Programming in ProvideX

Lab Manual

September 2001

Copyright © 2001 Best Software Canada Ltd. All rights reserved.
No part of this publication may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, electronic, mechanical, photocopied, recorded or othere, without prior written consent of Best Software Canada Ltd.

Lab 1 - Simple Screen Layout

Objectives:

In this section, we will look at how to:

- write a simple program
- use screen mnemonics
- use system function DTE()
- use system variable UID
- use the PRINT directive with screen positioning via @
- use a GLOBAL variable

---

1. Start a ProvideX Session; working directory should be **C:\PVXTRAIN**.
2. Create simple program to create a screen layout similar to the sample below:
   The title **Lab Program 1** is stored in a string variable **title$**

   ```
   010         LET TITLE$="Lab program 1"
   ```

   Use the PRINT directive with mnemonics to set text and background colours.
   Use the system variable **UID** for user name at the top-left corner.

   ```
   020       PRINT 'WHITE','_CYAN','CS',UID,
   ```

   Center title$ using numeric function int() and string function len().

   ```
   030       PRINT 'YELLOW',@(39-INT(LEN(TITLE$)/2)),TITLE$,
   ```

   Use system function **DTE( )** to get the date; use a format mask to output the system date as shown.

   ```
   040       LET D$=DTE(0:"%Dl %Ms %D")
   050       PRINT 'WHITE',@(80-LEN(D$),0),D$,
   ```

   The company name (e.g. **Kathryn's Cool Company**) is stored in GLOBAL string variable **%company$.**
   Center %company$ using PRINT directive with @(x,y) and the string function PAD().

   ```
   060       PRINT @(0,1), '_WHITE', 'BLUE', 'CE', PAD (%COMPANY$,80,2,"=") ,'BLACK',
   ```

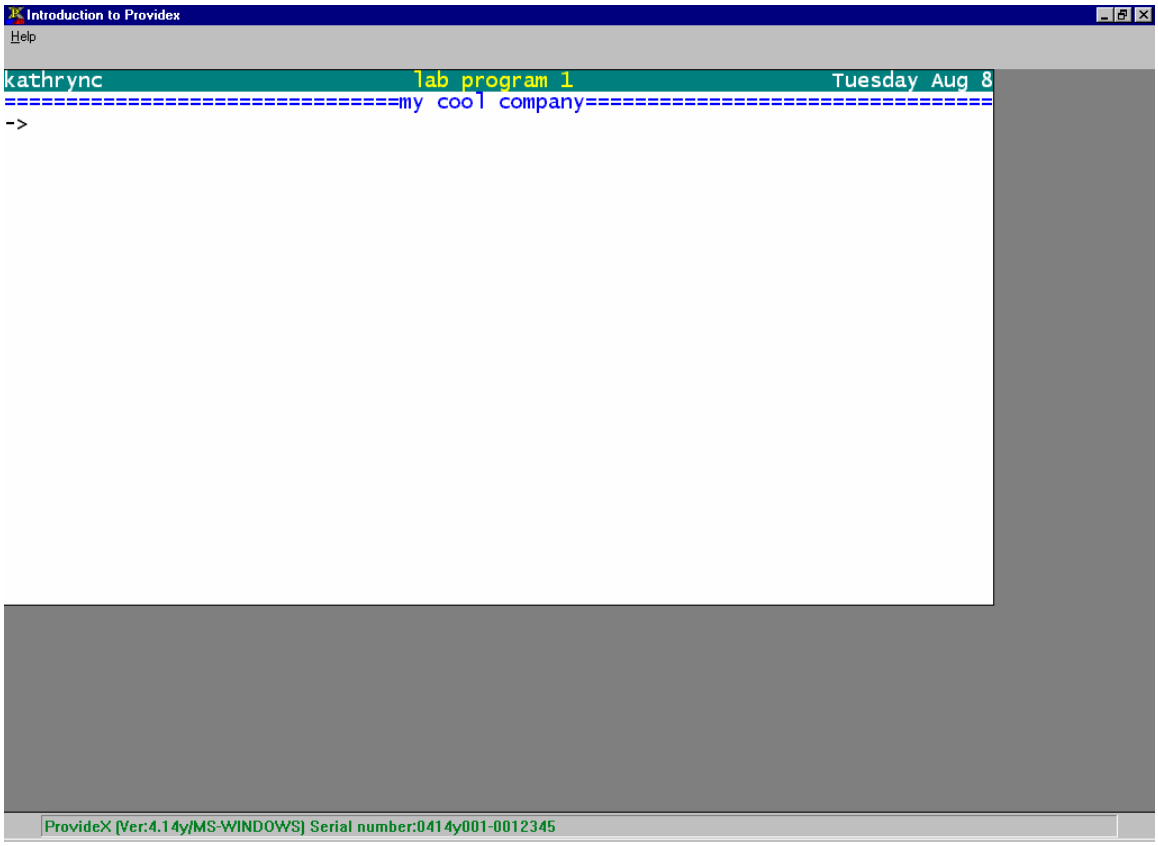
3. Save the program:
   SAVE "LAB1

4. Before running the program assign a value to **%company$:**
   %company$="Kathryn's Cool Company"

5. Run the program:
   RUN

Lab 1 - Simple Screen Layout

Screen layout should look like this:



Code:

```
00010  LET TITLE$="lab program 1"
00020  PRINT 'WHITE','_CYAN','CS',UID,
00030  PRINT 'YELLOW',@(39-INT(LEN(TITLE$)/2)),TITLE$,
00040  LET D$=DTE(0:"%DI %Ms %D")
00050  PRINT 'WHITE',@(80-LEN(D$),0),D$
00060  PRINT @(0,1),'_WHITE','BLUE','CE',PAD(%COMPANY$,80,2,"="),'MAGENTA'
```

Lab 1.2 - File Maintenance

Objectives:

In this section we will look at how to:

- create a file using the DIRECT command
- use INPUT EDIT to get information from user
- use SIZ= , LEN= options on INPUT to create scrolling input field
- use formatted INPUT to limit user entries
- perform file I/O; READ, WRITE, REMOVE
- use the IOLIST, MSGBOX and SETCTL directives

---

1. Create a DIRECT file with a 2-character key called **TERMCODE:**
   >DIRECT "TERMCODE",2

2. Load program "Lab1":
   >**LOAD "LAB1**

   and add the following code to create a File Maintenance program.

3. Open the TERMCODE file on the highest unopened channel number:
   00070 LET FL=HFN; OPEN (FL)"TERMCODE"

4. INIT & WRAPUP Logic:
   Use the SETCTL directive to accept a signal from the F4 key to quit the program:
   00080 ! initialization
   00090 INIT:
   00100 SETCTL 4:WRAPUP
   ...
   00430 ! wrap up
   00440 WRAPUP:
   00450 PRINT 'CS'
   00460 CLOSE (FL)
   00470 END

5. Establish a WHILE/WEND loop.
   Clear values from all variables in the IOLIST; set TRM_DATE$ to a value which will accept a format mask.
   Display an input prompt; use a format mask to ensure that only 2 characters can be input:
   00110 !
   00120 WHILE 1
   00130 READ DATA FROM "" TO IOL=1010
   00140 LET TRM_DATE$="00000000"
   00150 INPUT @(0,5),"Termcode: ",TRM_CD$:"00"
   ...
   00260 WEND

Lab 1.2 - File Maintenance

6. Read the file looking for a match on the input received from the user (TRM_CD$).
   Use **DOM=*next** option on the READ directive; if the record is missing from the file control will pass to the next line.
   Use the **MSGBOX** directive to prompt the user; depending on user response the program either passes control to the UPDATE label or continues to the SHOWIT logic:
   00160 READ (FL,KEY=TRM_CD$,DOM=*NEXT)IOL=0470; GOTO SHOWIT
   00170 MSGBOX "Create this record","record not on file","!,YESNO",X$
   00180 IF X$="YES" THEN GOSUB UPDATE
   ...
   01000 ! 1000 - iolist
   01010 IOLIST TRM_CD$,TRM_DESC$,TRM_DATE$

7. SHOWIT logic:
   Use the **PRINT** directive to display the existing value in TRM_DESC$ at column **0**, line **6**; use the PAD string function to display the value padded to 20 characters.
   Use the **PRINT** directive to display the existing value in TRM_DATE$ at column **0**, line **7**; the format mask "0000/00/00" to display date in YYYY/MM/DD format.
   Use the **INPUT** directive to display a prompt the user at column **0**, line **20**; user response is stored in RESP$.
   Depending on the value in RESP$ control passes to the UPDATE logic, the DELETE logic or to the WEND:
   00200 !
   00210 SHOWIT:
   00220 PRINT @(0,6),"Description: ",PAD(TRM_DESC$,20),@(0,7),"Date (YYYY/MM/DD): ",TRM_DATE$:"0000/00/00",
   00230 INPUT @(0,20),"Edit or Delete record? <E/D>",RESP$:"A"
   00240 IF RESP$="E" THEN GOSUB UPDATE
   00250 IF RESP$="D" THEN GOSUB DELETE

8. UPDATE logic:
   Use the INPUT EDIT directive to present the current value of **TRM_DESC$** at column **0**, line **6;** use **LEN=** and **SIZ=** options to create an input field which accepts 100 characters but only displays 20 at any time (a scrolling input field).
   Use the **MSGBOX** directive to issue a warning if this field is null; use the GOTO directive to return control to the INPUT.
   Use the INPUT EDIT directive to display the current value of **TRM_DATE$** at column **0**, line **7**; use the format mask "0000/00/00" to force valid input on the field.
   Use the **WRITE** directive to write the values back to the file, using the IOLIST at 1010; use the **MSGBOX** directive to confirm that the record has been updated.
   Use the **RETURN** directive to exit the gosub:
   00270 !
   00280 UPDATE:
   00290 INPUT EDIT (0,LEN=20,SIZ=10)@(0,6),"Description: ",TRM_DESC$
   00300 IF NUL(TRM_DESC$) THEN MSGBOX "Missing Desc","Big Err"; GOTO 0290
   00310 INPUT EDIT @(0,7),"Date (YYYY/MM/DD):",TRM_DATE$:"0000/00/00"
   00320 WRITE (FL,KEY=TRM_CD$)IOL=1010
   00330 MSGBOX "Record Update","Voila!"
   00340 RETURN

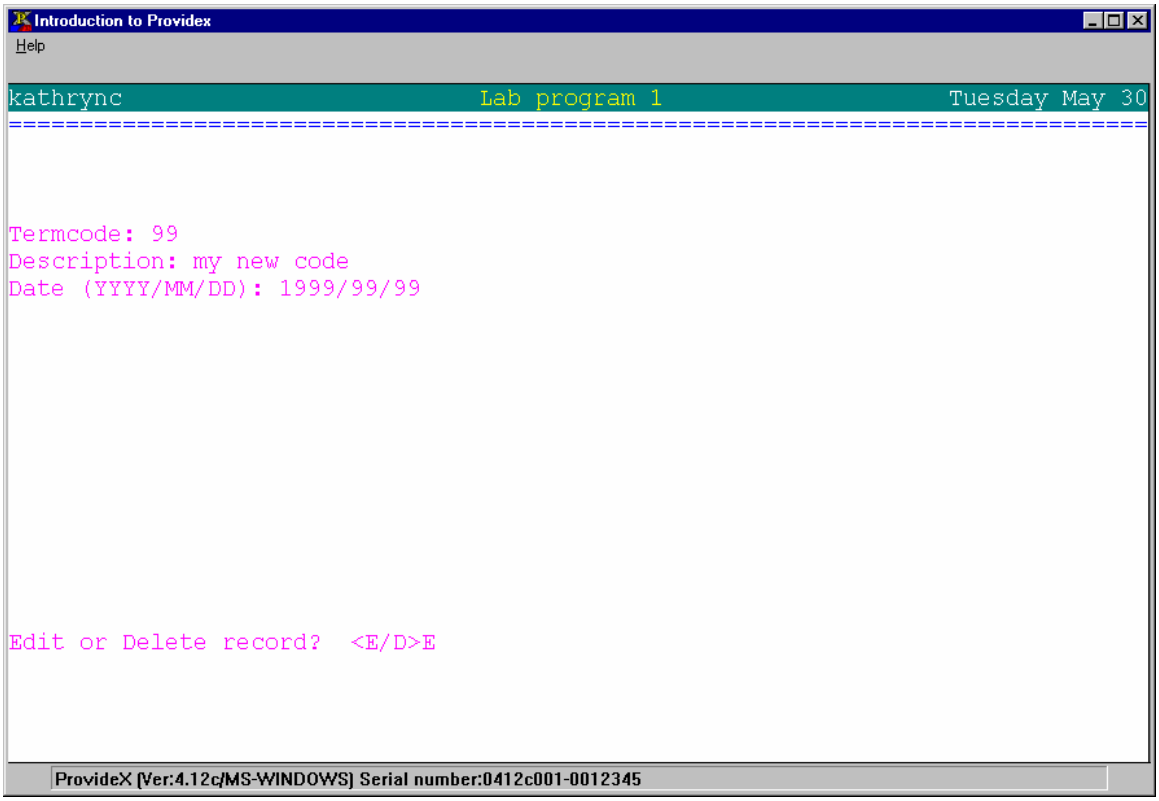
Lab 1.2 - File Maintenance

9. DELETE logic:
   Use the **REMOVE** directive to delete the record with the key equal to **TRM_CD$**; if the value does not exist on the file, control passes to the BADTRM label via the **DOM=** option.
   Use the **MSGBOX** directive to confirm that the record has been deleted.
   Use the **RETURN** directive to exit the gosub.
   In the **BADTRM** logic, use the **MSGBOX** directive to prompt user and use the **GOTO** directive to return control to the INPUT at line 230:

```
00350 !
00360 DELETE:
00370 REMOVE (FL,KEY=TRM_CD$,DOM=BADTRM)
00380 MSGBOX "Record Deleted!!","Way to go Man!!"
00390 RETURN
00400 !
00410 BADTRM:
00420 MSGBOX "No Such Record!","Whoops!!"; GOTO 0230
```

10. Renumber if necessary; save program and run.

Sample of application:



Lab 1.2 - File Maintenance

Code:

```
00010 LET TITLE$="lab program 1"
00020 PRINT 'WHITE','_CYAN','CS',UID,
00030 PRINT 'YELLOW',@(39-INT(LEN(TITLE$)/2)),TITLE$,
00040 LET D$=DTE(0:"%Dl %Ms %D")
00050 PRINT 'WHITE',@(80-LEN(D$),0),D$
00060 PRINT @(0,1),'_WHITE','BLUE','CE',PAD(%COMPANY$,80,2,"="),'MAGENTA'
00070 LET FL=HFN; OPEN (FL)"termcode"
00080 ! initialization
00090 INIT:
00100 SETCTL 4:WRAPUP
00110 !
00120 WHILE 1
00130 READ DATA FROM "" TO IOL=1010
00140 LET TRM_DATE$="00000000"
00150 INPUT @(0,5),"Termcode: ",TRM_CD$:"00"
00160 READ (FL,KEY=TRM_CD$,DOM=*NEXT)IOL=1010; GOTO SHOWIT
00170 MSGBOX "Create this record","record not on file","!,YESNO",X$
00180 IF X$="YES" THEN GOSUB UPDATE
00190 CONTINUE
00200 !
00210 SHOWIT:
00220 PRINT @(0,6),"Description: ",PAD(TRM_DESC$,20),@(0,7),"Date (YYYY/MM/DD):
",TRM_DATE$:"0000/00/00",
00230 INPUT @(0,20),"Edit or Delete record? <E/D>",RESP$:"A"
00240 IF RESP$="E" THEN GOSUB UPDATE
00250 IF RESP$="D" THEN GOSUB DELETE
00260 WEND
00270 !
00280 UPDATE:
00290 INPUT EDIT (0,LEN=20,SIZ=10)@(0,6),"Description: ",TRM_DESC$
00300 IF NUL(TRM_DESC$) THEN MSGBOX "Missing Desc","Big Err"; GOTO 0290
00310 INPUT EDIT @(0,7),"Date (YYYY/MM/DD):",TRM_DATE$:"0000/00/00"
00320 WRITE (FL,KEY=TRM_CD$)IOL=1010
00330 MSGBOX "Record Update","Voila!"
00340 RETURN
00350 !
00360 DELETE:
00370 REMOVE (FL,KEY=TRM_CD$,DOM=BADTRM)
00380 MSGBOX "Record Deleted!!","Way to go Man!!"
00390 RETURN
00400 !
00410 BADTRM:
00420 MSGBOX "No Such Record!","Whoops!!"; GOTO 0230
00430 ! wrap up
00440 WRAPUP:
00450 PRINT 'CS'
00460 CLOSE (FL)
00470 END
01000 ! 1000 - iolist
01010 IOLIST TRM_CD$,TRM_DESC$,TRM_DATE$
```

Lab 1.3 - File Maintenance - Enhanced

Objectives:

In this section we will learn how to:

- write a user-defined global function program to be PERFORMed from the main program
- change Screen Layout to a separate PERFORM'ed program
- use a CALL to *win/date utility to validate date field

Make the following changes to the File Maintenance program:

1. Launch the graphical editor "*IT" from ProvideX; include the program name as a parameter to the CALL:
   >**CALL "*IT","LAB1"**

2. Select File>New from the toolbar to open a second program for editing.

3. Enter the following code:
   0010 DEF FN%TM$(T)=STR(INT(T*60)+INT(T)*40:"00:00")

   Save the program.
   Select File >Save from the menu bar.
   Save the program as "**GBL_FUNC**"; when prompted select **Program** as the type of file.
   Close the file.

4. Open a new program file.

5. In "Lab1", highlight lines 0010 through 0060
   Select Edit>Cut from the menu or use the keyboard shortcut **Ctrl+X**; paste to the new program.

6. Insert the following code to the new program:
   00011 LET T=TIM
   00012 LET T$=FN%TM$(T)
   0070 EXIT

   Change line 0050 as follows:
   PRINT 'WHITE',@(80-(LEN(D$)+LEN(T$)+1),0),D$,"/",T$

   Renumber your program; use Tools> Renumber from the menu.
   Save the program as "**SCREEN_SET"**; when prompted select **Program** as type of file.

7. Select "LAB1" .
   In the **INIT** logic add the following code:
   LET %COMPANY$="Kathryn's Very Cool Company"
   PERFORM "GBL_FUNC"
   PERFORM "SCREEN_SET"

Lab 1.3 - File Maintenance - Enhanced

8. ProvideX includes a date validation and formatting utility "***WIN/DATE**". This utility can be CALLed from your program at the **VALIDATE** or **DISPLAY** labels. *WIN/DATE accepts three parameters - a string containing the date to be validated, a string to contain any error message to be returned and an optional tag string.

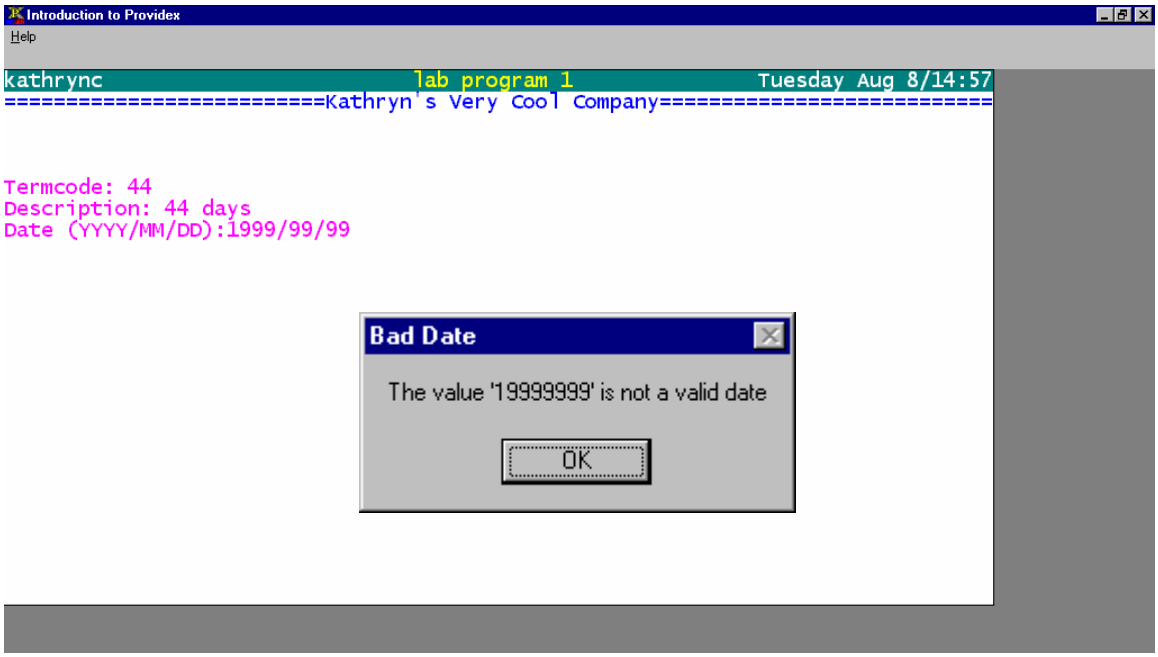
   Make the following changes to CALL the ProvideX date validation utility from the file maintenance program to validate the input on TRM_DATE$ field:

   In the UPDATE logic, add code to CALL "*WIN/DATE" at the VALIDATE label
   Pass the following parameters: TRM_DATE$, ERR$.
   Ensure that ERR$ is null when passed.
   Add logic to handle any errors returned by the validation program:
   00251 LET ERR$=""

   00281 CALL "*WIN/DATE;VALIDATE",TRM_DATE$,ERR$; IF ERR$<>"" THEN
   MSGBOX ERR$,"Bad Date"; LET TRM_DATE$="",ERR$=""; GOTO 0358

9. Save the program.
   Close the editor to return to the ProvideX command prompt.
   Load "**LAB1**" and run.

Sample:



Lab 1.3 - File Maintenance - Enhanced

Code:

Screen Set:

```
00010 LET TITLE$="lab program 1"
00020 LET T=TIM
00030 LET T$=FN%TM$(T)
00040 PRINT 'WHITE','_CYAN','CS',UID,
00050 PRINT 'YELLOW',@(39-INT(LEN(TITLE$)/2)),TITLE$,
00060 LET D$=DTE(0:"%Dl %Ms %D")
00070 PRINT 'WHITE',@(80-(LEN(D$)+LEN(T$)+1),0),D$,"/",T$
00080 PRINT @(0,1),'_WHITE','BLUE','CE',PAD(%COMPANY$,80,2,"="),'MAGENTA'
00090 EXIT
```

Lab1:

```
00010 LET FL=HFN; OPEN (FL)"termcode"
00020 ! initialization
00030 INIT:
00040 LET %COMPANY$="Kathryn's Very Cool Company"
00050 PERFORM "gbl_func"
00060 PERFORM "screen_set"
00070 SETCTL 4:WRAPUP
00080 !
00090 WHILE 1
00100 READ DATA FROM "" TO IOL=1010
00110 LET TRM_DATE$="00000000"
00120 INPUT @(0,5),"Termcode: ",TRM_CD$:"00"
00130 READ (FL,KEY=TRM_CD$,DOM=*NEXT)IOL=0440; GOTO SHOWIT
00140 MSGBOX "Create this record","record not on file","!,YESNO",X$
00150 IF X$="YES" THEN GOSUB UPDATE
00160 CONTINUE
00170 !
00180 SHOWIT:
00190 PRINT @(0,6),"Description: ",PAD(TRM_DESC$,20),@(0,7),"Date (YYYY/MM/DD):
",TRM_DATE$:"0000/00/00",
00200 INPUT @(0,20),"Edit or Delete record? <E/D>",RESP$:"A"
00210 IF RESP$="E" THEN GOSUB UPDATE
00220 IF RESP$="D" THEN GOSUB DELETE
00230 WEND
00240 !
00250 UPDATE:
00251 LET ERR$=""
00260 INPUT EDIT (0,LEN=20,SIZ=10)@(0,6),"Description: ",TRM_DESC$
00270 IF NUL(TRM_DESC$) THEN MSGBOX "Missing Desc","Big Err"; GOTO 0260
00280 INPUT EDIT @(0,7),"Date (YYYY/MM/DD):",TRM_DATE$:"0000/00/00"
00281 CALL "*win/date;validate",TRM_DATE$,ERR$; IF ERR$<>"" THEN MSGBOX
ERR$,"Bad Date"; LET TRM_DATE$="",ERR$=""; GOTO 0280
00290 WRITE (FL,KEY=TRM_CD$)IOL=1010
00300 MSGBOX "Record Update","Voila!"
00310 RETURN
00320 !
00330 DELETE:
00340 REMOVE (FL,KEY=TRM_CD$,DOM=BADTRM)
00350 MSGBOX "Record Deleted!!","Way to go Man!!"
00360 RETURN
```

```
00370 !
00380 BADTRM:
00390 MSGBOX "No Such Record!","Whoops!!"; GOTO 0200
00400 ! wrap up
00410 WRAPUP:
00420 PRINT 'CS'
00430 CLOSE (FL)
00440 END
01000 ! 1000 - iolist
01010 IOLIST TRM_CD$,TRM_DESC$,TRM_DATE$
```

Lab 1.4 - Terms Code Lookup

Objectives:

In this section, we will learn how to:

- use the 'window' mnemonic
- use the PREINPUT directive
- use DEFCTL directive
- use CTL values in program

Add hot-key functionality to the file maintenance program so that when a user presses the Ctrl+A combination at the Terms Code prompt they are presented with a Terms Code Lookup window. Any selection made in the Lookup will be passed back to the File Maintenance program.

Write a program to create a Terms Code Lookup window.
Values will be loaded from the TERMSCODE file:

1. Initialize variables and open the file:

```
00010 ! terms code lookup program
00020 LET DATE$="",OK$=""
00030 READ DATA FROM "" TO IOL=0510
00040 LET TRM=HFN; OPEN (TRM)"termcode"
```

2. Use the **'WINDOW'** mnemonic to create a window.
   Window start column is **5**, line is **5**, width is **50** columns, height is **20** rows.
   Window title is "Look Up";set **'SR'** mnemonic to eliminate scroll region on the window.

```
00050  PRINT 'WINDOW'(5,5,50,20,"Look Up"),'SR'
```

3. Use PRINT commands to display to the window:
   Use the **'CS'**, **'_BLUE'**, **'WHITE'** and **'CE'** mnemonics to set the background colour to blue, text to white for the window.
   PRINT a heading at row 0; position "Code:" at column **0**, "Description: " at column **10**, "Date: " at column **30**.
   Leave a blank line and then use the PAD string function to print a row of asterisks below the heading row.

```
00060  PRINT 'CS','_BLUE','WHITE','CE'
00070  PRINT "CODE: ",@(10),"DESCRIPTION: ",@(30),"DATE: ",'LF'
00080  PRINT PAD("",50,2,"*")
```

4. Read the TERMSCODE file into the variables on the IOLIST at line 510.
   Save a list of valid codes in **OK$** using the **+=** operator.
   Use **PRINT @** to position the output on the page.

```
00090 READ (TRM,END=DONE)IOL=0510
00100 LET OK$+=CODE$
00110 PRINT CODE$,@(10),DESC$,@(30),DATE$:"0000/00/00"; GOTO 0090

00500 ! 500 - IOLIST
00510  IOLIST CODE$,DESC$,DATE$
```

Lab 1.4 - Terms Code Lookup

5. Display an input prompt at column **0**, line **15**.
   Selected code will be passed to VAL$; apply input mask **"00"**.
   Set F4 key (**ctl 4**) to pass to WRAPUP.
   Use the POS function to check that entry is one of the valid codes; use the **INCREMENTS** option to scan OK$ in increments of 2.
   Include a MSGBOX to handle incorrect entry.
   Use the **PREINPUT** directive to pass the value in VAL$ to the input buffer.

```
00120 DONE:
00130 INPUT @(0,15),"SELECT CODE: ",VAL$:"00"
00140 IF CTL=4 THEN GOTO WRAPUP
00150 IF POS(VAL$=OK$,2)=0 THEN MSGBOX "Not a valid code. Please retry!";
            LET VAL$=""; GOTO 0130
00160 PREINPUT VAL$
```

6. WRAPUP Logic:
   Close the file.
   PRINT the 'POP' mnemonic to close the window and return to the previous screen.

```
00170 WRAPUP:
00180  CLOSE (TRM)
00190  PRINT 'POP'
```

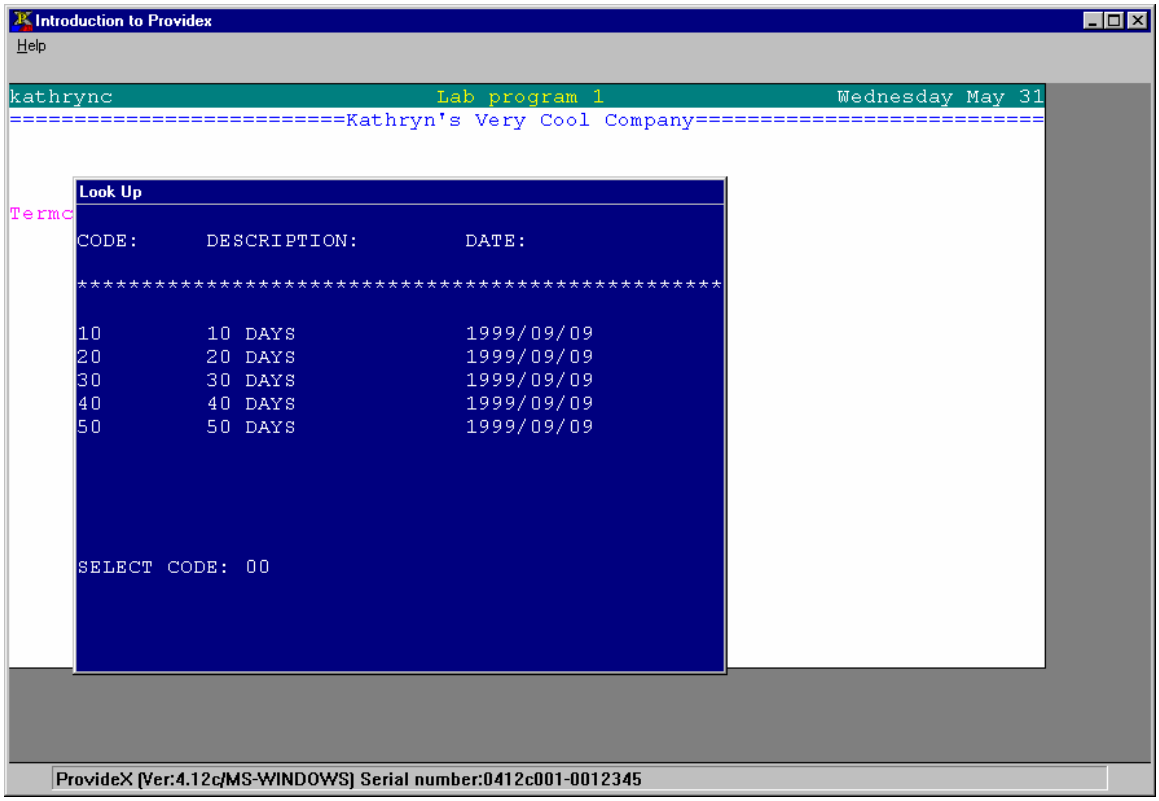
7. Save program as **"$CTL-666"**

8. The EOM value of the key combination Ctrl+A is 01; use the **DEFCTL** directive to assign as hot-key to the Table Lookup program.
   Load "**LAB1**" and add the following code.:

```
00041  DEFCTL $01$=-666
```

Lab 1.4 - Terms Code Lookup

Sample:



Code:

```
00010 ! terms code lookup program
00020 LET DATE$="",OK$=""
00030 READ DATA FROM "" TO IOL=0510
00040 LET TRM=HFN; OPEN (TRM)"termcode"
00050 PRINT 'WINDOW'(5,5,50,20,"Look Up"),'SR'
00060 PRINT 'CS','_BLUE','WHITE','CE'
00070 PRINT "CODE: ",@(10),"DESCRIPTION: ",@(30),"DATE: ",'LF'
00080 PRINT PAD("",50,2,"*")
00090 READ (TRM,END=DONE)IOL=0510
00100 LET OK$+=CODE$
00110 PRINT CODE$,@(10),DESC$,@(30),DATE$:"0000/00/00"; GOTO 0090
00120 DONE:
00130 INPUT @(0,15),"SELECT CODE: ",VAL$:"00"
00140 IF CTL=4 THEN GOTO WRAPUP
00150 IF POS(VAL$=OK$,2)=0 THEN MSGBOX "Not a valid code. Please retry!"; LET VAL$="";
        GOTO 0130
00160 PREINPUT VAL$
00170 WRAPUP:
00180 CLOSE (TRM)
00190 PRINT 'POP'
00500 ! 500 - IOLIST
00510 IOLIST CODE$,DESC$,DATE
```

Lab 1.5 - File Maintenance Enhanced

Objectives:

In this section we will enhance the File Maintenance program by:

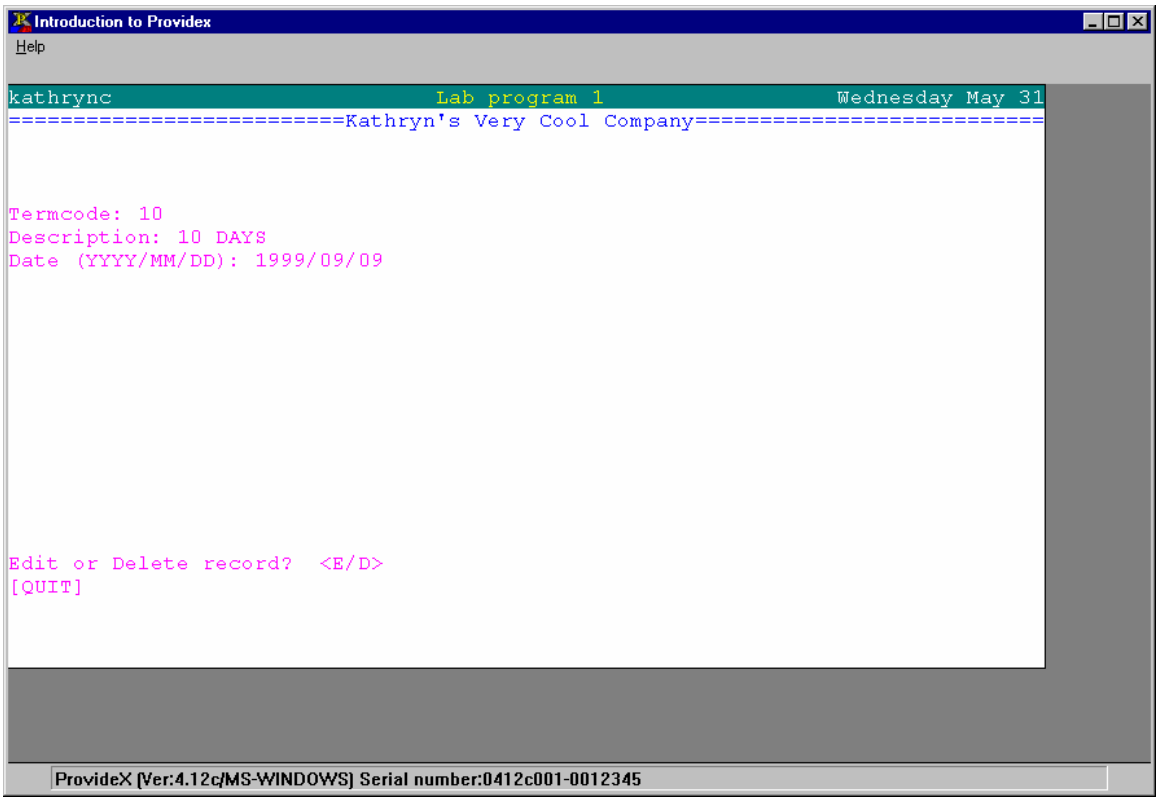
- adding a mouse region using the SETMOUSE directive
- using the PREFIX directive to establish search rules for Program & Data files
- create an .ini file to set application-specific settings
- creating a START_UP program
- creating a desktop shortcut

Use the **SETMOUSE** directive to create a screen area which will respond to a mouse-click:

1. Add the following lines to "LAB1" .
   Note: Ctl value of 4 is already set to pass control to the WRAPUP logic

   00031 SETMOUSE ON
   ...
   00121 PRINT @(0,21),"[QUIT]"; SETMOUSE "[QUIT]"=4

Sample:



Lab 1.5 - File Maintenance Enhanced

Code:

```
00010 LET FL=HFN; OPEN (FL)"termcode"
00020 ! initialization
00030 INIT:
00031 SETMOUSE ON
00040 ! LET %COMPANY$="Kathryn's Very Cool Company"
00041 DEFCTL $01$=-666
00050 PERFORM "gbl_func"
00060 PERFORM "screen_set"
00070 SETCTL 4:WRAPUP
00080 !
00090 WHILE 1
00100 READ DATA FROM "" TO IOL=1010
00110 LET TRM_DATE$="00000000"
00120 INPUT @(0,5),"Termcode: ",TRM_CD$:"00"
00121 PRINT @(0,21),"[QUIT]"; SETMOUSE "[QUIT]"=4
00130 READ (FL,KEY=TRM_CD$,DOM=*NEXT)IOL=1010; GOTO SHOWIT
00140 MSGBOX "Create this record","record not on file","!,YESNO",X$
00150 IF X$="YES" THEN GOSUB UPDATE
00160 CONTINUE
00170 !
00180 SHOWIT:
00190 PRINT @(0,6),"Description: ",PAD(TRM_DESC$,20),@(0,7),"Date (YYYY/MM/DD):
",TRM_DATE$:"0000/00/00",
00200 INPUT @(0,20),"Edit or Delete record? <E/D>",RESP$:"A"
00210 IF RESP$="E" THEN GOSUB UPDATE
00220 IF RESP$="D" THEN GOSUB DELETE
00230 WEND
00240 !
00250 UPDATE:
00251 LET ERR$=""
00260 INPUT EDIT (0,LEN=20,SIZ=10)@(0,6),"Description: ",TRM_DESC$
00270 IF NUL(TRM_DESC$) THEN MSGBOX "Missing Desc","Big Err"; GOTO 0260
00280 INPUT EDIT @(0,7),"Date (YYYY/MM/DD):",TRM_DATE$:"0000/00/00"
00281 CALL "*win/date;validate",TRM_DATE$,ERR$; IF ERR$<>"" THEN MSGBOX ERR$,"Bad
Date"; LET TRM_DATE$="",ERR$=""; GOTO 0280
00290 WRITE (FL,KEY=TRM_CD$)IOL=1010
00300 MSGBOX "Record Update","Voila!"
00310 RETURN
00320 !
00330 DELETE:
00340 REMOVE (FL,KEY=TRM_CD$,DOM=BADTRM)
00350 MSGBOX "Record Deleted!!","Way to go Man!!"
00360 RETURN
00370 !
00380 BADTRM:
00390 MSGBOX "No Such Record!","Whoops!!"; GOTO 0200
00400 ! wrap up
00410 WRAPUP:
00420 PRINT 'CS'
00430 CLOSE (FL)
00440 END
01000 ! 1000 - iolist
01010 IOLIST TRM_CD$,TRM_DESC$,TRM_DATE$
```

Lab 1.5 - File Maintenance Enhanced

In the Windows explorer, under C:\PVXTRAIN create 2 new folders - **Data** and **Progs.**
Select all program files and move to the Progs folder; select the Termcode file and move to the Data folder.

Create a **START_UP** program to run when your applications starts:

1. Use the MSGBOX directive to greet the user.
   Use the ERROR_HANDLER directive to specify a generic error handler.
   Assign a value to %COMPANY$.
   Use the PREFIX directive to specify the DATA and PROGS directories.

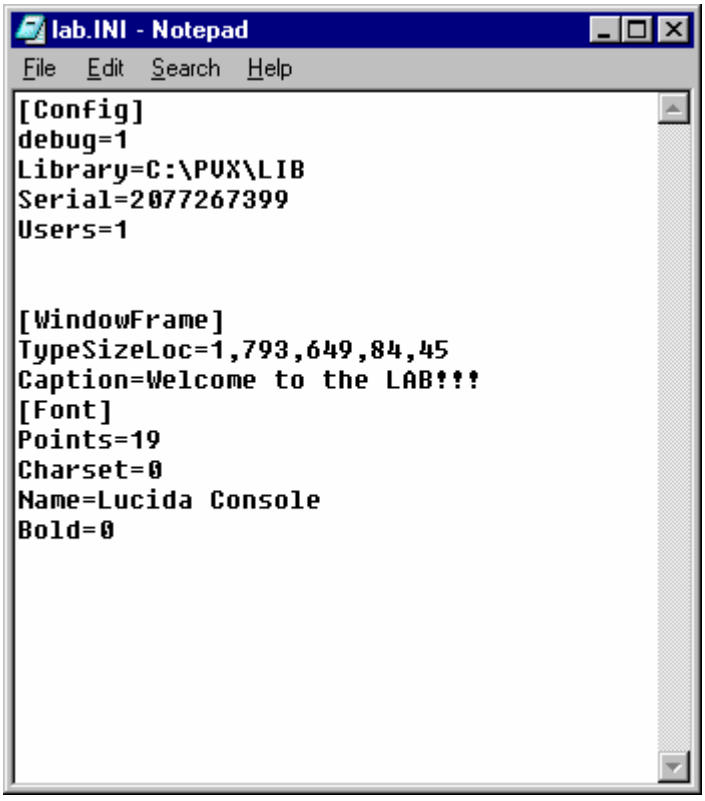
   ```
   00010  MSGBOX "welcome to providex, "+UID+" ! ","Happy Programming :)"
   00020  ERROR_HANDLER "*error"
   00030  LET %COMPANY$="Kathryn's Extremely Cool Company"
   00040  PREFIX "data/ progs/"
   ```

2. Save as "START_UP" in the C:\PVXTRAIN directory

3. Load "LAB1".
   Delete or comment out line 40.

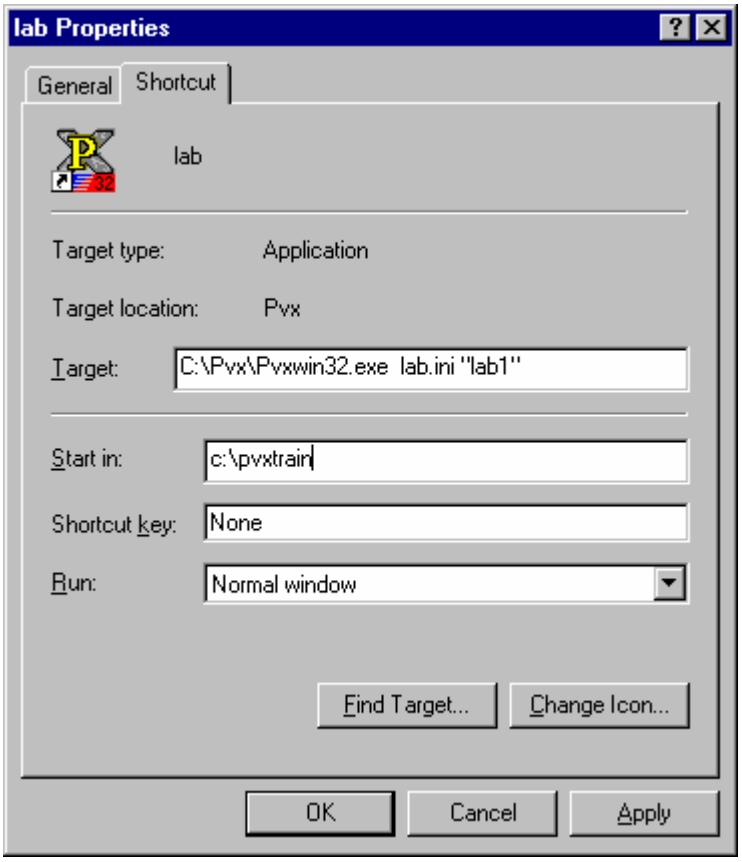
   ```
   00040 ! LET %COMPANY$="Kathryn's Very Cool Company"
   ```

In the Windows explorer, double-click the pvx.ini file to open it in Notepad. Save as "**lab.ini**" in the **C:\Pvxtrain** directory.
Change or Add the Caption as shown; Save.



Lab 1.5 - File Maintenance Enhanced

Create a **Desktop Shortcut** to your application, as shown:



Lab 1.6 - GUI Lookup

Objectives:

In this section, we will look at how to:

- create a graphical version of the Look Up screen
- use the BUTTON directive
- use the LIST_BOX, LIST_BOX LOAD and LIST_BOX READ directives

---

Based on your existing character Look Up program, create a program to display the contents of the TERMCODE file in a listbox; the user will be able to select a code from the listbox by double-clicking.

In Providex CALL "*IT" and load the **$CTL-666** program.
Save as "**$CTL-888**".
Make the following changes to your new program:

1. Insert the following line to use the SETCTL directive:
   00040 SETCTL 4:WRAPUP

2. System parameter **'3D'** must be set for ProvideX to display controls in 3D; add the following code to set the parameter:
   00050 PRINT '3D',; SET_PARAM '3D'

3. Change line 0060 to:
   00060 PRINT 'CS','B?','BLACK','CE' ! Windows standard grey background/black text

   Replace line 0070 with the following:
   00070 BUTTON 4,@(15,12,5,2)="Quit"
   This adds a "QUIT" button with the CTL value of 4 at column **15**, row **15**, width **5**, height **2**.

   Replace line 0080 with the following:
   00080 LIST_BOX 100,@(5,2,25,8),TIP="double-click entry to select",FNT="courier"
   This creates a listbox at column **5**, row **5**, width **30**, height **8**; includes values for the TIP and FNT options.

4. Replace the code following the READ statement with the LIST_BOX LOAD directive:
   00093 LIST_BOX LOAD 100,0,PAD (CODE$,5,1)+PAD (DESC$,12,1)+STR
   (DATE$:"0000/00/00"); GOTO 0090

5. Replace the logic in the DONE label with the following code.
   Use the OBTAIN directive to keep the window active, waiting for user response.
   Use the LIST_BOX READ command to get the selected value; this is passed to VAL$.
   Parse the first 2 characters from VAL$ before passing to the buffer with the PREINPUT statement.
   00120 DONE:
   00130 OBTAIN (SIZ=1)X$
   00140 LIST_BOX READ 100,VAL$
   00150 LET VAL$=VAL$(1,2)
   00160 PREINPUT VAL$

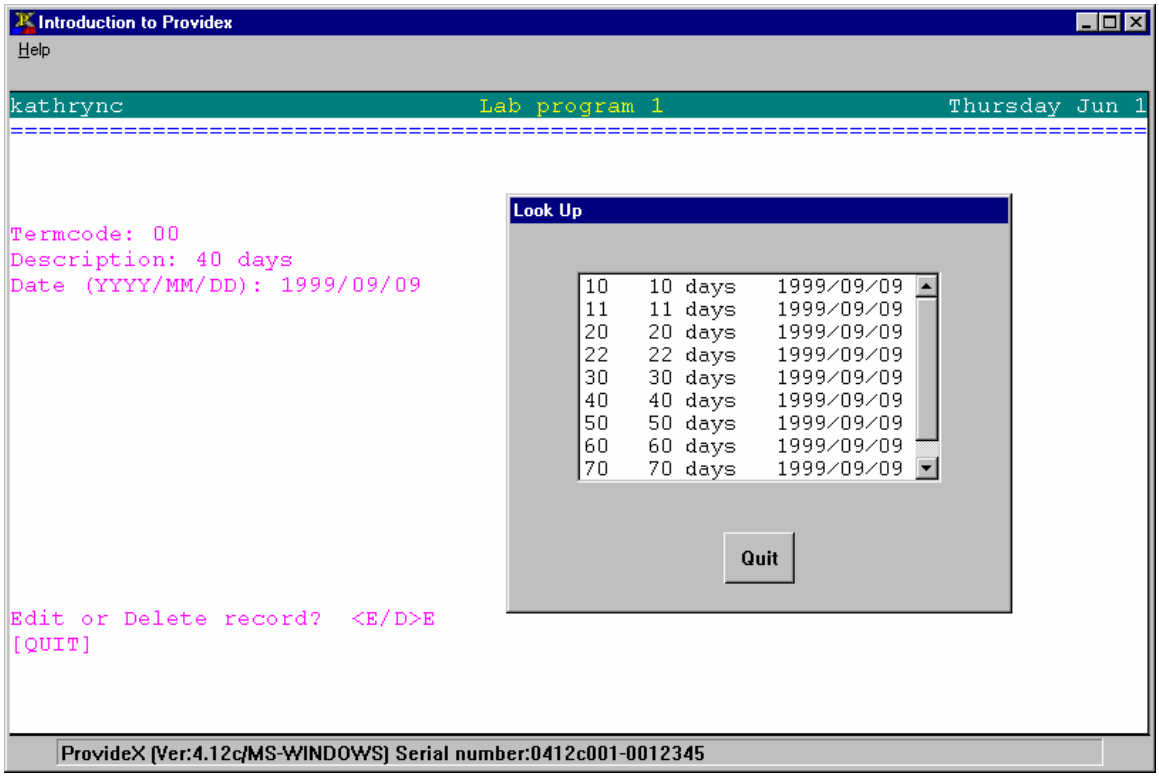
Lab 1.6 - GUI Lookup

6. Save the program.

7. Load "LAB1".
   Make the following change at line 00060:
   00060 DEFCTL 01=-888

8. Save as "LAB1_GUI".

Sample:



Lab 1.6 - GUI Lookup

Code:

lab1_gui:

```
00010 LET FL=HFN; OPEN (FL)"termcode"
00020 ! initialization
00030 INIT:
00031 SETMOUSE ON
00040 ! LET %COMPANY$="Kathryn's Very Cool Company"
00041 DEFCTL $01$=-888
00050 PERFORM "gbl_func"
00060 PERFORM "screen_set"
00070 SETCTL 4:WRAPUP
00080 !
00090 WHILE 1
00100 READ DATA FROM "" TO IOL=1010
00110 LET TRM_DATE$="00000000"
00120 INPUT @(0,5),"Termcode: ",TRM_CD$:"00"
00121 PRINT @(0,21),"[QUIT]"; SETMOUSE "[QUIT]"=4
00130 READ (FL,KEY=TRM_CD$,DOM=*NEXT)IOL=1010; GOTO SHOWIT
00140 MSGBOX "Create this record","record not on file","!,YESNO",X$
00150 IF X$="YES" THEN GOSUB UPDATE
00160 CONTINUE
00170 !
00180 SHOWIT:
00190 PRINT @(0,6),"Description: ",PAD(TRM_DESC$,20),@(0,7),"Date (YYYY/MM/DD):
",TRM_DATE$:"0000/00/00",
00200 INPUT @(0,20),"Edit or Delete record? <E/D>",RESP$:"A"
00210 IF RESP$="E" THEN GOSUB UPDATE
00220 IF RESP$="D" THEN GOSUB DELETE
00230 WEND
00240 !
00250 UPDATE:
00251 LET ERR$=""
00260 INPUT EDIT (0,LEN=20,SIZ=10)@(0,6),"Description: ",TRM_DESC$
00270 IF NUL(TRM_DESC$) THEN MSGBOX "Missing Desc","Big Err"; GOTO 0260
00280 INPUT EDIT @(0,7),"Date (YYYY/MM/DD):",TRM_DATE$:"0000/00/00"
00281 CALL "*win/date;validate",TRM_DATE$,ERR$; IF ERR$<>"" THEN MSGBOX ERR$,"Bad
Date"; LET TRM_DATE$="",ERR$=""; GOTO 0280
00290 WRITE (FL,KEY=TRM_CD$)IOL=1010
00300 MSGBOX "Record Update","Voila!"
00310 RETURN
00320 !
00330 DELETE:
00340 REMOVE (FL,KEY=TRM_CD$,DOM=BADTRM)
00350 MSGBOX "Record Deleted!!","Way to go Man!!"
00360 RETURN
00370 !
00380 BADTRM:
00390 MSGBOX "No Such Record!","Whoops!!"; GOTO 0200
00400 ! wrap up
00410 WRAPUP:
00420 PRINT 'CS'
00430 CLOSE (FL)
```

Lab 1.6 - GUI Lookup

```
00440 END
01000 ! 1000 - iolist
01010 IOLIST TRM_CD$,TRM_DESC$,TRM_DATE$

$CTL-888:
00010 ! terms code lookup program
00020 LET DATE$="",OK$=""
00030 READ DATA FROM "" TO IOL=0510
00031 SETCTL 4:WRAPUP
00032 PRINT '3D',; SET_PARAM '3D'
00040 LET TRM=HFN; OPEN (TRM)"termcode"
00050 PRINT 'WINDOW'(5,5,50,20,"Look Up"),'SR'
00060 PRINT 'CS','B?','BLACK','CE' ! Windows standard grey background/black text
00070 BUTTON 4,@(15,12,5,2)="Quit"
00080 LIST_BOX 100,@(5,2,25,8),TIP="double-click entry to select",FNT="courier"
00090 READ (TRM,END=DONE)IOL=0510
00093 LIST_BOX LOAD 100,0,PAD(CODE$,5,1) +PAD(DESC$,12,1)+
        STR(DATE$:"0000/00/00"); GOTO 0090
00120 DONE:
00130 OBTAIN (SIZ=1)X$
00140 LIST_BOX READ 100,VAL$
00150 LET VAL$=VAL$(1,2)
00160 PREINPUT VAL$
00180 WRAPUP:
00190 CLOSE (TRM)
00200 PRINT 'POP'
00500 ! 500 - IOLIST
00510 IOLIST CODE$,DESC$,DATE$
```