**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ**

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ **«БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ ИМ. В. Г. ШУХОВА»**

Кафедра программного обеспечения вычислительной техники и автоматизированных систем

**ЛАБОРАТОРНАЯ РАБОТА №2**

**Дисциплина: ЭВМ и периферийные устройства**

**Тема: Изучение принципов организации обмена данными по последовательному интерфейсу USB между микроконтроллером MSP430F1611 и ПЭВМ**

Выполнил: ст. группы ВТ-31

Подкопаев Антон Валерьевич

Проверил: доцент кафедры ПО и ВТАС

Шамраев Анатолий Анатольевич

**Белгород 2020**

**Цель работы:** изучить возможности сопряжения лабораторного стенда на базе микроконтроллера MSP430F1611 и ПЭВМ с помощью последовательного интерфейса USB, принципы программного управления двунаправленным обменом данных по последовательному интерфейсу USB.

**Указания по организации самостоятельной работы**

Перед работой необходимо проработать теоретический материал по литературе и конспекту лекций, ознакомиться с основными возможностями и принципами функционирования последовательного интерфейса USB, возможностями сопряжения лабораторного макета на базе микроконтроллера MSP430F1611 и ПЭВМ с помощью последовательного интерфейса USB, принципами программного управления двунаправленным обменом данных по последовательному интерфейсу USART.

**Порядок проведения работы и указания по ее выполнению**

Перед началом выполнения практической части лабораторной работы проводится экспресс-контроль знаний по принципам функционирования микроконтроллера MSP430, системе команд и возможностям организации программного управления выводом символов на экран цифрового индикатора. При подготовке к лабораторной работе необходимо составить предварительный вариант листинга программы, в соответствие с индивидуальным заданием.

**Задание.** Разработать в среде программирования IAR Embedded Workbench программу на языке С для связи микроконтроллера MSP430F1611 с ПЭВМ по интерфейсу USB в соответствие с параметрами протокола обмена, приведенными в таблице 4.2.8.

**Порядок выполнения задания:**

– создать файл ресурса для кода программы и подключить его к проекту.

– ввести код исходного модуля программы обмена данными между микроконтроллером MSP430F1611 с ПЭВМ по интерфейсу USB в соответствие с индивидуальным заданием, приведенным в таблице

– выполнить компиляцию исходного модуля программы и устранить ошибки, полученные на данном этапе.

– настроить параметры программатора.

– проверить правильность подключения интерфейсного USB кабеля к разъемам лабораторного макета и ПЭВМ.

– запустить на ПЭВМ программу Terminal, установить необходимые параметры протокола обмена данными, выбрать номер последовательного порта (СОМx), соответствующего виртуальному COM-порту, и нажать на кнопку Connect в верхнем левом углу рабочего окна программы.

Теоретические сведения

**Режим UART**

В режиме UART модуль USART передает и принимает символы на скорости, асинхронной другому устройству. Синхронизация каждого символа основана на выбранной скорости передачи USART. Для выполнения функций передачи и приема используется одинаковая скорость в бодах.

**Многопроцессорный формат со свободной линией**

При MM=0 выбирается многопроцессорный формат со свободной линией. Блоки данных на линиях передачи или приема должны быть разделены временем простоя (рисунок 4.2.3). Простой линии приема обнаруживается, когда приняты 10 или более непрерывных логических единиц (меток) после первого стопового бита символа. Если используются два стоповых бита, то второй стоповый бит воспринимается как первый маркерный бит периода простоя. Первый символ, принятый после периода простоя, распознается как адрес. Бит RXWAKE используется как адресный тэг для каждого фрэйма. В многопроцессорном формате свободной линии этот бит устанавливается в 1, когда принятый адрес помещается в UxRXBUF.

Бит URXWIE используется для приема управляющих данных в многопроцессорном формате со свободной линией. Когда бит URXWIE установлен, все неадресные символы обрабатываются, но не перемещаются в UxRXBUF и прерывания не генерируются. Когда принят адресный символ, приемник временно активизируется для переноса символа в UxRXBUF и установки флага прерывания URXIFGx. После приема адреса программное обеспечение пользователя должно проверить его корректность и сбросить URXWIE для продолжения приема данных. Если URXWIE остается установленным, будут приниматься только адресные символы. Бит URXWIE не изменяется аппаратно. При передаче адреса в многопроцессорном формате со свободной линией точный период простоя для генерации идентификаторов адресного символа на UTXDx может быть сгенерирован модулем USART с использованием флага временного пробуждения (WUT). Когда в передатчик загружаются данные из UxTXBUF, в WUT сохраняется состояние бита TXWAKE, а сам бит TXWAKE сбрасывается. Для формирования фрэйма простоя используется следующая процедура:

1. Устанавливается TXWAKE, что приводит к записи любого символа в UxTXBUF, при этом UxTXBUF должен быть готов для новых данных (UTXIFGx=1).

2. Значение TXWAKE сдвигается в WUT, а содержимое UxTXBUF сдвигается в сдвиговый регистр передачи, когда он готов для передачи новых данных. Это приводит к установке бита WUT, который препятствует нормальной передаче битов старта, данных и контроля четности, поэтому происходит передача периода простоя длительностью точно 11 бит, после чего бит TXWAKE сбрасывается автоматически.

3. Требуемый адресный символ записывается в UxTXBUF. 4. Новый символ, представляющий требуемый адрес, сдвигается наружу после периода простоя.

**Многопроцессорный формат с адресным битом**

При MM=1 выбирается многопроцессорный формат с адресным битом. Каждый обрабатываемый символ содержит дополнительный бит, используемый как указатель адреса (рисунок 4.2.4). Первый символ в блоке фрэймов несет установленный бит адреса, который указывает, что этот символ является адресом. Бит USART RXWAKE устанавливается, когда в UxRXBUF записывается принятый адресный символ фрейма. Бит URXWIE используется для приема управляющих данных в многопроцессорном формате с адресным битом. Когда бит URXWIE установлен, символы данных (бит адреса равен 0) обрабатываются приемником, но не перемещаются в UxRXBUF и прерывания не генерируются. Когда принятый символ содержит установленный адресный бит, приемник временно активизируется для переноса символа в UxRXBUF и установки флага прерывания URXIFGx. Если адрес принят, программное обеспечение пользователя должно сбросить URXWIE для продолжения приема данных. Если URXWIE остается установленным, будут приниматься только адресные символы (адресный бит равен 1). Бит URXWIE не изменяется аппаратно. При передаче адреса в многопроцессорном режиме с адресным битом, адресный бит символа может изменяться путем записи бита TXWAKE.

Значение бита TXWAKE загружается в адресный бит символа, перемещенного из UxTXBUF в сдвиговый регистр передачи, при этом бит TXWAKE автоматически очищается.

**Размещение интерфейса USB на плате**

**Изображение выглядит как снимок экрана

Автоматически созданное описание**

**Изображение выглядит как снимок экрана

Автоматически созданное описание**

*Приложение*

uart.c

// UART functions

#include "function\_prototype.h"

#include "sysfunc.h"

#include "uart.h"

// Инициализация режима UART

// speed = 0 - 38400, 1 - 57600, 2 - 115200 - скорость обмена

// databits = 7, 8 - длинна символа

// stopbits = 1, 2 - кол-во передаваемых стоповых бит

// parity = 0 - без контроля четности, 1 - контроль четности, нечетый, 2 - четный

// iface = 0 - USB, 1 - оптика

void UART\_init(byte speed, byte databits, byte stopbits, byte parity, byte iface)

{

P3SEL |= BIT6 | BIT7; // выбор функции USART1

U1CTL = 0; // инициализация состояния USART

ME2 |= UTXE1 + URXE1; // включить приемник и передатчик USART1

if (databits == 7) U1CTL &= ~CHAR; // 7-разрядная длинна символа

if (databits == 8) U1CTL |= CHAR; // 8-разрядная длинна символа

if (stopbits == 1) U1CTL &= ~SPB; // 1 стоповый бит

if (stopbits == 2) U1CTL |= SPB; // 1 стоповых бита

if (parity == 0) U1CTL &= ~PENA; // контроль четности отключен

if (parity == 1) U1CTL = (U1CTL & ~PEV) | PENA; // контроль четности, нечетный

if (parity == 2) U1CTL |= PENA | PEV; // контроль четности, четный

P5DIR |= BIT0; // переключение мультиплексора на USB/оптику

if (iface == 0)

P5OUT |= BIT0;

if (iface == 1)

P5OUT &= ~BIT0;

U1TCTL |= SSEL1; // BRCLK = SMCLK

U1BR0 = 69; // 8Mhz / 115200 = 69.44 (по-умолчанию)

if (speed == 0) U1BR0 = 208; // 8МГц / 38400 = 208.33

if (speed == 1) U1BR0 = 139; // 8МГц / 57600 = 138,89

if (speed == 2) U1BR0 = 0x2B; // 8МГц / 14400 = 555,55

U1BR1 = 0x02;

U1MCTL = 0x37;

// модуляция

}

// отключение режима UART

void UART\_off()

{

P3SEL |= BIT6 | BIT7; // выбор функции USART1

ME2 &= ~(UTXE1 + URXE1); // выключить приемник и передатчик USART1

U1CTL = SWRST; // отключение USART1

}

// вывод строки символов (символ с кодом 0 - конец строки)

void UART\_message(char \* buf)

{

word i=0;

while (buf[i])

UART\_sendbyte(buf[i++]); // передача сивола

}

// передача байта

void UART\_sendbyte(char byte)

{

while (!(IFG2 & UTXIFG1)); // проверка готовности буфера передачи USART1

U1TXBUF = byte; // передача байта

}

// получение байта

char UART\_getbyte()

{

while (!(IFG2 & URXIFG1)); // проверка готовности буфера приема USART1

return U1RXBUF; // возврат полученного байта

}

main.c

#include <msp430.h>

#include "stdio.h"

#include "system\_define.h"

#include "system\_variable.h"

#include "function\_prototype.h"

#include "main.h"

/\*

\* main.c

\*/

void main(void) {

WDTCTL = WDTPW + WDTHOLD;

Init\_System\_Clock();

Init\_System();

UART\_init(2, 7, 1, 0, 0);

char i;

while(1){

for(i = 10; i < 60; i++){

UART\_sendbyte(i);

}

UART\_off();

}

}