Федеральное государственное автономное образовательное учреждение высшего образования "Национальный исследовательский университет "Высшая школа экономики"

Факультет: Московский институт электроники и математики
Департамент компьютерной инженерии

Методические указания по выполнению лабораторной работы по дисциплине «Стандартные и специализированные интерфейсы»

по теме

ИЗУЧЕНИЕ ПОСЛЕДОВАТЕЛЬНЫХ ИНТЕРФЕЙСОВ

Составители
старший преподаватель ДКИ МИЭМ НИУ ВШЭ — Тув А.Л.
студент гр. БИВ 173 — Калинин Константин

Москва 2020

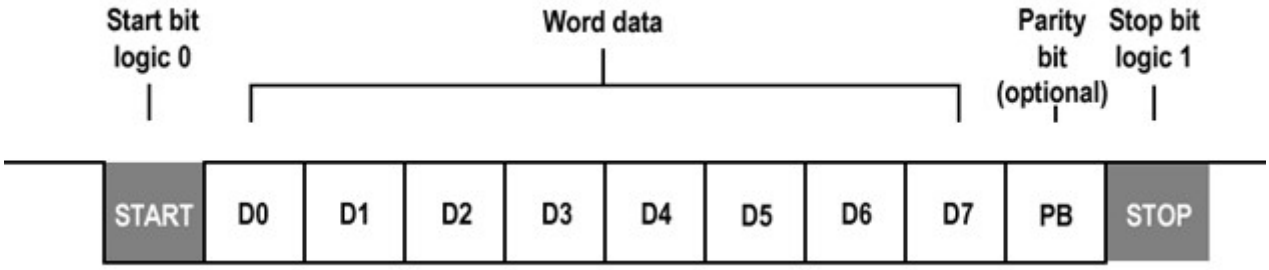
Оглавление

Цель лабораторной работы: освоить управление последовательным портом со стороны микроконтроллера PC18F4520, а также со стороны ПК, наблюдать формат сообщений протокола UART на осциллографе. Для разработки программного обеспечения на ПК использовать знания, полученные в курсе «Инструментальные средства программирования». Разработать утилиту, выполняющую функции терминала (отправка кодов нажатых клавиш из ПК по последовательному порту, прием данных из последовательного порта). Для разработки компьютерной части задания использовать фреймворк QT и IDE QtCreator.

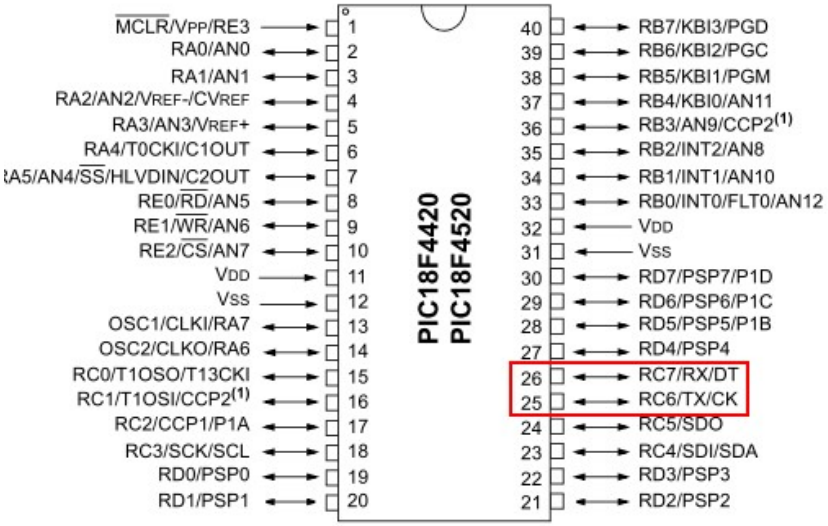
1. Описание протокола UART (Universal Asynchronous Receiver Transmitter)

Универсальный асинхронный приёмопередатчик (УАПП, англ. Universal Asynchronous Receiver-Transmitter, UART) — узел вычислительных устройств, предназначенный для организации связи с другими цифровыми устройствами. Преобразует передаваемые данные в последовательный вид так, чтобы было возможно передать их по одной физической цифровой линии другому аналогичному устройству. Метод преобразования хорошо стандартизован и широко применяется в компьютерной технике (особенно во встраиваемых устройствах и системах на кристалле (SoC)).

Представляет собой логическую схему, с одной стороны подключённую к шине вычислительного устройства, а с другой имеющую два или более выводов для внешнего соединения. UART может представлять собой отдельную микросхему (например, Intel I8251, I8250) или являться частью большой интегральной схемы (например, микроконтроллера). Используется для передачи данных через последовательный порт компьютера. UART имеет выход для передачи данных (TxD) и вход приема данных (RxD). Формат передаваемых данных приведен на рисунке.



2. Краткие сведения о модуле USART микроконтроллера PIC18F4520.



Микроконтроллер PIC18F4520 имеет на борту один расширенный модуль приемника-передатчика. Выходы передатчика и приемника модуля мультиплексированы с разрядами 7 и 6 порта С

Расширенный универсальный синхронный асинхронный модуль приемника-передатчика (EUSART) является одним из двух модулей последовательного ввода / вывода. В модуле Enhanced USART реализованы дополнительные функции, включая автоматическое определение скорости передачи и калибровка, автоматическое пробуждение при получении Sync Break и 12-битного символа прерывания. Полное описание работы модуля приведено в [].

Регистр статуса и контроля передачи TXSTA

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R-1	R/W-0
CSRC	TX9	TXEN(1)	SYNC	SENDB	BRGH	TRMT	TX9D
bit 7							bit 0

bit 7 CSRC: Clock Source Select bit

Асинхронный режим – не имеет значения

Синхронный режим :

1 = Master mode (clock generated internally from BRG)
0 = Slave mode (clock from external source)

bit 6 TX9: 9-Bit Transmit Enable bit

1 = Будет использоваться 9ти битный обмен данными
0 = Будет использоваться 8ми битный обмен данными

bit 5 TXEN: Разрешение передачи.

1 = Передача разрешена
0 = Передача запрещена

bit 4 SYNC: EUSART Режим работы модуля

1 = Синхронный режим
0 = Асинхронный режим

bit 3 SENDB: Send Break Character bit

Асинхронный режим

1 = Отправить синхронизующий символ следующей передаче (сбрасывается аппаратно после завершения)
0 = Передача синхросимвола завершена

Синхронный режим::

Не имеет значения.

bit 2 BRGH: Выбор множителя скорости передачи

Асинхронный режим

1 = Высокая скорость
0 = Низкая скорость

Синхронный режим:

Не используется.

bit 1 TRMT: Transmit Shift Register Status bit

1 = сдвиговый регистр передатчика (TSR) пуст
0 = сдвиговый регистр передатчика (TSR) полон

bit 0 TX9D: 9th Bit of Transmit Data

Девятый бит передаваемых данных

Регистр состояния и контроля приемника RCSTA

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R-0	R-0	R-x
SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D
bit 7							bit 0

bit 7 SPEN: Бит включения модуля последовательного порта

1 = Serial port enabled (configures RX/DT and TX/CK pins as serial port pins)
0 = Serial port disabled (held in Reset)

bit 6 RX9: Выбор длины принимаемого байта

1 = Выбор длины байта в 9 бит
0 =Выбор 8ми битного приема

bit 5 SREN: Разрешение одиночного приема

Асинхронный режим:

Не имеет значения.

Синхронный режим – Master:

1 = Разрешение одиночного приема
0 = Запрещение одиночного приема

Бит сбрасывается, по окончании приема .

Синхронный режим – Slave:

Не имеет значения.

bit 4 CREN: Разрешение последовательного приема

Асинхронный режим:

1 = Включение приемника.
0 = Выключение приемника

Синхронный режим:

1 = разрешает непрерывный прием до тех пор, пока бит разрешения CREN не будет сброшен (CREN имеет приоритет над SREN)

0 = Запрещает непрерывный прием

bit 3 ADDEN: Address Detect Enable bit

Асинхронный реим 9-Bit (RX9 = 1):

1 = Включает обнаружение адреса, разрешает прерывание и загружает приемный буфер, когда установлен RSR <8>

0 = Отключает определение адреса, все байты принимаются, девятый бит может использоваться как бит четности

Asynchronous mode 9-Bit (RX9 = 0):

Не используется.

bit 2 FERR Ошибка кадра

1 = Ошибка кадра была (можно очистить, прочитав регистр RCREG и получив следующий действительный байт)
0 = Ошибки кадра не было.

bit 1 OERR: Бит переполнения

1 = Ошибка была (может быть очищена очисткой бита, CREN)
0 = Ошибки не было.

bit 0 RX9D: 9th Bit of Received Data

9й бит принятых данных

Регистр BAUDCON

R/W-0	R-1	R/W-0	R/W-0	R/W-0	U-0	R/W-0	R/W-0
ABDOVF	RCIDL	RXDTP	TXCKP	BRG16	—	WUE	ABDEN
bit 7							bit 0

bit 3 BRG16: Выбор разрядности генератора скорости. Описание регистров генератора скорости ниже

1 = 16-bit Baud Rate Generator – SPBRGH and SPBRG

0 = 8-bit Baud Rate Generator – SPBRG only (Compatible mode), SPBRGH value ignored

BRG - это специальный 8-битный или 16-битный регистр генератора скорости модуля EUSART. По умолчанию BRG работает в 8-битном режиме; установка бита BRG16 (BAUDCON <3>) выбирает 16-битный режим.

Значение в регистрах SPBRGH: SPBRG управляет периодом автономного таймера. В таблице приведены формулы расчета скорости передачи в разных режимах работы модуля EUSART.

Configuration Bits			BRG/EUSART Mode	Baud Rate Formula
SYNC	BRG16	BRGH		
0	0	0	8-Bit/Asynchronous	Fosc/[64 (n + 1)]
0	0	1	8-Bit/Asynchronous	Fosc/[16 (n + 1)]
0	1	0	16-Bit/Asynchronous	
0	1	1	16-Bit/Asynchronous	Fosc/[4 (n + 1)]
1	0	x	8-Bit/Synchronous	
1	1	x	16-Bit/Synchronous	

Запись нового значения в регистры SPBRGH: SPBRG вызывает сброс (или очистку) таймера BRG.

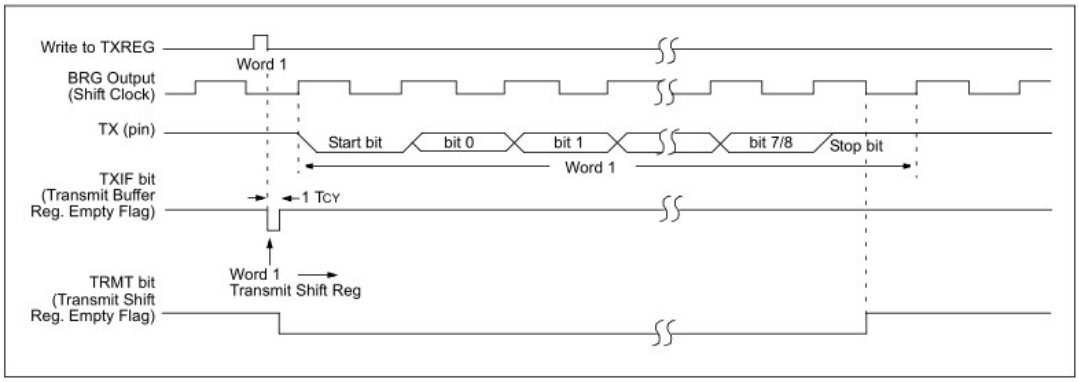
n – значение в регистрахSPBRGH::SPBRG.

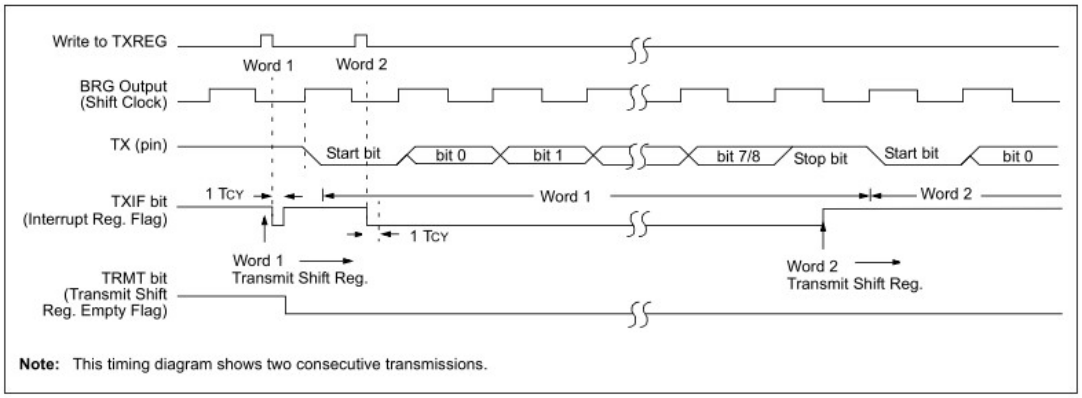
При выборе значения n необходимо задать желаемую скорость в бит/сек из ряда стандартных значений, применить соответствующую формулу () и округлить полученный результат. Для оценки погрешности по полученному провести обратные вычисления и получить реальную скорость передачи. Для обеспечения устойчивого обмена данными отклонение реальной скорости от стандартной должно быть меньше 5 %.

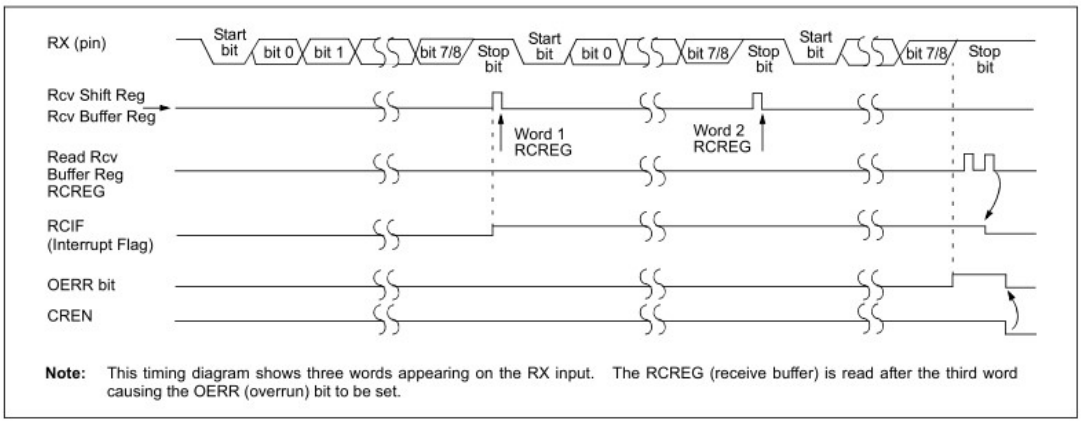
Регистр TXREG – регистр передаваемых данных. Запись в этот регистр инициирует начало передачи, при условии, что модуль включен.

Регистр RCREG – регистр принимаемых данных. Если принятые данные не будут прочитаны к приходу следующих, будет установлен флаг **OERR<RCSTA.1>.**

Временные диаграммы передачи и приема приведены на эпюрах







Краткие сведения о прерываниях, необходимых для настройки работы последовательного порта по прерываниям.

Подробно организация системы прерываний микроконтроллера PIC 18F4520 описана в [], менее подробно – в []. Ниже приведен лишь перечень бит, необходимых для настройки прерываний модуля EUSART.

Регистр INTCON

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-x
GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF(1)
bit 7							bit 0

Регистр флагов прерываний PIR1.

R/W-0	R/W-0	R-0	R-0	R/W-0	R/W-0	R/W-0	R/W-0
PSPIF(1)	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF
bit 7							bit 0

bit 5 RCIF: EUSART Receive Interrupt Flag bit

1 = буфер приема EUSART, RCREG, заполнен (очищается при чтении RCREG)

0 = буфер приема EUSART пуст

bit 4 TXIF: EUSART Transmit Interrupt Flag bit

1 = буфер передачи EUSART, TXREG, пуст (очищается при записи TXREG)

0 = буфер передачи EUSART заполнен.

Регистр разрешения прерываний PIE1.

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
PSPIE(1)	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE
bit 7							bit 0

bit 5 RCIE: EUSART Receive Interrupt Enable bit

1 = разрешает прерывание приема EUSART

0 = Отключает прерывание приема EUSART

bit 4 TXIE: EUSART Transmit Interrupt Enable bit

1 = разрешает прерывание передачи EUSART

0 = Отключает прерывание передачи EUSART

3. Настройка QT для работы с последовательным портом

Для работы с последовательным портом в Qt используется класс QSerialPort. Для его использования в проекте необходимо добавить в файл .pro строчку

```
QT += serialport
```

Пример инициализации и настройки порта COM1

```
QSerialPortport("COM1"); //Подключаемся к порту COM1
port.setBaudRate(QSerialPort::Baud4800); //Установка
скорости в 4800 бод
port.setDataBits(QSerialPort::Data5); //Установка
количества бит данных равное 5
port.setStopBits(QSerialPort::OneStop); //Установка 1
стопбита
port.open(QSerialPort::ReadWrite); //Открыть для
пересылки и получения данных
```

Для пересылки данных используется метод один из методов write.

Для обработки ошибок можно использовать метод waitForBytesWritten(), который ждет завершения отправки данных и возвращает true если она прошла успешно и false если в процессе возникли ошибки.

Пример пересылки одного байта:

```
char *oneByte = "A";
port.write(oneByte); //Посылаем
if(!port.waitForBytesWritten()) { //Ждем, пока
пересылка завершится
  //Обработка ошибки
}
```

Пример получения данных:

```
charbuffer[50];

intnumRead;

if(port.bytesAvailable() > 0) { //Проверяем, что нам
что-то пришло

numRead = port.read(buffer, 50); //Читаем максимум 50
байт и кладем их в buffer

} //В numRead окажется количество прочитанных байт
```

4. Настройка отладочной платы UNI DS-6

Отладочная плата UNI DS6 содержит 2 модуля UART – USB. Схема модулей показана на рисунке().

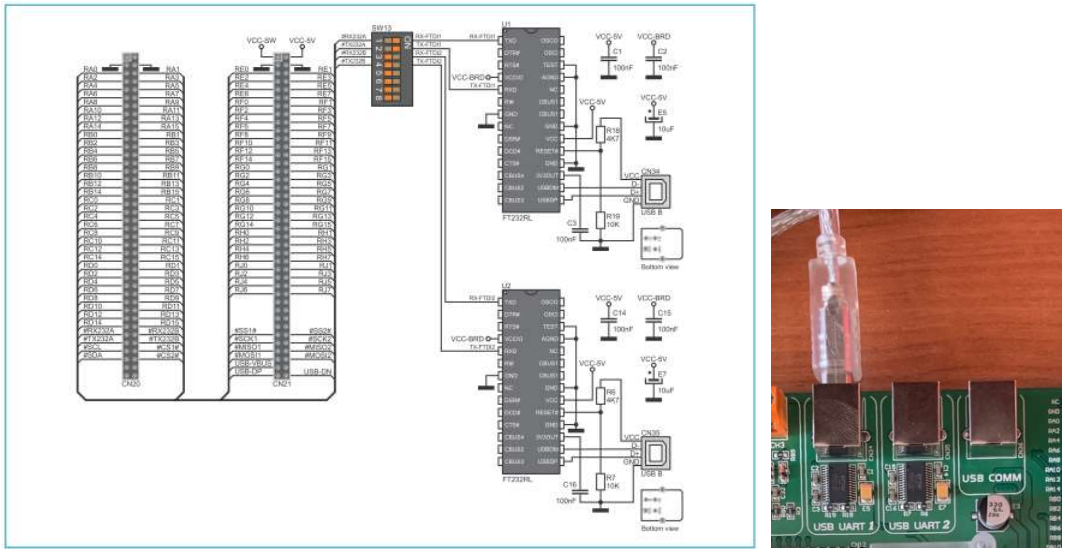


Рисунок 1 Принципиальная электрическая схема модулей UART1, UART2 и расположение модулей на отладочной плате UNI DS6

Для корректной работы с UART1 переключатель SW13 выставить в положение, указанное на рисунке ().

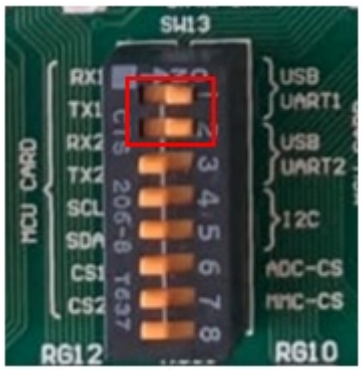


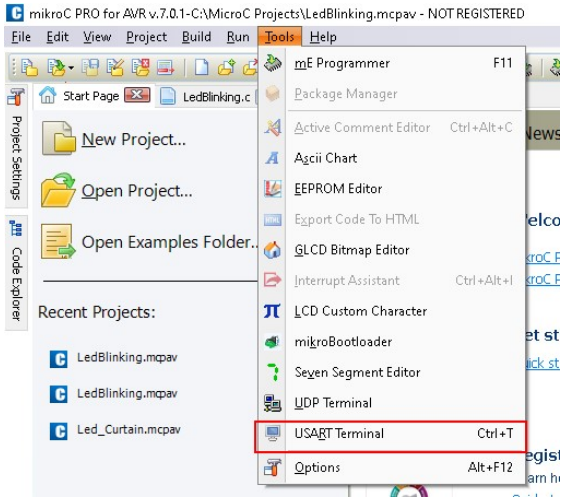
Рисунок 1. Положение переключателей для корректной работы UART 1

Для соединения с ПК использовать кабель USB A-B.

5. Задание

С помощью microCproForPIC разработать программу, которая должна обладать следующими возможностями:

1. Настраивать параметры UART: скорость работы, количество стоп-бит. (Используйте скорость 19200 или 57600 baud по указанию преподавателя, длина 8 бит, 1 стоп-бит);
2. Отправлять символы ‘0’ – ‘5’ в последовательный порт при нажатии на кнопки RA0 – RA5 соответственно;
3. Принимать данные из последовательного порта и в зависимости от полученного числа отобразить это на светодиодах RB0 - RB7. Например, если было получен символ, код которого равен числу 255, то зажечь все светодиоды, если 0, то никакие и т.д). Для того, чтобы работа была оценена на оценку выше 7 баллов необходимо организовать прием данных последовательного порта по прерываниям. Задания 2 и 3 оформить в рамках одного проекта micro C.
4. Доработать ПО таким образом, чтобы по нажатию кнопок RA0-RA5 выдавать в последовательный порт нультерминальные строки длиной от 7 символов. Для отличной оценки, необходимо организовать передачу байт в последовательный порт по прерываниям. Задание оформить отдельным проектом в micro C.
5. Для желающих получить оценки 9 или 10 необходимо предложить и реализовать способ обмена данными между ПК и микроконтроллером пакетами, которые бы гарантировали целостность принимаемых данных. Необходимо предложить, обосновать и реализовать на ПК и микроконтроллере требуемый вариант обмена. Задание оформить отдельным проектом в micro C.
6. Подключить модуль расширения RS485 к отладочной плате UNI DS6 и модуль USB-RS485 к ПК, предварительно взяв их у преподавателя. Соединить между собой отладочную плату и ПК интерфейсом Rs485. Изменить ПО заданий 2 и 3 микроконтроллера таким образом, чтобы передавать и принимать последовательные данные по интерфейсу RS485.*
7. При выполнении заданий запрещено пользоваться библиотечными функциями
8. Для отладки разрабатываемого ПО использовать терминал, поставляемый в составе IDE micro C. Вызов терминала, как показано можно провести из пункта меню «Tools», как показано на рисунке().



С помощью IDE Qt Creator и фреймворка Qt разработать ПО с графическим пользовательским интерфейсом, позволяющее:

1. Настраивать параметры последовательного порта (выбирать порт, настраивать скорость обмена)
2. Выводить в отдельное окно данные, полученные из последовательного порта.
3. В другом окне отображать символы нажатых клавиш клавиатуры.
4. Разработанное ПО должно позволять настраивать формат отображения данных в окнах: ASCII - BIN - HEX.

Результатом работы является разработанные и скомпилированные приложения, запускающиеся и работающие на отладочной плате UNI DS6 и ПК, в соответствие с заданием. Для подтверждения отличной оценки преподаватель вправе дать дополнительное задание, заключающееся в модернизации разработанного алгоритма или кода программы, которые должны быть выполнено в ограниченное время.

Критериями оценки являются:

1. Соответствие решений заданию, отсутствие ошибок в решениях.
2. Своевременность выполнения работы
3. Ответы на дополнительные вопросы и задания.