IR - NEC-Remote - Pic-Source - GitHub

Autor:
Data de publicació: 10-06-2014

Introduction

This little project will demonstrate how you can use your old NEC IR protocol based TV,DVD or VCR remote control to control you home appliances like fan bulb or virtually anything.

Consumer IR protocols

There are a number of consumer Infrared protocols out there and they have been used for every single purpose possible i guess, like PDA laptops and other consumer appliances. RC-5 & RC-6 by Phillips , RCA are few examples of consumer IR protocols.

In this demonstration we will stick the to NEC protocol by NEC corporation,

FOR NEC Protocol Based Remote Control for this Bord

Remote Control for This Board

NEC Infrared Protocol

A 9ms leading pulse burst (16 times the pulse burst length used for a logical data bit)
A 4.5ms space

The 8-bit address for the receiving device
The 8-bit logical inverse of the address
The 8-bit command
The 8-bit logical inverse of the command
Final 562.5µs pulse burst to show end of message transmission.

Bit Timing

Logical '0' – a 562.5µs pulse burst followed by a 562.5µs space, with a total transmit time of 1.125ms
Logical '1' – a 562.5µs pulse burst followed by a 1.6875ms space, with a total transmit time of 2.25

The transmission of 0 and 1 is shown in the image blow

There are four bytes of data bits are being sended in least significant bit first order the figure blow shows the format of an NEC IR transmission frame, for a command of 0xB1 (10110001b) and an address of 0x8D (10001101b) .

16 bits for the address (address + inverse) require 27ms to transmit time .and the 16 bits for the command (command + inverse) also require 27ms to transmit time.
because (address + address inverse) or (command+command inverse) will always contain 8 '0's and 8 '1's so (8 * 1.125ms) + (8 * 2.25ms) == 27 ms .
according to this total time required to transmit the frame is (9ms +4.5ms +27ms+27ms) = 67.5 ms.

Verify with Oscilloscope and logic analyser

the image give blow is take by Rigol DS1052E Oscilloscope

Logic Analyser shows the timing details
T1 leading pulse at 84.115ms
T2 space on 93.28ms
T3 Address starts at 97.580ms
T4 Address ends , address inverse starts 107.670ms
T5 address inverse ends , command starts at 124.486ms
T6 Command ends, command inverse starts 135.696ms
T7 Command inverse ends and last 562.5µs pulse to show end of transmission

Extended NEC protocol (not used in this demonstration)

The NEC protocol is so widely used that soon all possible addresses were used up. By sacrificing the address redundancy the address range was extended from 256 possible values to approximately 65000 different values. This way the address range was extended from 8 bits to 16 bits without changing any other property of the protocol. The command redundancy is still preserved. Therefore each address can still handle 256 different commands.in extended protocol instead of sending address and address inverse we send address low and address high as shown in the image blow.

Repeat Codes

If the key on the remote controller is kept depressed, a repeat code will be issued, typically around 40ms after the pulse burst that signified the end of the message. A repeat code will continue to be sent out at 108ms intervals, until the key is finally released. The repeat code consists of the following, in order:

A 9ms leading pulse burst
A 2.25ms space
A 562.5µs pulse burst to mark the end of the space (and hence end of the transmitted repeat code).
the figures give blow show the timing of repeat codes

if user keeps the key depressed the repeat codes keep coming

T8 shows the timing of repeat code

Decoding NEC Protocol with microcontroller

Decoding NEC is really easy ,there are certainly various methods to do, some examples i have see n used polling method in which the firmware keep polling the input pin of microcontroller which connects to IR sensor , other method is to user interrupt ,in this demonstration we will be using the interrupt method as this one is better, we will be using interrupt on change.

Schematic :

SOFTWARE

GitHub Repo

Click here DOWNLOAD THE SOURCE CODE , FIRMWARE And Schematic

---
https://www.edaboard.com/threads/ir-transmitting-and-receiving-using-pic12f675.321290/

PROTOTYPE

https://www.edaboard.com/threads/ir-transmitting-and-receiving-using-pic12f675.321290/

i have read this thread to operate relay using Remote .
file:///H:/Proteus%20Project/NEC%20IR%20REMOTE%20ALL/NEC%20IR%20REMOTE%20receiver/Microcontroller%20Projects%20%20IR%28infrared%29%20Remote%20Control%20Relay%20Board%20with%20PIC%2012F675%20Microcontroller.htm

Transmitter code is

Code:

```
// Global includes
#include <htc.h>

__CONFIG(FOSC_INTRCIO & WDTE_OFF & PWRTE_OFF & MCLRE_OFF & CP_OFF & CPD_OFF);
/*
 *
 */
#define LED  GPIObits.GPIO5  // IR port status indicator LED defination
#define RELAY1  GPIObits.GPIO0  // RELAYS PORT definations
#define RELAY2  GPIObits.GPIO1
#define RELAY3  GPIObits.GPIO2
#define RELAY4  GPIObits.GPIO4

#define IRSENSOR GPIObits.GPIO3  // IR PORT defination



//#define TICKSPERMS  1004      // tick in a milli second
#define TICKS11ms  11044       // ticks in 11ms
#define TICKS5o5ms  5522   // ticks in 5.5ms
#define TICKS2o3ms  2309   // ticks in 2.3ms
#define TICKS3ms   3012  // ticks in 3sm
#define TICKS0o2ms 200   // ticks in 0.2ms
#define TICKS8ms  8032  // Tick

unsigned int TIMEOUT  =   TICKS11ms;        // the pulse should occur before this time excede Otherwise it is an error
unsigned int PREPULSE = TICKS8ms;   // the interrupt should occur after this time Otherwise it is an error


static unsigned short long timer;   // varible to keep track of long timeouts  ( it can also be int if you want to save flash memory for some other purpose  )
static unsigned char dataready;    // varible to use as flag when data is completly received and  ready it is 1 else 0

static  unsigned char necpoj=0;       /* (necpoj=NEC position )this varible is used to keep track of the edges of the input singal
        as decoding of the singal is done by a state machine
        so this varible acutalley sotores what state we currently are
         and total bits 32 and 2 leading pulse */
```

```
static unsigned char address=0,notaddress=0; // these varible are used to store received address
static unsigned char command=0,notcommand=0; // these varible are used to store received address

void interruptOnChangeIsr(void);      // interrupt service routine for interrupt on change of input port for IR sensor of
mcu
void timerInterruptIsr(void);     // interrupt service rouine for timer0


void interrupt t0intr(void)
{
 if(INTCONbits.T0IF)       // check the timer0 over flow interrupt flag
 {
 timerInterruptIsr();      // timer0 overflow interrupt has been occur call the isr
 INTCONbits.T0IF =0;       // clear the timer0 interrupt flag
 }
 else if (INTCONbits.GPIF)     // check the interrupt on change flag
 {
  LED=1;         // to blink the LED when IR signal is received
  interruptOnChangeIsr();     // interrupt on change has been detected call the isr
  INTCONbits.GPIF =0;      // clear the interrupt on chage flag
  LED=0;         // to blink the LED when IR signal is received
 }

}

/* THE main source code Start here*/


void main()
{

    CMCON=0x7;    // disable the comparator
    ANSEL=0x00;    // all pin are Digital
    TRISIO=0x8;      // Only GP2 is set to input rest are out
 TMR0 = 0;    // clar the timer
 OPTION_REG = 0x88;  //pullups are disabled
       //timer0 clock source is internal
       //timer0 perscaller is 1:1 (disabled "assigned to WDT")
 IOC = 0x8;      //interrupt on change is only to the GPIO3
 INTCONbits.T0IE = 1; // Timer0 overflow interrupt enable
 INTCONbits.T0IF = 0;    // clar the timer0 intrrupt flags
 INTCONbits.GPIE = 1; // external interrupt on GPIO3 pin(4) is enabled
 INTCONbits.GPIF = 0; // clear the external interrrupt flag
 INTCONbits.PEIE = 1;    // peripheral intrrupt enable
 INTCONbits.GIE = 1;     // GLOBL interrupt enable



 EEADR = 0x00;   // load the state of port from EEPROM
 EECON1bits.RD = 1;  // Start reding EEPORM
 GPIO = EEDATA;    // LOAD The readed data form EEPORM to GPIO

 while(1)         // wait forever for  the data received and ready
 {

 if(dataready)   // data is received and ready to be procssed
 {


  // key1  0x50 to Toggle relay 1 // these are command of the IR remote control which i have
```

```
  // key2  0xD8 to Toggle relay 2
  // key3  0xF8 to Toggle relay 3
  // key4  0x30 to Toggle relay 4
  // key5  0xB0 to Turn off all the relays

  switch(command) // swich on
  {
  case 0x50: RELAY1 = !RELAY1;  //Toggle relay 1
      break;
  case 0xD8: RELAY2 = !RELAY2;  //Toggle relay 2
      break;
  case 0xF8: RELAY3 = !RELAY3;  //Toggle relay 3
      break;
  case 0x30: RELAY4 = !RELAY4;  //Toggle relay 4
      break;
  case 0xB0: RELAY1 = 0;    //Turn off all the relay
      RELAY2 = 0;
      RELAY3 = 0;
      RELAY4 = 0;
      break;
  default :
     break;
  }

  EEADR = 0x00;    //Write PORT status to EEPROM
  EEDATA = GPIO;   // load the current status of GPIO to EEPROM write register
  EECON1bits.WREN = 1;   // Enable EEPROM write
  INTCONbits.GIE = 0;  //1 disable the interrupts as it may currupt the EEPROM data
  EECON2 = 0x55;   //2
  EECON2 = 0xAA;   //3 (1,2,3) require sequence
  EECON1bits.WR = 1;   // satart writing
  INTCONbits.GIE = 1;   // Enable the interrupts

  dataready=0;   // data has been processed so clear the dataready flag

 }

 }

}

void interruptOnChangeIsr(void)
{

unsigned short long tdiff;
unsigned char pin;
static unsigned long rxbuffer;


tdiff = ((timer<<8)+TMR0) ;     // calculate how much time has been passed since last interrupt
       // the time shold be less then time out and greater than PREPULSE
pin = IRSENSOR;     // store the current status of Sensor
TMR0 = 0;      // reset the timer0 to measure the next edge(interrupt) of input
timer = 0;      // reset the timer varible to




/* state machine is started here and it totally managed and keeps track of its states using the varible necpoj
here are the details of necpoj ( NEC position ) varible
```

if
necpoj == 1 we just detected the first edge of the input singal it may also mean(if interrupt is not false) that the 9ms leading pulse started
  after the first edge THE next pulse is expected to arrive around 9ms so the TIMEOUT is set to 11ms and PREPULSE is set to 8ms

necpoj == 2 we just detected the second edge of the input signal and we finished the 9ms leding pulse and now 4.5ms space started
  after the second edge the next pulse is expected to arrive around 4.5ms so TIMEOUT is set to 5.5ms and PREPULSE is 3ms

necpoj == 3 we just detected the third edge of the input singal and we finished 4.5ms space and addres lsb is now started
  after the third edge the next pulse is expected to arrive around 562.5us so TIMEOUT is set to 2.3ms and PREPULSE is 0.2ms (timeout can be much less at this state but to do this i have to add one more if else statemetnt)

necpoj == 4 we just decected the forth edge and the 562.5 us burt of LSB of address has ended now a little space for '0'562.5us or for '1' 1.6875ms
  after the forth edge the next pulse is expected to arrive for '0' around 562.5us and for '1' 1.675ms so TIMEOUT is set to 2.3ms and PREPULSE is 0.2ms

necpoj ==5 to 66 data pulse keep comming
  TIMOUT and PREPLUSE remain same as above.

necpoj ==67 we just fined the command inverse MSB space not the final 562.5us burst has stated so we fined the receiveing
  now we will check the address and command for being correct
*/

if ((tdiff>PREPULSE) && (tdiff<TIMEOUT)) // the edge (interrupt) occurrence time should be less then the TIMOUT and greater then PREPULESE else it is an fake singal
{ // At the very first edge (necpoj==0) this conditon will always false and the false block of this if will bring the state machine (necpoj) to position 1(position 1 means 9ms leading pulse has started now we have to wait for 4.5ms start pulse to occur)

 if(necpoj==1 || necpoj==2) // when we are hear it means 9ms leding pulse has ended and now we are necpoj=1 or necpoj=2
 {

  if((pin==1) && (necpoj==1))
  {
  necpoj++;
  TIMEOUT = TICKS5o5ms; // timeout for 3rd pulse 5.5ms
  PREPULSE = TICKS3ms; // PREPULSE for 3rd pulse 3ms
  }
  else if((pin==0)&& (necpoj ==2))
  {
  necpoj++;

  TIMEOUT = TICKS2o3ms; // now data starts so timeout is 2.3ms
  PREPULSE = TICKS0o2ms;

  }
  else // this block handle the conditon if any error occur after the completing the pre pulses

```
  {
  necpoj = 0;       //reset the state machine
  TIMEOUT  =   TICKS11ms;
  PREPULSE  =  TICKS8ms;
  }
 }
 else if(necpoj>2)       //now we are picking the data
 {

  necpoj++;        //necpoj sill inrement on every edge

   if(necpoj&0x01)     // here we check the if necpoj is an odd number because when necpoj goes greater then 3
then
            //necpoj will always be and odd value when a single bit tranmission is over
   {
    rxbuffer=rxbuffer<<1;  //shift the buffer
    if(tdiff>1250)    //we are here means we just recevied the edge of finished tranmission of a bit
          // so if last edge was more than 1.24 ms then the bit which is just over is one else it is zero
    {
    rxbuffer = rxbuffer | 0x1;
   // GPIObits.GPIO5 = !GPIObits.GPIO5;
    }
    else
    {
    rxbuffer = rxbuffer |0x0;
      // GPIObits.GPIO4 = !GPIObits.GPIO4;
    }

   }

  if(necpoj >66)      // we have reached (Leading pulse 2 +address 16+~address16+ command 16+ ~command 16+
last final burst first edge 1)=67th edge of the message frame means the date tranmission is now over
  {

  address  = (rxbuffer>>24)& 0xFF;   //extract the data from the buffer
  notaddress  = (rxbuffer>>16)& 0xFF;
  command  = (rxbuffer>>8) & 0xFF;
  notcommand  = (rxbuffer) & 0xFF;
  rxbuffer=0;         //clear the buffer

    if((!(address & notaddress)) && (!(command & notcommand)))  // check weather the received data is vaild or not
    {
    dataready =1;
    }
    else
    {
    dataready=0;
    }
  TIMEOUT  =   TICKS11ms;     // weather we received the vaild data or not we have to reset the state machine
  PREPULSE  =  TICKS8ms;
  necpoj=0;
  }

 }
 else
 {

 TIMEOUT  = TICKS11ms;       // some error occured reset state machine
 PREPULSE  = TICKS8ms;
 }
```

```
}
else
{

 if(pin==0)    //we are here means that after a longtimeout or PREPULSE we just detect a pulse which may be the start of 9ms pulse
 {
 necpoj = 1;    // yes it could be the start of 9ms pulse
 }
 else
 {
 necpoj = 0;    // no it's not start of 9ms pulse
 }

 address  = 0xFF;
 notaddress  = 0xFF;
 command  = 0xFF;
 notcommand  = 0xFF;
 dataready  = 0x000;
 TIMEOUT =  TICKS11ms;  //default timing
 PREPULSE = TICKS8ms;
}

}

void timerInterruptIsr(void)
{
if(timer<0xFFFF)   // this code is to increment the variable timer's value on every over flow but this if conditon will prevent this variable form rollover when a long timeout occurs
timer++;
}
```

and Receiver code is

Code:

```
// Global includes
#include <htc.h>

__CONFIG(FOSC_INTRCIO & WDTE_OFF & PWRTE_OFF & MCLRE_OFF & CP_OFF & CPD_OFF);
/*
 *
 */
#define LED  GPIObits.GPIO5  // IR port status indicator LED defination
#define RELAY1  GPIObits.GPIO0  // RELAYS PORT definations
#define RELAY2  GPIObits.GPIO1
#define RELAY3  GPIObits.GPIO2
#define RELAY4  GPIObits.GPIO4

#define IRSENSOR GPIObits.GPIO3  // IR PORT defination
```

```
//#define TICKSPERMS  1004      // tick in a milli second
#define TICKS11ms  11044       // ticks in 11ms
#define TICKS5o5ms  5522   // ticks in 5.5ms
#define TICKS2o3ms  2309   // ticks in 2.3ms
#define TICKS3ms   3012  // ticks in 3sm
#define TICKS0o2ms 200   // ticks in 0.2ms
#define TICKS8ms  8032  // Tick

unsigned int TIMEOUT  =   TICKS11ms;        // the pulse should occur before this time excede Otherwise it is an error
unsigned int PREPULSE = TICKS8ms;   // the interrupt should occur after this time Otherwise it is an error


static unsigned short long timer;   // varible to keep track of long timeouts  ( it can also be int if you want to save flash memory for some other purpose  )
static unsigned char dataready;    // varible to use as flag when data is completly received and  ready it is 1 else 0

static  unsigned char necpoj=0;       /* (necpoj=NEC position )this varible is used to keep track of the edges of the input singal
          as decoding of the singal is done by a state machine
          so this varible acutalley sotores what state we currently are
           and total bits 32 and 2 leading pulse */


static unsigned char address=0,notaddress=0; // these varible are used to store received address
static unsigned char command=0,notcommand=0; // these varible are used to store received address

void interruptOnChangeIsr(void);      // interrupt service routine for interrupt on change of input port for IR sensor of mcu
void timerInterruptIsr(void);     // interrupt service rouine for timer0


void interrupt t0intr(void)
{
 if(INTCONbits.T0IF)       // check the timer0 over flow interrupt flag
 {
 timerInterruptIsr();      // timer0 overflow interrupt has been occur call the isr
 INTCONbits.T0IF =0;       // clear the timer0 interrupt flag
 }
 else if (INTCONbits.GPIF)     // check the interrupt on change flag
 {
  LED=1;         // to blink the LED when IR signal is received
  interruptOnChangeIsr();     // interrupt on change has been detected call the isr
  INTCONbits.GPIF =0;      // clear the interrupt on chage flag
  LED=0;         // to blink the LED when IR signal is received
 }

}

/* THE main source code Start here*/


void main()
{

    CMCON=0x7;    // disable the comparator
    ANSEL=0x00;    // all pin are Digital
    TRISIO=0x8;      // Only GP2 is set to input rest are out
 TMR0 = 0;    // clar the timer
 OPTION_REG = 0x88;  //pullups are disabled
       //timer0 clock source is internal
```

```
      //timer0 perscaller is 1:1 (disabled "assigned to WDT")
 IOC = 0x8;      //interrupt on change is only to the GPIO3
 INTCONbits.T0IE = 1; // Timer0 overflow interrupt enable
 INTCONbits.T0IF = 0;    // clar the timer0 intrrupt flags
 INTCONbits.GPIE = 1; // external interrupt on GPIO3 pin(4) is enabled
 INTCONbits.GPIF = 0; // clear the external interrrupt flag
 INTCONbits.PEIE = 1;    // peripheral intrrupt enable
 INTCONbits.GIE = 1;     // GLOBL interrupt enable



 EEADR = 0x00;   // load the state of port from EEPROM
 EECON1bits.RD = 1;  // Start reding EEPORM
 GPIO = EEDATA;    // LOAD The readed data form EEPORM to GPIO

 while(1)        // wait forever for  the data received and ready
 {

 if(dataready)   // data is received and ready to be procssed
 {


  // key1  0x50 to Toggle relay 1 // these are command of the IR remote control which i have
  // key2  0xD8 to Toggle relay 2
  // key3  0xF8 to Toggle relay 3
  // key4  0x30 to Toggle relay 4
  // key5  0xB0 to Turn off all the relays

  switch(command) // swich on
  {
  case 0x50: RELAY1 = !RELAY1;  //Toggle relay 1
      break;
  case 0xD8: RELAY2 = !RELAY2;  //Toggle relay 2
      break;
  case 0xF8: RELAY3 = !RELAY3;  //Toggle relay 3
      break;
  case 0x30: RELAY4 = !RELAY4;  //Toggle relay 4
      break;
  case 0xB0: RELAY1 = 0;    //Turn off all the relay
      RELAY2 = 0;
      RELAY3 = 0;
      RELAY4 = 0;
      break;
  default :
    break;
  }

  EEADR = 0x00;    //Write PORT status to EEPROM
  EEDATA = GPIO;   // load the current status of GPIO to EEPROM write register
  EECON1bits.WREN = 1;   // Enable EEPROM write
  INTCONbits.GIE = 0;  //1 disable the interrupts as it may currupt the EEPROM data
  EECON2 = 0x55;   //2
  EECON2 = 0xAA;   //3 (1,2,3) require sequence
  EECON1bits.WR = 1;   // satart writing
  INTCONbits.GIE = 1;   // Enable the interrupts

  dataready=0;   // data has been processed so clear the dataready flag

 }
```

```
 }

}

void interruptOnChangeIsr(void)
{

unsigned short long tdiff;
unsigned char pin;
static unsigned long rxbuffer;


tdiff = ((timer<<8)+TMR0) ;     // calculate how much time has been passed since last interrupt
       // the time shold be less then time out and greater than PREPULSE
pin = IRSENSOR;     // store the current status of Sensor
TMR0 = 0;      // reset the timer0 to measure the next edge(interrupt) of input
timer = 0;      // reset the timer varible to




/* state machine is started here and it totally managed and keeps track of its states using the varible necpoj
here are the details of necpoj ( NEC position ) varible
if
necpoj == 1       we just detected the first edge of the input singal it may also mean(if interrupt is not false) that the 9ms leading pulse started
     after the first edge THE next pulse is expected to arrive around 9ms so the TIMEOUT is set to 11ms and PREPULSE is set to 8ms


necpoj == 2    we just detected the second edge of the input signal and we finished the 9ms leding pulse and now 4.5ms space started
     after the second edge the next pulse is expected to arrive around 4.5ms so TIMEOUT is set to 5.5ms and PREPULSE is 3ms


necpoj == 3     we just detected the third edge of the input singal and we finished 4.5ms space and addres lsb is now started
     after the third edge the next pulse is expected to arrive around 562.5us so TIMEOUT is set to 2.3ms and PREPULSE is 0.2ms (timeout can be much less at this state but to do this i have to add one more if else statemetnt)

necpoj == 4    we just decected the forth edge and the 562.5 us burt of LSB of address has ended now a little space for '0'562.5us  or for '1' 1.6875ms
     after the forth edge the next pulse is expected to arrive for '0' around 562.5us  and for '1' 1.675ms so TIMEOUT is set to 2.3ms and PREPULSE is 0.2ms

necpoj ==5 to 66  data pulse keep comming
    TIMOUT and PREPLUSE remain same as above.

necpoj ==67    we just fined the command inverse MSB space not the final 562.5us burst has stated  so we fined the receiveing
     now we will check the address and command for being correct
*/


if ((tdiff>PREPULSE) && (tdiff<TIMEOUT) )  // the edge (interrupt) occurrence time should be less then the TIMOUT and greater then PREPULESE else it is an fake singal
{           // At the very first edge (necpoj==0)  this conditon will always false and the false block of this if will bring the state machine (necpoj) to position 1(position 1 means 9ms leading pulse has started now we have to wait for 4.5ms start pulse to occur)
```

```
 if(necpoj==1 || necpoj==2)     // when we are hear it means 9ms leding pulse has ended and now we are necpoj=1 or
necpoj=2
 {

  if((pin==1) && (necpoj==1))
  {
  necpoj++;
  TIMEOUT  = TICKS5o5ms;     // timeout for 3rd pulse 5.5ms
  PREPULSE    = TICKS3ms;    // PREPULSE for 3rd pulse 3ms
  }
  else if((pin==0)&& (necpoj ==2))
  {
  necpoj++;

  TIMEOUT  = TICKS2o3ms;     // now data starts so timeout is 2.3ms
  PREPULSE    = TICKS0o2ms;

  }
  else        // this block handle the conditon if any error occur after the completing the pre pulses
  {
  necpoj = 0;       //reset the state machine
  TIMEOUT  =   TICKS11ms;
  PREPULSE  =  TICKS8ms;
  }
 }
 else if(necpoj>2)       //now we are picking the data
 {

  necpoj++;        //necpoj sill inrement on every edge

   if(necpoj&0x01)     // here we check the if necpoj is an odd number because when necpoj goes greater then 3
then
            //necpoj will always be and odd value when a single bit tranmission is over
   {
    rxbuffer=rxbuffer<<1;  //shift the buffer
    if(tdiff>1250)    //we are here means we just recevied the edge of finished tranmission of a bit
          // so if last edge was more than 1.24 ms then the bit which is just over is one else it is zero
    {
    rxbuffer = rxbuffer | 0x1;
   // GPIObits.GPIO5 = !GPIObits.GPIO5;
    }
    else
    {
    rxbuffer = rxbuffer |0x0;
      // GPIObits.GPIO4 = !GPIObits.GPIO4;
    }

   }

  if(necpoj >66)      // we have reached (Leading pulse 2 +address 16+~address16+ command 16+ ~command 16+
last final burst first edge 1)=67th edge of the message frame means the date tranmission is now over
  {

  address  = (rxbuffer>>24)& 0xFF;   //extract the data from the buffer
  notaddress  = (rxbuffer>>16)& 0xFF;
  command  = (rxbuffer>>8) & 0xFF;
```

```
   notcommand  = (rxbuffer) & 0xFF;
   rxbuffer=0;         //clear the buffer

     if((!(address & notaddress)) && (!(command & notcommand)))  // check weather the received data is vaild or not
     {
     dataready =1;
     }
     else
     {
     dataready=0;
     }
   TIMEOUT  =   TICKS11ms;     // weather we received the vaild data or not we have to reset the state machine
   PREPULSE  =  TICKS8ms;
   necpoj=0;
   }

 }
 else
 {

 TIMEOUT  = TICKS11ms;       // some error occured reset state machine
 PREPULSE  = TICKS8ms;
 }




}
else
{

 if(pin==0)    //we are here means that after a longtimeout or PREPULSE we just detect a pulse which may be the
start of 9ms pulse
 {
 necpoj = 1;    // yes it could be the start of 9ms pulse
 }
 else
 {
 necpoj = 0;    // no it's not start of 9ms pulse
 }

 address  = 0xFF;
 notaddress  = 0xFF;
 command  = 0xFF;
 notcommand  = 0xFF;
 dataready  = 0x000;
 TIMEOUT =  TICKS11ms;  //default timing
 PREPULSE = TICKS8ms;
}

}

void timerInterruptIsr(void)
{
if(timer<0xFFFF)   // this code is to increment the variable timer's value on every over flow but this if conditon will
prevent this variable form rollover when a long timeout occurs
timer++;
}
```

and my proteus file is
View attachment SIMULATION.rar

but simulation not work please help me!!!!

- - - Updated - - -

ok sorry for some change on my circuit design please have look below simulation file . but relay is not trigger ?

Attachments

test.rar
34.6 KB · Views: 191