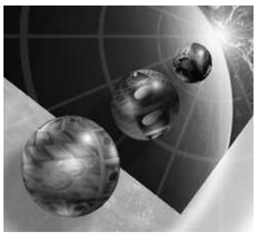


IBM WebSphere Development Tools for iSeries

VisualAge RPG
GUI subfile programming

Student Exercises

Course code 23LF
Session Id 407145

Claus Weiss
weiss@ca.ibm.com

IBM Toronto Laboratory

Version 5 Release 1 Modification0

homepage: www.ibm.com/software/ad/wdt400
www.ibm.com/software/ad/varpg
newsgroup: news://news.software.ibm.com/ibm.software.varpg

Fourth Edition (September, 2001)

Comments concerning this notebook and its usefulness for its intended purpose are welcome. You may send written comments to:

IBM Canada Software Solutions

3200 Warden Ave., Markham, Ontario, L6G 1C7

Attention: Claus Weiss, GUI subfile programming.

or email to**: weiss@ca.ibm.com**

Overall Lab Guide

The objective of the VisualAge RPG GUI subfile lab is to have the students work with the GUI subfile and get hands on experience with handling the GUI subfile. The GUI subfile is very similar to a 5250 (green screen) subfile but to exploit all the GUI feature some different programming techniques are needed. At the end of the lab, the student should know how to create a GUI subfile Visualage RPG application by creating the graphical user interface, writing RPG code to handle the interface events, and building the application.

Also, it is helpful if the student is familiar with basic Windows 95/98 operations such as working with the desktop and basic mouse operations such as opening folders and performing drag-and-drop operations.

The lab will have very detailed instructions on how to proceed. So all students should be able to proceed and finish the Lab.

For students without experience in using VARPG we have included some basic exercises to make it easier to get started, these beginner exercises are marked by a grey background in this document. We added pointers to these newcomer exercises throughout the class material.

Note: *The pictures in these labs show a similar application being built. Some of the names and icons may be different than the environment you are working with.*

Although not required, familiarity with the RPG language is preferred.

Trademarks

Using Subfiles

Working with Subfiles

In this lab you will go thru a couple of exercises that guide you from displaying all records of a file in a subfile, thru displaying the data page by page. You will also have the opportunity to use the new **READS VARPG** operation code and react to **ENTER** events from the subfile part. Handling attributes like **cell colour** and **column colour** will be part of these exercises as well. How do you enable subfile cells for input and secure cells from change? We will guide you thru an exercise concentrating on these capabilities of the GUI subfile. Dealing with the subfile **navigation bar** will show you some of the additional features of the GUI subfile. At the end of this Lab we will guide you thru a subfile exercise that exploits the extended selection feature of the GUI subfile and we will show you how to display data in a subfile resorted by a column selected at runtime.
During this first part of the lab you will:

- Create a new VARPG application
- Create a subfile with reference fields
- Write the VisualAge RPG logic to read records from an AS/400 database file and write the data to the subfile part
- Build and run the application

Creating a new application

In this first exercise, you will create a VisualAge RPG application that displays a list of customers in a subfile.

To begin, start the GUI designer if it is not already running. If it is running, choose ***Project→New*** from the menu bar to open a new project.

Starting the GUI Designer

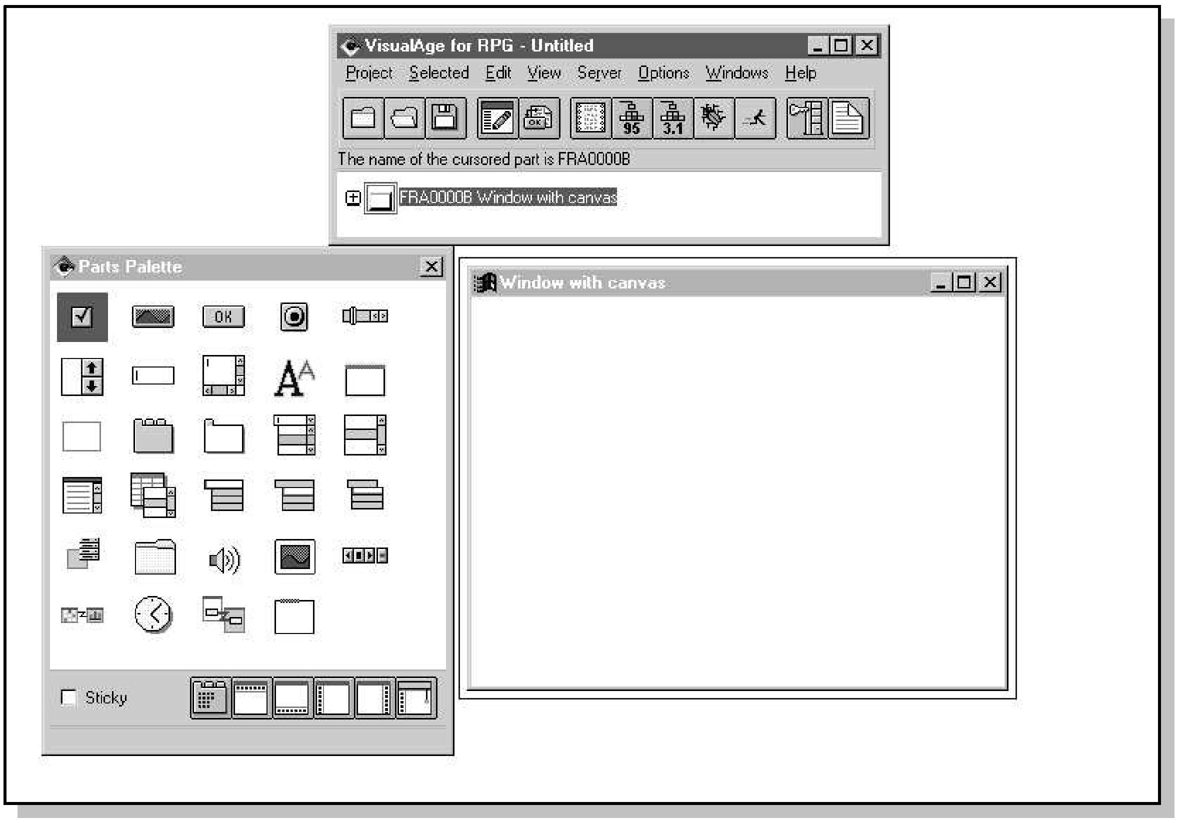
To start the GUI designer for the first time:
Click on the **VisualAge RPG projects** folder on your desktop and double click on the **New GUI project** icon.

Or

Press the **Start** button on the task bar and choose ***Start→Programs→IBM WebSphere Development Tools for AS/400→ Visualage RPG→GUI Designer***.

A status window indicating the progress in the initialization process of the **GUI designer** will appear

The GUI designer including the **parts palette** and the **first design window** appears.
Rearrange the windows if necessary so they are all visible.
Resize the **parts palette** so that all parts are visible.



The name of the cursored part is FRA0000B

FRA0000B Window with canvas

If you have not had a chance to work with the VARPG GUI designer we suggest to read through the following paragraphs to make yourself familiar. On the other hand if you already have worked with the GUI designer just skip to the header ***Adding a subfile part to the window*** that follows after the greyed out portions of text.

GUI designer Components

The VisualAge RPG GUI designer consists of these major components:

- The Menu Bar
- The Tool Bar
- The Project View
- The Parts Palette

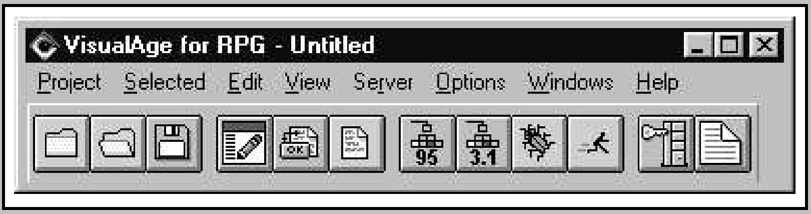
The Menu Bar

The **Menu Bar** is located just below the title bar of the GUI designer window. It provides a variety of tasks you can use to create and modify your GUI application, to customize the GUI designer, and to get online help.

Tip: To get additional information on a particular menu item, press F1 while the desired menu bar item is highlighted. This will display a help window with detailed information on the selected menu item.

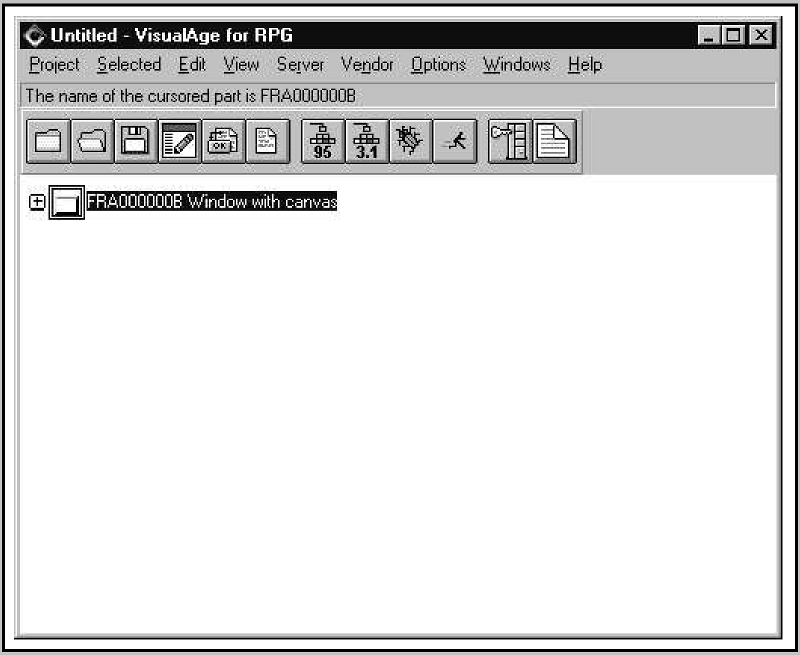
The Tool Bar

The **tool bar** is a menu of icons. It provides a fast path access to many of the menu items on the menu bar. Move the mouse pointer over the icons of the tool bar. A short description of the icons function is displayed in the form of 'flyover' help next to the pointer**.**


VisualAge for RPG - Untitled
Project Selected Edit View Server Options Windows Help
95
3.1

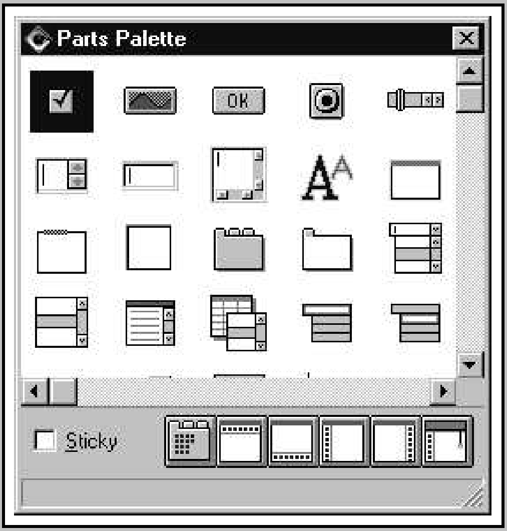
The Project View

The **Project View** is used to hold all the parts you will create during your GUI designer session. The project view can be changed to display the parts of a project in different views.


Untitled - VisualAge for RPG
Project Selected Edit View Server Vendor Options Windows Help
The name of the cursored part is FRA000000B
95
3.1
FRA000000B Window with canvas

The Parts Palette

The **parts palette** contains parts that you will use to create the graphical user interface (GUI) of your application.



1. Move the mouse pointer over the parts on the **parts palette.** Notice the information area at the bottom of the parts palette. It displays a short description of the part. Closing Windows

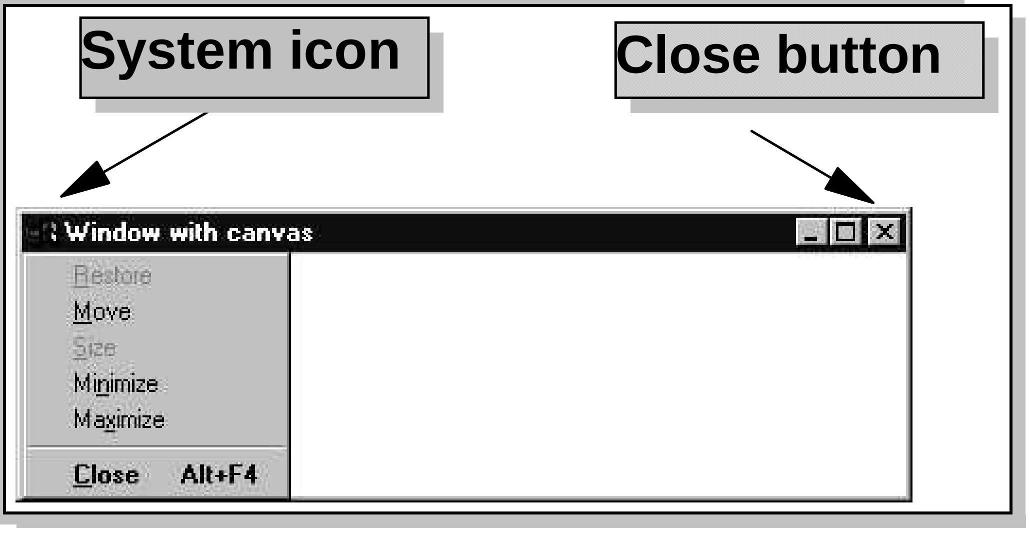
Several times throughout this lab you will be instructed to close a design window. In all there are four ways to close a design window. You can use which ever method you prefer:

2. Double click on the system icon in the upper left corner of the window

3. Single click on the system icon in the upper left corner of the window. From the menu choose ***Close***.

4. Single click on the **close** button. This is the **X** icon in the upper right corner of the window

5. Pressing **Alt+F4** will close the window that currently has focus System icon



Drag-and-Drop

One method of moving objects in **VARPG** is by the way of dragging an object from one place and dropping it to another. This is referred to as drag and drop. To drag and drop an object click and hold the left mouse button with the mouse pointer over the object then move the mouse pointer so the object is at the desired location and release the left mouse button.

Point and Click

Point and click provides an alternative method to drag-and drop, for creating the user interface. Select a part on the parts palette by single clicking on it. Move the mouse pointer to the new location and click the left mouse button to release the part at that location.

Point-and-click is the preferred method of creating parts in VARPG and is the default.
This method will be used during the labs.

Pop-up Menus

Pop-up menus are provided for practically every object and area in the GUI designer allowing you to perform some action on that object or area. To invoke a pop-up menu position the cursor over the object or area and click and release the right mouse button. To choose a menu item from a pop-up menu single click on the menu item.

A Note About Notebooks

As mentioned earlier the parts palette contains parts that act as templates. Once the part has been created, its attributes such as its part name, can be changed from their default properties. These changes are made by invoking the parts **properties notebook.**

There are three methods of opening a properties notebook:
1) Double click on the part,
or
2) right mouse click on the part and choose ***Properties*** from the pop up menu
or
3) select the part and from the Project window menu bar, choose ***Selected -->Properties***.

A properties notebook consists of **pages** as indicated by the labelled **Tabs**. You go to a notebook page by clicking with the left mouse button on its tab. Once you made your changes, press the **OK** push button at the bottom of the properties notebook.

Let's change some of the properties for the initial design window:

1. Double click on the **title bar** of the design window (see the tip below). The window part **properties notebook** appears.

Tip: *A **Window with canvas** part is made up of two parts. The Window, which consists of the title bar and frame, and the Canvas which is the white area enclosed by the window. Therefore, if you were to double click on the Canvas, you would get the properties notebook. for the **Canvas** part.*

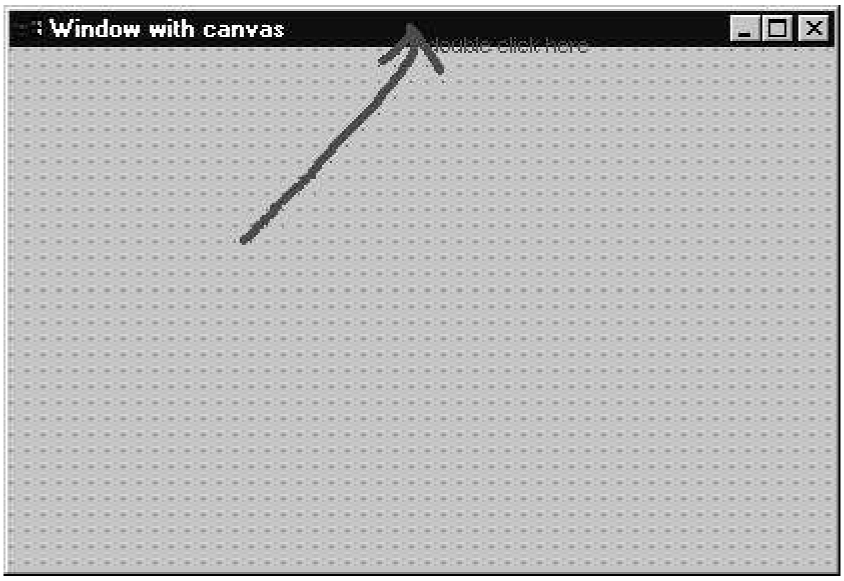
Adding a subfile part to the window

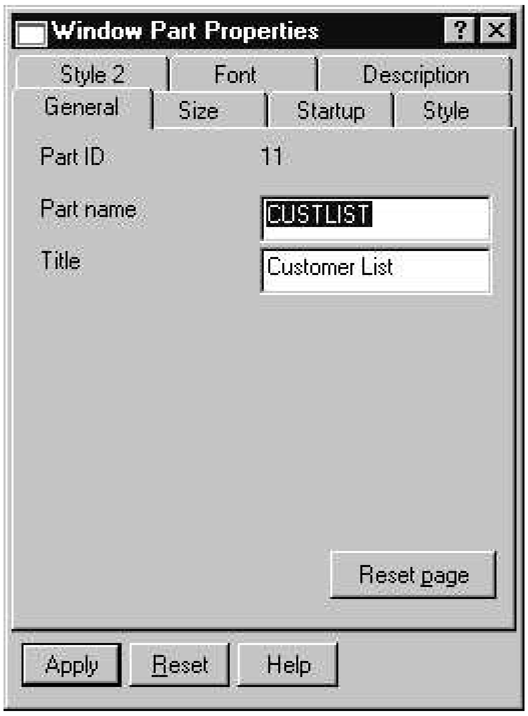
You will add a subfile part that will display records from a logical file on the AS/400. This file is named **CUSTOML3**, and will be in library **VARLABxx**. The record format name is **CUSTOM01**. The subfile you will create is supposed to display data located in the following fields:

CUSTNO
CUSTNA
CCITY
CADDR
CCOUNT

Perform these steps to change the design windows name and title, and add the subfile part to its canvas:

1. Double click on the **title bar** of the design window (see the tip below).





Part ID 11
Part name CUSTLIST
Title Customer List

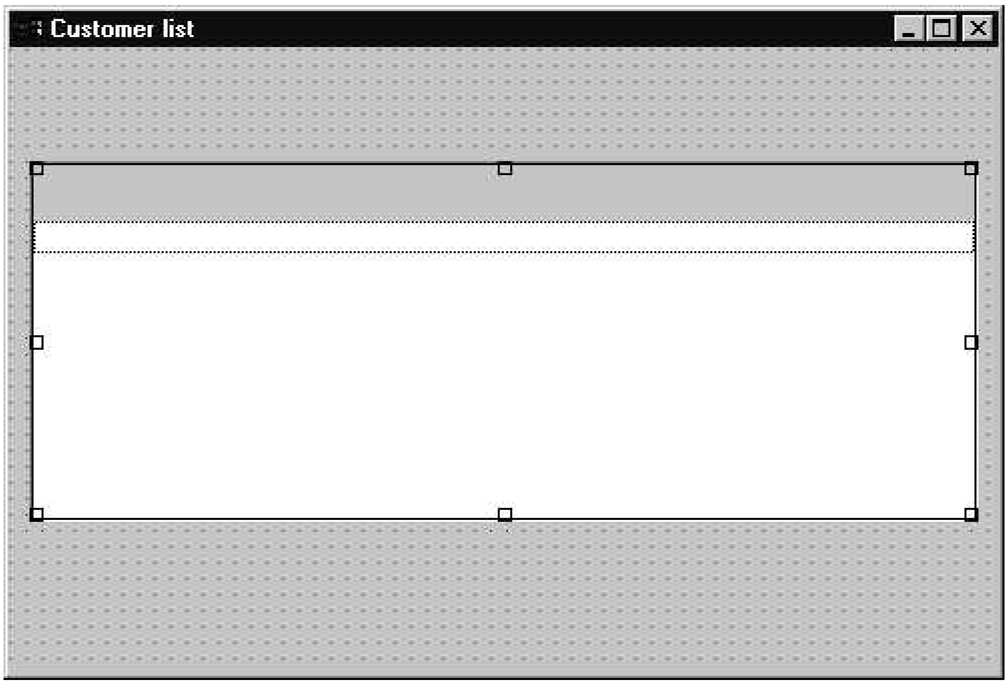
2. The **window part properties notebook** appears

3. Change the title of the design window to ***Customer list*** (or something equivalent).

4. Name the design window ***CUSTLIST***.

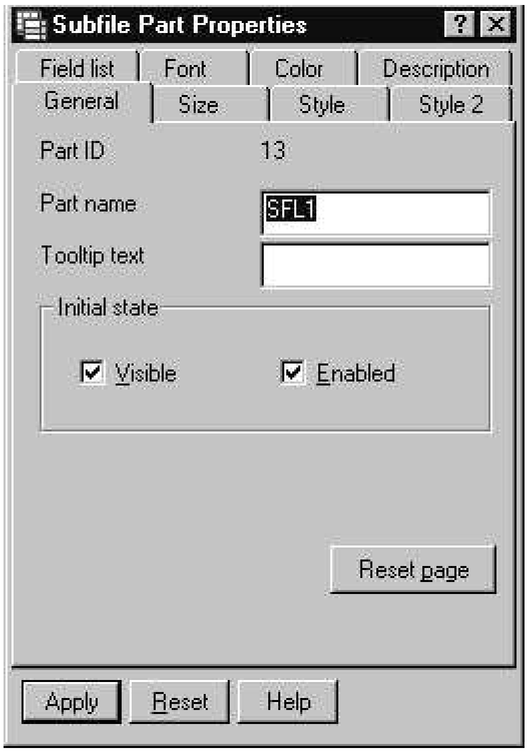
5. Locate the **Subfile** part on the parts palette and use **point-and-click** to create a subfile on the design window. (Point at the subfile icon on the parts palette then click on the Window canvas with your mouse cursor)

6. Position and size the **subfile** as required on the window canvas.



7. **Double click** on the subfile part

8. **Change** the name of the subfile part to ***SFL1*** and click on the Right upper corner **X** of the Properties notebook.



Part ID 13
Part name SFL1
Tooltip text
Visible ✓
Enabled ✓

Now you will add fields to the subfile. There are three ways to add fields to a subfile part:

a. by **creating entry fields** from the parts palette.
b. by **opening the Subfile parts properties notebook** and going to the **Field list** page.
c. by **creating** fields from a **reference file**.

For this exercise, you will use the **Define Reference Fields** method.

Using an AS/400 File as a Field Reference File

During this exercise you will experience how to add fields to subfile **SFL1** using an AS/400 file as a **Field Reference File** by using the **Define Reference Field** function of VisualAge RPG.

Starting with identifying the AS/400 to use

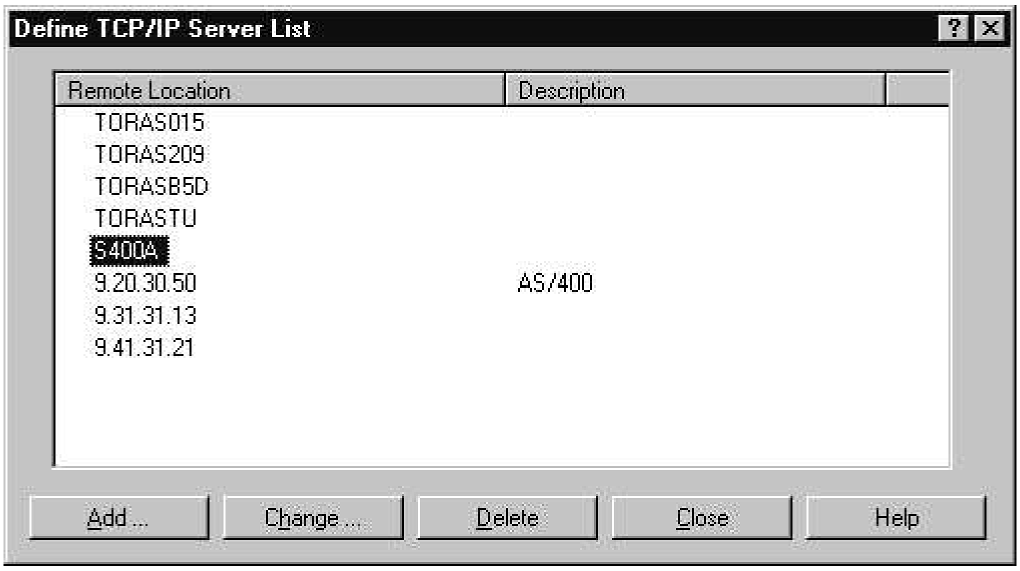
First you need to make sure to tell **VARPG** which **AS/400** system your reference file is located at. Please follow the steps below to specify the connection information.

Defining the AS/400 server

You are using TCP/IP to connect to the AS/400, just define the AS/400's **IP address** or **HOSTNAME** for the VARPG serverlist. This step is only required once and might have been already performed by another group in a previous class on this workstation.

To define the AS/400 in the **Server List**, choose ***Server→Define TCP/IP Servers***.

1. Check the list of **AS/400's** already defined,
2. **if** your AS/400 server is **already defined** just press **CLOSE** and proceed to **Defining database reference fields**,



Define TCP/IP Server List

Remote Location	Description
TORAS015	
TORAS209	
TORASB5D	
TORASTU	
S400A	
9.20.30.50	AS/400
9.31.31.13	
9.41.31.21	

3. otherwise continue here
4. On the **Define TCP/IP Server List** dialogue, press **Add...** to configure the AS/400.

Your instructor will give you the correct **AS/400 Hostname** or **IP-Address**

5. Press the **Ok** button

6. Press the **Close** button to leave the dialog

Defining userid and password

Now define the **userid** and **password** you will be using during this LAB to sign on to the AS/400.

Defining userid and password

To define the AS/400 user information,

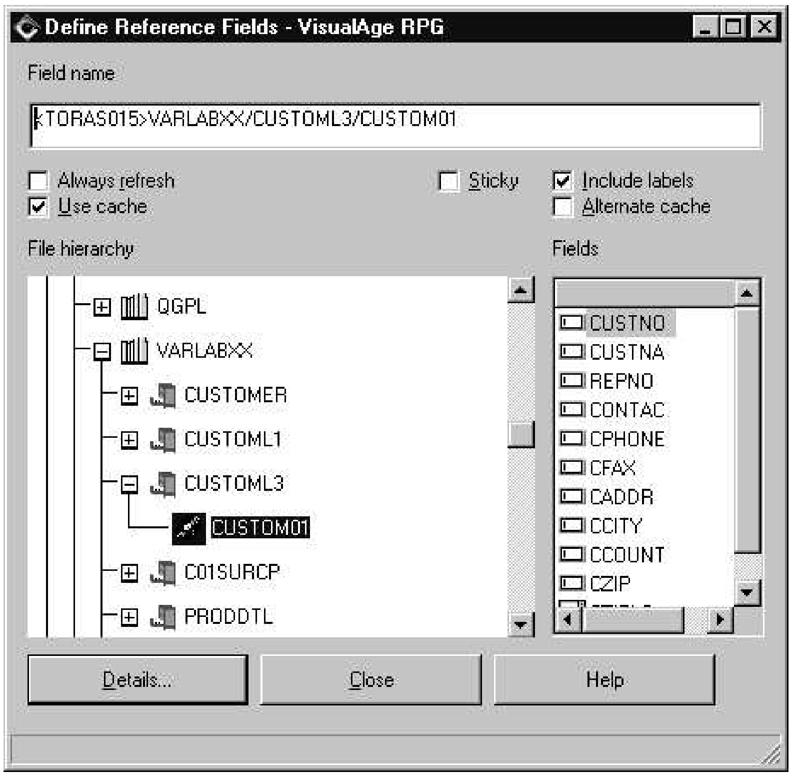
- choose ***Server→Define Server Logon***.
  When the **Define Server Logon** dialog appears
- press **Add...** to add the userid **WDTLABxx** (if your team number is **50**, the userid would be **WDTLAB50**) and the **password** (which is the **same as the userid**), notice that the password has to be specified **twice**.
- Press the **Ok** button to exit.
- Press **Close** to leave the dialog

Now you are ready for **specifying your reference fields** on the AS/400.

Defining data base reference fields

1. Choose ***Server→Define reference fields*** from the GUI designer menu bar to display the **Define Reference Field** dialog
2. If the **logon** is successful, the libraries of your **initial library list** are displayed ***(*LIBL***).
3. Find the library ***VARLABxx*** in this list and **expand** it by clicking on the '**+**' sign. The logical and physical files in this library are displayed. If the library ***VARLABxx*** does not appear in the list of libraries, it is not in your library list. In that case, type ***VARLABxx*** in the entry field at the top of this window next to the server name, and press enter. (e.g. **<S400A>VARLABxx**)
4. **Click** on the **'+'** next to **CUSTOML3**. A list of the record formats for this file will appear.
5. Double click on the only available format name **CUSTOM01**. All fields in this record format are shown in the **Fields** list on the right side of the

Dialog.



Define Reference Fields - VisualAge RPG

Field name

TORAS015>VARLABXX/CUSTOML3/CUSTOM01

☐ Always refresh
☑ Use cache
☐ Sticky
☑ Include labels
☐ Alternate cache

File hierarchy

```
QGPL
VARLABXX
  CUSTOMER
  CUSTOML1
  CUSTOML3
    CUSTOM01
  C01SURCP
  PRODDTL
```

Fields

```
CUSTNO
CUSTNA
REPNO
CONTAC
CPHONE
CFAX
CADDR
CCITY
CCOUNT
CZIP
```

You will now get the necessary fields for the **subfile** by using some of the fields of the **CUSTOM01** record format as references:

1. Rearrange the **Customer List** window with the subfile and the **VisualAge RPG - Define Reference Field** window so that you can see them both.

2. Move the mouse pointer over the **CUSTNO** entry in the **Fields** list.

3. Point and click on the **CUSTNO** entry field, and then point and click on the **subfile** in the **Customer List** window where you want to place the **CUSTNO** field, the **field** plus **heading** containing the header text will be created in the subfile.

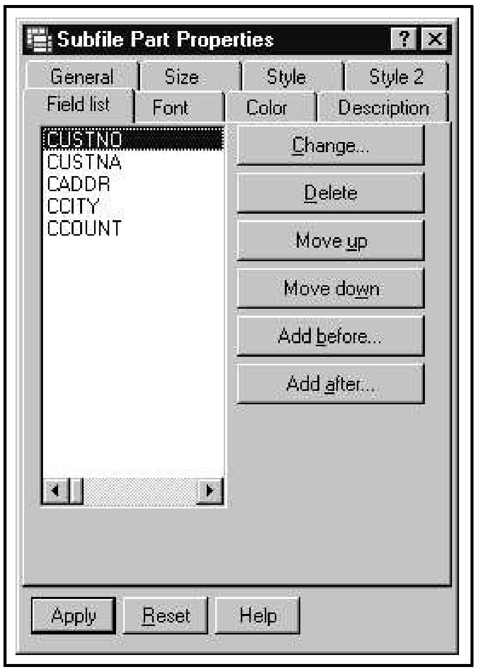
The Header will be automatically created with the field if the COLHDG DDS keyword is defined and the **Include Labels** option is checked on the define reference field dialog.

TIP: Several fields can be created from the Define Reference Field window at one time by selecting each field that is required with the **CTRL key** pressed, and then using **point and click** to move the fields to the design window. The point and click icon will indicate several fields are being created.

CUSTOML3 in library **VARLABxx**. Use **point-and-click** to add the remaining fields to your **subfile** part:

- ***CUSTNA***
- ***CADDR***
- ***CCITY***
- ***CCOUNT***

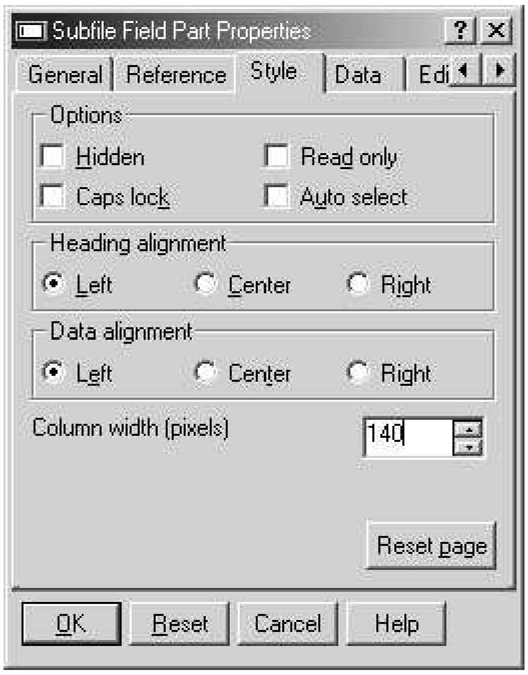
If you want to **change** the order of the fields in the subfile, open the properties notebook for the subfile, go to the **Fields** page and use the **Move up** and **Move down** push buttons.



To increase the width of the Customer Number field,

1. **Double click** on the **subfile** part, that will bring up the properties dialog.
2. **Select** the **Field List** tab
3. Select the ***CUSTNO*** field from the field list and **press** the **change button**

4. From the Subfile **Field Part Properties** notebook, select the **Style** tab
5. **Change** the **column width** to 140

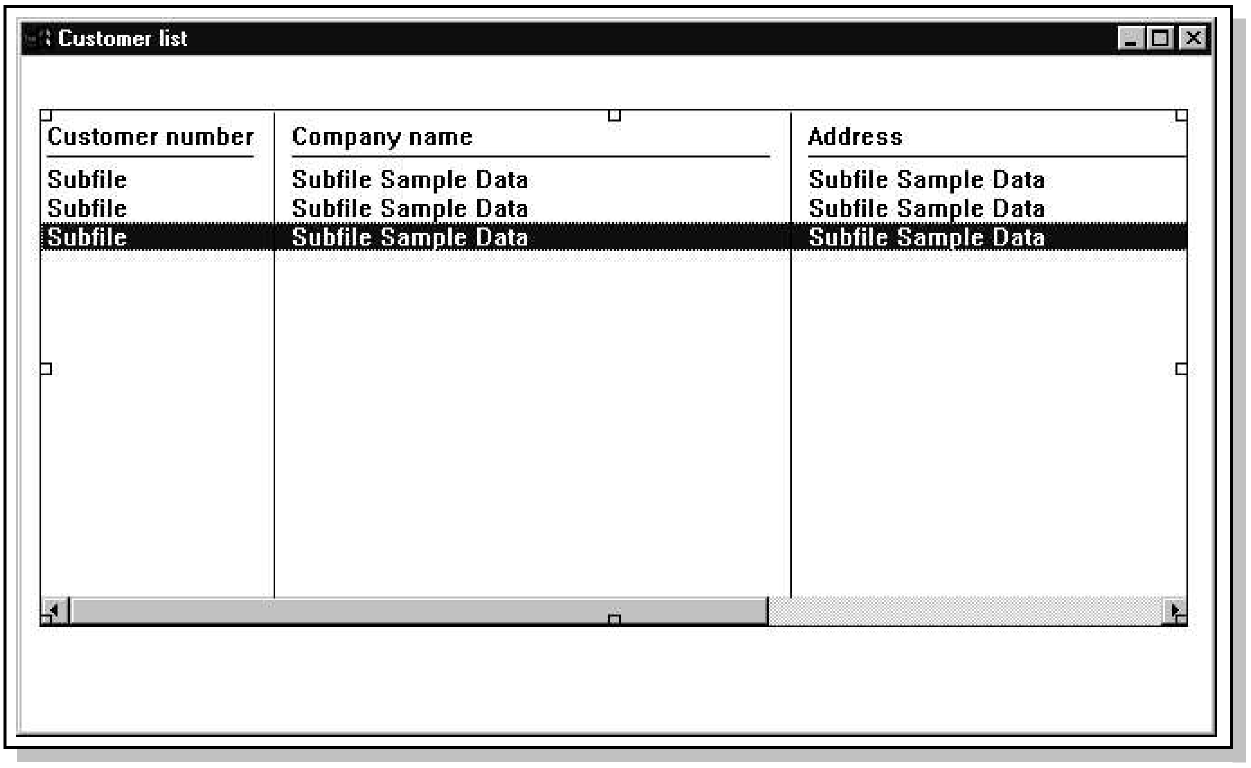


Heading alignment: Left
Data alignment: Left
Column width (pixels) 140

6. Press the **Ok** push button

Tip: You could also choose to split the heading into 2 lines to avoid stretching this column too much. You would do this on the **General Page** of the Subfile Field Part Properties Notebook**.**

Once you have **completed** the **previous steps**, you should have a design window that looks similar to the following:



Adding the Application Logic

Creating an Action Subroutine

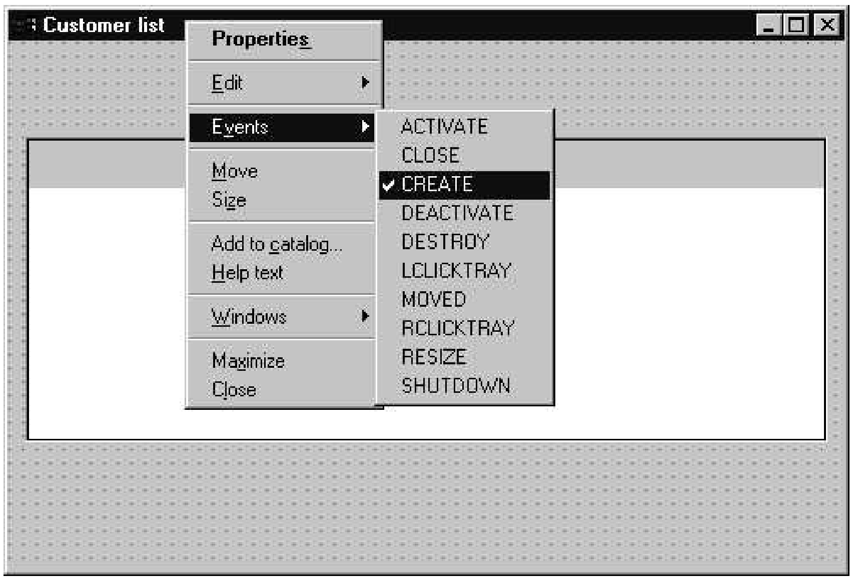
Now that you have finished designing the first window of your project, you will add some RPG logic to give it some functionality. An action subroutine is very similar to a "normal" subroutine, but the invocation is different. Instead of an **EXSR** operation code you would use to invoke a user subroutine, you will use an **Event** from the GUI to invoke an action subroutine.
In this case we want to fill the subfile with data when our Window gets displayed as part of our application startup, so we use the **CREATE** event of the ***CUSTLIST*** **window** to fill the subfile. This event gets send by the window when it gets created, which happens in this application at startup.

1. If the ***CUSTLIST*** design window is not open in the GUI designer, open it by **double-clicking** on its icon in the GUI designer **Project View**.

2. Select the **window** part (**the blue title bar**) using the left mouse button.

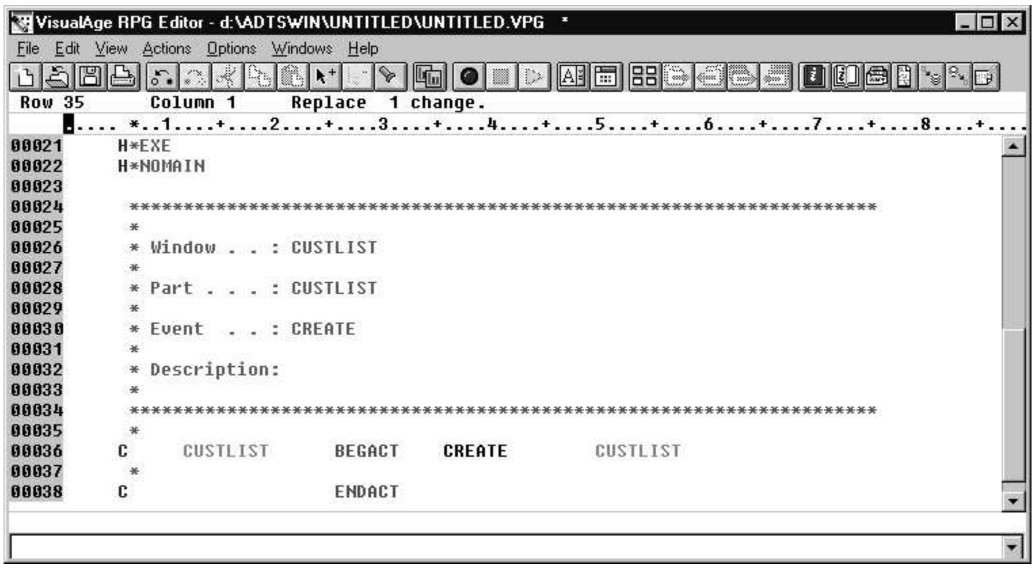
3. Now **click** the **right** mouse button on (**the blue title bar**) to get this parts **pop up** menu.

4. Choose ***Events→Create***



5. This will start the LPEX editor.

   When the LPEX editor first appears, you will notice that the **BEGACT** and **ENDACT** statements have already been inserted into the source. As well, there are a number of comment lines included to encourage documentation. You may change the comments to include any further information that you might deem necessary.


VisualAge RPG Editor - d:\ADTSWIN\UNTITLED\UNTITLED.VPG *
File Edit View Actions Options Windows Help
Row 35 Column 1 Replace 1 change.

```
     H*EXE
     H*NOMAIN

      *********************************************************************
      *
      * Window . . : CUSTLIST
      *
      * Part . . . : CUSTLIST
      *
      * Event  . . : CREATE
      *
      * Description:
      *
      *********************************************************************
      *
     C     CUSTLIST      BEGACT    CREATE        CUSTLIST
      *
     C                   ENDACT
```

Although the framework for the RPG code has been built, the application logic is not there. It must be added. In this example, you are going to fill the subfile with data when the window is created.

When the template action subroutine is displayed in the editor, you could add all logic needed to fill the subfile directly between the **BEGACT** and **ENDACT** statements.
Since we want you to add some more functionality to this application later on, we suggest to put the code in a user subroutine and just invoke the user subroutine ***FILLSUB*** here in this **action** subroutine:

```
CL0N01Factor1+++++++Opcode(E)+Factor2+++++++Result++++++++Len++D+HiLoEq
 * Fill subfile with records from database
C                   EXSR      FillSub
```

Subroutine ***FillSub*** will contain the code to **read** from the database file and **write** to the subfile until the **end of file** is reached when indicator **99** is set to ***ON**.

Add the user subroutine *FILLSUB* at the end of your program after the **ENDACT** statement of your action subroutine.
Just press the **ENTER** key to **add** lines to your source file, the next figure shows the code to add.

```
CL0N01Factor1+++++++Opcode(E)+Factor2+++++++Result++++++++Len++D+HiLoEq
C     FillSub       BEGSR
 * Read a record from database
C                   Read      CUSTOML3                               99
 * Do until end of file
C                   Dow       *IN99=*OFF
 * Add a record to the subfile
C                   Write     SFL1
 * Read a record from database
C                   Read      CUSTOML3                               99
C                   EndDo
C                   ENDSR
```

Now, you must add a **File** specification to define the database file that will be used to fill the subfile.

Adding the File Specification

1. File specification must come before any **C** specifications and after any **H** specifications, so scroll to the top of your program source.

2. Enter the following File specification for the ***CUSTOML3*** logical file:

```
FFilename++IPEASFRlen+LKlen+AIDevice+.Keywords+++++++++++++++++
FCustomL3  IF   E           K DISK    REMOTE
```

The following values were used to define this '**F**' specification:

Customl3	- The file name used by the program
I	- File is Input only
F	- File is being processed as Full-Procedural
E	- The file is Externally described
K	- The file is being processed by Key
DISK	- This is a Disk file
REMOTE	- The file resides on an AS/400 server

Tip: *You might want to add the file keyword **BLOCK(*YES)** in applications that use SETTLL or CHAIN. This will allow the RPG compiler to define blocking to access the file even when these operation codes are used*

Defining AS/400 Information

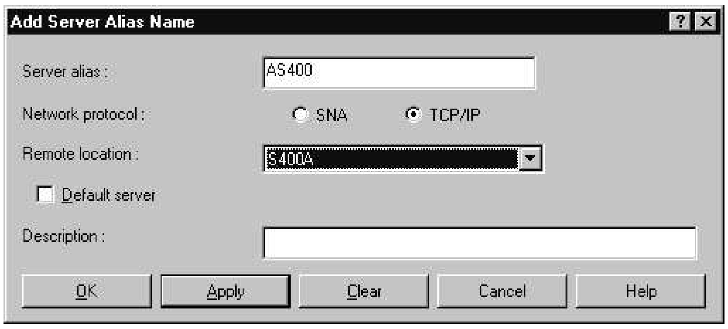
Before you can build the application, you must indicate **which file** on the AS/400 is being used, and on **which AS/400 server**, the file resides.
Do this by entering information in the **Define AS/400 information** notebook. Note that you must define this information for each VARPG project that accesses the AS/400 database.

- **Tip**: *To identify a specific AS/400 that you normally use in your environment you can specify your **default server** in the **User options dialog**. VARPG will then fill the AS/400 server name automatically for each of your projects.*

Defining the Server

1. Display the **Define AS/400 information** window by choosing ***Server→Define AS/400 information***, from the GUI designer menu bar.
2. On the server page, press the ***Add...*** push button.
3. On the ***Add Server Alias Name*** type any value for the ***Server alias*** name

4. From the ***Remote location*** combination box select the **server name** provided by the instructor.


Add Server Alias Name

Server alias : AS400

Network protocol : ○ SNA [✓ TCP/IP]

Remote location : S400A

☐ Default server

Description :

5. Press the ***OK*** push button to save your changes.

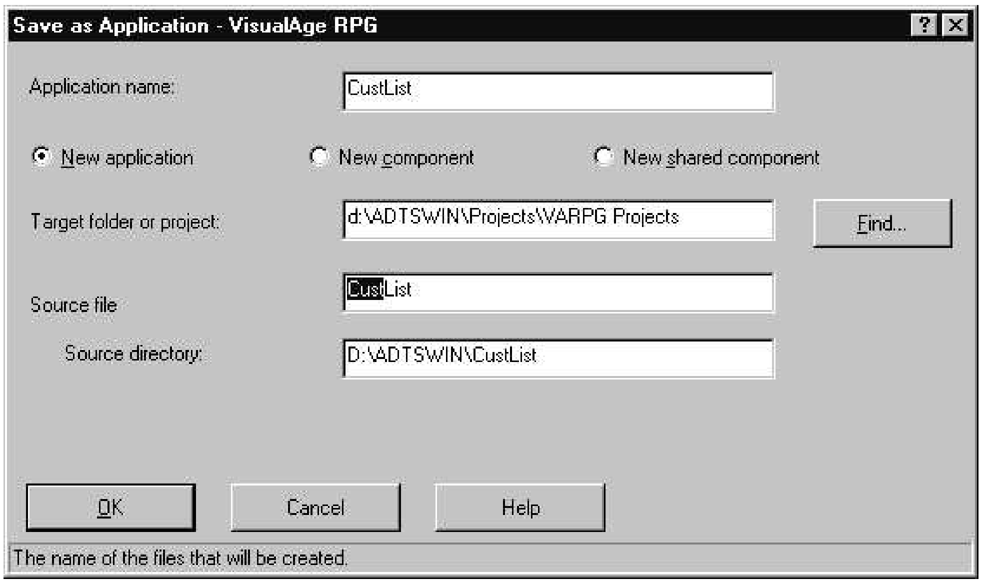
Note: You don't need to add file information since the filename on the AS/400 is the same as in your RPG program and the file resides in a library included in your library list.

Building the Subfile application

Before building this application you must first save it.

1. Choose ***Project→Save***.
2. The **Save As** window appears.
3. For the **Application Name** specify ***CUSTLIST***.

4. The **Source file and Source directory** will automatically change to the same name when you **click** in one of these fields.


Save as Application - VisualAge RPG

Application name: CustList

New application [✓] New component New shared component

Target folder or project: d:\ADTSWIN\Projects\VARPG Projects

Source file: CustList

Source directory: D:\ADTSWIN\CustList

The name of the files that will be created.

5. Press the **OK** push button to save the project.

Now build the project by choosing ***Project→Build→ Windows NT/95*** from the GUI designer menu bar.

1. A **status window** will be displayed indicating that the build is in progress.

   Once the build is complete a completion window is displayed indicating whether the build was successful.

Compile Time Errors?

If you have compile time errors, the ***Error List*** window will be displayed. The error list window shows one line for each error found during the compile. Notice that the error number (e.g. RNF7030E) is followed by a character indicating the severity of the error.

I type errors are informational and can usually be ignored.

E errors must be fixed for a successful compile.

To locate the line of code causing a specific error, **double click** on an error line in the Error List window and the **editor** will **position your RPG source** to the offending statement.

Correct any errors and recompile until you have a **successful build**.

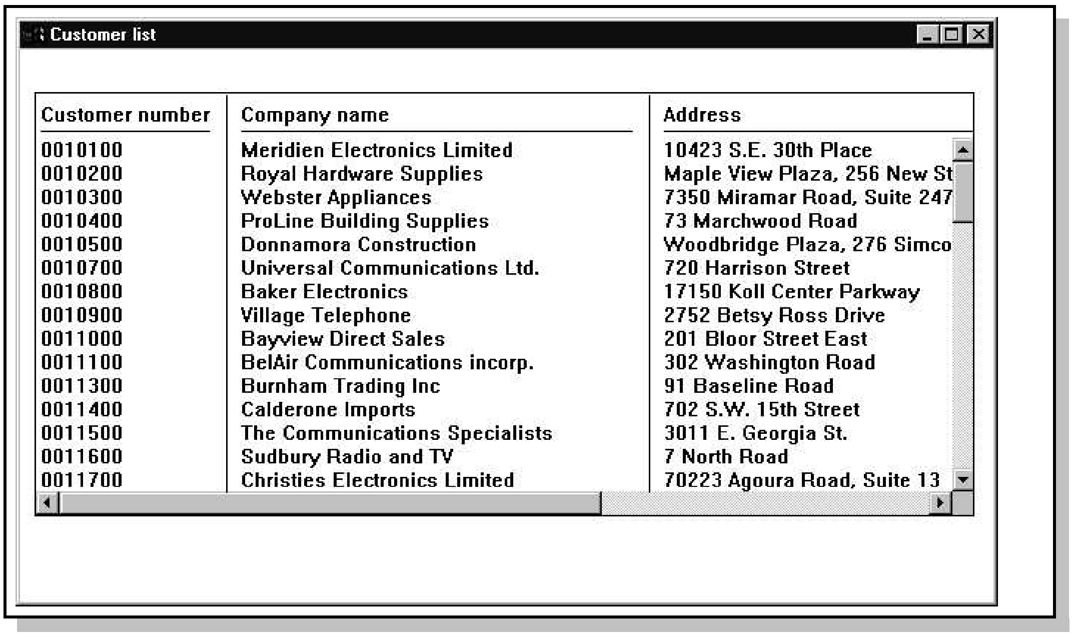
Viewing the Compile Listing

Recall that the VisualAge RPG compiler is resident on the workstation. Therefore, the compile listing is also resident on the workstation. At any time, it is possible to view the files that make up your project including the compile listing.

To display the compile listing for this latest build choose in the GUI designer ***Project→Browse compile listing***. The compile listing will now be displayed. Note the similarities between the **VisualAge RPG** compile listing and the compiler output of the AS/400 RPG compilers.

If the project built successfully test it by choosing ***Project→Run***.

The subfile should be displayed with a list of all customers from the data base and look similar to the figure below.



Customer number	Company name	Address
0010100	Meridien Electronics Limited	10423 S.E. 30th Place
0010200	Royal Hardware Supplies	Maple View Plaza, 256 New St
0010300	Webster Appliances	7350 Miramar Road, Suite 247
0010400	ProLine Building Supplies	73 Marchwood Road
0010500	Donnamora Construction	Woodbridge Plaza, 276 Simco
0010700	Universal Communications Ltd.	720 Harrison Street
0010800	Baker Electronics	17150 Koll Center Parkway
0010900	Village Telephone	2752 Betsy Ross Drive
0011000	Bayview Direct Sales	201 Bloor Street East
0011100	BelAir Communications incorp.	302 Washington Road
0011300	Burnham Trading Inc	91 Baseline Road
0011400	Calderone Imports	702 S.W. 15th Street
0011500	The Communications Specialists	3011 E. Georgia St.
0011600	Sudbury Radio and TV	7 North Road
0011700	Christies Electronics Limited	70223 Agoura Road, Suite 13

Congratulations!!!!!!!!!!!!!!!!!

Part **one** of this lab is completed, you have written a VARPG application to display data from an AS/400 database in a GUI subfile.

The next step will guide you thru writing the code to react to the selection of one of the records in the subfile.

.

Part 2 Selecting a Record from the GUI Subfile

First lets discuss what is different between handling a selection in a **5250** subfile and a **GUI** subfile.

Normally when you display data in a subfile you want the endusers to select one of the records displayed, so they can work with this specific record. In a 5250 subfile you would typically provide a selection input field as one of the columns of the subfile and have the enduser enter a character like "***/***" to identify the selected record. Currently you then program logic with a **READC** (Read changed) operation code to get the changed (selected) record from the subfile and you can then work with the information in this record.

In the **GUI** environment the way you work with selected records changes a little bit, instead of using an **input field** in the subfile for the selected record identification, the GUI allows us to identify a record by **selecting** it (clicking on it). That means you don't need a selection input field in a subfile anymore, you can work directly with a selected record. How do you know which record is selected? By using the new **READS** (Read selected) operation code, instead of **READC**.

Since you work in an event driven environment you also need an **event** that signals to you that the end user has **selected** a record. For a GUI subfile you have 2 choices:

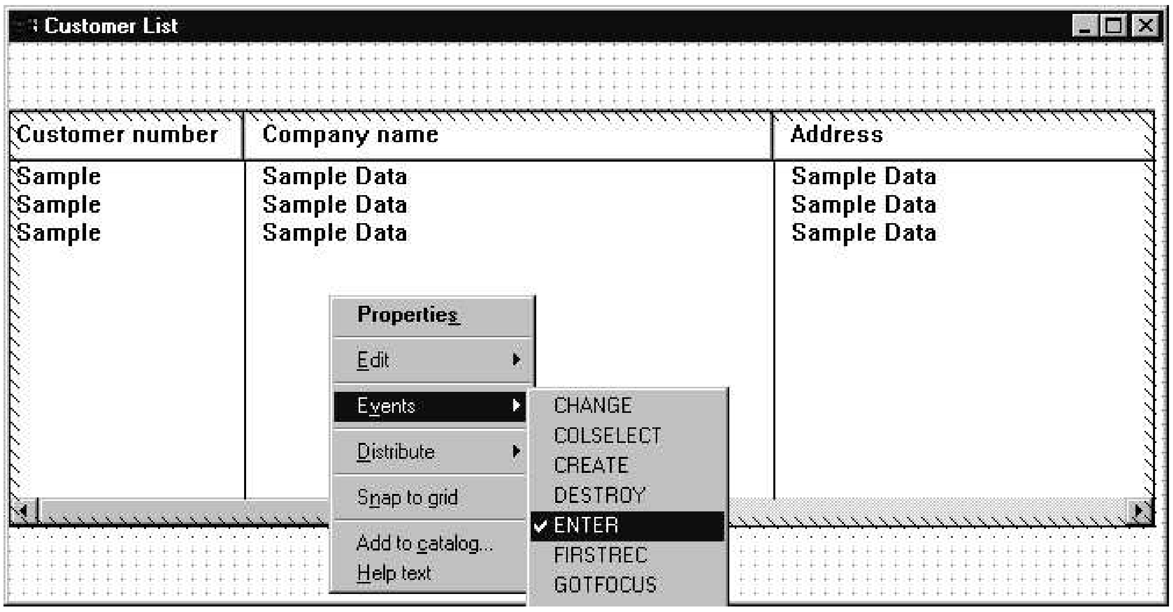
- **Select** event - **single click** on a record

  or

- **Enter** event - **double click** on a record

The **select event** gets triggered by a **single mouse click** on the subfile, the **enter event** gets triggered by a **double click** on the subfile.
In this example we want you to react to the **enter** event of the subfile. Therefore, you need to create an action subroutine for the subfiles **Enter** event.

1. **Right mouse click** on the subfile



2. From the subfiles' **pop up** menu, **select** the **Enter** event to create an action subroutine.
   In this action subroutine, you will use the **READS** (Read selected) operation code to get the selected record from the subfile. The **READS** will move the values for this record into the corresponding program variables.
   The **READS** operation code requires the subfile part name in factor 2 (**without the quotes**).
3. Key in the **READS** statement
   To verify that the correct record was read we suggest you start another **VARPG** component called ***SELECTED*** and **pass** to it the selected customer name as a parameter.
   The component ***SELECTED*** is already on your system, you only have to invoke it and pass the parameter you don't have to create it.

```
CL0N01Factor1+++++++Opcode(E)+Factor2+++++++Result++++++++Len++D+HiLoEq
*
C     SFL1          BEGACT    ENTER         CUSTLIST
C                   reads     SFL1                                    98
C                   if        not *in98
C                   Start     'SELECTED'                            97
C                   parm                    custna
C                   endif
C                   ENDACT
```

Key in the **START** statement and the PARM operation code, this will invoke the VARPG component called **SELECTED** which will display the selected **Customer name**.

Note: The **START** operation code behaves like a **CALL** but is **asynchronous**, so it allows to have 2 or more **VARPG components** active at the same time.

This is all the logic needed for this action subroutine for now.

Time to **build** the project:

1. Choose ***Project→Build→Windows NT/95*** from the GUI designer menu bar.

2. If you have made changes to any of the design windows, you will be prompted to save or discard your changes. Answer **Yes** (**SAVE**) to this message.

Testing Your Application

Let's see if the application runs:

There are two ways to run your project. Choose from one of the following:

- Choose ***Project→Run*** from the GUI designer menu bar

- Press the **Run** icon on the tool bar (the running Man).

Your window with data should be displayed.

Now **click once** on one of the records in the subfile, nothing happens, because this only caused a **select** event.

Double click on the record, the **enter** event gets triggered and the ***SELECTED*** component gets started showing another window, displaying the customer name selected in the subfile.

Part two is completed, **congratulations**, now we want to show you how to highlight individual cells in a subfile.

Part 3 Colour a subfile cell

We want you to highlight certain data in the subfile. One of the fields in this file is the country name CCOUNT. We want to draw the enduser's attention to records containing the country name **Canada**. In this example we are only looking for records with the name **Canada** in **exactly** the way it is shown **(uppercase/lowercase)**, all other records will not be highlighted.
To show a customer is located in Canada you will change the **foreground (**text) colour of the **row** to **red** and the **background** of the **country field** to **yellow**.
Before you can start, we want to recap how attributes get set. In the **5250** environment you use indicators to change attributes in your panels, for example to **highlight** a field.
In **VARPG** we have added 2 operation codes to the language allowing you to **set** and **get** attributes directly **without using indicators**.

These are the 2 operation codes:

SETATR - Set an attribute

GETATR - Get an attribute

The table below shows the syntax you need, to specify the name of the attribute you want to work with and the part containing the attribute:

Factor 1	Op-code	Factor2	Result
Part name	GETATR	Attribute name	AttrValue
Part name	SETATR	AttrValue	Attribute name

Lets look at some of the subfile attributes:
- **Index,** similar to relative record number
- **Colnumber**, identifies the column to be worked with
- **Rowfgclr**, Row foreground color
- **CellBgclr**, Cell background color

.
To identify a **cell** to be changed, you need to set the **index** attribute (record number) in our case the record just written to the subfile containing **"Canada**", the **column number** attribute has to be set to the value, in our case this is **column number 5** unless you rearranged the fields in the subfile. Then after the **cell** is identified by **index** and **column number** you can set the cell **background** colour to **Yellow**.

To change a **row foreground** color, just identify the row by setting the **index** attribute of the subfile and then change the **foreground color** attribute
Sounds complicated? Don’t worry it is not.

We need to change the user subroutine ***FILLSUB***,

1. Invoke the editor by pressing this button in the GUI designer
2. Scroll down to the subroutine.
   After the **WRITE** statement to the subfile you need to check the CCOUNT field.
   If its content is **Canada** you set the column number to **5** and change the background color to ***yellow.**
   Then you set the foreground color of the **current row** to ***red**
   You don’t have to set the index since it already got set by the write operation to the subfile.
   Here is the code to be added to the subroutine, highlighted with white background:

```
CL0N01Factor1+++++++Opcode(E)+Factor2+++++++Result++++++++Len++D+HiLoEq
C     FillSub       BEGSR
 * Read a record from database
C                   Read      Customl3                                99
 * Do until end of file
C                   Dow       *IN99=*OFF
 * Add a record to the subfile
C                   Write     SFL1
C* Check country name
C                   If        CCOUNT=’Canada’
C* Set column number to 5 (country name)
C     ‘SFL1’        SETATR    5             ‘colnumber’
C* Set cell background to *yellow
C      ‘SFL1’       SETATR    *yellow       ‘cellbgclr’
C      ‘SFL1’       SETATR    *red          ‘rowfgclr’
C                   ENDIF
 * Read a record from database
C                   Read      Customl3                                99
C                   EndDo
C                   ENDSR
```

Now you are ready to build and run your application and check out whether the code you just added really works.

Part 3 of your Lab is done, the next part deals with paging, instead of reading the complete file, you will add logic to only fill the subfile with one page at a time.

Part 4 Filling a subfile by page

Part 4 will show you how to fill a subfile **page by page** instead of filling it until the end of file is reached.

Filling the Subfile
In this part of the lab we want you to fill the subfile a page at a time and you will only display more data when a new page is requested. The subfile part will handle paging to previously loaded pages automatically.

Add the following logic to the **FillSub** subroutine to fill the next subfile page, the logic to add has a white background.

```
CL0N01Factor1+++++++Opcode(E)+Factor2+++++++Result++++++++Len++D+HiLoEq
C     FillSub       BEGSR
C                   EVAL      COUNT=1

C     'SFL1'        GETATR    'PAGESIZE'    NbrofRecs

 * Read a record from database
C                   Read      Customl3                               99
 * Do until end of file
C                   Dow      *IN99=*OFF and count <= NbrofRecs

 * Add a record to the subfile
C                   Write     SFL1
C                   EVAL      Count=Count+1
 * Check country name
C                   If        CCOUNT='CANADA'
C* Set column number to 5 (contry name)
C     'SFL1'        SETATR    2             'colnumber'
C* Set cell foreground to *Yellow
C      'SFL1'       SETATR    *YELLOW       'cellfgclr'
C      'SFL1'       SETATR    *GREEN        'rowbgclr'

C                   ENDIF
 * Read a record from database
C                   Read      Customl3                               99
C                   EndDo
C                   ENDSR
```

This code will fill the subfile with one page of records.
You need to define variables ***NbrofRecs, Count,*** both are defined as 2 bytes numeric.
Here are the 2 **D** specs needed:

```
DName+++++++++++ETDsFrom+++To/L+++IDc.Keywords
D NbrofRecs      s              2  0
D Count          s              2  0
```

Tip: **D** specs are located after **F** specs, before **I** or **C** specs.

How do you know how many records fit into the visible list area of the subfile**?**

By using the **subfile *PAGESIZE*** attribute. The ***PAGESIZE*** attribute will return the number of records (rows) that fit into the visible subfile list area.

You use the **GETATR** operation code to get this information from the subfile part, using this coding technique instead of using a constant has the advantage of being more flexible. If you decide to enlarge the subfile visible list area later on, your logic will adjust automatically to it.
The code you have added will only fill the first page when the application gets started. To get subsequent pages you need to get feedback from the enduser that he(she) wants to see more records. You have to use an additional event. The subfile **PAGEEND** event gives you the capability to receive this information.

Lets code the action subroutine for the **PAGEEND** subfile event.

1. In the GUI designer select the subfile part
2. Right mouse click on the subfile
3. Select **Event** and **PAGEEND** from the **POPUP** menu
4. That will bring up the editor with the **BEGACT/ENDACT** statements pre filled:
5. Key in the code to invoke subroutine **FillSub**

```
CL0N01Factor1+++++++Opcode(E)+Factor2+++++++Result++++++++Len++D+HiLoEq
C     SFL1          BEGACT    PAGEEND       CUSTLIST
 * Fill subfile with records from database
C                   EXSR      FillSub
C                   ENDACT
```

With this code you just invoke the **FILLSUB** subroutine to add another page of records to the subfile.

Tip: To make sure that you don't show text of half of a record in your subfile, set the **SizetoFit** attribute to 1. This will guarantee that all records are shown fully visible and not cut off in the middle. You can do this in the **subfile properties** notebook on the Styles page.

Test the application

Its time to test our program.

1. Build your project and
2. Fix any compile time errors you may have.

3. Run your project and,
   if all goes well, you should see your subfile filled with the first page of records from the subfile.
4. The problem with this solution is that you have filled the subfile exactly with one page, so it will not show a scroll bar to indicate there are more records to show. Unless you hit the page down key there is no way to trigger the pageend event.

You will need to change your program so a scrollbar gets displayed, you do that by adding one more record.

End your program and go back to the **GUI designer** and into the source editor.

Change your VARPG program source.

Add a variable called **First** that deals with the first page.

You initialize **First** to 1, and on the first page you subtract it from the starting count. That adds one more record and you will get a **scrollbar on** the subfile.

For the first page you have added **one record** too many, so you need to add one less record on the second page by setting **First to -1**, then your second page gets filled with the correct number of records, and you need to set **First to 0** for any additonal page.

Here is the code to add:

```
CL0N01Factor1+++++++Opcode(E)+Factor2+++++++Result++++++++Len++D+HiLoEq
C     FillSub       BEGSR
C                   EVAL      COUNT=1
C* Show scroll bar when first page is filled, add one record
C                   EVAL      COUNT=COUNT-FIRST
C                   IF        FIRST=1
C                   EVAL      FIRST=-1
C                   ELSE
C                   EVAL      FIRST=0
C                   ENDIF
**********************************************************
C     'SFL1'        GETATR    'PAGESIZE'    NbrofRecs
*********************************************************
 * Read a record from database
C                   Read      Customl3                                99
 * Do until end of file
C                   Dow      *IN99=*OFF and count <= NbrofRecs
                                        ********************
 * Add a record to the subfile
C                   Write     SFL1
C                   EVAL      Count=Count+1
 * Check country name
C                   If        CCOUNT='CANADA'
C* Set column number to 5 (contry name)
C     'SFL1'        SETATR    2             'colnumber'
C* Set cell foreground to *Yellow
C      'SFL1'       SETATR    *YELLOW       'cellfgclr'
C      'SFL1'       SETATR    *GREEN        'rowbgclr'

C                   ENDIF
 * Read a record from database
C                   Read      Customl3                                99
C                   EndDo
C                   ENDSR
```

Declare variable ***FIRST*** which is defined as a numeric 1 byte variable initialize to 1

```
DName+++++++++++ETDsFrom+++To/L+++IDc.Keywords
D First         s             1  0 Inz(1)
```

Compile your application and try it out, by **getting the slider down to the bottom** of the scroll bar you will trigger a **pageend** event.

Now to enhance the application, you will add a **navigation bar** to the subfile.

Part 5 Adding the Navigation Bar

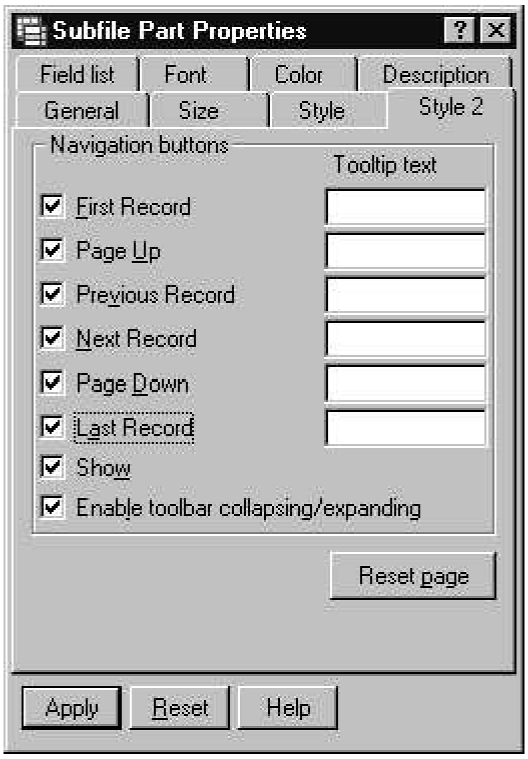
Let’s add some function to allow the user to **scroll** to the next page of records. You could add **Previous** and **Forward** push buttons to the window do accomplish this, but for this lab you will use the **Navigation** buttons available for the subfile part.

1.In the **GUI designer** bring up the **design window** if it is not visible by double clicking on the **CUSTLIST** Window Icon in the Project view tree view.

2.Open the properties notebook for the **subfile** part

3.Go to the ***Style*2** page.

4.In the **Navigation buttons** group, select **all** check boxes.


Subfile Part Properties

Navigation buttons — Tooltip text

- [✓] First Record
- [✓] Page Up
- [✓] Previous Record
- [✓] Next Record
- [✓] Page Down
- [✓] Last Record
- [✓] Show
- [✓] Enable toolbar collapsing/expanding

.

Notice that your subfile now has a series of **navigation buttons** attached to it’s right side. Pressing one of these buttons generates a corresponding subfile event. From the first (top) button down, the following events are available:

FIRSTREC	Go to first record
PAGEUP	Go to previous page
PREVREC	Go to previous record
NEXTREC	Go to next record
PAGEDOWN	Go to next page
LASTREC	Go to last record

***Note**: The event handling code for these events can perform any function. These are just suggested actions*

Using the navigation buttons, we want the user to be able to scroll to the next page of records. This is represented by the navigation button with the double down arrow icon. Pressing this button generates a **PAGEDOWN** event.
As you know, we already reacting to the **PAGEEND** event but this event is different it gets triggered when the visible list area is located near the end of the subfile. **PAGEDOWN** would get triggered when the visible area is not located at the end of the subfile.

Note: For the subfile, the **PageDown** event is triggered by the following user actions when the list is not located at the end of the subfile:

-- Pressing the **pageDown button** on the **keyboard**

-- Pressing the **pagedown button** on the **navigation bar**

-- Pressing the **lower part** of the **scroll bar**

```
CL0N01Factor1+++++++Opcode(E)+Factor2+++++++Result++++++++Len++D+HiLoEq
C     SFL1          BEGACT    PAGEDOWN      CUSTLIST
C     ‘SFL1’        GETATR    ‘TOPRECORD’   TOPLINE
C     ‘SFL1’        GETATR    ‘COUNT’       reccount
C                   EVAL      TOPLINE=TOPLINE+NbrofRecs
C                   IF        TOPLINE>reccount + first  AND NOT *IN99
C                   EXSR      FILLSUB
C                   ENDIF
 * SET the new top record
C     ‘SFL1’        SETATR    TOPLINE       ‘SETTOP’
C                   ENDACT
```

Declare variable **TOPLINE** and **RECCOUNT** as 5 numeric 0 decimal in your **D specs**

The subfile attribute that you need to get is called **TOPRECORD** this attribute will return the record number of the record on TOP of the visible subfile list area . This logic will increment the current **TOP record** in the list by the number of records that fill one subfile page. This number is used to identify the new **TOP** record in the subfile, scrolling exactly one page down. If the **end** of the **subfile** is another page will be read from the database, you use the COUNT subfile attribute to determine the number of records currently stored in the subfile.

Testing your application

Build your project and **test** it.

When the window is displayed, press the **PAGEDOWN** navigation button. The next page of records is read from the database and added to the subfile.
The program handles providing the next page of records and the system handles scrolling to previous pages.
You can use the slider to look at previous records, the system handles this for you

Press the other navigation buttons. Since you have not coded any event handling code for these buttons, **nothing** happens.

Handling page up events

Now you will implement the **page up** logic.
You'll use the **pageup** event from the **navigation bar** and move the subfile records accordingly, you need to know what record is currently displayed on **top** of the subfile visible list area.

Use the **TOPRECORD** attribute and calculate the new top record that should be displayed after the **PAGEUP** button gets pressed.
This is the **current top record** minus the PAGESIZE of your subfile which is stored in the **NbrofRecs** variable.
Use **this number** or **1** if you get a result of **0** or **negative** and set the attribute **SETTOP** with it.

Note: If you don't enable the navigation buttons the **keyboard** page/up page/down keys can always be used to scroll thru the subfile automatically.

This code should give you the **page up** capability, try it out:

```
 ***********************************************************
 * Window . . : CUSTLIST
 *
 * Part . . . : SFL1
 *
 * Event  . . : PAGEUP
 *
 * Description:
 *************************************************************
C     SFL1          BEGACT    PAGEUP        CUSTLIST
C     'SFL1'        GETATR    'TOPRECORD'   TOPLINE
C                   EVAL      TOPLINE=TOPLINE-NbrofRecs
C                   IF        TOPLINE < 1
C                   EVAL      TOPLINE=1
C                   ENDIF
C     'SFL1'        SETATR    TOPLINE       'SETTOP'
C                   ENDACT
```

Build and test your application.

If you use the PAGEUP and PAGEDOWN keys you will notice that the paging is off a bit. The reason for this is the fact that the PAGEUP/PAGEDOWN keys do their own page advance and than your program gets the PAGEUP/PAGEDOWN events as well and does additional scrolling. To avoid this you can add code to indicate that the keys have been pressed and in that case your code would not be invoked.
In **Appendix A** you can find the implementation to enable the PAGEDOWN/PAGEUP keys.

We suggest that you move on to the next exercise first. If you have time get back to Appendix A.

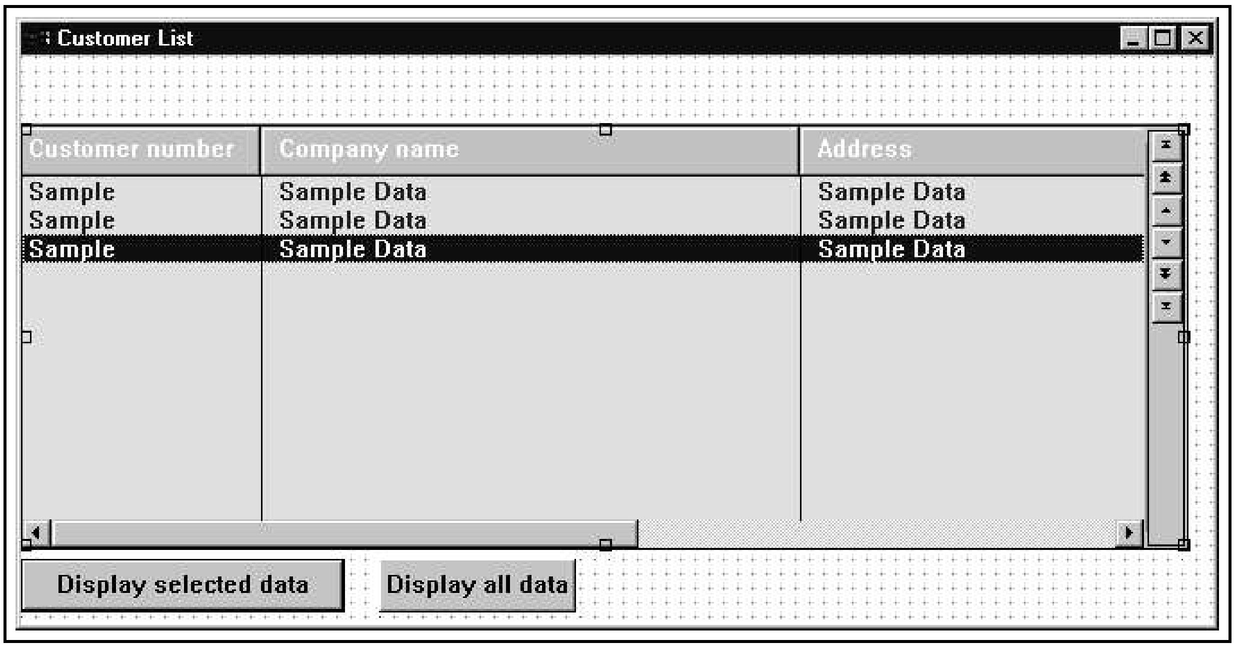
Part 6 Hiding Columns

In this lab you will use the subfile's **HIDE** attribute to selectively display certain columns of the subfile. Currently, when the subfile is displayed, all data for each customer is displayed. By adding logic to 2 new push buttons **PB_VIEW1** and **PB_VIEW2** you will allow the user to display all data including the **address** and the **country name**, ***or***, just display the **customer number**, **name** and **city**.

Display selected columns only

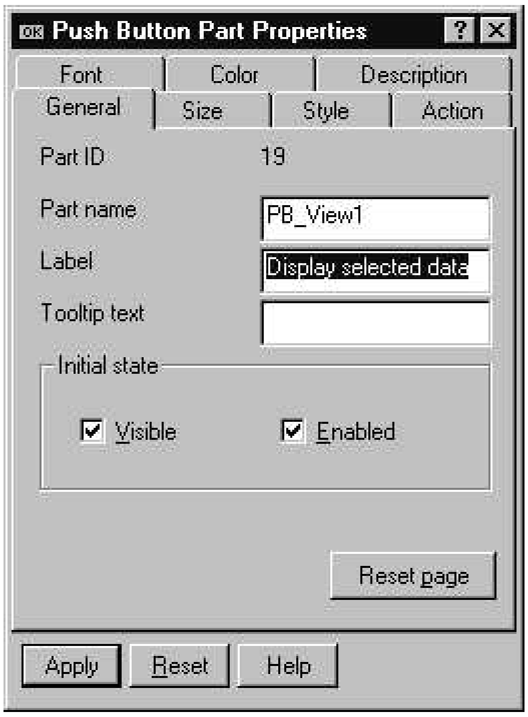
To display selected columns only, you need to hide the ***CADDRESS*** and the ***CCOUNT*** columns. You select these columns by setting the subfile's **COLNUMBER** attribute. In this case you want to affect columns **3** (Address), and **5** (Country).

First add 2 pushbuttons to the ***CUSTLIST*** window underneath the **subfile part**.

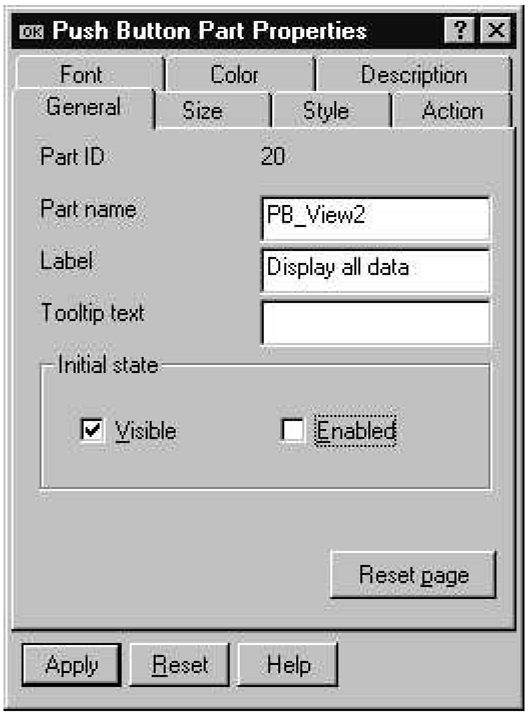


Change the properties of these pushbuttons:

Part	PartName	PartLabel
Pushbutton 1	***PB_VIEW1***	**Display Selected data**
Pushbutton 2	***PB_VIEW2***	**Display all data**


Push Button Part Properties

Part ID 19
Part name: PB_View1
Label: Display selected data
Tooltip text:
Initial state: [✓] Visible [✓] Enabled


Push Button Part Properties

Part ID 20
Part name: PB_View2
Label: Display all data
Tooltip text:
Initial state: [✓] Visible [ ] Enabled

Also check off the Enabled attribute on the **General Page** of **PB_VIEW2**.

Resize the pushbuttons so all label data is visible.

Create an action subroutine for the **PRESS** event of push button **PB_VIEW1** (Selected data only) and add the following logic:

```
CL0N01Factor1+++++++Opcode(E)+Factor2+++++++Result++++++++Len++D+HiLo
Eq....
C     PB_VIEW1      BEGACT    PRESS         CUSTLIST
C     ‘SFL1’        Setatr    3             ‘ColNumber’
C     ‘SFL1’        Setatr    1             ‘Hidden’
*
C     ‘SFL1’        Setatr    5             ‘ColNumber’
C     ‘SFL1’        Setatr    1             ‘Hidden’

C     ‘PB_VIEW2’    Setatr    1             ‘Enabled’
C     ‘PB_VIEW1’    Setatr    0             ‘Enabled’

C                   ENDACT
```

When this code runs columns **3** and **5** will be hidden since the **HIDE** attribute is set to **1**. The last two lines of code disable the **‘Selected Data’** push button and enable the **‘All Data’** push button.

Show all data

To show all data, add the following logic to the **PRESS action** subroutine of the push button **PB_VIEW2**:

```
CL0N01Factor1+++++++Opcode(E)+Factor2+++++++Result++++++++Len++

C     PB_VIEW2      BEGACT    PRESS         CUSTLIST
C     ‘SFL1’        Setatr    3             ‘ColNumber’
C     ‘SFL1’        Setatr    0             ‘Hidden’

C     ‘SFL1’        Setatr    5             ‘ColNumber’
C     ‘SFL1’        Setatr    0             ‘Hidden’

C     ‘PB_VIEW2’    Setatr    0             ‘Enabled’
C     ‘PB_VIEW1’    Setatr    1             ‘Enabled’
C                   ENDACT
```

1. Save your application
2. Build it
3. Run it
4. Press the pushbutton to hide the 2 columns
5. Press the other pushbutton to make the columns visible again

Lab Summary:

In this little exercise you got a chance to work with the **HIDE** attribute of the subfile If you want to permanently hide a column you can do that at design time in the properties note book for individual subfile fields.

Part 7 Opening cells for input

The 5250 subfile is very often used for input of data, and the GUI subfile provides this same capability.
To open a cell the end user has to indicate which cell he wants to work with. We will have you reacting to the **right mouse button click**. This is **POPUP** event for the subfile. You will **open** the selected record at the **column (field)** where the mouse cursor was positioned at.

What you need is an action subroutine for the **POPUP** event of the **subfile**, and the **%index** and **%colnumber** event attributes for the **popup** event.
We now introduce you to something new, an **event attribute.** Events can have attributes as well, these **event attributes** will return values like functions, so they have a **%** sign in front of their names like **built-in functions**. In case of the **POPUP** event, the **%colnumber** event attribute returns the **column number** of the column the mouse cursor is located on when the event occurs. The **%index** event attribute returns the index number of the record the mouse cursor was located on when the **popup** event got triggered.

Create the popup action subroutine:
1. In the GUI designer, **right mouse Click** on the subfile
2. Select **Events--> PopUp**
3. Write the code to identify the column
4. Write the code to open the cell for edit

```
CL0N01Factor1+++++++Opcode(E)+Factor2+++++++Result++++++++Len++

C     SFL1          BEGACT    POPUP         CUSTLIST
C     ‘SFL1’        SetAtr    %Index        ‘INDEX’
C     ‘SFL1’        Setatr    %colnumber    ‘ColNumber’
C     ‘SFL1’        Setatr    1             ‘OpenEdit’
C                   ENDACT
```

One more thing, before you build your application, you need to define ***%colnumber*** and ***%index***.
Create a **D** specification line for ***%colnumber*** as a single variable. Make it **numeric 2 bytes** length with **0** decimals.
The ***%index*** variable should be defined **4** bytes **0** decimals.
You are ready to build and test, go ahead.

Run your application, select a record and then right mouse click in a column, does the **selected** cell open for **Input**?

Tip: Endusers can open cells for input by pressing the **ALT** key and selecting a cell with the left mouse button.

Next we want you to protect the **customer number column** from being changed.

Part 8 Protect certain areas of the subfile

The subfile by default allows **all** cells to be changed during runtime, if you don’t like this feature you can protect the **entire** subfile from changes thru the properties notebook at design time. You can also set the **AllowEdit** attribute during **design** time or **runtime**. In addition the subfile allows for more granular protection, you can **allow/disallow** change on a **row/column/or cell** base.
In this little example here we want you to **not allow change** in the **first** column, we want you to protect the customer number column.
The attribute you use, is the **AllowEdit** attribute, setting this attribute to **3** will disallow editing of the selected column. That means before setting this attribute you have to set the **colnumber** attribute to **1**.

You will add this code at **Create** time since this protection should be applied during the lifetime of the subfile. You already got an action subroutine that gets invoked at **creation** of the window, add the code to **protect** the **customer number column** to this existing action subroutine.

1. Right mouse click on the Window frame
2. Select **Events** from the popup,
3. The **Create** event should have a **checkmark**, indicating an action subroutine already exists
4. Select the **Create** event
5. The editor gets positioned at the **CREATE** action subroutine for this Window
6. **Add** the following lines to the action subroutine, after the BEGACT

```
CL0N01Factor1+++++++Opcode(E)+Factor2+++++++Result++++++++Len++

C     CUSTLIST      BEGACT    CREATE        CUSTLIST
C     ‘SFL1’        Setatr    1             ‘ColNumber’
C     ‘SFL1’        Setatr    3             ‘AllowEdit’
C    ............
```

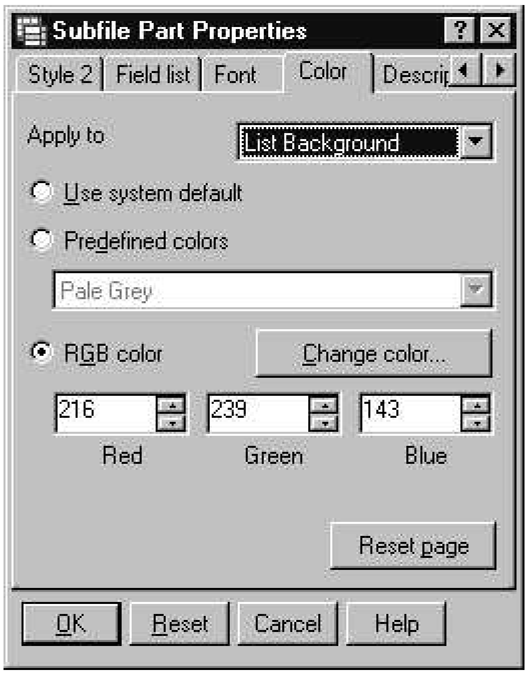
1. Save and built your application
2. Run the application
3. Try to open the column number field, by pressing **ALT** and left mouse click in this column

Part 9 Color the subfile

Changing subfile background /foreground colors

In this part of the lab you will see how you can change the colour of the entire subfile, as well as the heading. Then you will colour every **second row** differently and **highlight one column**.

The subfile colors are easily changed by using the properties notebook. To get some experience with **color mixes** you will use a **mix** in this exercise.
In the GUI designer:



Apply to: List Background
Use system default
Predefined colors: Pale Grey
RGB color (selected)
Red: 216
Green: 239
Blue: 143

1. Double click on the subfile to get the properties notebook
2. Select the **Color TAB** on the notebook
3. Select **List Background** in the **Apply to** Combo box
4. Select the **RGB** color radio button
5. Specify: **Red: 216, Green: 239, Blue: 143**

Now you need to change the text color on the same notebook page:
1. Select **List foreground** in the **Apply to** Combo box.
2. Select the Predefined colors Radio button
3. Select blue from the Combo box

Now you change the **heading color,** on the same notebook page:
1. Select **Heading background** in the Apply to Combo box
2. Select the **RGB color** radio button
3. Specify **Red: 248, Green: 170, Blue: 148**

The last change you need to provide in the GUI designer is the **heading text** color, again on the same notebook page:
1. Select **Heading foreground** from the *Apply to* **Combo box**
2. Select the *Predefined colors* **radio button**
3. Select **white** in the **Combo box**

Now we want you to **color** every **second row differently** from the subfile base colors to make the subfile data easier to read.

1. Press the **Edit** pushbutton in the GUI designer toolbar, it will bring up the editor window.
2. Position the cursor in the **FILLSUB** user subroutine.
3. Add the following code to the subroutine, **after** the **WRITE** to **SFL1** statement in **FILLSUB**.

```
CL0N01Factor1+++++++Opcode(E)+Factor2+++++++Result++++++++Len++D+HiLoEq....
C                   If        RowInd
C     'SFL1'        Setatr    '247:245:183' 'RowBgMix'
C     'SFL1'        Setatr    *DarkGreen    'RowFGClr'
C                   Eval      RowInd=*OFF
C                   Else
C                   EVAL      RowInd=*ON
C                   EndIf
C                   . . .
```

Two more things before you can build and test the application.

1. You need to define variable ***RowInd***, add a **D** spec for a **single** variable, data type **N**.

To highlight the subfile column containing the CITY field add the following code:

1. At the end of the ***FillSub*** user subroutine **add** 2 lines of code,
2. one to **select the column** to be changed
3. one to change the **column background** color.

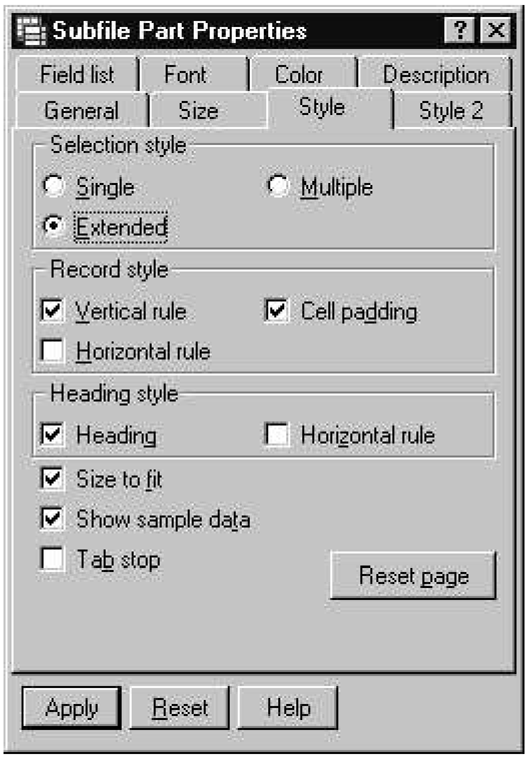
```
CL0N01Factor1+++++++Opcode(E)+Factor2+++++++Result++++++++Len++D+HiLoEq....
C     'SFL1'        Setatr    4             'ColNumber'
C     'SFL1'        Setatr    '64:192:208'  'ColBgMix'
```

Build and **test** your project.
When the subfile is displayed, **every other row** should be **colored differently**, and the **fourth column** should have a **different background** then the rest of the subfile.

Part 10 Support multiple selections

Now we want you to extend the subfile capabilities and allow **multiple** record selection. The **default** for the subfile is to only allow **single selection** of records, when one record gets selected the currently selected record gets de-selected. When you allow multiple selected records you also have to write code to deal with this fact. In your current logic you would only read the first selected record and not notice the other records at all.
In this example you will allow multiple selections and add a pushbutton that will indicate that the enduser wants to process the selected records. You will write logic to **read** the **selected** records and display a Window that has all the detailed information about the selected customer, it will allow the enduser to **update** the data and then **write** that updated information to the database.

First enable the subfile for extended selection,
1. Double click on the **subfile**
2. Select the **Style tab** in the properties notebook
3. Select the **Extended Selection** Radio button
4. Press **Ok**


Subfile Part Properties

Tabs: Field list, Font, Color, Description, General, Size, Style, Style 2

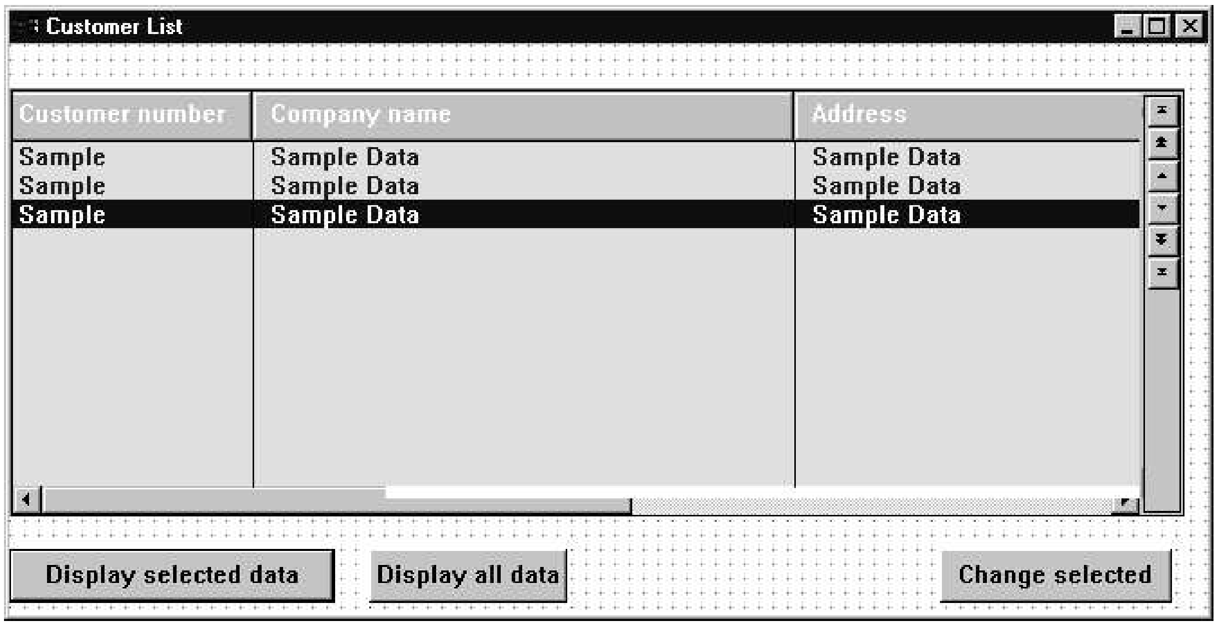
Selection style: ( ) Single, ( ) Multiple, (•) Extended

Record style: [✓] Vertical rule, [✓] Cell padding, [ ] Horizontal rule

Heading style: [✓] Heading, [ ] Horizontal rule

[✓] Size to fit
[✓] Show sample data
[ ] Tab stop

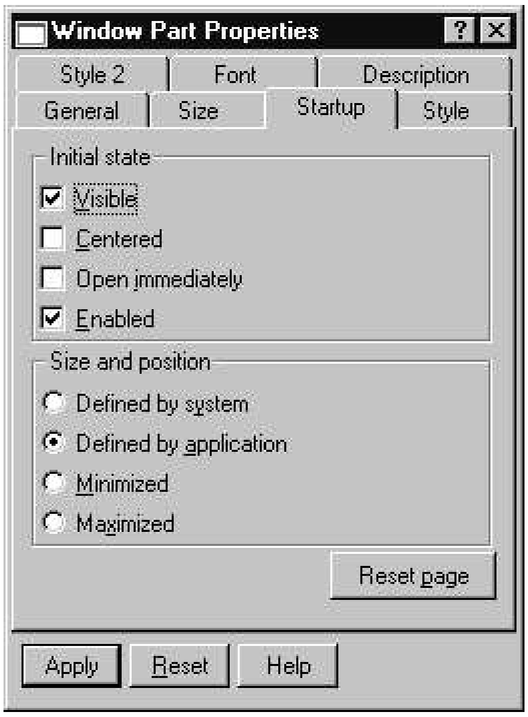
Add a pushbutton to the window under the right lower corner of the subfile,
1. Name it: **PSBProcess**
2. Change the Label to: **Change selected**



Now you have to create a **new** Window,

1. Click on the **Window icon** on the **parts palette**
2. Click on the **project view window** in the GUI designer
   A new Window gets created
3. Name the window: **WinChange**
4. Change its title to: **Update Customer Information**

5. On the **STARTUP** page of its property notebook, **unselect** the **open immediately check box**



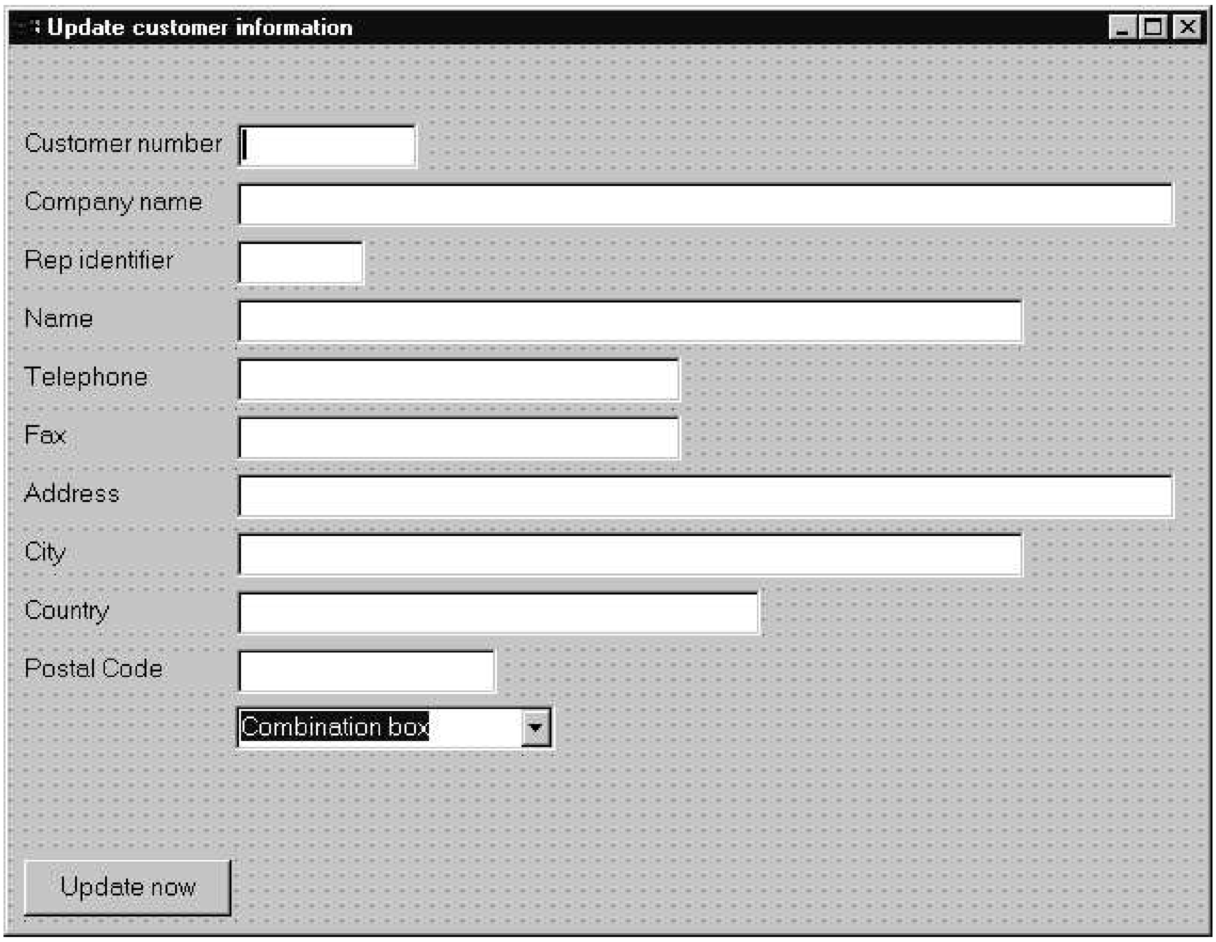
Initial state
- [✓] Visible
- [ ] Centered
- [ ] Open immediately
- [✓] Enabled

Size and position
- ○ Defined by system
- ● Defined by application
- ○ Minimized
- ○ Maximized

Define the reference fields to be used on this window. Select ***Server--->reference fields***

6. Select the AS/400, library **VARLABxx**, file **customer**, record format **CUSTOM01**
7. Double click on **CUSTOM01**
8. Select **all fields** from the **field list**
9. Position the cursor at the **upper left corner** of the new window you just created and click All fields will be positioned on the window
10. Click on the **Window frame** to expand it and the window size gets increased automatically so all fields are fully visible.
11. Add a **pushbutton** to the bottom of the window
12. Name this pushbutton: **PSBupdate**

13. Change its Label to: **Update now**



Now you have to write some code, first write the RPG source for the **press** event from this **update** pushbutton. When it gets pressed you want to **read** the **changed data** from this **window** and **write** the data to your **database file**.
Then you want your enduser to work with the next selected record in the subfile.

```
CL0N01Factor1+++++++Opcode(E)+Factor2+++++++Result++++++++Len++D+HiLoEq....
C     PSBUPDATE     BEGACT    PRESS         WINCHANGE
C                   READ      'WINCHANGE'
C                   UPDATE    CUSTOM01
C                   EXSR      Changewin
C                   EndAct
```

In order to **update** file **CUSTOML3** you need to change the F spec, instead of an **I**nput file make this file an **U**pdate file.

Now create a new user subroutine **CHANGEWIN:**

```
CL0N01Factor1+++++++Opcode(E)+Factor2+++++++Result++++++++Len++D+HiLoEq....
C     CHANGEWIN     BEGSR
C                   READS     SFL1                                  9595
C                   IF        not *IN95
C     CUSTNO        CHAIN     CUSTOML3                              93
C                   SHOWWIN   'WINCHANGE'                           93
C                   WRITE     'WINCHANGE'                           93
C                   ELSE
C                   CLSWIN    'WINCHANGE'                           93
C                   ENDIF
C                   EndSR
```

This subroutine will **read** the **next selected record** from the **subfile**, if there is still a selected record it will **read** the corresponding record from the database file, show your new window **WINCHANGE** and write the data to this window.
If **no more** record is selected, window **WINCHANGE** gets closed.

One more action subroutine to write, and then your are done, the pushbutton on the first window needs to get an action subroutine for its press event.

```
CL0N01Factor1+++++++Opcode(E)+Factor2+++++++Result++++++++Len++D+HiLoEq....
C     PSBPROCESS    BEGACT    PRESS         CUSTLIST
C                   EXSR      Changewin
C                   EndAct
```

Ok, you are all done, lets build and test

Run the application, select several records by holding the **CTRL key** down while selecting records
Then press the **Change selected** push button.
Your update window should show, change some data and press the **Update** push button.
You should get prompted for the next selected record and so on.

One more exercise and you are done

Part 11 Sort subfile by selected column

For this exercise you will be reacting to the **COLSELECT** event, this event happens when somebody left mouse clicks on the **subfile heading**. The **COLSELECT** event will update the **%COLnumber** event attribute that allows you to determine which column heading the **COLSELECT** occurred on.
We will assume the clicking on the header means the enduser wants to sort by this column. You will implement this feature for the CITY column, this is the column you already highlighted before by coloring its background differently. Clicking on the CITY column heading will present the data in the subfile sorted by city. You need to **clear** the subfile, and **open a logical file** that has an index specified by **CCITY.**

First you need to add an F spec for the new logical file ***CUSTOMS4*** this file should be using keywords **Remote** and **UsrOpn.**

Now you need to add an action subroutine for the **COLSELECT** event of the Subfile

```
CL0N01Factor1+++++++Opcode(E)+Factor2+++++++Result++++++++Len++D+HiLoEq....
C     SFL1          BEGACT    COLSELECT     CUSTLIST
C                   SELECT
C                   When      %COLNUMBER = 4
C                   OPEN      CUSTOMS4
C                   EVAL      SORTCITY=*ON
C                   WHEN      %COLNUMBER = 1
C                   CLOSE     CUSTOMS4
C    *START         SETLL     CUSTOML3
C                   EVAL      SORTCITY = *OFF
C                   Other
C                   LEAVESR
C                   ENDSL
C                   CLEAR                   SFL1
C                   EVAL      FIRST=-1
C                   EXSR      FILLSUB
C                   EVAL      FIRST=1
C                   EndAct
```

The **Select** statement allows to determine whether a valid column was selected, only column **4** or **1** are considered valid in this example. Variable **SORTCITY** is used to indicate whether the view is by **City** or by **customer number**.
You do need to specify a **D** spec that defines **SORTCITY** as an **indicato**r.
What does this code do ?
It enables to react to **COLSELECT** event from the subfile, when column **4** or **1** are selected the subfile will be filled with data from logical file **CUSTOMS4** sorted by **City** or file **CUSTOML3** sorted by **customer number**. **COLSELECT** events on any other column will be ignored.

You now need to enhance user subroutine **FILLSUB**, depending on the **SORTCITY** indicator it has to read from file **CUSTOMS4** or **CUSTOML3**.
Find the **READ** from file **CUSTOML3,** where it is located now you have to put the following lines around it.
Remember that we have **2** READ statements in subroutine **FILLSUB** so you have to add this code **twice**

```
CL0N01Factor1+++++++Opcode(E)+Factor2+++++++Result++++++++Len++D+HiLoEq....

C                   IF        not SORTCITY
C                   READ      CUSTOML3                                 9999
C                   ELSE
C                   READ      CUSTOMS4                                 9999
C                   ENDIF
```

Now you are all set to try it out, build it and test

Check whether your data in the subfile gets sorted correctly.

VisualAge RPG Lab Summary

See how easy it is to create a Client/Server application by using your RPG skills to produce these GUI applications your endusers are begging for .

This concludes the VisualAge RPG GUI subfile programming hands on lab. We hope you **enjoyed** the experience. Using what you have learned from this lab you can build on this experience and create your own **VisualAge RPG** applications by exploring many of the other features of **VisualAge RPG** that were not covered here.

For more details, including **downloadable** sample programs, tips and techniques, and service and support information, be sure to visit the **VisualAge RPG** web site at

www.software.ibm.com/ad/varpg.

Have fun with VisualAge RPG!

The VARPG development team from the IBM Toronto Laboratory

Appendix A Handling Pageup/Pagedown keys

This part will show you how to deal with PAgeup/PageDown keys as well as with the Navigation bar button.
You have coded the action subroutines to catch the PageUp/PageDown events, if the keys are pressed they will do their own paging and then your pageup/pagedown code gets invoked, this will cause double scrolling. Here are some hints how to avoid this.

Detecting that keys were used
To figure out that the keyboard keys were pressed use the VKeyPress event. If Event attribute %character has value X'14' or X'15' then you know the Pageup/Pagedown keys were pressed.
Here is some sample code:

```
CL0N01Factor1+++++++Opcode(E)+Factor2+++++++Result++++++++Len++D+HiLoEq
C     SFL1          BEGACT    VKEYPRESS     CUSTLIST
C                   if        %character = '14'
C                   eval      pageup=*on
C                   else
C                   eval      pageup=*off
C                   endif
C                   if        %character = '15'
C                   eval      pagedw=*on
C                   else
C                   eval      pagedw=*off
C                   endif
C                   ENDSR
```

Variables **pageup/pagedw** have to defined as indicator type (**N**) variables
Variable **% character** is a 2 byte character variable and has to be declared as well.

Now you need to change the action subroutines for the pageup and pagedown events from the subfile.

```
C      SFL1             BEGACT     PAGEUP         CUSTLIST
C                       IF         PAGEUP = *OFF
C      'SFL1'           GETATR     'TOPRECORD'    TOPLINE
C                       EVAL       TOPLINE=TOPLINE-NbrofRecs
C                       IF         TOPLINE < 1
C                       EVAL       TOPLINE=1
C                       ENDIF
C      'SFL1'           SETATR     TOPLINE        'SETTOP'
C                       ENDIF
C                       EVAL       PAGEUP = *OFF
C                       ENDACT
```

Pagedown is a bit more involved

```
CL0N01Factor1+++++++Opcode(E)+Factor2+++++++Result++++++++Len++D+HiLoEq
C     SFL1            BEGACT    PAGEDOWN      CUSTLIST
C     ‘SFL1’          GETATR    ‘TOPRECORD’   TOPLINE
C     ‘SFL1’          GETATR    ‘COUNT’       reccount
C                     IF         PAGEUP = *OFF
C                     EVAL      TOPLINE=TOPLINE+NbrofRecs
C                     IF        TOPLINE>reccount + first  AND NOT *IN99
C                     EXSR      FILLSUB
C                     ENDIF
  * SET the new top record
C     ‘SFL1’          SETATR    TOPLINE       ‘SETTOP’
C                     IF        TOPLINE + nbrofrecs > reccount
C                     EXSR      FILLSUB
C                     ENDIF
C                     ENDIF
C                     EVAL      PAGEDW = *OFF
              C                     ENDACT
```

The additional code will prefill the subfile with more records so the internal pageup routine always has enough records in the subfile to do the correct paging when getting to the bottom of the subfile.