NeoPixelLEDの制御

・LEDデータのイメージ図

赤	1	1	1	1	1	1	1	1
緑	0	0	0	0	0	0	0	0
青	0	0	0	0	0	0	0	0

1つのLEDが上記のような24ビットのデータを受け取ります。
上記の例であればLEDは赤に光ります。

この点においてはWebページなどによく用いられているカラーコードとまったく同じです。
上記の図をカラーコードにすれば単純に0xFF0000になります。

・データの送信
LED1つに対し、前述の通り24ビットのデータを送ります。
1ビットの0/1判定は電圧のハイ・ロー時間によって決定されます。

1ビット当たりの判定時間…1.25us(1250ns)
　　ビット1判定…電圧High時間が600ns
　　ビット0判定…電圧High時間が300ns

上記のデータ送信を実現する為として、**PWM**または**SPI**の2通りの制御が考えられます。
　　PWMの場合
　　1.25us周期のパルスを出力する。
　　1.25usは800KHz。この周波数からさらにデューティー比を決定する。

　　私が以前作成したプログラムでは
　　システムクロックを24MHzとし、タイマーのARR(自動再リロード)を29(30-1)に
　　設定。
　　High信号のデューティー比は 30 * 66% ≒ 20
　　Low信号のデューティー比は 30 * 33% ≒ 10
　　と設定していました。

PWMのメリットは使用するピンが1つのみである点とより直感的にパルスを設定できる点がメリットです。

　　SPIの場合
　　SPIクロックを8MHz(周期0.125us)とする。
　　High信号の場合 データは0b1111 0000
　　この場合High時間は0.125*4 = 0.5us

　　Low信号の場合 データは0b1000 0000
　　この場合High時間は 0.12581 = 0.125us

さきほど述べた判定時間とだいぶ違うのですが、私が使用しているLEDテープではこの値でないと正しく発色しない状態でした~~(なんかモヤモヤしますが)~~

SPIはデータ線に加え、使いもしないクロック線まで必要とする点がデメリットです。
しかしパルス幅の調整はPWMより容易です。

またAdafruit社からSPIのクロック線も用いた以下のような製品が発売されています。
https://www.switch-science.com/products/3239
クロック線を用いることでより円滑なデータ送信が可能になっているようです。
私も将来このDotStarを使う機会を想定して、SPIで書き直しました。
~~高いから買ってないけど~~

・DMAの利用
NeoPixelのデータ制御には膨大なメモリを必要とします。
LED数 * 24バイトですので、単純なSPI送信でもかなり手間がかかります。
そこでDMAというマイコンの機能にすべて丸投げしています。

どこかで見た記述ですが、DMAは**巨大なmemcpy()**です。
送信バッファをDMAに紐づけることでバッファからペリフェラル(今回であればSPIのデータレジスタ)へのコピーを逐次繰り返し、すべてのデータ送信まで面倒を見てくれます。
さらにHALライブラリであれば関数1つ呼ぶだけで済むのも楽です。

・プログラム全体の流れ
●InitNeoPixel
Hueの作成、Hue値をRGBに変換、RGB値をNeoPixelビットデータへ変換を行います。
今回はプログラム開始時しか呼んでいません。
もし外部スイッチ入力を追加したりして**色や明るさを変えたい場合**はこの関数を必ず呼ぶようにしてください。

●BuildHueData
Hue(色相)を作成します。
HSV色空間と呼ばれるものの内Hueのみを使用しています。
色相は0~360までの値を取り、RGB値を360段階で表現できます。
これによってRGB値を連続的に変化させ、虹色も表現できるようになります。

●HUEtoRGB
小細工の跡が残っていて少しややこしくなっていますが、端的に言えば
Hueが1増える = RGB値は(255/60)ずつ増減する
以上の関係にあります。
しかし実数型の変数を使っていない関係で(255/60)=4.25を保持できないため、このような処理をしています。

●SetLux
RGB値をLuxビットシフトすることでより小さい値に(暗く)します。

●MakeDMABuffer
LED数回RGB24ビットの各ビットをチェックします。

初期値1 << 23のmask変数でRGB24ビットをマスク、0でないならDMA送信バッファにHigh信号用のデータを格納します。
maskは右1ビットシフトを続けることでmaskが0になるまで解析します。

InitNeoPixel実行後、ShiftPixel関数を実行してDMAバッファを24バイト(1LED分)ずつバッファの内容をずらして行くことでLEDが流れるように色を変えていきます。
DMAバッファの値をHAL_SPI_Transmit_DMA()に渡し、最後にDMAが転送完了を通知する割り込み関数HAL_SPI_TxCpltCallback()を待ち、次のデータを送信します。