Using SPI Interface in *smart*BASIC with DotStar Example

*smart*BASIC

Application Note *v1.0*

INTRODUCTION

*smart*BASIC, Laird's event driven programming language, makes Bluetooth Smart technology development quicker and simpler, vastly cutting down time to market. *smart*BASIC is not just limited for use in Laird's Bluetooth and BLE modules. In fact, the programming language will be included in all the latest product developments from Laird. The goal of this document is to provide an example on how to implement SPI in smartBASIC using a DotStar LED strip and other SPI devices. In this demonstration we used a DotStar LED strip from AdaFruit, however it is acceptable to use a different product if it has a max of 255 LED pixels.

REQUIREMENTS

The following equipment and utilities are required:

- Windows PC with UwTerminal installed (v7.00 or greater)
- *smart*BASIC enabled module with SPI
- "dotstar.example.sb" and "DotStar.sblib"
- DotStar LED strip (255 LED pixels max)
- 5V power supply

MODULE SETUP

Setup of the smartBASIC module to the DotStar LED strip is a simple matter of connecting the SPI interface pins to the LED strip. The MISO line is not connected up as the DotStar does not return any data on the SPI interface. Refer to the documentation of your specific module for following pins listed in Table 1.

Table 1: Connecting SPI interface pins to LED strip

***smart*BASIC**	**DotStar LED Strip**
GND	GND
SPI MOSI	DI
SPI CLK	CI

Note: Power must be delivered separately to the DotStar LED strip using an external 5V power supply to ensure enough current is provided. The datasheet indicates a maximum of 26.5mA is pulled per RGB LED.

Plug the module into the PC and connect to it using UwTerminal with the appropriate serial settings.

OVERVIEW OF THE DOTSTAR FILES

"dotstar.example.sb" is a smartBASIC example of one way to implement SPI. It makes a call to '#include "DotStar.sblib"' and uses the library's functions to manage the DotStar LEDs. An overview of the SPI related functions in this library are located below.

Here we can see the function that primes the SPI interface and ensure it is working properly. If the interface fails to open the function it will return an error, otherwise it will return 0. This function should be called first to ensure that the SPI interface is operational and to configure the number of LEDs in the strip.

```
'//-------------------------------------------------------------------------
'// Prime SPI for output
'//-------------------------------------------------------------------------
FUNCTION DotStarBein(n as INTEGER)
    numLEDs = n
    resultcode = SPIOPEN(0, 1000000, 0, hSPI) 'set SPI mode to 0, 1MHz, MSB First, w/Handle attached to hSPI
        AssertRC(resultcode, 60)
ENDFUNC resultcode
```

From here the rest is handled in the main subroutine below.

```
'//-------------------------------------------------------------------------
'//Issue/Refresh color data to strip.
'//-------------------------------------------------------------------------
SUB DotStarUpdate()
    dim n : n = numLEDs
    dim r, g, b, i
    dim r$, g$, pixel$

    resultcode = SPIOPEN(0, 1000000, 0, hSPI) 'Set SPI mode to 0, 1MHz, MSB First, w/ Handle attached to hSPI
    for i = 1 to 4
       resultcode = SPIWRITE(startFrame$)
    next

    while (n > 0)
        n = n - 1
        pixelcolor = getDotStarPiexelColor(n, r, g, b)
        SPRINT #r$, INTEGER.H'r
        SPRINT #g$, INTEGER.H'g
        SPRINT #b$, INTEGER.H'b

        SPRINT #spiTX$, RIGHT$(brightness$,2); RIGHT$(b$, 2); RIGHT$(g$, 2); RIGHT$(r$, 2)
        'Sent out in reverse order for the DotStar to frame pixels properly.
        'See LED datasheet at http://www.adafruit.com/datasheets/APA102.pdf
        pixel$ = StrDehexize$(spiTX$)
        resultcode = SpiWrite(pixel$)
    endwhile

    'for i = 1 to 4
        'resultcode = SPIWRITE(endFrame$)
    'next
    SpiClose(hSPI)
ENDSUB
```

'DotStarUpdate()' manages the DotStar LEDs by retrieving colors of each of the pixels set by the user from an earlier state, formatting them as 'frames', then writes the color data over the SPI interface to the device(s).

Note: The SPI interface speed of 1 MHz may be changed to suit your specific module by modifying the second parameter of the SPIOpen(). Reference your module's Extensions Guide for more detail.

Implementing the DotStar Library

Compile and load the "dotstar.example.sb" script through UwTerminal to see implementation of the DotStar library in smartBASIC using SPI. An example can be seen in action in this video. To use the library simply '#include' it in your main .sb script and make the function calls available from the library.

Implementing the SPI with Other Devices

As shown, smartBASIC allows users to easily configure, write, read, and close the SPI interface using the SPIOpen(), SPIWrite(), SPIRead(), and SPIClose() functions respectively. Simultaneous reading and writing is also possible using the SPIReadWrite() function. They can be used in the top-level of the script or a custom library designed to use other devices. Refer to the smartBASIC Core Functionality guide for a full description of each of the functions.

To implement SPI, simply call the SPIOpen() function with appropriate parameters, read and write any data to the device, then close the SPI interface using SPIClose(). If using more than one SPI device, additional GPIO(s) will need to be configured in order to select which device the SPI bus is reading and writing to. Open "DotStar.sblib" to see an example interface is written for any particular SPI device. Feel free to use this as a guideline to design other libraries for different SPI devices.

References

Information regarding the 'framing' of the DotStar LED packets can be referenced in the data sheet: http://www.adafruit.com/datasheets/APA102.pdf

For more information on the SPI as well as any smartBASIC commands used in this application note, refer to the your specific smartBASIC Module user guide which can be accessed from the Embedded Wireless Solutions Support Center: https://laird-ews-support.desk.com/

Product information can also be accessed from the products page on the Laird website: http://www.lairdtech.com/products/category/577

Revision History

Version	Date	Notes	Approver
1.0	31 July 2015	Initial Release	Jonathan Kaye