Министерство образования и науки Российской Федерации

Санкт-Петербургский Политехнический Университет Петра Великого

—

Институт кибербезопасности и защиты информации

ЛАБОРАТОРНАЯ РАБОТА № 5

1. «Работа с внешними интерфейсами: UART и I2C»
2. по дисциплине «Аппаратные средства вычислительной техники»
3. Выполнил

студент гр. 4851004/10001 Тоцкий В.

<*подпись*>

Проверил:

Старший преподаватель Макаров А. С.

<*подпись*>

1. Санкт-Петербург
   1. 2023

# Цель работы

Изучение структуры данных, пересылаемых через последовательный порт. Получение практических навыков программирования встроенного в микроконтроллер UART модуля. Получение практических навыков

# Формулировка задания

Настройки UART-интерфейса: скорость 57600 бод, контроль чётности – even, один стоповый бит, 8 значащих битов в пакете. Программа для МК должна выполнять следующие действия:

1. ввод данных с терминала ПК (передача на МК по UARTинтерфейсу) и сохранение их во внешней памяти EEPROM (интерфейс TWI):

1.1. ввод строковых переменных {x, y, z}. Вводимая строка может содержать печатные ASCII-символы (коды символов должны быть больше 31), длина строки не должна превышать 20 символов. Формат запроса в терминале: «имя\_строковой\_переменной=значение»;

1.2. ввод 8-разрядных неотрицательных целочисленных переменных {A, B, C}. Формат запроса в терминале: «имя\_целочисленной\_переменной=значение»;

2. Запрос информации от МК (в том числе из подключённой к нему внешней памяти EEPROM) с выводом в терминал ПК:

2.1. вывод значений целочисленных и строковых переменных на терминал ПК. Формат запроса в терминале: «=имя\_переменной»;

2.2. вывод результата операции со строковыми переменными на терминал ПК, значения задействованных в операции целочисленных и строковых переменных считываются из внешней памяти непосредственно перед выполнением операции. Формат запроса в терминале «=название\_команды(параметры)».

3. выполнение операции со строковыми переменными, значения задействованных в операции целочисленных и строковых переменных считываются из внешней памяти непосредственно перед выполнением операции. Формат запроса в терминале:«имя\_переменной=название\_команды(параметры)».В качестве параметров могут выступать только целочисленные и строковые переменные;

Реализуемые операции: - x=strcmp(s1,s2) – сравнение строк s1 и s2 – 8-разрядное беззнаковое число: 0x00, если s1 = s2; 0x01, если s1 > s2; 0xFF, если s1 < s2; - s0=strcat(s1,s2) – объединение строк s1 и s2 и помещение результата в переменную s0. Если при объединении получается строка длиннее 20 символов, то остаются только первые 20 символов;

# Схема лабораторной установки

Изображение выглядит как текст, диаграмма, План, линия

Автоматически созданное описание

# Блок-схема алгоритма работы программы

Изображение выглядит как диаграмма, текст, План, Технический чертеж

Автоматически созданное описание

# Ответы на контрольные вопросы

1. Укажите назначение конфигурационных регистров модуля UART?

// UCSRA:

// RXC - завершение приёма

// |TXC - завершение передачи

// ||UDRE - отсутствие данных для отправки

// |||FE - ошибка кадра

// ||||DOR - ошибка переполнение буфера

// |||||PE - ошибка чётности

// ||||||U2X - Двойная скорость

// |||||||MPCM - Многопроцессорный режим

// UCSRB:

// RXCIE - прерывание при приёме данных

// |TXCIE - прерывание при завершении передачи

// ||UDRIE - прерывание отсутствие данных для отправки

// |||RXEN - разрешение приёма

// ||||TXEN - разрешение передачи

// |||||UCSZ2 - UCSZ0:2 размер кадра данных

// ||||||RXB8 - 9 бит принятых данных

// |||||||TXB8 - 9 бит переданных данных

// UCSRC:

// URSEL - всегда 1

// |UMSEL - режим: 1-синхронный 0-асинхронный

// ||UPM1 - UPM0: 1 чётность

// |||UPM0 - UPM0: 1 чётность

// ||||USBS - стоп биты: 0–1, 1–2

// |||||UCSZ1 - UCSZ0: 2 размер кадра данных

// ||||||UCSZ0 - UCSZ0: 2 размер кадра данных

// |||||||UCPOL - в синхронном режиме – тактирование

UBBRx задаёт скорость передачи и приёма данных.

1. Какими способами можно осуществлять последовательную передачу данных в микроконтроллерах AVR?

Существует 2 способа для последовательной передачи данных: синхронный и асинхронный. Синхронная передача — это метод передачи данных, который отправляет непрерывный поток данных в приемник, используя регулярные сигналы синхронизации, что обеспечивает синхронизацию как передатчика, так и приемника. Асинхронная передача данных — это метод передачи данных, который отправляет данные от передатчика к приемнику с битами четности (начальным и конечным битами) в неравных интервалах.

1. Какие события генерируются модулем UART?

Модуль UART вызывает прерывания по завершению приёма, передачи, а также по опустошению буфера.

1. Каков размер адресного пространства шины I2C?

В обычном режиме I2C имеет 7-битное адресное пространство, однако доступно редко используемое 10-битное расширение.

1. В каком режиме осуществляется связь по шине I2C, каковы могут быть роли устройств на шине?

Связь по шине I2C осуществляется в синхронном режиме, существуют две линии: линия данных (SDA) и линия тактирования (SCL). На линии могут быть ведущие и ведомые устройства. Каждое устройство может быть приёмником или передатчиком. Инициатором обмена всегда выступает ведущий, обмен между двумя ведомыми невозможен. Такты на линии SCL генерирует ведущий. Линией SDA могут управлять как ведущий, так и ведомый в зависимости от направления передачи.

# Выводы

В ходе лабораторной работы была изучена структура данных, пересылаемых через последовательный порт. Также были получены практические навыки программирования, встроенного в микроконтроллер UART модуля. Кроме этого, были получены практические навыки по работе с интерфейсом I2C и принципами работы памяти EEPROM.

Приложение А

Листинг программы «uart.c»

/\*

\* main.c

\*

\* Created: 10/12/2023 4:43:26 PM

\* Author: Денис

\*/

#define F\_CPU 8000000UL

#include <stdint.h>

#include <avr/interrupt.h>

#include <avr/sfr\_defs.h>

#include <util/delay\_basic.h>

#include <avr/eeprom.h>

#include <util/delay.h>

#include <string.h>

#define TRUE 1

#define FALSE 0

#define N 127

#define MAX\_LEN 20

struct \_in\_out

{

char input[N];

uint8\_t index;

uint8\_t x;

uint8\_t y;

uint8\_t z;

uint8\_t A;

uint8\_t B;

uint8\_t C;

};

struct \_in\_out io = { 0 };

void data\_init(void)

{

io.x = 0;

io.y = 20;

io.z = 40;

io.A = 41;

io.B = 42;

io.C = 43;

}

void usart\_init(void)

{

UBRRL = 16;

UBRRH = 0;

UCSRB |= (1 << TXEN) | (1 << RXEN) | (1 << RXCIE);

UCSRC = (1 << URSEL) |

(1 << UCSZ0) |

(1 << UCSZ1) |

(1 << UPM1);

UCSRA |= (1 << U2X);

}

void USARTTransmitChar(char c)

{

while(!( UCSRA & (1<<UDRE)));

UDR = c;

}

// Отправка строки

void USARTTransmitString(const char\* str)

{

register char i = 0;

while(str[i])

USARTTransmitChar(str[i++]);

}

// Отправка строки

void USARTTransmitStringLn(const char\* str)

{

USARTTransmitString(str);

USARTTransmitChar((char)13);

USARTTransmitChar((char)10);

}

int8\_t check\_string(void)

{

return io.index != 0;

}

int8\_t my\_strcmp(char\* s1, char\* s2)

{

int8\_t tmp = 0;

tmp = strncmp((char\*)s1, (char\*)s2, MAX\_LEN);

if(tmp > 0)

return 1;

if(tmp == 0)

return 0;

if(tmp < 0)

return 0xff;

}

char\* my\_strcat(char\* s1, char\* s2)

{

return strncat(s1, s2, MAX\_LEN);

}

void reverse(char\* str, int len) {

int i, j;

char temp;

for (i = 0, j = len - 1; i < j; i++, j--) {

temp = str[i];

str[i] = str[j];

str[j] = temp;

}

}

char\* itoa(int num, char\* str, int base) {

int i = 0;

int is\_negative = FALSE;

if (num == 0) {

str[i++] = '0';

str[i] = '\0';

return str;

}

if (num < 0 && base == 10) {

is\_negative = TRUE;

num = -num;

}

while (num != 0) {

int remainder = num % base;

str[i++] = (remainder > 9) ? (remainder - 10) + 'a' : remainder + '0';

num = num / base;

}

if (is\_negative) {

str[i++] = '-';

}

str[i] = '\0';

reverse(str, i);

return str;

}

uint8\_t Matoi(char\* buf)

{

uint8\_t num = 0;

for (;(\*buf >= '0') && (\*buf <= '9'); buf++)

num = num \* 10 + (\*buf - '0');

return num;

}

void eeprom\_write(int addr, uint8\_t val){

uint8\_t\* ptr = (uint8\_t\*)addr;

eeprom\_busy\_wait();

eeprom\_write\_byte(ptr, val);

}

void eeprom\_write\_string(int addr, char\* string)

{

for(uint8\_t i = 0;i < MAX\_LEN && string + i;i++)

eeprom\_write(addr + i, string[i]);

}

uint8\_t eeprom\_read(int addr)

{

eeprom\_busy\_wait();

return eeprom\_read\_byte((uint8\_t\*)addr);

}

char\* eeprom\_read\_string(int addr, char\* string)

{

for(uint8\_t i = 0;i < MAX\_LEN;i++)

string[i] = eeprom\_read(addr + i);

return string;

}

ISR(USART\_RXC\_vect)

{

while(!(UCSRA & (1<<RXC)));

io.input[io.index++] = UDR;

io.input[io.index] = 0;

}

ISR(USART\_TXC\_vect)

{

}

int main(void)

{

data\_init();

usart\_init();

sei();

while(1)

{

if(check\_string())

{

USARTTransmitStringLn("");

if (io.input[0] == '=')

{

char tmp[MAX\_LEN + 1] = { 0 };

switch (io.input[1])

{

case 'x': USARTTransmitStringLn(eeprom\_read\_string(io.x, tmp)); break;

case 'y': USARTTransmitStringLn(eeprom\_read\_string(io.y, tmp)); break;

case 'z': USARTTransmitStringLn(eeprom\_read\_string(io.z, tmp)); break;

case 'A': USARTTransmitStringLn(itoa(eeprom\_read(io.A), tmp, 10)); break;

case 'B': USARTTransmitStringLn(itoa(eeprom\_read(io.B), tmp, 10)); break;

case 'C': USARTTransmitStringLn(itoa(eeprom\_read(io.C), tmp, 10)); break;

default: USARTTransmitStringLn("Error: unknown command");

}

}

else

{

char var\_name = 0, op\_name[MAX\_LEN + 1] = { 0 }, op\_params[2 + 2 \* MAX\_LEN] = { 0 };

sscanf(io.input, "%[^=]=%[^(](%[^)])", &var\_name, op\_name, op\_params); // считываем имя переменной, имя операции и ее параметры

if (strcmp(op\_name, "strcmp") == 0)

{

char src1[MAX\_LEN + 1] = { 0 }, src2[MAX\_LEN + 1] = { 0 };

sscanf(op\_params, "%[^,],%s", src1, src2); // считываем параметры операции

switch (var\_name)

{

case 'A': eeprom\_write(io.A, (uint8\_t)my\_strcmp(src1, src2)); break;

case 'B': eeprom\_write(io.B, (uint8\_t)my\_strcmp(src1, src2)); break;

case 'C': eeprom\_write(io.C, (uint8\_t)my\_strcmp(src1, src2)); break;

default: USARTTransmitStringLn("Error param");

}

}

else if (strcmp(op\_name, "strcat") == 0)

{

char src1[MAX\_LEN + 1] = { 0 }, src2[MAX\_LEN + 1] = { 0 };

sscanf(op\_params, "%[^,],%s", src1, src2); // считываем параметры операции

switch (var\_name)

{

case 'x': eeprom\_write\_string(io.x, my\_strcat(src1, src2)); break;

case 'y': eeprom\_write\_string(io.y, my\_strcat(src1, src2)); break;

case 'z': eeprom\_write\_string(io.z, my\_strcat(src1, src2)); break;

default: USARTTransmitStringLn("Error param");

}

}

else if (io.input[1] == '=' && io.input[2])

{

switch (var\_name)

{

case 'x': eeprom\_write\_string(io.x, op\_name); break;

case 'y': eeprom\_write\_string(io.y, op\_name); break;

case 'z': eeprom\_write\_string(io.z, op\_name); break;

case 'A': eeprom\_write(io.A, Matoi(op\_name)); break;

case 'B': eeprom\_write(io.B, Matoi(op\_name)); break;

case 'C': eeprom\_write(io.C, Matoi(op\_name)); break;

default: USARTTransmitStringLn("Error: unknown command");

}

}

else

USARTTransmitStringLn("Error: unknown command");

}

io.index = 0;

}

\_delay\_ms(1000);

}

}