FlexiPanel Ltd

FlexiPanel™

BASIC Stamp Edition
version 2.2

www.FlexiPanel.com

Contents

Legal Notices

The FlexiPanel product range and its associated intellectual property are protected by pending patents and copyright law. You may only use FlexiPanel products and their associated intellectual property in ways as detailed on this page. We will protect our rights to the fullest possible legal extent.

Definitions: A *FlexiPanel client* is hardware or software which creates a control panel when requested by a FlexiPanel server. A *FlexiPanel server* is hardware or software, such as *FlexiPanel Remote Controls API,* which requests a control panel to be created on a FlexiPanel client. *FlexiPanel Designer* is a software tool to aid the design of FlexiPanel control panels.

What You May Do

- Use *FlexiPanel BASIC Stamp Edition* modules provided you have obtained them from FlexiPanel Ltd or an authorized distributor.
- Use FlexiPanel Designer to aid the design of FlexiPanel control panels used with licensed FlexiPanel servers.
- Use and distribute FlexiPanel client software for use with licensed FlexiPanel servers, for free or for profit.
- Use and distribute FlexiPanel product documentation and if necessary modify it and include it in your product documentation.
- Create custom FlexiPanel client hardware or software provided you have obtained a license from FlexiPanel Ltd. (This license is normally granted free.)

What You May Not Do

- Create, distribute or use any technology which infringes any patents or copyrights associated with FlexiPanel technology, except as detailed above.
- Attempt to bypass or reverse engineer the serial code licensing system incorporated into some FlexiPanel software products.
- Attempt to interpret these legal notices or any of our licensing agreements in such a way that attempts to avoid the need to obtain a license from us in order to use FlexiPanel technology.

Limitation Of Liability

FlexiPanel technology should not be used in life critical devices without the permission FlexiPanel Ltd.

FlexiPanel Ltd makes every effort to ensure, but cannot warrant, that its products and documentation are without errors and omissions. However FlexiPanel Ltd does not accept liability for consequent loss or injury as a result of using its products or interpreting its documentation. FlexiPanel Ltd will not be responsible for any third party patent infringements arising from the use of its products.

FlexiPanel Ltd reserves the right to make changes to its technology and documentation in order to improve reliability, function or design.

If any of this is not clear, contact FlexiPanel Ltd before proceeding.

Glossary and Notation

Hex Notation

Throughout this document, numbers with an '0x' prefix should be assumed to be in hex. For example, 0xFF is completely equivalent to decimal 255.

In some partners' documentation, '$' notation is used in place of '0x' notation. '$FF' is equivalent to '0xFF' and decimal 255.

Data Types

Data types are C standard data types; no floating point is used. C standard notation and calling conventions are assumed. Integers are explicitly defined as:

bool – unsigned char, zero for *false*, otherwise *true*

byte – 8 bit unsigned integer

int16 – 16 bit signed integer

int32 – 32 bit signed integer

signed char – 8 bit signed integer

uint16 – 16 bit unsigned integer

uint32 – 32 bit unsigned integer

unsigned char – 8 bit unsigned integer

word – 16 bit unsigned integer

Abbreviations

API - Application Program Interface

FBVS – FlexiPanel Bluetooth VirtualComPort Server

FBVSN – FlexiPanel Bluetooth VirtualComPort Server Notification

Glossary

Control panel – The controls that the FlexiPanel server wishes to be displayed on the FlexiPanel client.

FlexiPanel client – Hardware or software that creates a control panel when requested to by a FlexiPanel server.

FlexiPanel server – Hardware or software that requests a control panel to be created on a FlexiPanel client.

Target – The FlexiPanel environment system that a control panel is being designed for, e.g. FlexiPanel for BASIC Stamp.

Zero Terminator – A zero-valued character used to indicate the end of a string of characters.

FlexiPanel Bluetooth Remote User Interface

Summary

The *FlexiPanel BASIC Stamp Edition* module allows BASIC Stamps to create user interfaces on remote devices such as handhelds (PDAs), mobile phones and other computers, using a Bluetooth link. The remote device asks the module what controls it wants displayed and then it displays them. The user reads the display and modifies the controls as required, sending any changes back to the module.

The patented FlexiPanel system features:

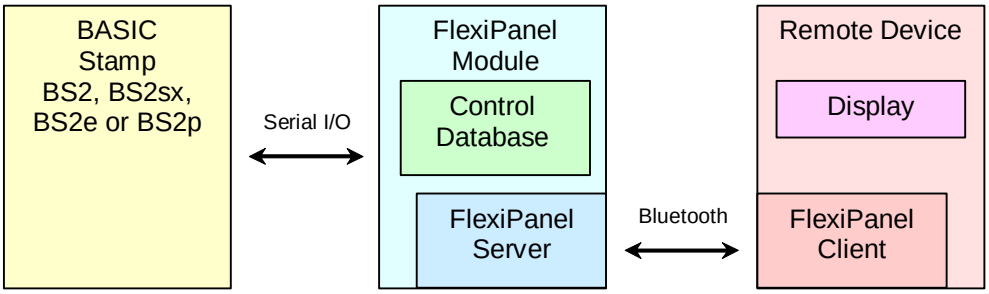
- Eleven control types.
- No programming required on the remote device (handheld, etc). FlexiPanel software for PDAs, phones and other computers is provided by FlexiPanel Ltd and is freely distributable.
- Serial interface to the BASIC Stamp.
- Compatible with BS2, BS2e, BS2sx and BS2p BASIC Stamps.

FlexiPanel provides a variety of controls, including:

Virtual Control	***Control Simulated***
Check Box	On / Off switch
Pushbutton	Pushbutton
Number	Meter, Dial, etc
Slider	Dial / Potentiometer
List Box	Selector Switch
Text	Text Display / Keyboard
Chart	Graphic Display
Message Box	Alert LED / Buzzer
Password Control	Key Switch

The *FlexiPanel BASIC Stamp Edition* module is intended to eliminate the need for expensive control panel components such as LCD displays, switches, etc, while offering greater flexibility to interact with a BASIC Stamp at run-time. Applications include industrial control, discrete-interface consumer goods, automotive diagnostics and data logging.

How FlexiPanel Works



1) FlexiPanel Module is programmed with desired control panel.

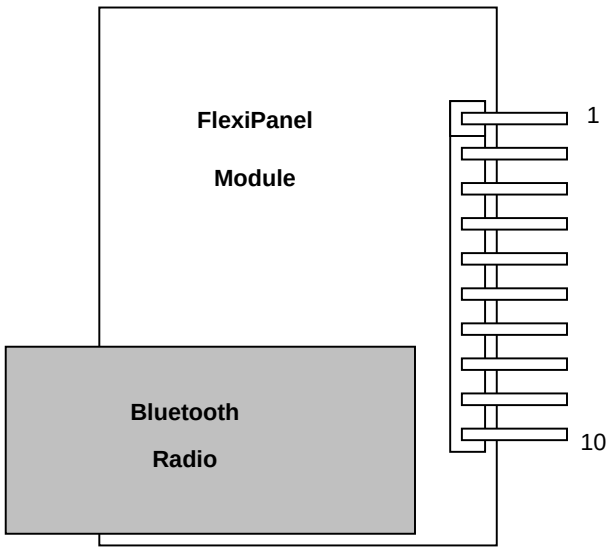
2) BASIC Stamp can read and write control panel values at any time.

3) A remote device can connect to the FlexiPanel module at any time and:

   a) Display the controls,

   b) Modify the controls and send the changes back to the FlexiPanel Module.

Device Pinout

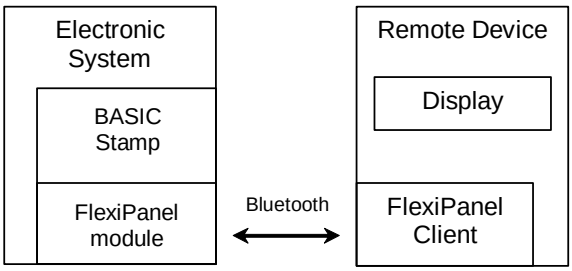


Pin	*Name*	*Purpose*
1	Vss	Connect to 0V
2		Not connected
3	RxD	Serial data input from BASIC Stamp for *serout* operations
4	TxD	Serial data output to BASIC Stamp for *serin* operations
5	RTS	Serial flow control output to BASIC Stamp for *serout* operations
6	CTS	Serial flow control input from BASIC Stamp for *serin* operations
7	Mod	See text.
8	Data	Data output high when a control has been updated by a FlexiPanel client.
9		Not connected
10	Vdd	Connect to +5v.

Care must be taken to insert the module into the correct side of the AppMod slot and in the correct orientation. Make sure Vss connects to Vss and Vdd connects to Vdd (not Vin!). Failure to do so may damage the module.

Overview

FlexiPanel is a generic technology for allowing one device such as the *FlexiPanel BASIC Stamp Edition* module (the *FlexiPanel server*) to create a control panel on another device (the *FlexiPanel client*). It was created to provide computer software and appliances with universal remote control facility and in some cases even replace all other user interface components entirely.



FlexiPanel Designer is a software design tool used to specify the desired control panel on the FlexiPanel module in the first place. The control panel may be specified and tested from within the design tool and then sent to a specific target.

This document is arranged as follows:

- An overview of the FlexiPanel system.
- Connecting the FlexiPanel module.
- Creating FlexiPanel control panels using FlexiPanel Designer.
- Connecting to the module from a FlexiPanel Client device.
- Runtime interaction with a BASIC Stamp.
- A sample application, the *All Controls Demo*.

Concepts

FlexiPanel clients connect to FlexiPanel servers such as the *FlexiPanel BASIC Stamp Edition* module. A client may connect to the server at any time. Once the connection is made, the server tells the client to display the desired control panel on its user interface.

The server may modify the contents and appearance of the controls at any time. If the client modifies the control panel, for example pressing a button, it sends a message to the FlexiPanel server.

The client may choose to disconnect at any time. Additionally, the link may be dropped if the devices go out of range of each other. The state of the controls is retained by the server so that if the client reconnects, or another client connects, the control panel will be in the same state as it was when it was last modified.

Devices incorporating FlexiPanel Servers must be designed taking into account the possibility of a dropped connection. Specifically, no action should be taken which relies on a client's ability to maintain a connection. If FlexiPanel is used to operate machinery, for example, the BASIC Stamp should provide a failsafe mode should the connection be dropped.

FlexiPanel Controls

Twelve control types are provided by FlexiPanel, but only eleven of these are available to the BASIC Stamp. (The Files control would not make sense since the BASIC Stamp does not have a file system.) These include controls familiar to Windows users and others that are particularly appropriate for FlexiPanel technology.

FlexiPanel clients are required to provide all the requested controls in some form or other. Since the user interface may vary from one FlexiPanel client to another, the appearance is not guaranteed.

If the developer expects a device to be used in conjunction with a specific type of FlexiPanel client (e.g. Pocket PC), it may specify those *Client Settings* in more detail; however, the control panel will still work on any FlexiPanel client.

Generic Control Properties

All controls possess certain generic properties. These are set when the control is first specified and they may be modified at any time.

- All controls have a title, although the client may not necessarily display this.

- Some controls are either modifiable or non-modifiable. If a control is non-modifiable, the server may change its value but the client may not.

- A control may be invisible, in which case is not accessible to the user.

- A control may request that it to be brought to the user's attention. If more controls are specified than can appear on the screen at once, the FlexiPanel client will implement some kind of scrolling or paging mechanism. Such a request will ensure that the control is brought into view.

- Right-to-left text style may be requested, although it is not guaranteed that the client can fulfill this request.

- Controls may be grouped together. If possible, the client will use this information to lay out the controls in a logical manner.

- Controls have a color. If possible, the client will display the control in this color.

Text Control

The text control contains a text string. It will have a fixed maximum length, specified when the control is created.

A text control may have password style, in which case the text entered in the control is not readable by the user.

It is possible to specify that text control is intended to be the title of the following control or group of controls. If possible, the client will use this information in laying out the controls in a logical manner.

Button Control

A button control registers when a button is pressed.

Latch Control

A latch control stores a binary (*on/off*) value. Latches may be arranged in groups so that when one latch is turned on the others are turned off.

Password Control

A password control stores a password and has an open and closed state. In the closed state, the user must enter the password to set it to the open state.

In the open state, the password control may be returned to the closed state at any time.

It is possible to specify that password may be modified by the user once the control is in the open state. A master password may also be provided.

Other controls may be directly linked to the password control so that they are only visible when the password is open.

Passwords are limited to 17 Unicode characters or 34 ASCII characters, including zero terminator.

Number Control

A number control stores a numeric value. It is essentially a signed four-byte integer, but its decimal place may be shifted left or right in order to represent any floating-point value.

Matrix Control

A matrix control stores an array of numbers. These might be displayed as a table or a chart. In this release of FlexiPanel, the values are not modifiable by the client.

List Control

A list control allows one item to be selected from a list.

Section Control

A section control acts like a pop-up menu. Controls enclosed within a section control are only visible when the section control is opened.

Controls enclosed inside a closed section control are not transmitted to the client, thereby minimizing communication time.

DateTime Control

A DateTime control stores a DateTime value, i.e. second, minute, hour, date, day-of-week, month and year.

Message Control

A message control displays a message on request.

Blob Control

The blob (Binary Large Object) control allows client and server to pass binary objects to each other. It is intended primarily for future expansion and customization. Due to the limitations of some client devices, a client is not obliged to support all features associated with this control; some clients may simply ignore it.

In this release of FlexiPanel, the only use of the blob object is to pass the name of a URL (i.e. web page address) to the client.

Files Control

The files control allows the server to pass files to the client. It is not supported by *FlexiPanel BASIC Stamp Edition*.

Requirements

A personal computer with Windows Millennium Edition (or later) operating system is required to run the FlexiPanel Designer software. A remote client device such as a Bluetooth-equipped Windows PC, Smartphone or Pocket PC will be needed to interact with the module in order to provide the remote control panel. The software required to run on the client device may be downloaded free from www.FlexiPanel.com.

FlexiPanel Protocol 2.0

The communication standard used by *FlexiPanel BASIC Stamp Edition* in order to communicate with clients is *FlexiPanel Protocol 2.0*.

FlexiPanel Protocol 1.0 is an infrared protocol and so no attempt is made to be backward compatible with it. Every effort will be made to keep future Bluetooth FlexiPanel Protocols backward compatible with FlexiPanel Protocol 2.0.

Connecting the FlexiPanel Module

The FlexiPanel BASIC Stamp Edition module has a 10-pin 0.1" connector allowing it to be inserted into most prototyping boards. It is also compatible with the AppMod slot in BASIC Stamp demo boards.

Care must be taken to insert the module into the correct side of the AppMod slot and in the correct orientation. Make sure Vss connects to Vss and Vdd connects to Vdd (not Vin!). Failure to do so may damage the module.

Current consumption of the module is 12mA average, 60mA peak when unconnected to a remote devices. When connected, consumption rises to 40mA average, 90mA peak.

The pins function as follows:

Vss – Connect to the same power ground as the BASIC Stamp

RxD – TTL-level serial input to the module from the BASIC Stamp. All serial I/O is at 19200 baud. It must be used in conjunction with the *RTS* output pin to provide handshaking. If the module is inserted in the AppMod slot, this pin will connect to P2.

TxD – TTL-level output from the module to the BASIC Stamp. All serial I/O is at 19200 baud. It must be used in conjunction with the *CTS* input pin to provide handshaking. If the module is inserted in the AppMod slot, this pin will connect to P4.

RTS – TTL-level handshaking output from the module to the BASIC Stamp. It is used in conjunction with the *RxD* pin to provide handshaking. If the module is inserted in the AppMod slot, this pin will connect to P6.

CTS – TTL-level handshaking input to the module to the BASIC Stamp. It is used in conjunction with the *TxD* pin to provide handshaking. If the module is inserted in the AppMod slot, this pin will connect to P8.

Mode – The behavior of the Mode pin depends on the *Target Settings* selected in *FlexiPanel Designer*:

- If "straight-through mode" operation is selected, this pin is an input. If this pin is held at logic high at start-up, the module will act as a Bluetooth Serial Port slave device. (See below.) If this pin is held at logic low at start-up, the module will operate as usual as a FlexiPanel module.

- If "indicate a client is connected" operation is selected, this pin is an output. When a client is connected to the module over a Bluetooth link, the pin will output logic high. Otherwise it will output logic low.

Data – During start-up, this pin will be high until the module is initialized. This may take several seconds. After initialization, this pin will be high whenever the client has updated any control's value more recently than the BASIC Stamp has either set or read that value.

Vdd – Connect to the +5V power supply.

Bluetooth features of the module

The Bluetooth radio is a Class I device and has a range of up to 100 meters (330 feet) when connecting with other Class I devices. If a Class II device connects to the module, the communication range will be limited to 10 meters (33 feet). Walls, metal objects, *etc.,* between the modules may reduce these ranges. The range may also be affected by the orientation of the antennae.

No authentication or encryption is employed and any device may connect to the module.

If the module is configured to operate as a Bluetooth Serial Port slave device and the Mode pin is at logic high at start-up, other devices may connect to the module and engage in serial communication. As with usual operation, all serial I/O is at 19200 baud and requires handshaking. Once the remote device disconnects, the module will not accept further connections unless the string **AT+BSLV** followed by carriage return are sent to the module. The BASIC Stamp cannot make outgoing connections.

Creating Control Panels

Control panels should first be designed and tested using the *FlexiPanel Designer* software tool available from FlexiPanel Ltd. Consult its documentation for more details.

FlexiPanel Designer has been developed with the intention of programming control panels on a range of devices including *FlexiPanel Stamp Edition* modules. Consequently it includes some options that the module cannot support. Specifically:

- Control Panel ROM space is limited to around 1780 bytes. It is sufficient for approximately 24 controls.
- Only BS2, BS2e, BS2sx and BS2p BASIC Stamps are supported.
- The Files Control is not supported.
- The Real Time Clock is not supported.
- Ping must be enabled.
- A control's visibility and color may not be modified at runtime. (Child controls of password and section controls are an exception. They are only visible when their parent control is open.)

To prepare for programming FlexiPanel BASIC Stamp Edition modules, select *BASIC Stamp Edition* from the Target Device menu. The *BASIC Stamp Properties* will appear in the right-hand properties list Set the BASIC Stamp type and select what you would like the *Mode* pin to do. Also specify which module pins are connected to which Stamp I/O pins. If you are using the AppMod slot, you can keep the default values.

Once a control panel has been designed it may be programmed into a FlexiPanel BASIC Stamp Edition module. The module will then perform all the tasks of a FlexiPanel server and will retain all control settings for the BASIC Stamp.

The FlexiPanel BASIC Stamp Edition module is programmed from FlexiPanel Designer as follows:

1. Create BASIC Stamp source code to program the module (*i.e.* step 1 above) by selecting *Create BASIC programming code…* from the *Target Device* menu. The *Storage Summary* dialog box appears (Figure 2), followed by a dialog box prompting for a file name for the source code file to be created. Select a file name and press OK.

   FlexiPanel Designer creates BASIC Stamp source code to program the module. See for example Figure 11. Note the extra sample code provided at the end of the file. It shows how the BASIC Stamp should read and write from the FlexiPanel once the Control Panel has been programmed.

2. Connect a FlexiPanel BASIC Stamp Edition module to the BASIC Stamp. For interest, you may wish to connect an LED to the Data pin (with a 1k current limiting resistor connected to Vss) to watch its behavior.

3. Open the source code file in the BASIC Stamp Editor and select Run from the Run menu to program the BASIC Stamp.

4. When the program runs, the BASIC Stamp writes the control panel specification into the FlexiPanel module. After a few seconds, the BASIC Stamp Editor will confirm the process is complete by writing *Acknowledge: ROM* to the Debug Terminal (Figure 1). The module will then reset itself. If the message does not appear after a few seconds, programming failed. (Try removing and reapplying the power.)

5. The control panel has been written to the FlexiPanel BASIC Stamp Edition module. It will be retained until it is re-programmed, even after power-off.

6. The BASIC Stamp may now be loaded with its run-time program for interacting with the control panel and perform its intended tasks. (See later section.)

Connecting to the Module from a FlexiPanel Client

Once a control panel is programmed into the FlexiPanel module, the control panel can be viewed from any device running FlexiPanel client software, even before writing a runtime program for the BASIC Stamp.

FlexiPanel Clients include a range of PDAs and cellphones. FlexiPanel client software may be downloaded free of charge from *www.FlexiPanel.com*.

An example of how a control panel might appear on a Pocket PC FlexiPanel client is shown in Figure 3 to Figure 9. A sample control panel for a Smartphone running Microsoft Phone Edition 2003 is shown in Figure 10.

If the FlexiPanel client modifies any controls, the *Data* pin on the module will go high, indicating that a control has been modified.

Runtime Interaction With A BASIC Stamp

Once a control panel is programmed into the FlexiPanel module, it will be retained, even if the power is removed. The BASIC Stamp may then be programmed with a "runtime" program which can then interact with the control panel and also perform whatever other tasks it needs to do.

Towards the bottom of the BASIC file generated by FlexiPanel Designer is some sample code specific to the control panel that was created. You can copy and paste from the sample code rather than reading the rest of this section in depth.

The All Controls Demo in the next section is an example of how to interact with the FlexiPanel Stamp Edition module at runtime.

Communication with the module is always at 19200 baud with handshaking, so sending bytes to the module always uses the following BASIC command:

```
SEROUT RxDpin\RTSpin, BaudR, [data]
```

where `RxDpin` is the Stamp I/O pin to which the FlexiPanel *RxD* pin is connected and `RTSpin` is the I/O pin to which the FlexiPanel *RTS* pin is connected. (If the AppMod slot is used, these will be 2 and 6 respectively.)

Select `BaudR` from the following constants:

Stamp in use	BaudR value
BS2	32
BS2e	32
BS2sx	110
BS2p	110

Receive bytes from the module using the following BASIC command:

```
SERIN  TxDpin\CTSpin, BaudR, [data]
```

where `TxDpin` is the Stamp I/O pin to which the FlexiPanel *TxD* pin is connected and `CTSpin` is the I/O pin to which the FlexiPanel *CTS* pin is connected. (If the AppMod slot is used, these will be 4 and 8 respectively.) `BaudR` is as above.

If desired, the BASIC Stamp can detect whether the FlexiPanel has new information by monitoring the state of the Data pin.

The BASIC Stamp can also detect whether a client is currently connected to FlexiPanel by monitoring the state of the Mode pin. The module must also have been appropriately configured during the design process.

In general operation, the BASIC Stamp should regularly check to see whether a client has changed the state of any of the controls and react accordingly. Such a test should be regular (at least 10 times a second) if the Stamp is to be seen to be reacting to the control panel information promptly. The Stamp can test whether the state of any of the controls has changed monitoring the Data pin or using the Control Modified command.

If the Stamp wishes to change the value of a control, it may do so at any time.

FlexiPanel Commands

Communicating with the FlexiPanel is achieved by sending commands to it. Depending on the command, the Stamp may then send data to the FlexiPanel module, or the module may send data to the Stamp.

The commands are as follows:

Control Modified

The *Control Modified* command asks the FlexiPanel module whether a specific control has been modified. It is sent as follows:

```
SEROUT RxDpin\RTSpin, BaudR, [CtlMod, CtlID]
SERIN  TxDpin\CTSpin, BaudR, [CtlState]
```

where `CtlMod` is the constant $08 and `CtlID` is the control ID. `CtlState` is a byte variable which receives the state of the control. It is $00 if it has not been modified, $FF otherwise.

Get Modified

The *Get Modified* command asks the FlexiPanel module whether any controls have been modified. It is sent as follows:

```
SEROUT RxDpin\RTSpin, BaudR, [GetMod]
SERIN  TxDpin\CTSpin, BaudR, [CtlID]
```


where `GetMod` is the constant $03 and `CtlID` is a byte variable which receives a control ID. If no controls have been modified by the client (corresponding to the Data pin being low), `CtlID` will be $00.

If the client modified a control since the BASIC Stamp last set or read its value, `CtlID` will contain the ID of that control. A *Get Data* command should then be sent to get the new value. If a further *Get Modified* command is sent, `CtlID` will contain the ID of another control which has been modified, or $00 if there are no others.

Acknowledge Data

The *Acknowledge Data* command clears all control modified flags so that any following *Get Modified* command would reply $00. The Data pin is also cleared. It is sent as follows:

```
SEROUT RxDpin\RTSpin, BaudR, [AckData]
```

where `AckData` is the constant $05.

Get Data

The *Get Data* command retrieves the value of a control. It is sent as follows:

```
SEROUT RxDpin\RTSpin, BaudR, [GetData, CtlID]
SERIN  TxDpin\CTSpin, BaudR, [ValBytes]
```

where `GetData` is the constant $01, `CtlID` is the ID of the control and `ValBytes` receives the control's value. The exact nature of `ValBytes` will vary according to the control:

Button – a single byte which will be $FF if it was pressed, or $00 otherwise.

Latch – a single byte which will be $FF if it is *on*, or $00 if it is *off*.

Section – a single byte which will be $FF if it is *open*, or $00 if it is *closed*.

Password – a single byte which will be $FF if it is *open*, or $00 if it is *closed*.

Text – a string of as many bytes as was originally specified as the maximum length of the string, including zero terminator. All bytes must be read, even if they are not all used, otherwise the handshaking will get out of sync.

Number – The numerical value as a four-byte integer, least significant byte first. All four bytes must be read, even if they are not all used, otherwise the handshaking will get out of sync.

Matrix – Matrix values cannot be read.

List – The selected item as a four-byte integer, least significant byte first. All four bytes must be read, even if they are not used, otherwise the handshaking will get out of sync.

Date Time – The second as a byte (0-59), followed by minute as a byte (0-59), hour as a byte (0-23), date as a byte (1-31), day of week as a byte (0-7, 0=Sunday, 6=Saturday, 7=unknown), month as a byte (1-12) and year as a two-byte (word) value (0-65536). All eight bytes must be read, even if they are not all used, otherwise the handshaking will get out of sync.

Message – Message values cannot be read.

Blob – Blob values cannot be read.

Set Data

The *Set Data* command sets the value of a control. It is sent as follows:

```
SEROUT RxDpin\RTSpin, BaudR,
                [SetData, CtlID, ValBytes]
```

where `SetData` is the constant $02, `CtlID` is the ID of the control and `ValBytes` is the new the control value. The exact nature of `ValBytes` will vary according to the control:

Button – Button values cannot be read.

Latch – a single byte, $FF for *on*, $00 for *off*.

Section – a single byte, $FF for *open*, $00 for *closed*.

Password – a single byte, $FF for *open*, $00 for *closed*.

Text – a string of as many bytes as was originally specified as the maximum length of the string, including zero terminator. All bytes must be sent,

even if they are not all required, otherwise the handshaking will get out of sync.

Number – The numerical value as a four-byte integer, least significant byte first. All four bytes must be sent, even if they are not all required, otherwise the handshaking will get out of sync.

Matrix – Matrix values should be set with the Add Matrix Row or Set Matrix Row commands.

List – The item to select as a four-byte integer, least significant byte first. All four bytes must be sent, even if they are not required, otherwise the handshaking will get out of sync.

Date Time – The second as a byte (0-59), followed by minute as a byte (0-59), hour as a byte (0-23), date as a byte (1-31), day of week as a byte (0-7, 0=Sunday, 6=Saturday, 7=unknown), month as a byte (1-12) and year as a two-byte (word) value (0-65536). All eight bytes must be sent, even if they are not all required, otherwise the handshaking will get out of sync.

Message – The new message text as a string of as many bytes as was originally specified as the maximum length of the string, including zero terminator. All bytes must be sent, even if they are not all required, otherwise the handshaking will get out of sync.

Blob – The new URL as a string of as many bytes as was originally specified as the maximum length of the string, including zero terminator. All bytes must be sent, even if they are not all required, otherwise the handshaking will get out of sync.

Add Row

The *Add Row* command appends a row of data onto the end of a matrix. If there are no spare rows in the matrix, the first row is discarded and the new data is appended onto the bottom. The format of the command is:

```
SEROUT RxDpin\RTSpin, BaudR, [AddRow,
            CtlID, yData, xData]
```

where `AddRow` is the constant $07 and `CtlID` is the ID of the control. The exact nature of `yData` will depend on the details of how the control was specified on *FlexiPanel Designer:*

- If 1-byte y-axis storage was specified, `yData` will be a single byte signed integer.
- If 2-byte y-axis storage was specified, `yData` will be a two byte (word) signed integer, least significant byte first, least significant byte first.
- If 4-byte y-axis storage was specified, `yData` will be a four byte signed integer, least significant byte first, least significant byte first.

Similarly, nature of `xData` will depend on its original specification:

- If List or Labels x-axis type were specified, no `xData` bytes are sent.
- If XY x-axis type and 1-byte x-axis storage were specified, `xData` will be a single byte signed integer.
- If XY x-axis type and 2-byte x-axis storage were specified, `xData` will be a two-byte (word) signed integer, least significant byte first.
- If XY x-axis type and 4-byte x-axis storage were specified, `xData` will be a four-byte signed integer, least significant byte first.
- If DateTime x-axis type was specified, `xData` will be a eight bytes: The second as a byte (0-59), followed by minute as a byte (0-59), hour as a byte (0-23), date as a byte (1-31), day of week as a byte (0-7, 0=Sunday, 6=Saturday, 7=unknown), month as a byte (1-12) and year as a two-byte (word) value (0-65536).

Set Row

The *Set Row* command sets the values in one row of the matrix. The format of the command is:

```
SEROUT RxDpin\RTSpin, BaudR, [SetRow,
            CtlID, Row, yData, xData]
```

where `SetRow` is the constant $06 and `CtlID` is the ID of the control. Row is the row of the matrix as a two byte value. The first row is row zero. `yData` and `xData` are the same as described for *Add Matrix Row*.

Show Message

The *Show Message* makes the message of a message control visible. The format of the command is:

```
SEROUT RxDpin\RTSpin, BaudR,
            [ShowMsg, CtlID]
```

where `ShowMsg` is the constant $04 and `CtlID` is the ID of the message control.

If no client is connected, the message will not be shown, even if a client subsequently connects. This contrasts with some other FlexiPanel servers which will show the message to the first client to connect.

All Controls Demo Sample

The All Controls Demo sample demonstrates most of the controls available to FlexiPanel. It is a large control panel; much larger than you are likely to use in practice. Twenty-four controls are created, which is approaching the limit of what can be stored in the BASIC Stamp in order to program the FlexiPanel module.

First the control panel is created in FlexiPanel Designer. Then it is programmed into FlexiPanel. Finally, a program is written for the BASIC Stamp which interacts with the control panel.

In this example, the Stamp will simply check whether a client has changed the state of the controls and react by writing appropriate information to other controls in the control panel. This uses only a small proportion of the programming space. Plenty of space remains at runtime for the Stamp to perform a useful task.

Control Panel Design

The All Controls Demo control panel is created in *FlexiPanel Designer*. Start *FlexiPanel Designer* and open the file *FxP Stamp Demo.FxP* supplied with the Development Kit. This file is configured for a BS2p. If you wish to run the demo for a different BASIC Stamp, select the *BASIC Stamp properties…* from the View menu and select the correct *Stamp type* in the properties list on the left. Also, if you are not plugging the module in the AppMod slot, you must indicate in the properties list which pins on the module are connected to which I/O ports on the Stamp.

Programming the Control Panel

Once you have opened the file, it is programmed into the FlexiPanel module as follows:

1. In *FlexiPanel Designer*, select *Create BASIC programming code* from the *Target Device* menu. The *Storage Summary* dialog box appears (Figure 2), followed by a dialog box prompting for a file name for the source code file to be created. Choose a file name and press OK.

2. Connect a FlexiPanel BASIC Stamp Edition module to the BASIC Stamp according to the settings you chose in the *Stamp I/O Settings* dialog. (The default values are those for the AppMod slot.) For interest, you may wish to connect an LED to the Data pin (with a 1k current limiting resistor connected to Vss) to watch its behavior.

3. In the BASIC Stamp Editor, open the source file created in step 1.

4. Select Run from the Run menu to program the BASIC Stamp.

5. When the program runs, the BASIC Stamp writes the control panel specification into the module. After a few seconds, the BASIC Stamp Editor will confirm the process is complete writing *Acknowledge: ROM* to the Debug Terminal (Figure 1). The module will then reset itself. If the message does not appear after a few seconds, programming failed. (Try removing and reapplying the power.)

Viewing the Control Panel on a FlexiPanel client

Once the All Controls Demo control panel has been programmed into the FlexiPanel module, the control panel can be viewed from any device running FlexiPanel client software, even before writing a runtime program for the BASIC Stamp.

FlexiPanel Clients include a range of PDAs and cellphones. FlexiPanel Client software may be downloaded free of charge from *www.FlexiPanel.com*.

The All Controls Demo appears on a Pocket PC FlexiPanel client as shown in Figure 3 to Figure 9.

If the FlexiPanel client modifies any controls, the *Data* pin on the module will go high, indicating that a control has been modified.

Writing the Runtime Program

At the bottom of the source file created by FlexiPanel Designer, note the commented-out code showing how the BASIC Stamp should interact with the FlexiPanel module at runtime.

Work from the commented-out code to create the runtime BASIC program shown in Figure 12. (It is

also available as the BASIC file *All Controls Demo.bsp* in the Development Kit.) This code is configured for a BS2p. If you wish to run the demo on a different BASIC Stamp, change the following line:

```
'{$STAMP BS2p}
```

to suit your device.

The structure of the program is as follows:

- Constants are defined which link text names to the control ID codes, pin numbers, etc. If changes are made to these in the FlexiPanel designer, only these constants need to be updated in the runtime program.

- The serial port is initialized and the matrix data controls are given data to display.

- The main program loop is the three lines following the *loop:* label. Inside this loop, the Data pin is inspected to see if the client has updated the control panel data. If it has, the program branches to *ProcessData:*. Normally you would insert any other tasks the BASIC Stamp has to perform in here. (The loop should complete at least 10 times a second for the control panel to be responsive.)

- The *ProcessData:* branch finds out which controls have been modified and reads their new values. According to the controls modified, various tasks may be performed:

  - Pressing the Button makes the Message Box appear and adds a row of data to the Time Chart.

  - Modifying the text of either the Edit Text or the Pass Text sends the new text to the Status text control.

  - Operating any latch control sends the latch's name to the Status text control.

  - Modifying the value of either the *SpinNum* or *OKNum* controls sends the new value to the *FixedNum* control (shifted two decimal places).

  - Modifying the text of either the Edit Text or the Pass Text sends the new text to the Status text control.

  - Locking or unlocking the password control is reported in the Status text control.

  - Modifying the selected item of the *list* control sends the new selected item number to the *FixedNum* control (shifted two decimal places).

  - Opening or closing the *section* control is reported in the Status text control.

  - Modifying the *Set Time* Date Time control sends the new value to the *Clock* Date Time control.

  When you run the program shown in Figure 12, the BASIC Stamp will process any changes the client makes to the controls according to these rules.

Modifying the Runtime Program for Your Application

To modify the All Controls Demo program to suit your application, you will need to take the following steps:

- Create the control panel you want in FlexiPanel Designer.

- Change the constant definitions to those created by FlexiPanel Designer for your control panel.

- Remove matrix data control initialization from the initialization section of the program.

- Replace `PAUSE 10` in the main program loop with any activities you want the BASIC Stamp to perform. These steps may include sending control data to the FlexiPanel module as suggested by the commented out code created by FlexiPanel Designer.

- Modify the *ProcessData:* branch of the program to inspect the values of the controls you created. Use the commented out code created by FlexiPanel Designer to see how to inspect the values of your control panel.

Figures

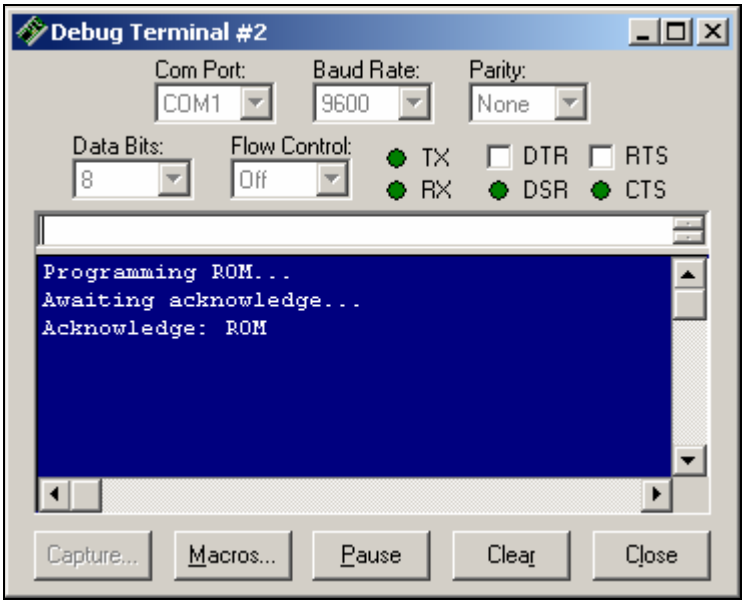


Figure 1 – Programming a FlexiPanel BASIC Stamp Edtion module

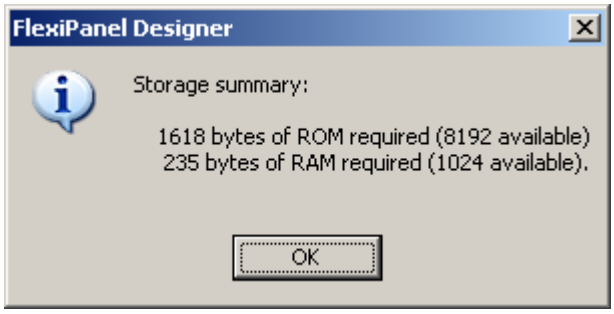


Figure 2 – FlexiPanel BASIC Stamp Storage Summary

Figure 3 – All Controls Demo on a Pocket PC client, page 1.

Figure 4 – All Controls Demo on a Pocket PC client, page 2.

Figure 5 – All Controls Demo on a Pocket PC client, page 3.

Figure 6 – All Controls Demo on a Pocket PC client, section control open.

Figure 7 – All Controls Demo on a Pocket PC client, message box visible.

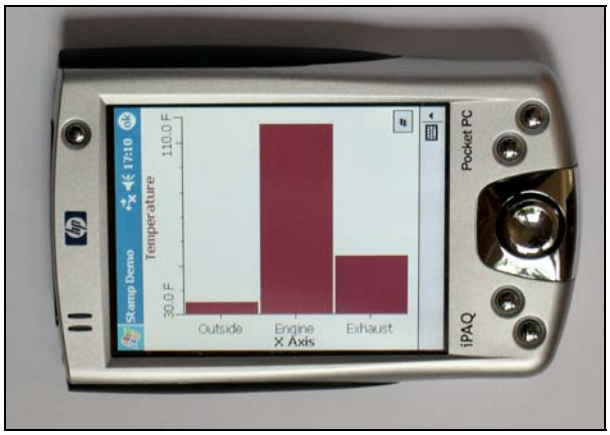


Figure 8 – All Controls Demo on a Pocket PC client, column chart visible

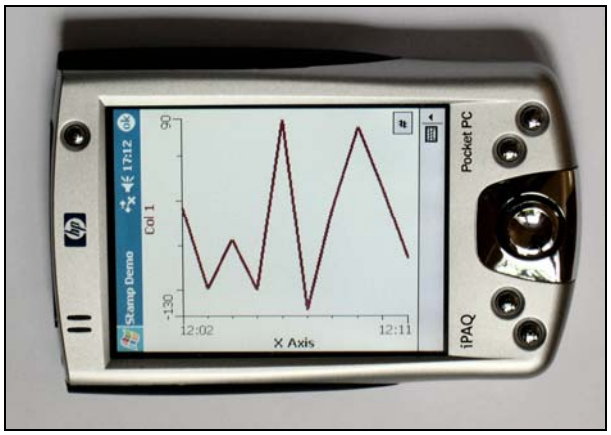


Figure 9 – All Controls Demo on a Pocket PC client, line chart visible

Figure 10 – Example FlexiPanel Smartphone Client Screens


```
'-----------------------------------------------------------------------------------------
'
'  FxP FlexiPanel.bs2
'
'  BS2 code for programming FlexiPanel control panel
'
'  Created at 15:24:30 on 3/27/04
'
'-----------------------------------------------------------------------------------------

'  Initialization

'{$STAMP BS2}

'-----------------------------------------------------------------------------------------

'  Data

ROMData DATA $01, $00, $00, $00, $00, $00, $46, $6C, $65, $78, $69, $50, $61, $6E, $65, $6C
                    {Many rows of data like this}
        DATA $01, $91, $00, $00, $00, $00, $00, $00, $00, $00, $00, $00, $00, $00, $00, $00

'-----------------------------------------------------------------------------------------

' Variables

cdata   VAR     BYTE
ack     VAR     BYTE(3)
i       VAR     WORD


'-----------------------------------------------------------------------------------------

' Start of code

' 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)

'  Wait 50 ms for FxP to initialize; if you use pullup resistors on CTS/Rx this is not needed

        pause 50

'  Program ROM

        debug "Programming ROM...", CR
        serout 2\6, 32, [$81, $06]
        for i = 0 to $5F
            read ROMData+i, cdata
            serout 2\6, 32, [cdata]
        next
        debug "Awaiting acknowledge...", CR
        serin  4\8, 32, [STR ack\3]
        debug "Acknowledge: ", STR ack\3, CR


'  Finished

        stop
```

Figure 11 – FlexiPanel BASIC Stamp Edtion programming script. (Continues…)

```
'-------------------------------------------------------------------------------------------


'  The following constants will be useful:

        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     32      ' Baud mode

        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_Text1 con $01      ' Text1 control ID


'-------------------------------------------------------------------------------------------


'  Use the following sample code to set up the Stamp I/O
'
'       DIR2  = 1     ' Tx to FleixPanel (SEROUT output from Stamp)
'       OUT2  = 1     ' Initialize Tx as high
'       DIR6  = 0     ' CTS from FleixPanel  (SEROUT flow control input to Stamp)
'       DIR4  = 0     ' Rx from FleixPanel (SERIN input to Stamp)
'       DIR8  = 1     ' RTS to FleixPanel   (SERIN flow control output from Stamp)
'       OUT8  = 1     ' Initialize RTS as high
'       DIR12  = 0     ' Data from FleixPanel (SERIN input to Stamp)


'-------------------------------------------------------------------------------------------


'  Use the following sample code to read the text of the 'Text1' text control:
'
'       text  VAR     BYTE(10)
'
'       serout TxPin\CTSPin, BaudM, [GetData, ID_Text1]
'       serin  RxPin\RTSPin, BaudM, [STR text\10]


'  Use the following sample code to set the text of the 'Text1' text control:
'
'       serout TxPin\CTSPin, BaudM, [SetData, ID_Text1, STR text\10]

'  All strings must be zero terminated!
'  Note that all 10 bytes must be read or written each time, even if you don't need them all


'-------------------------------------------------------------------------------------------


'  Use the following sample code to see if any controls have been modifed by client
'  substituting for [] as appropriate
'  (returns $00 if no controls have been modified, otherwise the ID of a modified control):
'
'       modctl   VAR     BYTE
'
'       serout TxPin\CTSPin, BaudM, [GetMod]
'       serin  RxPin\RTSPin, BaudM, [modctl]


'-------------------------------------------------------------------------------------------
```

Figure 11 – FlexiPanel BASIC Stamp Edition programming script. (Continued)
Note the commented out sample code automatically provided for the control panel.

Figure 12 – FlexiPanel BASIC Stamp Edition runtime BASIC Program.

```
'-----------------------------------------------------------------------------------------------
'
'  All Controls Demo.bsp
'
'-----------------------------------------------------------------------------------------------

'  Initialization

'{$STAMP BS2p}

'-----------------------------------------------------------------------------------------------


'  The following constants will be useful:

        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

        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_Button CON $01      ' Button control ID
        ID_Status CON $02      ' Status control ID
        ID_EditText CON $03      ' EditText control ID
        ID_PassText CON $04      ' PassText control ID
        ID_Latch CON $05      ' Latch control ID
        ID_Reset CON $06      ' Reset control ID
        ID_Radio_1 CON $07      ' Radio_1 control ID
        ID_Radio_2 CON $08      ' Radio_2 control ID
        ID_Momentary CON $09      ' Momentary control ID
        ID_Fixed_Num CON $0A      ' Fixed_Num control ID
        ID_Spin_Num CON $0B      ' Spin_Num control ID
        ID_OK_Num CON $0C      ' OK_Num control ID
        ID_Fixed_Password CON $0D      ' Fixed_Password control ID
        ID_F_Pwd_Child CON $0E      ' F_Pwd_Child control ID
        ID_Table CON $0F      ' Table control ID
        ID_XY_Chart CON $10      ' XY_Chart control ID
        ID_Label_Chart CON $11      ' Label_Chart control ID
        ID_Time_Chart CON $12      ' Time_Chart control ID
        ID_List CON $13      ' List control ID
        ID_Section CON $14      ' Section control ID
        ID_Section_Child CON $15      ' Section_Child control ID
        ID_Set_Time CON $16      ' Set_Time control ID
        ID_Clock CON $17      ' Clock control ID
        ID_Message CON $18      ' Message control ID


'-----------------------------------------------------------------------------------------------

' Start of code

' Test the flexipanel device


modctl    VAR     Byte
state    VAR  Byte
dummyByte  VAR     state
numval    VAR  Word
text    VAR  Byte(10)
Mnt    VAR  Byte
Rnd     VAR  Word
ss    VAR  Byte
mm    VAR  Byte
hh    VAR  Byte


' 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)


'    wait for FxP module to initialize
    PAUSE 50
WaitInit
    IF DataPin = 1 THEN WaitInit:

'   set up data in matrices

    SEROUT TxPin\CTSPin, BaudM, [SetRow, ID_Table, 0, 0, 1, 2 ]
    SEROUT TxPin\CTSPin, BaudM, [SetRow, ID_Table, 1, 0, 3, 4 ]
    SEROUT TxPin\CTSPin, BaudM, [SetRow, ID_Table, 2, 0, 5, 6 ]
    SEROUT TxPin\CTSPin, BaudM, [SetRow, ID_Table, 3, 0, 7, 8 ]
    SEROUT TxPin\CTSPin, BaudM, [SetRow, ID_Table, 4, 0, 9, 10 ]

    SEROUT TxPin\CTSPin, BaudM, [AddRow, ID_XY_Chart, 1, 1 ]
    SEROUT TxPin\CTSPin, BaudM, [AddRow, ID_XY_Chart, 1, 9 ]
    SEROUT TxPin\CTSPin, BaudM, [AddRow, ID_XY_Chart, 9, 1 ]
    SEROUT TxPin\CTSPin, BaudM, [AddRow, ID_XY_Chart, 9, 9 ]
    SEROUT TxPin\CTSPin, BaudM, [AddRow, ID_XY_Chart, 5, 5 ]

    SEROUT TxPin\CTSPin, BaudM, [SetRow, ID_Label_Chart, 0, 0, 100, 1 ]
    SEROUT TxPin\CTSPin, BaudM, [SetRow, ID_Label_Chart, 1, 0, 52, 4 ]
    SEROUT TxPin\CTSPin, BaudM, [SetRow, ID_Label_Chart, 2, 0, 34, 2 ]

    Mnt= 0
    Rnd = 20567

loop:
    IF DataPin = 1 THEN ProcessData
    PAUSE 10
    GOTO loop


ProcessData:
    ' see what controls have been modified
    SEROUT TxPin\CTSPin, BaudM, [GetMod]
    SERIN  RxPin\RTSPin, BaudM, [modctl]

dummyByte2  VAR     Byte

    IF modctl = ID_Button THEN ProcessButton
    IF modctl = ID_EditText THEN ProcessEditText
    IF modctl = ID_PassText THEN ProcessPassText
    IF modctl = ID_Latch THEN ProcessLatch
    IF modctl = ID_Momentary THEN ProcessMomentary
    IF modctl = ID_Reset THEN ProcessReset
    IF modctl = ID_Radio_1 THEN ProcessRadio_1
    IF modctl = ID_Radio_2 THEN ProcessRadio_2
    IF modctl = ID_Fixed_Password THEN ProcessFixed_Password
    IF modctl = ID_F_Pwd_Child THEN ProcessF_Pwd_Child
    IF modctl = ID_Spin_Num THEN ProcessSpinNum
    IF modctl = ID_OK_Num THEN ProcessOKNum
    IF modctl = ID_List THEN ProcessList
    IF modctl = ID_Section THEN ProcessSection
    IF modctl = ID_Section_Child THEN ProcessSection_Child
    IF modctl = ID_Set_Time THEN ProcessSet_Time
    GOTO loop

ProcessButton:

    ' get value; can ignore it since it will be $FF = "pressed", but this clears data pin
    SEROUT TxPin\CTSPin, BaudM, [GetData, ID_Button ]
    SERIN  RxPin\RTSPin, BaudM, [dummyByte]

    ' tell status about it (must pad string out with zeroes to make 10 bytes, zero terminating)
    SEROUT TxPin\CTSPin, BaudM, [SetData, ID_Status , "Button", 0, 0, 0, 0]

    ' log something in the data log (last two bytes are year 2000)
    Mnt= Mnt+ 1
    RANDOM Rnd
    SEROUT TxPin\CTSPin, BaudM, [AddRow, ID_Time_Chart, Rnd.LOWBYTE, 0, Mnt, 12, 1, 6, 1, 208, 8 ]

    ' show message; not this only
    SEROUT TxPin\CTSPin, BaudM, [ShowMsg, ID_Message ]

    GOTO ProcessData


ProcessEditText:
```



```
    ' get value
    SEROUT TxPin\CTSPin, BaudM, [GetData, ID_EditText ]
    SERIN  RxPin\RTSPin, BaudM, [STR text\10]

    ' send to fixed text
    SEROUT TxPin\CTSPin, BaudM, [SetData, ID_Status , STR text\10]

    GOTO ProcessData


ProcessPassText:

    ' get value
    SEROUT TxPin\CTSPin, BaudM, [GetData, ID_PassText ]
    SERIN  RxPin\RTSPin, BaudM, [STR text\10]

    ' send to fixed text
    SEROUT TxPin\CTSPin, BaudM, [SetData, ID_Status , STR text\10]

    GOTO ProcessData


ProcessLatch:

    ' get value
    SEROUT TxPin\CTSPin, BaudM, [GetData, ID_Latch]
    SERIN  RxPin\RTSPin, BaudM, [state]

    ' tell status about it
    IF state = $FF THEN LatchOn
    SEROUT TxPin\CTSPin, BaudM, [SetData, ID_Status , "Latch Off", 0]
    GOTO ProcessData

LatchOn:
    SEROUT TxPin\CTSPin, BaudM, [SetData, ID_Status , "Latch On", 0, 0]
    GOTO ProcessData


ProcessMomentary:

    ' get value
    SEROUT TxPin\CTSPin, BaudM, [GetData, ID_Momentary]
    SERIN  RxPin\RTSPin, BaudM, [state]

    ' tell status about it
    IF state = $FF THEN MomentaryOn
    SEROUT TxPin\CTSPin, BaudM, [SetData, ID_Status , "Momty Off", 0]
    GOTO ProcessData

MomentaryOn:
    SEROUT TxPin\CTSPin, BaudM, [SetData, ID_Status , "Momty On", 0, 0]
    GOTO ProcessData


ProcessReset:

    ' get value - always zero since non latching button, but this clears data pin
    SEROUT TxPin\CTSPin, BaudM, [GetData, ID_Reset]
    SERIN  RxPin\RTSPin, BaudM, [dummyByte]

    ' tell status about it
    SEROUT TxPin\CTSPin, BaudM, [SetData, ID_Status , "Reset", 0, 0, 0, 0, 0]
    GOTO ProcessData


ProcessRadio_1:

    ' get value
    SEROUT TxPin\CTSPin, BaudM, [GetData, ID_Radio_1]
    SERIN  RxPin\RTSPin, BaudM, [state]

    ' tell status about it
    IF state = 0 THEN ProcessData
    SEROUT TxPin\CTSPin, BaudM, [SetData, ID_Status , "Radio 1", 0, 0, 0]
    GOTO ProcessData


ProcessRadio_2:

    ' get value
    SEROUT TxPin\CTSPin, BaudM, [GetData, ID_Radio_2]
    SERIN  RxPin\RTSPin, BaudM, [state]
```

```
    ' tell status about it
    IF state = 0 THEN ProcessData
    SEROUT TxPin\CTSPin, BaudM, [SetData, ID_Status , "Radio 2", 0, 0, 0]
    GOTO ProcessData


ProcessFixed_Password:

    ' get value
    SEROUT TxPin\CTSPin, BaudM, [GetData, ID_Fixed_Password]
    SERIN  RxPin\RTSPin, BaudM, [state]

    ' tell status about it
    IF state = $FF THEN PassOn
    SEROUT TxPin\CTSPin, BaudM, [SetData, ID_Status , "Pass Off", 0, 0]
    GOTO ProcessData

PassOn:
    SEROUT TxPin\CTSPin, BaudM, [SetData, ID_Status , "Pass On", 0, 0, 0]
    GOTO ProcessData


ProcessF_Pwd_Child

    ' get value; this clears data pin
    SEROUT TxPin\CTSPin, BaudM, [GetData, ID_F_Pwd_Child]
    SERIN  RxPin\RTSPin, BaudM, [dummyByte]

    GOTO ProcessData


ProcessSpinNum:

    ' get value, ignoring high word
    SEROUT TxPin\CTSPin, BaudM, [GetData, ID_Spin_Num ]
    SERIN  RxPin\RTSPin, BaudM, [numval.LOWBYTE, numval.HIGHBYTE, dummyByte, dummyByte]

    ' send it to fixed value
    SEROUT TxPin\CTSPin, BaudM, [SetData, ID_Fixed_Num, numval.LOWBYTE, numval.HIGHBYTE, 0, 0]
    GOTO ProcessData


ProcessOKNum:

    ' get value, ignoring high word
    SEROUT TxPin\CTSPin, BaudM, [GetData, ID_OK_Num ]
    SERIN  RxPin\RTSPin, BaudM, [numval.LOWBYTE, numval.HIGHBYTE, dummyByte, dummyByte]

    ' send it to fixed value
    SEROUT TxPin\CTSPin, BaudM, [SetData, ID_Fixed_Num, numval.LOWBYTE, numval.HIGHBYTE, 0, 0]
    GOTO ProcessData


ProcessList:
    ' get value
    SEROUT TxPin\CTSPin, BaudM, [GetData, ID_List ]
    SERIN  RxPin\RTSPin, BaudM, [numval.LOWBYTE, numval.HIGHBYTE, dummyByte, dummyByte]

    ' send it, plus one, to fixed value
    numval = numval + 1
    SEROUT TxPin\CTSPin, BaudM, [SetData, ID_Fixed_Num, numval.LOWBYTE, numval.HIGHBYTE, 0, 0]
    GOTO ProcessData


ProcessSection:
    ' get value
    SEROUT TxPin\CTSPin, BaudM, [GetData, ID_Section]
    SERIN  RxPin\RTSPin, BaudM, [state]

    ' tell status about it
    IF state = $FF THEN SectOn
    SEROUT TxPin\CTSPin, BaudM, [SetData, ID_Status , "Sctn Off", 0, 0]
    GOTO ProcessData

SectOn:
    SEROUT TxPin\CTSPin, BaudM, [SetData, ID_Status , "Sctn On", 0, 0, 0]
    GOTO ProcessData


ProcessSection_Child
    ' get value; this clears data pin
    SEROUT TxPin\CTSPin, BaudM, [GetData, ID_Section_Child]
```



```
    SERIN  RxPin\RTSPin, BaudM, [dummyByte]
    GOTO ProcessData


ProcessSet_Time:
    ' get time value, ignore date
    SEROUT TxPin\CTSPin, BaudM, [GetData, ID_Set_Time]
    SERIN  RxPin\RTSPin, BaudM, [ss, mm, hh, dummyByte, dummyByte, dummyByte, dummyByte, dummyByte]

    ' set time of fixed control
    SEROUT TxPin\CTSPin, BaudM, [SetData, ID_Clock, ss, mm, hh, 1, 6, 1, 208, 8]
    GOTO ProcessData
```

FlexiPanel BASIC Stamp Edition Development Kit

The FlexiPanel BASIC Stamp Edition Development Kit is comprised of the following files:

- This documentation, *FlexiPanelBasicStamp.pdf*.
- The control panel design tool *FlexiPanelDesigner.exe* application.
- The documentation for FlexiPanel Designer design tool, *FlexiPanelDesigner.pdf*.
- The *FxP Stamp Demo.FxP* FlexiPanel Designer control panel design file for the All Controls Demo.
- The *FxP Stamp Demo.bsp* BASIC file as created by FlexiPanel Designer.
- The *All Controls Demo.bsp* BASIC file for the All Controls Demo.

Revision History

The table below details the minor revisions that have been made to FlexiPanel BASIC Stamp Edition.

Date	***Revision Type***	***Detail***
23 Mar 2004	Product release	Initial release
8 Apr 2004	Code	Section & password not closing on disconnect – corrected
28 Apr 2004	Enhancement	AckData function added
		CtlMod function added
28 Apr 2004	Code	RAM storage error if more than 0x100 bytes allocated – corrected
3 May 2004	Code	If Stamp updates Section or Password state, resend control panel – done
		If Stamp queries state of button state must be reset - done

Contacting FlexiPanel Ltd

For technical support regarding FlexiPanel technology, please contact:

technical.support@FlexiPanel.com

We recognize our customers come up with the best ideas for improving our products. If you have any comments regarding FlexiPanel products or documentation, please send also send them to the technical support address.

For other contact points at FlexiPanel Ltd, please consult the Contact Us page of our web site, *www.FlexiPanel.com*.

