Лабораторная работа 2

Монахов Д.И.

Вариант 10.

**Задание**

Устройство управления духовкой, состоящее из трех нагревательных

элементов. Программа функционирования:

a) нагрев попеременно каждым нагревательным элементом;

b) при нажатии на кнопку: отключение.

Доработать программу из лабораторной 1 так, чтобы индикация работы устройств выполнялась с помощью платы TM-1638.

Получить осциллограмму для сигналов STB, CLK и DIO (порты Port1.26, Port1.27, Port1.28) для команды записи регистра управления LED.

**Листинг программы функционирования микроконтроллера**

Модуль main.c

/\* Monakhov D.I. IU7-54

Device controlls the oven. Oven consists of 3

heating elements. Functioning program:

1) Heating via each at once heating element

2) Pushing button turns oven off

\*/

// LPC2300.s EMC\_SETPU EQU 1. Line 435 - changed to make project to

// work in emulation

#include <LPC23xx.H>

#include "controller.h"

#define OUTPUT\_PINS 0x38000000 // Bit mask for all output pins

#define HEATER1\_PIN 0x20000000 // Bit mask of first heater

#define HEATER2\_PIN 0x10000000 // Second heater

#define HEATER3\_PIN 0x8000000 // Third heater

#define BUTTON\_PIN 0x4000000 // Button

struct tm1638 tm;

// Describes binary pin state

enum pin\_state

{

ENABLED = 1, DISABLED = 0

};

void set\_led\_enabled(int led\_num, int enabled)

{

int n = led\_num \* 2 + 1;

tm1638\_setadr(&tm, n);

if(enabled)

tm1638\_sendbyte(&tm, n);

else

tm1638\_sendbyte(&tm, 0);

}

int read\_key\_state()

{

tm1638\_setadr(&tm, 0);

tm1638\_sendcmd(&tm, 0x46); // Sending READ Command

return tm1638\_receivebyte(&tm) & (1); // We need only first button (SEG1)

}

// Sets state of selceted pin (or several pins)

void turn\_heater(int heater\_pin, enum pin\_state state)

{

int led\_num;

switch(heater\_pin)

{

case HEATER1\_PIN:

led\_num = 0;

break;

case HEATER2\_PIN:

led\_num = 1;

break;

case HEATER3\_PIN:

led\_num = 2;

break;

default: led\_num = 0;

}

switch(state)

{

case ENABLED:

//IOSET0 = heater\_pin;

set\_led\_enabled(led\_num, 1);

break;

case DISABLED:

//IOCLR0 = heater\_pin;

set\_led\_enabled(led\_num, 0);

break;

}

}

// Reads state of selected pin

enum pin\_state read\_pin(int pin)

{

if((IOPIN0 & pin) != 0)

return ENABLED;

return DISABLED;

}

int main(void)

{

// Variable to contain previous state of a button

enum pin\_state prev\_button\_state;

// Current state of device (1 - turned on, 0 - turned off)

int device\_enabled = 1;

int prev\_enabled\_state = 0;

// Timer

int current\_tick = 0;

// Time to switch heater

int cycle\_tick\_count = 1000000; // ~~ 1 second

// Current active heater pin

int current\_heater = HEATER1\_PIN;

/\*// Set controll to General Purpose Input/Outpu

PINSEL0 = 0;

/// Set output direction for heaters' pins

IODIR0 = OUTPUT\_PINS;\*/

tm.STB = 26;

tm.CLK = 27;

.

tm.DIO = 28;

tm1638\_init(&tm);

while(1)

{

// Checking button

enum pin\_state button\_state = read\_pin(BUTTON\_PIN);

if(button\_state == DISABLED && prev\_button\_state == ENABLED)

{

// Switching device state

device\_enabled = !device\_enabled;

}

// Updating previous button state

prev\_button\_state = button\_state;

// Checking if device is enabled

if(device\_enabled)

{

// If timer was reset

if(current\_tick == 0)

{

// Enabling current heated

turn\_heater(current\_heater, ENABLED);

}

// Checking how much time past since timer reset

if(current\_tick >= cycle\_tick\_count)

{

// Reset timer

current\_tick = 0;

// Turn current heater off

turn\_heater(current\_heater, DISABLED);

// Switch heater to next

if(current\_heater == HEATER3\_PIN)

current\_heater = HEATER1\_PIN;

else

current\_heater >>= 1;

}

else

current\_tick++; // Timer increment

prev\_enabled\_state = 1;

}

else // The device if turned off

{

//turn\_heater(HEATER1\_PIN | HEATER2\_PIN | HEATER3\_PIN, DISABLED); // Turn off all heaters

if(prev\_enabled\_state == 1)

{

turn\_heater(HEATER1\_PIN, DISABLED);

turn\_heater(HEATER2\_PIN, DISABLED);

turn\_heater(HEATER3\_PIN, DISABLED);

}

prev\_enabled\_state = 0;

current\_tick = 0; // Reset timer

}

}

}

Модуль controller.c

#include "controller.h"

#include <LPC23xx.H>

// Delay

void delay(unsigned int ticks)

{

unsigned int i;

for(i = 0; i < ticks; i++)

{

}

}

void tm1638\_sendbyte(struct tm1638\* controller, unsigned int data)

{

int i;

int outputDioPin = 1 << controller->DIO;

int outputClkPin = 1 << controller->CLK; // Shifting 1 to pin with number CLK

// Setting DIO to output

IODIR1 |= (1 << outputClkPin);

for(i = 0; i < 8; i++)

{

IOCLR1 = outputClkPin; // Clearing CLK pin

delay(0xfff); // Delay

if(data & 1) // Checking if current data's bit is 1

IOSET1 = outputDioPin;

else

IOCLR1 = outputDioPin;

delay(0xfff);

data >>= 1; // Shifing data towards right;

IOSET1 = outputClkPin; // Setting CLK pin

delay(0x1fff);

}

}

unsigned int tm1638\_receivebyte(struct tm1638\* controller)

{

unsigned int data = 0;

int i;

int outDioPin = 1 << controller->DIO;

int outClkPin = 1 << controller->CLK;

IODIR1 = IODIR1 & ~outDioPin; // Setting DIO pin to input;

for(i = 0; i < 32; i++)

{

IOCLR1 = outClkPin; // Clear CLK

delay(0xfff);

if(IOPIN1 & outDioPin) // Reading DIO

{

data = data | (1 << i); // Setting bit

}

delay(0xfff);

IOSET1 = outClkPin; // Setting CLK;

delay(0x1fff);

}

return data;

}

void tm1638\_sendcmd(struct tm1638\* controller, unsigned int cmd)

{

IOSET1=(1<<controller->STB);

IODIR1 = (1<<controller->CLK)|(1<<controller->DIO)|(1<<controller->STB);

IOCLR1=(1<<controller->STB);

tm1638\_sendbyte(controller, cmd);

}

void tm1638\_setadr(struct tm1638\* controller, unsigned int adr)

{

// C0h == 1100 0000

// 0xC0 | adr => 1100 XXXX

tm1638\_sendcmd(controller, 0xC0 | adr);

}

void tm1638\_init(struct tm1638\* controller)

{

int i;

tm1638\_sendcmd(controller, 0x88); // Enable work of indication;

tm1638\_setadr(controller, 0); // Clear address

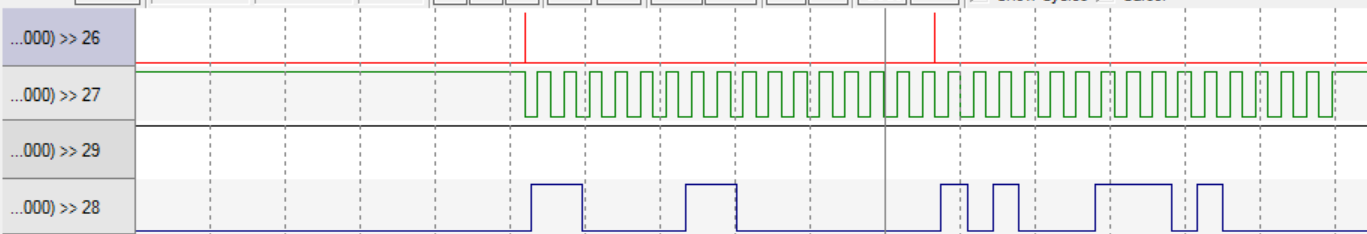
for(i = 0; i < 0xf; i++)

tm1638\_sendbyte(controller, 0); // Setting fixed addressing mode

tm1638\_sendcmd(controller, 0x44);

}

**Осциллограмма сигналов**

****

Фрагмент осциллограммы демонстрирует процесс передачи команды и данных с ведущего микроконтроллера на плату TM-1638.

**Вывод**

Микросхема TM-1638 является узкоспециализированным устройством, она берет на себя управление светодиодами и клавишами, снимая эту задачу с основного микроконтроллера. Последнему требуется задействовать всего три пина для работы с целым набором клавиш и светодиодов. Принцип действия основан на передаче команд и данных от ведущего микроконтроллера к TM-1638, и получение от неё информации.