```
/*************************************/
/******* USART Declarations ************/
/*************************************/

#include <avr/io.h>

/******* USART *************************/

void USART_Init(void);
void USART_Send(unsigned char);
uint8_t USART_Receive(void);
```

```
/*************************************/
/******* USART Declarations ************/
/*************************************/

#include <avr/io.h>

/******* USART *************************/

void USART_Init(void);
void USART_Send(unsigned char);
uint8_t USART_Receive(void);
```

```
/************************************/
/******** USART ********************/
/************************************/

#define BAUD 4800
#include <util/setbaud.h>
#include "USART_Dec.h"


/******** USART Init ***************/
void USART_Init(void)
{
    UBRR0H = UBRR_VALUE >> 8;
    UBRR0L = UBRR_VALUE & 0xFF;

    UCSR0C = (1<<UCSZ01)|(1<<UCSZ00);              // Asynchron 8N1
    UCSR0B |= (1<<RXEN0);                          // Enable UART RX
    UCSR0B |= (1<<TXEN0);                          // Enable UART TX
    UCSR0B |= (1<<RXCIE0);                         // Enable Interrupt
}

/******* USART Send Character *******/

void USART_Send(unsigned char c)
{
    while(!(UCSR0A & (1<<UDRE0))) {}               // Wait for empty SendBuffer

    UDR0 = c;
}

/******* USART Receive Character ****/

uint8_t USART_Receive(void)
{
    while (!(UCSR0A & (1<<RXC0)));                 // Wait for full ReceiveBuffer

    return UDR0;
}
```

```
/************************************/
/***** USART PROGRAMMER MODE **********/
/************************************/

#include "USART_Dec.h"
#include "Converter.h"
#include <util/delay.h>
#include <avr/interrupt.h>

/* Global Data for EEPROM */

extern volatile uint8_t RFSendPower;
extern volatile uint8_t SensingTime;
extern volatile uint8_t SendRetry;
extern volatile unsigned char RelayPWM[4][1];
extern volatile unsigned char RelayAddress[4][4];
extern volatile unsigned char RemoteAddress[40][4];
extern volatile unsigned char GateAddress[4];
extern volatile long long int Switch1;
extern volatile long long int Switch2;
extern volatile long long int Switch3;
extern volatile long long int Switch4;

volatile uint8_t USARTFlag = 0x00;
volatile uint8_t configFlag = 0x00;
volatile uint8_t Received = 0x00;

ISR(USART0_RX_vect)
{
    uint8_t tmp_sreg = 0x00;
    volatile static uint8_t Flag = 0x00;
    volatile static uint8_t i = 0x00, j = 0x00, k = 0x00;
    uint8_t tempUDR = 0x00;

    tmp_sreg = SREG;   // save Interruptstateregister
    cli();

    switch(Flag)
    {
        case 0x5A:      // one Byte commands
            switch (tempUDR = UDR0)
            {
                case 0xB0 :         // Read all current config
                case 0xB1 :         // Read EEPROM Data
                case 0xB2 :         // Write Data to EEPROM
                case 0xB3 :         // Start Testphase
                    USARTFlag = tempUDR;
                    break;

                default :
                    break;
            }
            Flag = 0x00;
            Received = 0xFF;
            break;

        case 0x6B :                             // PWM and other configurations
                switch(i)
                {
                    case 0x00:                  // Set Retry
                        SendRetry = UDR0;
                        break;

                    case 0x01 :                 // Set RF send power
```

```
                    RFSendPower = UDR0;
                    break;

                case 0x02 :                 // Set Switchsening Time
                    SensingTime = UDR0;
                    break;

                case 0x03 :                 // Set PWM1 compare
                    RelayPWM[0][0] = UDR0;
                    break;

                case 0x04 :                 // Set PWM2 compare
                    RelayPWM[1][0] = UDR0;
                    break;

                case 0x05 :                 // Set PWM3 compare
                    RelayPWM[2][0] = UDR0;
                    break;

                case 0x06 :                 // Set PWM4 compare
                    RelayPWM[3][0] = UDR0;
                    Flag = 0x00;
                    break;
            }
            i++;
            break;

        case 0x7C :
            switch(i)
            {
                case 0x00 :              // Set Addresse P1
                    RelayAddress[0][0] = UDR0;
                    break;
                case 0x01 :
                    RelayAddress[0][1] = UDR0;
                    break;
                case 0x02 :
                    RelayAddress[0][2] = UDR0;
                    break;
                case 0x03 :
                    RelayAddress[0][3] = UDR0;
                    break;

                case 0x04 :              // Set Addresse P2
                    RelayAddress[1][0] = UDR0;
                    break;
                case 0x05 :
                    RelayAddress[1][1] = UDR0;
                    break;
                case 0x06 :
                    RelayAddress[1][2] = UDR0;
                    break;
                case 0x07 :
                    RelayAddress[1][3] = UDR0;
                    break;

                case 0x08 :              // Set Addresse P3
                    RelayAddress[2][0] = UDR0;
                    break;
                case 0x09 :
                    RelayAddress[2][1] = UDR0;
                    break;
                case 0x0A :
                    RelayAddress[2][2] = UDR0;
```

```
                break;
            case 0x0B :
                RelayAddress[2][3] = UDR0;
                break;

            case 0x0C :             // Set Addresse P4
                RelayAddress[3][0] = UDR0;
                break;
            case 0x0D :
                RelayAddress[3][1] = UDR0;
                break;
            case 0x0E :
                RelayAddress[3][2] = UDR0;
                break;
            case 0x0F :
                RelayAddress[3][3] = UDR0;
                break;

            case 0x10 :             // Default Gateaddress
                GateAddress[0] = UDR0;
                break;
            case 0x11 :
                GateAddress[1] = UDR0;
                break;
            case 0x12 :
                GateAddress[2] = UDR0;
                break;
            case 0x13 :
                GateAddress[3] = UDR0;
                break;

            default :               // RemoteAddress 1 - 40
                RemoteAddress[k][j] = UDR0;
                j++;
                if(j == 4)
                {
                    k++;
                    j = 0;
                }
                if(i == 44)
                {
                    Flag = 0x00;
                    k = 0x00;
                    j = 0x00;
                }
                break;
        }
        i++;
        break;

    case 0x8D :
        Switchregconvert.c[5-i] = UDR0;
        i++;
        if(i==6)
        {
            if(j == 0)              // Switchregister 1
            {
                Switch1 = Switchregconvert.u48i;
            }
            if(j == 1)              // Switchregister 2
            {
                Switch2 = Switchregconvert.u48i;
            }
            if(j == 2)              // Switchregister 3
```

```
                {
                    Switch3 = Switchregconvert.u48i;
                }
                if(j == 3)              // Switchregister 4
                {
                    Switch4 = Switchregconvert.u48i;
                }
                i = 0x00;
                j++;
                if(j==4)
                {
                    j = 0x00;
                    i = 0x00;
                    Flag = 0x00;
                }

            }
            break;

        default :
            switch(tempUDR = UDR0)
            {
                case 0x5A :             // one Byte commands
                case 0x6B :             // General Settings
                case 0x7C :             // Address Settings
                case 0x8D :             // Switchsettings
                    Flag = tempUDR;
                    i = 0x00;
                    break;

                default:                // false Command
                    break;
            }
            configFlag = 0xFF;
            break;
    }
    SREG = tmp_sreg;
}

void SendtoProgrammer(uint8_t USARTFlag)
{
    uint8_t j,i, tmp_sreg = 0x00;

    tmp_sreg = SREG;            // save Interruptstateregister

    cli();

    switch(USARTFlag)
    {
        case 0xB0 :                 // Read current config
            USART_Send(0x87);
            USART_Send(SensingTime);
            USART_Send(RFSendPower);
            USART_Send(SendRetry);

                // Send RelayPWM
            for(i=0; i<4; i++)
            {
                USART_Send(RelayPWM[i][0]);
            }

            // Send Address P1 to P4
            for(i=0; i<4; i++)
            {
```

```
                for(j=0; j<4; j++)
                {
                    USART_Send(RelayAddress[i][j]);
                }
            }

            // Send Gateaddress
            for(i=0; i<4; i++)
            {
                USART_Send(GateAddress[i]);
            }

            _delay_ms(100);
            // Send Remoteaddresses
            for(i=0; i<40; i++)
            {
                for(j=0; j<4; j++)
                {
                    USART_Send(RemoteAddress[i][j]);
                }
            }

            _delay_ms(100);
            Switchregconvert.u48i = Switch1;

            for(i=0; i<6; i++)
            {
                USART_Send(Switchregconvert.c[5-i]);
            }

            Switchregconvert.u48i = Switch2;

            for(i=0; i<6; i++)
            {
                USART_Send(Switchregconvert.c[5-i]);
            }

            Switchregconvert.u48i = Switch3;

            for(i=0; i<6; i++)
            {
                USART_Send(Switchregconvert.c[5-i]);
            }

            Switchregconvert.u48i = Switch4;

            for(i=0; i<6; i++)
            {
                USART_Send(Switchregconvert.c[5-i]);
            }
            break;

        case 0xB1 :                     // Read EEPROM Config
            // ToDo EEPROM
            break;

        case 0xB2 :                     // Write EEPROM Config
            // ToDo EEPROM
            break;

        case 0xB3 :
            configFlag = 0x00;
            break;
```

```
        default :
            break;
    }
    SREG = tmp_sreg;
}
```