



**Министерство образования и науки Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего
образования
«Московский государственный технический университет
имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)**

ФАКУЛЬТЕТ _____ Информатика и системы управления

КАФЕДРА Программное обеспечение ЭВМ и информационные технологии

Отчет по лабораторной работе №3 по курсу: «Архитектура ЭВМ»

По теме: «Синхронизация микроконтроллера и управление таймерами»

Студент: Аминов Т.С
Группа: ИУ7-55Б

Преподаватель:
Попов А.Ю.

Москва, 2019 г.

Цель работы - Изучение системы синхронизации микроконтроллера NXP LPC2368 и принципов функционирования таймеров общего назначения. В ходе работы студенту необходимо ознакомиться с теоретическим материалом, касающимся системы синхронизации и таймеров, разработать и отладить программу функционирования микроконтроллера NXP LPC2368 с использованием отладочных плат SK-LPC2368 и TM1638LED&KEY.

Задание.

Вариант 1. Устройство прогрева двигателя внутреннего сгорания, включающее клапан подачи горючей смеси, устройство зажигания, стартер.

Программа функционирования:

- a) Пуск стартера, кратковременное открытие клапана горючей смеси на 0.05 секунды и зажигание при закрытом клапане оставшееся время такта (частота: 10 Гц);
- b) при нажатии на кнопку: отключение стартера;
- c) через 5 секунд после нажатия – отключение зажигания и закрытие клапана;
- d) отключение.

Частота внешнего генератора: 12 МГц.

Частота процессорного ядра: 24 МГц.

Частота синхронизации таймера: 12 МГц.

Расчет параметров таймера

Для корректной работы используемого в программе таймера были проведены необходимые расчеты параметров. По условию задания необходимо произвести задержки длительностью 5 с и 0.05 с. Частота процессорного ядра должна соответствовать 24 МГц.

$M = 40 = 0x0028 \text{ hex}$

$N = 2 = 0x0002 \text{ hex}$

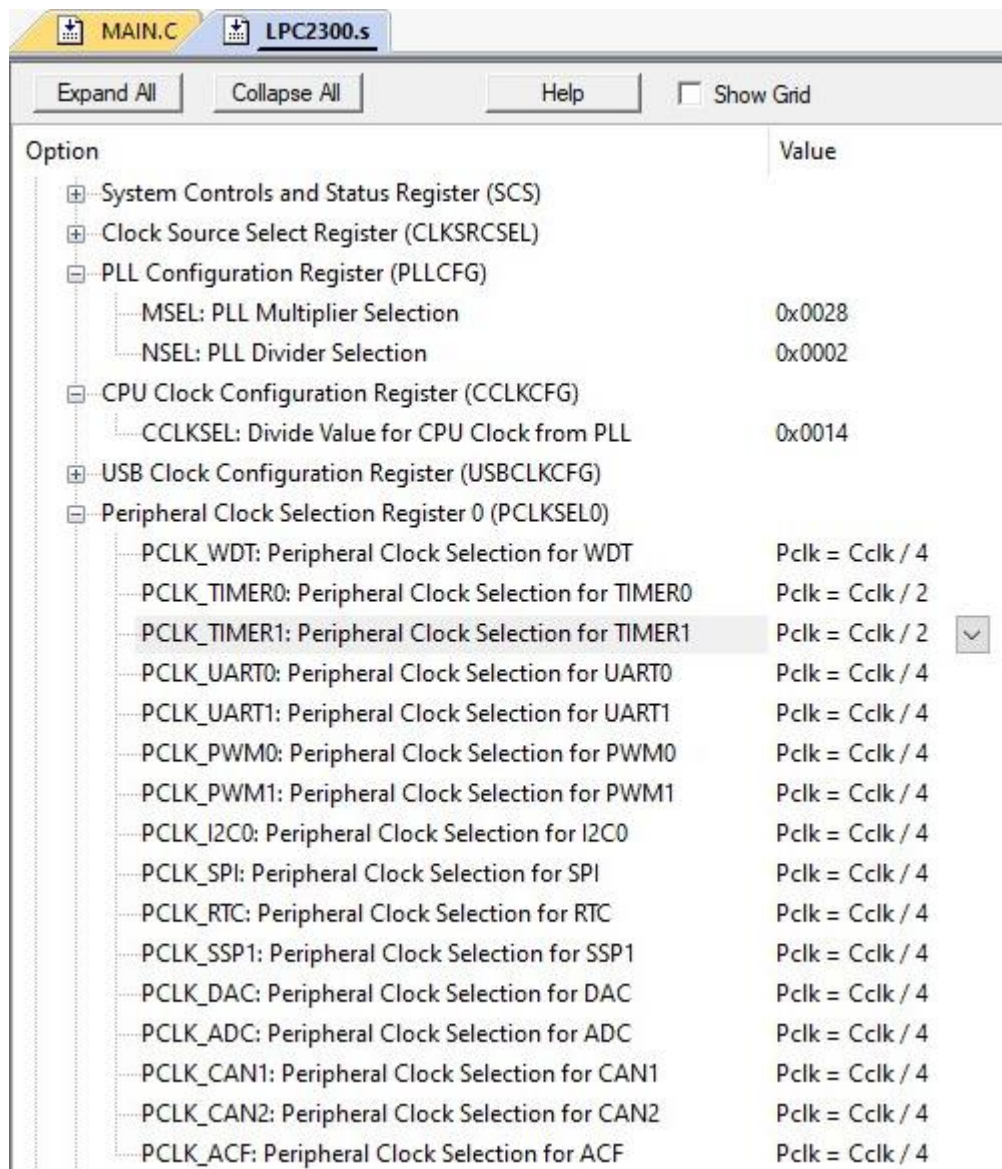
$CCLKSEL = 20 = 0x0014 \text{ hex}$

$Fin = 12 \text{ МГц}$ по условию задания, тогда $Fcco = \frac{2*Fin*M}{N} = 480 \text{ МГц}$

$Fcpu = \frac{Fcco}{CCLKSEL+1} = 24 \text{ МГц}$

Частота синхронизации таймера $= 12 \text{ МГц} = \frac{Fcpu}{2}$

Настройки в программе



Option	Value
System Controls and Status Register (SCS)	
Clock Source Select Register (CLKSRCSEL)	
PLL Configuration Register (PLLCFG)	
MSEL: PLL Multiplier Selection	0x0028
NSEL: PLL Divider Selection	0x0002
CPU Clock Configuration Register (CCLKCFG)	
CCLKSEL: Divide Value for CPU Clock from PLL	0x0014
USB Clock Configuration Register (USBCLKCFG)	
Peripheral Clock Selection Register 0 (PCLKSEL0)	
PCLK_WDT: Peripheral Clock Selection for WDT	Pclk = Cclk / 4
PCLK_TIMER0: Peripheral Clock Selection for TIMER0	Pclk = Cclk / 2
PCLK_TIMER1: Peripheral Clock Selection for TIMER1	Pclk = Cclk / 2
PCLK_UART0: Peripheral Clock Selection for UART0	Pclk = Cclk / 4
PCLK_UART1: Peripheral Clock Selection for UART1	Pclk = Cclk / 4
PCLK_PWM0: Peripheral Clock Selection for PWM0	Pclk = Cclk / 4
PCLK_PWM1: Peripheral Clock Selection for PWM1	Pclk = Cclk / 4
PCLK_I2C0: Peripheral Clock Selection for I2C0	Pclk = Cclk / 4
PCLK_SPI: Peripheral Clock Selection for SPI	Pclk = Cclk / 4
PCLK_RTC: Peripheral Clock Selection for RTC	Pclk = Cclk / 4
PCLK_SSP1: Peripheral Clock Selection for SSP1	Pclk = Cclk / 4
PCLK_DAC: Peripheral Clock Selection for DAC	Pclk = Cclk / 4
PCLK_ADC: Peripheral Clock Selection for ADC	Pclk = Cclk / 4
PCLK_CAN1: Peripheral Clock Selection for CAN1	Pclk = Cclk / 4
PCLK_CAN2: Peripheral Clock Selection for CAN2	Pclk = Cclk / 4
PCLK_ACF: Peripheral Clock Selection for ACF	Pclk = Cclk / 4

CCLKSEL = 20 = 0x0014 hex, т.к. необходимо вводить значение уже на единицу больше. Получено опытным путем.

Проверка вычислительных расчетов

The screenshot displays the STM32CubeMX configuration interface. The 'Phase Locked Loop (PLL)' window is open, showing the following settings:

- Control Register:** PLLCON: 0x03, ☒ PLLE, ☒ PLLC
- Configuration Register:** PLLCFG: 0x00010027, MSEL: 0x0027, NSEL: 0x01
- Status Register:** PLLSTAT: 0x07010027, MSEL: 0x0027, NSEL: 0x01, ☒ PLOCK, ☒ PLLE, ☒ PLLC
- Feed Register:** PLLFEED: 0x55, Feed Sequence: Write 0xAA, Write 0x55
- PLL Clocks:** PLLIN: 12.000000 MHz, HCLK: 480.000000 MHz

The 'Clock Dividers' window is also open, showing the following settings:

- CPU Clock Configuration:** CCLKCFG: 0x13, CCLKSEL: 0x13, CCLK: 24.000000 MHz
- USB Clock Configuration:** USBCLKCFG: 0x07, USBSEL: 0x7, USBCLK: 60.000000 MHz
- Peripheral Clock Selection:**

Peripheral	Selection	Clock [MHz]
WDT	CCLK/4	6.000000
TIMER0	CCLK/2	12.000000
TIMER1	CCLK/2	12.000000
UART0	CCLK/4	6.000000
UART1	CCLK/4	6.000000
PWM1	CCLK/4	6.000000
I2C0	CCLK/4	6.000000
SPI	CCLK/4	6.000000
- Selected Peripheral:** PCLK_WDT: CCLK/4
- PCLKSEL0:** 0x00000028
- PCLKSEL1:** 0x00000000

In the background, a code editor shows the following C code snippet:

```

129     n = 0;
130
131     while(1)
132     {
133         tm1638_sendcmd(0x46);
    
```

PLLIN = 12 МГц, HCLK = 480 МГц, CCLK = 24 МГц,
Timer0 = Timer1 = 12 МГц

Листинг программы:

```
#include <LPC23xx.H>

#define BIT_BTN (1<<29)

#define STB 26 //Port1.26
#define CLK 27 //Port1.27
#define DIO 28 //Port1.28

#define STATE_OFF 0
#define STATE_ON 1

#define DIOD_VALVE 1
#define DIOD_IGNITE 2
#define DIOD_STARTER 3

void TimerDelay(void)
{
    T0TC = 0x00000000;
    T0TCR = 0x00000001;
    while (T0TCR&0x1);
}

void delay(unsigned int count)
{
    unsigned int i;
    for (i=0;i<count;i++){ }
}

void Timer0_Init(void)
{
    //Предделитель таймера = 600
    T0PR = 600;
    //Сбросить счетчик и делитель
    T0TCR = 0x00000002;
    //При совпадении останавливаем, сбрасываем таймер
    T0MCR = 0x00000006;
    //Регистр совпадения = 1000 (1 Гц)
    T0MR0 = 100;
}

void tm1638_sendbyte(unsigned int x)
{
    unsigned int i;
    IODIR1 |= (1 << DIO); //Устанавливаем пин DIO на вывод

    for(i = 0; i < 8; i++)
    {
        IOCLR1 = (1 << CLK); //Сигнал CLK устанавливаем в 0
        delay(1); //Задержка

        if (x & 1)
        {
            IOSET1 = (1 << DIO); //Устанавливаем значение на выходе DIO
        }
        else
        {
            IOCLR1 = (1 << DIO);
        }

        delay(1); //Задержка
        x >>= 1;
        IOSET1 = (1 << CLK); //Сигнал CLK устанавливаем в 1
        delay(2);
    }
}
```

```

unsigned int tm1638_receivebyte()
{
    unsigned int i;
    unsigned int x = 0;
    IODIR1 &= ~(1 << DIO); //Устанавливаем пин DIO на ввод

    for(i = 0; i < 32; i++)
    {
        IOCLR1 = (1 << CLK); //Сигнал CLK устанавливаем в 0
        delay(1);

        if (IOPIN1 & (1 << DIO))
        {
            x |= (1 << i);
        }

        delay(1);
        IOSET1 = (1 << CLK); //Сигнал CLK устанавливаем в 1
        delay(2);
    }

    return x;
}

void tm1638_sendcmd(unsigned int x)
{
    //Устанавливаем пассивный высокий уровень сигнала STB
    IOSET1 = (1 << STB);
    //Устанавливаем пины CLK,DIO,STB на вывод
    IODIR1 = (1 << CLK) | (1 << DIO) | (1 << STB);
    //Устанавливаем активный низкий уровень сигнала STB
    IOCLR1 = (1 << STB);
    tm1638_sendbyte(x);
}

void tm1638_setadr(unsigned int adr)
{
    //Установить адрес регистра LED индикации
    tm1638_sendcmd(0xC0 | adr);
}

void tm1638_init()
{
    unsigned int i;
    //Разрешить работу индикации
    tm1638_sendcmd(0x88);
    //Установить режим адресации: автоинкремент
    tm1638_sendcmd(0x40);
    //Установить адрес регистра LED индикации
    tm1638_setadr(0);
    //Сбросить все
    for (i = 0; i <= 0xf; i++)
        tm1638_sendbyte(0);
    //Установить режим адресации: фиксированный
    tm1638_sendcmd(0x44);
}

void SetDiod(unsigned diodNo, int state)
{
    diodNo *= 2;
    diodNo--;
    tm1638_setadr(diodNo); //устанавливаем адрес
    tm1638_sendbyte(state); //шлем данные
}

int main (void)
{

```

```

unsigned i;
unsigned flag = 0;
unsigned j = 0;
unsigned temp = 0;

tm1638_init();
Timer0_Init();

SetDiod( DIOD_VALVE, STATE_OFF );
SetDiod( DIOD_IGNITE, STATE_OFF );
SetDiod( DIOD_STARTER, STATE_OFF );

while (1)
{
    switch(flag)
    {
        case 0:
            // Включение стартера и попеременное включение и выключение диода зажигания и клапана
            SetDiod(DIOD_STARTER, STATE_ON );
            SetDiod(DIOD_VALVE, STATE_ON );
            TimerDelay();
            SetDiod(DIOD_VALVE, STATE_OFF );
            SetDiod(DIOD_IGNITE, STATE_ON );
            TimerDelay();
            SetDiod(DIOD_IGNITE, STATE_OFF );
            // Обработка нажатия кнопки
            tm1638_sendcmd(0x46);
            i = tm1638_receivebyte();
            if (i == 1)
            {
                SetDiod(DIOD_STARTER, STATE_OFF ); // Отключение стартера
                temp = 1;
            }
            if (temp == 1) // Счетчик после нажатия кнопки
                j++;
            if (j == 50)
            {
                SetDiod(DIOD_IGNITE, STATE_OFF ); // Отключение зажигания с закрытием клапана
                SetDiod(DIOD_VALVE, STATE_OFF );
                flag = 1;
            }
        case 1:
            // Полное отключение
            SetDiod(DIOD_IGNITE, STATE_OFF );
            SetDiod(DIOD_VALVE, STATE_OFF );
            SetDiod(DIOD_STARTER, STATE_OFF );
    }
}
}

```

Вывод.

Программа была успешно протестирована, функционирует на плате в соответствии указанному заданию. Работа светодиодов правильно имитирует устройство работы стиральной машины. Как следствие выполнения практического задания, изучены системы синхронизации микроконтроллера NXP LPC2368 и принципов функционирования таймеров общего назначения.