フルカラーシリアルLEDテープ(1m)を GR-KURUMIで使ってみる

2014/2/25

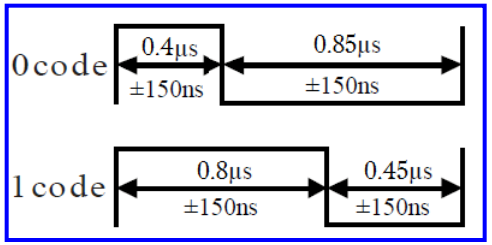
がじぇっとるねさす 鈴木

Rev. 1.00

フルカラーシリアルLEDの特徴

http://www.switch-science.com/catalog/1399/

・3570円
・1mで60個のLEDがついている
・電源と信号線1本で制御する



x 24

この信号を24個送信して1個のLED
を点灯させる。60個を点灯するには
24x60＝1440**個データ送信が必要**

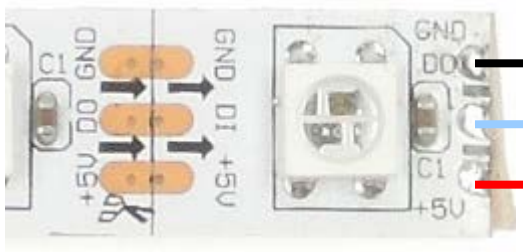
※現在販売されているLEDのタイミング

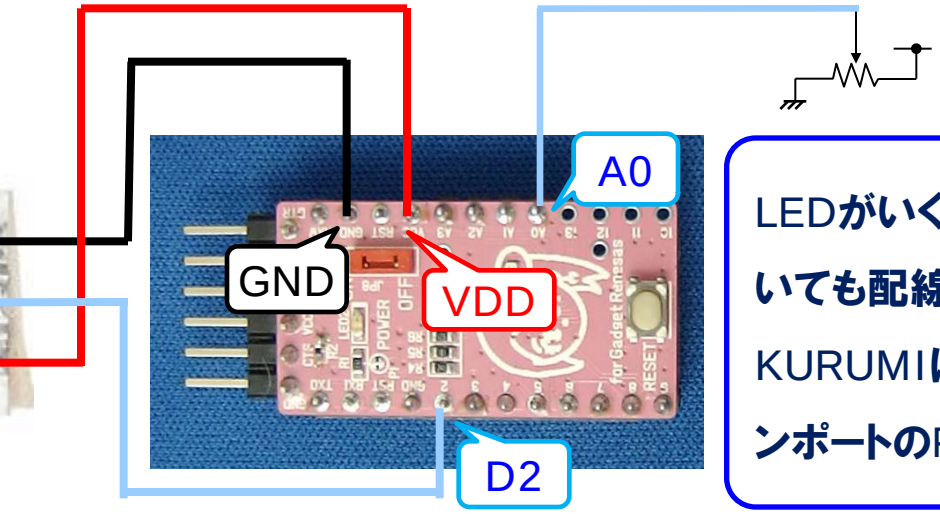
このLEDを制御するための信号、0.4usは400ナノ秒で相当に早い。400ナノ秒は0.00000004秒なので、1秒間に30万Km進む光でも、0.00000004秒は120mしか進めない。

配線図、動作概要

配線図

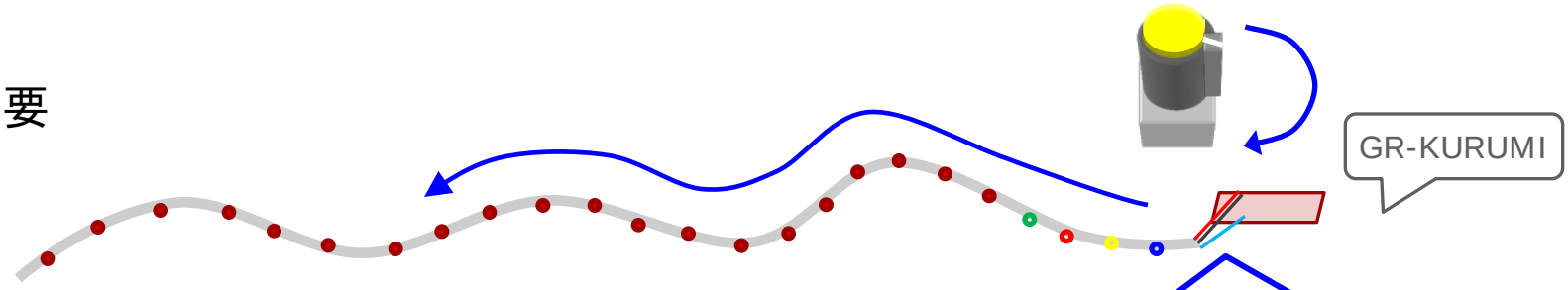
フルカラーシリアルLED





LEDがいくつも接続されていても配線はこれだけ。GR-KURUMI**は端子**D2 (**マイコンポートの**P30),A0**を使用**

動作概要



半固定抵抗をまわすとそれに応じてLEDの色が変化する。
最初のLEDの色を次のLEDへコピーしていくように流れる
表示を行うプログラム

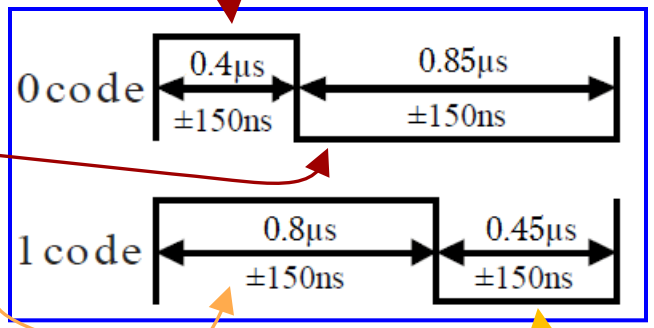
プログラム

```
#include <RLduino78.h>
#define DI()      asm("di")
#define EI()      asm("ei")

// Pin 22,23,24 has an LED connected on most Arduino boards
#define LED_R   22
#define LED_G   23
#define LED_B   24

// 端子         マイコンリソース
// D2           P3 テープLEDの制御  0xFFF03 アドレス
#define TAPELED 2
#define VOLUME  A0

//  テープLEDの信号を制御
#define TAPELED_ON      (*(volatile unsigned char *)0xFFF03) = 1
#define TAPELED_OFF     (*(volatile unsigned char *)0xFFF03) = 0

//     テープLEDの時間待ち処理
#define TAPELED_ON_WAIT1   dummy2++; dummy++
#define TAPELED_ON_WAIT0   for ( dummy = 0; dummy<0; dummy++ ) { ; }; dummy1++
#define TAPELED_OFF_WAIT1  for ( dummy = 0; dummy<0; dummy++ ) { ; }; dummy1++
#define TAPELED_OFF_WAIT0  for ( dummy = 0; dummy<1; dummy++ ) { ; }; dummy1++
#define TAPELED_WAIT       for ( dummy2 = 0; dummy2<100; dummy2++ ) { ; }
#define LEDMAX             60          // テープLEDに接続されているLEDの数
#define LEDDATAMAX  160         // テープLEDのカラーデータ数

//  テープLEDにデータを送る関数
void SetLed( void );

int gAd0value;      // A0の値を保存する変数

//  テープLED LEDMAX分のバッファ
uint8_t  gtLedData[ LEDMAX ][ 3 ] =
{
    { 0x00,   0x00,   0x00 } ,
    { 0x00,   0x00,   0x00 } ,
    { 0x00,   0x00,   0x00 } ,
    省略
}
```

LEDにデータを転送する関数

60個分LEDの色データ(R, G, B)を領域確保

下記の時間待ちを確保するために
ダミーで簡単な演算を行っている



プログラム

```
// テープLED 色を変化させるためのデータテーブル 160色分
uint8_t  gtLedColorData[ LEDDATAMAX ][ 3 ] =
{   //  G       R       B
    { 0x00,   0x10,   0x00 } ,     //  1
    { 0x00,   0x20,   0x00 } ,
    { 0x00,   0x30,   0x00 } ,
    { 0x00,   0x40,   0x00 } ,
    { 0x00,   0x50,   0x00 } ,
    { 0x00,   0x60,   0x00 } ,
    { 0x00,   0x70,   0x00 } ,
    { 0x08,   0x80,   0x00 } ,
    { 0x00,   0x90,   0x00 } ,
    { 0x08,   0xa0,   0x00 } ,
    { 0x00,   0xb0,   0x00 } ,
    { 0x08,   0xc0,   0x00 } ,
    { 0x00,   0xd0,   0x00 } ,
    { 0x08,   0xe0,   0x00 } ,
    { 0x00,   0x00,   0x10 } ,
    { 0x00,   0x00,   0x20 } ,
    省略
    { 0x00,   0x00,   0x00 }
};

// the setup routine runs once when you press reset:
void setup()
{
    pinMode( LED_R, OUTPUT );
    pinMode( LED_G, OUTPUT );
    pinMode( LED_B, OUTPUT );
    pinMode( TAPELED, OUTPUT );
}
```

LEDの色データ(R, G, B)を計算でなく、データとして登録しておく。const宣言でもよいのだが、参照速度が遅くなるのでRAMに配置。

実際に出力するのは TAPELED(D2)端子

```
// the loop routine runs over and over again forever:
void loop()
{
    uint8_t datg, datr, datb;
    uint16_t datadrs;

    // ポテンションメータの直値をgAdvalへ代入
    gAd0value = analogRead( VOLUME ) / 6;
    datadrs = gAd0value;
    if ( datadrs  > ( LEDDATAMAX - 1 ) )
    {
        datadrs = LEDDATAMAX - 1;
    }

    datg = gtLedColorData[ datadrs ][ 0 ];
    gtLedData[ 0 ][ 0 ] = datg;

    datr = gtLedColorData[ datadrs ][ 1 ];
    gtLedData[ 0 ][ 1 ] = datr;

    datb = gtLedColorData[ datadrs ][ 2 ];
    gtLedData[ 0 ][ 2 ] = datb;

    SetLed();

    delay( 10 );
}
```

ボリュームの値によってLEDの色データを決める。160 色までになるように調整する。

LED色データ(G, R, B)をLEDデータバッファ先頭にのみ代入。

LEDデータバッファから実際のLED60個にデータを転送する。LEDデータバッファをひとつづつずらし、流れるような表示にする

プログラム

```
//  テープLEDにデータを送る
void SetLed( void )
{
    uint8_t datg, datr, datb, ledloop;
    volatile uint8_t dummy;
    volatile uint16_t dummy1;
    volatile uint32_t dummy2;

    // 割り込みを禁止して、データ送信がずれないようにする
    DI();

    for ( ledloop = 0; ledloop < LEDMAX; ledloop++ )
    {
        datg = gtLedData[ ledloop ][ 0 ];
        datr = gtLedData[ ledloop ][ 1 ];
        datb = gtLedData[ ledloop ][ 2 ];

        // 緑LED
        if ( datg & 0x80 )
        {   // on data send
            TAPELED_ON;
            TAPELED_ON_WAIT1;
            TAPELED_OFF;
            TAPELED_OFF_WAIT1;
        }
        else
        {   // off data send
            TAPELED_ON;
            TAPELED_ON_WAIT0;
            TAPELED_OFF;
            TAPELED_OFF_WAIT0;
        }

        if ( datg & 0x40 )
        {   // on data send
            TAPELED_ON;
            TAPELED_ON_WAIT1;
            TAPELED_OFF;
            TAPELED_OFF_WAIT1;
        }
        else
        {   // off data send
            TAPELED_ON;
            TAPELED_ON_WAIT0;
            TAPELED_OFF;
            TAPELED_OFF_WAIT0;
        }

        if ( datg & 0x20 )
        {   // on data send
            TAPELED_ON;
            TAPELED_ON_WAIT1;
            TAPELED_OFF;
            TAPELED_OFF_WAIT1;
        }
        else
        {   // off data send
            TAPELED_ON;
            TAPELED_ON_WAIT0;
            TAPELED_OFF;
            TAPELED_OFF_WAIT0;
        }

        if ( datg & 0x10 )
        {   // on data send
            TAPELED_ON;
            TAPELED_ON_WAIT1;
            TAPELED_OFF;
            TAPELED_OFF_WAIT1;
        }
        else
        {   // off data send
            TAPELED_ON;
            TAPELED_ON_WAIT0;
            TAPELED_OFF;
            TAPELED_OFF_WAIT0;
        }

        if ( datg & 0x08 )
        {   // on data send
            TAPELED_ON;
            TAPELED_ON_WAIT1;
            TAPELED_OFF;
            TAPELED_OFF_WAIT1;
        }
        else
        {   // off data send
            TAPELED_ON;
            TAPELED_ON_WAIT0;
            TAPELED_OFF;
            TAPELED_OFF_WAIT0;
        }

        if ( datg & 0x04 )
        {   // on data send
            TAPELED_ON;
            TAPELED_ON_WAIT1;
            TAPELED_OFF;
            TAPELED_OFF_WAIT1;
        }
        else
        {   // off data send
            TAPELED_ON;
            TAPELED_ON_WAIT0;
            TAPELED_OFF;
            TAPELED_OFF_WAIT0;
        }

        if ( datg & 0x02 )
        {   // on data send
            TAPELED_ON;
            TAPELED_ON_WAIT1;
            TAPELED_OFF;
            TAPELED_OFF_WAIT1;
        }
```

緑LEDの7bit目の処理

緑LEDの6bit目の処理

緑LEDの5bit目の処理

緑LEDの4bit目の処理

緑LEDの3bit目の処理

緑LEDの2bit目の処理

緑LEDの1bit目の処理

プログラム

```
    else
    {   // off data send
        TAPELED_ON;
        TAPELED_ON_WAIT0;
        TAPELED_OFF;
        TAPELED_OFF_WAIT0;
    }

    if ( datg & 0x01 )
    {   // on data send
        TAPELED_ON;
        TAPELED_ON_WAIT1;
        TAPELED_OFF;
        TAPELED_OFF_WAIT1;
    }
    else
    {   // off data send
        TAPELED_ON;
        TAPELED_ON_WAIT0;
        TAPELED_OFF;
        TAPELED_OFF_WAIT0;
    }


    // 赤LED
    省略

    // 青LED
    省略
}
```

緑LEDの0bit目の処理

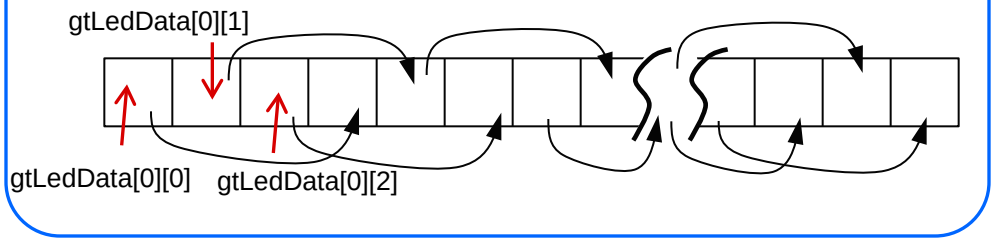
赤LEDの7～0bit目の処理

青LEDの7～0bit目の処理

ここまでの処理に for 文を使わないのは、データの転送を一定時間に保つため。1つのLEDにデータを送る為の処理、これを60回繰り返す。

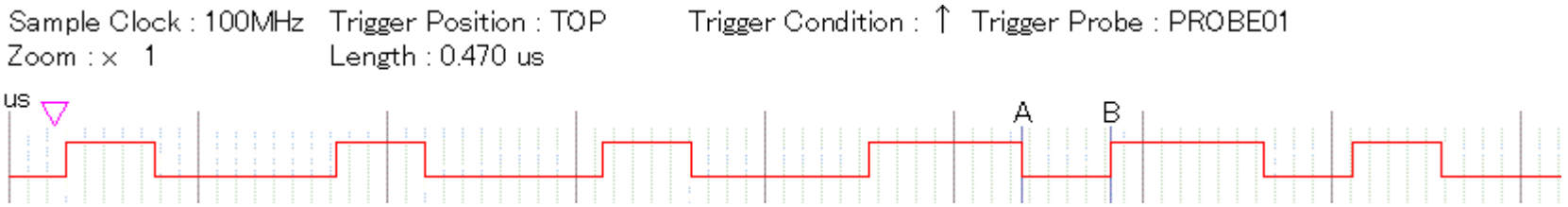
```
    // テープLEDのバッファをずらして、流れる表示にする
    for ( ledloop = LEDMAX - 1; ledloop > 0; ledloop-- )
    {
        datg = gtLedData[ ledloop - 1 ][ 0 ];
        gtLedData[ ledloop ][ 0 ] = datg;

        datr = gtLedData[ ledloop - 1 ][ 1 ];
        gtLedData[ ledloop ][ 1 ] = datr;

        datb = gtLedData[ ledloop - 1 ][ 2 ];
        gtLedData[ ledloop ][ 2 ] = datb;
    }

    EI();
}
```

LEDのデータバッファを流れる表示にするためにデータコピーする



実行結果

D2端子の波形



電源は5Vを使うこと

60個のLEDを全て点灯させるとおよそ1.75Aを消費する

半固定抵抗をグルグル回して色を変えているところ

以上です。

ご不明な点は質問をお願いします