**МИНОБРНАУКИ РОССИИ**

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ

ВЫСШЕГО ОБРАЗОВАНИЯ

**«БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ**

**ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ им. В.Г.ШУХОВА»**

**(БГТУ 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-32

Воскобойников И. С.

Проверил: Шамраев А.А.

Белгород 2021

**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.

**Задание**: Разработать программу передачи 20 чисел (от 10 до 29) из

микроконтроллера MSP430F1611 в ПЭВМ по интерфейсу USB в

соответствие с протоколом: модуль USART0, скорость обмена

данными 57600 бит/с, режим обмена асинхронный, 7 битов данных без

бита четности.

**Теоретические сведения**

**Организация модулей USART в микроконтроллере MSP430**

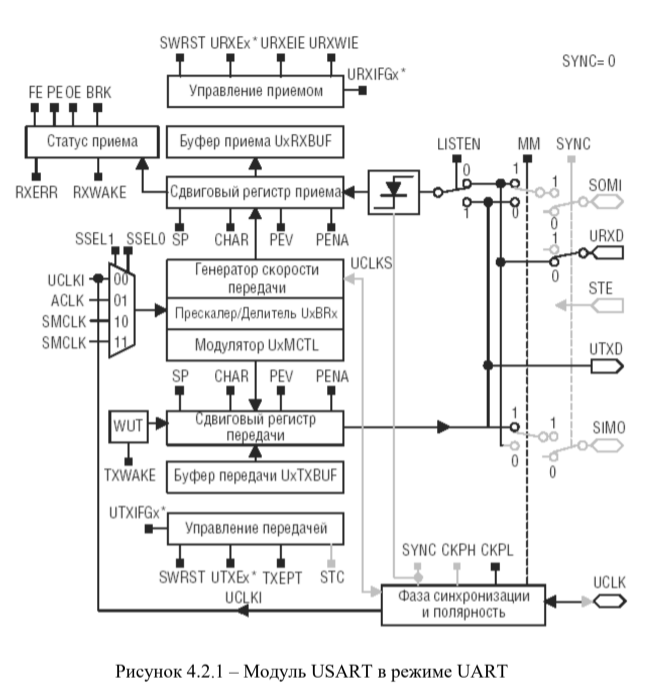
Универсальный синхронно/асинхронный приемопередающий (USART) периферийный интерфейс поддерживает два последовательных режима. USART0 реализован в устройствах MSP430x12xx, MSP430x13xx и MSP430x15x. В дополнение к USART0 в устройствах MSP430x14x и MSP430x16x реализован второй идентичный USART модуль – USART1.

В асинхронном режиме USART подключает MSP430 к внешней системе через два внешних вывода: URXD и UTXD. Режим UART выбирается при очистке бита SYNC.

На рисунке 4.2.1 показан модуль USART, настроенный на режим работы UART.

**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.



**Инициализация и сброс USART**

Модуль USART сбрасывается сигналом PUC или при установке бита SWRST. После PUC бит SWRST автоматически устанавливается, переводя USART в состояние сброса. Когда бит SWRST установлен, биты URXIEx, UTXIEx, URXIFGx, RXWAKE, TXWAKE, RXERR, BRK, PE, OE, FE сбрасываются, а биты UTXIFGx и TXEPT – устанавливаются. Флаги разрешения приема и передачи URXEx и UTXEx не изменяются битом SWRST.

Очистка SWRST переводит модуль USART в активный режим.

Процесс инициализации/реконфигурирования USART необходимо

выполнять в следующей последовательности:

– установить SWRST (BIS.B #SWRST,&UxCTL);

– проинициализировать все регистры USART при установленном

SWRST=1 (включая UxCTL);

– включить модуль USART через MEx SFRs (URXEx и/или UTXEx);

– программно очистить SWRST (BIC.B #SWRST,&UxCTL);

– разрешить прерывания (если необходимо) через IEx SFRs (URXIEx

и/или UTXIEx).

Невыполнение этой последовательности может привести к

непредсказуемому поведению USART.

**Формат символа**

Формат символа USART (рисунок 4.2.2) содержит стартовый бит, семь или восемь битов данных, бит контроля четности, адресный бит (в адресном режиме) и один или два стоповых бит. Период битов определяется выбранным источником тактовых импульсов и настройкой регистров скорости передачи.



**Асинхронные коммуникационные форматы**

Когда два устройства обмениваются информацией асинхронно, в качестве

протокола используется формат «свободная линия». Для обмена данными

между тремя или более устройствами, USART поддерживает

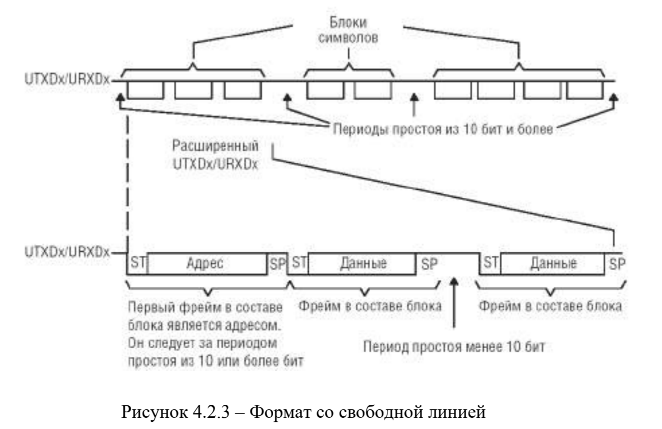
многопроцессорные коммуникационные форматы: формат со свободной

линией и формат с адресным битом.

**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**

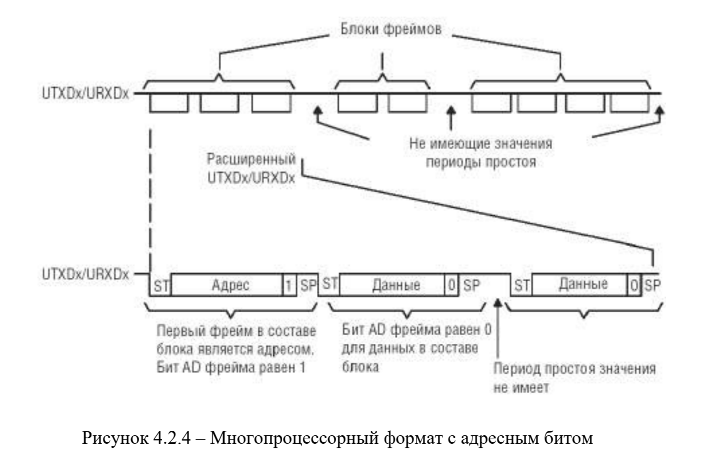
При MM=1 выбирается многопроцессорный формат с адресным битом. Каждый обрабатываемый символ содержит дополнительный бит, используемый как указатель адреса (рисунок 4.2.4). Первый символ в блоке фрэймов несет установленный бит адреса, который указывает, что этот символ является адресом. Бит USART RXWAKE 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 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.



**Автоматическое обнаружение ошибок**

Подавление импульсных помех предотвращает случайный запуск USART. Любой сигнал низкого уровня на URXDx продолжительностью менее t=300 нс игнорируется.

Если длительность сигнала низкого уровня на URXDx превышает t , этот

сигнал мажоритарно принимается за стартовый бит. Если стартовый бит не будет распознан, то модуль USART приостанавливает прием символа и ожидает следующего периода низкого уровня на URXDx. Мажоритарный принцип также используется для предотвращения поразрядных ошибок при приеме каждого бита символа.

Модуль USART при приеме символов автоматически обнаруживает ошибки фрэйма, четности, переполнения и прерывания (разрыва). Обнаружение ошибки приводит к установке соответствующих битов FE, PE, OE и BRK. При

установке любого из этих флагов также устанавливается RXERR. Ситуации

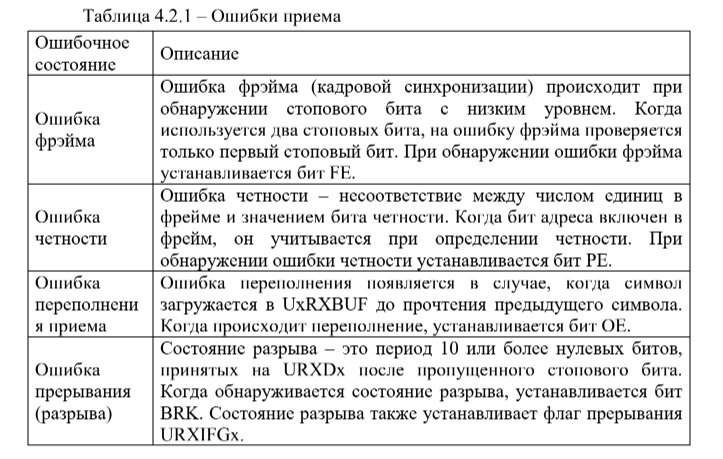
сбоев описаны в таблице 4.2.1.

Если обнаружена ошибка фрэйма, четности или состояние разрыва и URXEIE=0, то символы приниматься не будут. Когда URXEIE=1, символы принимаются и сохраняются в UxRXBUF, при этом устанавливаются соответствующие биты ошибок.

Если любой из битов FE, PE, OE, BRK или RXERR был установлен, то он

сохраняет свое состояние до сброса программным обеспечением или до чтения

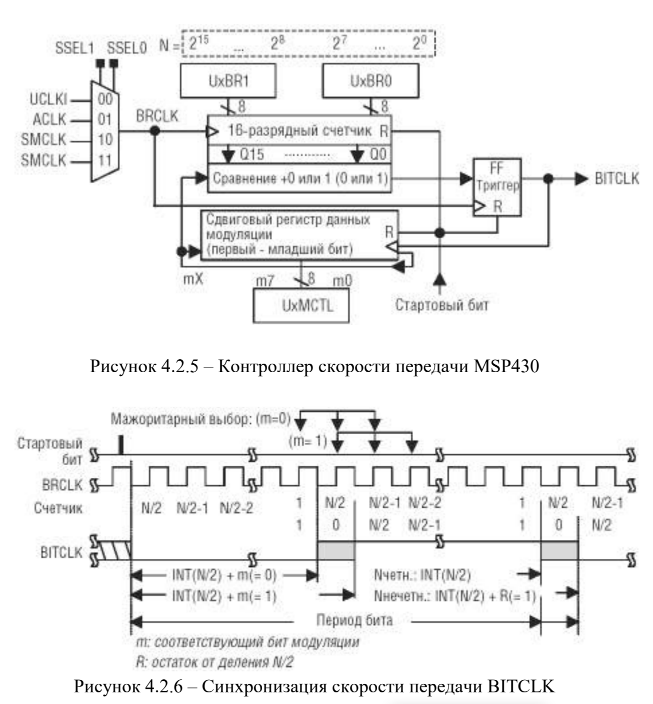
UxRXBUF.



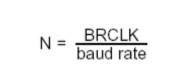
**Контроллер скорости передачи UART**

Контроллер скорости передачи включает в себя один прескалер/делитель и модулятор (рисунок 4.2.5). Такая структура позволяет получить дробные коэффициенты деления при генерации скорости передачи в бодах. Максимальная скорость передачи USART составляет одну треть от тактовой частоты BRCLK.

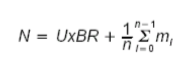
Синхронизация каждого бита показана на рисунке 4.2.6. Для каждого полученного бита используется мажоритарный принцип определения значения бита. Мажоритарные выборки происходят в N/2-1, N/2 и N/2+1 периоды BRCLK, где N – число импульсов BRCLKs на один импульс BITCLK.

**Синхронизация скорости передачи бит**

Первая ступень контролера скорости передачи – 16-разрядный счетчик компаратор. В начале передачи или приема каждого бита счетчик загружается величиной INT(N/2), где N – значение, сохраненное в регистрах UxBR0 и UxBR1. Счетчик перезагружает INT(N/2) каждый полупериод периода бита, обеспечивая полный период бита N BRCLK. Для данного источника тактирования BRCLK, скорость передачи определяется требуемым коэффициентом деления N:



Коэффициент деления N зачастую является нецелым числом, целочисленная часть которого может быть принята прескалером/делителем. Вторая ступень генератора скорости передачи – модулятор, используется для максимально точного учета дробной части. Коэффициент деления N в этом случае определяется так:



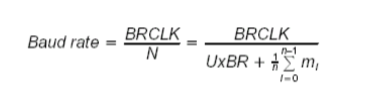
где N - получаемый коэффициент деления;

UxBR - 16-разрядное представление регистров UxBR0 и UxBR1;

i - позиция бита в фрэйме;

n - общее количество битов в фрэйме;

mi - данные каждого соответствующего модуляционного бита (1 или 0).



**Определение модуляционного значения**

Определение модуляционного значения – интерактивный процесс. Использование формулы ошибки синхронизации, начиная со стартового бита, позволяет рассчитать ошибку для каждого бита с последующей установкой или сбросом соответствующего бита модуляции. Модуляционный бит устанавливается с наименьшей выбранной и рассчитанной ошибкой следующего бита. Этот процесс продолжается до минимизации ошибок всех битов. Если фрэйм содержит более 8 бит, модуляционные биты повторяются. К примеру, 9-ый бит фрэйма использует бит модуляции 0.

**Типовые скорости передачи и ошибки**

Стандартные скорости передачи данных в бодах для UxBR и UxMCTL

приведены в таблице 4.2.2 для часового кристалла (ACLK) на 32768 Гц и для

типичного значения SMCLK 1048576 Гц.

Ошибка приема – это накопленное время в сравнении с идеальным временем загрузки сдвигового регистра в середине каждого бита. Ошибка передачи – накопленное время ошибки в сравнении с идеальным временем

периода бита.

**Прерывания USART**

USART имеет один вектор прерывания для передачи и один вектор прерывания для приема.

**Функционирование прерывания USART при передаче**

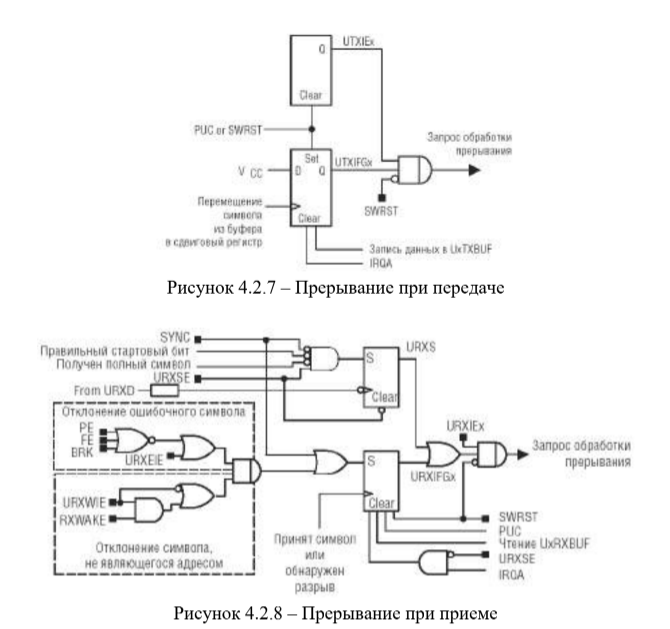
Флаг прерывания UTXIFGx устанавливается передатчиком для индикации готовности UxTXBUF к приему другого символа. Запрос прерывания генерируется, если установлены флаги UTXIEx и GIE. UTXIFGx автоматически сбрасывается при входе в обработчик прерывания или призаписи нового символа в UxTXBUF.

UTXIFGx устанавливается после PUC или при SWRST=1. UTXIExсбрасывается после PUC или когда SWRST=1 (рисунок 4.2.7).

**Функционирование прерывания USART при приеме**

Флаг прерывания URXIFGx устанавливается каждый раз при приеме символа и его загрузки в UxRXBUF. Запрос прерывания генерируется, если установлены флаги URXIEx и GIE. URXIFGx и URXIEx сбрасываются сигналом системного сброса PUC или при SWRST=1. URXIFGx сбрасывается автоматически при входе в обработчик прерывания (если URXSE=0) или при чтении данных из UxRXBUF (рисунок 4.2.8).

Форматы управляющих регистров Модуля UART в [1] стр. 240-249.



**Описание лабораторной установки**

Задания выполняются на лабораторном стенде на базе 16-ми разрядного микроконтроллера MSP430. Дополнительно в работе используется USB для соединения лабораторного стенда с ПЭВМ через последовательный интерфейс USB.

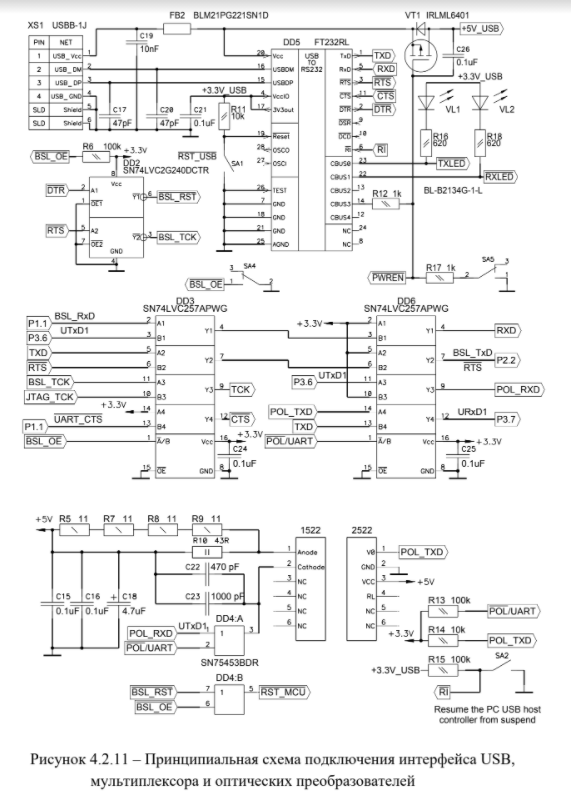
В лабораторном стенде реализовано подключение последовательного порта UART1 к интерфейсу USB и оптоволоконной линии (POL). Переключение между интерфейсами выполняется с помощью сигнала POL /UART , при высоком его уровне мультиплексор коммутирует вход и выход UART1 на USB, при низком уровне – на оптические преобразователи.

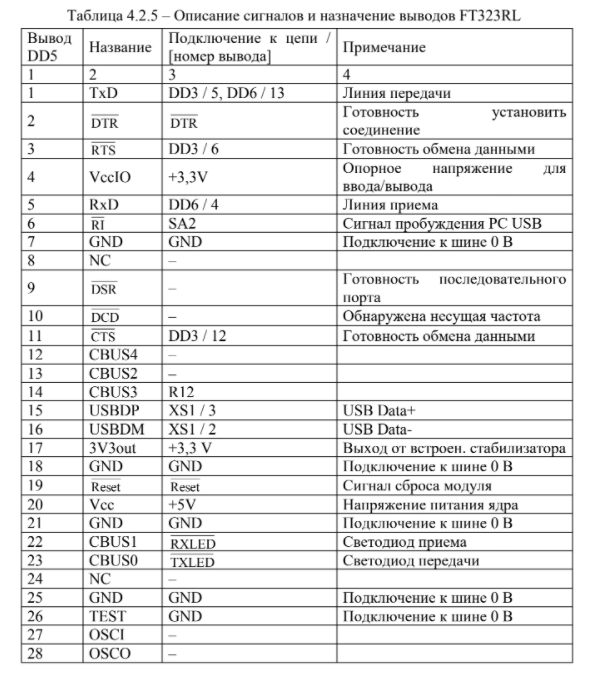
Для подключения по протоколу USB используется микросхема FT232RL,

которая обеспечивает преобразования формата данных из USB в UART. Данная микросхема содержит внутренний генератор 24 МГц и энергонезависимую память, что не требует установки внешних компонентов, как для предыдущих версий этих микросхем. Начиная с ОС Windows 2000, в ядре системы имеются драйвера, автоматически распознающие данную микросхему как дополнительный COM-порт со скоростью передачи до 3 МБод.

Оптические преобразователи HFBR1522/2522 позволяют организовать обмен данными через оптоволоконную линию связи на скорости до 1МБод. Подключение оптических преобразователей позволяет реализовать все преимущества ВОЛС: обеспечение высокой скорости передачи, гальваническую развязку, помехозащищенность и защиту данных от несанкционированного доступа.

На рисунках 4.2.9 и 4.2.10 показано размещение интерфейсов USB и ВОЛС на плате стенда. Схемы подключения интерфейса USB, оптических приемопередатчиков и мультиплексора, осуществляющего выбор необходимого модуля, показаны на рисунке 4.2.11. В таблицах 4.2.5 - 4.2.7 приведено описание сигналов этих модулей.



****

**Порядок выполнения работы**:

– включить лабораторный макет.

– запустить компилятор IAR Embedded Workbench.

– создать пустой проект.

– 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.

– 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.

– создать загрузочный модуль программы и выполнить программирование микроконтроллера.

– проверить работоспособность загруженной в микроконтроллер программы и показать результаты работы преподавателю. В случае некорректной работы разработанной программы, выполнить аппаратный сброс микроконтроллера, провести отладку исходного модуля программы и заново проверить функционирование программы.

**Содержимое файла main.c**

#include <msp430.h>

#include "stdio.h"

#include "system\_define.h"

#include "system\_variable.h"

#include "function\_prototype.h"

#include "main.h"

//байт для передачи

char tx\_byte = 0;

//для второго задания

/\*

\* main.c

\*/

void main(void){

WDTCTL = WDTPW + WDTHOLD;

Init\_System\_Clock();

Init\_System();

LCD\_init();

// задание 1 вариант 3

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

int i;

/\*

for (i = 0; i < 10; i++){

UART\_sendbyte(i);

}

UART\_off();

\*/

// задание 2

for (i=0; i<10; i++)

UART\_sendbyte(i);

UART\_off();

}**Содержимое файла sysfunc.c**

// System functions

**#include** <msp430.h>

**#include** "sysfunc.h"

// инициализация портов системы

**void** **Init\_System**()

{

P1DIR |= (nSS + nWR\_nRST + MCU\_SEL\_0 + MCU\_SEL\_1); // установка направления портов на вывод

DB\_DIR = 0x00; // шина данных настроена на ввод

}

// инициализация системы тактирования

**void** **Init\_System\_Clock**()

{

**volatile** byte i;

BCSCTL1 &= ~XT2OFF; // включение осцилятора XT2

// MCLK = XT2, SMCLK = XT2

**do** // ожидание запуска кварца

{

IFG1 &= ~OFIFG; // Clear OSCFault flag

**for** (i = 0xFF; i > 0; i--); // Time for flag to set

}

**while** ((IFG1 & OFIFG)); // OSCFault flag still set?

BCSCTL2 |= SELM\_2 | SELS; // установка внешнего модуля тактирования

}

// 2do: сделать точную задержку

**void** **wait\_1ms**(word cnt)

{

**for** (wait\_i = 0; wait\_i < cnt; wait\_i++)

**for** (wait\_j = 0; wait\_j < 1000; wait\_j++);

}

**void** **wait\_1mks**(word cnt)

{

**for** (wait\_i = 0; wait\_i < cnt; wait\_i++);

}

**Содержимое файла 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 |= SSEL0; // BRCLK = SMCLK

//U1BR0 = 107; // 32768 Гц / 115200 = 69.44 (по-умолчанию)

//if (speed == 0) U1BR0 = 107; // 32768 Гц / 38400 = 208.33

if (speed == 1) U1BR0 =109; // 32768 Гц / 300 = 109,23

U1BR1 = 0x00;

U1MCTL = 0x44; // 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

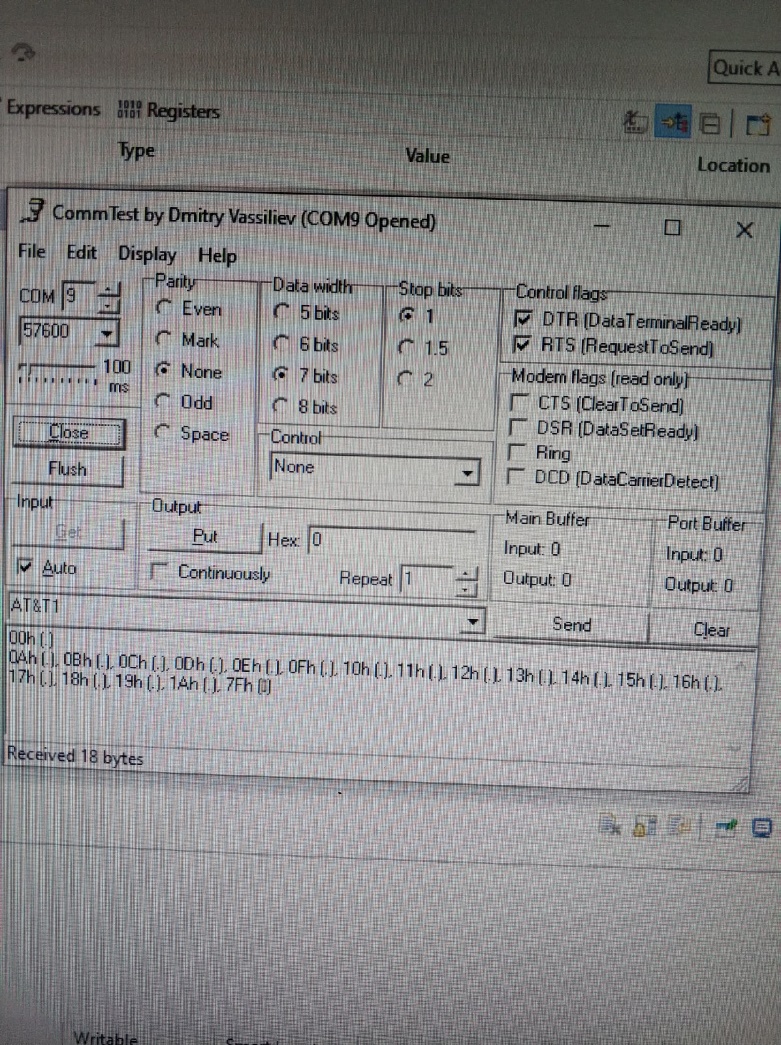
char UART\_getbyte()

{

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

return U1RXBUF; // возврат полученного байта

}



**Вывод:**  Изучили 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.