```
#include <16F877A.h>
#device ADC=16
#FUSES NOBROWNOUT               //No brownout reset
#FUSES NOLVP                 //No low voltage prgming, B3(PIC16)
or B5(PIC18) used for I/O
#use delay(crystal=20000000)
#define LCD_RS_PIN      PIN_D1
#define LCD_RW_PIN      PIN_D2
#define LCD_ENABLE_PIN  PIN_D3
#define LCD_DATA4       PIN_D4
#define LCD_DATA5       PIN_D5
#define LCD_DATA6       PIN_D6
#define LCD_DATA7       PIN_D7
#include <lcd.c>

int32 ovrflow=0;
#byte TMRO=0x01
#byte OPREG=0x81
#bit OPREG4=OPREG.4
#byte INTCON=0x0B
#byte PIE1=0x8C
#byte T1CON=0x10
#bit T1CON0=T1CON.0
#byte TMR1L=0x0E
#byte TMR1H=0x0F
#byte TRISA=0x85
#byte TRISD=0x88
```

```
#int_timer1
void ngat()
{
   ovrflow+=1;
   TMR1L=0;
   TMR1H=0;
}
void main()
{
   TRISA=0xff;
   TRISD=0x00;
   lcd_init();
   lcd_putc('\f');

   INTCON=0b11000000;
   PIE1=0b00000001;
   T1CON=0;
   TMR1L=0;
   TMR1H=0;
   OPREG=0b00101000; //low-to-high transition initially
   TMRO=0;

   int done=0;
   int32 valueT1,valueT2;
   float freq;
   while(TRUE)
   {
      while(TMRO<2);
      T1CON0=1;   //turn on timer1
      OPREG4=1;   //high-to-low transition
      while(TMRO<3);
      valueT1=make16(TMR1H,TMR1L)+ovrflow*65536;
      OPREG4=0;   //low-to-high transition
      while(TMRO<4);
      T1CON0=0;   //turn off timer1
      valueT2=make16(TMR1H,TMR1L)+ ovrflow*65536;
      freq=5000000.0/valueT2;
      if(done==0)
      {
         lcd_gotoxy(1,1);
         printf(lcd_putc,"FREQ:%fHz",freq);
         lcd_gotoxy(1,2);

printf(lcd_putc,"HIGH:%lf%%",((double)valueT1/valueT2)*100);
         done=1;
      }
      ovrflow=0;
   }
}
```

1.1. Program explanation

```
#include <16F877A.h>
```

```
#device ADC=16
#FUSES NOWDT                    //No Watch Dog Timer
#FUSES NOBROWNOUT               //No brown out reset
#FUSES NOLVP //No low voltage programming, B3(PIC16) or B5(PIC18)
used for I/O
#use delay(crystal=20000000)
```

#include <16F877A.h>: Use header file 16F877A.h to determine the constants, the register, and other fundamental functions for MCU PIC16F877A.
#device ADC=16: Set up 16 bits for the output bit of the ADC module.
#FUSES NOBROWNOUT: No brown out reset.
#FUSES NOLVP: No low voltage programming, B3(PIC16) or B5(PIC18) used for I/O.
#use delay(crystal=20000000): Set up the 20 MHz crystal frequency that we use for the MCU.

```
#define LCD_RS_PIN      PIN_D1
#define LCD_RW_PIN      PIN_D2
#define LCD_ENABLE_PIN  PIN_D3
#define LCD_DATA4       PIN_D4
#define LCD_DATA5       PIN_D5
#define LCD_DATA6       PIN_D6
#define LCD_DATA7       PIN_D7
#include <lcd.c>
```

Define the connection pins between the MCU and LCD. In this exercise, we just need to determine 4 pins RS, RW, ENABLE of the LCD as we are using the 4-bit interface.
#include <lcd.c>: This line includes the lcd.c file, which contains the fundamental functions and definitions to communicate with the LCD.

```
int32 ovrflow=0;
```

```
#byte TMRO=0x01

#byte OPREG=0x81

#bit OPREG4=OPREG.4

#byte INTCON=0x0B

#byte PIE1=0x8C

#byte T1CON=0x10

#bit T1CON0=T1CON.0

#byte TMR1L=0x0E

#byte TMR1H=0x0F

#byte TRISA=0x85

#byte TRISD=0x88
```

int32 ovrflow = 0: Define the variable used to count Timer1 overflows.

#byte: Defines a byte variable at a specific memory address.

#bit: Defines a bit variable at a specific bit within a byte.

Address	Name	Description
01h	TMR0	Timer0 Module Register, use Timer 0
81h	OPREG	The OPTION_REG Register is a readable and writable register, which contains various control bits to configure the TMR0 prescaler
0Bh	INTCON	The INTCON register is a readable and writable register, which contains various enable and flag bits for the TMR0 register overflow
8Ch	PIE1	The PIE1 register contains the individual enable bits for the peripheral interrupts.
10h	T1CON	Timer1 Operation in Timer Mode Timer mode is selected by clearing the TMR1CS (T1CON<1>) bit. In this mode, the input clock to the timer is *the crystal clock's frequency/4.*
0Eh	TMR1L	The Timer1 module is a 16-bit timer/counter consisting

0Fh	TMR1H	of two 8-bit registers (TMR1H and TMR1L) which are readable and writable.
85h	TRISA	PORTA Data Direction Register
88h	TRISD	PORTD Data Direction Register

```
#int_timer1
void ngat()
{
   ovrflow+=1;
   TMR1L=0;
   TMR1H=0;
}
```

#int_timer1: Defines an interrupt function called when there's an interrupt from Timer1. In the ngat() function, the ovrflow variable is incremented to 1 as Timer1 overflows, then Timer1 is reset to 0.

```
void main()
{
   TRISA=0xff;
   TRISD=0x00;
   lcd_init();
   lcd_putc('\f');

   INTCON=0b11000000;
   PIE1=0b00000001;
   T1CON=0;
   TMR1L=0;
   TMR1H=0;
   OPREG=0b00101000; //low-to-high transition initially
   TMRO=0;

   int done=0;
   int32 valueT1,valueT2;
   float freq;

   while(TRUE)
   {
      while(TMRO<2);
      T1CON0=1;   //turn on timer1
      OPREG4=1;   //high-to-low transition
      while(TMRO<3);
      valueT1=make16(TMR1H,TMR1L)+ovrflow*65536;
      OPREG4=0;   //low-to-high transition
      while(TMRO<4);
      T1CON0=0;   //turn off timer1
      valueT2=make16(TMR1H,TMR1L)+ ovrflow*65536;
      freq=5000000.0/valueT2;
      if(done==0)
      {
         lcd_gotoxy(1,1);
         printf(lcd_putc,"FREQ:%fHz",freq);
         lcd_gotoxy(1,2);

printf(lcd_putc,"HIGH:%lf%%",((double)valueT1/valueT2)*100);
         done=1;
```

```
        }
        ovrflow=0;
    }
}
```

PORTA is a 6-bit wide, bidirectional port. The corresponding data direction register is TRISA. Setting a TRISA bit (= 1) will make the corresponding PORTA pin an input. Clearing a TRISA bit (= 0) will make the corresponding PORTA pin an output.

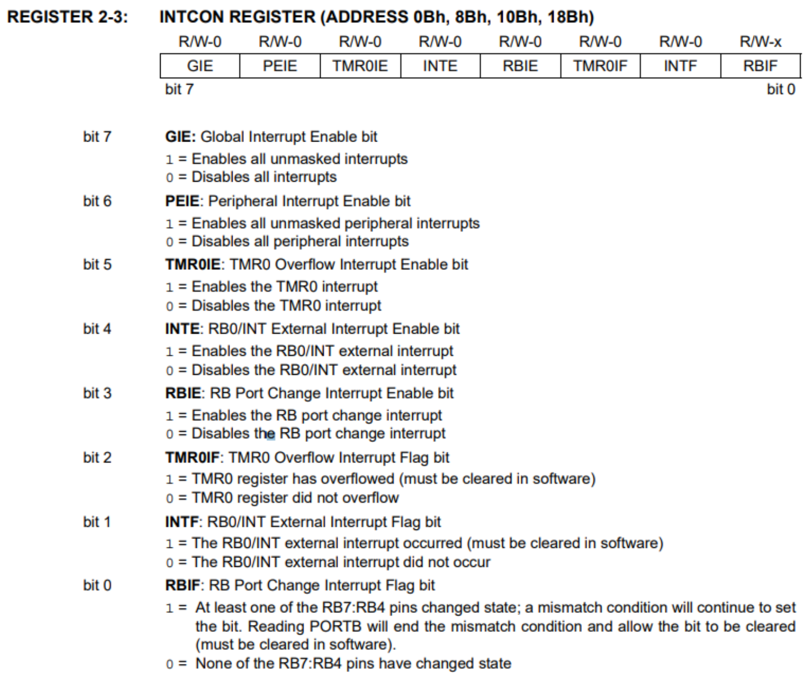
TRISA=0xff: 0xff = 0b11111111 which turns all PORTA pin into inputs, for pulse input

TRISD=0x00: 0x00 = 0b00000000; which turns all PORTD pin into outputs, for LCD output

lcd_init(): Set up the LCD

lcd_putc('\f'): Print the blank space after activating the LCD

INTCON=0b11000000: Turn on bit 7 and 6 to of INTCON register to 1, which will enable all unmasked interrupts



REGISTER 2-3: **INTCON REGISTER (ADDRESS 0Bh, 8Bh, 10Bh, 18Bh)**

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-x
GIE	PEIE	TMR0IE	INTE	RBIE	TMR0IF	INTF	RBIF
bit 7							bit 0

bit 7 **GIE:** Global Interrupt Enable bit
1 = Enables all unmasked interrupts
0 = Disables all interrupts

bit 6 **PEIE**: Peripheral Interrupt Enable bit
1 = Enables all unmasked peripheral interrupts
0 = Disables all peripheral interrupts

bit 5 **TMR0IE**: TMR0 Overflow Interrupt Enable bit
1 = Enables the TMR0 interrupt
0 = Disables the TMR0 interrupt

bit 4 **INTE**: RB0/INT External Interrupt Enable bit
1 = Enables the RB0/INT external interrupt
0 = Disables the RB0/INT external interrupt

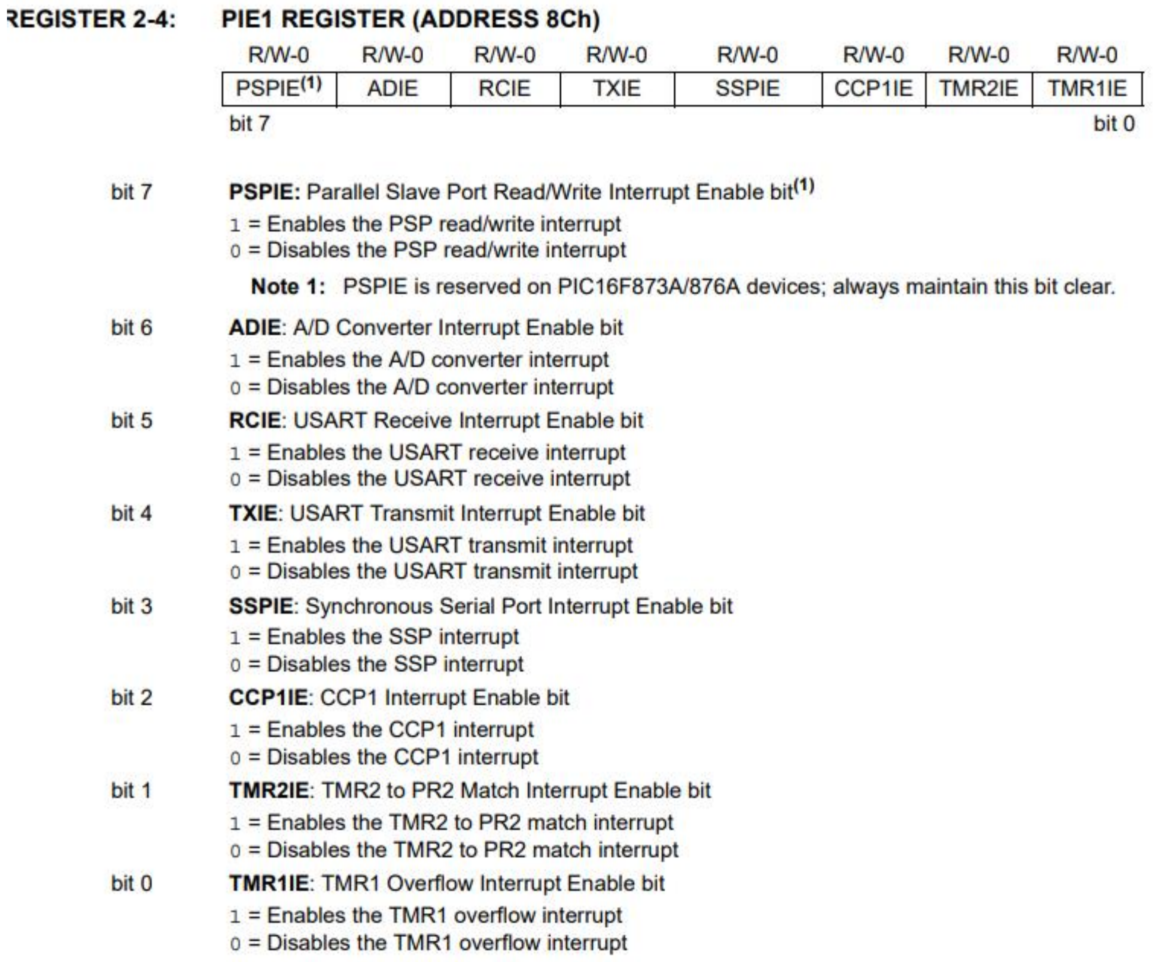
bit 3 **RBIE**: RB Port Change Interrupt Enable bit
1 = Enables the RB port change interrupt
0 = Disables the RB port change interrupt

bit 2 **TMR0IF**: TMR0 Overflow Interrupt Flag bit
1 = TMR0 register has overflowed (must be cleared in software)
0 = TMR0 register did not overflow

bit 1 **INTF**: RB0/INT External Interrupt Flag bit
1 = The RB0/INT external interrupt occurred (must be cleared in software)
0 = The RB0/INT external interrupt did not occur

bit 0 **RBIF**: RB Port Change Interrupt Flag bit
1 = At least one of the RB7:RB4 pins changed state; a mismatch condition will continue to set the bit. Reading PORTB will end the mismatch condition and allow the bit to be cleared (must be cleared in software).
0 = None of the RB7:RB4 pins have changed state

PIE1=0b00000001: Turn on bit 0 of PIE1 register, which enables the TMR1 overflow interrupt



REGISTER 2-4: PIE1 REGISTER (ADDRESS 8Ch)

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
PSPIE(1)	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE
bit 7							bit 0

bit 7 **PSPIE:** Parallel Slave Port Read/Write Interrupt Enable bit(1)
1 = Enables the PSP read/write interrupt
0 = Disables the PSP read/write interrupt

Note 1: PSPIE is reserved on PIC16F873A/876A devices; always maintain this bit clear.

bit 6 **ADIE**: A/D Converter Interrupt Enable bit
1 = Enables the A/D converter interrupt
0 = Disables the A/D converter interrupt

bit 5 **RCIE**: USART Receive Interrupt Enable bit
1 = Enables the USART receive interrupt
0 = Disables the USART receive interrupt

bit 4 **TXIE**: USART Transmit Interrupt Enable bit
1 = Enables the USART transmit interrupt
0 = Disables the USART transmit interrupt

bit 3 **SSPIE**: Synchronous Serial Port Interrupt Enable bit
1 = Enables the SSP interrupt
0 = Disables the SSP interrupt

bit 2 **CCP1IE**: CCP1 Interrupt Enable bit
1 = Enables the CCP1 interrupt
0 = Disables the CCP1 interrupt

bit 1 **TMR2IE**: TMR2 to PR2 Match Interrupt Enable bit
1 = Enables the TMR2 to PR2 match interrupt
0 = Disables the TMR2 to PR2 match interrupt

bit 0 **TMR1IE**: TMR1 Overflow Interrupt Enable bit
1 = Enables the TMR1 overflow interrupt
0 = Disables the TMR1 overflow interrupt

T1CON=0: Turn off the timer 1

TMR1L=0, TMR1H=0:Clear all data in TMR1L and TMR1H registers.

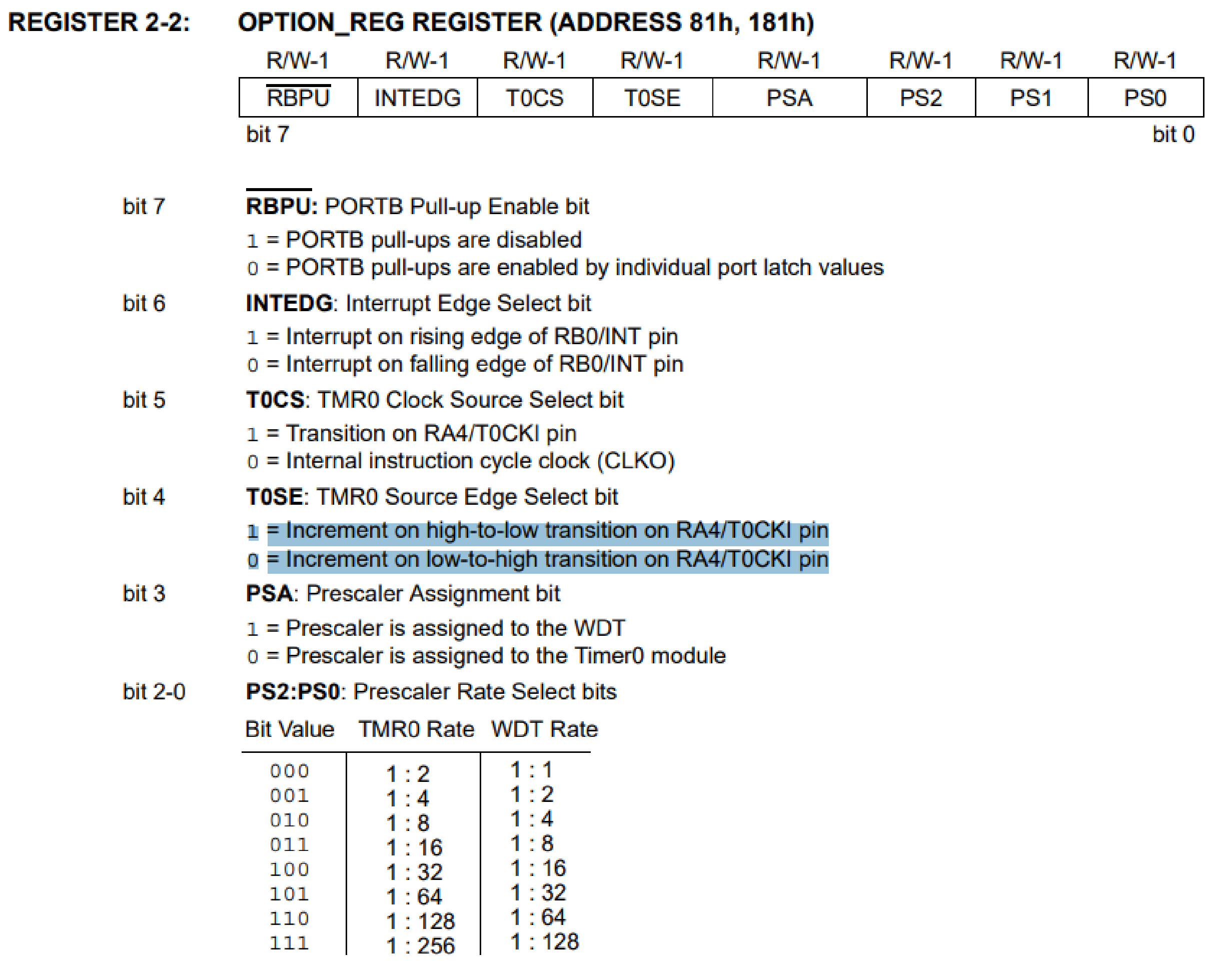
OPREG=0b00101000: Setting the working condition for timer 0, which uses timer0 as counter, detect transition on T0CKI, increment on low to high transition and 1:1 WDT prescaler.

TMRO=0; int done=0; int32 valueT1,valueT2; float freq: Declare the variable used for the loop, value1 stores the high time of the pulse, whereas value2 stores the cycle time of the pulse. The freq variable stores the value of the frequency of the pulse.

```
while(TRUE)
   {
      while(TMRO<2);
      T1CON0=1;   //turn on timer1
      OPREG4=1;   //high-to-low transition
      while(TMRO<3);
      valueT1=make16(TMR1H,TMR1L)+ovrflow*65536;
      OPREG4=0;   //low-to-high transition
      while(TMRO<4);
      T1CON0=0;   //turn off timer1
      valueT2=make16(TMR1H,TMR1L)+ ovrflow*65536;
      freq=5000000.0/valueT2;
      if(done==0)
      {
         lcd_gotoxy(1,1);
         printf(lcd_putc,"FREQ:%fHz",freq);
         lcd_gotoxy(1,2);

printf(lcd_putc,"HIGH:%lf%%",((double)valueT1/valueT2)*100);
         done=1;
      }
      ovrflow=0;
   }
}
```

This while loop is the main part of the program where the frequency and high mode cycle of the input signal are measured continuously and displayed on the LCD screen.



REGISTER 2-2: OPTION_REG REGISTER (ADDRESS 81h, 181h)

R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
$\overline{\text{RBPU}}$	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0
bit 7							bit 0

bit 7 **$\overline{\text{RBPU}}$:** PORTB Pull-up Enable bit
1 = PORTB pull-ups are disabled
0 = PORTB pull-ups are enabled by individual port latch values

bit 6 **INTEDG**: Interrupt Edge Select bit
1 = Interrupt on rising edge of RB0/INT pin
0 = Interrupt on falling edge of RB0/INT pin

bit 5 **T0CS**: TMR0 Clock Source Select bit
1 = Transition on RA4/T0CKI pin
0 = Internal instruction cycle clock (CLKO)

bit 4 **T0SE**: TMR0 Source Edge Select bit
1 = Increment on high-to-low transition on RA4/T0CKI pin
0 = Increment on low-to-high transition on RA4/T0CKI pin

bit 3 **PSA**: Prescaler Assignment bit
1 = Prescaler is assigned to the WDT
0 = Prescaler is assigned to the Timer0 module

bit 2-0 **PS2:PS0**: Prescaler Rate Select bits

Bit Value	TMR0 Rate	WDT Rate
000	1 : 2	1 : 1
001	1 : 4	1 : 2
010	1 : 8	1 : 4
011	1 : 16	1 : 8
100	1 : 32	1 : 16
101	1 : 64	1 : 32
110	1 : 128	1 : 64
111	1 : 256	1 : 128

We add 1 to the value of TMR0 when there is a high to low or low to high signal. Initially, the transition that we set for timer0 to detect is low-to-high. When the pulse changes from low to high for the second time, we do not do anything. But from the third time, when the pulse changes from low to high, when the pulse is at high voltage, the timer1 will start counting with T1CON0=1.

After that, the detection of pulse edge of timer 0 is changed to detect from high to low with the change of OPTION_REG bit 4 OPREG4=1. So when the pulse changes from high to low, it will increase TMRO value by 1 value, when that happens, the time is then recorded. This is the high time of the pulse.

The value of valueT1 variable determines the high time of the pulse: valueT1= make16(TMR1H,TMR1L)+ovrflow*65536.

The detection of pulse edge of timer 0 is changed to detect from low to high again with the change of OPREG bit 4 OPREG4=0. Next, when the pulse changes from low to high, it will stop the timer and read the value in TMR1L and TMR1H registers. The value of valueT2 variable determines the time for one cycle of the pulse: valueT2=make16(TMR1H,TMR1L)+ ovrflow*65536. The make16 function helps combine two 8 bit values into a single 16 bit value. As the timer1 may overflow during the measurement, we must add the overflow variable to it to handle pulses with low frequency that may make the timer1 overflow.
freq=5000000.0/valueT2: Because the prescaler we are using is 1, and the frequency that enters the timer1 is divided by 4, we must take 20/4=5MHz to divide valueT2 to find the frequency of the pulse

```
if(done==0)
  {
lcd_gotoxy(1,1);
printf(lcd_putc,"FREQ:%fHz",freq);
lcd_gotoxy(1,2);
printf(lcd_putc,"HIGH:%lf%%",((double)valueT1/valueT2)*100);
done=1;
}
 ovrflow=0;
}
```

These lines of codes are used to print the value of the frequency, and the percentage of high time of pulse to the LCD. The done variable ensures that the values are only printed once on the LCD. The high time can be calculated by divide valueT1/valueT2*100%.