCFUNCPF) INZ
D OPTDS         E DS                  EXTNAME(SCOPTNPF) INZ
D FUNCT         E DS                  EXTNAME(SCMACRPF) INZ
```

3) Retrieve the hex definition of the keys using the SOFTCODE module FUNCTIONKEYS.
4) Invoke the SOFTCODE procedure GETFUNCTION to return the macro instruction.
5) Execute the function returned.

```
          FunctionKey = FunctionKeys();

          fkeyds = KeyPressed;
          funct = *BLANKS;
          fpgmid = PRGNAM;
          fpnlid = FMTNAM;
          fmacro = *BLANKS;
          MONITOR;
             GetFunction(fpgmid:fpnlid:fkeyds:fkeyid:fmacro:authl);
          ON-ERROR;
             msgid = 'MIS0003';
             EXSR @GetMsg;
          ENDMON;
```

At its simplest, the GETFUNCTION is performing a look-up to determine what function is associated with the key pressed, then having determined that, returns the function macro data to the requesting program.



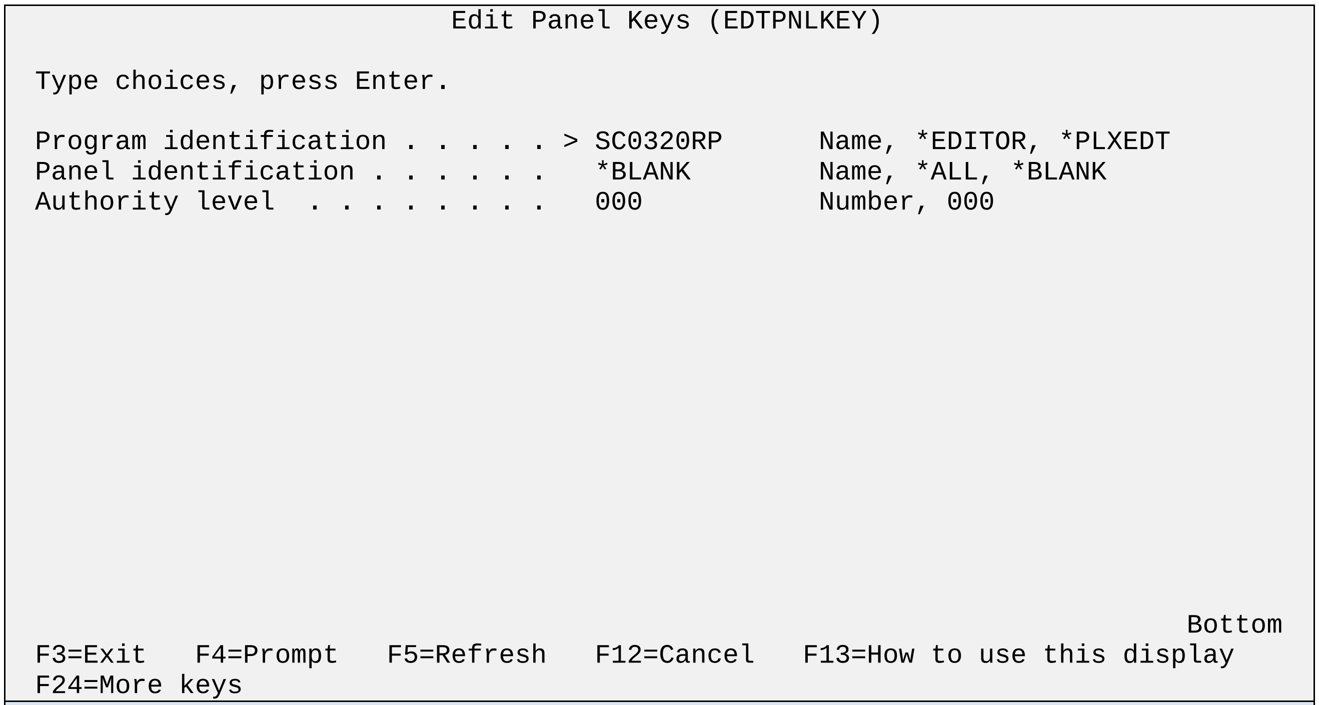
```
Select;
When KeyPressed = FunctionKey.ENTER;
      Exsr @ENTER;
When KeyPressed = FunctionKey.ROLLUP;
      Exsr @LOAD;
When KeyPressed = FunctionKey.ROLLDN;
      Exsr @DOWN;
When SUBOP = 'CALL';
      Exsr @CALLS;
When FUNCT = 'EXIT';
      Quit();
When FUNCT = 'RESET';
      Exsr @RESET;
When FUNCT = 'MORE';
      DisplayKeys(cmdkey: z$key1: z$key2: M);
When FUNCT = 'MOREOPT';
      DisplayOptions(option: z$opt1: z$opt2: O);
Endsl;
```

An in-line case structure usually follows the GETFUNCTION. The case structure may be looking for a specific function, or a number of different functions. In this example the variable, KEYPRESSED is compared against specific function keys, ENTER (qualified notation, FUNCTIONKEY.ENTER), but the other program actions are defined by the function returned by SOFTCODE process.

## Soft coded Authorization Levels

The soft function database has been instructed to include a Function Authorization Level (FAL). The FAL code field coincides with the generic security system authorization level, a three-digit (zero decimal), code field.

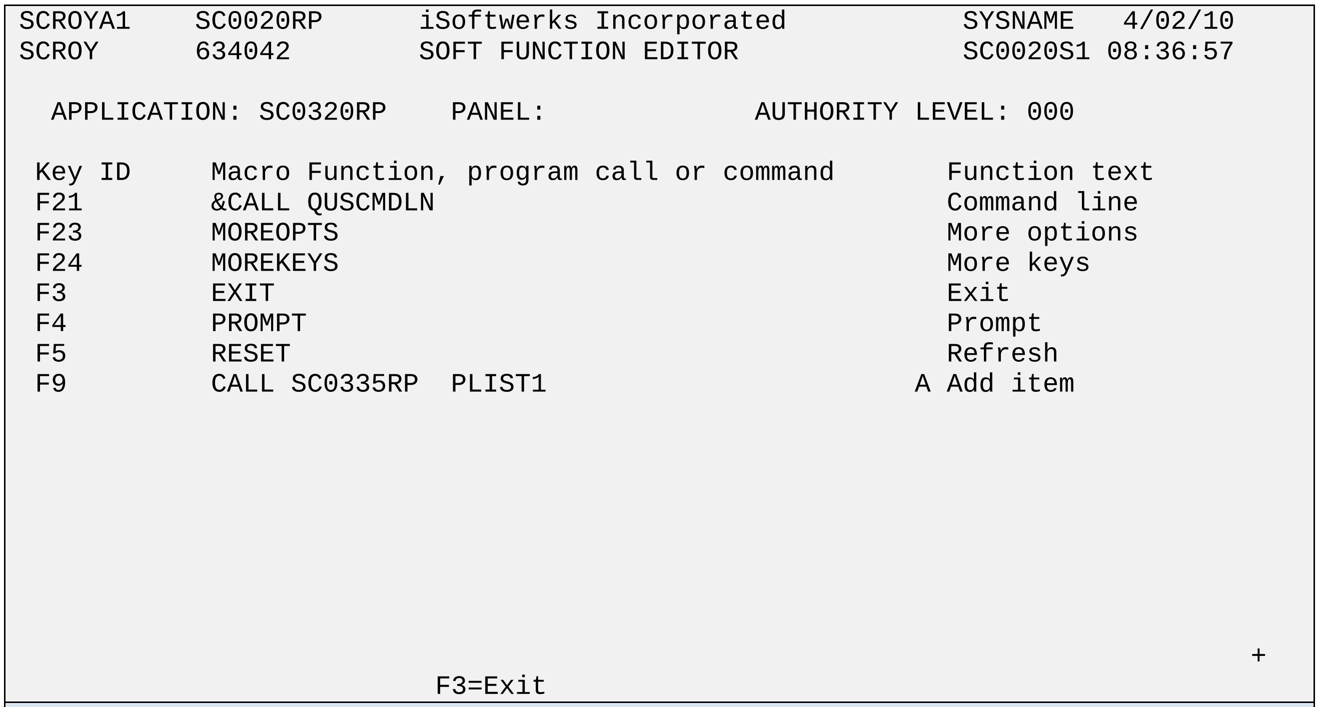
The authorization level is an added command parameter for the Edit Command Key command, EDTCMDKEY, and to the program options command, EDTPGMOPT (Edit Program Options). The default in both commands is an authorization level of zero (000).



```
                          Edit Panel Keys (EDTPNLKEY)

 Type choices, press Enter.

 Program identification . . . . . > SC0320RP      Name, *EDITOR, *PLXEDT
 Panel identification . . . . . .   *BLANK        Name, *ALL, *BLANK
 Authority level  . . . . . . . .   000           Number, 000


























                                                                         Bottom
 F3=Exit   F4=Prompt   F5=Refresh   F12=Cancel   F13=How to use this display
 F24=More keys
```

FIG. 2

The editor displays the application name, the panel name and the authorization level. The editor will allow changes to the function key, macro or text, but not to the authorization level. The panel displayed shows only application entries that correspond to the command parameters, program, panel, and any functions at the designated FAL.



```
SCROYA1    SC0020RP      iSoftwerks Incorporated           SYSNAME   4/02/10
SCROY      634042        SOFT FUNCTION EDITOR              SC0020S1 08:36:57

  APPLICATION: SC0320RP    PANEL:            AUTHORITY LEVEL: 000

 Key ID     Macro Function, program call or command       Function text
 F21        &CALL QUSCMDLN                                Command line
 F23        MOREOPTS                                      More options
 F24        MOREKEYS                                      More keys
 F3         EXIT                                          Exit
 F4         PROMPT                                        Prompt
 F5         RESET                                         Refresh
 F9         CALL SC0335RP  PLIST1                       A Add item







                                                                            +
                           F3=Exit
```

Fig. 3

The parameter lists for the SCF/SCP processor programs include the FAL as a parameter. For example:

```
D GetFunction     PR
D  pgmnam                       10
D  pnlnam                       10
D  keyDS                         1
D  keyID                        10
D  function                     45
D  level                         3

 BEGSR @EditKeyPressed                                                ;
     fkeyds = KeyPressed                                              ;
     funct = *BLANKS                                                  ;
     fpgmid = PRGNAM                                                  ;
     fpnlid = UpperCase(z$mode:%size(z$mode))                         ;
     fmacro = *BLANKS                                                 ;
     MONITOR                                                          ;
        GetFunction(fpgmid:fpnlid:fkeyds:fkeyid:fmacro:authl)         ;
     ON-ERROR                                                         ;
        MessageString ='Error occurred editing function key'          ;
        DisplayMessage(MessageString)                                 ;
     ENDMON                                                           ;
     FUNCT = FMACRO                                                   ;
  ENDSR
```

The authorization level should be set to 999 if the program does not use FAL sensitive displays. However if an application is designed to present command keys to a certain type of user, or different levels of functions for supervisory personnel, then the authority level parm can be used in conjunction with the   security checker program to prevent users from accessing command keys, or program options they are not authorized to perform.

```
D CheckAuthority  PR
D  CkUsr                        10
D  CkGate                       10
D  CkCat                         3S 0
D  CkPass                        1
D  CkLvl                         3
```

```
//* The security checking program tests whether the user is allowed
CheckAuthority(USER:PRGNAM:CATEG:PASSC:AUTHL);
IF PASSC <> 'P';
   QUIT();
ENDIF;
```

If the security program has detected a fault, the security monitory screen presents a pop-up display with the gate ID, program ID and the pass code. This display does not use a display file but the IBM supplied long-text API. It will appear for interactive programs but does not attempt to issue a message display in the batch mode.

## Security monitor example

```
SOFTMENU                    Softcode Services Menu
                                                            System:   SYSNAME
Select one of the following:

..............................................................................
:                        . Security SC0460RM Error .                         :
:   X-Gate record not on file. Contact Information Systems to have the       :
:   gate SC0320RP installed.                                                 :
:                                                                            :
:                                                                            :
:                                                                            :
:                                                                            :
:                                                                   Bottom  :
:  F12=Cancel                                                                :
:                                                                            :
:............................................................................:
    13. Restore save file from IFS
    14. Save save file to IFS
    15. Display softcode documentation

Selection or command
===>
F3=Exit   F4=Prompt   F5=Workbench   F9=Retrieve   F12=Cancel
(C) iSoftwerks, Incorporated 2010
```

Fig.4

Once the authorization level has been established, the program user will only see those function keys for the established authorization level, or lower. Soft functions defined for the application with a higher authorization level will be omitted from the command key, and option text. In addition, even if the function or option is selected, the authority level will prevent an unauthorized user from executing the function.

## *WRKPNLKEY – Work with panel key*

The command, WRKPNLKEY, provides an easy maintenance mechanism for viewing, or editing a program's function keys. It lists all program panels defined to the soft function editor and allows the user to work with an entire panel, or a single panel entry.

```
SCROYA1    SC0300RP       iSoftwerks Incorporated          BLUGRND   4/02/10
SCROY      634042         Work with function keys          SC0300C2 08:52:24

  Type in option, press enter.  Program.: SC0320RP   Panel...: *BLANK
          2=Edit              4=Delete            9=Edit Panel


Op Key          Panel              Macro           Act      Text            Lvl
   F21                   &CALL QUSCMDLN               Command line         000
   F23                   MOREOPTS                     More options         000
   F24                   MOREKEYS                     More keys            000
   F3                    EXIT                         Exit                 000
   F4                    PROMPT                       Prompt               000
   F5                    RESET                        Refresh              000
   F9                    CALL SC0335RP  PLIST1      A Add item             000






                                                                          Bottom
         F3=Exit             F5=Refresh

Copyright (c) iSoftwerks, Inc.
```

Fig. 5

## *WRKPNLOPT – Work with panel options*

The command, WRKPNLOPT, is designed to allow the maintenance of a group of panel options, or individual program options.

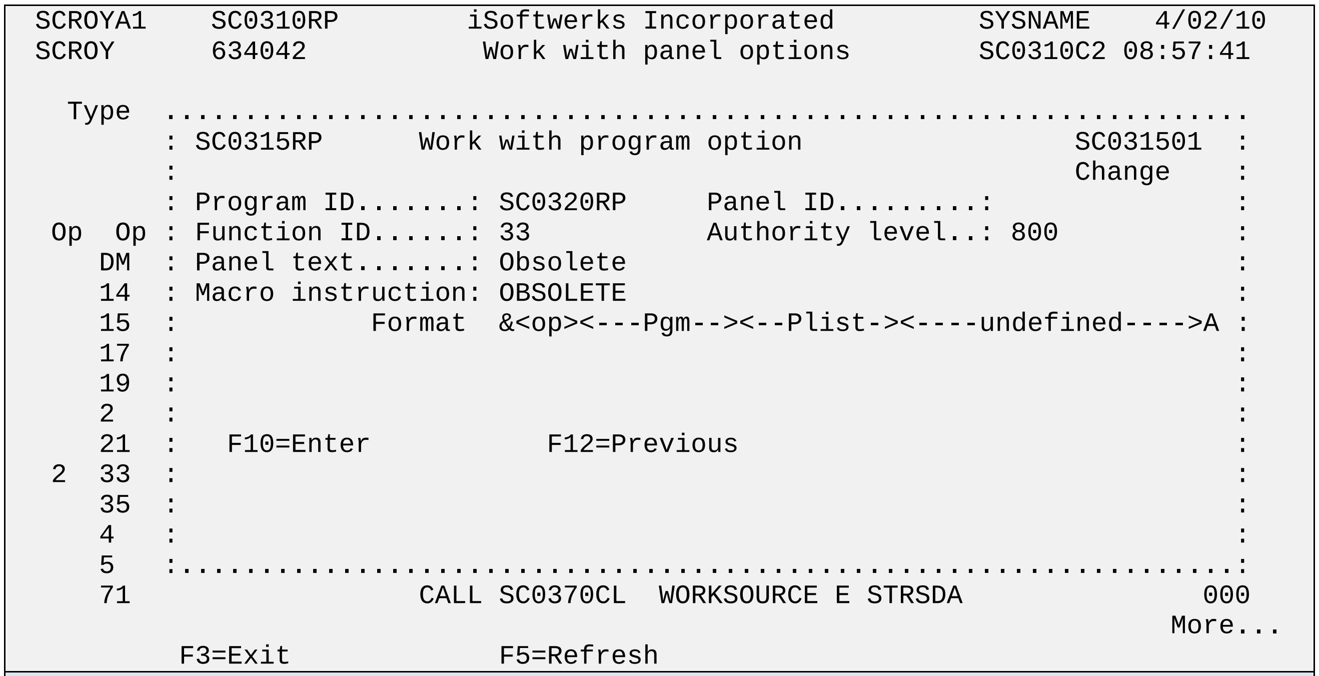
```
SCROYA1    SC0310RP        iSoftwerks Incorporated         SYSNAME   4/02/10
SCROY      634042           Work with panel options        SC0310C2 08:54:22

   Type in option, press enter.  Program.: SC0320RP   Panel...: *BLANK
           2=Edit              4=Delete            9=Edit panel


  Op  Option    Panel            Macro            Act        Text         Lvl
     DM                  &DSPMSG                     Messages             000
     14                  CALL SC0380CL  WORKOBJECT C Create               000
     15                  PROCESS                     Activate             000
     17                  RESET                       Reset                000
     19                  EXCEPTION                   Review               000
     2                   CALL SC0335RP  PLIST1     C Edit                 000
     21                  CALL SC0345CL  WORKSOURCE E Edit source          000
     33                  OBSOLETE                    Obsolete             800
     35                  EXTEND                      Extended             000
     4                   CALL SC0335RP  PLIST1     D Delete               000
     5                   CALL SC0335RP  PLIST1     V View                 000
     71                  CALL SC0370CL  WORKSOURCE E STRSDA               000
                                                                        More...
          F3=Exit             F5=Refresh
```

Fig. 6

The option to edit a panel invokes and RPG program for the Command Processing Program (CPP). The program and display work in similar fashion to the EDTPNLKEY CPP. See the section titled Editing Macro Instructions for a more detailed explanation of the soft function editor processes.

## Editing an Option

The option to edit a single entry calls a subprogram to allow the user to change the option macro information, or the option text, or the security level on the program option.



```
 SCROYA1    SC0310RP        iSoftwerks Incorporated         SYSNAME    4/02/10
 SCROY      634042           Work with panel options        SC0310C2 08:57:41

   Type   ...................................................................
          : SC0315RP      Work with program option                SC031501  :
          :                                                       Change    :
          : Program ID.......: SC0320RP     Panel ID.........:              :
  Op  Op  : Function ID......: 33           Authority level..: 800          :
     DM   : Panel text.......: Obsolete                                     :
     14   : Macro instruction: OBSOLETE                                     :
     15   :            Format  &<op><---Pgm--><--Plist-><----undefined---->A :
     17   :                                                                 :
     19   :                                                                 :
     2    :                                                                 :
     21   :   F10=Enter           F12=Previous                              :
  2  33   :                                                                 :
     35   :                                                                 :
     4    :                                                                 :
     5    :.................................................................:
     71                  CALL SC0370CL  WORKSOURCE E STRSDA               000
                                                                        More...
          F3=Exit             F5=Refresh
```

Fig. 7

## *WRKSFTUSR – Work with Soft Code User*

The command, WRKSFTUSR, provides a maintenance facility to define those users authorized to use soft code functions. The display lists the users defined to the system.

```
SCROYA1    SC0190RP      iSoftwerks Incorporated           SYSNAME   4/02/10
SCROY      634042        Work With Softcode Users          SC0190C2 09:02:26

  Type in option, press enter.                Position to:
          2=Edit              5=View              8=Controls


Op User        Status    Eff Date Exp Date              Email
   SCROY       *active    1/01/00 99/99/99 scroy@gate.net













                                                                        Bottom
         F1=Help             F3=Exit             F5=Refresh
         F9=Add user
Copyright (c) iSoftwerks, Inc.
```

Fig. 8

User records are stored in the file, SCUSRSPF. The individual control entries are contained in the database file, SCUSRCPF.

## User Controls

The user controls option provides a method of assigning the user an authority level within the categories defined by the soft code application system.

A user may be assigned different levels of authority within the various categories defined in the system. A user may be an administrator in Customer Service, but allowed Special Operations rights in the Accounting area.

```
SCROYA1    SC0290RP        iSoftwerks Incorporated         SYSNAME   4/02/10
SCROY      634042        Work with controls for SCROY      SC0290C2 09:03:43


Options: 1=Add               2=Edit              4=Delete


Op Cat     Category Description          Lvl      Level Description
     0                                   000
   100 Customer Relations                500 Operational Security
   200 Financial Services                800 System Level Security
   400 Inventory                         800 System Level Security
   450 Purchasing                        800 System Level Security
   500 Information Systems               700 Adminstrative Security
   600 Administration                    400 Manager Level 2 Security
   700 Executive                         300 Manager Level 1 Security



                                                                       Bottom

         F3=Exit             F5=Refresh

Copyright (c) iSoftwerks, Inc.
```
Fig. 9

Individual programs may be assigned to a category. This is how a group of users may share a common application program and are assigned functions (keys) and options based on the authority level that they are assigned within the application category.

## *Soft Code Tables*

There are two tables key to the soft code operations, the category table and the authority level table. Additional tables may be defined as required, as the number of tables defined is not limited.

```
SCROYA1    SC0230RP        iSoftwerks Incorporated         BLUGRND   4/02/10
SCROY      634042           Work with Definitions          SC0230C2 09:05:04


Options: 1=New definition    2=Edit              4=Delete
         8=Details

 Op Definition                     Description                      Table

    CATEGORY    Application Category Table                             10
    LEVEL       Authorization Level Table                               5








                                                                           Bottom

         F3=Exit

Copyright (c) iSoftwerks, Inc.
```
Fig. 10

## Category table entries

Category table entries are used to define application categories. The current category list defines the following application areas:

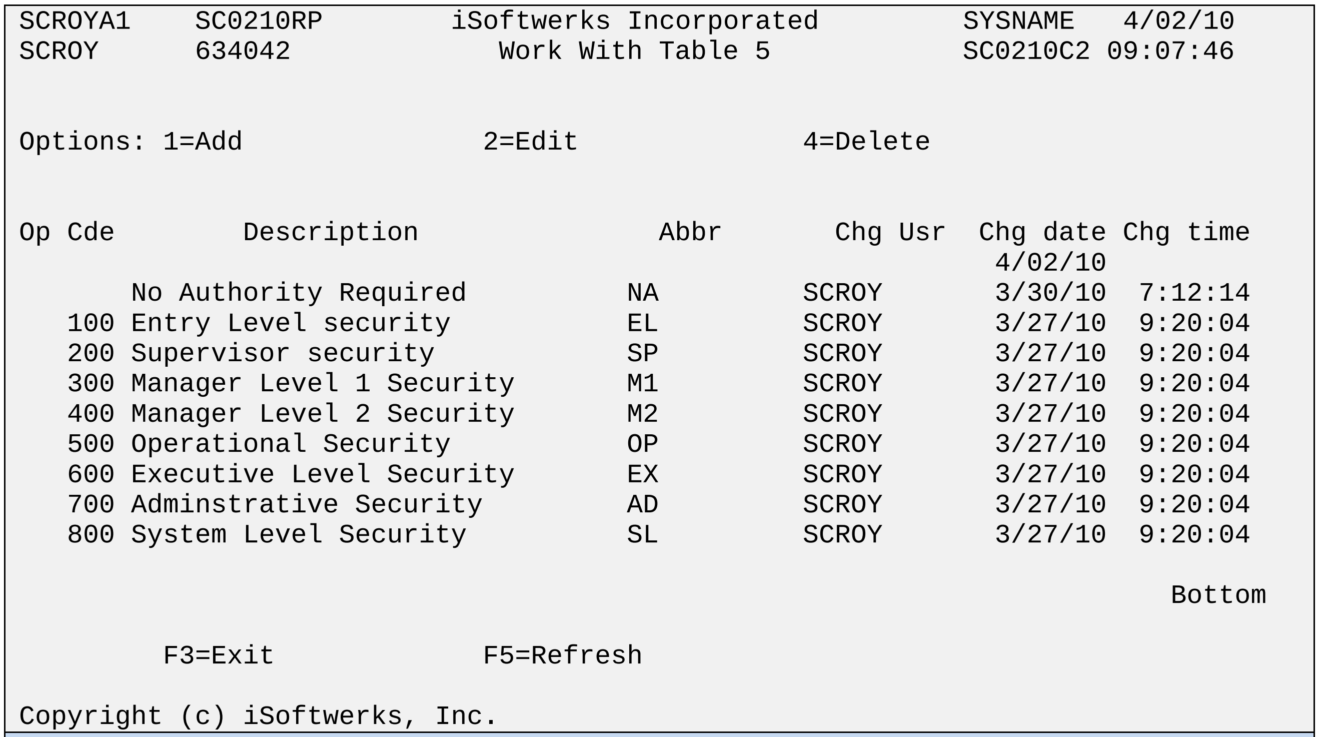
```
SCROYA1    SC0210RP        iSoftwerks Incorporated         BLUGRND   4/02/10
SCROY      634042             Work With Table 10           SC0210C2 09:06:20


Options: 1=Add               2=Edit              4=Delete


Op Cde        Description               Abbr       Chg Usr  Chg date Chg time
                                                             4/02/10
   100 Customer Relations             CS          SCROY       3/27/10  9:20:04
   200 Financial Services             FS          SCROY       3/27/10  9:20:04
   300 Accounting                     AC          SCROY       3/27/10  9:20:04
   400 Inventory                      IN          SCROY       3/27/10  9:20:04
   450 Purchasing                     PO          SCROY       3/27/10  9:20:04
   500 Information Systems            IN          SCROY       3/27/10  9:20:04
   600 Administration                 AD          SCROY       3/27/10  9:20:04
   700 Executive                      EX          SCROY       3/27/10  9:20:04



                                                                          Bottom

         F3=Exit             F5=Refresh

Copyright (c) iSoftwerks, Inc.
```

Fig. 11

These text descriptions and abbreviations may be changed without any impact on soft code functions. Deleting category codes may lead to a problem with any program that has been assigned that specific category.

## Authority Level Entries

Authority levels are defined by this table. The authority level text description and abbreviation may be changed at any time without any impact on the operation of the soft code function editor. However removing code entries or adding entries may affect the functions and/or options that soft coded applications present to the various users.



```
SCROYA1    SC0210RP        iSoftwerks Incorporated         SYSNAME   4/02/10
SCROY      634042             Work With Table 5            SC0210C2 09:07:46


Options: 1=Add               2=Edit              4=Delete


Op Cde        Description              Abbr       Chg Usr  Chg date Chg time
                                                            4/02/10
       No Authority Required          NA         SCROY      3/30/10  7:12:14
   100 Entry Level security           EL         SCROY      3/27/10  9:20:04
   200 Supervisor security            SP         SCROY      3/27/10  9:20:04
   300 Manager Level 1 Security       M1         SCROY      3/27/10  9:20:04
   400 Manager Level 2 Security       M2         SCROY      3/27/10  9:20:04
   500 Operational Security           OP         SCROY      3/27/10  9:20:04
   600 Executive Level Security       EX         SCROY      3/27/10  9:20:04
   700 Adminstrative Security         AD         SCROY      3/27/10  9:20:04
   800 System Level Security          SL         SCROY      3/27/10  9:20:04

                                                                       Bottom

         F3=Exit             F5=Refresh

Copyright (c) iSoftwerks, Inc.
```

Fig. 12

## Soft Code Program Example

The following is an actual program designed with the Soft Code functions in place. The program is a typical 5250 sub-file display program—except it doesn't contain a database file. And, the function keys are not defined in the display. And the program options are contained in a separate database file. And the prompt is a text-driven, dynamically sized window. And the help process is external to the program—other than those oddities it is a typical interactive ILE program.

The program SC0190RP allows a user to view and update database information. It invokes multiple subprograms, some with the options on the panel, some with function keys. It provides help, even though it doesn't appear to have a help function, (the help is actually managed by the function key editor). And, even with the identification block the program totals less than 515 lines of source.

```
  /TITLE ** WORK WITH SOFTCODE USERS **
H DEBUG(*YES)
H OPTION(*SRCSTMT : *NODEBUGIO) DFTACTGRP(*NO) ACTGRP('QILE')
H BNDDIR('SC0000_BD':'QC2LE')
 *******************************************************************
 * Program Name - SC0190RP                                          *
 *                                                                  *
 * Function     - This program was designed to allow a user to      *
 *                manage the softcode application users             *
 *                                                                  *
 * Programmer   - Steve Croy        iSoftwerks, Inc.                *
 *******************************************************************
 *******************************************************************
 *                  Modification log                                *
 *                                                                  *
 *   Date    Programmer      Description                            *
 *                                                                  *
 *******************************************************************
FSC0190DF  CF   E             WORKSTN
F                                     SFILE(SC0190S1:RRNSI)
F                                     INFDS(DSPDS)
 *==============================================================
D FunctionKey   E DS                  EXTNAME(SCKEYSPF) qualified          Function keys
D PGMDS         ESDS                  EXTNAME(SCPSTSPF)                    Pgm status map
D DSPDS         E DS                  EXTNAME(SCDSPFPF)                    Display INFDS
D MACDS         E DS                  EXTNAME(SCFUNCPF) INZ                Key map
D OPTDS         E DS                  EXTNAME(SCOPTNPF) INZ                Option map
D FUNCT         E DS                  EXTNAME(SCMACRPF) INZ                Macro map
D USERPR        E DS                  extname(SCUSRSPF)
D BEFORE        E DS                  extname(SCUSRSPF) prefix(b_)   inz
D AFTER         E DS                  extname(SCUSRSPF) prefix(a_)   inz

D SC0190RP        PR
D  p$find                       10    Const options(*nopass)

D SC0190RP        PI
D  p$find                       10    Const options(*nopass)

 *-- Common prototypes
 /copy qrpglesrc,sc0000_pr
 *---
d indPtr          s               *   inz( %addr(*in) )

 * define named indicators
d indicators      ds            99    based( indPtr )
d  ScreenChange                  n    overlay( indicators : 22 )
d  SflControl                    n    overlay( indicators : 50 )
d  SflDisplay                    n    overlay( indicators : 51 )
```

```
d  SflInitialize                  n   overlay( indicators : 52 )
d  SflClear                       n   overlay( indicators : 53 )
d  SflEnd                         n   overlay( indicators : 54 )
d  SflDelete                      n   overlay( indicators : 55 )
d  SflNxtChange                   n   overlay( indicators : 58 )
d  SflMSGQdisplay...
d                                 n   overlay( indicators : 59 )

D WithParms       PR                  extpgm(SUBPGM)
D  p$user                       10

D WithParms2      PR                  extpgm(SUBPGM)
D  p$user                       10
D  p$mode                        1

D NoParms         PR                  extpgm(SUBPGM)

D Quit            pr
D Exit            pr                  extproc('exit')
D                                3u 0 value

D                 DS
D DEC                            4B 0
D  BIN                           1    OVERLAY(DEC:2)
 * Default cursor position
D #DFPOS          DS                  INZ
D #DFROW                         2  0 INZ(4)
D #DFCOL                         3  0 INZ(61)
 *---------------------------------------------------------------------
 * Define constants
 *---------------------------------------------------------------------
D #MSGF           C                   CONST('SCMSSGF')
D #SAME           C                   CONST('*SAME')
D #TITLE          C                   CONST('Work With Softcode Users')
 *---------------------------------------------------------------------
 * START of work fields
 *---------------------------------------------------------------------
D MessageToDisplay...
D                 S               n
D ADJ             S              2  0
D AUTHL           S              3
D CATEG           S              3S 0 inz(500)
D CMDKEY          S            720
D DTAFLD          S            256
D EOFIND          S              1    INZ('0')
D FKEYDS          S              1
D INDLR           S              1
D M               S              3S 0
D MSG             S             80
D MSGDTA          S            132
D MSGF            S             10
D MSGPGM          S             10
D MSGRLQ          S              5
D O               S              3S 0
D OPTION          S            720
D p$catg          S              3s 0
D p$mode          S              1
D PNLNAM          S             10    INZ('PROMPT')
D prmdata         S            256
D PARM1           S              9  0
D PARM2           S            256
D  FieldName      s             10
D PASSC           S              1
D RCDNBR          S              5  0 INZ(1)
D RRNSI           S              4  0
D SAVRRN          S              4  0
D ScanRequested...
D                 S               n   inz(*ON)
D MoreRecordsRemain...
D                 S               n   inz(*OFF)
D NoMoreRecords...
```

```
D                 S               n   inz(*OFF)
D ErrorOccurred...
D                 S               n   inz(*OFF)
D SFLLOD          S              4  0
D SFLMAX          S              4  0 INZ(12)
D SFLPOS          S              4  0
D SFLRCN          S              4  0
D W$SCAN          S                   LIKE(Z$SCAN)
d P$GATE          s             10
d P$USER          s             10
d P$MOD           s              1
d P$ERR           s              7
d P$RTN           s              7
d   orderBy       s             50a   inz('order by msusrp')
d   selectOnly    s             50a
 *-----------------------------------------------------------------------
 * END of work fields
 *-----------------------------------------------------------------------
 /free
  z$seq1 = #TITLE;
  z$seq1 = CenterTxt(z$seq1:%size(z$seq1));
   //* The security checking program tests whether the user is allowed
  CheckAuthority(USER:PRGNAM:CATEG:PASSC:AUTHL);
  IF PASSC <> 'P';
     QUIT();
  ENDIF;
  FunctionKey = FunctionKeys();
  MSGID = 'MIS0001';
  EXSR @GetMsg;
  IF %parms = 1;
     z$scan = p$find;
  ELSE;
     z$scan = *blanks;
  ENDIF;
  EXSR @reset;

  DOU FUNCT = 'EXIT';
    IF MessageToDisplay;
       SflMSGQdisplay = *ON;
    ELSE;
       SflMSGQdisplay = *OFF;
    ENDIF;
     // write subfile message queue
    WRITE SC0190C2;

   IF ScanRequested;
      #ROW = #DFROW;
      #COL = #DFCOL;
   ENDIF;
   // Display subfile, test for EOJ, and function requested.
   rrnsi = 0;
   sflControl = *ON;
   IF sflrcn > 0;
      sflDisplay = *ON;
   ENDIF;
   WRITE SC019001;
   EXFMT SC0190C1;
   sflControl = *OFF;
   sflDisplay = *OFF;
   DEC = 0;
   EVALR BIN = ROW;
   #ROW = DEC;
   DEC  = 0;
   EVALR BIN = COL;
   #COL = DEC;

   IF MessageToDisplay;
      RmvMessage(prgnam);
      MessageToDisplay = *OFF;
   ENDIF;
   // Get the program function based on the key detected.
      EXSR @EditKeyPressed;
      SELECT;
      WHEN KeyPressed = FunctionKey.ENTER;
         EXSR @ENTER;
      WHEN KeyPressed = FunctionKey.ROLLUP;
         EXSR @LOAD;
      WHEN KeyPressed = FunctionKey.ROLLDN;
         EXSR @DOWN;
      WHEN SUBOP = 'CALL';
         EXSR @CALLS;
      WHEN FUNCT = 'EXIT';
         QUIT();
      WHEN FUNCT = 'RESET';
         EXSR @RESET;
      WHEN FUNCT = 'PROMPT';
         EXSR @PROMPT;
      WHEN FUNCT = 'MORE';
         DisplayKeys(cmdkey: z$key1: z$key2: M);
      WHEN FUNCT = 'MOREOPT';
         DisplayOptions(option: z$opt1: z$opt2: O);
   ENDSL;
   FUNCT = *BLANKS;
   ENDDO;
   //*=================================================================
   //* Process ENTER key
   //*=================================================================
   BEGSR @ENTER;
      IF Z$RRN1 > 0;
         DOU %eof(SC0190DF);
            READC SC0190S1;
            IF NOT %eof(SC0190DF);
               IF Z$OPT <> *BLANK;
                  z$opt = %triml(z$opt);
                  EXSR @EditOptions;
                  IF SUBOP = 'CALL';
                     EXSR @CALLS;
                  ENDIF;
                  z$opt = *BLANK;
                  IF before <> after;
                     SELECT;
                        WHEN subact = 'C';
                           z$stat = '*changed';
                           USERPR = after;
                        WHEN subact = 'D';
                           z$stat = '*deleted';
                           *IN30 = *ON;
                     ENDSL;
                  ENDIF;
                  z$rrn2 =  z$rrn1;
                  UPDATE SC0190S1;
                  *IN30 = *OFF;
               ENDIF;
               funct = *BLANKS;
            ENDIF;
         ENDDO;
      ENDIF;
      EXSR @READ;
   ENDSR;
   //*=================================================================
   //* Read the scan record format to determine if there is a change.
   //*=================================================================
   BEGSR @READ;
      ScanRequested = *OFF;
      READ SC019001;
      IF ScreenChange;                      // Change indicator on
         ScanRequested = *ON;
         EXSR @RESET;
      ENDIF;
      z$scan = *BLANKS;
    ENDSR;
   //*=================================================================
    //* Prompt for a scan value
    //*================================================================
    BEGSR @prompt;
      Prompter(PrgNam: PnlNam: Prmdata);
      IF Prmdata <> *blanks;
         Z$SCAN = PRMDATA;
         ScanRequested = *ON;
         EXSR @RESET;
         Z$SCAN = *BLANKS;
      ENDIF;
   ENDSR;
   //*================================================================
   //* Initialize subfile, and reposition file for subfile load
   //*================================================================
   BEGSR @reset;
      rrnsi = 1;
      rcdnbr = 1;
      sflrcn = 0;
      sflpos = 0;
      SflInitialize = *ON;
      WRITE SC0190C1;
      SflInitialize = *OFF;
      SflEnd = *OFF;
      selectOnly = %trim(z$scan);
      CloseUserCursor();
      ClearUserRec();
      SetUserCursor(orderby: selectOnly);
      EXSR @LOAD;
      EXSR @GetCmdKeys;
      EXSR @GetOptions;
   ENDSR;
    //*================================================================
    //* Subroutine to process rolldown key
    //*================================================================
    BEGSR @DOWN;
       Z$RRN2 = SFLPOS - SFLMAX;
       SFLPOS = SFLPOS - SFLMAX;
       IF SFLPOS < 1;
          SFLPOS = 1;
       ENDIF;
       IF Z$RRN2 < 1;
          Z$RRN2 = 1;
          MSGID = 'MIS0006';
          EXSR @GetMsg;
       ENDIF;
    ENDSR;
    //*================================================================
    //* Set lower limits using search argument, load subfile from I/O
    //* manager until end of file, or max nbr of records loaded
    //*================================================================
    BEGSR @LOAD;
       sfllod = 0;
       savrrn = z$rrn2;
       SflEnd = *OFF;
       NoMoreRecords = *OFF;
       DOU NoMoreRecords or sfllod >= sflmax;
          MoreRecordsRemain = GetNextUser();
          IF MoreRecordsRemain;
             USERPR = GetUserData();
             sflrcn = sflrcn + 1;
             sfllod = sfllod + 1;
             RRNSI = SFLRCN;
             Z$RRN1 =SFLRCN;
             Z$DBRN = 0;
             z$opt = *BLANKS;
             IF msstat = '1';
                z$stat = '*active';
             ELSE;
                z$stat = '*expired';
             ENDIF;
             IF msxdte = 99999999;
```

```
            z$xdte = 999999;
         ELSE;
            z$xdte = CvtToDate6('*MDY':msxdte:'*ISO');
         ENDIF;
         IF msedte = 00000000;
            z$efdt = 0;
         ELSE;
            z$efdt = CvtToDate6('*MDY':msedte:'*ISO');
         ENDIF;
            z$mail = msmail;
         WRITE SC0190S1;
      ELSE ;
         Sflend = *ON;
         NoMoreRecords = *ON;
      ENDIF;
   ENDDO;
   sflpos = sflpos + sflmax;
   IF sfllod > 0;
      z$rrn2 = (sflrcn - sfllod) + 1;
   ELSE;
      z$rrn2 = z$rrn2 + SFLPOS;
   ENDIF;
   IF z$rrn2 > sflrcn;
      z$rrn2 = sflrcn;
      sflpos = sflpos - sflmax;
      msgid = 'MIS0007';
      EXSR @GetMsg;
   ENDIF;
ENDSR;
//*=====================================================================
//* Get message text from message file and turn on message flag
//*=====================================================================
BEGSR @getmsg;
   msgdta = *BLANKS;
   msg = *BLANKS;
   msgf = #MSGF;
   MONITOR;
      RtvMessage(msgid:msgf:msgdta:msg);
   ON-ERROR;
      MessageToDisplay = *ON;
   ENDMON;
   msgtxt = msg;
   EXSR @SendMessage;
ENDSR;
//*=====================================================================
//* Subroutine to send messages to program message queue
//*=====================================================================
BEGSR @SendMessage;
   msgdta =  msgtxt;
   msgpgm = PRGNAM;
   msgrlq = #SAME;
   msgf = #MSGF;
   SndMessage(msgid:msgf:msgdta:msgrlq:msgpgm);
   MessageToDisplay = *ON;
ENDSR;
//*=====================================================================
//* Subroutine to edit command key functions
//* The program name, panel ID and the key are used to retrieve the
//* function macro. If the call fails, default to EXIT.
//*=====================================================================
BEGSR @EditKeyPressed;
  fkeyds = KeyPressed;
  funct = *BLANKS;
  fpgmid = PRGNAM;
  fpnlid = FMTNAM;
  fmacro = *BLANKS;
  MONITOR;
     GetFunction(fpgmid:fpnlid:fkeyds:fkeyid:fmacro:authl);
  ON-ERROR;
     msgid = 'MIS0003';
     EXSR @GetMsg;
   ENDMON;
   FUNCT = FMACRO;
ENDSR;
//*=====================================================================
// * Subroutine to Get the command keys for the application
//*=====================================================================
BEGSR @GetCmdKeys;
   fpgmid = PRGNAM;
   fpnlid = FMTNAM;
   CMDKEY = *BLANKS;
   MONITOR;
      GetKeyText(fpgmid:fpnlid:cmdkey:authl);
   ON-ERROR;
      msgid = 'MIS0005';
      EXSR @GetMsg;
   ENDMON;
   m=0;
   DisplayKeys(cmdkey: z$key1: z$key2: M);
  ENDSR;
//*=====================================================================
//* Subroutine to edit program option functions
//*=====================================================================
BEGSR @EditOptions;
  FUNCT = *BLANKS;
  opgmid = PRGNAM;
  OPTNID = Z$OPT;
  opnlid = FMTNAM;
  OMACRO = *BLANKS;
  MONITOR;
     GetOption(opgmid: opnlid: optnid: omacro: authl);
  ON-ERROR;
     MSGID = 'MIS0002';
     EXSR @GetMsg;
     OMACRO = *BLANKS;
   ENDMON;
   FUNCT = OMACRO;
ENDSR;
//*=====================================================================
//* Subroutine to get program options for the application
// *=====================================================================
BEGSR @GetOptions;
   OPTION = *BLANKS;
   opnlid = FMTNAM;
   MONITOR;
      GetOptText(prgnam: opnlid: option: authl);
   ON-ERROR;
      MSGID = 'MIS0004';
      EXSR @GetMsg;
   ENDMON;
   o = 0;
   DisplayOptions(option: z$opt1: z$opt2: O);
ENDSR;
 //*=================================================================
 //* This subroutine allows program calls using pre-defined  PLISTs
 //*=================================================================
 BEGSR @CALLS;
    EXSR @SETPM;
    MONITOR;
    SELECT;
       WHEN CALLPM = 'PLIST1';
          CALLP WithParms(p$USER);
       WHEN CALLPM = 'PLIST2';
          CALLP WithParms2(p$USER:p$mode);
       OTHER;
          CALLP NoParms();
    ENDSL;
    ON-ERROR;
       P$ERR = 'MIS0012';
    ENDMON;
    EXSR @RETPM;
 ENDSR;
```

```
   //*=================================================================
   //* This subroutine sets values for pre-defined  PLISTs
   //*=================================================================
   BEGSR @SETPM;
      IF UserFound(msusrp);
         before = GetUserData();
      ELSE;
         CLEAR before;
      ENDIF;
      p$user = msusrp;
      p$mode = subact;
      P$RTN = *BLANK;
   ENDSR;
   //*=================================================================
   //* This subroutine determines actions based on returned parms
   //*=================================================================
    BEGSR @retpm;
      IF UserFound(msusrp);
         after = GetUserData();
      ELSE;
         CLEAR after;
      ENDIF;
       IF p$err<>*BLANKS;
          msgid = p$err;
          EXSR @GetMsg;
          MessageToDisplay = *ON;
       ENDIF;
    ENDSR;
 /end-free
P Quit            b
 /free
     ErrorOccurred = CloseUserCursor()      ;
     *inlr = *on                            ;
     exit(0)                                ;
 /end-free
P Quit            e
```