```
/* Generation of a PWM signal with 10KHz frequency und a variable
 * duty cyrcle using PIC 18F25k22.
 * The percentage of time in which the PWM signal remains HIGH
 * is called as "duty cycle".
 * If the signal is always HIGH it is in 100% duty cycle and
 * if it is always LOW it is 0% duty cycle.
 * The frequency of a PWM signal determines how fast it completes one
period.
 * One Period is a complete ON and OFF of a PWM signal.
 * ECCP1 selected but as standard PWM module configured.
 * Fosc=8MHz intern/stable; PWM_freq = 10kHz:
 * Timer 2 resourse selected; TMR2prescale=1
 * PWMperiod=(PR2+1)*4*Tosc*TMR2prescale=(PR2+1)*4*0,125*1=100µsec
 * Period counter: PR2=[Fosc/(PWM_freq*4*TMR2prescale)]-1 = 199
 * Duty cyrcle=(ADC_read/1023)*(Period counter+1)
 * So, load MSB 8-bits of Duty to the CCPR1L = 0b00101000
 * and 2 LSB bits in CCP1CON <5:4>; CCP1CON <5:4> = 0b00.
 * NOTE 1: CCPR1L (duty cycle) value should be always less than or equal
 * to the PR2 (period) value. If is greater than the PR2, the pin CCP1
 * will not be cleared and allows a duty cycle of 100% at the output.
 * NOTE 2: If PR2 value is exceeding 8-bit (i.e. 255)
 * then we have to increase Timer2 pre-scale value.
 * File: PWM_variable.c
 * Author: Lasaros Goumas
 * Created on 27. November 2021, 14:53
 */


/* Includes

**********************************************************************
**/
#include <xc.h>
#include <pic18f25K22.h>


/*Configuration

**********************************************************************
***/
#pragma config FOSC = INTIO67
#pragma config PWRTEN = ON
#pragma config WDTEN = OFF
#pragma config BOREN = ON
#pragma config CCP2MX =PORTC1
#pragma config LVP = ON
#pragma config CP1 = OFF
#pragma config CPB = OFF
#pragma config WRT0 = OFF
#pragma config WRTC = OFF
#pragma config EBTR0 = OFF
#pragma config EBTRB = OFF


/*Declarations

**********************************************************************
```

```
**/
#define _XTAL_FREQ 8000000      // Fosc frequency for _delay() library
#define TMR2prescale 1
long PWM_freq = 10000;          //PWMperiod = 100µsec
unsigned int ADC_read;
unsigned int duty_cyrcle;



/*Funktionen

*****************************************************************************
/
void init_PIC (void){
    TRISA = 0b00000001;         //AN0 input
    ANSELA = 0x01;              //AN0 analoge Eingang
    ANSELCbits.ANSC2 =0x00;     //PIN RC2 is Digital
    TRISCbits.TRISC2 = 0;       //Pin RC2 is PWM output
    OSCCON = 0b01100111;        //8MHz; Internal Oscilator; stable
    CCPTMRS0bits.C1TSEL = 0b00; //Timer 2 resourse selected (CCP1 Modul)
    T2CONbits.TMR2ON = 0x00;    //Timer 2 "OFF"
    T2CONbits.T2CKPS = 0b00;    //Timer 2 PRESCALE = 1;
    PIR1bits.TMR2IF = 0x00;    //TMR2 to PR2 interrupt flag cleared
    TMR2 = 0x00;                //Clear Timer 2
}


void init_ADC (void){
     ADCON2 = 0b10010101;            //Right justified; Fosc/16; 4 TAD
    ADCON0bits.ADON = 0;       //ADC depowered
}


void ADC_value (void){
    ADCON0 = 0b00000001;             //AN0 selected; ADC Powered
    __delay_us(20);                 //20µsec warten
    ADCON0bits.GO = 0x01;           //ADC in progress
    while(ADCON0bits.NOT_DONE);
    ADC_read = ADRES;               //ADC Spannungsergebnis
    ADCON0bits.ADON=0;              //ADC depowered
}


void init_PWM (void){
    PR2 = (_XTAL_FREQ/(PWM_freq*4*TMR2prescale)) - 1;
}


void PWM_duty (void){
    duty_cyrcle = (ADC_read/4)*(PR2+1)/(1023/4);
    CCPR1L = duty_cyrcle;
    CCP1CON = 0x0C;                 //Load PWM duty cyrcle
    T2CONbits.TMR2ON = 0x01;        //Timer 2 enabled
}


/*Main Routine
```

```
**************************************************************************
/
void main(void) {
    init_PIC ();
    init_ADC();
    init_PWM();                 //Set PWM to work on 10KHZ

    do
    {
    ADC_value();            //Read the analog input voltage
    PWM_duty();             //Set the duty cyrcle
    __delay_ms(50);         //Warte 50msec
    }
    while(1);                       //Infinite Loop
}
```