

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

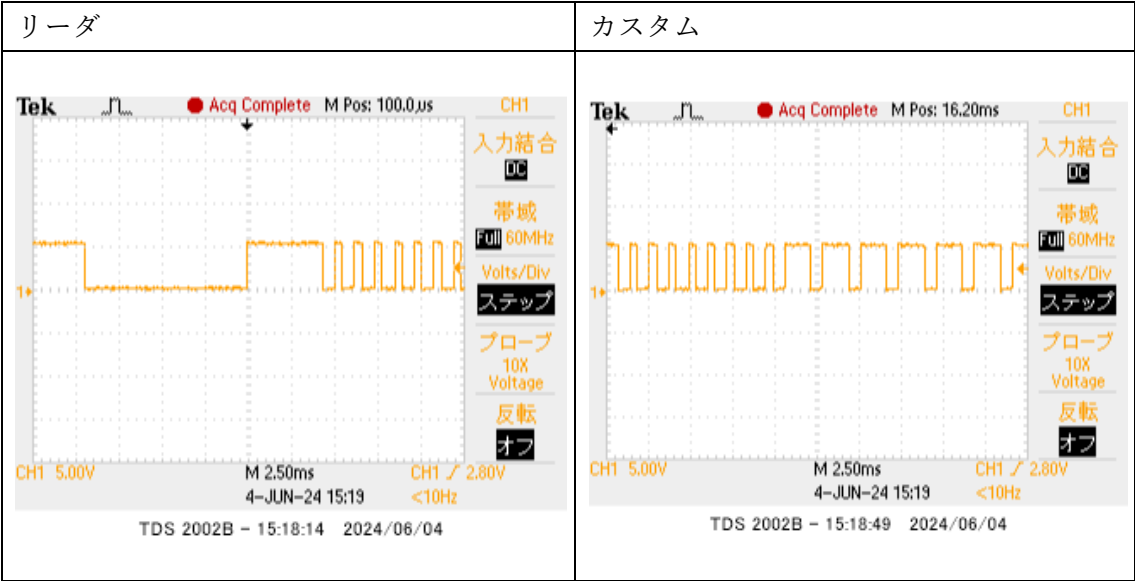
赤外線リモコンは 38KHz の搬送波を用いた通信機器である。ここでは、マイコン“ATmega328p”を用いて赤外線リモコンを作成し、動作を学ぶ。

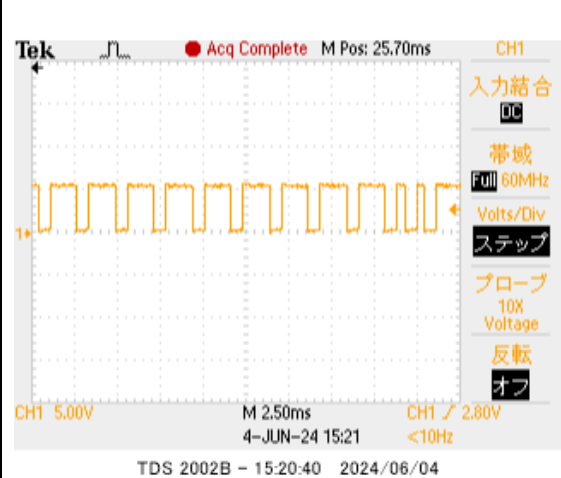
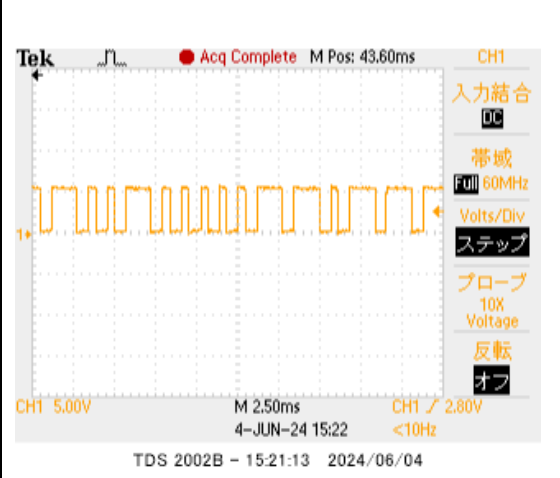
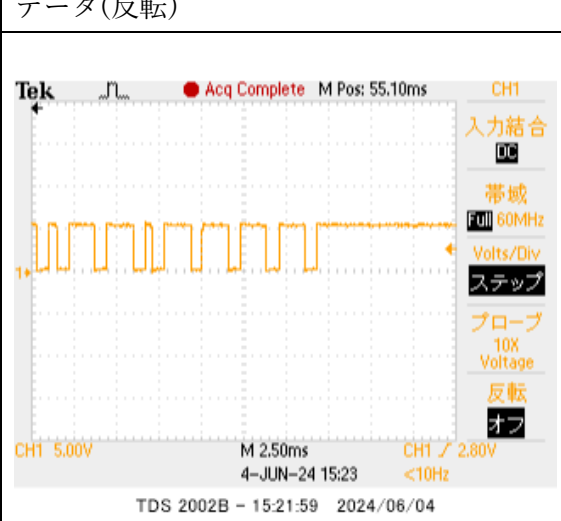
実験結果

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



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);

//リーダーコード許容範囲(カウンタ値)プリスケアラ設定 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 = 0b00000000; //タイマ0割り込み停止
                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;
}

```

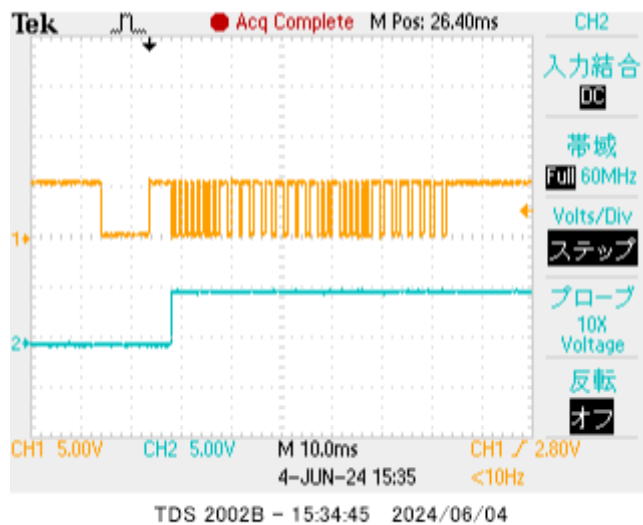
```
}

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、プリスケール設定 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



上よりリーダー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 : //立ち下がリエッジ
            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 : //立ち上がリエッジ
            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
            }
            else //範囲外ならリスタート
                restart();
            break;
    }
}

```



```

        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 : //立ち下がリエッジ
    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 : //立ち下がリエッジ
    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 : //立ち下がリエッジ
    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 : //立ち下がリエッジ
    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
}

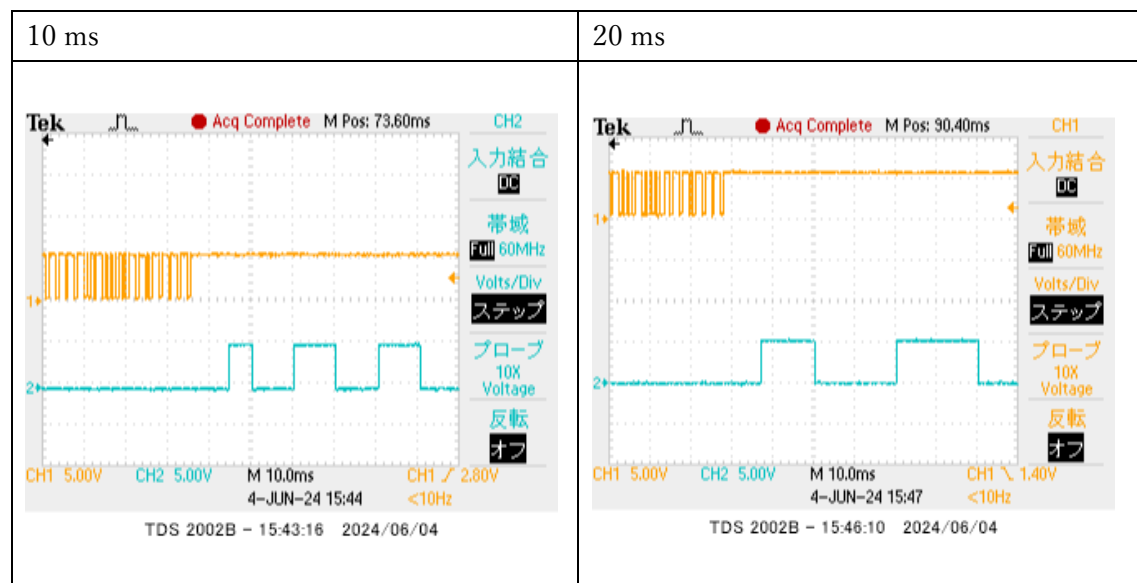
void restart(void) //リスタート
{
    start(); //受信開始
    return;
}

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



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。