Application Note

AN_371

FT90x WS2812 Example

Version 1.0

Issue Date: 2015-10-06

This note describes how to use GPIO to communicate with the WS2812 "Intelligent control LED integrated light source" on the MM900 EV Module.

Future Technology Devices International Limited (FTDI)
Unit 1, 2 Seaward Place, Glasgow G41 1HH, United Kingdom
Tel.: +44 (0) 141 429 2777   Fax: + 44 (0) 141 429 2758
Web Site: http://ftdichip.com

Table of Contents

1 Introduction

This Application Note describes and explains the FT90x WS2812 Example. The FT90x WS2812 Example demonstrates how to interface with the World Semi WS2812 "Intelligent control LED integrated light source" with an FT90x microcontroller.

1.1 Overview

This document will describe the design and implementation of the FT90x WS2812 Example. The FT90x WS2812 Example will control two WS2812 LEDs on a MM900 Evaluation Board and fade them through various hues. WS2812 LEDs use a proprietary 1-Wire-like protocol to shift in colour values and allow data to be daisy chained to other WS2812 LEDs.

This document is intended to demonstrate the bridging capabilities of the FT90x family of microcontrollers.

1.2 License

2 Theory

2.1 WS2812 LEDs

WS2812 Serial LEDs use a One Wire protocol for communication which determines a bit based upon the pulse width of the symbol. The WS2812 protocol uses three symbols: a 0 bit, a 1 bit, and a Reset symbol (used to latch in data). These three symbols are described in Figure 1: Timing Diagram for WS2812 Symbols

and Table 1: Timing for WS2812 Symbols

.

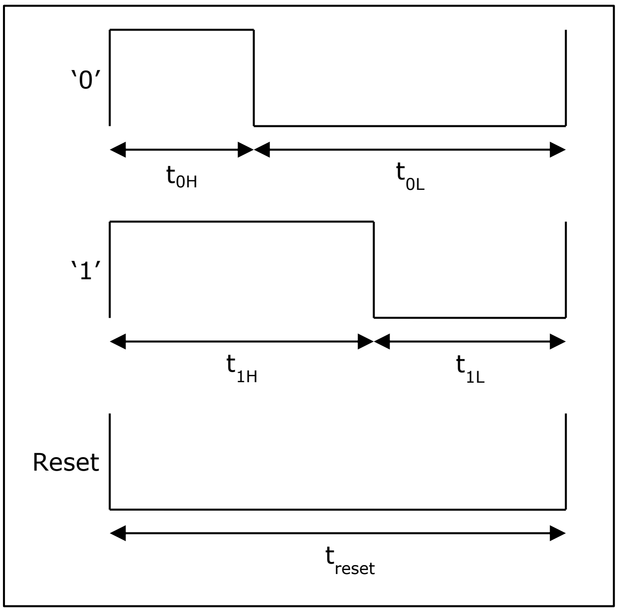

Figure 1: Timing Diagram for WS2812 Symbols

Name	Description	Min	Typ	Max	Unit
t_{0H}	0 code, High Voltage Time	200	350	500	nsec
t_{0L}	0 code, Low Voltage Time	550	700	850	nsec
t_{1H}	1 code, High Voltage Time	650	800	950	nsec
t_{1L}	1 code, Low Voltage Time	450	600	750	nsec
t_{reset}	Reset code, Low Voltage Time	50	-	-	µsec

Table 1: Timing for WS2812 Symbols

Colour is transferred to a WS2812 LED in 24 bit chunks (an 8-bit value of Green, an 8-bit value of Red, and an 8-bit value of Blue) Most Significant Bit first. Multiple colours can be transferred in order to update multiple LEDs. At the end of an update a Reset must be sent which will cause the LEDs to latch in the new colour.

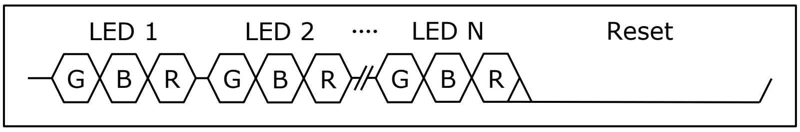

Figure 2: Timing Diagram for WS2812 Update

3 Implementation

3.1 Bit-banging the WS2812 Protocol via GPIO

Several methods of communicating with WS2812 are available (UART, SPI, GPIO); however for the purposes of this demo, GPIO was selected.

In order to simplify transferring bits to the WS2812, a common symbol time for '1's and '0's is derived and separated up into 8 sections to allow the use of an 8-bit long pattern. The duration of the '1' and '0' symbols lie within 1.25 μsec allowing for a 150 nsec bit time for the 8-bit pattern.

For a '1' symbol, the signal needs to be high for 650 to 950 nsec which corresponds to either 5 or 6 bit times.

For a '0' symbol, the signal needs to be high for 200 to 500 nsec which corresponds to either 2 or 3 bit times.

Assuming that a '1' symbol uses 5 bit times and a '0' symbol uses 3 bit times, this means that a '1' symbol will be high for 750 nsec and low for 450 nsec giving a mask of $F8_h$ (0b 1111 1000), and a '0' symbol will be high for 450 nsec and low for 750 nsec giving a mask of $E0_h$ (0b 1110 0000).

The function ws2812_init in ws2812_gpio.c (shown in Table 2: Code Listing for Function ws2812_init

) sets up the GPIO pin as an output.

```
void ws2812_init(void)
{
    gpio_function(WS2812_GPIO, pad_func_0);
    gpio_dir(WS2812_GPIO, pad_dir_output);
}
```

Table 2: Code Listing for Function ws2812_init

The function ws2812_shiftout in ws2812_gpio.c (shown in Table 3: Code Listing for Function ws2812_shiftout

) implements shifting out a byte of data to a GPIO pin using the WS2812 protocol.

- The corresponding GPIO register and bitmask is determined for the GPIO pin defined in *WS2812_GPIO*.
- A loop iterates over every bit in parameter *b* starting at Most Significant Bit and working down to the Least Significant Bit.
- Within the loop, the corresponding symbol mask is chosen and bit-banged out on the GPIO pin. NOOP instructions are used to generate the precise timing.

```
void ws2812_shiftout(uint8_t b)
{
    uint8_t i;
    register uint8_t sym = 0, sym0 = WS2812_GPIO_SYM0, sym1 = WS2812_GPIO_SYM1;
    register uint32_t mask = 0;
    register volatile uint32_t* gpioval;

#if (WS2812_GPIO < 32)
    /* 0 - 31 */
    gpioval = &(GPIO->GPIO00_31_VAL);
    mask = 1 << WS2812_GPIO;
#elif (WS2812_GPIO < 64)
    /* 32 - 63 */
    gpioval = &(GPIO->GPIO32_63_VAL);
    mask = 1 << (WS2812_GPIO - 32);
#else
    /* 64 - 66 */
    gpioval = &(GPIO->GPIO64_66_VAL);
    mask = 1 << (WS2812_GPIO - 64);
#endif

    for(i = 0; i < 8; ++i)
    {
        if(b & 0x80)     sym = sym1;
        else             sym = sym0;

        if (sym & 0x80) *gpioval |= mask;
        else            *gpioval &= ~mask;
        asm_noop();
        asm_noop();
        asm_noop();
        asm_noop();
        if (sym & 0x40) *gpioval |= mask;
        else            *gpioval &= ~mask;
        asm_noop();
        asm_noop();
        asm_noop();
        if (sym & 0x20) *gpioval |= mask;
        else            *gpioval &= ~mask;
        asm_noop();
        asm_noop();
        asm_noop();
        asm_noop();
        if (sym & 0x10) *gpioval |= mask;
        else            *gpioval &= ~mask;
        asm_noop();
        asm_noop();
        asm_noop();
        if (sym & 0x08) *gpioval |= mask;
        else            *gpioval &= ~mask;
        asm_noop();
        asm_noop();
        asm_noop();
```

```
        asm_noop();
        if (sym & 0x04) *gpioval |= mask;
        else            *gpioval &= ~mask;
        asm_noop();
        asm_noop();
        asm_noop();
        if (sym & 0x02) *gpioval |= mask;
        else            *gpioval &= ~mask;
        asm_noop();
        asm_noop();
        asm_noop();
        asm_noop();
        if (sym & 0x01) *gpioval |= mask;
        else            *gpioval &= ~mask;

        b <<= 1;
    }
}
```

Table 3: Code Listing for Function ws2812_shiftout

Using this function, we can create the function ws2812_write in ws2812_gpio.c (shown in Table 4: Code Listing for Function ws2812_write

) which sends out a RGB colour value on the GPIO pin.

```
void ws2812_write(uint8_t r, uint8_t g, uint8_t b)
{
    ws2812_shiftout(g);
    ws2812_shiftout(r);
    ws2812_shiftout(b);
}
```

Table 4: Code Listing for Function ws2812_write

3.2 HSV to RGB Colour Space Conversion

Colour can be described in many different ways. Typically, colour is given as a level of Red, Green and Blue (RGB) which is what the WS2812 LED uses to display a colour. However, in order to simplify linearly fading through colour, another colour space can be used. This application uses the Hue, Saturation and Value (HSV) colour space which allows for a more intuitive approach to colour manipulation:

- Hue defines what base colour to use.
- Saturation defines its colorfulness. A high saturation gives a bright colour, a low saturation gives grey.
- Value defines the level of brightness relative to full white.

Therefore, by having a fixed saturation and value, and cycling through the Hue, a gradual fade of colour can be generated.

A conversion function is needed to convert from a HSV to a RGB colour space in order for HSV values to be used with WS2812 LEDs. The function hsv2rgb in rgbhsv.c (shown in Table 5: Code Listing for Function hsv2rgb

) converts HSV values to RGB values. This function uses fixed point math since the FT32 core does not support floating point.

```
void hsv2rgb(uint8_t h, uint8_t s, uint8_t v,
        uint8_t* r, uint8_t* g, uint8_t* b)
{
    /* Based on http://stackoverflow.com/a/14733008 */

    uint8_t region, remainder, p, q, t;

    if (s == 0)
    {
        *r = v;
        *g = v;
        *b = v;
    }
    else
    {
        region = h / 43;
        remainder = (h - (region * 43)) * 6;

        p = (v * (255 - s)) >> 8;
        q = (v * (255 - ((s * remainder) >> 8))) >> 8;
        t = (v * (255 - ((s * (255 - remainder)) >> 8))) >> 8;

        switch (region)
        {
        case 0:
            *r = v; *g = t; *b = p;
```

```
            break;
        case 1:
            *r = q; *g = v; *b = p;
            break;
        case 2:
            *r = p; *g = v; *b = t;
            break;
        case 3:
            *r = p; *g = q; *b = v;
            break;
        case 4:
            *r = t; *g = p; *b = v;
            break;
        default:
            *r = v; *g = p; *b = q;
            break;
        }
    }
}
```

Table 5: Code Listing for Function hsv2rgb

3.3 Main Application

This application (shown in Table 6: Code Listing for the Main Application

) will fade the two WS2812 LEDs on the MM900 board through different hues with one LED having the opposite hue of the other LED.

```
int main(void)
{
    setup();
    for(;;) loop();

    return 0;
}

void setup()
{
    uint8_t i;

    for (i=0; i<N_LEDS; ++i)
    {
        /* Every LED has a different Hue Offset */
        ledhsv[i].hue = (255/N_LEDS)*i;
        ledhsv[i].val = VALUE;
        ledhsv[i].sat = SATURATION;
    }

    ws2812_init();
}

void loop()
{
    uint8_t i;

    /* Fade to a new colour */
    for (i=0; i<N_LEDS; ++i)
    {
        hsv2rgb(ledhsv[i].hue, ledhsv[i].sat, ledhsv[i].val,
                &(ledrgb[i].r), &(ledrgb[i].g), &(ledrgb[i].b));
        ledhsv[i].hue++;
```

```
    }

    /* Write the new values to the LEDs */
    for (i=0; i<N_LEDS; ++i)
    {
        ws2812_write(ledrgb[i].r, ledrgb[i].g, ledrgb[i].b);
    }
    delayus(50); /* Have to wait this much for the WS2812 to latch their colour */

    delayms(25);
}
```

Table 6: Code Listing for the Main Application

4 Conclusion

The FT90X devices make for an ideal controller in a home automation or advertising application where the control of multiple LED lighting is required.

The GPIO interface allows for a relatively simple coding exercise to realize full colour, and brightness control of the LEDs.

5 Contact Information

Head Office – Glasgow, UK

Future Technology Devices International Limited
Unit 1, 2 Seaward Place, Centurion Business Park
Glasgow G41 1HH
United Kingdom
Tel: +44 (0) 141 429 2777
Fax: +44 (0) 141 429 2758

E-mail (Sales) sales1@ftdichip.com
E-mail (Support) support1@ftdichip.com
E-mail (General Enquiries) admin1@ftdichip.com

Branch Office – Tigard, Oregon, USA

Future Technology Devices International Limited (USA)
7130 SW Fir Loop
Tigard, OR 97223-8160
USA
Tel: +1 (503) 547 0988
Fax: +1 (503) 547 0987

E-Mail (Sales) us.sales@ftdichip.com
E-Mail (Support) us.support@ftdichip.com
E-Mail (General Enquiries) us.admin@ftdichip.com

Branch Office – Taipei, Taiwan

Future Technology Devices International Limited (Taiwan)
2F, No. 516, Sec. 1, NeiHu Road
Taipei 114
Taiwan , R.O.C.
Tel: +886 (0) 2 8791 3570
Fax: +886 (0) 2 8791 3576

E-mail (Sales) tw.sales1@ftdichip.com
E-mail (Support) tw.support1@ftdichip.com
E-mail (General Enquiries) tw.admin1@ftdichip.com

Branch Office – Shanghai, China

Future Technology Devices International Limited (China)
Room 1103, No. 666 West Huaihai Road,
Shanghai, 200052
China
Tel: +86 21 62351596
Fax: +86 21 62351595

E-mail (Sales) cn.sales@ftdichip.com
E-mail (Support) cn.support@ftdichip.com
E-mail (General Enquiries) cn.admin@ftdichip.com

Web Site

http://ftdichip.com

Distributor and Sales Representatives

Please visit the Sales Network page of the FTDI Web site for the contact details of our distributor(s) and sales representative(s) in your country.

Appendix A – References

Document References

FT900/901/902/903 Datasheet

FT905/906/907/908 Datasheet

FT900 User Manual

World Semi WS2812 Intelligent control LED integrated light source Datasheet

Acronyms and Abbreviations

Terms	Description
GPIO	General Purpose I/O
HSV	Hue Saturation Value
LED	Light Emitting Diode
RGB	Red Green Blue