```c
unsigned int PRN;                                        //Global memory location used to store "pseudo random numbers"
char Dimmer_control;

int main (void){
setup_HW;
wdt_enable(WDTO_250MS);                                  //Following a WD reset the PRN is re-initialised to 0xFFFF

config_sw1_and_sw2_for_PCI;                              //SW1 is not used

UCSR0B |= (1 << RXCIE0);                                 //Set up interrupt on key press
Dimmer_control = 1;
sei();

while(1){                                                //Infinite while loop

PRN = PRN_16bit_GEN (0);                                 //Generate a new PRN using the previous value as input
I2C_Tx_2_integers(PRN, (PRN<<1));                        //Display two "pseudo random numbers"
Timer_T0_10mS_delay_x_m(10);                             //Pause before repeating
wdr();}}                                                 //Reset the watchdog timer which avoids the possibility
                                                         //of a reset for another 250mS


/*****************************************************************************************************/
ISR(PCINT2_vect)
{if (switch_2_up)return; else while(1);}                 //If switch_2 is pressed put program execution on hold
                                                         //The watchdog timer will not be reset and will "time out"


/*****************************************************************************************************/
ISR(USART_RX_vect){
receiveChar();
I2C_Tx(1, 'Q', &Dimmer_control);}



/*Local version of subroutine "I2C_Tx()"

void I2C_Tx_local(char num_bytes, char mode, char* s){
waiting_for_I2C_master;                                  //Turn on I2C slave and await call from master
send_byte_with_Ack(num_bytes);                           //send data byte, request acknowledgement
send_byte_with_Ack(mode);
for (int m = 0;  m < num_bytes; m++){
if (m==num_bytes-1){send_byte_with_Nack(s[m]);}          //Last byte, no ackowledgement needed
else {send_byte_with_Ack(s[m]);}}
TWCR = (1 << TWINT);}                                    //Clear interrupt and close I2C slave*/
```