BCI433 Lab 3 (updated Winter 2020)

**Writing an interactive RPGLE screen program**

**Lab objectives:**

* **Create a display file (screen)**
* **Code an interactive RPGLE program using the display file**

**Lab Requirements:**

**Hand in the compile listing for INCTAXRPG**

**Successfully run INCTAXRPG similar to instructor program INCTAXRPGA.**

Start an RDi session

Start a ‘Green Screen’ (emulator) Session.

**Using** **Rational Developer for i (RDi):**

**Part A**

**Objectives:**

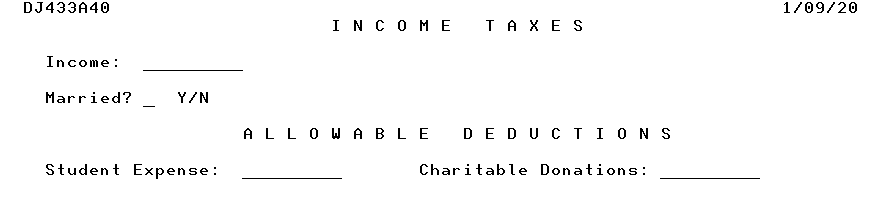
* **Use RDi Screen Designer to Create a Display File**

Display File screens in IBM i are very similar to forms you see on Web pages. Entering and compiling DDS code can produce these interactive screens. One option that may be used for entering DDS code is to use a GUI tool that allows you to visually select and enter what appears on the screen. Your selections are converted to DDS program code that is compiled to produce interactive screen records in a display file object. Those interactive screens are available to various program languages.

We will create a display file object called INCOMETDSP that uses two overlapping screen records. RPGLE, CLLE and COBOL programs can interact with the screens in the INCOMETDSP object. We will code an RPGLE program called INCTAXRPG to view and interact with the screen records. The following shows a run of the RPGLE program.

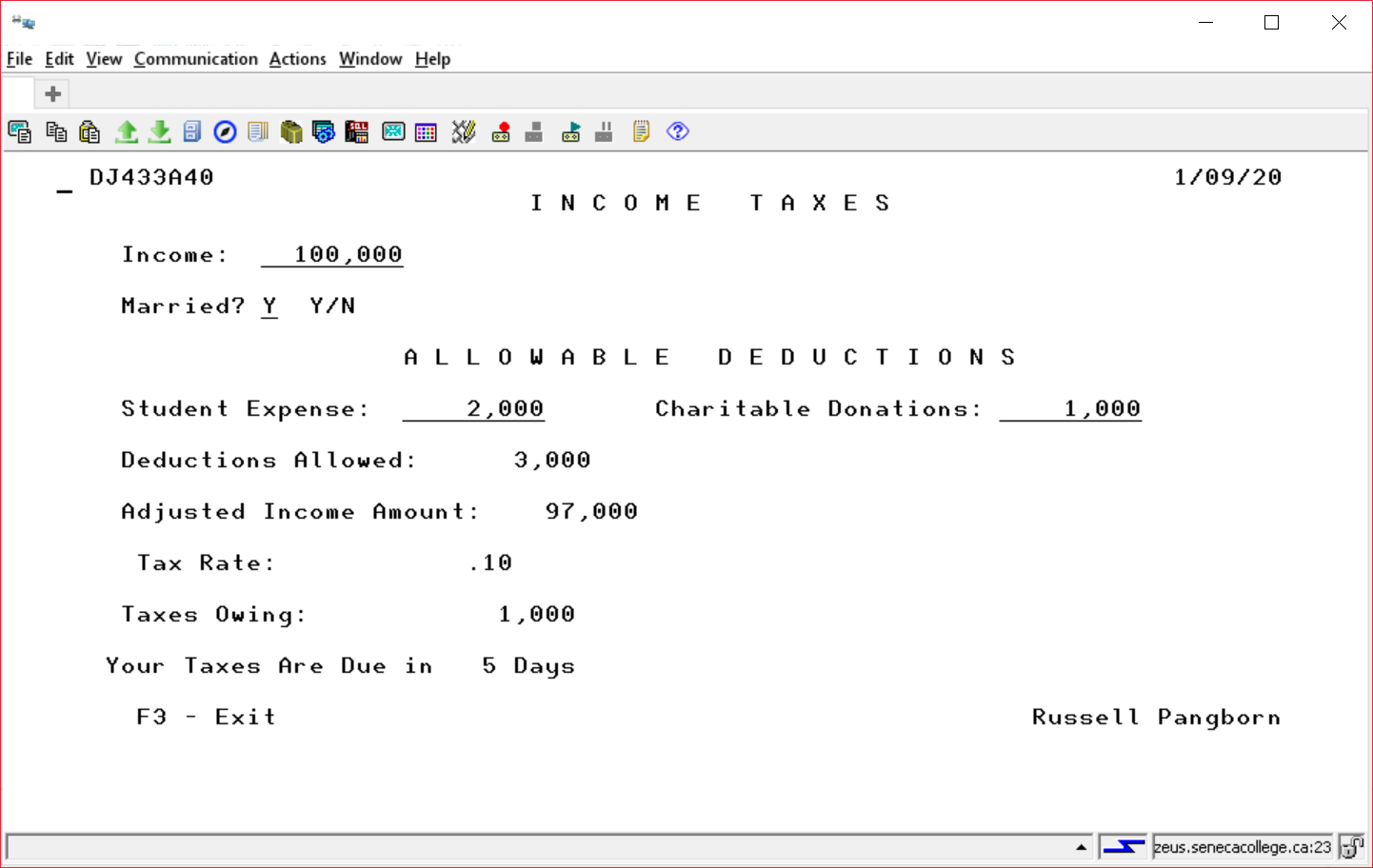
==>CALL INCTAXRPG

first screen called GETINCOME is displayed



Data is entered for all the fields and the ENTER key is pressed

A second screen called YOURTAX overlays the first screen and the input fields are protected



Some stub coding is done with this first run through of the tax program. For now the tax rate will always be .10, the Taxes owing will always be 1,000 and the taxes are always due in 5 days. Some calculations will be immediately supported. The deductions allowed will be the total of the student expense and the charitable donation amount. The adjusted income amount will take into account the total deductions.

(your program will not have the name Russell Pangborn in the bottom right corner – it will have your name)

The user can either press F3 to exit the program or enter to check out other tax quote combinations. F3 is only available for use from YOURTAX.

## Using Screen Designer to Produce a Display File

Source code for a Display File may be stored in a source file called **QDDSSRC**. You have already created this file in Lab 2 to store DDS code in the STUDENTS member that produced a physical file called STUDENTS.

DDS Stands for Data Description Specifications. The native IBM i language used to describe \*FILE objects.

Create a new member inside of QDDSSRC. The member name should be specified as INCOMETDSP and the member type should be DSPF. If you are not immediately put in the GUI and are just in the editor after creating the member, close the INCOMETDSP tab.   
  
Right click on your INCOMETDSP member, click on Open With and select SCREEN DESIGNER.

You should still see a Remote Systems view on your upper right. The following should appear:

An outline view on the right.

A tab in the centre that has your member name (INCOMETDSP) with two boxes below the tab. One box has a title of Screens and the other box includes a Screen and Records tab

Below this a work area should appear with a ruler that counts from column 1 to column 100

A palette is available to the right of this work area and this will be used to select and place fields and constants on our screen record.

Immediately below this work area will be tabs that allow us to work with the GUI or to work with actual DDS code

Further down is a Source Prompter tab. We have already used a source prompter in lab 2 to enter RPGLE code and to enter DDS code for a physical file program.

A properties box on the lower left which can be used to enter useful information about records and fields.

|  |  |
| --- | --- |
|  | .  Left click on Standard Record in the Palette area and drag this over to the left work area.  Record1 should show up in several spots. In a right top window and in a Properties box on the lower left. This name can be changed to something else here. We will change it to GETINCOME |

Create a date constant ( MM/DD/YY ) by clicking on Date Constant which is found in the palette underneath the Constants folder. Move your cursor over to the top left of your black grid screen and click once to place the date constant. As you move your cursor with the mouse, you should see the row and column numbers changing.

After you have placed this date, go back to the constants folder in the palette and use the arrows to scroll through other constant selections. We are also using the user constant. Refer to the finished first screen on a previous page to determine the placement of these constants and then add them to your work area.

A text constant shows below the user constant on the demo screen above as “Your Name”. Put your actual name here to identify this screen as your screen for the professor marking the interactive program.

There are several text constants to include on the first screen record to be named GETINCOME: “Income”, “Married?”, “Student Expense”, “Charitable Donations” and two titles. (check the screen shot for these text constants)

Click on Text Constant in the palette and then click to place this centred at the top of your screen.

You can change the default text of “Text constant” to “I N C O M E T A X E S” in one of two ways. Type directly in the box provided. If this is awkward click on the word “Text Constant” in the properties box in the lower left corner.

Put in all the Text constants using the finished product screenshot.

We need to put in fields beside the some of the text constant prompts.

Click on Named Field below the Fields folder in the palette. Place the named field beside the “Income:” constant.

This shows up as BBBBBBBBB on your work screen. Double click on the Properties box to specify information about this field. Change the provided field name to INCOMEAMT.

The default properties for this field are set as 9 character Input/Output field. The usage reflects this in Properties with a setting of “Both”.

Make this an 8 digit field by typing over the “9” that appears under Length on the right side of the properties view. This field will not include decimal positions.

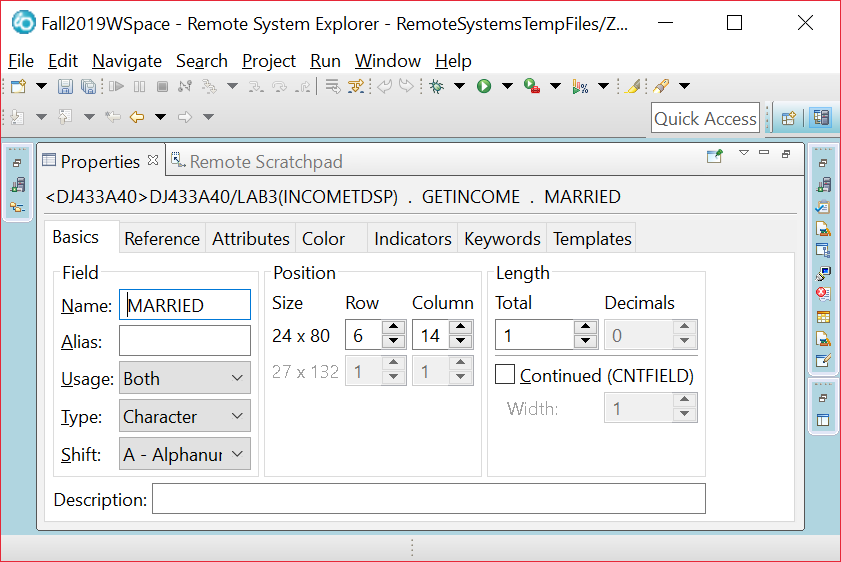
The usage for the field should remain as Both. This means it is input capable (the user can enter something into the field when it is presented on the screen record and it is output capable, the program or user can set the field contents and it will show up to the user the next time the screen is displayed. The type of field should be Zoned and Shift should be “Y” – Numeric only.

The next field will be an input/output character field with a length of one. The field name is MARRIED.

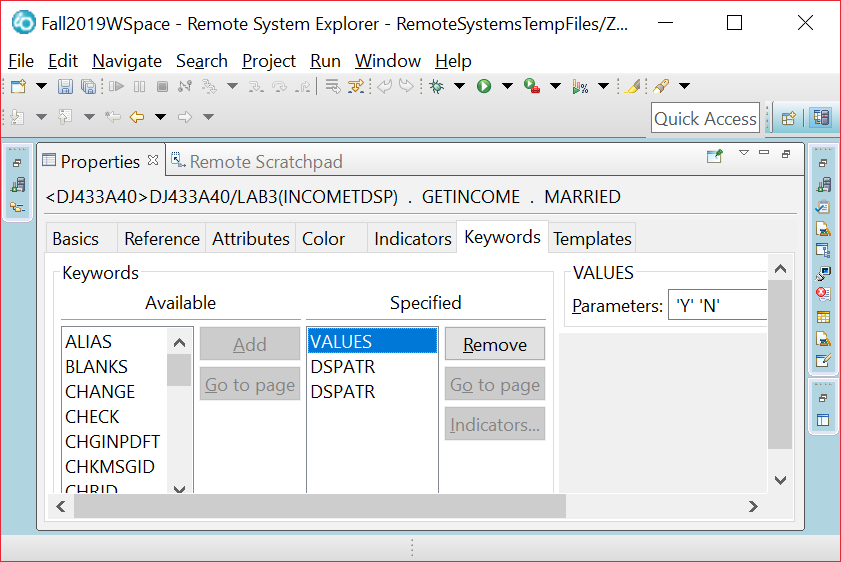
We are going to include some data validation with the display file rather than have the program handle it. There should only be two possible answers – ‘Y’ or ‘N’.

Click on the Keywords tab and look for the VALUES keyword from available keywords on the left hand side. When you click on VALUES, the Add button should become available. Click on this to add VALUES to the specified column box. There is a box on the top right where we can enter the allowable input values.

The following screen shots demonstrate this activity.

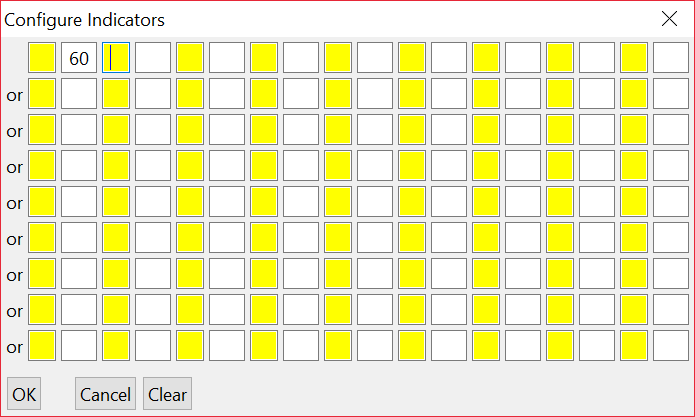


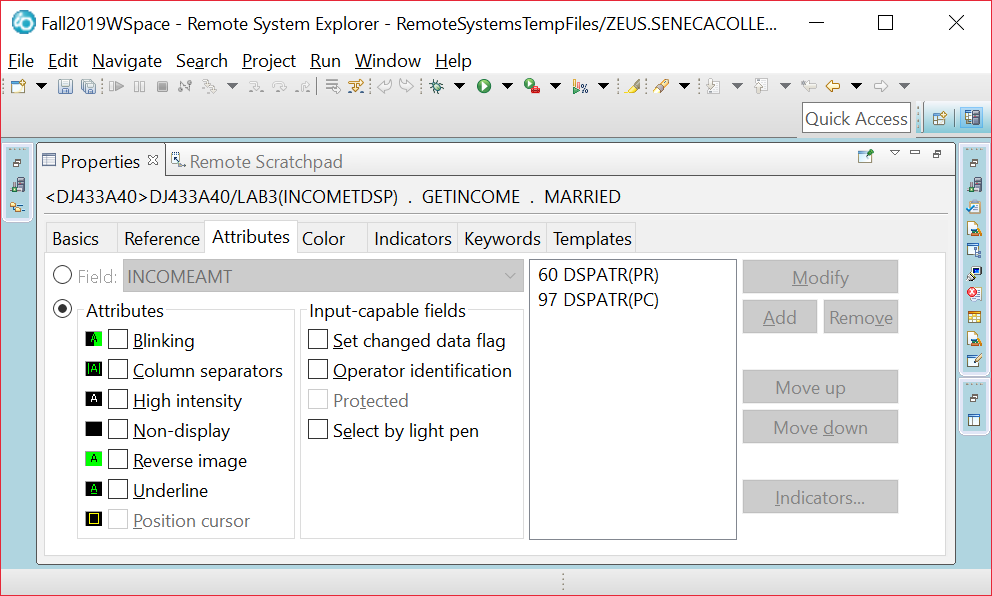
Keywords is selected, VALUES is added and the Parameters ‘Y’ and ‘N’ are specified.



The MARRIED field needs to be protected when the screen record shows and you do not want to allow input to the field. This can be done by clicking on the Attributes tab and then clicking on Protected. An Add button on the right becomes available so you can add this and this shows in a box on the right. If you click on what became added to the box, you can click on an indicators button on the lower right. You want to enter a 60 in the box as shows on the next page. The yellow box is for the letter N. We can have the field protected when indicator 60 is on or protected when indicator 60 is off which shows as N60. So, if we want the field protected when indicator 60 is off, we include the N. We will leave the yellow box blank.

(this screen shows when the indicator button is pressed – shown on the next page)





With the above screen, Protected was selected, then the Add button was clicked and finally the Indicators button was clicked to enter indicator 60. A position cursor feature was also included so the cursor would be on this field if indicator 97 was on.

Double click on the Properties tab to get back to the multi view screen

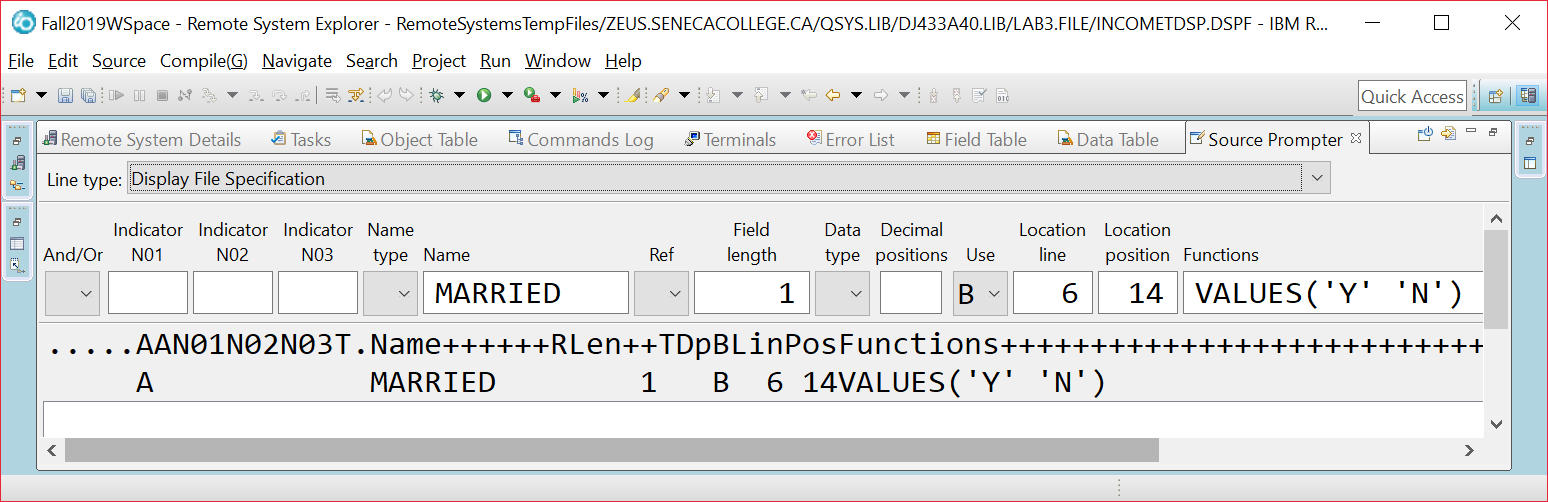
Include a 7 digit input/output numeric field for the student expense and for the charitable donations.

There are three tabs showing midway down your screen. **Design Source** and **Preview.** Click on the Source tab and press F4.

Double click on the Source Prompter tab that appears at the bottom view on your screen. On my screen the tabs show up as Remote System Details, Tasks, Object Table, Commands Log, Error List and Source Prompter.

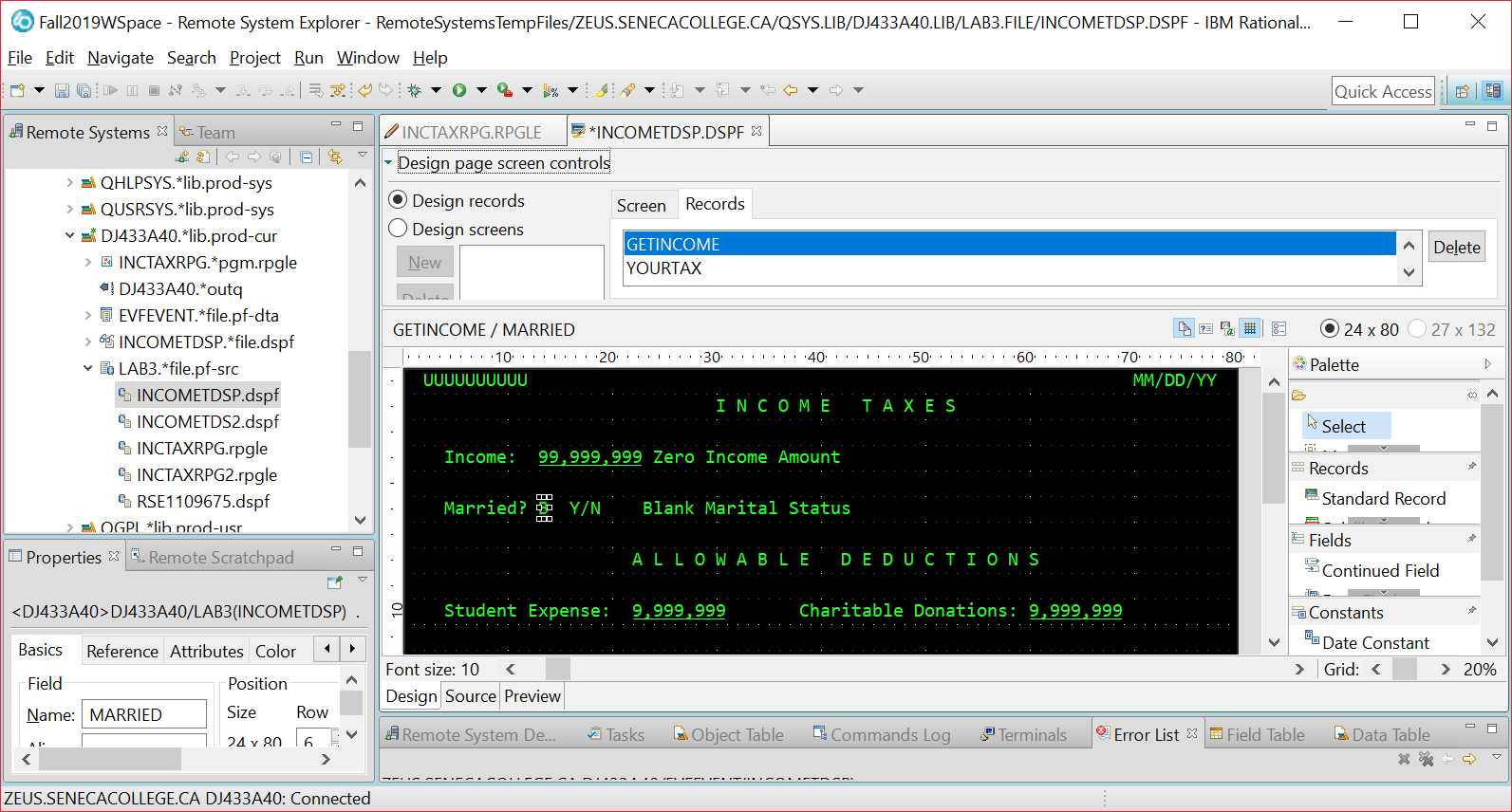
You want to view the following code.

The one character MARRIED field should be limited to the entries of Y or N.

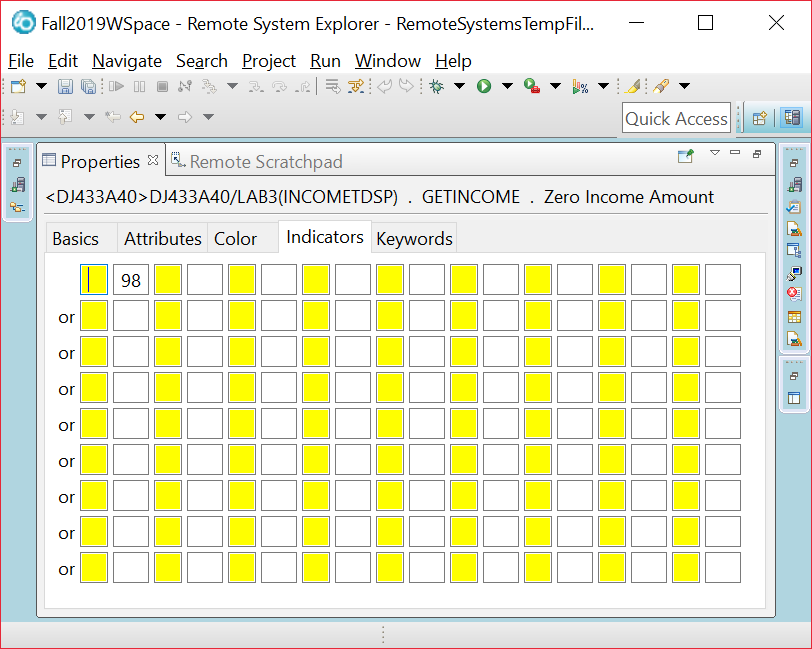


You could have directly entered the code you see in the Functions box.

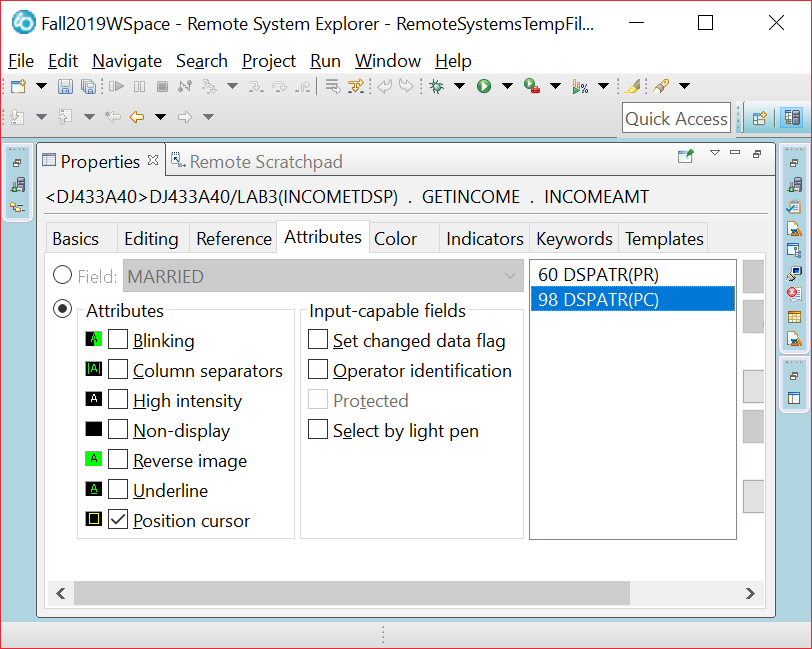
You can use the screenshots of the running program as a guide or the finished screen in Screen Designer is shown below. Remember to click on the Design tab to get back to the GUI interface.



There are two more text constants you need to add to your first interactive screen. “Zero Income Amount” and “Blank Marital Status”. These two text constants only show if you press enter and leave the IncomeAmt at zero or the Married field blank. In order to control the appearance of “Zero Income Amount”, after you have entered the text constant, right click on it and select properties. Then click on indicators and enter 98 as the indicator that has to be on in order for that text to show.

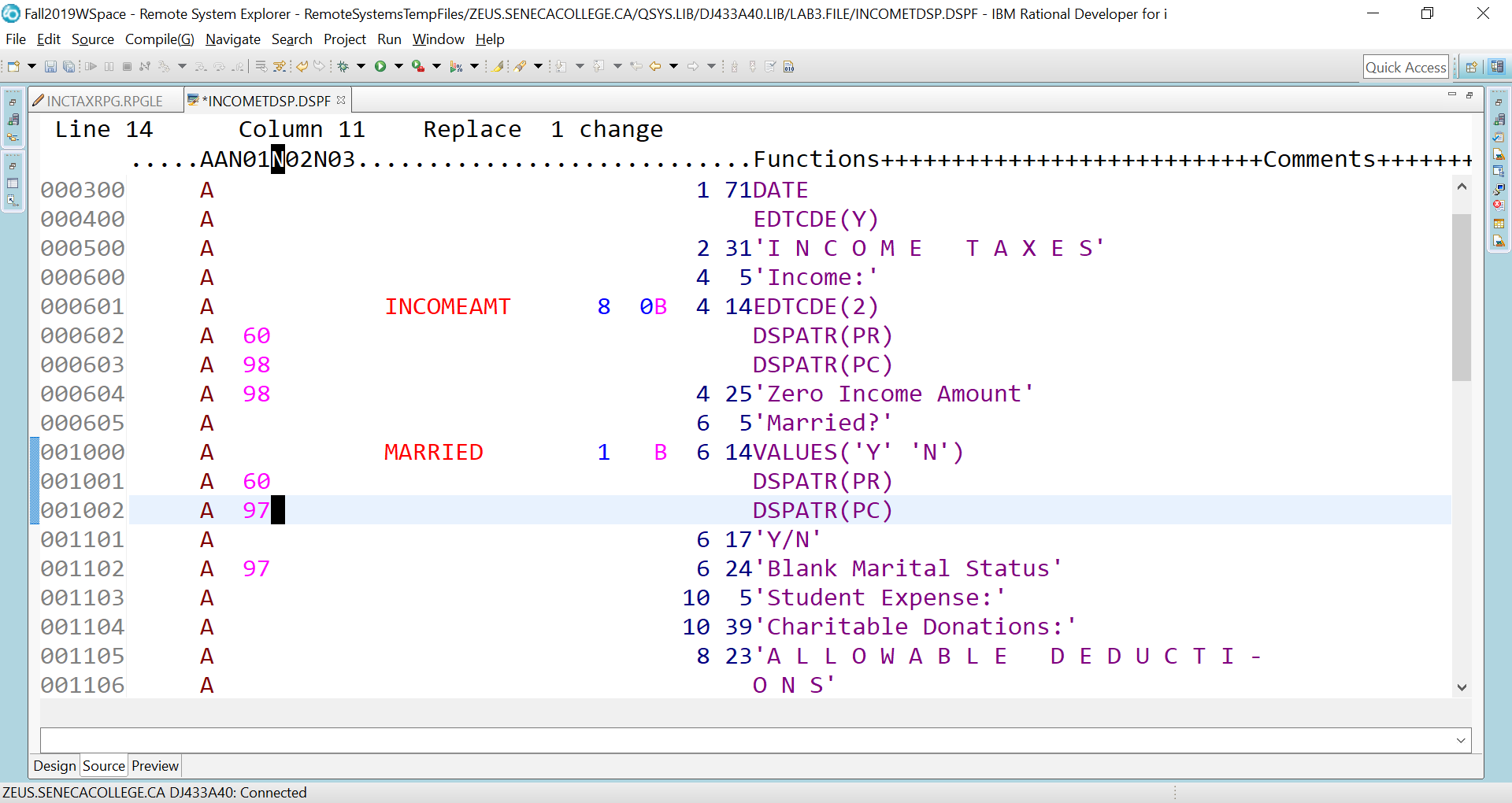


You want the cursor to be positioned at the IncomeAmt field when someone does leave it at zero. So go to the properties for the IncomeAmt field and include a position cursor attribute that is active when indicator 98 is on.



Do the same thing for Married using indicator 97.

DDS code has been generated for you. You can view and change this code. You have to be careful when changing it that your syntax is correct. In order to see the code, click on the **Source** tab. This is found just below your work screen. There are three tabs. **Design** (the one that provides a GUI for creating an interactive screen) **Source** (that allows you to view and change DDS code) and **Preview** (Not used in this lab).



Here is how the finished source code should appear for the first screen record

A R GETINCOME

A 1 3USER

A 1 71DATE

A EDTCDE(Y)

A 2 31'I N C O M E T A X E S'

A 4 5'Income:'

A INCOMEAMT 8 0B 4 14EDTCDE(2)

A 60 DSPATR(PR)

A 98 DSPATR(PC)

A 98 4 25'Zero Income Amount'

A 6 5'Married?'

A MARRIED 1 B 6 14VALUES('Y' 'N')

A 60 DSPATR(PR)

A 97 DSPATR(PC)

A 6 17'Y/N'

A 97 6 24'Blank Marital Status'

A 10 5'Student Expense:'

A 10 39'Charitable Donations:'

A 8 23'A L L O W A B L E DEDUCTIONS’

A STDEXPENSE 7 0B 10 23EDTCDE(2)

A 60 DSPATR(PR)

A DONATIONS 7 0B 10 61EDTCDE(2)

A 60 DSPATR(PR)

We have to be careful here. You could change the “VALUES” keyword to VALUE” and introduce an error to your code and possibly confuse the Design Tab’s interpretation of the code.

According to the DDS code showing on this lab, what line and column(or position) number does the MARRIED

field appear on? \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 6 21 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Your code can have this placed on a different line and position.

Go back to the Design tab and place a named field somewhere beside the MARRIED field. Check this out in the code by clicking on the Source tab.

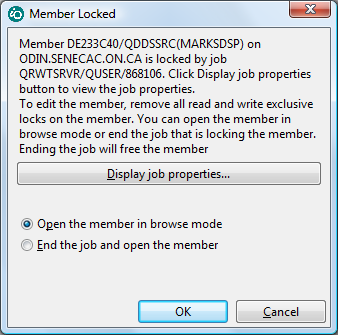
Notice that this new field appears seemingly out of order in the code. It appears at the bottom of the generated code. If you take a look at the line and position number it should show as the same line number you had for your MARRIED field.

If this bothers you, you could type an “M” where the line numbers for your code shows (Move).

Then put your cursor on the line number for the MARRIED field and type an “A” (After). Try it.

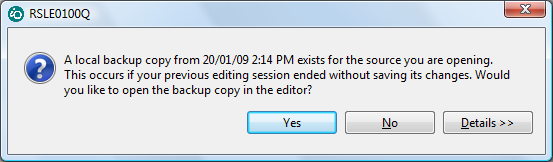
Click on File at the top right hand corner of your screen and select Save. This is a good periodic practice to recover from a crash or other problem.

If you do have a crash, you may restart and get the following message.



You can change the action to End the Job and open the member.

Your selection here depends on your situation.



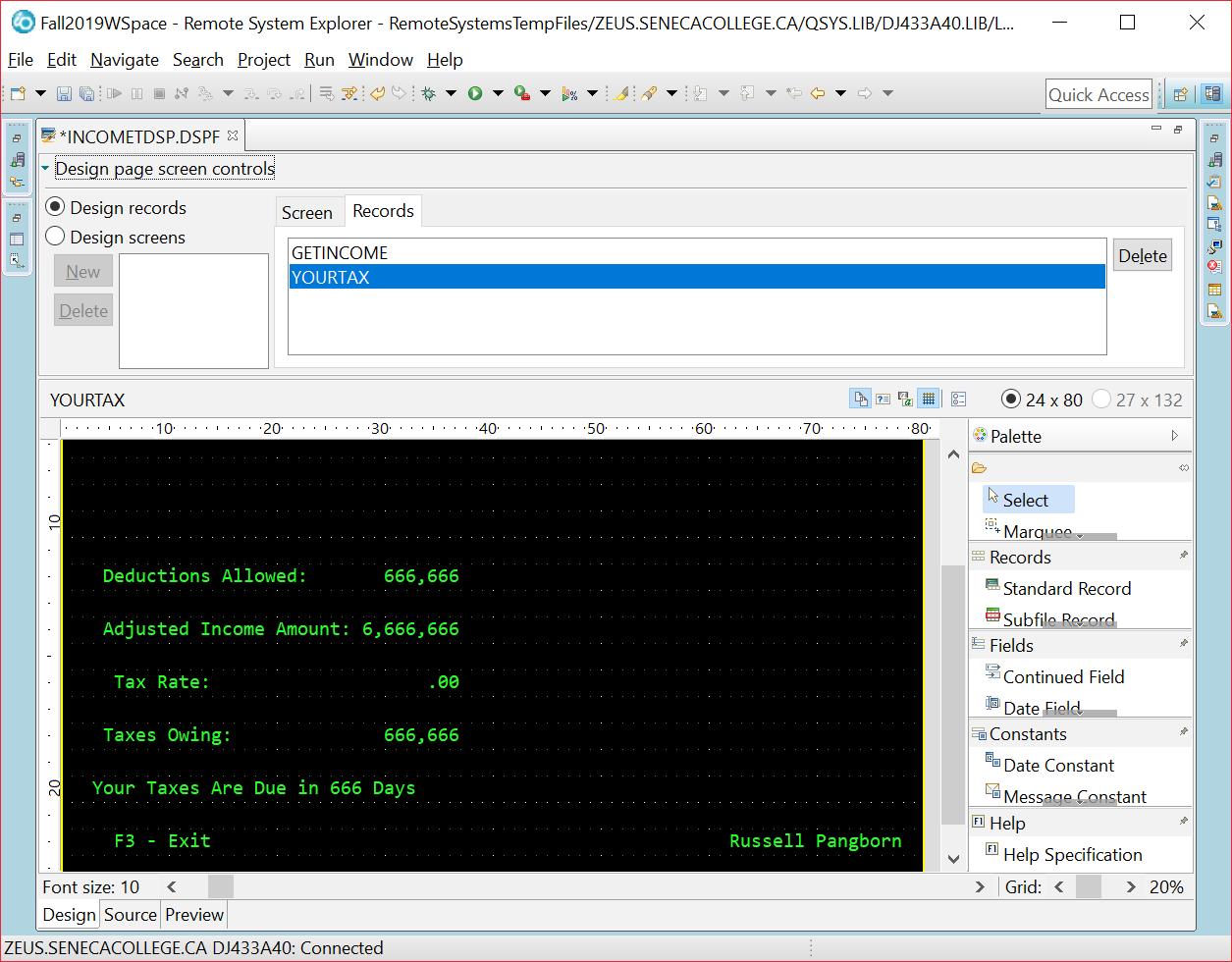
If you had messed up the current session, you don’t want the local backup. If you have done a lot of work and not messed up the current session then you do want the local backup.

You are seeing this error dialogue because while doing this lab, my internet provider caused the connection to drop causing me to have to sign in again. This is less likely to happen to you at school, but may happen at home.

You are ready to provide the second screen record that will overlay the first screen record.

Click on Standard Record in the palette and drop it onto your work area. You should get a RECORD1 with a blank work area. You can toggle between GETINCOME and RECORD1 by clicking on the appropriate name at the top box in the screen. Try it. Change RECORD1 to the record name YOURTAX

Make sure all your entries are below the area for GETINCOME (in order for OVERLAY to work properly)

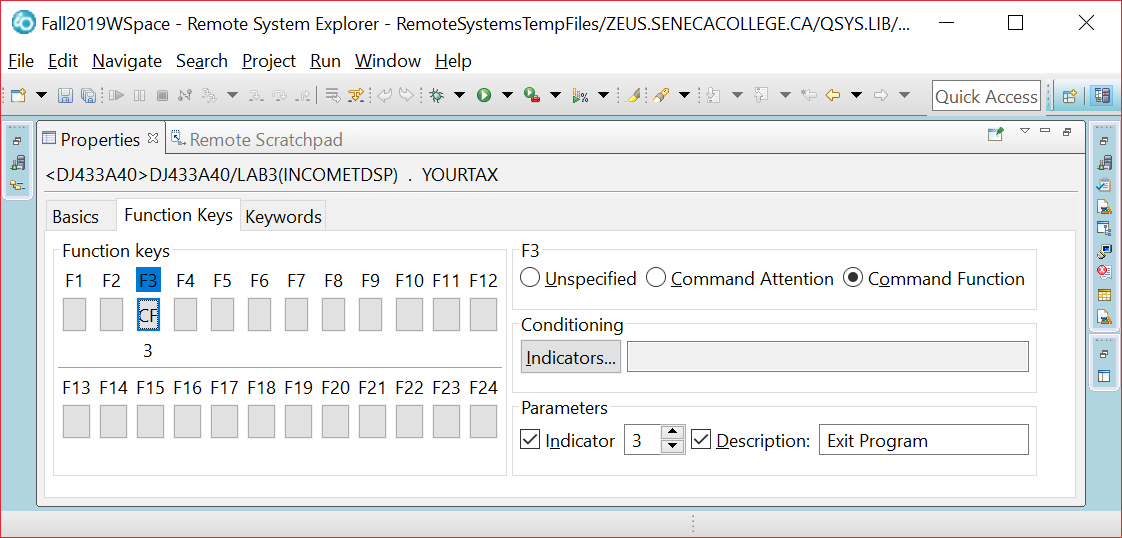


The fieldnames used for the above screen are: DEDUCTIONS, AINCOMEAMT, TAXRATE, TAXAMT and DAYSTOPAY.

**Specifying Function Keys**

We specified a constant which told the user to use F3 to exit the program. Constants are only text, they do not do anything! So, to make the F3 function key available to the user...

Making sure your focus is not on a constant or a field in the YOURTAX work area, double click on the properties view tab at the lower left of your screen and then select the Function Keys tab.



Click on F3 and on the right hand side select the Command Function radio button.

Under Parameters click on Indicator and beside that change the number to show as a three.

Click on Description and type in **Exit Program**.

Let’s review what you have done:

By clicking on F3 you have enabled the F3 key for YOURTAX. That means when the user is looking at YOURTAX they can press F3. They can not press F4 or F5 because those keys have not been enabled.

The Enter key is by default enabled for the screen records. Now, the user can press F3 or Enter. Our program will exit when F3 is pressed and reshow GETINCOME if Enter is pressed.

Command Attention means that no data is passed back to the program. If you entered data into an input capable field, that data would not be available to the program. Command Function allows the data entered on the screen to be passed back to the program. Our application could have used either the CA or CF setting because when F3 is pressed our program does not do anything with the input data entered on that final screen.

The program needs to know if F3 or Enter was pressed. You have enabled a response indicator by checking of Indicator 3 under Parameters. There are 99 indicators available to you. They are like switches with an “On” or an “Off” setting. This logical field either has a ‘1’ or a ‘0’ in it.

The EXIT Description is a comment that is placed in your code.

What was the actual DDS code for all of this? \_\_\_\_\_\_ CF03(03 'Exit Program') \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(You already know how to look this up)

Does this function key setting apply to GETINCOME and YOURTAX or just one of those records? YOURTAX

Add the OVERLAY keyword for YOURTAX by using the KEYWORDS tab when setting YOURTAX properties.

This will allow YOURTAX to display without wiping out GETINCOME. If any of YOURTAX’s fields or constants are on the same line as a GETINCOME field or constant, GETINCOME will still be wiped out.

Close your INCOMETDSP tab and save your work. So far, we only have DDS code. We do not have a Display File Object that can be used by programs.

Right click on you closed Display file member (in the Remote Systems View) and compile it.

What is the command used to compile a display file DDS member? \_\_\_\_\_\_\_\_\_\_\_\_CRTDSPF\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

You should see feedback that the “Events file contains no messages” under the Error List Tab.

Click on the Commands Log Tab. You should see dialogue similar to the following:

SBMJOB CMD(CRTDSPF SRCFILE(DJ433A40/LAB3) SRCMBR(INCOMETDSP) REPLACE(\*YES) OPTION(\*EVENTF) FILE(DJ433A40/INCOMETDSP)) JOBD(\*USRPRF)

Job 262777/DJ433A40/QDFTJOBD submitted to job queue QBATCH in library QGPL.

CRTDSPF SRCFILE(DJ433A40/LAB3) SRCMBR(INCOMETDSP) REPLACE(\*YES) OPTION(\*EVENTF) FILE(DJ433A40/INCOMETDSP)

**File INCOMETDSP created in library DJ433A40.**

If you don’t get this there may be an error introduced into the code when you were working with the Source tab.

There also may be trouble with your connection. If the **File INCOMETDSP created in library** dialogue does not appear and no syntax errors show, then close up RDi, reopen it and then try the compile.

You can also compile this code using the green screen. Sign on to a Client Access session.

Type WRKMBRPDM QDDSSRC at the command line.

(This won’t work properly if you didn’t call your source physical file QDDSSRC)

Use option 14 beside the INCOMETDSP member name.

Instead of typing WRKSPLF at the command line to view your listing, type in SP where you had typed in “14”.

Use option 5 and scroll up and down your listing or just type in “B” at the top to get to the bottom of your listing and look for the **File INCOMETDSP created in library** message.

We are ready to code our program. Here is some help with entering the code.

Although a lot of help is provided, this help assumes that you made it to class and saw your instructor enter the code. If you missed that class, you should be able to get help from a lab aid or an instructor if you are stuck on any step. If you missed both the class and the lab, you should see a BCI433 tutor.

Add a new member called **INCTAXRPG** with an RPGLE member type to your QRPGLESRC source physical file. It will contain your program code for an RPGLE program that uses the display file records you have just created.

We are going to code a free format RPGLE program. In the past students have coded fixed format file and definition specs and then coded free format statements to indicate the processing logic and order. This was acceptable to the editor we worked with in lab 1 (PDM - SEU – Program Development Manager – Source Entry Utility). It was possible to work with the code in that environment and only code errors would be flagged.

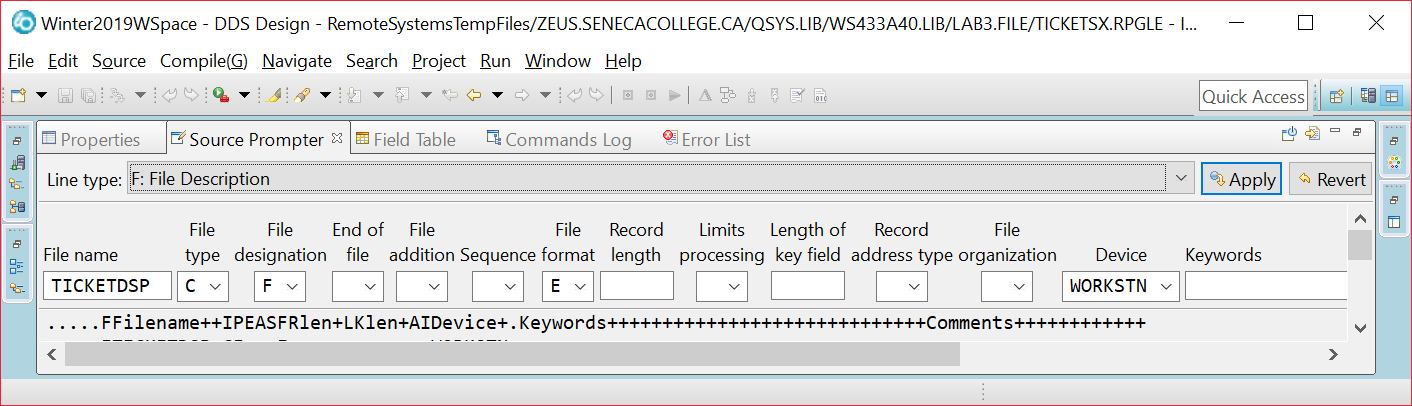
This semester we are going to code the file and definition information using free format code. If you look at this code using the source entry utility, syntactically correct free format file lines will be flagged as errors.

They are not flagged as errors when using RDi. So lets be aware of that problem and use RDi.

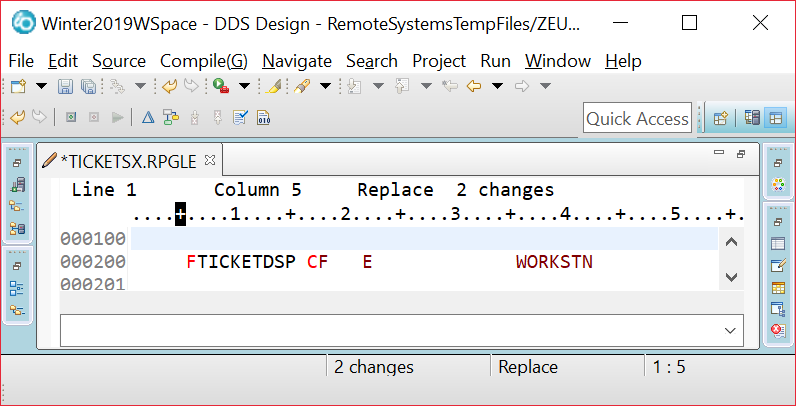
What file name are we processing? \_ **INCTAXRPG** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ . We are using a workstation device file.

Make sure you start your code anywhere after column 7.

**dcl-f filename workstn;**

This is the fixed format way to do the same thing that shows above with a workstation file called TICKETDSP. With a prompter 

and how it appears in code.



**Enter your file declaration using the easier free format technique.**

\_\_\_\_DCL-F INCOMETDSP workstn;

If we had started with a fixed format file spec would would have had to type /FREE on the next line to indicate we were going to free format RPG. The slash would have gone in column 7.

As you move your cursor, feedback should be provided about which column number you are in.

The code you need to enter for the main routine will be supplied for you. This version will not be perfect and may require a few adjustments in a later lab.

It is important to remember that each program statement needs to end with a semicolon. If you don’t do this, a syntax error may pop up when you go to the next line. The compiler will also reject an RPGLE statement that does not end with a ” ; ”

If a syntax error pops up it may not always appear immediately below the line that is missing the semicolon. Figure out where you need to make the adjustment and then press CTRL + F5 to get rid of the message that shows up in the LPEX editor.

Common PDM and LPEX editor commands:

A line can be deleted by typing a D in the first position of the line number and pressing enter.

A line can also be copied by typing a C at the line and a B or A at the line it is to be copied before or after

Multiple lines can be deleted by typing DD at the start of the block and DD at the end of the block

CC and MM also work with a block of lines. There are more modern ways of editing available with the LPEX editor.

Here is the code you need to enter.

**EXFMT GETINCOME;**

**DOW NOT(\*IN03); //DOW \*IN03 = \*OFF**

**IF IncomeAmt = 0;**

**\*IN98 = \*ON ; // Zero Income Amount**

**EXFMT GetIncome;**

**\*IN98 = \*OFF;**

**ITER; // GO TO THE ENDDO FOR NEXT LOOP ITERATION**

**ELSEIF Married = ' ';**

**?**

**ELSE;**

**EXSR GetTaxRate;**

**DEDUCTIONS = STDEXPENSE + DONATIONS;**

**AINCOMEAMT = (INCOMEAMT - DEDUCTIONS);**

**ENDIF;**

**DAYSTOPAY = 5;**

**TAXAMT = 1000;**

**// PROTECT FIRST SCREEN RECORD FIELDS**

**// REDISPLAY FIRST SCREEN RECORD AND THEN OVERLAY SECOND RECORD**

**\*IN60 = \*ON;**

**WRITE GETINCOME;**

**EXFMT YOURTAX;**

**\*IN60 = ‘0’;**

**IF \*IN03=\*OFF;**

**EXSR CLEAR;**

**EXFMT GetIncome;**

**ENDIF;**

**ENDDO;**

**\*INLR = \*ON;**

**RETURN;**

EXFMT is a Read/Write operation

A write of the screen record to the display station and a pause while the user reads it. The user may fill in some fields if there are input capable fields on the screen. At minimum the user will press the return key. A read back to the program occurs and the next program statement is executed.

DOW / ENDDO is for looping. The test is done at the start of the loop and if the test is not passed, none of the loop statements are executed.

NOT(\*IN03) is a test of the response indicator 03 which was specified when developing INCOMETDSP.

Where was that specified? \_\_\_\_\_\_\_\_In display file:… … CF03(03 ‘EXIT’)

You could have also said DOW (\*IN03 = ‘0’) or DOW (\*IN03 = \*OFF)

EXSR GetTaxRate is sending control down to a subroutine called GetTaxRate. This subroutine needs to appear after the RETURN statement and will determine the contents of YOURTAX output only fields.

Here is some sample code to use for this subroutine:

BEGSR GetTaxRate;

TaxRate = .10;

ENDSR;

Notice that all those fields have not been determined properly in the code above. You can demonstrate your program with this stub code for this lab and the next lab will include the proper demonstration of income tax rates according to a chart of values.

Compile IncTaxRPG.

Fix any errors

Switch to your Access Client session and run INCTAXRPG. This is done by typing

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_CALL INCTAXRPG\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ at the command line.

Enter values for all the input fields and press enter.

When your results screen shows, can you still type data into the top screen fields? \_\_\_\_\_\_\_NO\_\_\_\_

Make sure you can exit your program by pressing the F3 key. A previous lab showed what do to if you can’t stop your program.

What do you need to do to stop this program if it won’t end?

1.Sift +Esc then 2 + Enter

2.WRKACTJOB

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Instructor sample programs are available to check yours against. They have reverse image text to self identify as an instructor program. There is IncTaxRPGA allowing you to check how your IncTaxRPG program should behave after doing LAB3A (showing screens and doing validation). Next week you will compare your IncTaxRPG program to the instructor program IncTaxRPGB which correctly calculates income tax rates.

To run the first sample program, change your current library to BCI433LIB and then invoke the program:

CHGCURLIB BCI433LIB

CALL INCTAXRPGA

CHGCURLIB DM434C26

After you have finished, you should reset your current library back to your own library.

What happens if you do not change the current library when running the instructor’s version of this program?

It will depend on the existence of IncomeTDsp in your library.

Try running this program with your library as the current library after you have successfully created your version of INCOMETDSP.

What happened?

YOUR own program should run

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Lab 3 Summary

A display file is coded with DDS code in a member that has a DSPF type.

Once this DDS code has been successfully compiled a Display File object is produced that can be used by different programming languages that want to take advantage of the interactive screens allowing data entry and data display.

The easiest way to define how the display file screens or records will look is to use the Screen Designer GUI in RDi.

The GUI allows you to define field names, types, usage, editing, validations and optional indicators that allow viewing of the field.

You would use the GUI to provide this information and the appropriate line is entered as DDS code.

Here is a sample line with explanations

This is how it actually appears



Some separation is shown here to better highlight concepts – the line would not be entered this way.

Field Field Data Decimal Line Column

Indicator Name Length Type Positions Use Number Position Functions

A 90 TEST1 3 S 0 B 5 32 RANGE(0 100)

Indicator 90 has to be on in order to view this field called TEST1 on the screen. If indicator 90 is off, the TEST1 field will not be visible when the screen record is displayed. For our application it would not be a good idea to use an indicator with the TEST1 field.

L shows a field length of three. So at this point XXX could be entered

T or Type shows a field type of zoned decimal and D or Decimal shows 0 decimal positions. So now we cannot put X’s in the field. 999 would work.

U or Usage indicates if this field is for input only (I) , output only (O) or both input and output (B).

P or position on the screen shows that this field would reside on line 5 at column 32.

The RANGE function allows us to validate the field before it is passed to a program. The user can enter 0 to 100. They cannot enter a negative number or 101.

Here is some more partial DDS code with explanations

A R RECORD2

A CA03(03 'EXIT')

A OVERLAY

A 11 18'Tests:'

A 13 18'Final Mark:'

A 15 18'Final Grade:'

A 22 3'F3=Exit'

A TESTOVRALL 3Y 0O 11 32EDTCDE(1)

A NUMGRADE 3Y 0O 13 32EDTCDE(1)

A LETGRADE 2A O 15 33

This screen record is called RECORD2 and has some text and fields that are displayed.

At the record level we see CA03 – that enables the pressing of function key F3 when the screen record is displayed

(03 ‘Exit’) indicates when F3 is pressed indicator 03 will be turned on and the comment is ‘Exit’

OVERLAY is used to allow RECORD2 to overlay a screen record that is already being displayed. This only works of the first screen record does not use lines 11 – 25. You can see RECORD2 is using lines 11 to 22.

EDTCDE(1) can be used to make a numeric field show with commas, a decimal point and suppressed leading zeros

So 090 would show as 90. If the field was larger 0293334^33 would show as 293,334.33. The decimal point is not stored in the field but can be shown with proper editing. We used the “^” symbol to show where the placeholder for a decimal point is.

RPGLE

A Display file is declared in an RPGLE program with a DCL-f statement

Dcl-F HWYTOLLDSP Workstn;

EXFMT – write a screen record from a display file, pause and when the user presses enter read back what has been inputted into the screen record fields. Sometimes there are no fields and the read back just acknowledges that the screen has been viewed and the user is letting the rest of the program proceed.

DOW - a loop where the test on entering the loop or repeating the loop is done at the start of the loop.

ENDDO

DOU – the loop code is executed at least once and the test on repeating the loop is done at the bottom.

WRITE – can be used to have a screen record display. There is no pause and the program continues. This is useful when showing a screen record and then overlaying a second record and then pausing to let the user look at both screen records.

EXSR GETGRADE – control goes from here to a subroutine at the bottom of the program. The named subroutine has a BEGSR and ENDSR to indicate what code is executed and then control goes back up and executes the next line after the EXSR line.

\*IN01 - \*IN99 – in RPGLE indicators are referred to by an asterisk and a number. There are 99 indicators available for use and their default setting is \*OFF or ‘0’ . They can be turned on during the program run and then checked to see if they are on or off to determine a course of action.

\*INLR = \*ON – these two statements are how you should end all your RPGLE programs.

RETURN The last record indicator is turned \*ON and RETURN is used to return control to the operating system.