電子情報工学実験

赤外線リモコン受信

3I14 公文健太

# 実験の目的

赤外線リモコンは 38KHz の搬送波を用いた通信機器である。ここでは、マイコン“ATmega328p” を用いて赤外線リモコンを作成し、動作を学ぶ。

# 実験結果

## 実験1

下のオシロスコープから

データ部は10010000であることがわかった。

### オシロスコープ

|  |  |
| --- | --- |
| リーダ | カスタム |
|  |  |
| カスタム(反転) | データ |
|  |  |
| データ(反転) |
|  |

## 実験2

### コード

#include<asf.h>

void io\_init(void);

void start(void);

void restart(void);

void timer0\_ctcmode\_init(uint8\_t top);

//リーダコード許容範囲（カウンタ値）プリスケーラ設定 1024 分周

#define LEADER\_LMIN (uint8\_t) 158 //9ms x 90%

#define LEADER\_LMAX (uint8\_t) 193 //9ms x 110%

#define LEADER\_HMIN (uint8\_t) 79 //4.5ms x 90%

#define LEADER\_HMAX (uint8\_t) 97 //4.5ms x 110%

#define TIME\_OUT (uint8\_t) 195 //10ms

//ステート定義

enum DATA\_type {LEADER\_LOW, LEADER\_HIGH, START\_READING};

enum DATA\_type State, NEXT\_State;

uint8\_t intv = 0; //パルス幅

int main (void)

{

    io\_init(); //I/O ポート初期設定

    start(); //受信スタート

    sei();

    while(1);

    return 0;

}

void io\_init(void) //I/O ポート設定

{

    DDRB = 0b00010000; //LED(PB5)を出力に設定

    PORTB = 0b00000000; //LED 消灯

    DDRD = 0b00000000; //外部割り込み（INT1）を入力に設定

    return;

}

void start(void) //受信スタート

{

    NEXT\_State = LEADER\_LOW;

    EICRA = 0b00001000; //INT1 立ち下がり割り込み

    EIMSK = 0b00000010; //INT1 割り込み有効

    return;

}

ISR(INT1\_vect) //INT1 割り込みサブルーチン

{

    intv = TCNT0; //カウンタ読み出し

    TCNT0 = 0; //カウンタリセット

    State = NEXT\_State; //ステート更新

    switch (State) {

        case LEADER\_LOW :

            timer0\_ctcmode\_init(TIME\_OUT); //タイムアウトセット

            EICRA = 0b00001100; //立ち上がり割り込みをセット

            NEXT\_State = LEADER\_HIGH; //次の状態へ

            break;

        case LEADER\_HIGH :

            if (LEADER\_LMIN <= intv && LEADER\_LMAX >= intv) {//L レベルが範囲内なら

                NEXT\_State = START\_READING; //次の状態へ

                EICRA = 0b00001000; //立ち下がり割り込みをセット

            }

            else //範囲外ならリスタート

            restart();

            break;

            case START\_READING :

            if (LEADER\_HMIN <= intv && LEADER\_HMAX >= intv) {//H レベルが範囲内なら

                EIMSK = 0b00000000; //割り込み停止

                TIMSK0 = 0b000000000; //タイマ０割込み停止

                PORTB ^= 0b00100000; //受信処理（LED 反転）

                start(); //受信開始

            }

            else //範囲外ならリスタート

            restart();

            break;

    }

    return;

}

ISR(TIMER0\_COMPA\_vect) //タイムアウト

{

    restart();

    return;

}

void restart(void) //リスタート

{

    start(); //受信開始

    return;

}

// タイマ／カウンタ０ CTC モード

// top : カウンタ最大値

void timer0\_ctcmode\_init(uint8\_t top)

{

    OCR0A = top; //タイマ／カウンタ 0 最大値

    TCCR0A = 0b00000010; //CTC モード

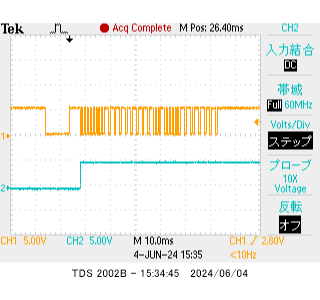
    TCCR0B = 0b00000101; //CTC モード、プリスケーラ設定 1024 分周

    TIMSK0 = 0b00000010; //コンペアマッチ A 割り込み有効

    return;

}

### オシロスコープ



上よりリーダコードを読み終わった後LEDが点灯していることがわかる。

## 実験3

### コード

//赤外線 NEC フォーマット 受信プログラム

#include <asf.h>

void start(void);

void restart(void);

void timer0\_ctcmode\_init(uint8\_t top);

void timer1\_ctcmode\_init(uint16\_t top);

void io\_init(void);

int8\_t receive\_bit(uint8\_t intv);

#define LEADER\_LMIN (uint8\_t) 158 //9ms x 90%

#define LEADER\_LMAX (uint8\_t) 193 //9ms x 110%

#define LEADER\_HMIN (uint8\_t) 79 //4.5ms x 90%

#define LEADER\_HMAX (uint8\_t) 97 //4.5ms x 110%

#define D1\_TMIN (uint8\_t) 40 //2.25ms x 90%

#define D1\_TMAX (uint8\_t) 48 //2.25ms x 110%

#define D0\_TMIN (uint8\_t) 20 //1.125ms x 90%

#define D0\_TMAX (uint8\_t) 24 //1.125ms x 110%

#define TIME\_OUT (uint8\_t) 195 //10ms

#define CUSTOM1\_CODE (uint8\_t)0x00

#define CUSTOM2\_CODE (uint8\_t)0xFF

#define OFF\_CODE 0x01 //OFF ボタン

#define LIGHT\_CODE 0x09 //ON(明)ボタン

#define DARK\_CODE 0x11 //ON(暗)ボタン

enum DATA\_type {LEADER\_LOW, LEADER\_HIGH, START\_READING, CUSTOM1, CUSTOM2, DATA1, DATA2, END, Error};

enum DATA\_type State, NEXT\_State;

uint8\_t bit\_pos = 0, recv\_custom1\_code = 0, recv\_custom2\_code = 0, recv\_data1\_code = 0, recv\_data2\_code = 0;

int8\_t bit;

int main (void)

{

    io\_init();

    start();

    sei();

    while(1);

    return 0;

}

void io\_init(void) //I/O ポート設定

{

    DDRB = 0b00010000; //LED(PB5)を出力に設定

    PORTB = 0b00000000; //LED 消灯

    DDRD = 0b00000000; //外部割り込み（INT1）を入力に設定

    return;

}

void start() //受信スタート

{

    NEXT\_State = LEADER\_LOW;

    EICRA = 0b00001000; //INT1 立ち下がり割り込み

    EIMSK = 0b00000010; //INT1 割り込み有効

    return;

}

ISR(INT1\_vect)

{

    uint8\_t intv = 0;

    intv = TCNT0; //カウンタ読み出し

    TCNT0 = 0; //カウンタリセット

    State = NEXT\_State; //ステート更新

    switch (State) {

        case LEADER\_LOW : //立ち下がりエッジ

            timer0\_ctcmode\_init(TIME\_OUT); //タイムアウトセット

            EICRA = 0b00001100; //立ち上がり割り込みをセット

            NEXT\_State = LEADER\_HIGH;

            break;

        case LEADER\_HIGH : //立ち上がりエッジ

            if (LEADER\_LMIN <= intv && intv <= LEADER\_LMAX) { //L レベルが範囲内なら

                NEXT\_State = START\_READING; //次は START\_READING

                EICRA = 0b00001000; //立ち下がり割り込みをセット

            }

            else //範囲外ならリスタート

                restart();

            break;

        case START\_READING :

            if (LEADER\_HMIN <= intv && intv <= LEADER\_HMAX) { //H レベルが範囲内なら

                NEXT\_State = CUSTOM1; //次はカスタム１

                recv\_custom1\_code = 0; //初期設定

                bit\_pos = 0b00000001; //最下位ビットから読み込み

            }

            else //範囲外ならリスタート

                restart();

            break;

        case CUSTOM1 : //立ち下がりエッジ

            bit = receive\_bit(intv); //１ビット受信

            if (bit < 0) { //受信エラーならリスタート

                restart();

            }

            else {

                if(bit == 1) {

                    recv\_custom1\_code |= bit\_pos; //カスタムコードにセット

                }

                bit\_pos <<= 1; //ビット位置をずらす

                if (bit\_pos == 0) { //8bit 受信したら

                    NEXT\_State = CUSTOM2; //次はカスタム２

                    recv\_custom2\_code = 0; //初期設定

                    bit\_pos = 0b00000001; //最下位ビットから読み込み

                }

            }

            break;

        case CUSTOM2 : //立ち下がりエッジ

            bit = receive\_bit(intv); //１ビット受信

            if (bit < 0) //受信エラーならリスタート

                restart();

            else {

                if(bit == 1) {

                    recv\_custom2\_code |= bit\_pos; //カスタム(反転)コードにセット

                }

                bit\_pos <<= 1; //ビット位置をずらす

                if (bit\_pos == 0) { //8bit 受信したら

                    if (recv\_custom1\_code != ((~recv\_custom2\_code) & 0xFF)) {

                        restart(); //データが一致しないならリスタート

                    }

                    else {

                        NEXT\_State = DATA1; //次の状態、初期設定

                        recv\_data1\_code = 0; //初期設定

                        bit\_pos = 0b00000001; //最下位ビットから読み込み

                    }

                }

            }

            break;

        case DATA1 : //立ち下がりエッジ

            bit = receive\_bit(intv); //１ビット受信

            if (bit < 0) { //受信エラーならリスタート

                restart();

            }

            else {

                if(bit == 1) {

                    recv\_data1\_code |= bit\_pos; //カスタムコードにセット

                }

                bit\_pos <<= 1; //ビット位置をずらす

                if (bit\_pos == 0) { //8bit 受信したら

                    NEXT\_State = DATA2; //次はカスタム２

                    recv\_data2\_code = 0; //初期設定

                    bit\_pos = 0b00000001; //最下位ビットから読み込み

                }

            }

            break;

        case DATA2 : //立ち下がりエッジ

            bit = receive\_bit(intv); //１ビット受信

            if (bit < 0) //受信エラーならリスタート

                restart();

            else {

                if(bit == 1) {

                    recv\_data2\_code |= bit\_pos; //カスタム(反転)コードにセット

                }

                bit\_pos <<= 1; //ビット位置をずらす

                if (bit\_pos == 0) { //8bit 受信したら

                    if (recv\_data1\_code != ((~recv\_data2\_code) & 0xFF)) {

                        restart(); //データが一致しないならリスタート

                    }

                    else { //8 ビット目で受信データが正しいなら

                        State = END;

                        NEXT\_State = Error;

                    }

                }

            }

            break;

        case END:

            break;

        case Error : //33bit 目エラー

            restart();

            break;

    }

}

ISR(TIMER0\_COMPA\_vect) //タイムアウト検出

{

    if (State == END) { //受信完了

        TIMSK0 = 0b00000000; //タイマ０割込停止

        if (recv\_data1\_code == OFF\_CODE) { //OFF スイッチ

            TIMSK1 = 0x00; //タイマ１割込停止、LED 消灯

            PORTB=0x00;

        }

        else if (recv\_data1\_code == LIGHT\_CODE) //オン（明）スイッチ

            timer1\_ctcmode\_init(9766); //約 0.5 秒間隔点滅

        else if (recv\_data1\_code == DARK\_CODE) //オン（暗）スイッチ

            timer1\_ctcmode\_init(19531); //約１秒間隔点滅

        EIMSK = 0b00000010; //INT1 割り込み

    }

    else //タイムアウトエラー

        restart();

}

ISR(TIMER1\_COMPA\_vect) //受信処理

{

    PORTB ^= 0b00100000; //LED 反転(PB5)

}

int8\_t receive\_bit(uint8\_t intv) //データ受信、チェック

{

    if (D0\_TMIN <= intv && intv <= D0\_TMAX)

        return 0; //データ０受信

    else if (D1\_TMIN <= intv && intv <= D1\_TMAX)

        return 1; //データ１受信

    else

        return -1; //エラー

}

void restart(void) //リスタート

{

    start(); //受信開始

    return;

}

void timer0\_ctcmode\_init(uint8\_t top)

{

    OCR0A = top; //タイマ／カウンタ 0 最大値

    TCCR0A = 0b00000010; //CTC モード

    TCCR0B = 0b00000101; //CS02 CS01 CS00 プリスケーラ

    TIMSK0 = 0b00000010; //コンペアマッチ A 割り込み有効

}

void timer1\_ctcmode\_init(uint16\_t top)

{

    OCR1A = top; //タイマ／カウンタ 1 最大値

    TCCR1A = 0b00000000; //CTC モード

    TCCR1B = 0b00001101; //CS12:CS11:CS10 プリスケーラ

    TIMSK1 = 0b00000010; //コンペアマッチ A 割り込み有効

}

### オシロスコープ

|  |  |
| --- | --- |
| 10 ms | 20 ms |
|  |  |

上より、それぞれ10msと20msの周期でLEDが点滅していることがわかる。

# 感想

赤外線リモコンの信号を入力してオシロスコープで表示して、波形を観察することができた。また、それを用いて信号を分析しコードに応じた処理を実行することができた。カスタム部やデータ部などのコードの種類に応じて受け取る部分をそれぞれ作ることが難しかった。しかし、普段使うような赤外線リモコンの波形を分析し、その信号に応じてプログラムを制御することができてたのしかった。