USB File List

# Objectives

* Set up a GUI project that displays the file list from a USB device attached to the Disco board

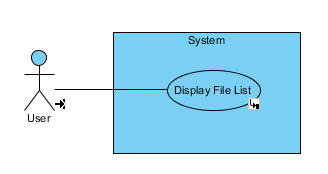
# Description

In this lab you will be putting together many pieces from previous labs. You will be given descriptions of the design and you will be required to build the system as designed. There will be steps given in the lab procedure but you will be responsible for using material that you learned in previous labs to complete this lab.

For each part in this lab, save a new thread file Thread\_X.c where X is the part number. Make sure you remove the old file from your project and add the new one!!!! Save the file in the src directory.

In this part of the lab you will set up a VB project that communicates with the Disco board and receives a file name. The file name will be displayed on the PC GUI. The FS thread will not read from the USB device in this part, it will just send a stored file name.

The following figures show the design of the system for this part. The user interface is very simple. It includes a button and a list box. You will simply click on the button and the system will show the file name in the list box. The sequence diagram shows the messaging within the system. The system will have a state machine in the state dependent control object. Notice that when the button is pressed the system will go into the List state and begin sending filenames to the PC GUI. In this part we will just send one file name. So that the PC GUI knows when the file is to starting and stopping the system will send a message before it began sending the file list and a different message when it is done sending the file list. This way the PC GUI will know when the file list begins and ends.



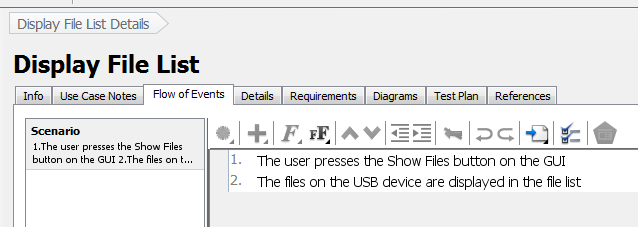


Figure 1: Use case diagram and scenario

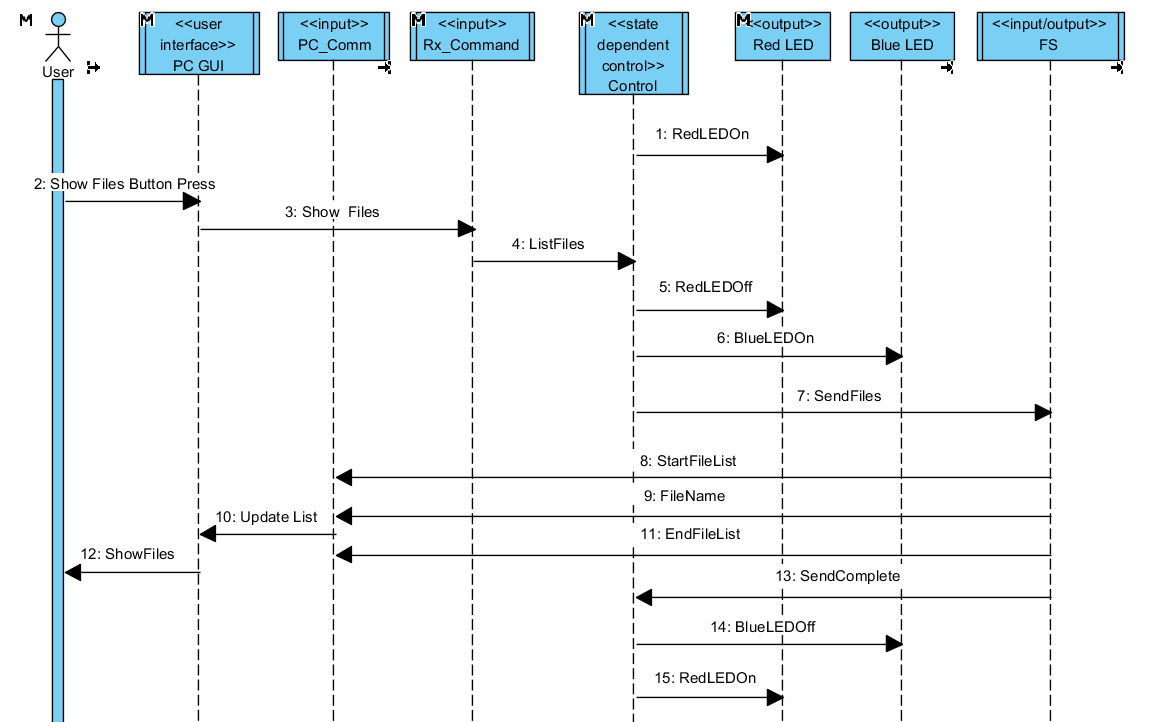


Figure 2: Sequence diagram

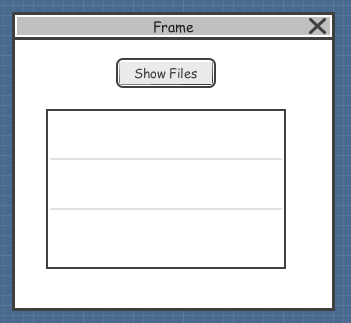


Figure 3: PC GUI

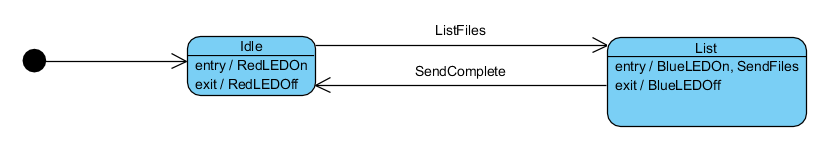


Figure 4: State diagram

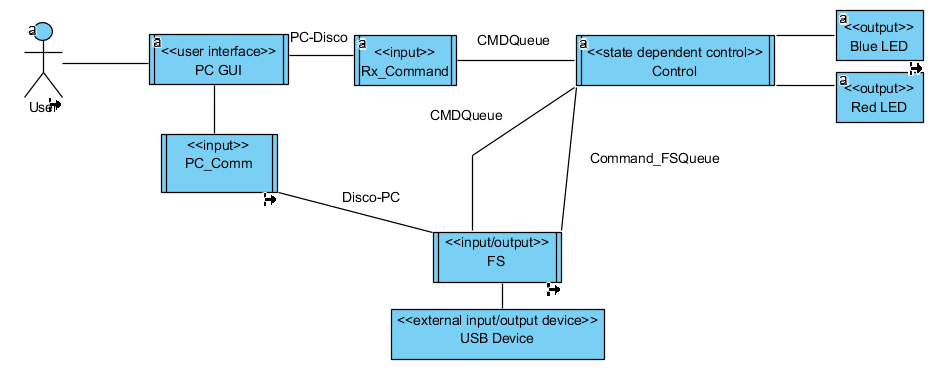


Figure 5: Communication diagram showing threads and queues

## Build the PC GUI

* Start a new VB Form project.
* Build the GUI like Figure 1. The Button will have the name Show\_Files and the ListBox will have the name File\_List. Also add a SerialPort object with the correct COM port number. If you want the files to be in alphabetical order you can change the ListBox property Sorted to True.
* Use the following code for the GUI.

Imports System.Threading

Public Class Form1

Delegate Sub ListBoxDelegate(ByVal command As Integer, ByVal myStr As String)

Dim ListBoxDel As New ListBoxDelegate(AddressOf ListBoxDelMethod)

Dim PC\_Comm As New Thread(AddressOf PC\_Comm\_method)

Dim ShowFiles As Integer = 1

Dim StartFileList As Integer = 2

Dim EndFileList As Integer = 3

Dim ShowFilesStr As String = "1"

Dim StartFileListStr As String = "2"

Dim EndFileListStr As String = "3"

Private Sub Form1\_Load(sender As Object, e As EventArgs) Handles MyBase.Load

'Try to open the seial port

Try

SerialPort1.Open()

Catch

Debug.WriteLine("Failed to open serial port")

End Try

' Make this a background thread so it automatically

' aborts when the main program stops.

PC\_Comm.IsBackground = True

' Set the thread priority to lowest

PC\_Comm.Priority = ThreadPriority.Lowest

' Start the thread

PC\_Comm.Start()

End Sub

' Delegate function that accesses the ListBox object

' command = 2 - clear the contents of the ListBox

' command = 3 - add the string to the ListBox

Public Sub ListBoxDelMethod(ByVal command As Integer, ByVal myStr As String)

If command = StartFileList Then

File\_List.Items.Clear()

ElseIf command = EndFileList Then

File\_List.Items.Add(myStr)

End If

End Sub

' Thread that monitors the receive items on the serial port

Private Sub PC\_Comm\_method()

Dim str As String

While 1

If SerialPort1.IsOpen Then

Try

str = SerialPort1.ReadLine() ' Wait for start string

Catch ex As Exception

End Try

'String.Compare return values:

' Less than zero: strA precedes strB in the sort order.

'Zero" strA occurs in the same position as strB in the sort order.

'Greater than zero: strA follows strB in the sort order.

If Not String.Compare(str, StartFileListStr) Then

' Received a StartFileList response

' clear the list

' Use the delegate to access the ListBox

File\_List.Invoke(ListBoxDel, StartFileList, "")

' get next string

Try

str = SerialPort1.ReadLine() ' read file name

Catch ex As Exception

End Try

While String.Compare(str, EndFileListStr)

' The read string is a file name and not the EndFileList

File\_List.Invoke(ListBoxDel, EndFileList, str)

Try

str = SerialPort1.ReadLine() ' read file name

Catch ex As Exception

End Try

End While

' While loop ends when a 3 is received

End If

End If

End While

End Sub

Private Sub Show\_Files\_Click(sender As Object, e As EventArgs) Handles Show\_Files.Click

If SerialPort1.IsOpen Then

' Send Show\_Files command

SerialPort1.Write(ShowFilesStr, 0, 1)

End If

End Sub

End Class

In this code you can see that when the Show\_Files button is clicked it will send a "1" to the Disco board. This is the Show\_Files message. The Thread\_0 is always waiting for a receive command so it is waiting for a "2". This is the StartFileList message. Finally, the Thread\_0 will keep putting strings that it receives into the ListBox until it receives a "3" which is the EndFileList message.

## Build the Disco code

* From the course Blackboard site download the file BaseProject\_XX\_XX\_XX.zip. The XXs will be numbers for the date of the file.
* Extract the content to a folder.
* Open the folder and double click the file BaseProject.uvprojx. This will open the project in uVision.
* Save Thread.c to Thread\_1.c and put Thread\_1.c in your project and remove Thread.c.
* The following are threads/functions you already know how to set up. Here is a summary of their contents
* RX\_Command thread
* Receives the Show\_Files message and sends ListFiles to the Control thread.
* Use :

#define Show\_Files\_char "1"

enum commands{

ListFiles,

SendComplete,

SendFiles

};

* Control Thread
* Implements the state machine, uses Process\_Event()
* Turns on and off the Red LED and Blue LED
* Sends SendFiles command to the FS thread on the queue Comman\_FSQueue
* Use:

// State Machine definitions

enum state{

NoState,

Idle,

List,

};

The FS thread will be sending information back to the PC GUI. The PC GUI will be using a command like:

str = SerialPort1.ReadLine()

This ReadLine command reads the "contents of the input buffer up to the first occurrence of a NewLine value." A NewLine value is "\n". Therefore, every time the Disco code sends something to the PC GUI the string has to end with a "\n".

* FS Thread
* Initialize the USB thumb drive and get it ready to get the file names off the thumb drive.
* Infinite while loop that waits for the command SendFiles in the queue Comand\_FSQueue.
* When it receives the SendFiles command it will
* send the StartFileList message to the PC
* send the file name to the PC
* send the EndFileList message to the PC
* send SendComplete command back to the Control thread on the CMDQueue
* To send the StartFileList and EndFileList commands back to the PC use:

char \*StartFileList\_msg = "2\n";

char \*EndFileList\_msg = "3\n";

UART\_send(StartFileList\_msg,2); // Send start string

UART\_send(EndFileList\_msg,2); // Send start string

* To send the test file name to the PC use:

char \*file\_name = "TestFileName.txt";

int len;

len = strlen(file\_name);

UART\_send(file\_name,len);

UART\_send("\n",1); // this is the VB string terminator "\n"

* Demonstrate that you are able to press the Show\_Files button and see the file name "TestFileName.txt" displayed on the GUI.

In this part you will need to add code to the FS thread to make it read from the USB device and send the files to the PC GUI. In previous labs you learned how to read the file names from a USB device and send them to the serial port.

The FS Thread will be the only thread that accesses the USB device. You should have all the code in it for doing anything with the USB device. Do not open the USB device in one thread and use it in another thread. Also, in this project this is the only thread that will write to the PC through the serial port. If you want to have multiple threads write to the serial port then you should have another thread that receives commands on a queue and then only that thread will send to the PC.

* From a previous lab, add the code that gets the USB device ready and mounted. At that point the USB is ready to use. Add this before the while loop.
* In the while loop, after sending the StartFileList message, add the code that gets the file names from the USB device and then sends them to the serial port.
* Demonstrate that you can display all the files on the USB device on the PC GUI.

In this part you will add a button to the PC GUI that will send the selected file name to the Disco board.

* Add a button to the PC GUI that will send the selected file name to the Disco board. Use:

Dim b(1) As Byte

b(0) = 0

' A value of negative one (-1) is returned if no item is selected

If Not (File\_List.SelectedIndex = -1) Then

SerialPort1.Write("4", 0, 1)

SerialPort1.Write(File\_List.SelectedItem)

SerialPort1.Write(b, 0, 1) ' New Line character at the end of the string

End If

* The "4" is the command that tells the Disco board that the next string is the file name. Make a variable for the "4" similar to what was given in a previous part. You should not hard code values like the "4".
* On the Disco board use the UART\_receivestring command in the Rx\_Command function to save the string to a character array after receiving the "4".
* Demonstrate that you have received the selected file on the Disco board.