Column #112 August 2004 by Jon Williams:

More Control from the Couch

With Bluetooth technology, remote control blues may be a thing of the past.

Yep, I'm still a real man, alright. I still live in the great state of Texas, I still drink milk right out of the carton, I still leave the toilet seat up and, in addition to the five remotes I have to run the electronics in my entertainment center, I can now control anything else in my home from my Pocket PC. Yeah, buddy, enough control to make Tim Allen grunt with manly-man joy.

How is this possible from a Pocket PC? It's possible with FlexiPanel. The FlexiPanel architecture – available on many platforms – is an interesting approach to the client-server paradigm. The FlexiPanel server provides the interface (through a Bluetooth connection) that runs on the FlexiPanel client (a specialized browser application that runs on the PPC). The user is able to interact with the client interface, and this interaction is made available to the BASIC Stamp microcontroller through a serial connection and [optional] status lines.

For our applications, the FlexiPanel BASIC Stamp Edition is the server, and will connect to the BASIC Stamp using 19.2 kBaud serial connections with flow-control. The most popular client is for the Pocket PC, but more are available and many planned. If you have the correct cell phone, for example there is a client for it. How cool would it be to control things from your cell phone?

Let me start by saying that the BASIC Stamp side of this project is pretty easy – it's a straight serial connection and the system design is quite straightforward. The tougher part is creating the client UI, and as I get lots of requests to cover that kind of thing we're going to devote most of our time there. So, our project is dirt-simple: we're just going to control a few LEDs from our PPC. Not horribly exciting – but fraught with possibilities because if we can control LEDs, we can control almost anything. I, for example, will be using this kind of project around Halloween (my favorite day of the year) to replace my keychain remotes for activating yard displays.

Creating the FlexiPanel Client UI

Okay, you know the drill: Plan your work, work your plan. Our plan, then, is to control three LEDs from a Pocket PC. Okay ... how, exactly? Well, let's make it simple and provide a button for each LED, where pressing the button will toggle the LED's state. And just for fun, let's add two more buttons: one for "All On" and another for "All Off." Of course, we'll also need some sort of feedback to the display since we could be operating our device remotely. The range will depend on the Bluetooth module in your PPC, but could be up to 300 feet.

If you're like me, you probably have a quad-lined pad handy for sketching schematics. After doodling a bit, and aware that the Pocket PC display is laid out like a playing card, I decided to put the three individual buttons across the top of the display, with the "All" buttons and status indication beneath them, and using the full width.

With an idea of where we're going we need to fire-up the FlexiPanel Designer program. Figure 112.1 shows the program interface.

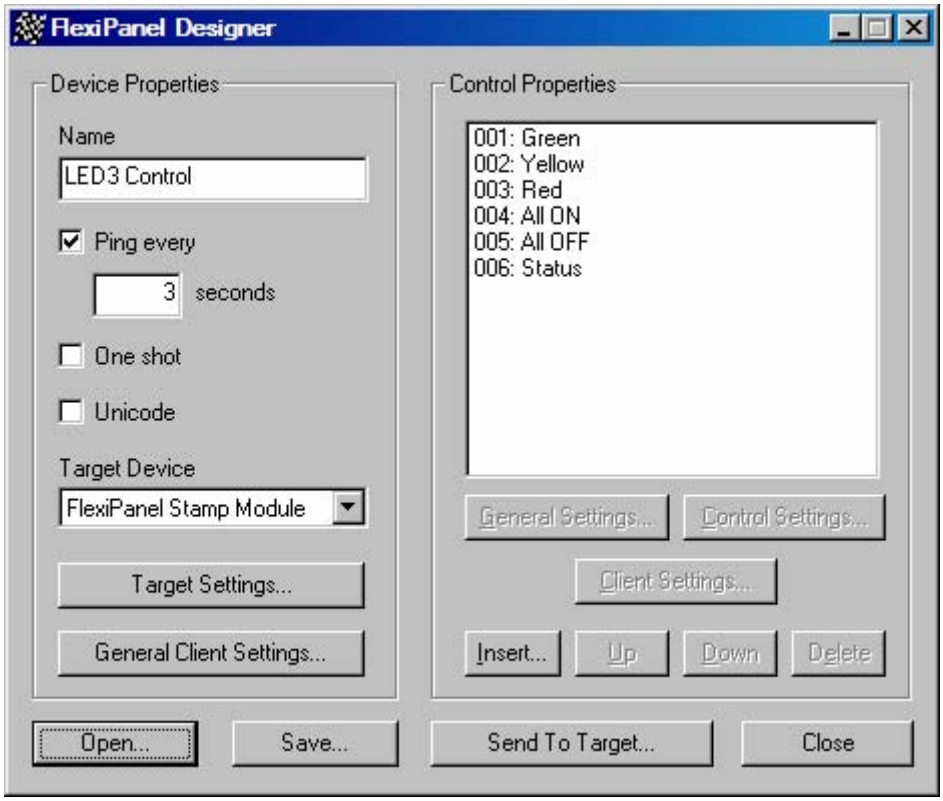

Figure 112.1: FlexiPanel Designer Main Window

Author's Note: Creating a custom UI for the FlexiPanel client is fairly involved, so I will not be discussing all of the features of the FlexiPanel Designer. I will, however, focus on those features that I've found are important for achieving the stated goal.

Starting with the Device Properties section, we'll give our client a name that gets displayed in the PPC title bar. Let's use "LED3 Control" – boring, yes, but obvious.

The next thing we need to do, specifically when our server controller is the BASIC Stamp, is to enable the Ping property. This will cause the server to verify the connection with the client. Three seconds is recommended by Hoptroff.

In the Target Device drop-down, select FlexiPanel Stamp Module. The other choice is Simulation which is useful when quick-testing UI designs (this works only if you have a Bluetooth-enabled PC). By selecting the BASIC Stamp as our target, the FlexiPanel software will generate a ROM image for the FlexiPanel server in the form of PBASIC source code. We'll get to more of this later.

Now click the Target Settings button and you'll get the dialog shown in Figure 112.2. This dialog lets us select the BASIC Stamp module used in our project. The reason Designer wants to know is so that it can provide the correct $STAMP directive and baud mode constant for the generated source code. The other thing we'll want to check in this dialog is the selection that lets us use the Mode pin as a client-connected signal. By using the Mode pin to indicate the connection, our BASIC Stamp program is able to deal with its presence, or more specifically, when it drops. Some projects will need a "fail safe" behavior and using the Mode pin as an connection indicator makes this very simple.

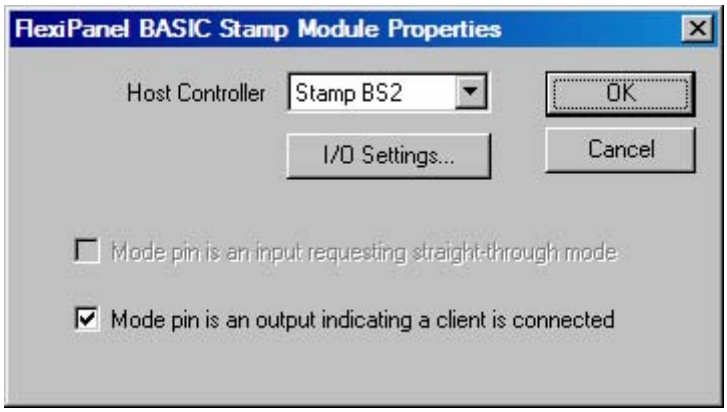

Figure 112.2: BASIC Stamp Module Properties Dialog

With the BASIC Stamp module type selected, click on the I/O Settings button. This dialog (Figure 112.3) lets assign BASIC Stamp pins to the IO point on the FlexiPanel module. Again, this is used for the generation of PBASIC source code, and we're certainly free to modify the connections later if we need to.

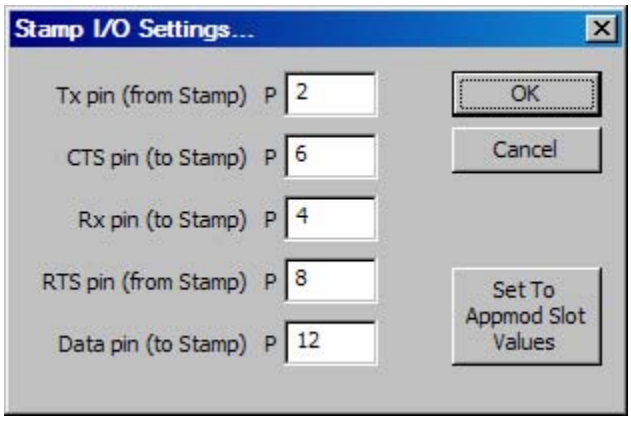

Figure 112.3: BASIC Stamp I/O Settings Dialog

Alright, enough of the preliminaries, let's get going with some controls. In the Control Properties section click on the Insert button. This will bring up the Insert Control dialog (Figure 112.4). Select Button, and then click OK. Since this is a new control you'll be presented with the General Control Settings dialog (Figure 112.5). Change the name property to "Green" (we'll use this button to toggle our green LED) and for some pizzazz, check Preferred Color and enter the values 0, 255, and 0. After clicking OK you should see the new button in the controls list.

Finally, we need to manually place and size the button. If we don't, the button will be a half-screen wide and about 57 pixels tall (this is the standard button size for the FlexiPanel client) – a bit chunky in my opinion, so let's fix it. In the Control Properties section click on Client Settings, and then click on Pocket PC Settings. With the Client Setting dialog (Figure 112.6) we can specify the attributes for the control. In my projects I have found that the size, position, and page attributes are important to specify; the others can be left as is.

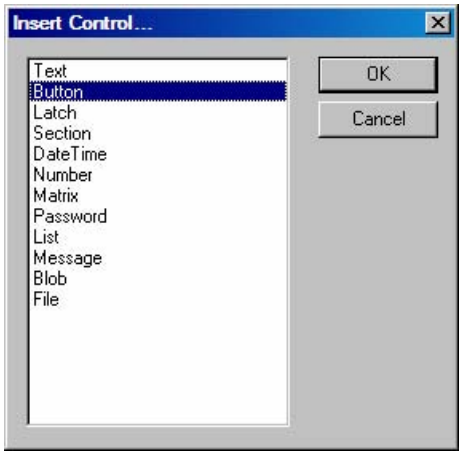

Figure 112.4: Insert Control Dialog

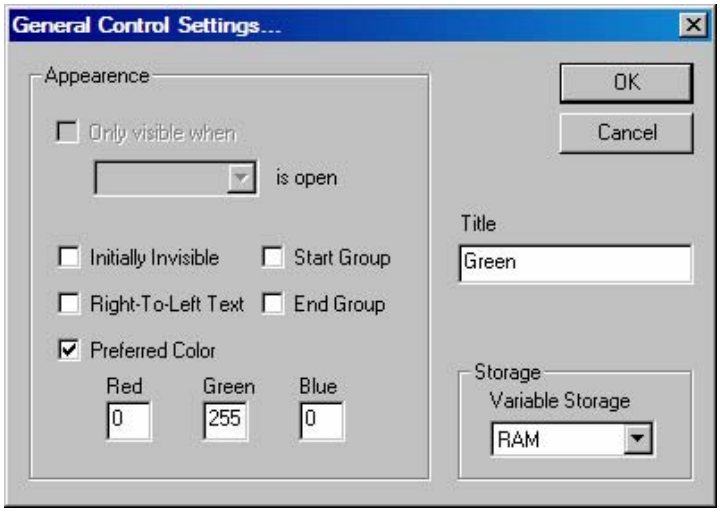

Figure 112.5: General Control Settings Dialog

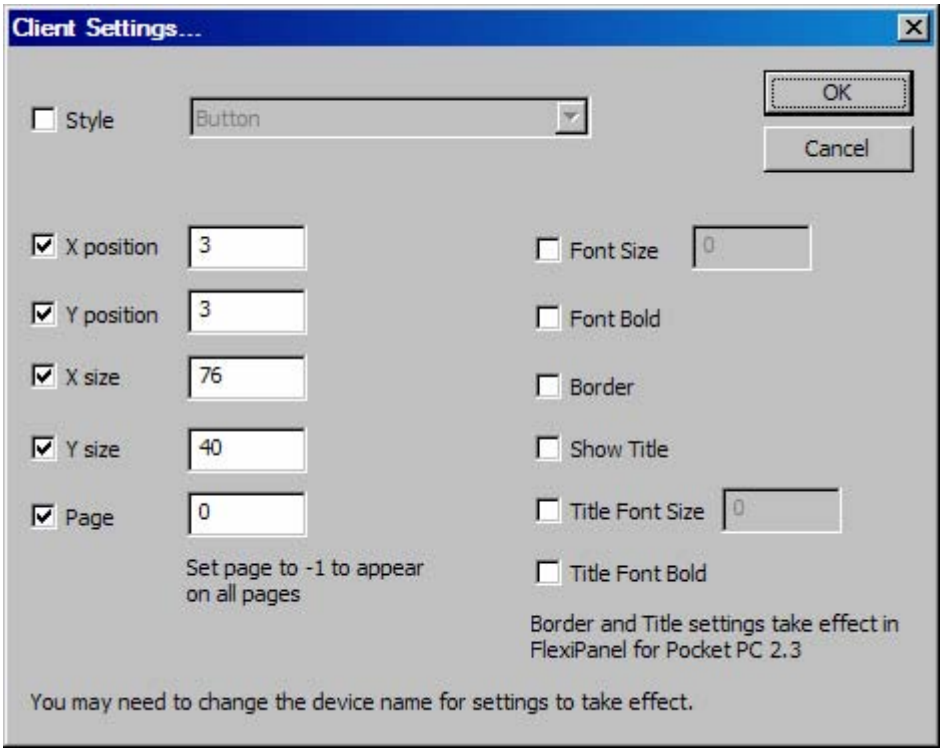

Figure 112.6: Control Client Settings Dialog

Control	Title	R-G-B	X_{pos}, Y_{pos}, X_{size} x Y_{size}, Page
Button	Green	255-0-0	3, 3, 76 x 40, 0
Button	Yellow	255-255-0	82, 3, 76 x 40, 0
Button	Red	0-0-255	161, 3, 76 x 40, 0
Button	All On	Default	3, 46, 234 x 40, 0
Button	All Off	128-128-128	3, 89, 234 x 40, 0
Text	Status	Default	3, 132, 234 x 40, 0

Table 112.1: Control Client Settings Values

Using the steps we just covered and the information in Table 112.1, add four more buttons to the project. Finally, add a Text control. In the Text Control Properties dialog (Figure 112.7), enter "Status: ???" in the Initial Text field. What you should see is that when you increase the length of the Initial Text field, the Maximum Length field will automatically update. The value in the Maximum Length field should always be one greater than the longest string to be displayed (the extra byte is a terminating zero). And make sure that Modifiable is not checked. The status string is an output only and we don't want the user to change this on the client screen.

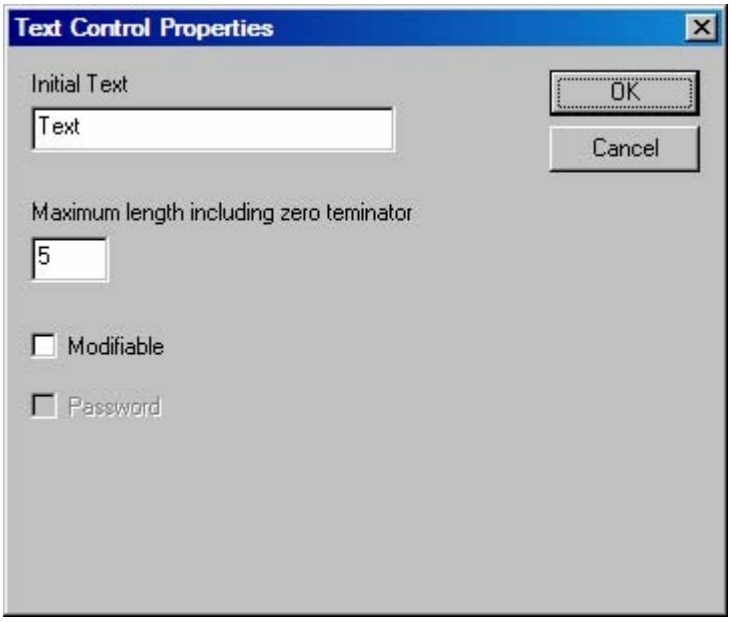

Figure 112.7: Text Control Properties Dialog

Before we forget or are struck with one of those inconvenient power outages, click the Save button and give the file the name "LED3_Panel.FxP" (note that you need to manually add the extension – but this may change in future version of the Designer).

Targeting the BASIC Stamp Microcontroller

With the client interface designed we can move on to the next phase of the project which is the BASIC Stamp module's interaction with the FlexiPanel server. The Designer program actually does a nice chunk of the work for us. Before exiting Designer, click the Send to Target button. A dialog that shows server resource allocation will be displayed – just click OK to move past it. We'll now be presented with a standard Save As dialog for the PBASIC program. You can use the default name (same as project) if you'd like; just be aware that if you change the name of the program you need to add the correct file extension.

If you haven't already done so, start the BASIC Stamp Editor and find the file we just saved. When you open it the first thing you'll be struck by is the large DATA table. This is the ROM image for the server and is actually downloaded by the BASIC Stamp module. For small

programs we can leave the ROM image and loader in our control program. For bigger projects we'll want to create a control that is separate from the loader.

Before we can run the program the FlexiPanel server needs to be connected to the BASIC Stamp module. The FlexiPanel is conveniently designed to plug into the AppMod header on a Parallax BOE, but if you don't have one, you can plug it into a standard breadboard and make the connections with hookup wire. If you're using the BOE AppMod header to hold the FlexiPanel module be very careful with alignment. The Bluetooth radio and microcontroller will face the breadboard area, and the module will plug into the left side of the AppMod header; the pins marked Vss, P0, P2, P4, etc. The schematic for the project circuit is shown in Figure 112.8a, and the FlexiPanel module connections are shown in Figure 112.8b.

When everything is connected and powered-up, run the program generated by the FlexiPanel Designer. What this will do is download the ROM image for the client to the FlexiPanel server. At this point no other code is active, so we'll just get the controls, but won't be able to interact with them. When the FlexiPanel download is complete, the DEBUG window will display "Acknowledge: ROM." At this point we can connect the client.

Make sure the latest FlexiPanel client is installed on your Pocket PC, and then run it (from the Programs menu). Click the Connect button and you'll see the Bluetooth Console screen as shown in Figure 112.9. Now, you probably won't see an icon for LED3 Control at this point, so click on the refresh button at the bottom of the screen. This will cause the PPC to search for other Bluetooth devices. When the LED3 Control icon appears, click it and after a moment you should see a screen with buttons laid out as in Figure 112.10.

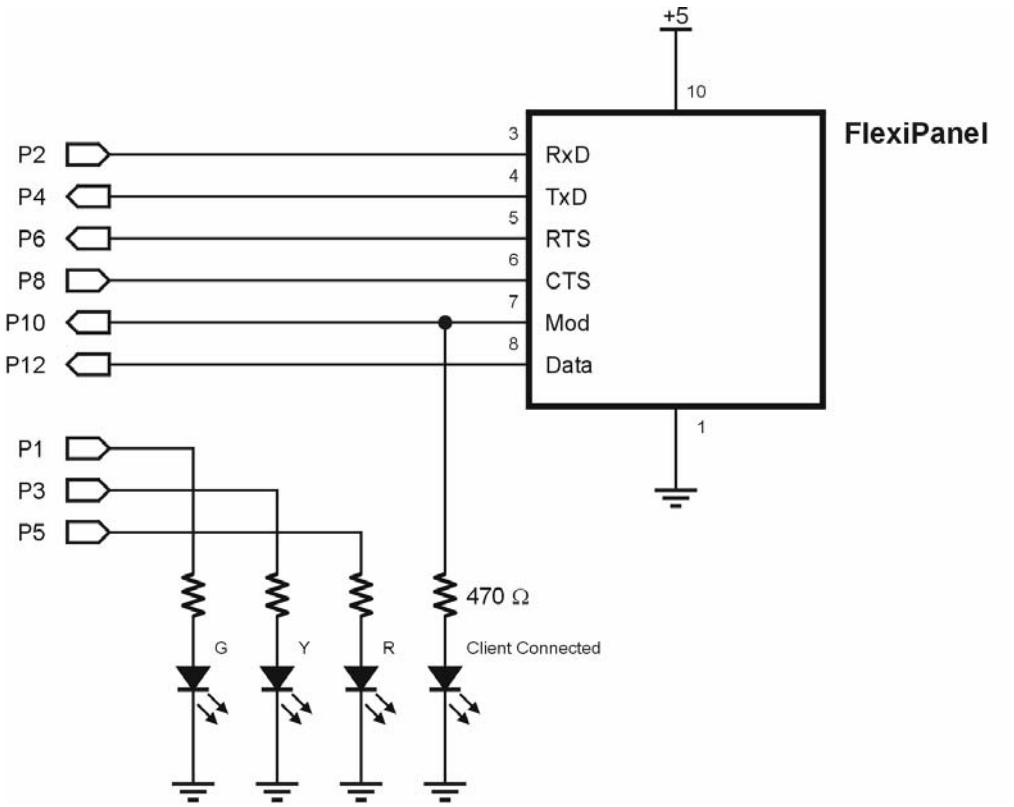

Figure 112.8a: Project Schematic

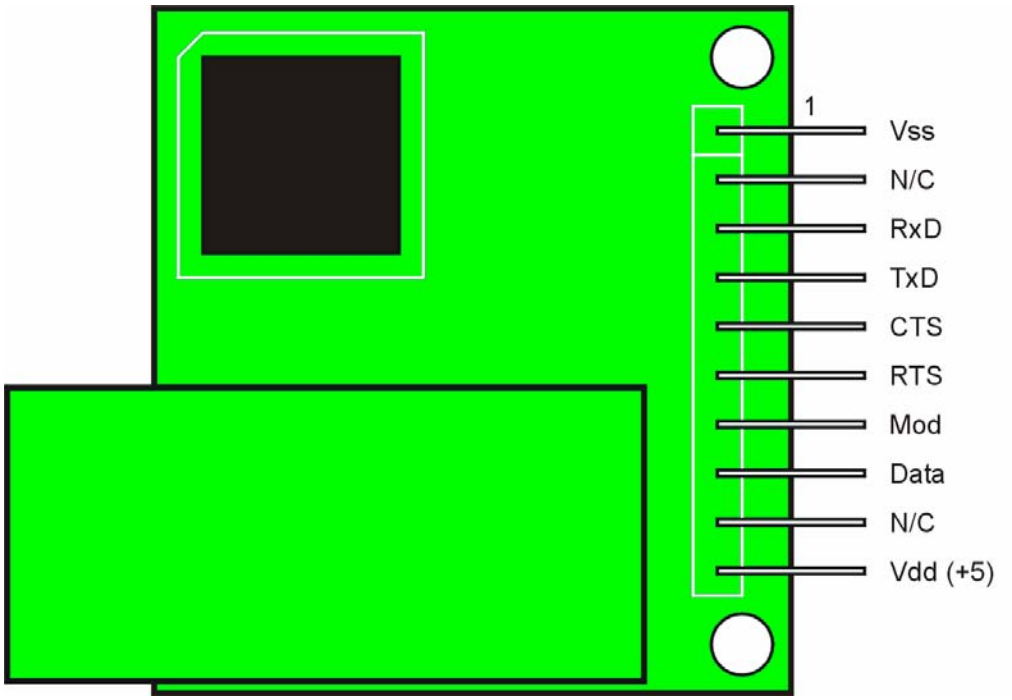

Figure 112.8b: FlexiPanel Module Connections

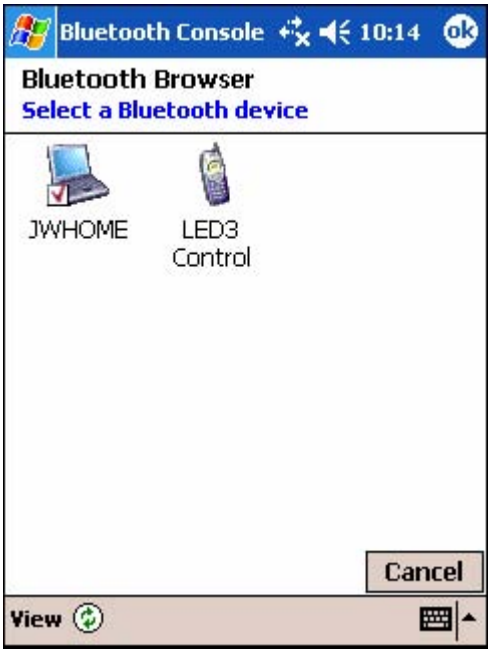

Figure 112.9: PPC Bluetooth Console

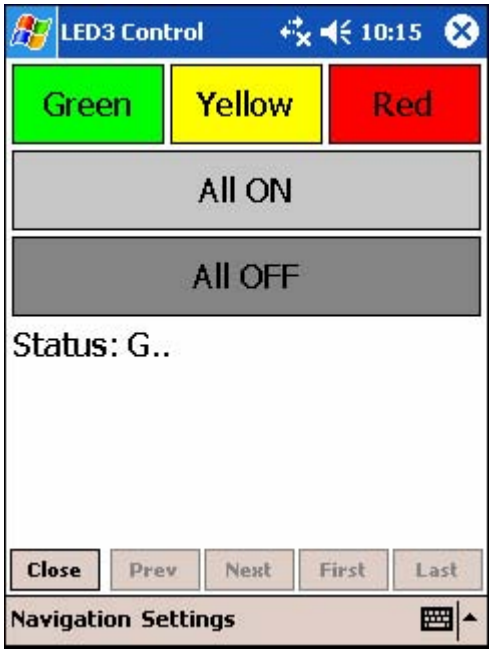

Figure 112.10: FlexiPanel Client

Once we're satisfied with the look of the layout we can modify the PBASIC code to interact with the client controls. You'll see that Designer does a lot of the work for us, providing useful definitions and snippets of code that we can easily adapt for our own use. As I write I am working with version 2.2 of the Designer, and I fully expect future versions to add even more conveniences.

One of the changes I make to the auto-generated code is a progress indicator for the ROM image download process. This can take quite some time with complex interfaces, and the progress indicator lets me know things are moving. In an embedded application that doesn't have access to the DEBUG terminal, I would consider using an extra IO point as a "Loading" indicator. Here's my final version of the loader:

```
  DEBUG CLS, "Programming ROM...", CR
  SEROUT TX\CTS, Baud, [$81, $31]
  FOR idx = 0 TO $30F
    DEBUG CRSRX, 0, DEC idx, " of ", DEC $30F
    READ (FxpROM + idx), cData
    SEROUT TX\CTS, Baud, [cData]
  NEXT
  DEBUG CR, CR, "Awaiting acknowledge...", CR
  SERIN  RX\RTS, Baud, [STR text\3]
  DEBUG "Acknowledge: ", STR text\3, CR
```

After the DEBUG window is cleared, the programming command ($81) is sent to the FlexiPanel module, along with the number of 16-byte blocks to expect. Note that flow-control serial communications is used. This is required as the FlexiPanel module has to handle communications with the BASIC Stamp module as well as the Bluetooth radio module.

A FOR-NEXT loop handles dumping the ROM image to the FlexiPanel server. What I added was the line that shows the current progress of the download. At the end of the loop the module will respond with "ROM" to indicate that we're ready to go.

In this project we're taking advantage of all FlexiPanel connections. One of those connections, as specified in the Designer, is the Mode pin. We've told the FlexiPanel module to use this pin as an output and an indicator when the client is connected. You can see in the schematic that an LED is hung off this output, and it is also connected to P10 on the BASIC Stamp module. Here's the code that monitors that pin:

```
Get_Client:
  DEBUG CR, "Waiting for client...", CR
  DO : LOOP UNTIL (HasClient = Yes)
  DEBUG CLS,
        "Client is Connected", CR,
        "Use buttons to toggle LEDs"

  GOSUB Update_Status
```

The only work is the DO-LOOP line that waits for Mode output (called HasClient our program) to go high. When it does we can update the client with the current status of the LEDs. When first run, the LEDs will be off of course, but that doesn't mean that we can't disconnect and then reconnect later. Let's see how to send information to the text control of our client.

```
Update_Status:
  text(0) = "." + (25 * OUTS.LOWBIT(GrnLED))
  text(1) = "." + (43 * OUTS.LOWBIT(YelLED))
  text(2) = "." + (36 * OUTS.LOWBIT(RedLED))
  text(3) = 0
  SEROUT TX\CTS, Baud, [SetData, ID_Status,
                        "Status: ", STR text\4]
  RETURN
```

This code looks a little trickier than it is – its purpose is to display the current state of the LEDs, using a dot for off, or the first letter of the LED color for on. Let's just go through the green LED. If that LED is off, the OUTS bit that controls it will be zero, hence the expression in the parenthesis will evaluate as zero. When the LED is on and the OUTS bit for the green LED is one, the expression in the parenthesis will evaluate as 25. Yes, the BASIC Stamp can actually evaluate the expression "." + 25. It can do this because the "." is represented by its ASCII value (46) internally. So when the green LED is on the array element called text(0) will be assigned a value of 71, the ASCII value for 'G'. Once the status string is constructed (along with trailing zero) it is transmitted to the FlexiPanel using the SetData command.

Now, I'm betting that some of you are wondering why the use of the PIN definition for LED control outputs didn't help us much here. The reason is that the compiler will on evaluate PIN definitions as OUTx when the pin name is on the left side of an expression, but when it's on the right it will evaluate as an input. When used as an index value like in the code above, it will also evaluate as a constant, so in the end, the compiler will evaluate OUTS. LOWBIT(GrnLED) as OUT1.

Okay, the display is updated, so now we can wait for some input from the user and then deal with it accordingly. The Data pin from the FlexiPanel module will go high when a control on the client has changed. When that happens we will ask the client which control changed. Here's how we do that:

```
Main:
  DO
    IF (HasClient = No) THEN Get_Client
  LOOP UNTIL (HasData = Yes)

Get_Changed_Control:
  SEROUT TX\CTS, Baud, [GetMod]
  SERIN  RX\RTS, Baud, [ctrl]
```

The initial DO-LOOP in this section is setup to wait for the HasData line to go high. Inside the loop we can check the HasClient line to make sure that it hasn't dropped. In other applications, we could also put our normal "background" code in this loop.

When a control is changed on the client and the HasData line goes high, we'll drop out of the loop and then transmit the GetMod instruction to the FlexiPanel. The module will reply with the ID number of the control that changed.

The listing below shows how to handle a couple of the controls; the rest are identical (so the code has been removed for clarity). The SELECT-CASE structure lets us jump right to the code for the changed control. In the case of FlexiPanel buttons, we need to poll them for their status. While this may seem redundant (since the FlexiPanel already told us that the button had changed) but it's quite useful as the client will hold onto the button press until we have the opportunity to read the button. When we do, the "pressed" status will be cleared.

A value of $FF (aliased as Pressed in the program) tells us the button has indeed been pressed. For the single LED buttons, we use the pressed indication to toggle the LED status. For the "All On" and "All Off" buttons the handle code follows the button function. When the button is dealt with, we update the display and go back to the top where we wait for another user input.

```
Poll_Control:
  SELECT ctrl
    CASE ID_Green
      SEROUT TX\CTS, Baud, [GetData, ID_Green]
      SERIN  RX\RTS, Baud, [cData]
      IF (cData = Pressed) THEN
        TOGGLE GrnLED
      ENDIF

    ' snip

    CASE ID_AllOff
      SEROUT TX\CTS, Baud, [GetData, ID_AllOff]
      SERIN  RX\RTS, Baud, [cData]
      IF (cData = Pressed) THEN
        GrnLED = IsOff
        YelLED = IsOff
        RedLED = IsOff
      ENDIF
  ENDSELECT
  GOSUB Update_Status
  GOTO Main
```

You Have the Control

Let me be honest and say that we've gone through – at a reasonably quick clip – what is in fact a fairly complicated product. This is the reason I kept the BASIC Stamp end of things simple. What I encourage you to do is load up the project files (download from the Nuts & Volts web site) and play with them to get the hang of things.

If you don't have a Bluetooth connection on your PC, you can add one with an USB-Bluetooth adapter (I use the DBT-120 from D-Link). This will help the interface design cycle by using Simulation as the target for the Designer. In this mode, the client interface is downloaded from the PC, and interface messages from the client are displayed in a dialog.

Spend some time with this – it may take a few days to get the hang of things, but I think it's worth it. And keep in mind that the FlexiPanel is a new product, so we can certainly expect improvements as more customers put it to use in various applications. I have certainly found the folks at Hoptroff very open to ideas and suggestions, and helpful when I was in a pinch. I'm sure you will too.

Okay, it's your turn ... now there is no need to get up from the easy-chair to turn something on or off. What will you control in your home? I have a pretty long list for mine....

Until next time, Happy Stamping.

(additional) Resources

Société HOPTROFF
www.hoptroff.com
www.flexipanel.com

```
' =============================================================================
'
'   File...... FxP_LED3.BS2
'   Purpose... Simple controls with FlexiPanel module
'   Author.... Jon Williams
'   E-mail.... jwilliams@parallax.com
'   Started...
'   Updated... 14 JUN 2004
'
'   {$STAMP BS2}
'   {$PBASIC 2.5}
'
' =============================================================================


' -----[ Program Description ]---------------------------------------------

' This program uses all available FlexiPanel connections.  By monitoring
' the HasClient input the BASIC Stamp can put its program into a fail-safe
' mode should the connection get dropped.


' -----[ Revision History ]------------------------------------------------


' -----[ I/O Definitions ]-------------------------------------------------

' FlexiPanel connections

TX              PIN     2                       ' to FxP RxD
RX              PIN     4                       ' from FxP TxD
CTS             PIN     6                       ' from FxP RTS
RTS             PIN     8                       ' to FxP CTS
HasClient       PIN     10                      ' from FxP Mod
HasData         PIN     12                      ' from FxP Data

' Circuit connections

GrnLED          PIN     1
YelLED          PIN     3
RedLED          PIN     5


' -----[ Constants ]------------------------------------------------------

#SELECT $STAMP
  #CASE BS2, BS2E, BS2PE
    Baud        CON     32                      ' 19.2 kBaud
  #CASE BS2SX, BS2P
    Baud        CON     110                     ' 19.2 kBaud
```

```
#ENDSELECT

Yes             CON     1
No              CON     0

IsOn            CON     1
IsOff           CON     0

Pressed         CON     $FF                     ' button was pushed

GetData         CON     $01                     ' Get control data
SetData         CON     $02                     ' Set control data
GetMod          CON     $03                     ' Get modified control
ShowMsg         CON     $04                     ' Show message

ID_Green        CON     $01                     ' Green control ID
ID_Yellow       CON     $02                     ' Yellow control ID
ID_Red          CON     $03                     ' Red control ID
ID_AllOn        CON     $04                     ' All_On control ID
ID_AllOff       CON     $05                     ' All_Off control ID
ID_Status       CON     $06                     ' Status control ID


' -----[ Variables ]------------------------------------------------------

idx             VAR     Word                    ' loop counter
cData           VAR     Byte                    ' data to/from FxP
text            VAR     Byte(4)                 ' text IO
ctrl            VAR     Byte                    ' control id


' -----[ EEPROM Data ]----------------------------------------------------

FxpROM          DATA    $06, $02, $00, $00, $00, $00, $4C, $45, $44, $33
                DATA    $20, $43, $6F, $6E, $74, $72, $6F, $6C, $00, $00
                DATA    $00, $00, $00, $03, $41, $01, $38, $00, $AC, $00
                DATA    $00, $00, $C3, $00, $17, $00, $01, $00, $01, $00
                DATA    $00, $00, $00, $00, $01, $02, $01, $42, $01, $00
                DATA    $00, $00, $C4, $00, $01, $00, $DC, $00, $18, $00
                DATA    $01, $00, $01, $00, $00, $00, $00, $00, $01, $02
                DATA    $01, $42, $02, $00, $00, $00, $DD, $00, $02, $00
                DATA    $F2, $00, $15, $00, $01, $00, $01, $00, $00, $00
                DATA    $00, $00, $01, $02, $01, $42, $03, $00, $00, $00
                DATA    $F3, $00, $03, $00, $02, $01, $0F, $00, $01, $00
                DATA    $01, $00, $00, $00, $00, $00, $01, $02, $01, $42
                DATA    $04, $00, $00, $00, $03, $01, $04, $00, $1C, $01
                DATA    $19, $00, $01, $00, $01, $00, $00, $00, $00, $00
                DATA    $01, $02, $01, $42, $05, $00, $00, $00, $1D, $01
                DATA    $05, $00, $35, $01, $18, $00, $0C, $00, $0C, $00
                DATA    $00, $00, $00, $00, $01, $02, $01, $54, $06, $00
                DATA    $00, $00, $02, $00, $00, $00, $5A, $63, $05, $00
```

```
                DATA    $00, $00, $47, $72, $65, $65, $6E, $04, $00, $00
                DATA    $00, $00, $FF, $00, $00, $00, $02, $00, $00, $00
                DATA    $5A, $63, $06, $00, $00, $00, $59, $65, $6C, $6C
                DATA    $6F, $77, $04, $00, $00, $00, $FF, $FF, $FF, $00
                DATA    $00, $02, $00, $00, $00, $5A, $63, $03, $00, $00
                DATA    $00, $52, $65, $64, $04, $00, $00, $00, $FF, $00
                DATA    $FF, $00, $00, $01, $00, $00, $00, $5A, $06, $00
                DATA    $00, $00, $41, $6C, $6C, $20, $4F, $4E, $00, $02
                DATA    $00, $00, $00, $5A, $63, $07, $00, $00, $00, $41
                DATA    $6C, $6C, $20, $4F, $46, $46, $04, $00, $00, $00
                DATA    $80, $80, $80, $00, $00, $02, $00, $00, $00, $5A
                DATA    $6D, $06, $00, $00, $00, $53, $74, $61, $74, $75
                DATA    $73, $04, $00, $00, $00, $0D, $00, $00, $00, $53
                DATA    $74, $61, $74, $75, $73, $3A, $20, $3F, $3F, $3F
                DATA    $00, $00, $01, $00, $FF, $00, $01, $02, $00, $00
                DATA    $01, $00, $FF, $10, $01, $F5, $00, $00, $01, $00
                DATA    $FF, $20, $01, $2D, $00, $00, $01, $00, $FF, $30
                DATA    $01, $15, $00, $00, $01, $00, $FF, $60, $01, $0C
                DATA    $00, $00, $01, $01, $FF, $00, $01, $34, $00, $00
                DATA    $01, $01, $FF, $10, $01, $F5, $00, $00, $01, $01
                DATA    $FF, $20, $01, $2D, $00, $00, $01, $01, $FF, $30
                DATA    $01, $15, $00, $00, $01, $01, $FF, $60, $01, $0C
                DATA    $00, $00, $01, $02, $FF, $00, $01, $63, $00, $00
                DATA    $01, $02, $FF, $10, $01, $F5, $00, $00, $01, $02
                DATA    $FF, $20, $01, $2D, $00, $00, $01, $02, $FF, $30
                DATA    $01, $15, $00, $00, $01, $02, $FF, $60, $01, $0C
                DATA    $00, $00, $01, $03, $FF, $00, $01, $92, $00, $00
                DATA    $01, $03, $FF, $10, $01, $F5, $00, $00, $01, $03
                DATA    $FF, $20, $01, $2D, $00, $00, $01, $03, $FF, $30
                DATA    $01, $15, $00, $00, $01, $03, $FF, $60, $01, $0C
                DATA    $00, $00, $01, $04, $FF, $00, $01, $C1, $00, $00
                DATA    $01, $04, $FF, $10, $01, $F5, $00, $00, $01, $04
                DATA    $FF, $20, $01, $2D, $00, $00, $01, $04, $FF, $30
                DATA    $01, $15, $00, $00, $01, $04, $FF, $60, $01, $0C
                DATA    $00, $00, $01, $05, $00, $30, $01, $28, $00, $00
                DATA    $01, $01, $00, $00, $01, $03, $00, $00, $01, $01
                DATA    $00, $10, $01, $03, $00, $00, $01, $01, $00, $20
                DATA    $01, $4C, $00, $00, $01, $01, $00, $30, $01, $28
                DATA    $00, $00, $01, $02, $00, $00, $01, $52, $00, $00
                DATA    $01, $02, $00, $10, $01, $03, $00, $00, $01, $02
                DATA    $00, $20, $01, $4C, $00, $00, $01, $02, $00, $30
                DATA    $01, $28, $00, $00, $01, $03, $00, $00, $01, $A1
                DATA    $00, $00, $01, $03, $00, $10, $01, $03, $00, $00
                DATA    $01, $03, $00, $20, $01, $4C, $00, $00, $01, $03
                DATA    $00, $30, $01, $28, $00, $00, $01, $05, $00, $00
                DATA    $01, $03, $00, $00, $01, $05, $00, $10, $01, $59
                DATA    $00, $00, $01, $05, $00, $20, $01, $EA, $00, $00
                DATA    $01, $04, $00, $00, $01, $03, $00, $00, $01, $04
                DATA    $00, $10, $01, $2E, $00, $00, $01, $04, $00, $20
                DATA    $01, $EA, $00, $00, $01, $04, $00, $30, $01, $28
                DATA    $00, $00, $01, $06, $00, $00, $01, $03, $00, $00
```

```
                DATA    $01, $06, $00, $10, $01, $84, $00, $00, $01, $06
                DATA    $00, $20, $01, $EA, $00, $00, $01, $06, $00, $30
                DATA    $01, $28, $00, $00, $01, $00, $00, $00, $01, $01
                DATA    $00, $00, $01, $06, $00, $40, $01, $00, $00, $00
                DATA    $01, $05, $00, $40, $01, $00, $00, $00, $01, $04
                DATA    $00, $40, $01, $00, $00, $00, $01, $01, $00, $40
                DATA    $01, $00, $00, $00, $01, $02, $00, $40, $01, $00
                DATA    $00, $00, $01, $03, $00, $40, $01, $00, $00, $00
                DATA    $00, $00, $00, $00, $00, $00, $00, $00, $00, $00
                DATA    $00, $00, $00, $00


' -----[ Initialization ]----------------------------------------------------

Reset:
  LOW GrnLED                                    ' start with LEDs off
  LOW YelLED
  LOW RedLED
  PAUSE 50                                      ' let FxP initialize

  ' Program FxP ROM

  DEBUG CLS, "Programming ROM...", CR
  SEROUT TX\CTS, Baud, [$81, $31]
  FOR idx = 0 TO $30F                           ' loop through ROM image
    DEBUG CRSRX, 0, DEC idx, " of ", DEC $30F   ' show progress
    READ (FxpROM + idx), cData
    SEROUT TX\CTS, Baud, [cData]
  NEXT
  DEBUG CR, CR, "Awaiting acknowledge...", CR
  SERIN  RX\RTS, Baud, [STR text\3]
  DEBUG "Acknowledge: ", STR text\3, CR

Get_Client:
  DEBUG CR, "Waiting for client...", CR
  DO : LOOP UNTIL (HasClient = Yes)
  DEBUG CLS,
        "Client is Connected", CR,
        "Use buttons to toggle LEDs"

  GOSUB Update_Status


' -----[ Program Code ]------------------------------------------------------

Main:
  DO
    IF (HasClient = No) THEN Get_Client         ' abort if client drops
  LOOP UNTIL (HasData = Yes)                    ' wait for user input

Get_Changed_Control:
```

```
  SEROUT TX\CTS, Baud, [GetMod]                 ' get changed control ID
  SERIN  RX\RTS, Baud, [ctrl]

Poll_Control:
  SELECT ctrl                                   ' poll changed control
    CASE ID_Green
      SEROUT TX\CTS, Baud, [GetData, ID_Green]
      SERIN  RX\RTS, Baud, [cData]
      IF (cData = Pressed) THEN
        TOGGLE GrnLED
      ENDIF

    CASE ID_Yellow
      SEROUT TX\CTS, Baud, [GetData, ID_Yellow]
      SERIN  RX\RTS, Baud, [cData]
      IF (cData = Pressed) THEN
        TOGGLE YelLED
      ENDIF

    CASE ID_Red
      SEROUT TX\CTS, Baud, [GetData, ID_Red]
      SERIN  RX\RTS, Baud, [cData]
      IF (cData = Pressed) THEN
        TOGGLE RedLED
      ENDIF

    CASE ID_AllOn
      SEROUT TX\CTS, Baud, [GetData, ID_AllOn]
      SERIN  RX\RTS, Baud, [cData]
      IF (cData = Pressed) THEN
        GrnLED = IsOn
        YelLED = IsOn
        RedLED = IsOn
      ENDIF

    CASE ID_AllOff
      SEROUT TX\CTS, Baud, [GetData, ID_AllOff]
      SERIN  RX\RTS, Baud, [cData]
      IF (cData = Pressed) THEN
        GrnLED = IsOff
        YelLED = IsOff
        RedLED = IsOff
      ENDIF
  ENDSELECT

  GOSUB Update_Status
  GOTO Main

  END
```

```
' -----[ Subroutines ]------------------------------------------------------

Update_Status:
  text(0) = "." + (25 * OUTS.LOWBIT(GrnLED))    ' rebuild status bits
  text(1) = "." + (43 * OUTS.LOWBIT(YelLED))
  text(2) = "." + (36 * OUTS.LOWBIT(RedLED))
  text(3) = 0

  ' send to client

  SEROUT TX\CTS, Baud, [SetData, ID_Status, "Status: ", STR text\4]
  RETURN
```