NS-RX231 のソフトウェアガイド

〈赤外線リモコン信号の送信編〉

目次

1 プロジェクトのインポート

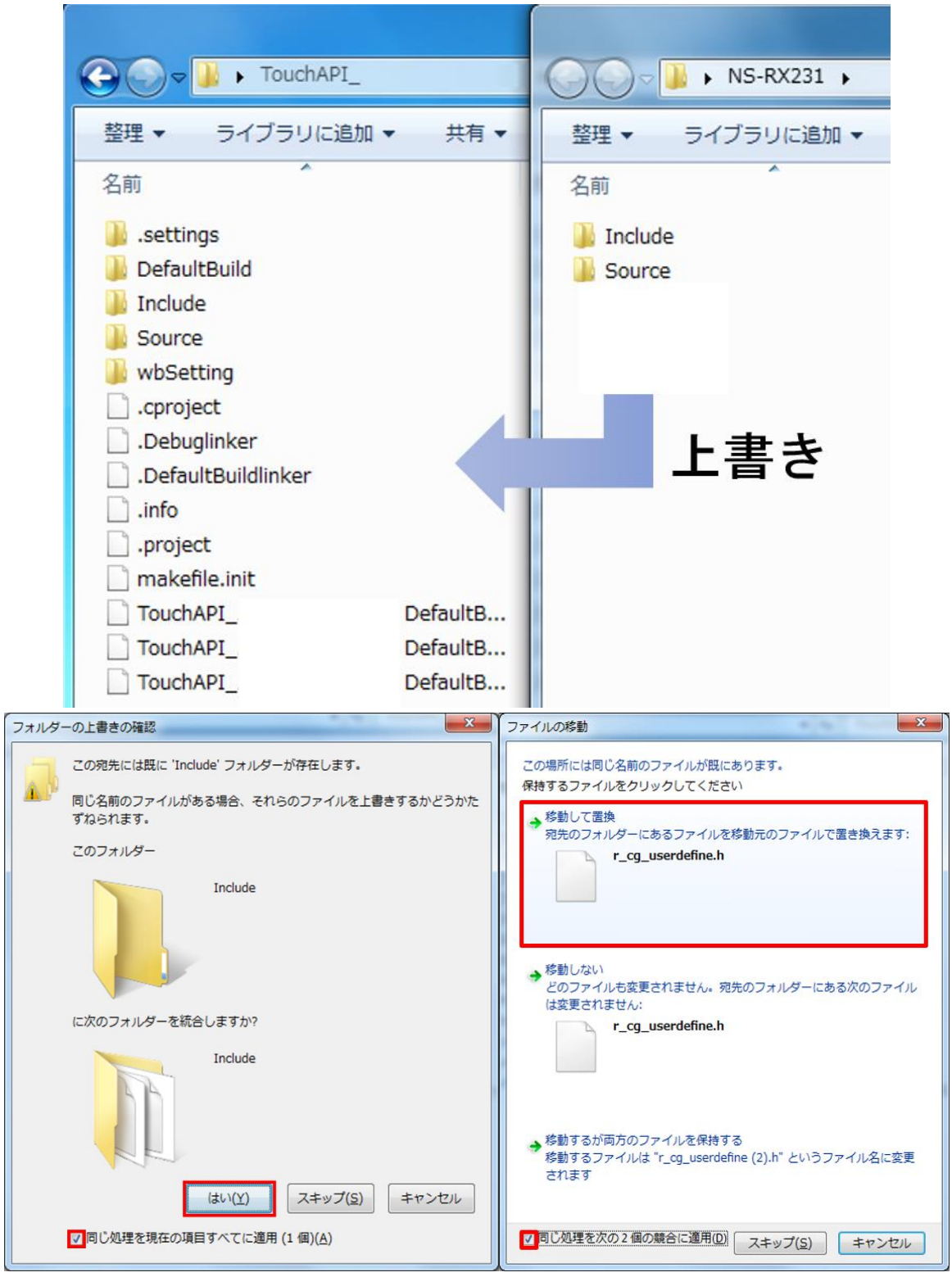


図 1-1 ソースファイルを上書き

添付されたソースファイルをWorkbench6 First step guideに従って，ウィザードで作成したプロジェクトに上書きした後、e2studioを実行します

2 概要

図 2-1 電磁波の波長による分類

赤外線とは、人が見ることができる波長の可視光線より波長の長い電磁波です。赤を表現する波長よりも外側に位置しているので，赤外線と呼ばれます。

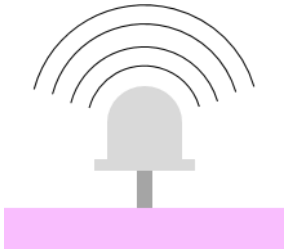


図 2-2 赤外線発光素子IR LED

IR LEDは一般のLEDのように電流が流れると発光しますが、IR LEDは可視光線ではなく，赤外線を出力するので人の目には見えない点が異なります。赤外線を利用して無線通信を実現している良い例が赤外線リモコンです。

赤外線通信は、人体への影響が少ない無線通信であり、比較的簡単に実装することができるので、小容量のデータ通信に多く使われています。

キャリア周波数	30 kHz	TSOP4130
	33 kHz	TSOP4133
	36 kHz	TSOP4136
	38 kHz	TSOP4138
	40 kHz	TSOP4140
	56 kHz	TSOP4156

図 2-3 キャリア周波数と受信デバイスの関係

NS-RX231に装着されている赤外線受信素子（TSOP4138）は、38kHzのキャリア周波数に対応する特性を持っているので、タイマーは38kHzを生成するために、約13.15usごとに割り込みを発生させ、波形を作成します。

3 ソースコード

3.1 信号フォーマット

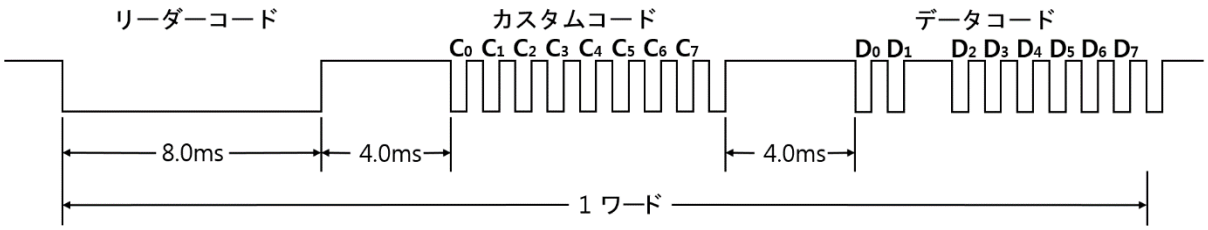


図 3-1リモコン信号フォーマット例

まず、簡単な信号フォーマットを例としてプロジェクトを作成してみました。

信号フォーマットは、通信の開始を知らせるリーダコード、リモコン機器の区分のためのカスタムコード、そしてデータを転送するためのデータのコードになっています。

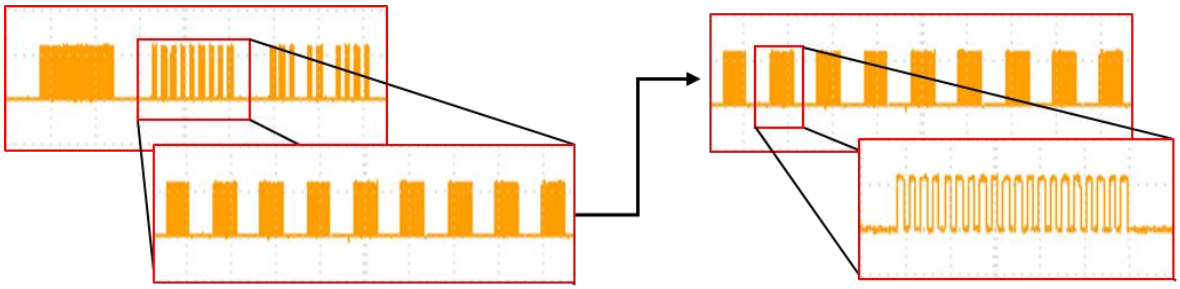


図 3-2 赤外線通信の送信時の波形

そして、ビットの0と1の違いは、次のとおりです。

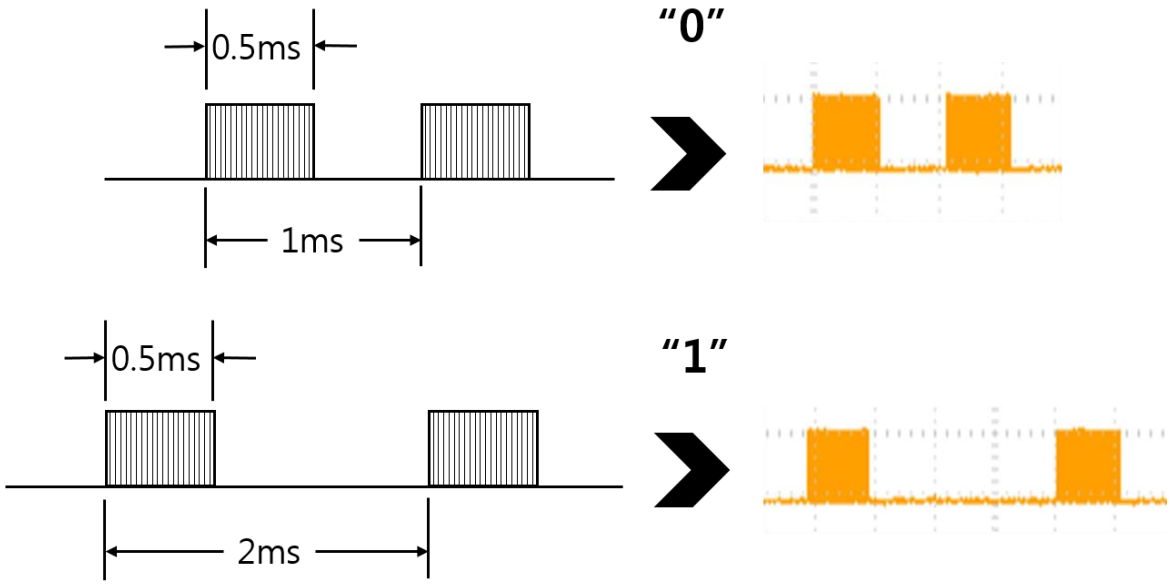


図 3-3 赤外線信号の0と1の違い

3.2 ソースコード

r_main.c

```
if (_0_SUCCESS == R_Set_Cap_Touch_Result_Create( method ))
{
    ts_result = R_Get_Cap_Touch_Result( method );
    if(1 == ts_result.button[1] >> 0)
    {
        PORTB.PODR.BYTE = 0x02;
        sendArr = IR_send(232,40);
    }
    else if(1 == ts_result.button[1] >> 6)
    {
        PORTB.PODR.BYTE = 0x08;
        sendArr = IR_send(232,24);
    }
    else if(1 == ts_result.button[1] >> 7)
    {
        PORTB.PODR.BYTE = 0x20;
        sendArr = IR_send(232,168);
    }
}
```

r_cg_tmr_user.c

```
if(sendArr[37] >= 1){
    sendArr[cnt]--;
    if(flg == 0){
        PORTA.PODR.BIT.B4 ^= 1;
    }

    if(sendArr[cnt] < 1){
        flg ^= 1;
        cnt++;
        if(cnt == 38){
            cnt = 0;
        }
    }
}
```

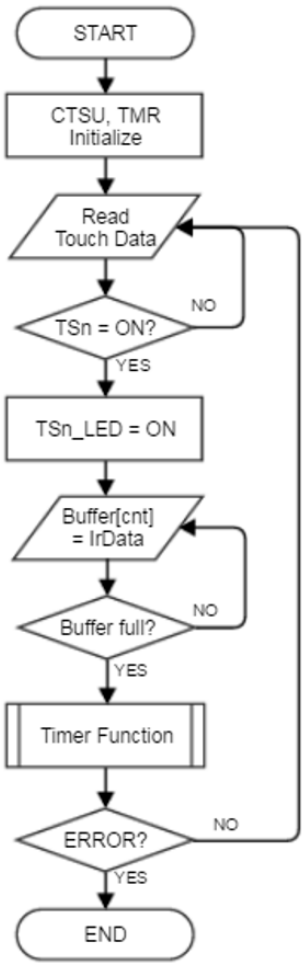


図 3-4 ソースコードとフローチャート

タイマー割り込みは、約13usごとに起動されて処理を行います。（38kHzの周期は26us）

sendArr配列の最後の要素が1以上の時に処理を行います。flgの値が0の場合には、ポートの出力を反転させて38kHzの波形を作成しますが、1の場合には、ポート出力しないで、配列の要素で指定された時間待ちます。
動作例としては次のようになります。8msの波形を作成するために、sendArr配列の0番目の要素を参照すると、616となっており割り込み処理を616回（13us*616=8008us）繰り返すような指定になっています。変数flagが0なので，タイマー割り込みごとにポートへの出力を反転させて配列の0番目の要素を1ずつ減算します。616回繰り返して0になると、flgの値を反転させ，配列の次の要素を参照するように変数cntの値を+1します。

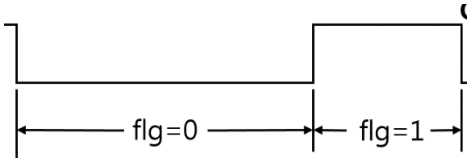
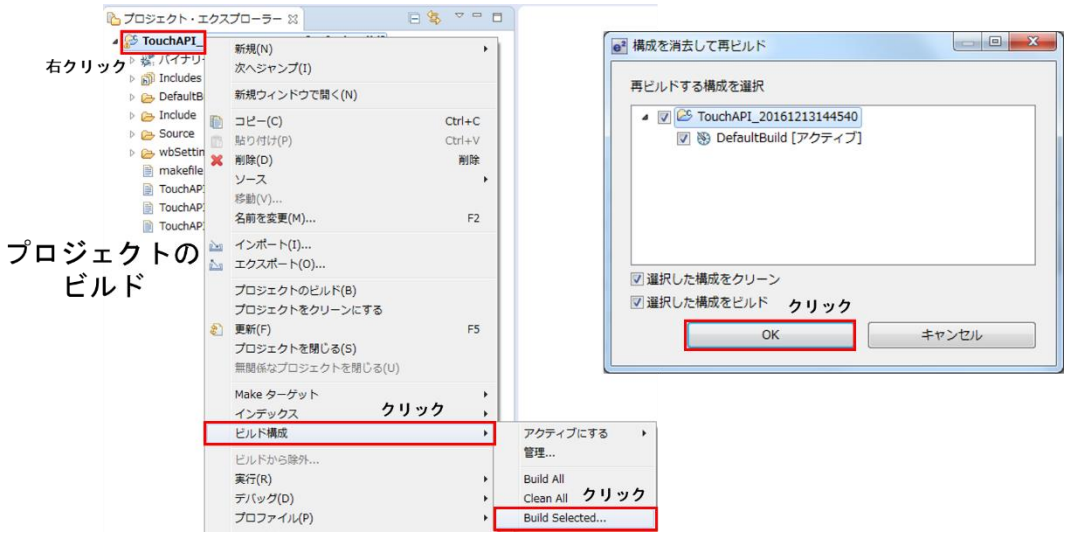


図 3-5 信号波形とflagの関係

4 デバッグ

図 4-1 NS-RX231を電源アダプタとE1デバッガを接続した様子



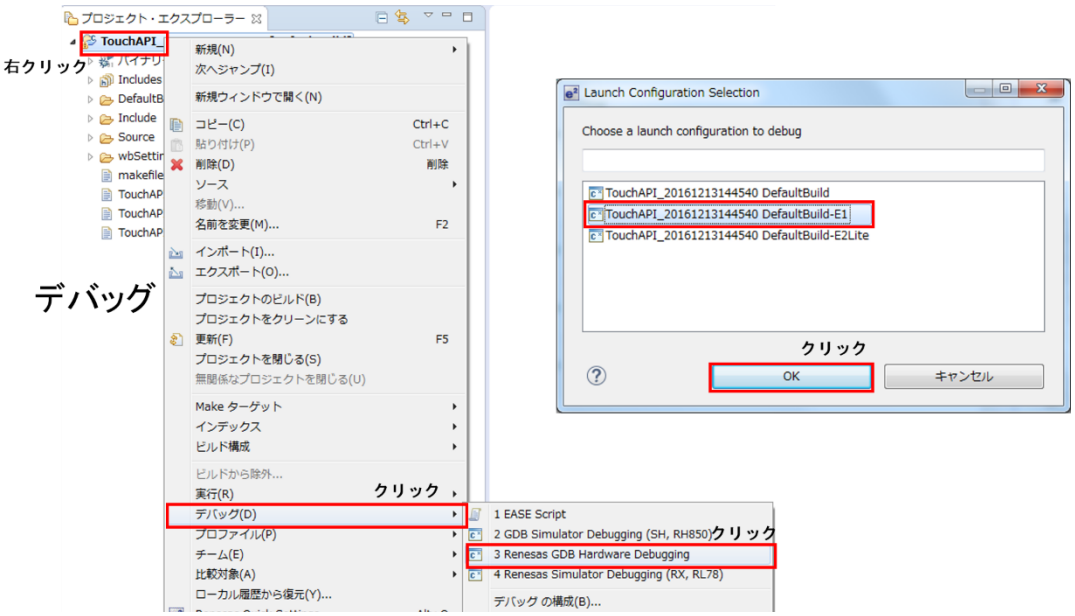


図 4-2 プロジェクトのビルドとデバッグ

5 実行

上記のプログラムをボードにアップロードした後、TS16にタッチしてみます。

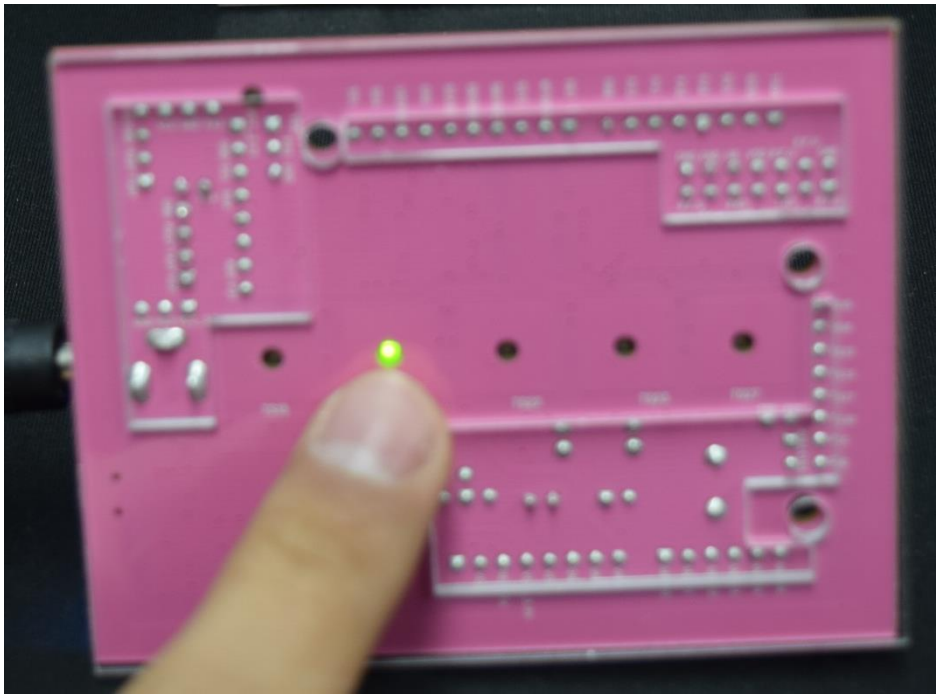


図 5-1 TS16へのタッチ

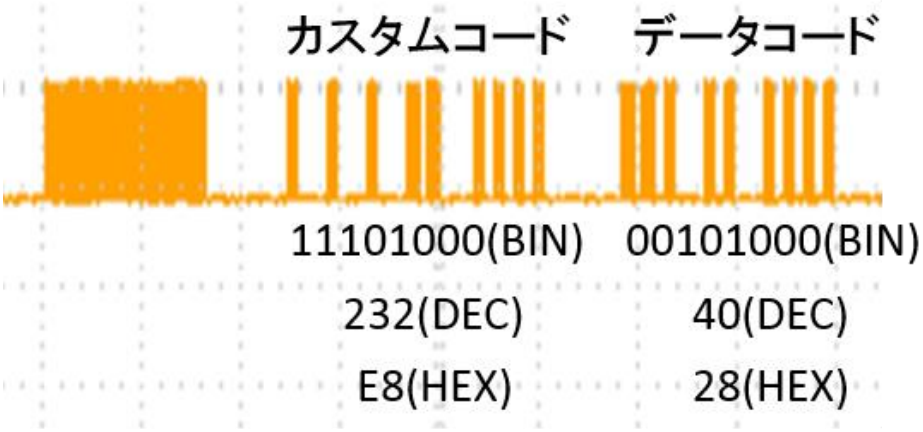


図 5-2 オシロスコープで観測された波形

オシロスコープでは、上記の写真のような波形が観測されました。

6 回路図

下の図は、赤外線リモコン送信部の回路図です。

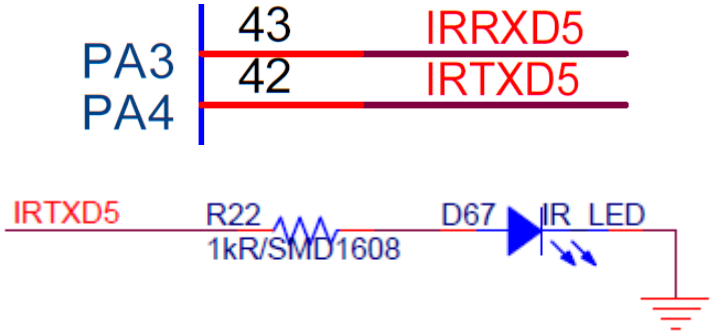


図 6-1 NS-RX231の赤外線送信部回路図

7 追加

もしリーダーコードの長さを変更したい場合や、データコード部でのエラーを減らすために、データコードを送信した後に反転されたデータコードを送信したい場合、自分が作った他のプロトコルにしたい場合は、r_detectIR.cソースファイルからIR_send関数部分を変更します。

returnArr[38]は、関数から返される配列であり、38は、配列のサイズを示します。

配列の内容は、次のとおりです。

リーダーコード 8msは returnArr[0] = 616
リーダーコード 4msは returnArr[1] = 308

カスタムコード C0は returnArr[2] = 38
カスタムコード C0は returnArr[3] = 1であれば116、0であれば38

…

カスタムコード C7は returnArr[17] = 1であれば116、0であれば38

終わりを示す短い波形 = 38
待機時間 4ms = 308

…

1と0を区別するためにビットは、短い波形（0.5ms）を出力するための最初の引数と1と0を区別するための（0.5ms or1.5ms）2番目の引数のペアで構成されており、合計8つのビットが1バイトを達成のために、16個の引数が必要です。そして、バイトの終わりを知らせる終了コードも2つの引数が一組となるので、バイトのデータコードを追加したい場合は、合計18個の配列要素が追加で必要になります。

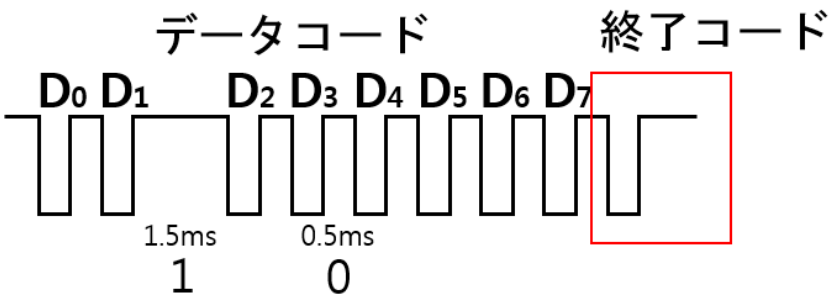


図 7-1 1バイトのデータコード例

※ 追加する配列要素の値は必ず偶数にしてください。