Simple Python

A general Windows code example for all Matrix Orbital Displays

Application Note

Revision 1.0

Installing Python

Python is a simple yet powerful language marked by a unique indentation structure which differentiates major code sections. Installations for a number of operating systems and a tremendous amount of documentation on the language can be found at http://python.org/download/. For this example, code revision 2.7.2 was implemented.

While Python is installed with a number of useful modules, a serial port encapsulation was required for this example. For serial communication, PySerial, specifically build 2.6, can be downloaded from http://pyserial.sourceforge.net/. Installation instructions are provided, but essentially you'll open the Python command line application with administrator rights, navigate to unpacked PySerial folder, and run setup.py install.

Modules, such as PySerial, can be loaded in the Python environment using the import command. In addition to the main program and serial port module described above, it may be useful to download a text editor to create Python modules, for this application Notpad++ was used. With all of the right tools in place, it is easy to construct the perfect Python program for your Matrix Orbital display.

Serial Communication

Interfacing to any Matrix Orbital display begins with the correct selection of the serial port and associated attributes. In Python a serial port object can be defined as a file using just the port reference, where COM1 is designated by 0, or with a whole host of other settings.

```
import serial
port = serial.Serial(8,19200,serial.EIGHTBITS,serial.PARITY_NONE,
serial.STOPBITS_ONE,timeout=5,rtscts = False)
```

Once defined, serial protocol specific attributes can be altered. Most Matrix Orbital displays will run at a default speed of 19200 baud, with 8 bits of data, no parity bit, and 1 stop bit. Flow control is normally turned off, especially for alphanumeric displays, but please consult your display manual to confirm.

```
port.baudrate = 19200
port.bytesize = 8
port.parity = 'N'
port.stopbits = 1
port.rtscts = False
```

Finally, if information is to be read from the serial port it is important to set the timeout value. This will ensure that the program will not wait infinitely for a byte to be read should an error occur. Please note Python specifies the timeout attribute in seconds.

```
port.setTimeout(5)
```

Once a port is selected and its attributes set, text can easily be transmitted to an attached display.

Text

Once a serial port is created and configured, text can be written to it as a series of bytes. In Python a string is stored simply as an array of bytes which can be transmitted directly to a display.

```
message = "Hello World!"
port.write(message)
```

While no conversion is required to display plain text, it is slightly more difficult to send non-printable characters, such as commands, to a Matrix Orbital display in the Python language.

Commands

Commands are often unprintable characters, and as such, they are usually defined as an array of bytes as decimal or hexadecimal values, using chr() to convert these values to single bytes.

```
clearscreen = [chr(254), chr(88)]
for i in clearscreen:
        port.write(i)
```

This time the values are written to the serial port individually to ensure they are not formatted as a string, but sent as individual bytes.

On Reading

Python offers an extremely easy way to read key presses from a Matrix Orbital display through an implementation much like file input/output. All responses from the display are saved in a buffer and can be read at any time by specifying the number of bytes to be read.

```
key = port.read(1)
```

As mentioned, the setTimeout() function should be used before reading to ensure the module does not hang when no response is received. Boolean tests can be performed on the information received from the display to determine its' validity. For example:

```
key.isalpha()
key.isdigit()
```

Finally, as the serial port object is treated as a buffer, it may be useful to flush it at times to ensure the information read is current.

```
port.flushInput()
```

In Closing

Once all transactions with an attached display are complete the serial port used should be closed before exiting the program.

```
port.close()
```

With this, all basic input and output aspects of a Matrix Orbital display have been explored in the Python programming environment. These concepts provide a scratchpad from which an epic tome of applications may be adapted for your Matrix Orbital display.

Contact

Sales
Phone: 403.229.2737
Email: sales@matrixorbital.ca

Support
Phone: 403.204.3750
Email: support@matrixorbital.ca

Online
Purchasing: www.matrixorbital.com
Support: www.matrixorbital.ca