Bluetooth Remote Control...

From Your Mobile Phone

Richard Hoptroff

Published (in abridged form)
in Elektor Electronics, Summer 2004
Bluetooth Remote Control… From Your Mobile Phone

Flick through any electronics magazine – professional or hobbyist – and you will see a wide range of single board computers and microcontroller boards. For many applications, they make product development so much simpler than it used to be. Attach a few auxiliary components and a control panel, write the computer program, and you’re finished.

What makes the process easier is that the computer board is programmable, so one off-the-shelf component can be applied to many tasks. Could this concept be taken even further? A few auxiliary components will always be needed in any product, but what about the control panel? Couldn’t an off-the-shelf programmable component be made to replace custom control panels on electronic devices?

FlexiPanel’s remote user interface module (Figure 1) does just this. Using Bluetooth radio, it asks a remote device – a mobile phone or handheld computer, perhaps – to create the required control panel on its behalf (Figure 2). The module has a Class 1 radio, so the remote device can be up to 100m away. The module operates at TTL levels, and a standalone RS232 device (Figure 3) will soon also be in production.
A user may connect to the appliance at any time using any suitable device. The device will display the required control panel, but its appearance may vary according to the remote device used. For example, compare the same user interface on a mobile phone (Figure 17, Figure 19) and a handheld computer (Figure 18, Figure 20). On the handheld’s touch sensitive screen, large buttons are used. The phone, on the other hand, assigns “hot keys” to controls on the user interface display.

The software on the remote devices is the same for each application and does not require customization or re-installation. It is freely downloadable from www.FlexiPanel.com. At the time of writing, Pocket PCs, Windows PCs, and Smartphones (e.g. SPV E200 from Orange) software are supported. Software for Palm Operating System and Java phones supporting the JABWT standard (e.g. Nokia 6600 and Sony Ericsson P900) is due for release in June 2004.

**A Bluetooth Protocol**

Bluetooth is a 2.4GHz digital radio communication protocol developed and licensed by Ericsson. Serving the “personal area network”, Bluetooth devices can come and go *ad hoc*. In contrast, the WiFi protocol, operating at the same frequency, is more suited to longer-term wireless infrastructure, with each individual node needing to be assigned a fixed IP (internet protocol) address.
Thanks to Bluetooth headsets, Bluetooth is now solidly entrenched in the mobile phone market. Intel intends to incorporate Bluetooth into its Centrino 2 chipset, to be launched in Autumn 2004. Not only will this allow PCs to connect wirelessly to printers, etc, but it will boost the growth of VoIP (voice over internet protocol), i.e. phone calls over the internet.

The Bluetooth standard provides interfaces for a wide range of communications protocols, from a simple serial port to audio. Like many higher-level protocols such as OBEX file exchange, FlexiPanel sits on top of the serial port emulation layer of the Bluetooth protocol stack (Figure 4). It is not part of the “official” Bluetooth standard. However, the standard is relatively open in that anyone is free to create software for remote devices, and product-side components such as the FlexiPanel module are manufactured under license, just like any Bluetooth radio module. The first FlexiPanel products were software libraries to provide remote control for Windows applications and high-end embedded systems.

From the electronic product’s perspective, the FlexiPanel module is a peripheral providing graphical user interface services. It maintains a list of the controls required by the product, and the current state of the controls. (The ten types of control available are shown in Table 1.) The product can update a control at any time. If a user modifies a control, the product is notified. The pin functions of the FlexiPanel module are detailed in Table 2.
Three Projects In One

Developing applications of the FlexiPanel module is simple enough that three mini-projects can be detailed within this article. They all use a BASIC Stamp BS2p and the Board of Education development board featured in *Elektor* in September 1999. Parallax Inc, who supply the Basic Stamp, also distribute the FlexiPanel module.

The BASIC Stamp can be programmed using the BASIC programming language from any PC computer using a serial cable. The same link is used to program the control panel into the FlexiPanel module for each of the three mini-projects.

The first example project is a temperature logger. The second is a secure access control device. The last is a robot controller with route tracking.

The BASIC programs and FlexiPanel designer data files used in these projects are available from *Elektor Readers’ Services* as free software downloads.

Temperature Logger

Measuring environmental variables such as temperature is vital for quality control in the food, chemicals and drugs industries. In this project, the FlexiPanel module logs the output of a temperature sensor. It is a simple project but sufficient to illustrate the four steps in the design process:

- Designing the electronic circuit
- Design of the user interface
- Programming the user interface into the FlexiPanel module
- Writing the BASIC program to interface between the circuit and the FlexiPanel module.

First, the circuit is designed. In addition to the BASIC Stamp and the FlexiPanel module, a real time clock (a Philips PC8583 and 32768Hz crystal) and a digital temperature sensor (a Dallas DC1921) are connected over an I2C bus. The circuit is shown as a schematic diagram in Figure 5 and on the prototyping area of the Board of Education in Figure 6.

Next, the user interface is designed using the PC-based FlexiPanel Designer software freely available from www.FlexiPanel.com. Four controls are required:

- a readout of the current temperature
- a readout of the current time
- a table of historical temperatures
- a control to set the time

Once controls are created using FlexiPanel Designer (Figure 7), their appearance may be simulated and tested on remote devices, provided the PC is Bluetooth enabled.
When the user interface design is complete, it is programmed into the FlexiPanel module. Pressing the Send To Target button in FlexiPanel Designer creates a BASIC program. When this is run in the BASIC Stamp editor (supplied with the BASIC Stamp), the Stamp programs the user interface into the FlexiPanel module’s memory. The FlexiPanel module acknowledges successful programming with an “Acknowledge: ROM” message (Figure 8) and remote devices may connect to view the user interface.

The final step is to write a runtime program for the BASIC Stamp to interface between the electronics and the FlexiPanel module. To make this step easy, the BASIC program created by FlexiPanel Designer contains commented-out example code showing exactly how to read and write control values.

The runtime BASIC program for the Temperature Logger is shown in Listing 1. After initialization, the program reads the temperature and the time each second. These values are written to the temperature, time and temperature history controls. Then, the program checks to see whether the user has set the clock time. If so, the new time is read and programmed into the real time clock.

In operation, the controls appear on a Smartphone as shown in Figure 9. The controls update once every five seconds. Figure 10 shows a temperature history full-screen view after a brief tour in the refrigerator. To the author’s surprise, the
project maintained radio contact throughout, despite being confined by the steel
case of the refrigerator.

**Access Controller**

The second mini-project is a secure access controller for garage doors, safes, *etc.*

It is essentially a password entry system, but has the following advantages:

- each user has a separate password, and a log is kept of who accessed when
- no custom transmitter is needed – any suitable mobile phone or handheld
  computer may be used
- the lack of any visible components makes the system vandal proof and
does not attract burglars

In addition to the BASIC Stamp and the FlexiPanel module, a real time clock (the
same Philips PC8583 and 32768Hz crystal) is connected over an I2C bus and a
relay is used to provide an isolated switch for opening the electric lock. The
circuit is shown as a schematic diagram in Figure 11 and on the prototyping area
of the Board of Education in Figure 12. The circuit illustrated is suitable for low
voltage circuits. For mains-voltage lock releases, a higher power relay would be
required along with more robust connection to the external circuit. In particular, a
PCB with screw-terminals should be used to connect to mains voltages.
In *FlexiPanel Designer*, a user interface is created containing:

- three password controls, one each for Alice, Bob and Clare
- the current time
- a table of who accessed when
- a control to set the time

When the user interface has been programmed into the FlexiPanel module, the BASIC Stamp is loaded with the runtime program shown in Listing 2. After initialization, the program updates the clock display in the user interface. Then it tests whether a password lock has been opened or closed. If the state of one of the locks has changed, the electric lock is controlled appropriately and the access log is updated. Finally it checks to see if the user has adjusted the clock time.

In operation, the controls appear on a Smartphone as shown in Figure 13. The controls update once per second. Figure 14 shows the access log, showing who opened the lock when. The log is stored in Flash memory, so even after power loss, the log is retained.

**Tracking Robot**

The final mini-project is a robot controller. In some senses, this is a traditional remote controller. It differs, however, in being able to send information back to
the handheld device. By using an onboard compass, a route trace is recorded and reported back to the handheld unit.

The Board of Education is mounted on the BoE-Bot robot superstructure available from Parallax Inc. This has motorized wheels which may be controlled by pulse width modulation direct from the BASIC Stamp. An I2C electronic compass (the Devantech CMPS03 from Milford Instruments) is also used. The circuit is shown as a schematic diagram in Figure 15 and on the prototyping area of the Board of Education in Figure 16.

In *FlexiPanel Designer*, a user interface is created containing:

- compass bearing display
- latching pushbuttons for stop, forward, reverse, left and right
- a table showing the route traced by the robot

When the user interface has been programmed into the FlexiPanel module, the BASIC Stamp is loaded with the runtime program shown in Listing 3. After initialization, the program tests to see what kind of motor control pulse it is supposed to output. Then it reads the compass and writes the bearing to the bearing display and the route tracker.
In operation, the controls appear on a Smartphone as shown in Figure 17. Note the numbers on the right hand side: these correspond to the keypad buttons which may be used for real-time control of the robot. Figure 18 shows the robot track as an X-Y plot. For comparison, Figure 19 and Figure 20 show the same user interface when displayed on a Pocket PC.

Readers wishing to take this mini-project further should consider the following points:

- The FlexiPanel module can generate a signal indicating when a remote unit is connected. The robot could automatically halt if it went out of range of the remote unit.

- The compass is significantly influenced by the surrounding metal and direct currents. These influences may be counteracted by local calibration of the compass.

**A Universal Controller**

The FlexiPanel module is an interesting development providing a user interface for electronic projects on a wide range of existing devices. The fact that no custom development is required on the remote devices simplifies product development significantly. My wife, for one, looks forward to the day that she
can “hide that ugly hi-fi in the cupboard” and control it remotely using a mobile phone.
Figures and Tables

Figure 1 – The FlexiPanel module.

(Available as BthModule.jpg 1071 x 1168 pixels)
Figure 2 – A FlexiPanel in operation. The electronic application asks the remote device to display a user interface for it.

*(Available as a Microsoft Word Drawing)*
Figure 3 – Prototype RS232 version of the FlexiPanel Bluetooth module.

(Shown uncased.)

(Available as RS232Module.jpg 1156 x 1801 pixels)
Figure 4 – FlexiPanel in the Bluetooth Protocol Stack

(Available as a Microsoft Word Drawing)
Figure 5 – Schematic diagram of the temperature data logger.

(Available as a Microsoft Word drawing)
Figure 6 – Prototyped circuit for the Temperature Logger.

The FlexiPanel module is mounted vertically, just above the green Power LED.

(Available as the file FxPDesigner.jpg 467 x 403 pixels)
Figure 7 – Screenshot of FlexiPanel Designer for the Data Logger project.

(Available as the file FxPDesigner.jpg 467 x 403 pixels)
Figure 8 – The BASIC Stamp editor programming the FlexiPanel module.

*(Available as the file ProgramModule.jpg 450 x 337 pixels)*
Figure 9 – The Temperature Logger user interface on a Smartphone.

(Available as the file SmPhDatLogCtls.jpg 1716 x 2281 pixels)
Figure 10 – The Temperature Logger temperature history.

(Available as the file SmPhDatLogChart.jpg 1619 x 2201 pixels)
Figure 11 – Schematic diagram of the Access Controller.

(Available as a Microsoft Word drawing)
Figure 12 – Prototyped circuit for the Access Controller.

The red wires lead to the low-voltage electric lock release.

*(Available as the file AccessPlan.jpg 2195 x 1606 pixels)*
Figure 13 – The Access Controller user interface on a Smartphone.

Bob has entered his password and his padlock icon is in the open position.

*(Available as the file SmtPhAccessCtls.jpg 1650 x 2244 pixels)*
Figure 14 – Access Controller log displayed on a Smartphone.

*(Available as the file SmtPhAccessLog.jpg 1651 x 2199 pixels)*
Figure 15 – Schematic diagram of the route tracking robot.

(Available as a Microsoft Word drawing)
Figure 16 – Prototyped circuit for the Tracking Robot.

The Board of Education is mounted on the Boe-Bot from Parallax Inc.

The motors should be connected to socket X4 at the top right of the board.

*(Available as the file AccessPlan.jpg 2272 x 1704 pixels)*
Figure 17 – The Tracking Robot user interface on a Smartphone.

(Available as the file SmtPhRobotCtls.jpg 1642 x 2222 pixels)
Figure 18 – The Tracking Robot route trace displayed on a Smartphone.

(Available as the file SmtPhRobotRoute.jpg 1720 x 2284 pixels)
Figure 19 – The Tracking Robot user interface on a Pocket PC.

(Available as the file PPCRobotButtons.jpg 1704 x 2272 pixels)
Figure 20 – The Tracking Robot route trace displayed on a Pocket PC.

(Available as the file PPCRobotTrace.jpg 2272 x 1704 pixels)
<table>
<thead>
<tr>
<th>Logical control</th>
<th>Example depiction on a remote client</th>
<th>Function / value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Button</td>
<td>Button</td>
<td>Single-press event</td>
</tr>
<tr>
<td>Latch</td>
<td>Check box</td>
<td>Binary value</td>
</tr>
<tr>
<td></td>
<td>Radio button</td>
<td></td>
</tr>
<tr>
<td>Text</td>
<td>Static text</td>
<td>Character string</td>
</tr>
<tr>
<td></td>
<td>Edit text</td>
<td></td>
</tr>
<tr>
<td>Number</td>
<td>Progress bar</td>
<td>Integer or fixed-point value</td>
</tr>
<tr>
<td></td>
<td>Slider</td>
<td></td>
</tr>
<tr>
<td>Matrix</td>
<td>Table</td>
<td>2-D array of numeric values</td>
</tr>
<tr>
<td></td>
<td>Column chart</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Line chart</td>
<td></td>
</tr>
<tr>
<td>Date Time</td>
<td>Date time picker</td>
<td>Seconds to years plus day of week</td>
</tr>
<tr>
<td>List</td>
<td>List box</td>
<td>1-of-n selection</td>
</tr>
<tr>
<td>Section</td>
<td>Popup menu</td>
<td>Arranges controls in a hierarchy</td>
</tr>
<tr>
<td>Password</td>
<td>Client-specific dialogs</td>
<td>Controls access to user interface</td>
</tr>
<tr>
<td>Message</td>
<td>Message box</td>
<td>Alerts user</td>
</tr>
</tbody>
</table>

Table 1 – Controls provided by the FlexiPanel protocol.
<table>
<thead>
<tr>
<th>Pin</th>
<th>Name</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Vss</td>
<td>Power ground reference</td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>RxD</td>
<td>Serial data input</td>
</tr>
<tr>
<td>4</td>
<td>TxD</td>
<td>Serial data output</td>
</tr>
<tr>
<td>5</td>
<td>RTS</td>
<td>Data handshaking output</td>
</tr>
<tr>
<td>6</td>
<td>CTS</td>
<td>Data handshaking input</td>
</tr>
<tr>
<td>7</td>
<td>Mode</td>
<td>May be left unconnected (see product documentation)</td>
</tr>
<tr>
<td>8</td>
<td>Data</td>
<td>Outputs high when initializing or new data available</td>
</tr>
<tr>
<td>9</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>Vdd</td>
<td>Power supply +5V regulated input</td>
</tr>
</tbody>
</table>

Table 2 – FlexiPanel module pin functions.
Data Logger Runtime.bsp
BS2p code for Data Logger runtime program

' Initialization

'($STAMP BS2p)
'($PBASIC 2.5)

' Constants (code generated by FlexiPanel Designer)
TxPin CON 2 ' Transmit pin (from stamp)
RxPin CON 4 ' Receive pin (to Stamp)
CTSPin CON 6 ' Transmit flow control pin (input to Stamp)
RTSPin CON 8 ' Receive flow control pin (output from Stamp)
DataPin VAR IN12 ' Data ready pin (input to Stamp)
BaudM CON 110 ' Baud mode
AckData CON $05 ' Acknowledge data signal (sends DataPin low again)
GetData CON $01 ' Get control data command
SetData CON $02 ' Set control data command
SetRow CON $06 ' Set a row of matrix data command
AddRow CON $07 ' Append a row of matrix data command
GetRow CON $03 ' Get modified control command
ShowMsg CON $04 ' Show message command
ID_Temp_Now CON $01 ' Temp_Now control ID
ID_TimeNow CON $02 ' TimeNow control ID
ID_Temp_History CON $03 ' Temp History control ID
ID_Set_Time CON $04 ' Set_Time control ID

' I2C Constants
SerPt CON 0 ' I2C is on P0 and P1 of BASIC Stamp
TmpOut CON $90 ' I2C Write Address of thermometer
TmpIn CON $91 ' I2C Read Address of thermometer
GetTemp CON $AA ' Read temperature command
Start CON $EE ' Start temperature measurement command
ClikOUT CON $A0 ' I2C Write Address of clock
ClikIN CON $A1 ' I2C Read Address of clock
TmAdd CON $02 ' Read/write time address

' Start of code

' Variables used in program
Temp VAR Byte ' Temperature now, degrees C
sec VAR Byte ' time seconds
mnt VAR Byte ' time minute
hour VAR Byte ' time hour
dayofweek VAR Byte ' time day of week
date VAR Byte ' time date
month VAR Byte ' time month
year VAR Word ' time year

' Initialize Serial Port

DIR2 = 1 ' Tx to FxP (SEROUT output from Stamp)
OUT2 = 1 ' Initialize Tx as high
DIR6 = 0 ' CTS from FxP (SEROUT flow control input to Stamp)
DIR4 = 0 ' Rx from FxP (SERIN input to Stamp)
DIR8 = 1 ' RTS to FxP (SERIN flow control output from Stamp)
OUT8 = 1 ' Initialize RTS as high
DIR12 = 0 ' Data from FxP (SERIN input to Stamp)

' The following code assumes both Basic Stamp and FlexiPanel module were powered up at the same time, it gives flexipanel sufficient time to send Data pin high indicating initialization and then waits for it to go low
PAUSE 50
Init: IF DataPin = 1 THEN Init

Temp = 27
' Start temperature logging
I2COUT SerPt, TmpOut, Start , [ 0 ]

' Set clock
GOSUB GetTime

MainLoop:
' Get temperature & write to flexipanel
I2CIN SerPt, TmpIn, GetTmp, [Tempr]
SEROUT TxPin\CTSPin, BaudM, [SetData, ID_Temp_Now, Tempr, 0, 0, 0]

' Read time
I2CIN SerPt, ClkIn, TmAdd, [sec, mnt, hour, date, month]

' convert date to flexipanel format
sec = (sec & $0F) + ((sec>>4)*10)
mnt = (mnt & $0F) + ((mnt>>4)*10)
year = date>>6                  ' year is 0 - 3, 0 being a leap year
date = (date & $0F) + (((date>>4) & $03)*10)
month = (month & $0F) + (((month>>4) & $01)*10)

' write time to datetime control
SEROUT TxPin\CTSPin, BaudM, [SetData, ID_TimeNow, sec, mnt, hour, date, 7, month, year.LOWBYTE, year.HIGHBYTE] —

' write time to data log
SEROUT TxPin\CTSPin, BaudM, [AddRow, ID_Temp_History, Tempr, sec, mnt, hour, date, 7, month, year.LOWBYTE, year.HIGHBYTE ]

' check to see if date has been updated
IF DataPin = 1 THEN
  GOSUB GetTime
ENDIF

' wait five seconds
FAUSE 5000
GOTO MainLoop

GetTime:
' Clear data pin
SEROUT TxPin\CTSPin, BaudM, [AckData]

' get new time
SEROUT TxPin\CTSPin, BaudM, [GetData, ID_Set_Time]
SERIN RaPin\RTSPin, BaudM, [sec, mnt, hour, date, dayofweek, month, year.LOWBYTE, year.HIGHBYTE]

' convert to RTC's format
sec = (sec // 10) + ((sec / 10) << 4)     ' converts to BCD
mnt = (mnt // 10) + ((mnt / 10) << 4)
year = date>>6                  ' year is 0 - 3, 0 being a leap year
date = (date // 10) + (((date>>4) & $03)*10)
month = (month // 10) + (((month>>4) & $01)*10)

' Set time
I2COUT SerPt, ClkOut, TmAdd, [sec, mnt, hour, date, month]
RETURN

Listing 1 - Runtime BASIC code for Temperature Logger

(Available as Data Logger Runtime.bsp)
BS2p code for Access Controller runtime program

Constants (code generated by FlexiPanel Designer)

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TxPin</td>
<td>CON</td>
<td>2</td>
<td>Transmit pin (from stamp)</td>
</tr>
<tr>
<td>RxPin</td>
<td>CON</td>
<td>4</td>
<td>Receive pin (to Stamp)</td>
</tr>
<tr>
<td>RTSPin</td>
<td>CON</td>
<td>8</td>
<td>Receive flow control pin (output from Stamp)</td>
</tr>
<tr>
<td>DataPin</td>
<td>VAR</td>
<td>IN12</td>
<td>Data ready pin (input to Stamp)</td>
</tr>
<tr>
<td>BaudM</td>
<td>CON</td>
<td>110</td>
<td>Baud mode</td>
</tr>
<tr>
<td>AckData</td>
<td>CON</td>
<td>$05</td>
<td>Acknowledge data signal (sends DataPin low again)</td>
</tr>
<tr>
<td>GetData</td>
<td>CON</td>
<td>$01</td>
<td>Get control data command</td>
</tr>
<tr>
<td>SetData</td>
<td>CON</td>
<td>$02</td>
<td>Set control data command</td>
</tr>
<tr>
<td>SetRow</td>
<td>CON</td>
<td>$06</td>
<td>Set a row of matrix data command</td>
</tr>
<tr>
<td>AddRow</td>
<td>CON</td>
<td>$07</td>
<td>Append a row of matrix data command</td>
</tr>
<tr>
<td>GetMod</td>
<td>CON</td>
<td>$03</td>
<td>Get modified control command</td>
</tr>
<tr>
<td>CtlMod</td>
<td>CON</td>
<td>$08</td>
<td>Was specific control modified command</td>
</tr>
<tr>
<td>ShowMsg</td>
<td>CON</td>
<td>$04</td>
<td>Show message command</td>
</tr>
<tr>
<td>ID_Alice_s_Password</td>
<td>CON</td>
<td>$01</td>
<td>Alice_s_Password control ID</td>
</tr>
<tr>
<td>ID_Bob_s_Password</td>
<td>CON</td>
<td>$02</td>
<td>Bob_s_Password control ID</td>
</tr>
<tr>
<td>ID_Clare_s_Password</td>
<td>CON</td>
<td>$03</td>
<td>Clare_s_Password control ID</td>
</tr>
<tr>
<td>ID_TimeNow</td>
<td>CON</td>
<td>$04</td>
<td>TimeNow control ID</td>
</tr>
<tr>
<td>ID_Access_History</td>
<td>CON</td>
<td>$05</td>
<td>Access_History control ID</td>
</tr>
<tr>
<td>ID_Set_Time</td>
<td>CON</td>
<td>$06</td>
<td>Set_Time control ID</td>
</tr>
</tbody>
</table>

I2C Constants

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SerPt</td>
<td>CON</td>
<td>0</td>
<td>I2C is on P0 and P1 of BASIC Stamp</td>
</tr>
<tr>
<td>TempOut</td>
<td>CON</td>
<td>$90</td>
<td>I2C Write Address of thermometer</td>
</tr>
<tr>
<td>TempIn</td>
<td>CON</td>
<td>$91</td>
<td>I2C Read Address of thermometer</td>
</tr>
<tr>
<td>GetTemp</td>
<td>CON</td>
<td>$AA</td>
<td>Read temperature command</td>
</tr>
<tr>
<td>Start</td>
<td>CON</td>
<td>$EE</td>
<td>Start temperature measurement command</td>
</tr>
<tr>
<td>ClkOut</td>
<td>CON</td>
<td>$A0</td>
<td>I2C Write Address of clock</td>
</tr>
<tr>
<td>ClkIn</td>
<td>CON</td>
<td>$A1</td>
<td>I2C Read Address of clock</td>
</tr>
<tr>
<td>TmAdd</td>
<td>CON</td>
<td>$02</td>
<td>Read/write time address</td>
</tr>
</tbody>
</table>

Lock constants

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lock</td>
<td>VAR</td>
<td>OUT9</td>
<td>Lock control output pin</td>
</tr>
</tbody>
</table>

Start of code

Variable used in program

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>state</td>
<td>VAR</td>
<td>Byte</td>
<td>state of a control</td>
</tr>
<tr>
<td>sec</td>
<td>VAR</td>
<td>Byte</td>
<td>time seconds</td>
</tr>
<tr>
<td>mnt</td>
<td>VAR</td>
<td>Byte</td>
<td>time minute</td>
</tr>
<tr>
<td>hour</td>
<td>VAR</td>
<td>Byte</td>
<td>time hour</td>
</tr>
<tr>
<td>dayofweek</td>
<td>VAR</td>
<td>Byte</td>
<td>time day of week</td>
</tr>
<tr>
<td>date</td>
<td>VAR</td>
<td>Byte</td>
<td>time date</td>
</tr>
<tr>
<td>month</td>
<td>VAR</td>
<td>Byte</td>
<td>time month</td>
</tr>
<tr>
<td>year</td>
<td>VAR</td>
<td>Word</td>
<td>time year</td>
</tr>
<tr>
<td>NewLockState</td>
<td>VAR</td>
<td>Byte</td>
<td>new lock state</td>
</tr>
</tbody>
</table>

Initialize Serial Port

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DIR2</td>
<td>=</td>
<td>1</td>
<td>Tx to FxP (SEROUT output from Stamp)</td>
</tr>
<tr>
<td>OUT2</td>
<td>=</td>
<td>1</td>
<td>Initialize Tx as high</td>
</tr>
<tr>
<td>DIR6</td>
<td>=</td>
<td>0</td>
<td>CTS from FxP (SEROUT flow control input to Stamp)</td>
</tr>
<tr>
<td>DIR4</td>
<td>=</td>
<td>0</td>
<td>Rx from FxP (SERIN input to Stamp)</td>
</tr>
<tr>
<td>DIR8</td>
<td>=</td>
<td>1</td>
<td>RTS to FxP (SERIN flow control output from Stamp)</td>
</tr>
<tr>
<td>OUT8</td>
<td>=</td>
<td>1</td>
<td>Initialize RTS as high</td>
</tr>
<tr>
<td>DIR12</td>
<td>=</td>
<td>0</td>
<td>Data from FxP (SERIN input to Stamp)</td>
</tr>
</tbody>
</table>

initialize lock

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DIR9</td>
<td>=</td>
<td>1</td>
<td>Lock pin is an output</td>
</tr>
<tr>
<td>Lock</td>
<td>=</td>
<td>0</td>
<td>Lock is initially closed</td>
</tr>
</tbody>
</table>
' The following code assumes both Basic Stamp and FlexiPanel module were powered up at the
' same time; it gives flexipanel sufficient time to send Data pin high indicating
' initialization and then waits for it to go low

PAUSE 50
Init: IF DataPin = 1 THEN Init

  * Initialize clock
  GOSUB SetTime

MainLoop:

  ' Read time
  I2CIN SerPt, ClkIn, TmAdd, [sec, mnt, hour, date, month]

  ' convert date to flexipanel format
  sec = (sec & $0F) + ((sec>>4)*10)
  mnt = (mnt & $0F) + ((mnt>>4)*10)
  hour = (hour & $0F) + (((hour>>4) & $03)*10)
  year = (date>>6)                            ' year is 0 - 3, 0 being a leap year
  date = (date & $0F) + (((date>>4) & $03)*10)
  month = (month & $0F) + (((month>>4) & $01)*10)

  ' write time to datetime control
  SEROUT TxPin\CTSPin, BaudM, [SetData, ID_TimeNow, sec, mnt, hour, date, 7, month,
                              year.LOWBYTE, year.HIGHBYTE]

  ' check to see if password changed state or time set
  IF DataPin = 1 THEN
    GOSUB DataChanged
  ENDIF

  ' wait one second
  PAUSE 1000
  GOTO MainLoop

DataChanged:

  ' password open?
  NewLockState = 0

  ' check Alice's password (password is '111')
  SEROUT TxPin\CTSPin, BaudM, [GetData, ID_Alice_s_Password]
  SERIN RxPin\RTSPin, BaudM, [state]
  IF NOT state = 0 THEN
    ' Alice's password is open
    NewLockState = 1

    ' log Alice's access as a '1' in the access history
    SEROUT TxPin\CTSPin, BaudM, [AddRow, ID_Access_History, 1, sec, mnt, hour, date, 7,
                                 month, year.LOWBYTE, year.HIGHBYTE]
  ENDIF

  ' check Bob's password (password is '222')
  SEROUT TxPin\CTSPin, BaudM, [GetData, ID_Bob_s_Password]
  SERIN RxPin\RTSPin, BaudM, [state]
  IF NOT state = 0 THEN
    ' Bob's password is open
    NewLockState = 1

    ' log Bob's access as a '2' in the access history
    SEROUT TxPin\CTSPin, BaudM, [AddRow, ID_Access_History, 2, sec, mnt, hour, date, 7,
                                 month, year.LOWBYTE, year.HIGHBYTE]
  ENDIF

  ' check Clare's password (password is '333')
  SEROUT TxPin\CTSPin, BaudM, [GetData, ID_Clare_s_Password]
  SERIN RxPin\RTSPin, BaudM, [state]
  IF NOT state = 0 THEN
    ' Clare's password is open
    NewLockState = 1

    ' log Clare's access as a '3' in the access history
    SEROUT TxPin\CTSPin, BaudM, [AddRow, ID_Access_History, 3, sec, mnt, hour, date, 7,
                                 month, year.LOWBYTE, year.HIGHBYTE]
  ENDIF
' set state of lock
Lock = NewLockState

' time changed?
SEROUT TxPin\CTSPin, BaudM, [CtlMod, ID_Set_Time]
SERIN RxPin\RTSPin, BaudM, [state]

IF NOT state = 0 THEN
GOSUB SetTime
ENDIF
RETURN

SetTime:

' get new time
SEROUT TxPin\CTSPin, BaudM, [GetData, ID_Set_Time]
SERIN RxPin\RTSPin, BaudM, [sec, mnt, hour, date, dayofweek, month, year.LOWBYTE, year.HIGHBYTE]

' convert to RTC's format
sec = (sec // 10) + ((sec / 10) << 4)  ' converts to BCD
mnt = (mnt // 10) + ((mnt / 10) << 4)
hour = (hour // 10) + ((hour / 10) << 4)
date = (date // 10) + ((date / 10) << 4) + ((year // 4) << 6)
month = (month // 10) + ((month / 10) << 4)

' Set time
I2COUT SerPt, ClkOut, TmAdd, [sec, mnt, hour, date, month]
RETURN

Listing 2 - Runtime BASIC code for Access Controller

(Available as Access Controller Runtime.bsp)
Bluetooth Remote Control… From Your Mobile Phone

Robot.bsp

BS2p code for control Robot at runtime

-------------------------------------------------------------------------------------------

{STAMP BS2p}
{PBASIC 2.5}

Constants (code generated by FlexiPanel Designer)

| TxPin   | CON     | 2 | Transmit pin (from stamp) |
| RaPin   | CON     | 4 | Receive pin (to Stamp)   |
| RTSPin  | CON     | 6 | Transmit flow control pin (input to Stamp) |
| DataPin | VAR     | IN12 | Data ready pin (input to Stamp) |
| BaudM   | CON     | 110 | Baud mode |
| AckData | CON     | $05 | Acknowledge data signal (sends DataPin low again) |
| GetData | CON     | $01 | Get control data command |
| SetData | CON     | $02 | Set control data command |
| SetRow  | CON     | $06 | Set a row of matrix data command |
| AddRow  | CON     | $07 | Append a row of matrix data command |
| GetMod  | CON     | $03 | Get modified control command |
| ShowMsg | CON     | $04 | Show message command |
| ID_Bearing | CON     | $01 | Bearing control ID |
| ID_Forward | CON     | $02 | Forward control ID |
| ID_Left | CON     | $03 | Left control ID |
| ID_Right | CON     | $04 | Right control ID |
| ID_Stop | CON     | $05 | Stop control ID |
| ID_Reverse | CON     | $06 | Reverse control ID |
| ID_Route_trace | CON     | $07 | Route_trace control ID |

I2C Constants

| CmpIn | CON | $C1 |
| SerPt | CON | 0 |

Motor control PWM constants - may need to be adjusted for different motors

| rMax  | CON | 2497 |
| rMid  | CON | 2025 |
| rStop | CON | 1873 |
| rMidZ | CON | 1700 |
| rMaxZ | CON | 1249 |
| lMax  | CON | 1249 |
| lMid  | CON | 1700 |
| lStop | CON | 1873 |
| lMidZ | CON | 2025 |
| lMaxZ | CON | 2497 |

| lPort | CON | 14 | Left motor PWM output pin |
| rPort | CON | 15 | Right motor PWM output pin |

Variables used in program

| brad | VAR | Byte | Bearing in binary radians (0-255 in a full circle) |
| degs | VAR | Word | Bearing in tenths of a degree (0-3599 full circle) |
| xloc | VAR | Word | X position in mm |
| yloc | VAR | Word | Y position in mm |
| bdata | VAR | Byte | one byte utility variable |
| CmpCount | VAR | Byte | counts down from when last log was taken |
| FwRvSp | VAR | Byte | Current state: Forward/turn=1, Reverse=2, Stop=0 |

xloc = 0
yloc = 0
CmpCount = 50
DIR14 = 1
DIR15 = 1
FwRvSp = 0
' Initialize Serial Port

<table>
<thead>
<tr>
<th>DIR2</th>
<th>DIR6</th>
<th>DIR8</th>
<th>OUT2</th>
<th>OUT8</th>
<th>DIR12</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Tx to FxP (SEROUT output from Stamp)  
CTS from FxP (SEROUT flow control input to Stamp)  
Rx from FxP (SERIN input to Stamp)  
RTS to FxP (SERIN flow control output from Stamp)  
Initialize Rx as high  
Data from FxP (SERIN input to Stamp)

The following code assumes both Basic Stamp and FlexiPanel module were powered up at the same time; it gives flexipanel sufficient time to send Data pin high indicating initialization and then waits for it to go low

PAUSE 50

Init: IF DataPin = 1 THEN Init

' Main program loop

ReadControls:

' Acknowledge data pin
SEROUT TxPin\CTSPin, BaudM, [AckData]

' test for forward
SEROUT TxPin\CTSPin, BaudM, [GetData, ID_Forward]
SERIN RxPin\RTSPin, BaudM, [bdata]
IF bdata = $FF THEN GoForward

' test for veer left
SEROUT TxPin\CTSPin, BaudM, [GetData, ID_Left]
SERIN RxPin\RTSPin, BaudM, [bdata]
IF bdata = $FF THEN VeerLeft

' test for veer right
SEROUT TxPin\CTSPin, BaudM, [GetData, ID_Right]
SERIN RxPin\RTSPin, BaudM, [bdata]
IF bdata = $FF THEN VeerRight

' test for reverse
SEROUT TxPin\CTSPin, BaudM, [GetData, ID_Reverse]
SERIN RxPin\RTSPin, BaudM, [bdata]
IF bdata = $FF THEN BackUp

' Must be in stopped state; wait, record compass, test for button presses

StopWait:
PAUSE 20
FwRvSp = 0
GOSUB CheckCompass
IF DataPin = 1 THEN ReadControls
GOTO StopWait

' In forward state; send forward pulses, record compass, test for button presses

GoForward:
PULSOUT lPort, lMax
PULSOUT rPort, rMax
PAUSE 20
FwRvSp = 1
GOSUB CheckCompass
IF DataPin = 1 THEN ReadControls
GOTO GoForward

' In veer left state; send veer left pulses, record compass, test for button presses

VeerLeft:
PULSOUT lPort, lMax
PULSOUT rPort, rMid
PAUSE 20
FwRvSp = 1
GOSUB CheckCompass
IF DataPin = 1 THEN ReadControls
GOTO VeerLeft

' In veer right state; send veer right pulses, record compass, test for button presses

VeerRight:
PULSOUT lPort, lMid
PULSOUT rPort, rMax
PAUSE 20
FwRvSp = 1
GOSUB CheckCompass
IF DataPin = 1 THEN ReadControls
GOTO VeerRight
' In reverse state; send veer right pulses, record compass, test for button presses

BackUp:

PULSOUT lPort, lMaxZ
PULSOUT rPort, rMaxZ
PAUSE 20
FwRvSp = 2
GOSUB CheckCompass
IF DataPin = 1 THEN ReadControls
GOTO BackUp

CheckCompass:

' only check every 50 pulses
CmpCount = CmpCount - 1
IF CmpCount > 0 THEN GoBack
CmpCount = 50

' Get compass direction in binary radians and in tenths of a degree
I2CIN SerPt, CmpIn, 1, [brad, degs.HIGHBYTE, degs.LOWBYTE]

' Send degrees value to bearing control (code generated by FlexiPanel Designer)
SEROUT TxPin\CTS\Pin, BaudM, [SetData, ID_Bearing, degs.LOWBYTE, degs.HIGHBYTE, 0, 0]

' Calculate position with Send binary radians value to bearing control
' (code generated by FlexiPanel Designer & cut'n'pasted)

IF FwRvSp = 1 THEN
  xloc = xloc + COS( brad )
yloc = yloc + SIN( brad )
ELSEIF FwRvSp = 2 THEN
  xloc = xloc - COS( brad )
yloc = yloc - SIN( brad )
ENDIF

' if moving, send to trace
IF NOT FwRvSp = 0 THEN
  SEROUT TxPin\CTS\Pin, BaudM, [AddRow, ID_Route_trace, yloc.LOWBYTE, yloc.HIGHBYTE, xloc.LOWBYTE, xloc.HIGHBYTE ]
ENDIF

' return to motor control
GoBack:
RETURN

Listing 3 - Runtime BASIC code for Robot Controller

(Available as Robot Runtime.bsp)
Editor’s Appendix: Files to be provided to readers

(this page not to be typeset)

FlexiPanel Designer files:

\[AccessController.FxP\]
\[DataLog.FxP\]
\[Robot.FxP\]

BASIC Stamp FlexiPanel programming programs:

\[AccessController.bsp\]
\[DataLog.bsp\]
\[Robot.bsp\]

BASIC Stamp runtime programs:

\[AccessController Runtime.bsp\]
\[DataLog Runtime.bsp\]
\[Robot Runtime.bsp\]