情報システム工学実験Ⅰ 2023
第三回レポート

2022531033 22 班 関川謙人
d2531033@eng.kitakyu-u.ac.jp

2023 年 11 月 14 日

1　目的

1. 基本的な IrDA の仕組みを理解する
2. 自力で簡単な IrDA のハード、ソフトを組み立てる
3. 実践したことや理解したことを全てレポートにまとめる

この実験の目的は以上のステップを踏むことによって赤外線通信を学び、デバッグ法やハードウェア及びソフトウェアを活用する力を身に着けることにある。

2　実験方法

2.1　ハード面

以下の図において、白いスイッチが押されているとき、赤色の LED が光り、緑色のスイッチを押したときに緑色の LED が光る仕様になっている。

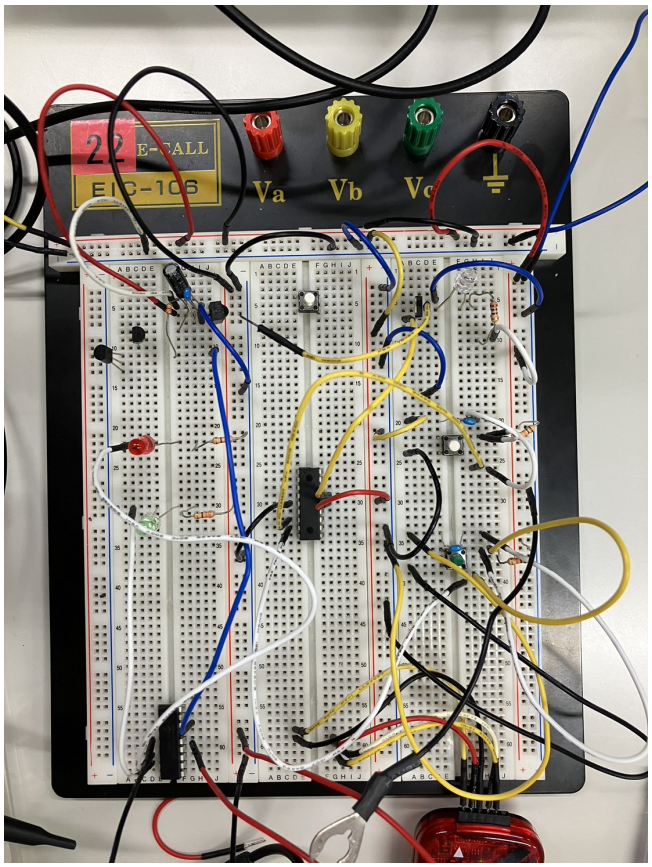

図 1: ブレッドボード

送信部の仕様は以下の通りである。

- PB1 で白いスイッチを読み取る
- PB2 で緑色のスイッチを読み取る
- PA6 で赤外線 LED に信号を送る。

また受信部の仕様は以下の通りである。

- 受光機で受け取った信号を PA7 で読み取る。
- PB1 が High のとき、赤色の LED は OFF。Low の時、赤色の LED は ON
- PB2 が HIgh のとき、緑色の LED は OFF。Low の時、緑色の LED は ON

以上の仕様をブロックダイアグラムと電気回路で示した。このブロックダイアグラムでは白いスイッチを sw1、緑色のスイッチを sw2。赤色の LED を LED1、緑色の LED を LED2 としている。

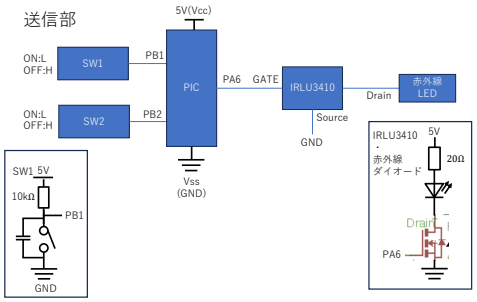

図 2: 送信側のブロックダイアグラム

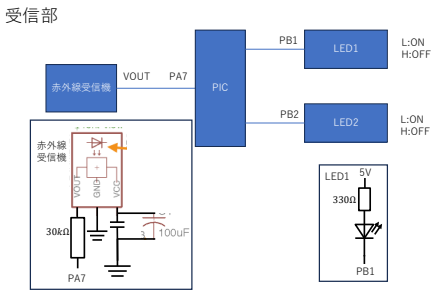

図 3: 受信側のブロックダイアグラム

2.2 ソフト面

送信部の仕様は以下の通りである。

1. 100 ミリ秒待機する
2. PB1,PB2 を読み取り、ON、OFF どちらの信号を送信するか決める。
3. スタートビット信号を送信する。
4. CH1 の信号を、PB1 の判定を基に送信する。
5. CH2 の信号を、PB2 の判定を基に送信する。
6. ストップビット信号を送信する。
7. 始めに戻る。

ON の信号はバースト信号を 920us、OFF の信号はバースト信号を 310us 送信するものである。

受信部の仕様は以下の通りである。CH1、CH2 について、500us を基準に信号の ON/OFF を判定している。

1. スタートビットを検知する
2. CH1 を検知する
3. CH2 を検知する
4. ストップビットを検知する
5. CH1、CH2 について、信号の ON/OFF を判定する
6. ON の時はポートを Low、OFF の時はポートを High にする。

この手順を以下のフローチャートに表した。

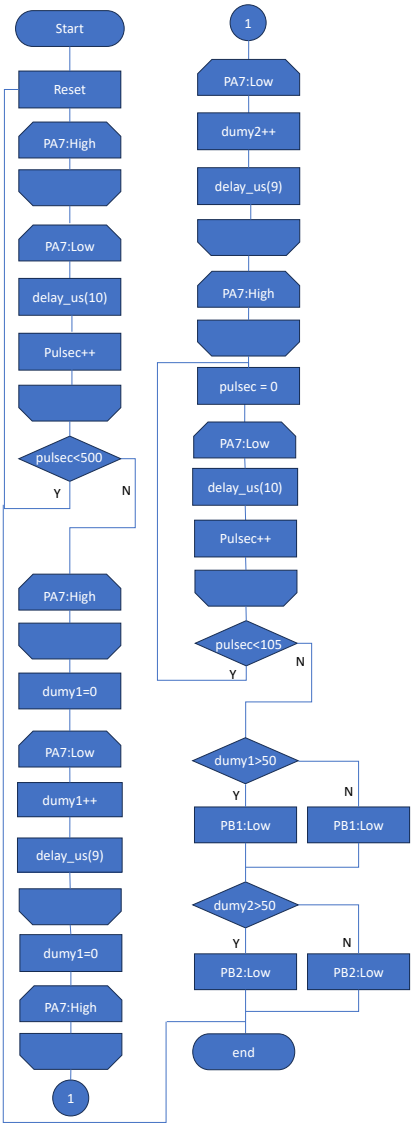

図 4: 送信側のフローチャート

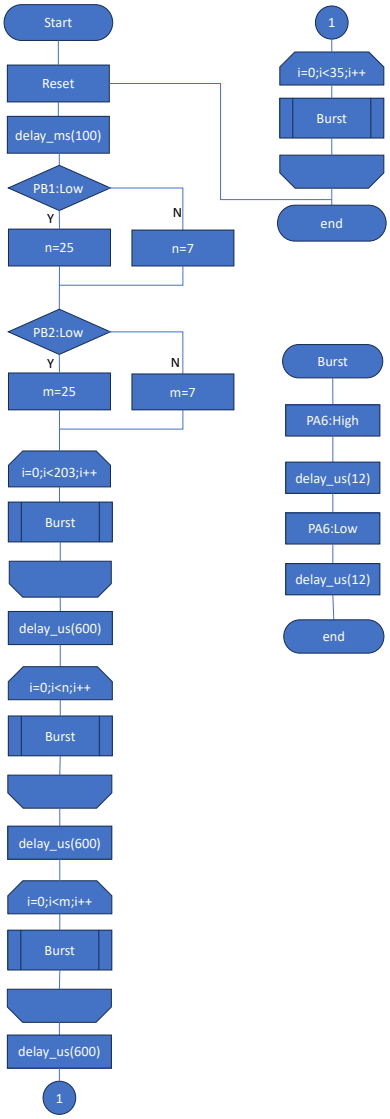

図 5: 受信側のフローチャート

3　実験結果

オシロスコープで送信側の LED の信号と受光機から出るシグナルを計測した結果、以下の図のようになった。上の信号が送信側、下の信号が受信側の信号である。

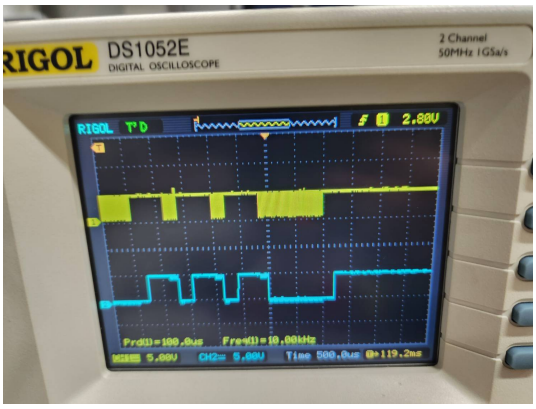

図 6: スイッチが押されていないとき

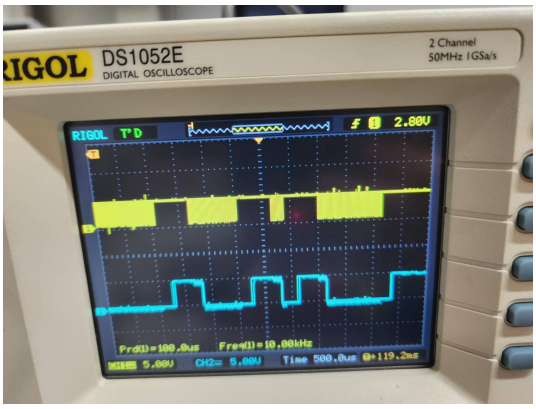

図 7: 白いスイッチが押されているとき

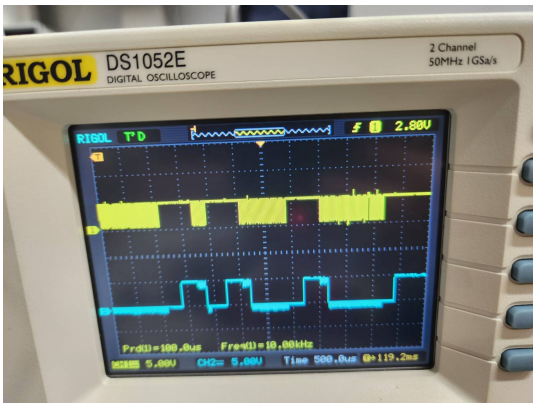

図 8: 緑色のスイッチが押されているとき

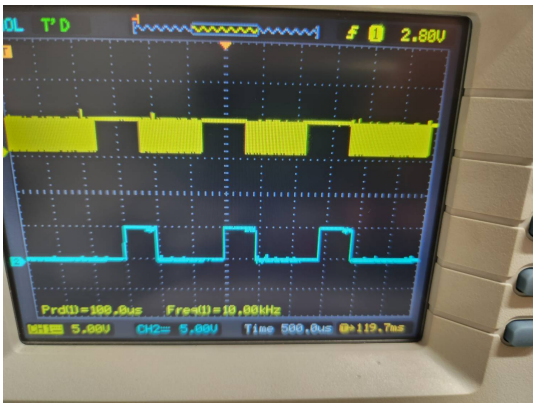

図 9: スイッチが両方押されているとき

この図からスイッチが押されたとき送信側はバースト信号を、受信側は Low の信号を押されていない時よりも長い時間示すことがわかる。またこの結果、ブレッドボード上の LED の挙動は以下のようになった。

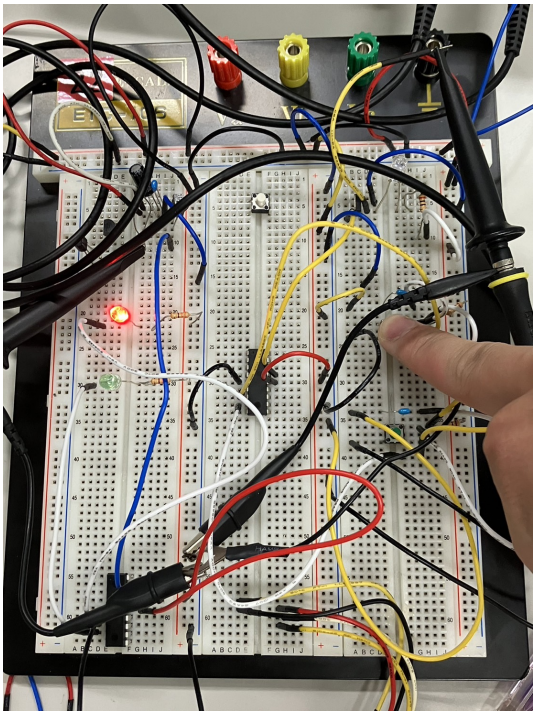

図 10: 白いスイッチが押されているとき

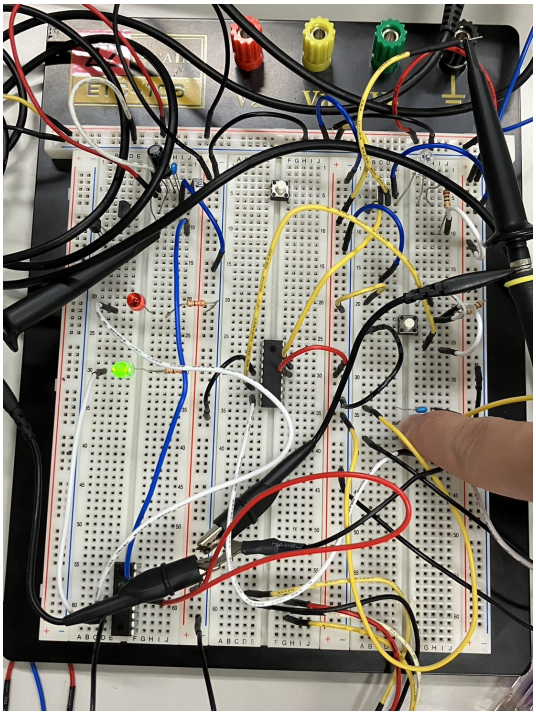

図 11: 緑色のスイッチが押されているとき

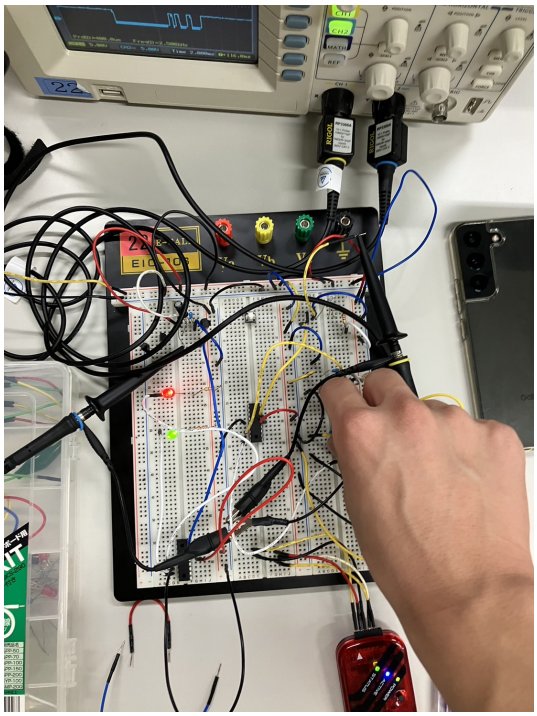

図 12: スイッチが両方押されているとき

4　考察

4.1　LED 回路

受信部の PIC の PB1 及び PB2 を High にした時、LED1 と LED2 が消灯し、Low にしたときに点灯する問題が発生した。この問題が発生した原因は、LED 部分の回路が左側の図のようになっていたことにある。

このようになっているとポート側を High にしたとき電流が流れず、LED が消灯する。そこで右側のようにするとポート側を High にしたときに LED が点灯するようになるということに気が付いたが時間が足りず、実現できなかった。

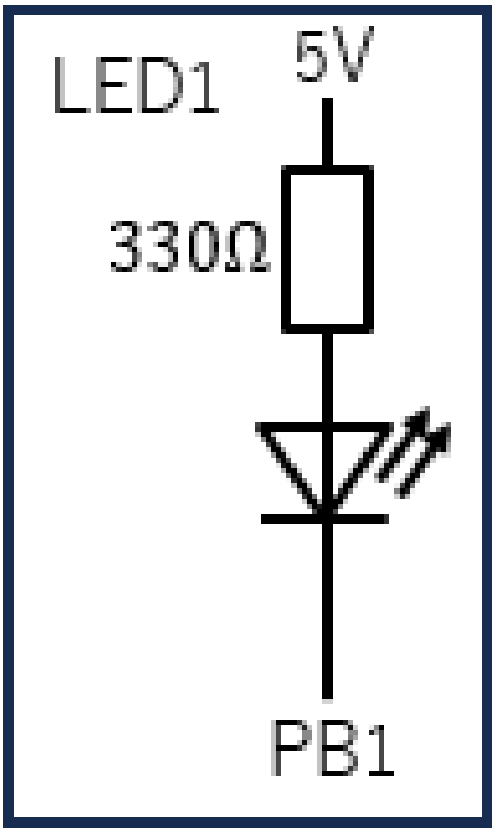

図 13: LED 部分の回路

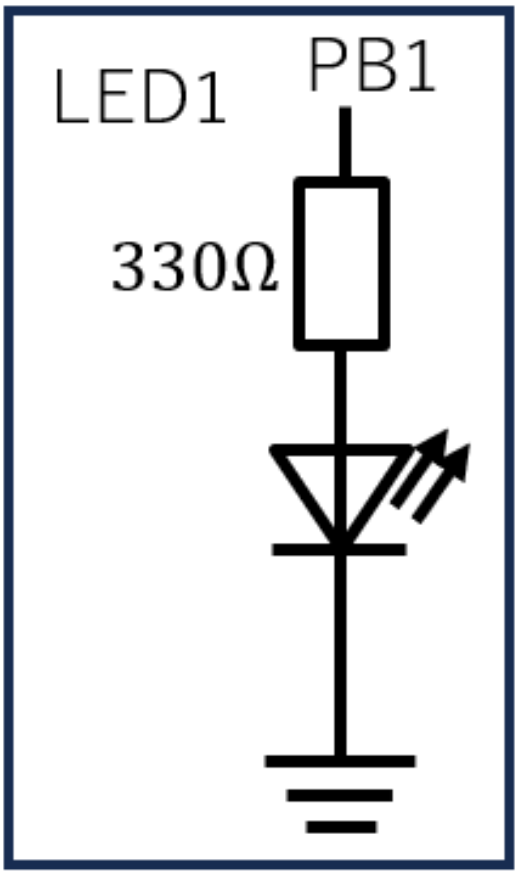

図 14: LED 部分の改善案

4.2　反省

他にも実験終了後に気が付いた以下の反省点、前回からの改善点ある。

- 電気回路を組む前にブロックダイアグラムを組む等してハードについての理解を深めるべきであった。
- プログラムに使用するユーザー関数をもっと有用に使うべきであった。
- オリジナルのプログラムを組む前にフローチャートを組んでペアと打ち合わせを行うことで、ソフトウェア面でのミスを減らすことができた。
- インデントの配置を意識してプログラムを組んだ。

ユーザー関数の使い方については、Burst 関数の使用法に改善の余地があった。Burst 関数を引数を利用して以下のように

```
1     Burst(n)
2     {
3         int i;
4         for(i=0;i<n;i++)
5         {
6             BIT_SET(*PORTA,6);
7             delay_us(12);
8             BIT_CLEAR(*PORTA,6);
9             delay_us(12);
10        }
11    }
```

とすれば main 関数の中身を簡略化できたのではないかと考えている。

4.3 実験を通しての考察

実験結果から、送信側のバースト信号を受光機が受信することで、赤外線による LED 制御が可能であることがわかる。

5 感想

5.1 IrDA 基礎の感想

IrDA 通信に入った時最初に考えたことは絶対にどこかで詰まるだろうということであった。ハード面での私の理解が足りずに最初のバースト信号の理解で詰まり友人に助けてもらったことがあったためハード面を追及して行きたい。

5.2 全体を通しての感想

この授業全体を通して特定の課題で詰まるなどして苦労した。なぜ実現できないのかということがわからずに特定の課題に 2 回分かかってしまったこともあった。しかしこの経験を通してソフト面、ハード面の仕組みの理解を深めてから実行に移すことの重要性を理解することができた。

6　ソースコード

ソースコードを以下に示す。

6.1　送信側のソースコード

Listing 1: 送信側プログラム

```
//
// Seminer :Key & LED Program for 16F84A
//             7/3/2003
// Key & LED Replace Version 11/05/2003
#include <16f819.h>
#fuses   NOWDT,NOPROTECT,NOPUT,INTRC_IO,NOMCLR
#use     delay(clock=8000000)
#include <stdlib.h>

// 16F84 Definition

#DEFINE  PORTA 0x05        // Define Port A Reg. Address
#DEFINE  PORTB 0x06        // Define Port B Reg. Address
//#DEFINE  PORTC 0x07        // Define Port B Reg. Address
#DEFINE  STATUS 0x03       // Define STATUS Reg. Address
#DEFINE  TRISA  0x85       // Define TRISA Reg. Address
#DEFINE  TRISB  0x86       // Define TRISB Reg. Address
//#DEFINE  TRISC   0x87      // Define TRISB Reg. Address
//
//   IO Init :   Subrutines Example
//
void INT_IO(void){
   #use fast_io(A)
   #use fast_io(B)
   //#use fast_io(C)
   // RA4, RA3, RA2, RA1, RA0, Push SW Info Obtain. Set to INPUT.
   // TRISA ***1 1111   0x1F

   set_tris_A(0x00);        // Set Port A Conditions(Mode)

   // RB6-RB0:OUT, RB7:OUT    LED Control
   // TRISB 0000 0000, 0x00

   set_tris_B(0xFF);       // Set Port B Conditions

   //set_tris_C(0x00);

   setup_oscillator(OSC_8MHZ);

   return;
}
//
// IrDA Burst Routine
//
void Burst()
{
    BIT_SET(*PORTA,6);
   delay_us(12);
   BIT_CLEAR(*PORTA,6);
   delay_us(12);
}
//
//    Main Programs Begin from Here.
//
void   main()
{
```

```
57     INT_IO();     // Call INT_IO() subrutines for INITIALIZE PORTs.
58     //IrDA send
59     long int i;//count times of burst
60     long int n,m;//times of burst
61     START:
62  // Wait 100ms
63     delay_ms(100);
64  //
65  // Search Switch and Judge ON/OFF ON:0 OFF:1
66  //
67      //CH1 ON/OFF ON:25 OFF:7
68      if(BIT_TEST(*PORTB,1) == 0){n = 25;}
69      else{n = 7;}
70      //CH2 ON/OFF ON:25 OFF:7
71      if(BIT_TEST(*PORTB,2) == 0){m = 25;}
72      else{m = 7;}
73  //
74  // Start Bit(6.7ms)
75  //
76     for(i=0;i<203;i++)
77     {
78      Burst();
79     }
80     delay_us(600);
81  //
82  // CH1 Burst loop(500us)
83  //
84     for(i=0;i<n;i++)
85     {
86      Burst();
87     }
88     delay_us(600);
89  //
90  // CH2 Burst loop(310us)
91  //
92     for(i=0;i<m;i++)
93     {
94      Burst();
95     }
96     delay_us(600);
97  //
98  // Stop Bit(1.2ms)
99  //
100    for(i=0;i<35;i++)
101    {
102     Burst();
103    }
104 // Back Routine
105    GOTO START;
106 }
```

6.2 受信側のソースコード

Listing 2: 受信側プログラム

```
1  //
2  // Seminer :Key & LED Program for 16F84A
3  //             7/3/2003
4  // Key & LED Replace Version 11/05/2003
5  #include <16f819.h>
6  #fuses   NOWDT,NOPROTECT,NOPUT,INTRC_IO,NOMCLR
7  #use     delay(clock=8000000)
8  #include <stdlib.h>
9
10 // 16F84 Definition
11
12 #DEFINE  PORTA 0x05        // Define Port A Reg. Address
```

```
#DEFINE  PORTB 0x06        // Define Port B Reg. Address
//#DEFINE  PORTC 0x07        // Define Port B Reg. Address
#DEFINE  STATUS 0x03       // Define STATUS Reg. Address
#DEFINE  TRISA  0x85       // Define TRISA Reg. Address
#DEFINE  TRISB  0x86       // Define TRISB Reg. Address
//#DEFINE  TRISC   0x87      // Define TRISB Reg. Address
//
//   IO Init :   Subrutines Example
//
void INT_IO(void){
   #use fast_io(A)
   #use fast_io(B)
   //#use fast_io(C)
   // RA4, RA3, RA2, RA1, RA0, Push SW Info Obtain. Set to INPUT.
   // TRISA ***1 1111   0x1F

   set_tris_A(0xFF);        // Set Port A Conditions(Mode)

   // RB6-RB0:OUT, RB7:OUT    LED Control
   // TRISB 0000 0000, 0x00

   set_tris_B(0x00);       // Set Port B Conditions

   //set_tris_C(0x00);

   setup_oscillator(OSC_8MHZ);

   return;
}
//
//    Main Programs Begin from Here.
//
void   main()
{
   INT_IO();    // Call INT_IO() subrutines for INITIALIZE PORTs.
   //IrDA read
   long pulsec;
   int dumy1,dumy2;
   START:
   CHK_START_BIT:
    pulsec = 0;
    while(BIT_TEST(*PORTA,7));//IrDA High:= No Work
    while(!BIT_TEST(*PORTA,7)) //IRDA Low: Count signals
    {
        delay_us(10);
        pulsec++;
    }
    if(pulsec<500)      //JUDGE START BIT
        GOTO CHK_START_BIT;
   //
   // Read Fisrt Parameter(CH1)
   //
    while(BIT_TEST(*PORTA,7));// Next Bit Waiting
    dumy1 = 0;
    while(!BIT_TEST(*PORTA,7))
    {
        dumy1++;
        delay_us(9);
    }
   //
   // Read Second Parameter(CH2)
   //
    dumy2 = 0;                  //Temp Param CLR
    while(BIT_TEST(*PORTA,7));//Next Bit Waiting
    while(!BIT_TEST(*PORTA,7))
    {
        dumy2++;
        delay_us(9);
    }
   //
   // Read Stop Bit
   //
```

```
85      CHK_STOP_BIT:
86      pulsec = 0;
87      while(BIT_TEST(*PORTA,7));
88      while(!BIT_TEST(*PORTA,7))
89      {
90          delay_us(10); //10 us Interrive
91          pulsec++;
92      }
93      if(pulsec < 105)    //Judge Stop Bit
94          GOTO CHK_STOP_BIT;
95     //
96     //Arrange
97     //
98      //ON:920us OFF:310us
99      //CH1
100     if(dumy1>50) //judge 500us
101         BIT_CLEAR(*PORTB,1); //LED:ON
102     else
103         BIT_SET(*PORTB,1);//LED:OFF
104     //CH2
105     if(dumy2>50)
106         BIT_CLEAR(*PORTB,2);//LED:ON
107     else
108         BIT_SET(*PORTB,2);//LED:OFF
109    //Back Routine
110    GOTO START;
111 }
```