ISPF Editor – Beyond the Basics Hands-On Lab

Liam Doherty dohertl@au1.ibm.com

Peter Van Dyke pvandyke@au1.ibm.com

QR Code for session 12746

QR Code for session 12747

SHARE 120

San Francisco, CA

February, 2013

Contents

Getting Started

You will be logging on to the SHARE z/OS 1.13 system using the ID and password provided.

To login, follow these steps:

- Double click on the “SHARE LPAR” icon on your desktop.
- At the application prompt, type **TSO** and press Enter.
- At the prompt, enter your userid (SHARAxx – you will be allocated a number).
- On the TSO/E Logon Panel, enter your password which is **FIRSTPW**.
- If you are presented with the ISPF main menu screen presented, enter **X** to exit ISPF. If you are already in **READY** mode continue to the next step.

    - If presented with the “Specify Disposition of Log Data Set” screen select process option 2 and press enter.
- At the **READY** prompt, enter one of the following commands to setup your ISPF lab environment:
    - If this is your first time logging on for this lab, enter:
      `ex 'share.rexx(ispfelab)' 'refresh' exec`
    - If this is not your first time logging on for this lab, enter:
      `ex 'share.rexx(ispfelab)' exec`

You will now be back in ISPF with your student datasets created and allocated so that you can run your application from them.

Your student datasets are:

userid.ISPFELAB.REXX
userid.ISPFELAB.PANELS
userid.ISPFELAB.SKELS
userid.ISPFELAB.COBOL
userid.ISPFELAB.SAMPLE

Userid will be equal to your logon userid.

You can use ISPF option 3.4 to display and work with your datasets. Enter **3.4** on the command line. At the “Dsname” prompt, enter ***userid*.ISPFELAB** and press Enter and all of your lab datasets will be displayed.

The ISPF Editor Lab

This ISPF Editor lab is split into 3 sessions. You can start at the beginning and work through or skip over things that you are already familiar with. The 3 main sections that are covered are:

1. The Edit environment and Adding power to edit
2. Edit Models
3. Edit Macros

For the first section, we will use a member in your COBOL library to demonstrate the various aspects of the ISPF editor. In the next sections we will create members in your other ISPF libraries to demonstrate ISPF Models and Edit macros.

If you need more information on the ISPF editor, here is a link to the two manuals that may be helpful:

ISPF Edit and Edit Macros:
http://publibz.boulder.ibm.com/cgi-bin/bookmgr_OS390/BOOKS/ISPZEM90

ISPF Reference Summary:
http://publibz.boulder.ibm.com/cgi-bin/bookmgr_OS390/BOOKS/ISPZRS90

The Lab Exercises

General tips for completing the exercises:

- When editing an object, you can use highlighting to help with formatting. On the command line enter one of:
  - HILITE REXX
  - HILITE COBOL
  - HILITE PANEL
  - HILITE SKEL
- The various commands described in the Lab might be in upper, lower or mixed case. On z/OS case is not important. Commands have been described in the lab in upper case to make them stand out. You can enter them on the screens in ISPF in any case.
- When done making a change, save your work either by entering **SAVE** on the command line or using F3 which exits the editor and saves your work.

Lab 1 - The ISPF editor and adding power to edit

We will be doing most of our work in a single member that already exists in your own COBOL data set. Go to ISPF Edit (option 2 from the main menu) and enter '***userid*.ISPFELAB.COBOL'** in the name field and press enter to bring up a member list consisting of a single member, IGYTSALE.

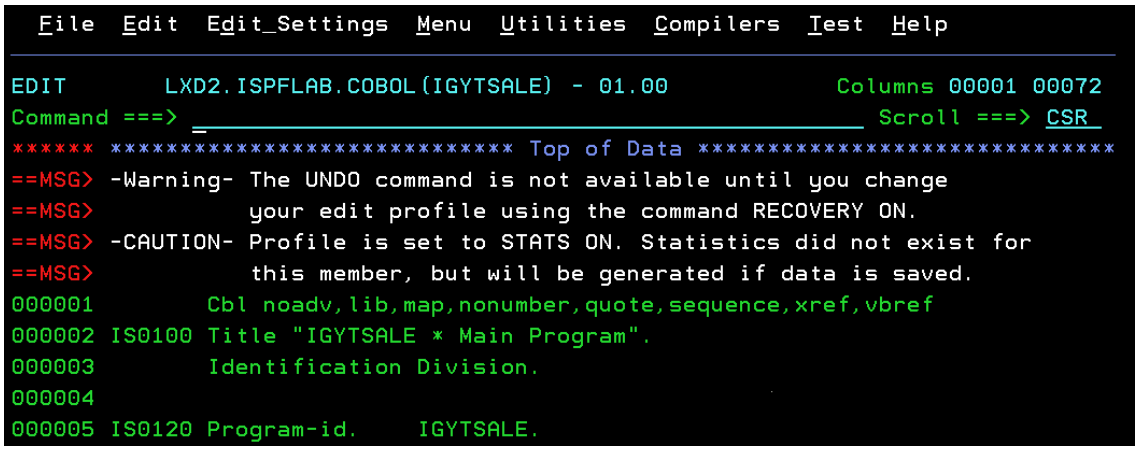
Optionally, go to the Data set List utility (Option 3.4 from the main menu) and enter the data set name of '***userid*.ISPFELAB.COBOL'** in the Dsname Level field and press enter, then enter an E next to the data set and press enter again.

To start working on a member, enter an **E** or an **S** next to the member and press enter. You are now ready to begin.

Exercise 1 – Edit Recovery/UNDO

In this exercise you will become familiar with the Edit settings for recovery and Undo.

- As you come into the member the first thing that you will see is the following message in the message area at the top of the member:



- The same setting "RECOVERY ON" handles edit session recovery and the UNDO feature. The former is a feature such that if you are in the middle of an edit session and your machine crashes, when you come back into edit you will be provided with the option to recover back to the point of failure. This is difficult to demo in a lab.
- The more useful feature however is the UNDO feature. This allows you to undo any interactions up to the last save point.
- On the command line type **RECOVERY ON** and press enter. The warning message should now have disappeared.
- Make a number of changes in the file. For example.
  - Overtype **A. Programmer** with your name. Press **Enter**.

    - On the command line type **Change IBM SHARE all** and press **Enter**.
- Now on the command line type **UNDO** and press enter. You should see that the last change you made has been reversed. If you type **UNDO** again the change before that is reversed. Keep undoing until you get the message "No more to UNDO".

Exercise 2 – Edit Settings

In this exercise you will become familiar with the Edit settings dialog.

- There are a number of settings that you may wish to change from the default; these are stored in your ISPF profile data set.
- Before you start, make sure you are at the top of the member.

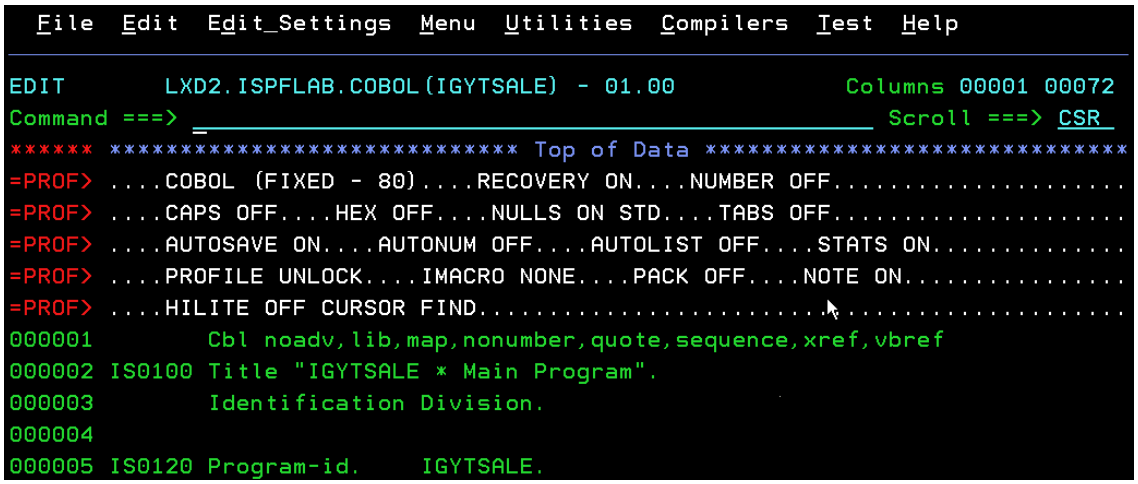
  Hint: *Type an **M** on the command line and press function key **F7** to return to the top. Alternatively type **TOP** on the command line and press **enter**.*

  On the command line type **F NONE** and you will see that the found line is positioned at the 2nd line on the screen.
- On the command line enter the **EDSET** command and press enter, or click on the **Edit_Settings** menu bar option, then select **Edit Settings**.
- The main options to look at are :
    - **Target line for Find/Change/Exclude string** – This determines how many lines are displayed before the find string. Previously you will have seen the found line was positioned at line 2. Change this value to 5 and press **F3** to go back to the member. Go to the top of the member again, and then type F NONE again. You will see now that the found line is positioned at the 5th line on the screen.
    - **Always position Find/Change/Exclude string to target line** – This will position the cursor on the found line, rather than leaving it on the command line. Change this if you wish by entering a / in the field.
    - **Cut default** and **Paste default** – This determines the cut and paste behaviour. We will talk about Cut and Paste in more detail in a later lab. Just familiarize yourself with the settings.

Exercise 3 – Edit profiles

In this exercise you will become familiar with the Edit profile.

- ISPF stores information about the type of member you are editing in an Edit profile. It determines default settings based on the data set LLQ (Low level qualifier) or other methods
- On the command line type **PROFILE** and press enter. You should see something similar to this:



- You will see a number of different options. For example it knows that this is a COBOL type.
- On the command line type HILITE COBOL or HILITE ON and you will see the option change in the profile area. Plus the program will now be colour coded for ease of reading
- Press PF3 to leave the member. Then come back in again by selecting the member. You will see that the edit profile has been saved and used again on entry.
- On the command line, enter the **PROFILE 9** command and press Enter. This will display all of the profile information including the =TABS>, =MASK>, =BNDS> and =COLS> all of which will be covered later.
- To turn off the profile information, or to generally "reset" things in your ISPF edit session type **RESET** or **RES** on the command line and press **Enter**.

Exercise 4 – Edit Line Commands

In this exercise you will become familiar with the various Edit line commands.

All line commands are entered in what is called the prefix area in the editor screen. This is the area to the left of the display with the 6 digit line number in it. You can overtype anywhere within this area with the line command you want.

Exercise 4.1 – Bounds/BNDS command

This is the first line command we will look at as many of the other line and primary commands are affected by it.

- On the first line in the member, in the prefix area, enter the **BNDS** line command and press **Enter**. You will see a line appear which marks the bounds of the line by a **<** and **>** markers. These are in columns 1 and 80.
- All operations, such as many of the line commands we will try, and primary commands, such as Find, only work within the BNDS.
- Change the bounds by overtyping the existing markers with blanks and entering new < and > markers somewhere else in the line, for example, columns 10 and 20. Use the **COLS** line command see where the columns are by entering COLS in the prefix area of a line.

- If you have set the bounds in columns 10 and 20, now try finding the string "**Ident**" by typing **F IDENT ALL** on the command line. You will see a message saying that it cannot find that word. Even though you know it is there.
- You can also set the bounds by using the primary command **BNDS 10 20** entered on the command line at the top of the screen
- Reset the bounds now by blanking out the < and > markers you set.

HINT: Whenever you are getting strange inexplicable results try turning BNDS on and see if you have the bounds set to something you are not expecting.

Exercise 4.2 – LC and UC commands

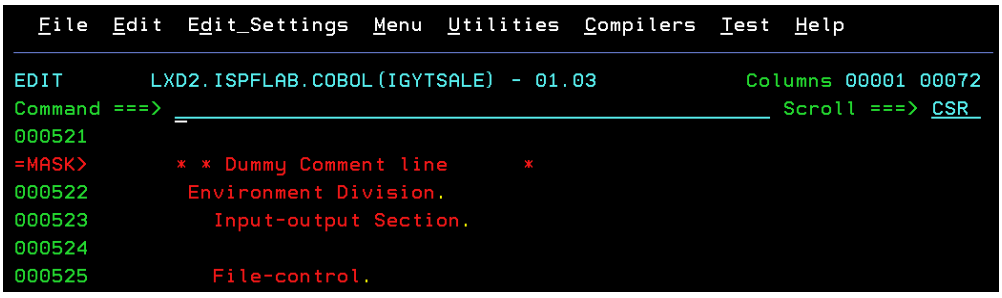
These line commands are the LowerCase and UpperCase line commands

- Go to the top of the member by placing an **M** on the command line and pressing **F7**.
- On the command line type **F INTRO** to find and positions the screen on the comment block in the code
- On the line that says "This is a hypothetical…", in the prefix area, type **UC** and press **Enter**. You will see that the line has been changed to uppercase.
- Now enter **LC** on the line and press **Enter** and you will see that the line changes to lower case.
- Line commands can be entered with a number, to indicate a number of lines or as a block command. On the same line, in the prefix area, enter **UC4**. You will see that the block of text moves to upper case
- On the first line in the block, in the prefix area enter **LCC** and on the 4th line in the comment block also enter **LCC**, then press enter. You will see the block reverts to lower case.
- LC and UC commands work with the bounds of the line, normally columns 1-80, but this can be changed with the BNDS command.

Exercise 4.3 – Mask command

The mask command can be used to specify characters that are entered automatically by the editor when the I (Insert), TS (Text Split) or TE (Text Entry) line commands are used, or when a new member is created. The mask is saved in the edit profile.

- On the command line type **L 522** to locate line 522, the Environment division.
- In the prefix area of line 522 type the word MASK and press enter. You will be presented with a mask line where you can enter your string.
- In the mask area enter a comment string starting in column 7 that will be used as a default mask on subsequent inserted lines



- Press **Enter**.
- Now on line 523 enter a line command of **I3** to Insert 3 lines. You will see that all the inserted lines now contain the mask data.

- To make a line remain you need to type something in the inserted line.
- Let's get rid of the mask data, by blanking out the string in the **=MASK>** line.

Exercise 4.4 – MD (Makedata) Command

One of the things you can do in ISPF edit is make the various notes and message lines actual data lines in your code. The most useful time for this is when using Edit Model (covered later in more detail) as the models often provide sample code that you can use.

- In your test program go to the bottom of the code. Do this by entering an **M** on the command line and pressing **F8** (Max down). Alternatively type **BOTTOM** on the command line and press **enter**.
- We are going to use one of the ISPF models to demonstrate this. Don't worry too much for now about what Models do that is covered later. On the command line type **MOD DISPLAY** (this is the model command for a display ISPF panel). Before pressing enter, tab your cursor down to the last line of code and in the prefix area enter the letter **A** (After). Press **Enter**.
- If you scroll down through the inserted code you will see a number of **=NOTE=** lines. There are some examples in the **=NOTE=** lines also.
- On one of the example blocks, overtype the first **=NOTE=** line with **MDD**, and the last **=NOTE=** line in the block with **MDD**, and press Enter. This is a block MD command.
- You will see that the lines now become part of the code.
- On the command line type **RES** (Reset) and press **Enter**. You will see that all the other **=NOTE=** lines disappear.
- The MD can be used on =NOTE=, ==MSG>, =COLS> or ====== lines. The later being useful when using Edit COMPARE (covered later).

Exercise 4.5 – Shifting data

There are 2 different ways to shift data in ISPF using different types of brackets. These are called Column Shift and Data Shift.

- Column shift will move data the required number of characters, but will drop characters off the end of the "bounds" of the member.
- On any line, in the prefix area, type **)50**. You will see that the data in the line has moved to the right 50 characters
- On the same line, in the prefix area, now type **(50** to bring the data back. However you will notice that you have lost the data that went past the bounds of the line (column 80 in this case).
- On a different line, in the prefix area, type **>50**. Now if the data on the line is unable to shift that far you will see an error message "Data shifting incomplete" and the data is only moved as far as is possible.
- Data shifting works with the bounds of the line, normally columns 1-80, but this can be changed with the BNDS command.

Note: In COBOL programs any data in the COBOL prefix area, columns 1-7 is not shifted.

Exercise 4.6 – Text Split (TS) and Text Flow (TF)

These are useful command that split data at the cursor position or join lines back together.

- On the command line type **F IS0875** to position the screen at line **461**.
- We are going to split the line so that each parameter is on a different line.
- In the prefix area of line 461, the line that contains **Call "CEEDATM"** enter the **TS** line command, do not press enter yet.
- Move your cursor so that it is under the **p** of **pict-hdr-v** and press **Enter**. You will see that the **pict-hdr-v**, string has now moved to the line below in column 1. This is because it tries to position the split line in the same position as data in the line above or the line below.
- You can now shift the data across by entering **)36** in the prefix area of the split line and pressing **Enter**.
- Next on line 463, the line that contains **hdr-out-date**, shift the data right to line it up under the previous line. Use an **)19** in the prefix area.
- In the prefix area of line 463, the line that contains **hdr-out-date** enter the **TS** line command, do not press enter yet.
- Move your cursor so that it is under the **f** of **fc** and press **Enter**. You will see that the **fc** string has now moved to the line below. Let's now move that so it lines up by placing a **)25** in the prefix area of line 464 and press **Enter**.

The line should now look like this:

```
000461 IS0875          Call "CEEDATM" Using loct-seconds,
000462                                      pict-hdr-v,
000463                                      hdr-out-date,
000464                                      fc
000465            Else
```

- Of course we may not be happy with this, so we can use the TF (Text Flow) line command to rejoin lines of data. You can use the TF command by itself, in which case it is affected by the bounds of the line, or you can enter a column number so that it does not have any data after a certain column.
- In our example, on line 461 enter, in the prefix area, the line command **TF72**. This will rejoin the lines and allow data to flow up to column 72. The rejoined line will look like this:

```
000461 IS0875          Call "CEEDATM" Using loct-seconds, pict-hdr-v,
000462                                      hdr-out-date, fc
000463            Else
```

Exercise 4.7 – Labels

Labels can be used to place ranges on the data in the member. ISPF has a number of pre defined labels, namely .ZCSR , .ZFIRST and .ZLAST.

- Go to the top of the program by typing an **M** in the command line and pressing **F7**.
- Let us go down to the Procedure Division. Type **F Procedure** on the command line and press **Enter**. This will take you down to line 429.
- In the prefix area if line 429 enter **.A** and press **Enter**. Labels have to begin with a period. They don't have to be meaningful names as they are not remembered between sessions.

- We can now perform some other actions, such as Find or Change and limit the range so it is between the labels. On the command line type **F CALL .A .ZLAST** and press **Enter**. This will find all occurrences of the word CALL between the label we created **.A** and the last line of the member.
- Let's go back to the Procedure Division. Type **L .A** (Locate) on the command line and press **Enter**. This will position us directly at the label we defined.
- You can try defining another label and try and do a find specifying just your own label names.
- To remove the labels either just blank them out in the prefix area or type **RESET LABELS** on the command line and press **Enter**.

Note: Labels are used by Find, Change, Exclude, Delete, Locate, Replace, Reset, Sort and Submit

Exercise 4.8 – Tabs

Tabs can be used to allow you to use the tab key or Enter key to jump to pre-defined columns in the edit area. They can be useful in COBOL programs to jump to Area A and Area B. TAB settings are saved in the edit profile for the type of data you are editing.

There are 3 different types of TABs that behave differently. Let's explore them all.

Software tabs

- Go to the top of the program by typing an **M** in the command line and pressing **F7**.
- Let us go down to the Procedure Division. Type **F procedure** on the command line and press **Enter**. This will take you down to line 429.
- On line 429 enter the line command **TABS** in the prefix area and press **Enter**.
- Enter a hyphen (-) or underscore in column 8 and another one in column 12. This defines the start columns for Area A and Area B. Press **Enter**.
- Insert a new line with an I in the prefix area of any line and press **Enter**. A new line is inserted and the cursor is immediately positioned in column 8, the first tab.
- Press the **Enter** key and your cursor will then jump to column 12. So you can type data in which area you wish.
- These are software tabs so you are not constrained by the tab markers and can start in column 8 and just type through column 12. So Software tabs are perfect for this type of use.
- Blank out the tabs in the **=TABS>** line

Hardware tabs

- Hardware tabs can be used to limit field size. So we will use them to create a declaration block in our Data Division.
- Go to the top of the program by typing an **M** in the command line and pressing **F7**.
- Let us go down to the Data Division. Type **F "Data Division"** on the command line and press **Enter**. This will take you down to line 186.
- On line 193 (01 start-of-working-storage) enter the line command **TABS** in the prefix area and press **Enter**.
- Enter a * in columns 9, 12 and 43 so that the **=TABS>** line looks like this:

```
=COLS> ----+----1----+----2----+----3----+----4----+----5----+----6--
=TABS>         *  *                              *
000193          01 start-of-working-storage.
000194            05 program-name                 Pic x(30)   Value
000195              "IGYTSALE                     ".
```

- You now need to activate the tabs. Go to the command line and enter **TABS** and press **Enter**. Tabs are now activated.
- On the **=TABS>** line (or anywhere as TABS are in effect everywhere) enter the **I** line command in the prefix area (overtype the =TABS> if you wish) to insert a new line, and press **Enter**. The cursor is immediately positioned on the column after the first tab position, column 10. You can enter the number 77 here and the cursor will jump to the next tab position. In column 13 where you can enter the variable name. Enter Variable1 and press the Tab key. You are now positioned on the final tab position, column 44. Enter a data format of PIC X(9). and press **Enter**.
- You will be taken to the next line where you can enter your next declaration.

Logical tabs

- Used by the editor to reposition strings of data.
- Leave your tab positions defined with the * in columns 9, 12 and 43.
- On the command line issue the **TAB #** command and press **Enter**. This will set the # character as the tab skip character.
- Insert a line and in the position where the cursor is enter **#1#2#3** and press **Enter**. The Editor will reposition the strings that follow the tab character # in the next available column. So you should see this on the insert line:

```
=COLS> ----+----1----+----2----+----3----+----4----+----5----+----6--
=TABS>         *  *                              *
000193          77 Variable1                     Pic x(9).
000194          1  2                             3
''''''
000195          01 start-of-working-storage.
```

- To remove Tabs just blank them out in the **=TABS>** line.

Exercise 4.9 – Hexadecimal

Sometimes it is necessary to edit hex data. For example if you are changing attribute characters in an ISPF Panel. Another common use would be to add tab characters to a data file so that you can download it to your PC and import it into a spreadsheet program such as Microsoft Excel. The tabs characters will be honoured by the import. There are a number of ways to work with hexadecimal data.

Changing hex data via the change command

- You test program does not have any non-displayable hex characters. However we can still change normal alpha characters using their hex representation.
- Go to the top of your program. You can enter **Top** on the command line and press enter.
- On the command line enter **c x'98' x'd8' all .zfirst .zfirst** and press enter.

- You will see that the lower-case letter **q** has been changed to an upper-case letter **Q**.

Displaying the complete program in HEX mode

- You may want to display a complete data set of member in HEX mode to manually change characters.
- On the command line type **HEX** and press enter.
- You will see that each line has now become a 3 lines. The character is on the first line and the hex representation is shown vertically below it on the next 2 lines:

```
****** ***************************** Top of Data ********************************
000001        Cbl noadv,lib,map,nonumber,quote,sequence,xref,vbref
       4444444C8949988A69886989699A988969A9A86A89A89886A9886A89884444444444444
       0000000323056145B392B417B56544259B84635B25845535B7956B529560000000000000
------------------------------------------------------------------------------
000002 IS0100 Title "IGYTSALE * Main Program".
       CEFFFF4E8A9847CCEEECDC454D8894D9989897444444444444444444444444444444444
       9201000393350F978321350C0419507967914FB000000000000000000000000000000000
------------------------------------------------------------------------------
```

- You can overtype the hex representation to change the character. For example on line 2 change the **5C** to **6F** by overtyping the **5** with a **6** then using your cursor tab down and change the **C** to an **F**. Press enter and the character will change to a **?**.
- To turn hex mode off go to the command line and type **HEX OFF** and press enter.

Displaying a single line in HEX mode

- In ISPF for z/OS 1.11 it became possible to turn hex on just a single line. This is particularly useful if you want to see the program in context as turning HEX on for the complete program makes the display very confusing.
- Go to the top of your program and in the prefix area on line 1 type **HX** and press enter.
- You will see now that only the first line is in HEX mode, and the rest are displayed normally.

```
****** ***************************** Top of Data ********************************
000001        Cbl noadv,lib,map,nonumber,quote,sequence,xref,vbref
       4444444C8949988A69886989699A988969A9A86A89A89886A9886A89884444444444444
       0000000323056145B392B417B56544259B84635B25845535B7956B529560000000000000
------------------------------------------------------------------------------
000002 IS0100 Title "IGYTSALE * Main Program".
000003        Identification Division.
000004
000005 IS0120 Program-id.    IGYTSALE.
000006
000007        Author.        A. Programmer.
000008        Installation.  IBM - Santa Teresa Laboratory.
```

- Like many other line commands it is possible to turn HEX on for multiple lines. So you can use the **HX2** command in the prefix area to turn HEX on for 2 lines. Also you can use block commands to turn hex on for a block of code by entering **HXX** on the first line and **HXX** on the last line of the block.
- To turn HEX off just go to the command line and type **HEX OFF** and press enter.

Exercise 5 – Edit Primary Commands

In this exercise you will become familiar with the Edit primary commands.

Exercise 5.1 – COLS

We looked at the COLS line command in use through out the examples above. There is also a COLS primary command that does not move around as the line COLS does.

- On the command line enter **COLS** and press **Enter**. A Cols column will appear at the top of the screen and stay there.
- Enter the COLS command again to turn it off

Exercise 5.2 – Finding, changing and excluding data

There are different ways to find data in a file, there are simple strings, delimited strings, character strings, picture strings and hexadecimal strings. Let's try a few of those.

- Go to the top of the program by typing an **M** in the command line and pressing **F7**.
- On the command line type **f program all** and press **Enter**. This is a simple string and finds all words program regardless of case or if it is part of another word.
- On the command line type **f 'program ' all** and press **Enter**. This is a delimited string, so we are looking for words where program is followed by a blank.
- On the command like type **f C'program'** and press **Enter**. This is a character string and finds the next occurrence of the word **program** where the full word is in lower case.
- On the command line type **f x'F1F9F9F1'** and press **Enter**. This is a hexadecimal string so will find a delimited string of hex characters, in this case decimal 1991.
- On the command line type **f program all word** and press **Enter**. This will find all occurrences of the full word **program**. So it will not find it if it is part of another word, like programmer.
- On the command line type **f '80' all suf** and press **Enter**. This will find all occurrences where the number 80 appear as a suffix to a word.

  Hint: It is a good idea when finding numerics to put them in delimiters to avoid ISPF getting the find string confused with the columns.

- On the command line type **f p' ----- ###' all** and press **Enter**. This uses the picture strings and will find all strings that start with a blank, have 5 non-numeric characters, another blank and then 4 numeric characters. In this program it only finds 1 occurrence. Have a play around with the other Picture strings:
  - ***P'='*** – Any character
  - ***P'¬'*** – Any non-blank character
  - ***P'.'*** – Any non-displayable character
  - ***P'#'*** – Any numeric character, 0-9
  - ***P'-'*** – Any non-numeric character
  - ***P'@'*** – Any alphabetic character, upper or lower case
  - ***P'<'*** – Any lowercase alphabetic character
  - ***P'>'*** – Any uppercase alphabetic character
  - ***P'$'*** – Any special character
- On the command line type **X ALL** and press **Enter**. This will exclude all of the lines in the program. We can now do some of our find strings again but only see the lines where they are found.

- You can also use a combination of other options such as labels, columns and the X (Excluded lines) or NX (Not excluded lines).
- On the command line type **RES** and press **Enter**.
- Type **F Procedure** on the command line and press **Enter**. This will take you down to line 429.
- In the prefix area if line 429 enter **.A** and press **Enter** to assign a label.
- On the command line type **X ALL .ZFIRST .A** and press **Enter**. This will exclude all the lines from the display from the first to the label we just added.
- On the command line enter **f procedure all word .a .zlast 8 17 nx** and press **Enter**. This will finds all occurrences of the full word procedure between columns 8 and 17 and between labels .a and .zlast but only in non-excluded lines.

Exercise 5.5 – Hide and Flip

You will have noticed in the previous exercise that when you exclude a line there is a marker line left in its place. These can take up a lot of real estate on a screen for example if you are checking if your Ends and Dos line up in a program.

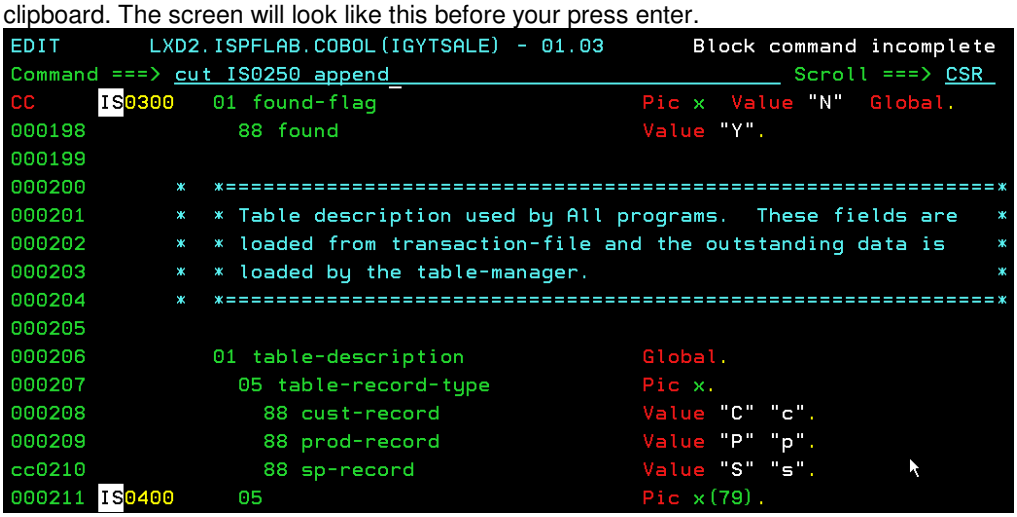
- On the command line type **X ALL** and press **Enter**. This will exclude all of the lines in the program.
- On the command line type **f program all** and press **Enter**. You will see all the found lines and the exclude markers telling you how many lines are excluded.
- On the command line type HIDE X and press Enter. You will now see that those marker lines are hidden and replaced with an underline in the prefix area.
- Now, on the command line type **FLIP** and press **Enter**. This command will flip the excluded lines for the non excluded lines.
- On the command line type **RESET HIDE** and press **Enter**. This will revert to the normal excluded line view with the excluded line indicators.

Exercise 5.3 – Cut and Paste

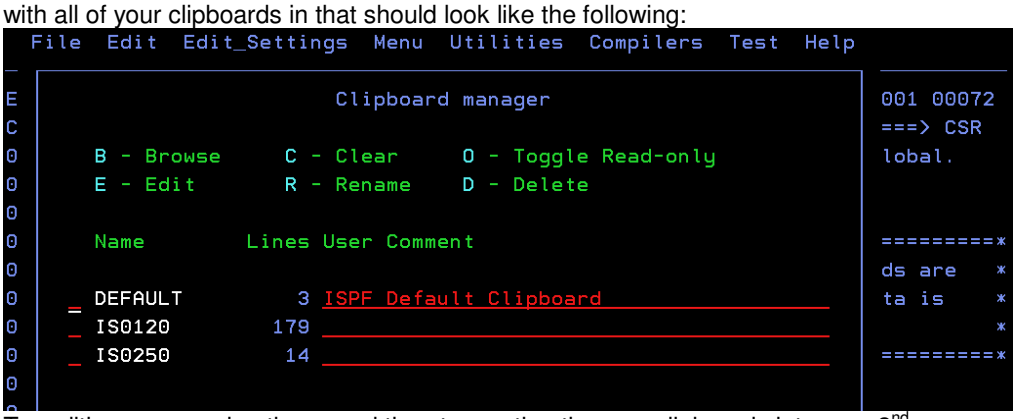
ISPF provides a cut and paste facility with multiple clipboards.

- Go to the top of the program by typing an **M** in the command line and pressing **F7**.
- On line 2, the line containing the IS0100 line, enter **C3** in the prefix area as we are going to cut 3 lines. On the command line, enter **CUT** and press **Enter**. You will get a message **"3 lines cut to DEFAULT".**
- Move your cursor to the top of the screen and press **F2** to split the screen.
- Go into ISPF Edit in the 2nd screen and edit '***userid*.ISPFELAB.COBOL'** entering a new member. Call this what you like.
- On the command line of the new member type **PASTE** and press **Enter**. The cut data is now pasted into the new member.
- Swap screens using F9 to go back to the 1st Edit session.
- On line 5, the line containing the IS0120 line, enter **C179** in the prefix area as we are going to cut 179 lines. On the command line, enter **CUT IS0120** and press **Enter**. You will get a message **"179 lines cut to IS0120".**
- Next, type **F IS0250** on the command line. In the prefix area next to the line containing IS0250 (should be 187) enter **C10**. Then on the command line type **CUT IS0250** and press **Enter**.

- Swap screens back to your 2nd edit session and place an A in the prefix area of the last line. On the command line type **PASTE IS0120** and press **Enter**. The clipboard called IS0120 will be copied after the last line in the member.
- You will remember the edit settings exercise earlier where you set up your defaults for cut and past. The default for cut is replace. But if you want to append some data to one of the clipboards you can do that by using the Append keyword. You can try this is you wish.
- In the 1st edit session do a **CC** block command around the IS0300 block of code and type **CUT IS0250 APPEND** and press **Enter** to append the data to the IS0250 clipboard. The screen will look like this before your press enter.



- You will get the message “14 lines added to IS0250”.
- You can work with your clipboards, such as edit them, rename them or delete them. On the command line type **CUT DISPLAY** and press **Enter**. You will see a display with all of your clipboards in that should look like the following:



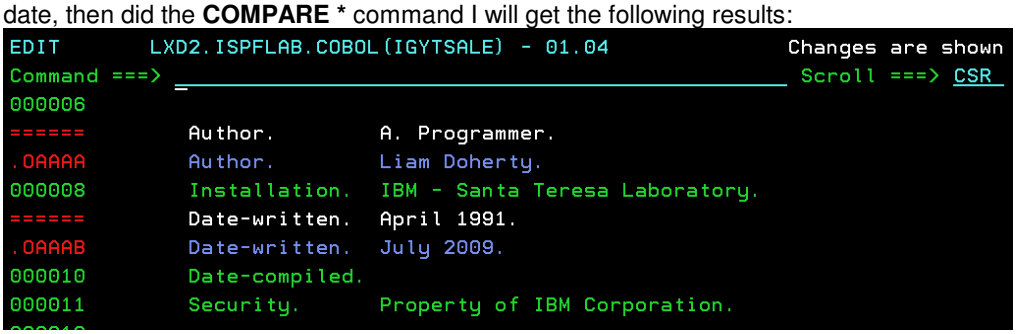
- Try editing or renaming them, and then try pasting the new clipboards into your 2nd edit session.

Exercise 5.4 – Edit Compare

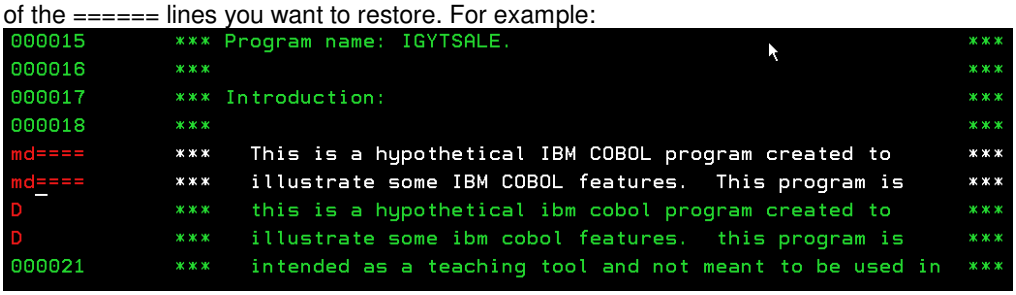
Edit has a compare command that is useful in seeing changes you have made in your current edit session or changes between different files.

- On the command line type **COMPARE *** and press **Enter**. This will show you the changes you have made in this current Edit session since you last saved the member.
- If you get the message “Files are the same” make some random changes. Then do the COMPARE * command again to see the changes. The previous version of the line is marked with a **=====** and the new changed line is identified with **.OAAAA**. If there are subsequent changes they are marked .OAAAB and so on.

- For example if I changed the Programmer name to my name and the Date-written date, then did the **COMPARE *** command I will get the following results:



- If you have made some changes save the member.
- You can also compare against other external data sets. On the command line type **COMPARE 'IGY.V3R2M0.SIGYSAMP(IGYTSALE)'** and press **Enter**.
- This will show any differences you have made to the original member we copied. You can scroll down and look at the changes.
- If you wish to revert any changes back to the original, put a D in the prefix area of the lines that begin .0AAA (You also need to blank the rest of the prefix area out). Now you can use the MD (Make Data) line command we looked at earlier in the prefix area of the ====== lines you want to restore. For example:



- You may also only be interested in seeing the changes and a few lines around the change. On the command line type **COMPARE 'IGY.V3R2M0.SIGYSAMP(IGYTSALE)' X** and press **Enter**. You will see 5 lines around the changed lines shown.

Lab 2 – Edit Models

In this Lab we will be looking at Edit Models, what they are used for and how to create our own edit models.

Edit models are predefined templates that can be included in the member being edited. ISPF ships Edit models for all of its APIs to make it easier to include the required code in the language that you are working in. For example, REXX, COBOL or PLI.

The model has 2 parts, the actual data that will be included in the member and notes line that provide instructions or other information such as example code.

Exercise 1

In this exercise learn how to use the provided ISPF models. We will work with some REXX members to show how models can be included.

Go to ISPF Edit (option 2 from the main menu) and enter **'*userid*.ISPFELAB.REXX(SHARE120)'** in the name field to begin editing the member SHARE120.

- ISPF works out the correct type of model type, or class, based on the low level qualifier (LLQ) of the data set. However it is possible to chose a class, and switch the class if you for example store a COBOL member in a data set that has a LLQ of SOURCE.
- On the command line type **MOD CLASS** and press **Enter**. You will see a list of available classes that contain models.
- Select one of them by entering the number in the command line, for example SKELS.
- You will be returned to the edit session with the message **"Model class changed".**
- Now if you type MOD on the command line and press enter you will be presented with a list of ISPF Skeleton models. You can select one and the skeleton code will be copied into your REXX member. For example ITERATE.
- As we are in a REXX member lets switch the class to REXX. If you know the name of the class you can just type **MOD CLASS REXX** on the command line and press **Enter**. The class will have been changed to REXX.
- Delete the already modelled lines so that the member is empty again.
- Now when you type **MOD** on the command line and press **Enter** you will be presented with a list of REXX models.
- Select **D1** to select the model for the **ISPF DISPLAY** service and press **Enter**. You will see that the format of the service API is displayed, along with notes that describe all of the options. If you scroll down you will see there is some example code and a return code block.
- If you want to use the example code you can change the =NOTE= lines to MD by overtyping them thus turning them into data lines.

- By typing RES on the command line the rest of the =NOTE= lines will disappear.
- You can now modify the code as you like.
- If you know the name of the model you are interested in you can enter it directly from the edit session. Place an **A** in the prefix are of the line where you want the model code to be inserted.
- On the command line type **MOD LMINIT** and press **Enter**. You will see the different types of format the LMINIT API can take plus =NOTE= lines explaining everything.
- Play around with different models, don't worry about where you insert them as we are not going to be running the REXX code.

Exercise 2

One of the things many development sites do not realize is the fact that you can easily create your own models. This can be useful to provide developers with sample code, JCL procs, etc, that they can use. They don't have to be ISPF related models. For example there are already models provided to SCLM Architecture definition members.

In this exercise you will create your own model and make it available in the list of available models.

The steps we need to perform are:

1. Create the model source and save it in an ISPF skeleton library that is allocated to the ISPSLIB DD.
2. Make the model accessible from a model selection panel
3. Make your model selection panel available as a class in the model class list

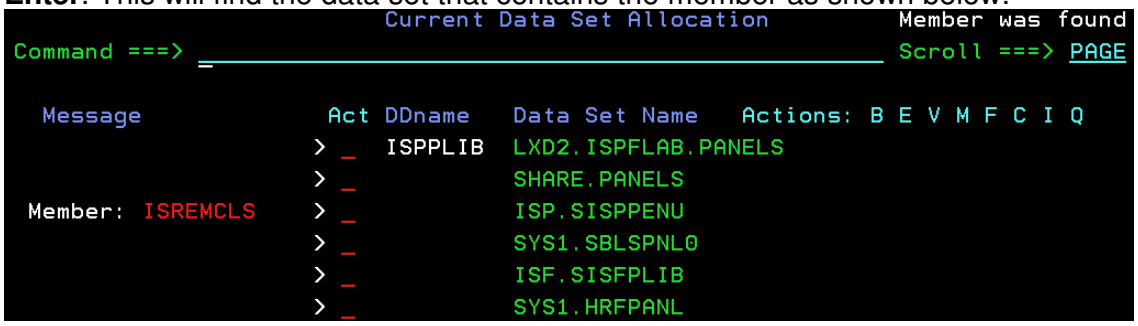
So let's get started!

- Go to ISPF Edit (option 2 from the main menu) and enter '***userid*.ISPFELAB.SKELS(IEFBR14)'** in the name field to begin editing the member IEFBR14.
- If the member is empty, on the command line type **copy 'SHARE.ISPFELAB.SKELS(IEFBR14)'** to copy in a sample JCL. (The member may already exist from a previous running of the lab).
- We will now add some model comments that will appear as NOTE lines when the model is displayed.
- Insert a line at the top of the member and insert the following data on the inserted line :
)N THIS IS A SAMPLE IEFBR14 JCL
- Save the member

- Go to ISPF Edit (option 2 from the main menu) and enter '***userid*.ISPFELAB.PANELS'** in the name field and enter a member name of **ISREMCLS** to begin editing the member. The member name is important as

we are going to allocate our version of this member higher in the ISPPLIB concatenation than the system one.

- We now need to go and find the version of this panel that the system is using. On the command line type **DDLIST** and press **Enter**. This will start ISRDDN.
- On the command line in ISRDDN type **M ISREMCLS ISPPLIB** and press **Enter**. This will find the data set that contains the member as shown below:



- Make a note where the member comes from, in this case 'ISP.SISPPENU(ISREMCLS)'
- Press F3 to come out of View on the member and then F3 to come out of ISRDDN so that you are back in your empty member. On the command line type **COPY 'ISP.SISPPENU(ISREMCLS)'** and press **Enter**.
- On line 33 (the line with 14 ARCHDEF) repeat this line with the **R** line command and over type to **15 PERSONAL – My personal templates** such that it looks like this:

```
003200  13 SCLM     - SCLM Project Definition Macros
003300  14 ARCHDEF  - SCLM Architecture Definition templates
003310  15 PERSONAL - My personal templates
003400 )INIT
```

- Change line 46 to add **15,USER** between 14,ARCHDEF and *,* such that the line looks like this:

```
004200 IF (&ISRMONCL = 'Y')                /* ARE WE ONLY OBTAINING A CLASS ? */
004300     &ISRMCLAS = TRANS(TRUNC(&ZCMD,'.')  1,CLIST 2,COB 3,EXEC 4,FORT
004400        5,MSGS 6,PANELS 7,PLI 8,SKELS 9,PASCAL 10,REXX 11,DTL 12,C
004500        13,SCLM 14,ARCHDEF 15,USER *,*)
```

- At line 58 we need to add the new option to the TRANS. You may need to modify the line to add USER,15 after ARCHDEF,14. You may need to drop some of the existing data to the next line such that it looks like the following:

```
005400 &ZCMD = TRANS (&ZCMD
005500 CLIST,1 COB,2 EXEC,3 FORTRAN,4 FTN,4A FORT,4B MSGS,5 ISPMLIB,5A PANELS,6
005600 MENUS,6A ISPPLIB,6B PLI,7 PL1,7A PLIOPT,7B PLS,7C SKELS,8 PROCS,8A ISPSL
005700 PASCAL,9 REXX,10 DTL,11 C,12 CPP,1C SCLM,13 ARCHDEF,14 USER,15
005710 ' ',' ' *,*)
```

- At line 84 (the line with **14,'PANEL(ISREMARC)'**) insert a new line after this line and enter **15,'PANEL(USERMODL)'** such that it looks like this:

```
008300  13,'PANEL(ISREMFLM)'
008400  14,'PANEL(ISREMARC)'
008410  15,'PANEL(USERMODL)'
008500  ' ',' '
008600    *,'?')
```

- That's all the changes for this member. So press F3 to save it.

- We now need to create our own model selection panel. You will have one of these for each selection of models. We are only creating a single selection list with a single model in it.
- Go to ISPF Edit (option 2 from the main menu) and enter '***userid*.ISPFELAB.PANELS'** in the name field and enter a member name of **USERMODL** to begin editing the member. The member name is the member we entered as a selection in panel ISREMCLS.
- We can use an existing ISPF panel to base our example on. So once again we have to go find the ISPF example. On the command line type **DDLIST** and press **Enter**. This will start ISRDDN.
- On the command line in ISRDDN type **M ISREMASG ISPPLIB** and press **Enter**. This will find the data set that contains the member as shown below:

```
                        Current Data Set Allocation          Member was found
Command ===> _____________________________________________ Scroll ===> PAGE

  Message             Act DDname   Data Set Name    Actions: B E V M F C I Q
 Member: ISREMASG     > _  ISPPLIB  ISP.SISPPENU
                      > _           SYS1.SBLSPNL0
                      > _           ISF.SISFPLIB
```

- Make a note where the member comes from, in this case 'ISP.SISPPENU(ISREMASG)'
- Press F3 to come out of View on the member and then F3 to come out of ISRDDN so that you are back in your empty member. On the command line type **COPY 'ISP.SISPPENU(ISREMASG)'** and press **Enter**.
- On line 13 change the literal **Assignment Statements** to **My SHARE models**.
- On line 20 next to the A1, change the option and description to **IEFBR14 - Simple IEFBR14 job**. Then delete the lines with A2 through A9. Such that the)AREA SAREA40 looks like this:

```
EDIT       LXD2.ISPFLAB.PANELS(USERMODL) - 01.00          Columns 00001 00072
Command ===> _____________________________________________ Scroll ===> CSR
001900 )AREA SAREA40
002000  A1 IEFBR14    - Simple IEFBR14 job
002900 )INIT
```

- We now need to change the TRANS statement, so on line 35 change SIMPLE,A1 to IEFBR14,A1 and remove the superfluous options, such that the line looks like this:

```
004100 IF (&ZCMD = 'TRANSTRUNC') &ZCMD = TRUNC(&ZCMD,8)
004200 &ZCMD = TRANS (&ZCMD
004300 IEFBR14,A1
004400 ' ',' ' *,*)
004500 IF (&ZCMD ¬= ' ') &ZCMD = '&ZCMD..&ZTMPTRL'
```

- Next we need to pass our skeleton name through to the ISPF program ISRECMBR. On line 39 change **ISREMAS1** to **IEFBR14** which is the name of the member we saved in the SKELS library. Also remove options A2 through A9, such that the &ZSEL assignment looks like this:

```
004600 &ZSEL = TRANS (TRUNC (&ZCMD,'.')
004700  A1,'PGM(ISRECMBR) PARM(IEFBR14)'
005600  ' ',' '
005700    *,'?')
```

- Finally we need to also modify the point and shoot field assignments by removing the ones we no longer require. Remove the point and shoot assignments for A2 through A9 such that the)PNTS area looks like this:

```
007200 )PNTS
007300 FIELD(ZPS01001) VAR(ZCMD) VAL(A1)
008200 )END
```

- Press F3 to save the member. That's all the changes that were required. So now we can test our changes.
- At this point it might be a good idea to come out of ISPF and come back in again to pick up the new panels and skeletons
- Go into edit on your REXX member **SHARE120** and on the command line type MOD CLASS. This will bring up a list of the model classes, except this time you should see your "Personal" class at the bottom, like this:

```
                          Model Classes
 Option ===> _____________________________________________________________

 Enter number or Class of model.
 Enter END command to cancel MODEL command.

 1  CLIST    - ISPF services in CLIST commands
 2  COBOL    - ISPF services in COBOL programs
 3  EXEC     - ISPF services in EXEC commands
 4  FORTRAN  - ISPF services in FORTRAN programs
 5  MSGS     - Message format
 6  PANELS   - Panel formats and statements
 7  PLI      - ISPF services in PLI programs
 8  SKELS    - File tailoring control statements
 9  PASCAL   - ISPF services in PASCAL programs
 10 REXX     - ISPF services in TSO/REXX commands
 11 DTL      - ISPF Dialog Tag Language formats and statements
 12 C        - ISPF services in C/370 programs
 13 SCLM     - SCLM Project Definition Macros
 14 ARCHDEF  - SCLM Architecture Definition templates
 15 PERSONAL - My Personal templates
```

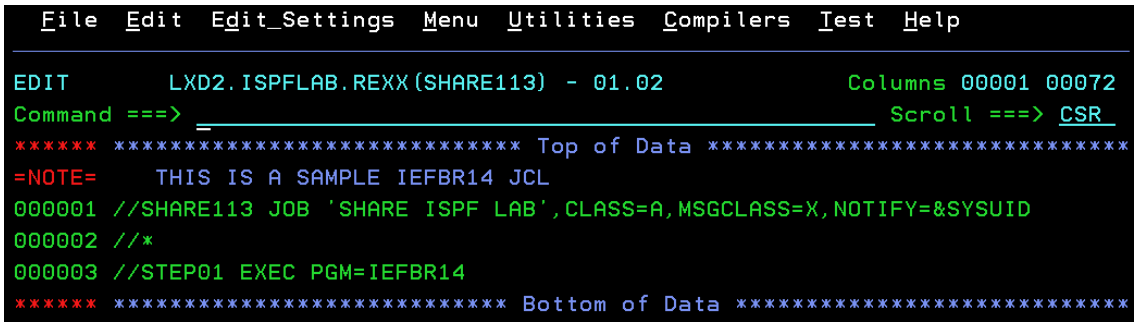
- Select 15 to make your personal class the active one, and press Enter.
- Now when you enter MOD on the command line and press Enter you will see a list of your created models, in this case just the one for now:

```
                        My SHARE models
Option ===> _____________________________________

Enter number or statement name.
Enter END command to cancel MODEL command.

A1 IEFBR14    - Simple IEFBR14 example
```

- Select A1 to use the IEFBR14 model and press Enter. You will see your sample IERBR14 code with the comment you added:



- If you have time you can now try and add another model to your model selection panel following the instructions above. Remember you will only now have to create the SKELS members and then modify your USERMODL panel to point to them.

Lab 3 – Edit Macros

In this Lab we will be looking at Edit Macros, what they are used for and how to create our own edit macros.

Edit macros are executable modules that can mimic the actions of the ISPF editor, in conjunction with other more complex code. They can be written in a number of different languages but are most commonly written in REXX these days. There are a number of examples in the ISPF Edit and Edit Macros guide :
http://publibz.boulder.ibm.com/cgi-bin/bookmgr_OS390/BOOKS/ISPZEM80

We will go through a couple of exercises to step through some simple edit macros and look at different ways of invoking them.

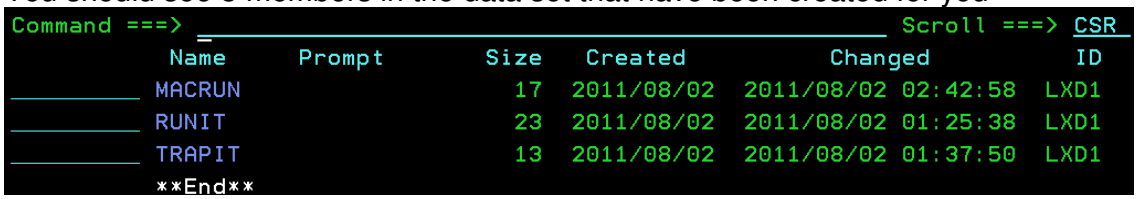
In order for edit macros to be invoked they must be in a data set that is allocated to either the SYSPROC or SYSEXEC allocation. This was done for you when we created your REXX libraries and started the lab. So we will be creating members in your **userid.ISPFELAB.REXX** library.

Exercise 1

In this exercise we will look at an edit macro that has been created for you to directly run a REXX exec whilst you are editing it.

Let's take a look at the first edit macro

- Go to ISPF Data set List (option 3.4 from the main menu) and enter ***userid*.ISPFELAB.REXX** in the Dsname level field to list your REXX library.
- Next to the data set enter an **E** to begin editing it and press Enter.
- You should see 3 members in the data set that have been created for you



- Let's enter a **V** next to the **RUNIT** member and press enter to look through it.
- The first thing to point out is line 3 which says **'ISREDIT MACRO'**. This is required for Edit macros so that ISPF knows this is a macro rather than a normal exec.
- This exec will get the data set name and member, if a member exists, and then build up an execute command for that exec to run it.
- Line 8 calls the edit macro command to get the current data set name.
- Line 9 calls the edit macro command to get the current member name if it exists.
- Lines 11 and 12 check if the High Level Qualifier is your userid, if it is then it issues a SAVE of the current member.
- Lines 14 and 16 then build up the execute command, depending on if the file we are in is a PDS member or a sequential data set.

- Finally line 22 runs the exec.

Let's try running the macro to see what happens

- Come out of View on the RUNIT member by pressing PF3
- On the command line enter **S TESTIT** to create a new member and press enter.
- You should be presented with an empty member.
- On line 1 type /*** REXX ***/
- On Line 2 type **Say 'Hello world from SHARE'**
- You should have 2 lines that look like this:

```
Command ===> _____________________________________________ Scroll ===> CSR
****** ***************************** Top of Data ********************************
==MSG> -Warning- The UNDO command is not available until you change
==MSG>           your edit profile using the command RECOVERY ON.
000001 /* REXX */
000002 Say 'hello world from SHARE'
****** **************************** Bottom of Data ******************************
```

- So let's invoke our edit macro. On the command line enter **RUNIT** and press enter.
- On the resulting screen you should see the result of your Say command.

Exercise 2

In this exercise we will look at an edit macro that has been created for you to trap the contents of a TSO command into a member. This is useful if for example you want to capture a list of member names in a certain data set.

Let's take a look at the next edit macro

- Come out of edit on the exec you just wrote in Exercise 1, at this point you should be back in the member list.
- Let's enter a **V** next to the **TRAPIT** member and press enter to look through it.
- Again, at the top, in line 2, we see the **ISREDIT MACRO** statement at the top. However this time we are going to be passing a parameter to it. So you can see the **(PARM)** next to the MACRO statement.
- We are going to be using the REXX **outtrap** command to trap SYSOUT output into a stem variable, called **STEM.** In line 3.
- In line 4 we then run the TSO command that has been passed as a parameter and then we turn output trapping off.
- We now have some edit macro commands. The first, in line 6, excludes all lines between the system labels **.ZFIRST** and **.ZLAST**.
- We then process a loop, going through each element in the stem array and use the edit macro **LINE_AFTER** command to create a new line in the member, after the last line, with the contents of the stem element.
- One finished we issue the **RESET** command to display all the lines again, then locate the first line and finally page up 1 page to get the cursor onto the command line at the top.

Let's try running the macro to see what happens

- Come out of View on the TRAPIT member by pressing PF3
- On the command line enter **S memlist** to create a new member and press enter.
- You should be presented with an empty member.
- On the command line type **trapit listd 'HLA.SASMMOD1' m** and press enter.
- You will see that the **TSO LISTD** command is run for the specified data set and the contents of the command are trapped into your member.

Exercise 3

In this exercise we will look at invoking an edit macro against a number of different members in a data set. This is useful if you have a lot of boring edit changes to make to a lot of members in a data set. You can create the edit change commands in an edit macro and then run it against every member in a data set.

In the exercise, we will write a macro to change the copyright date in all members in a sample data set. Alternatively, if you are tired of typing then you can copy the the edit macro from 'LXD1.EXEC(SHAREMAC)' rather than following the following instructions.

Let's create the edit macro

- Come out of edit on the member you created in Exercise 2, at this point you should be back in the member list.
- On the command line enter **s sharemac** to create a member in our REXX data set.
- On the first line enter /*** REXX ***/ so that ISPF knows this is a REXX exec.
- On line 2 we are going to tell ISPF that this is an edit macro so enter **"ISREDIT MACRO"**
- We now need to find the line with the copyright statement, so let's issue an edit find command. Type the following on the next blank line: **"ISREDIT Find 'Copyright IBM Corporation' First"**
- Based on the return code of the find we will now perform a change command. So check the return code and add a Do block with the following code : **If RC = 0 Then Do**
- We now have some macro commands. The first is to get the cursor position to give us the current line. So enter the command **"ISREDIT (Line,Col) = CURSOR"**
- On the next line we now set a label against the current line with the command : **"ISREDIT LABEL "Line" = .a"**
- Finally we perform the change command with the command : **"ISREDIT CHANGE '2008' '2011' .A .A ALL"**
- Finish the do block off with an **End** statement.

- Finally add the **"ISREDIT END"** command to leave the member once the macro runs.
- To put that all into context you should have a member that looks like this:

```
/* REXX */
"ISREDIT MACRO"
"ISREDIT Find 'Copyright IBM Corporation' First"
If RC = 0 Then
Do
  "ISREDIT (Line,Col) = CURSOR"
  "ISREDIT LABEL "Line" = .a"
  "ISREDIT CHANGE '2008' '2011' .A .A ALL"
End
"ISREDIT END"
```

- Press PF3 to save the member.

Let's take a look at the REXX exec that will run the macro and run it

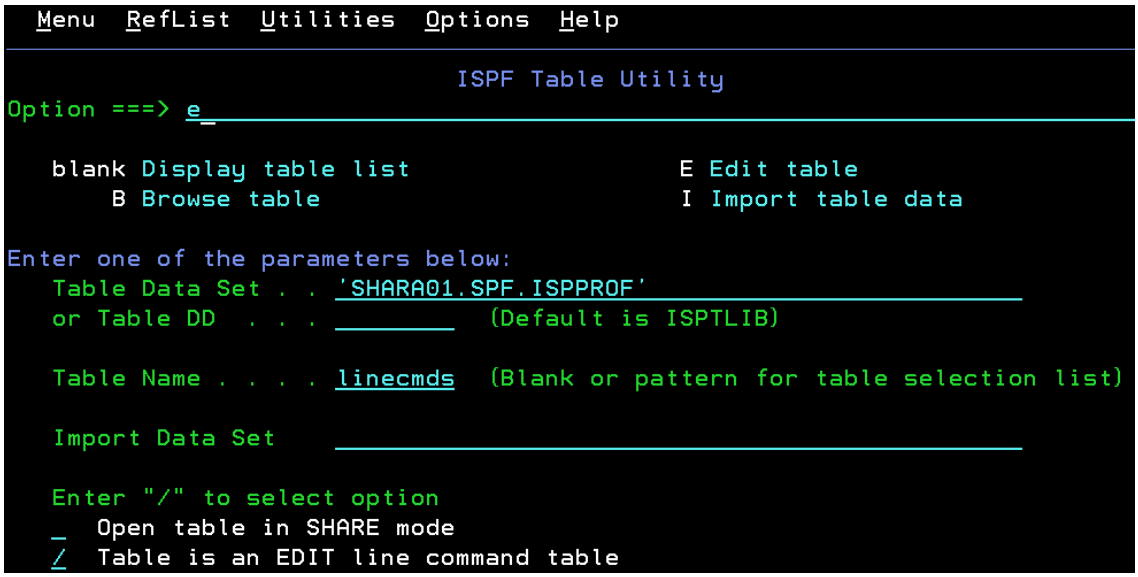
- Come out of edit on the exec you just created, at this point you should be back in the member list.
- Let's enter a **E** next to the **MACRUN** member in your EXEC library
- We won't go into too much detail as to what is happening as this is not an ISPF developers lab. But the **MACRUN** exec uses the ISPF LM* services to allocate a dataset, then loop through the members, and then call the EDIT service with a macro for each member in the data set.
- On line 3 change the Dataset specified from **SHARxxx.ISPFELAB.SAMPLE** to the userid you are using, **<userid>.ISPFELAB.SAMPLE**.
- We can now use the **RUNIT** exec we looked at in Exercise 1 to execute this. So on the command line enter **RUNIT** and press enter.
- You should now see the exec looping though the members as it runs the macro against them.
- You can go to ISPF 3.4 and take a look in <userid>.ISPFELAB.SAMPLE to see that the changes were made.

Lab 4 – Customizable line commands

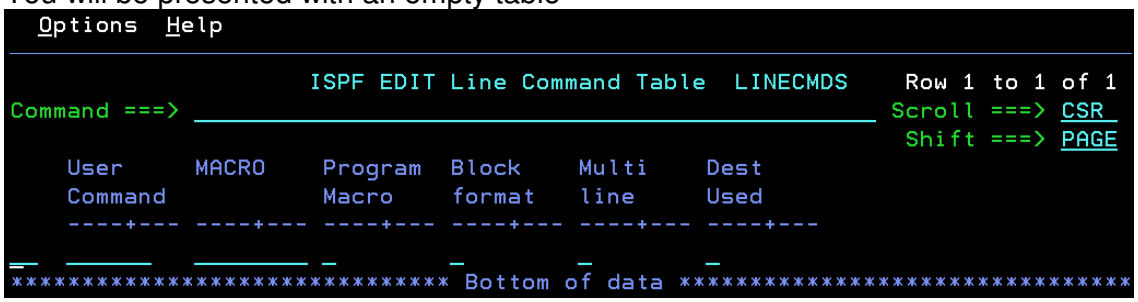
In this Lab we will be looking at a feature added in ISPF 6.3 (z/OS 1.13) where the ability to create your own line commands for the ISPF Editor was provided. The facility was provided through a combination of ISPF table configuration and edit macros, which you will have learnt about in Lab 3. This lab will talk you through the set-up to enable you to run your own line commands and will provide a sample line command edit macro that you can use as a model to get ideas from.

Let's take a look at how ISPF knows about the commands

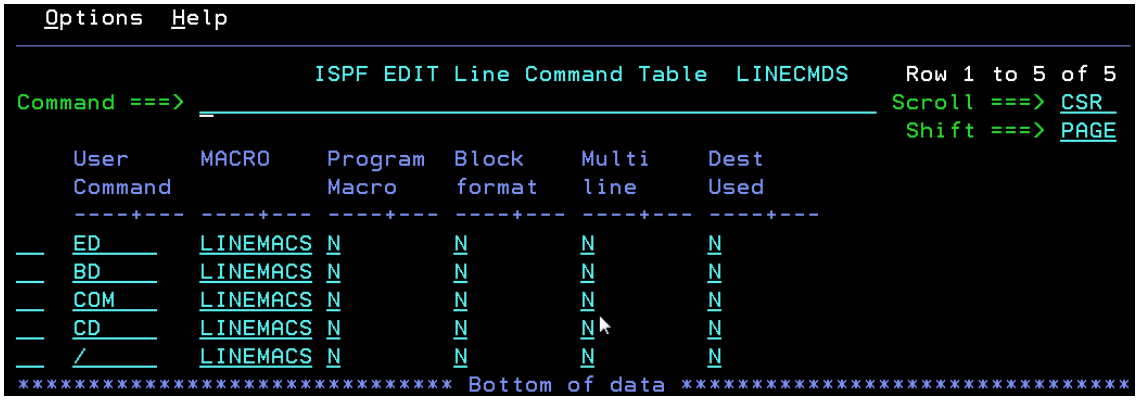
- From the ISPF primary options menu go into option 3.16, the ISPF Table Utility
- in the Table Data Set entry field enter the name of your ISPF profile data set. This will be called **'SHARxxx.SPF.ISPPROF'** where SHARExxx is the userid you are logged onto TSO with.
- Next in the Table Name entry field enter the name of a new ISPF table that will contain the new line commands you are going to create. For this lab you can use **LINECMDS**.
- Next, enter a / next to the option **"Table is an EDIT line command table"**
- Finally, enter **E** on the command line and press enter.



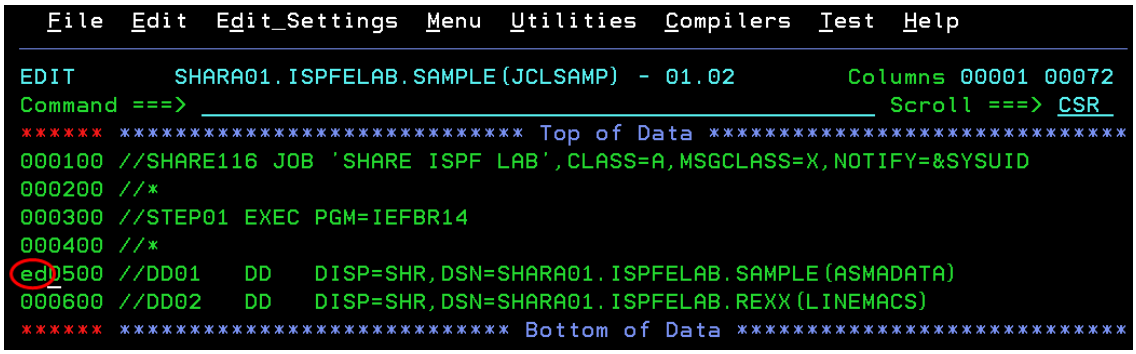
- You will be presented with an empty table



- The edit macro has already been written for you, so we are just going to associate the various line commands to the macro that processes them. In this example we have a single edit macro that processes all of our commands, but it is possible to have separate macros for each command if you wish.
- Add the User Command of **ED** to the list with the MACRO set to **LINEMACS** and all 4 options on the screen set to **N**.
- Do the same for the **BD**, **COM**, **CD** and / User commands. To insert a new line enter an **I** in the prefix area on the left and press enter.



- Press PF3 to leave the table editor and then PF3 again to leave option 3.16.
- Next go to ISPF Edit, option 2. You need to tell the editor the name of your line commands table. There is a new field in the ISPF editor panel. So in the ***Line Command Table*** entry field on the editor panel enter the name of the ISPF table you just created, **LINECMDS** in this example.
- Enter the name of a data set that you are going to edit. For the purposes of this lab enter **'SHARxxx.ISPFELAB.SAMPLE'** in the other data set entry field. Where SHARxxx is the userid you are logged onto TSO with.
- Select member JCLSAMP for edit. First of all change the string **SHARE.ISPF** to **SHARxxx.ISPF**, where SHARxxx is your userid that you are logged on with. This way we will have no conflicts when testing the line commands out.
- In the prefix area next to //DD01 you can now try out your new line commands. Enter CD and you will be told the data set exists. Enter BD and you will be taken into browse on the member. Enter ED and you will be taken into edit. Enter COM and the line will be commented out.



- Let's go take a look at the ISPF Edit Macro that processes these commands. There is a copy in your **SHARxxx.ISPFELABS.REXX** so just enter ED on the line next to //DD02 to go into edit on the LINEMACS REXX exec.
- You will see from the top of the exec that ISPF just passes in the command that is being processed. The exec then works out what it needs to do to each line for the particular commands.
- Take a look through the exec so that you understand how it works out what to do.

Congratulations!

You have successfully completed the ISPF Editor Lab!

Appendix A - Code samples

MACRUN

```
/* REXX */
  Dataset = 'SHARxxx.ISPFELAB.SAMPLE'
  Macro   = 'SHAREMAC'
  MEMBER  = ''

  Address ISPEXEC "LMINIT DATAID(DID) DATASET('"dataset"') ENQ(SHR)"
  Address ISPEXEC "LMOPEN DATAID("DID") OPTION(INPUT)"
  Address ISPEXEC "LMMLIST DATAID("DID") OPTION(LIST) MEMBER(MEMBER)"
  Do while (RC = 0)
    Say 'Processing member' member
    Address ISPEXEC "EDIT DATAID("DID") MEMBER("MEMBER") MACRO("MACRO")"
    Address ISPEXEC "LMMLIST DATAID("DID") OPTION(LIST) MEMBER(MEMBER)"
  End
  Address ISPEXEC "LMCLOSE DATAID("DID")"
  Address ISPEXEC "LMFREE  DATAID("DID")"

Return
```

RUNIT

```
/* REXX */
   Address ISPEXEC
   'ISREDIT MACRO'
   If rc = 0 Then
   Do
     'ISPEXEC CONTROL ERRORS RETURN'
     did = ''
     'ISREDIT (DID) = DATASET'
     'ISREDIT (MEM) = MEMBER'
     Parse Var did hlq '.' .
     If hlq = Userid() Then
       'ISREDIT SAVE'
     If mem = '' Then
       exe = "'" || did || "'"
     Else
       exe = "'" || did || '(' || mem || ")'"
   End
   Else
   Do
     Arg exe
   End
   Address TSO 'EX' exe
Exit
```

TRAPIT

```
/* REXX */
  Address ISREDIT 'MACRO (PARM)'
  xx = outtrap('STEM.')
  Address tso parm
  xx = outtrap('OFF')
  Address ISREDIT 'X ALL .ZF .ZL'
  Do p = 1 to stem.0
    line = stem.p
    Address ISREDIT 'LINE_AFTER .ZL = (LINE)'
```

```
  End
  Address ISREDIT 'RESET'
  Address ISREDIT 'LOCATE 1'
  Address ISREDIT 'UP PAGE'
```

LINEMACS

```
/* REXX ---------------------------------------------------------------*/
/*                                                                     */
/* Simple ISPF EDIT user line command processor.  Valid line commands */
/* are setup using option 3.16 with the resulting table name being    */
/* specified in the "Line Command Table . ." field of the Edit panel. */
/*                                                                     */
/* Commands processed by this EXEC are                                 */
/*                                                                     */
/*    /   - reposition current line                                    */
/*    CD  - Check if data set exists                                   */
/*    BD  - Browse data set                                            */
/*    ED  - Edit data set                                              */
/*    COM - Comment out line based on Data set type.                   */
/*                                                                     */
/*---------------------------------------------------------------------*/
  ValidCommands = ' CD / COM BD ED ? Q '

  Address ISREDIT
  "MACRO (PARM) NOPROCESS"                    /* Get line command      */
  Address ISPEXEC "CONTROL ERRORS RETURN"     /* Return ISPF errors    */
  If Wordpos(PARM,ValidCommands) = 0 Then
  Do                                          /* Only process valid    */
    ZINFO=PARM                                /* commands....          */
    Address ISPEXEC "SETMSG MSG(ISRE041)"
    Exit 8
  End

  "PROCESS RANGE" PARM                        /* Get range for command */
  If RC > 0  Then
  Do
    Address ISPEXEC  "SETMSG MSG(ISRZ002)"
    Exit 8
  End

  "(THISDS) = DATASET"

  RepositionLine = 0

  "(START) = LINENUM .ZFRANGE"          /* Start line number in range */
  "(STOP)  = LINENUM .ZLRANGE"          /* End line number in range   */
  "(DW)  = DATA_WIDTH"                  /* Width of the editable data */

  Do a = START TO STOP                  /* Loop through the lines     */
    "(LINE) = LINE "a                   /* get existing line contents */
    Select                              /* Process this line          */
      When(PARM = "/")  Then            /* Make this line the current */
        RepositionLine = a              /* Line.                      */
      When(PARM = "CD")  Then           /* Check if Data set exists   */
        Call CheckDSN(line)
      When(PARM = "BD")  Then           /* Browse Data set            */
        Call BrowseDSN(line)
      When(PARM = "ED")  Then           /* Edit Data set              */
        Call EditDSN(line)
      When(PARM = "COM")  Then          /* Comment out this line      */
        line = Comment(line)
      When(PARM = "?" | PARM = 'Q')  Then
      Do
```

```
        Say '/   - reposition current line'
        Say 'CD  - Check if data set exists'
        Say 'BD  - Browse data set'
        Say 'ED  - Edit data set'
        Say 'COM - Comment out line'
      End
      Otherwise
        Nop
    End
    "LINE "A" = (LINE)"                /* replace line contents      */
  End

  IF RepositionLine <> 0 Then
  Do
    'UP MAX'
    'DOWN' RepositionLine
  End

Exit 0

/*--------------------------------------------------------------------*/
/* Check if the dataset on a DSN=nnnnnn exists or not                 */
/*--------------------------------------------------------------------*/
CheckDSN: Procedure

  Arg Inline
  Parse Var Inline . 'DSN=' dsn .
  If dsn <> '' Then
  Do
    Parse Var dsn dsn ',' .
    dsn = "'" || Strip(dsn) || "'"
    Exists = Sysdsn(dsn);
    If Exists <> 'OK' Then
      Say 'Dataset' dsn 'does not exist.'
    Else
      Say 'Dataset' dsn 'exists.'
  End
  Else
    Say 'No DSN= on current line...'

Return

/*--------------------------------------------------------------------*/
/* Browse data set listed on the selected line                        */
/*--------------------------------------------------------------------*/
BrowseDSN: Procedure

  Arg Inline
  Parse Var Inline . 'DSN=' dsn .
  If dsn <> '' Then
  Do
    Parse Var dsn dsn ',' .
    dsn = "'" || Strip(dsn) || "'"
    Exists = Sysdsn(dsn);
    If Exists = 'OK' Then
      Address ISPEXEC "BROWSE DATASET("dsn")"
    Else
      Say 'Dataset' dsn 'does not exist.'
  End
  Else
    Say 'No DSN= on current line...'

Return

/*--------------------------------------------------------------------*/
/* Edit data set listed on the selected line                          */
```

```
/*-------------------------------------------------------------------*/
EditDSN: Procedure

  Arg Inline
  Parse Var Inline . 'DSN=' dsn .
  If dsn <> '' Then
  Do
    Parse Var dsn dsn ',' .
    dsn = "'" || Strip(dsn) || "'"
    Exists = Sysdsn(dsn);
    If Exists = 'OK' Then
      Address ISPEXEC "EDIT DATASET("dsn")"
    Else
      Say 'Dataset' dsn 'does not exist.'
  End
  Else
    Say 'No DSN= on current line...'

Return

/*-------------------------------------------------------------------*/
/* Comment out the selected line                                     */
/*-------------------------------------------------------------------*/
Comment: Procedure Expose THISDS

  Parse Arg Inline
  ds = Reverse(THISDS)
  Parse Var ds lq '.' .
  lq = Reverse(lq)

  Select
    When Pos('JCL',lq) > 0 |,
         Pos('JOBS',lq) > 0 |,
         Pos('SAMPLE',lq) > 0 Then
      If Left(Inline,3) = '//*' Then
        Inline = Substr(Inline,4)
      Else
        Inline = '//*' || Inline

    When Pos('COB',lq) > 0 Then
      If Substr(Inline,7,1) = '*' Then
        Inline = Left(Inline,6) || ' ' || Substr(Inline,8)
      Else
        Inline = Left(Inline,6) || '*' || Substr(Inline,8)

    Otherwise
    Do
      Tline = Strip(Inline)
      If Left(Tline,2) = '/*' Then
      Do                                  /* line is already commented */
        Parse Var Inline b1 '/*' b2 '*/' .
        Inline = b1 || '  ' || b2
      End
      Else
      Do
        Inline = Strip(Inline,'T')
        If Left(Inline,2) = '  ' Then
          Inline = '/*' || Substr(Inline,3)|| '*/'
        Else
         Inline = '/*' || Strip(Inline) || '*/'
      End
    End
  End
Return(Inline)
```