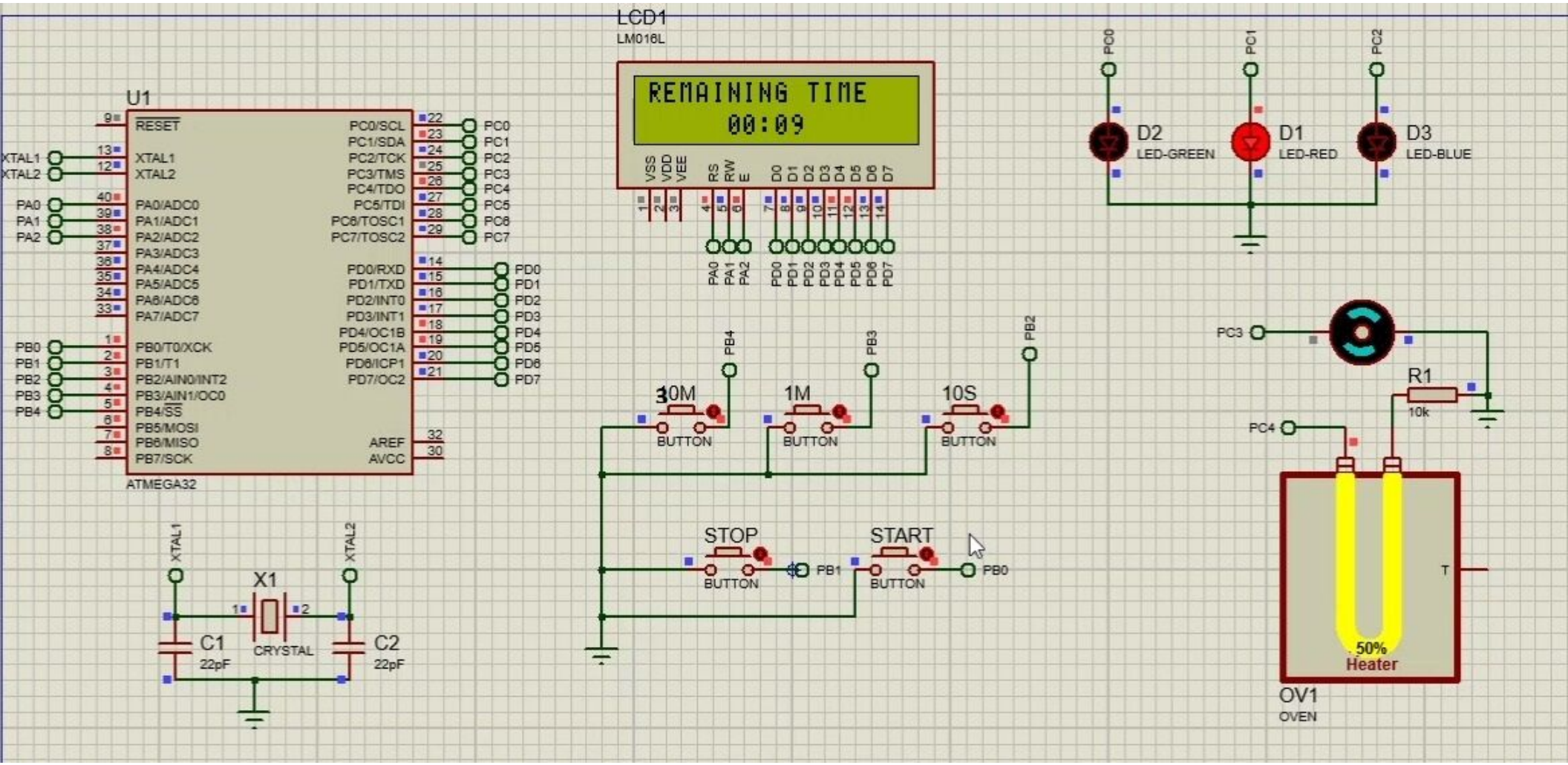
U1
RESET
XTAL1
XTAL2
PA0/ADC0
PA1/ADC1
PA2/ADC2
PA3/ADC3
PA4/ADC4
PA5/ADC5
PA6/ADC6
PA7/ADC7
PB0/T0/XCK
PB1/T1
PB2/AIN0/INT2
PB3/AIN1/OC0
PB4/SS
PB5/MOSI
PB6/MISO
PB7/SCK
PC0/SCL
PC1/SDA
PC2/TCK
PC3/TMS
PC4/TDO
PC5/TDI
PC6/TOSC1
PC7/TOSC2
PD0/RXD
PD1/TXD
PD2/INT0
PD3/INT1
PD4/OC1B
PD5/OC1A
PD6/ICP1
PD7/OC2
AREF
AVCC
ATMEGA32
XTAL1
XTAL2
X1
CRYSTAL
C1
22pF
C2
22pF
LCD1
LM016L
REMAINING TIME
00:09
VSS
VDD
VEE
RS
RW
E
D0
D1
D2
D3
D4
D5
D6
D7
PA0
PA1
PA2
PD0
PD1
PD2
PD3
PD4
PD5
PD6
PD7
PB4
PB3
PB2
10M
1M
10S
BUTTON
STOP
START
PB1
PB0
PC0
PC1
PC2
D2
LED-GREEN
D1
LED-RED
D3
LED-BLUE
PC3
R1
10k
PC4
50%
Heater
T
OV1
OVEN

```
#include <avr/io.h>
#include "avr/interrupt.h"
#include "avr/delay.h"
#include "LCD_interface.h"
#include "DIO_int.h"
#include "Timer.h"
#include "microwave_int.h"

#define INIT_TICK 178

extern uint32_t Tick = 0;

void init_system();


int main(void)
{

      init_system();

    while (1)
    {
           state();
    }
}


void init_system()
{
      DIO_VidSetPortDirection(PORTD, OUTPUT);
      DIO_VidSetPortDirection(PORTA, OUTPUT);
      DIO_VidSetPortDirection(PORTC, OUTPUT);
      DIO_VidSetPortDirection(PORTB, INPUT);
      DIO_VidSetPortValue(PORTB, HIGH);

      LCD_vidInit();

      /* Generate interrupt every 10ms */
      TIM0_OVF_Init(TIMER_PRESCALER_1024, INIT_TICK);
      state = IDLE;
}


/* Generate interrupt every 10ms */
ISR(TIMER0_OVF_vect)
{
      Tick++;
      TCNT0 = INIT_TICK;
}
```

```
#include "stdint.h"
#include "LCD_interface.h"
#include "DIO_int.h"
#include "microwave_int.h"

uint32_t coock_time = 0;
uint32_t start_heat_time = 0;
uint32_t Rtime = 0;
uint32_t pause_time = 0;
uint32_t off_time = 0;
extern uint32_t Tick;

void IDLE()
{
	if ( last_state != IDLE)
	{
		DIO_VidSetPinValue(PORTC, MOTOR_PIN, LOW);
		DIO_VidSetPinValue(PORTC, HEATER_PIN, LOW);
		DIO_VidSetPinValue(PORTC, RED_PIN, LOW);
		DIO_VidSetPinValue(PORTC, YELLOW_PIN, LOW);
		DIO_VidSetPinValue(PORTC, GREEN_PIN, HIGH);

		LCD_vidSendCommand(lcd_Clear);
		LCD_vidWriteString("TIME MM:SS", 10);
		Gotoxy(1,5);
		LCD_vidWriteString("00:00", 5);
		last_state = IDLE;

	}


	if (!DIO_u8GetPinValue(PORTB, START_PIN) && coock_time > 0)
	{
		state = HEAT;
		last_state = IDLE;
	}
	else if (!DIO_u8GetPinValue(PORTB, SS_PIN))
	{
		while(!DIO_u8GetPinValue(PORTB, SS_PIN));
		coock_time += 1000;

	}
	else if (!DIO_u8GetPinValue(PORTB, MM_PIN))
	{
		while(!DIO_u8GetPinValue(PORTB, MM_PIN));
		coock_time += 6000;
	}
	else if (!DIO_u8GetPinValue(PORTB, MM10_PIN))
	{
		while(!DIO_u8GetPinValue(PORTB, MM10_PIN));
		coock_time += 60000;
	}
	else if (!DIO_u8GetPinValue(PORTB, STOP_PIN))
	{
		while(!DIO_u8GetPinValue(PORTB, STOP_PIN));
		coock_time = 0;
		Gotoxy(1,5);
		LCD_vidWriteString("00:00", 5);
	}
```

```
	if (coock_time/6000 >= 10)
	{
		Gotoxy(1,5);
		LCD_vidWriteNumber(coock_time/6000);
	}
	else
	{
		Gotoxy(1,6);
		LCD_vidWriteNumber(coock_time/6000);
	}

	if (coock_time >= 6000)
	{
		Gotoxy(1,8);
		LCD_vidWriteNumber((coock_time%6000)/100);
	}
	else
	{
		Gotoxy(1,8);
		LCD_vidWriteNumber(coock_time/100);
	}



}



void HEAT()
{
	if (last_state == IDLE)
	{
		start_heat_time = Tick;
	}
	else if (last_state == PAUSE)
	{
		off_time = Tick - pause_time;
	}

	if ( last_state != HEAT)
	{
		DIO_VidSetPinValue(PORTC, MOTOR_PIN, HIGH);
		DIO_VidSetPinValue(PORTC, HEATER_PIN, HIGH);
		DIO_VidSetPinValue(PORTC, RED_PIN, HIGH);
		DIO_VidSetPinValue(PORTC, YELLOW_PIN, LOW);
		DIO_VidSetPinValue(PORTC, GREEN_PIN, LOW);

		LCD_vidSendCommand(lcd_Clear);
		Gotoxy(0, 0);
		LCD_vidWriteString("REMAINING TIME", 14);
		Gotoxy(1,5);
		LCD_vidWriteString("00:00", 5);
		last_state = HEAT;
	}

	Rtime = off_time + (coock_time + start_heat_time) - Tick;

	if (Rtime == 0)
	{
		coock_time = 0;
```

```
            Rtime = 0;
            state = IDLE;
      }
      else if (!DIO_u8GetPinValue(PORTB, STOP_PIN))
      {
            while(!DIO_u8GetPinValue(PORTB, STOP_PIN));
            state = PAUSE;
      }


      print_time(Rtime);


}


void PAUSE()
{
      if ( last_state != PAUSE)
      {
            pause_time = Tick;
            DIO_VidSetPinValue(PORTC, MOTOR_PIN, LOW);
            DIO_VidSetPinValue(PORTC, HEATER_PIN, LOW);
            DIO_VidSetPinValue(PORTC, RED_PIN, LOW);
            DIO_VidSetPinValue(PORTC, YELLOW_PIN, HIGH);
            DIO_VidSetPinValue(PORTC, GREEN_PIN, LOW);

            LCD_vidSendCommand(lcd_Clear);
            LCD_vidWriteString("PAUSE", 5);
            Gotoxy(1,5);
            LCD_vidWriteString("00:00", 5);
            last_state = PAUSE;
      }

      if (!DIO_u8GetPinValue(PORTB, START_PIN))
      {
            while(!DIO_u8GetPinValue(PORTB, START_PIN));
            state = HEAT;
      }
      else if (!DIO_u8GetPinValue(PORTB, STOP_PIN))
      {
            while(!DIO_u8GetPinValue(PORTB, STOP_PIN));
            coock_time = 0;
            state = IDLE;
      }

      print_time(Rtime);

}

void print_time(uint32_t time)
{
      if (time/6000 >= 10)
      {
            Gotoxy(1,5);
            LCD_vidWriteNumber(time/6000);
      }
      else
      {
            Gotoxy(1,6);
```

```
            LCD_vidWriteNumber(time/6000);
      }

      if (time >= 6000)
      {
            if ((time%6000)/100 >= 10)
            {
                  Gotoxy(1,8);
                  LCD_vidWriteNumber((time%6000)/100);
            }
            else
            {
                  Gotoxy(1,8);
                  LCD_vidWriteNumber(0);
                  Gotoxy(1,9);
                  LCD_vidWriteNumber((time%6000)/100);
            }
      }
      else if (time/100 >= 10)
      {
            Gotoxy(1,8);
            LCD_vidWriteNumber(time/100);
      }
      else
      {
            Gotoxy(1,8);
            LCD_vidWriteNumber(0);
            Gotoxy(1,9);
            LCD_vidWriteNumber(time/100);
      }

}
```

```
#include <avr/interrupt.h>
#include <avr/io.h>
#include <util/delay.h>
#include "std_macros.h"
#include "Timer.h"


void TIM0_CTC_Init(uint8_t prescaler, uint8_t Init_Ticks)
{

	/* set ctc mode */
	CLR_BIT(TCCR0,WGM00);
	SET_BIT(TCCR0,WGM01);

	/* set timer count */
	OCR0 = Init_Ticks ; // Example: 256 or 1024


	/* set  pin OC0 mode */
	CLR_BIT(TCCR0,COM00);
	CLR_BIT(TCCR0,COM01);


	/* set prescaller */

	switch (prescaler)
	{
		case 1 :
		SET_BIT(TCCR0,CS00);
		CLR_BIT(TCCR0,CS01);
		CLR_BIT(TCCR0,CS02);
		break;
		case 2 :
		CLR_BIT(TCCR0,CS00);
		SET_BIT(TCCR0,CS01);
		CLR_BIT(TCCR0,CS02);
		break;
		case 3 :
		SET_BIT(TCCR0,CS00);
		SET_BIT(TCCR0,CS01);
		CLR_BIT(TCCR0,CS02);
		break;

		case 4 :
		CLR_BIT(TCCR0,CS00);
		CLR_BIT(TCCR0,CS01);
		SET_BIT(TCCR0,CS02);
		break;

		case 5 :
		SET_BIT(TCCR0,CS00);
		CLR_BIT(TCCR0,CS01);
		SET_BIT(TCCR0,CS02);
		break;
	}

	SET_BIT(TIMSK,OCIE0);
	sei();
}
```

```
void TIM0_OVF_Init(uint8_t prescaler, uint8_t Init_Ticks)
{

	/* set normal mode */
	CLR_BIT(TCCR0,WGM00);
	CLR_BIT(TCCR0,WGM01);

	/* set timer count */
	TCNT0 = Init_Ticks ;

	/* set prescaller */

	switch (prescaler)
	{
		case 1 :
		SET_BIT(TCCR0,CS00);
		CLR_BIT(TCCR0,CS01);
		CLR_BIT(TCCR0,CS02);
		break;

		case 2 :
		CLR_BIT(TCCR0,CS00);
		SET_BIT(TCCR0,CS01);
		CLR_BIT(TCCR0,CS02);
		break;

		case 3 :
		SET_BIT(TCCR0,CS00);
		SET_BIT(TCCR0,CS01);
		CLR_BIT(TCCR0,CS02);
		break;

		case 4 :
		CLR_BIT(TCCR0,CS00);
		CLR_BIT(TCCR0,CS01);
		SET_BIT(TCCR0,CS02);
		break;

		case 5 :
		SET_BIT(TCCR0,CS00);
		CLR_BIT(TCCR0,CS01);
		SET_BIT(TCCR0,CS02);
		break;
	}

	SET_BIT(TIMSK,TOIE0);
	sei();
}

void TIM0_Stop()
{
	TCCR0 = 0;
}


void TIM1_OVF_Init(uint8_t prescaler, uint16_t Init_Ticks)
{
	/* set normal mode */
	CLR_BIT(TCCR1A,WGM00);
```

```
	CLR_BIT(TCCR1A,WGM01);

	/* set timer count */
	TCNT1 = Init_Ticks ;

	/* set prescaller */

	switch (prescaler)
	{
		case 1 :
		SET_BIT(TCCR1B,CS10);
		CLR_BIT(TCCR1B,CS11);
		CLR_BIT(TCCR1B,CS12);
		break;

		case 2 :
		CLR_BIT(TCCR1B,CS00);
		SET_BIT(TCCR1B,CS11);
		CLR_BIT(TCCR1B,CS12);
		break;

		case 3 :
		SET_BIT(TCCR1B,CS00);
		SET_BIT(TCCR1B,CS11);
		CLR_BIT(TCCR1B,CS12);
		break;

		case 4 :
		CLR_BIT(TCCR1B,CS00);
		CLR_BIT(TCCR1B,CS11);
		SET_BIT(TCCR1B,CS12);
		break;

		case 5 :
		SET_BIT(TCCR1B,CS00);
		CLR_BIT(TCCR1B,CS11);
		SET_BIT(TCCR1B,CS12);
		break;
	}

	SET_BIT(TIMSK,TOIE1);
	sei();
}

void TIM1_CTC_Init(uint8_t prescaler, uint16_t Init_Ticks)
{
	/* set CTC mode */
	CLR_BIT(TCCR1A,WGM00);
	CLR_BIT(TCCR1A,WGM01);
	SET_BIT(TCCR1B,WGM12);
	CLR_BIT(TCCR1B,WGM13);

	/* set timer count */
	OCR1A = Init_Ticks ;

	/* set  pin OC0 mode */
	CLR_BIT(TCCR1A,COM1A0);
	CLR_BIT(TCCR1A,COM1A1);
```

```
	/* set prescaler */

	switch (prescaler)
	{
		case 1 :
		SET_BIT(TCCR1B,CS10);
		CLR_BIT(TCCR1B,CS11);
		CLR_BIT(TCCR1B,CS12);
		break;

		case 2 :
		CLR_BIT(TCCR1B,CS00);
		SET_BIT(TCCR1B,CS11);
		CLR_BIT(TCCR1B,CS12);
		break;

		case 3 :
		SET_BIT(TCCR1B,CS00);
		SET_BIT(TCCR1B,CS11);
		CLR_BIT(TCCR1B,CS12);
		break;

		case 4 :
		CLR_BIT(TCCR1B,CS00);
		CLR_BIT(TCCR1B,CS11);
		SET_BIT(TCCR1B,CS12);
		break;

		case 5 :
		SET_BIT(TCCR1B,CS00);
		CLR_BIT(TCCR1B,CS11);
		SET_BIT(TCCR1B,CS12);
		break;
	}

	SET_BIT(TIMSK,OCIE1A);
	sei();
}

void TIM1_Stop()
{
	TCCR1A = 0;
}
```