|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

Факультет: «Специальное машиностроение»

Кафедра: СМ11 «Подводные аппараты и роботы»

**Лабораторная работа № 3**

по курсу «Электронные устройства мехатронных и робототехнических систем»

Вариант 8

Выполнил: Ионин Даниил

Группа: СМ11-61Б

Проверил(а):

Москва, 2024 г.

ОГЛАВЛЕНИЕ

[ОГЛАВЛЕНИЕ 2](#_Toc166253696)

[Введение 3](#_Toc166253697)

[Тема работы: 3](#_Toc166253698)

[Цель работы: 3](#_Toc166253699)

[Условные обозначения 4](#_Toc166253700)

[Блок схема ATMEGA16 5](#_Toc166253701)

[Работа со стендом EasyAVR5 6](#_Toc166253702)

[Теоретическая часть 8](#_Toc166253703)

[Модуль USART 8](#_Toc166253704)

[Интерфейс RS232 (COM-порт) 9](#_Toc166253705)

[Протокол последовательной передачи данных 10](#_Toc166253706)

[Порядок обмена информацией. 11](#_Toc166253707)

[Практическая часть 13](#_Toc166253708)

[Постановка задачи 13](#_Toc166253709)

[Блок-схема программы 14](#_Toc166253710)

Введение

Тема работы:

Работа с универсальным синхронно-асинхронным передатчиком

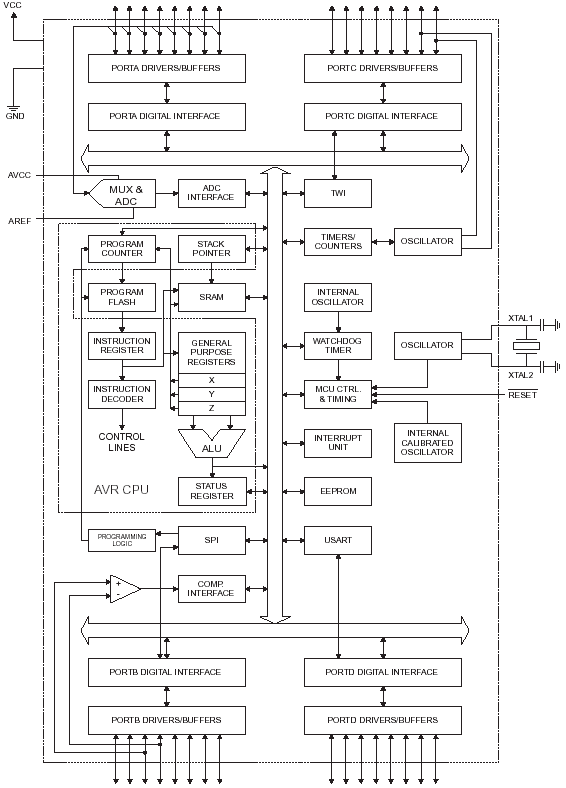
Цель работы:

Изучить работу универсального синхронно-асинхронного приемопередатчика. Изучить принципы сопряжения микроконтроллерных систем с ПК.

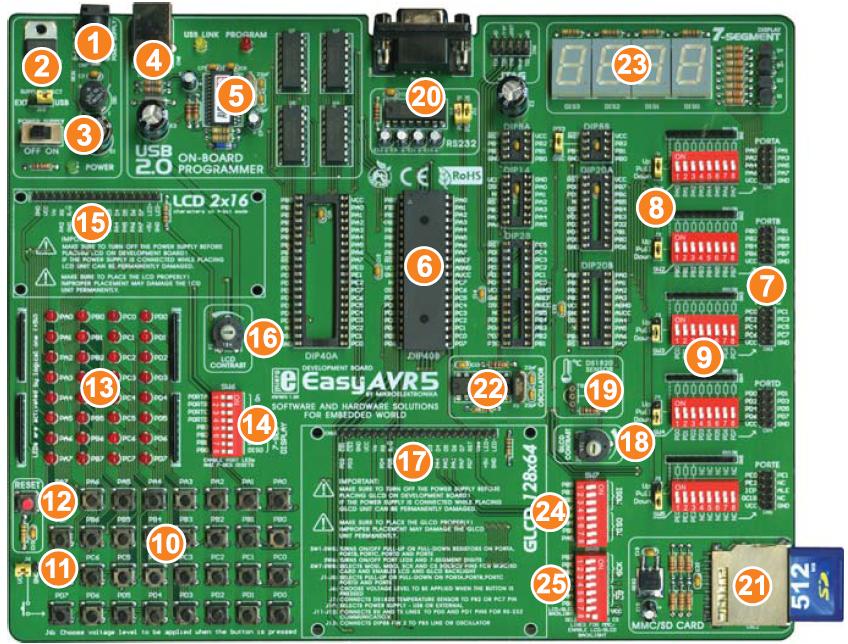
Условные обозначения

* Flash ROM - объем энергонезависимой памяти программ (в килобайтах);
* EEPROM - объем энергонезависимой памяти данных (в байтах);
* RAM - объем статической памяти данных (в байтах);
* External RAM - возможность подключения к микроконтроллеру дополнительной микросхемы внешней статической памяти данных (в килобайтах);
* ISP - возможность программирования микроконтроллера в системе (на целевой плате) при основном напряжении питания;
* SPM - функция самопрограммирования Flash ROM памяти микроконтроллера в системе без участия внешнего программатора;
* JTAG - встроенный JTAG - интерфейс;
* I/O (pins) - максимальное количество доступных линий ввода / вывода;
* Timer(s) 8/16 bit - количество и разрядность таймеров/счетчиков;
* USI - универсальный коммуникационный интерфейс;
* AC - аналоговый компаратор;
* ADC (channels) - количество каналов аналого-цифрового преобразования;
* Internal RC - наличие внутренней RC-цепочки для автономной работы микроконтроллера (без внешнего источника опорной частоты);
* WDT - сторожевой таймер;
* BDC - аппаратный программируемый блок защиты от сбоев при внезапном (в том числе и кратковременном) пропадании напряжения питания микроконтроллера;
* UART - асинхронный последовательный приемопередатчик;
* SPI - синхронный трехпроводной последовательный интерфейс;
* I2C - двухпроводной последовательный интерфейс;
* RTC - система реального времени;
* PWM (channels) - количество независимых каналов широтно - импульсной модуляции;
* Command Set - количество различных инструкций в системе команд микроконтроллера;
* Vcc - диапазон рабочих напряжений питания (в Вольтах);
* Clock - диапазон рабочих частот (в мегагерцах);

Блок схема ATMEGA16

 Рисунок 1 – Блок схема ATmega16

Работа со стендом EasyAVR5

 Рисунок 2 – Внешний вид стенда EasyAVR5

1 – разъем для подключения внешнего питания 8–16В;

2 – выбор между внешним питанием и питанием от USB;

3 – выключатель;

4 – разъем USB;

5 – встроенный внутрисхемный программатор;

6 – панельки для установки различных типов МК;

7 – разъемы для доступа к портам МК;

8 – джамперы для выбора типа подтягивающих резисторов (к 1 или к 0);

9 – группы переключателей для подключения подтягивающих резисторов к выводам МК;

10 – 32 кнопки, подключенные к выводам МК;

11 – джампер выбора высокого или низкого состояния при нажатии кнопки:

12 – кнопка сброса;

13 – светодиоды, каждый подключенный к выводу МК;

14 – группа переключателей, подключающая или отключающая светодиоды к МК;

15 – разъем для подключения символьной ЖК-матрицы;

16 – потенциометр для регулировки яркости символьной ЖК-матрицы;

17 – разъем для подключения графической ЖК-матрицы;

18 – потенциометр для регулировки яркости графической ЖК-матрицы;

19 – разъем для подключения датчика температуры DS1820;

20 – разъем порта RS232;

21 – слот для подключения MMC/SD карт памяти;

22 – встроенный генератор частоты;

23 – 4 семисегментных индикатора;

24 – группа переключателей для разрешения внутрисхемного программирования;

25 – группа переключателей для разрешения работы с MMC/SD картами памяти.

Теоретическая часть

Модуль USART

Через модуль USART (универсальный синхронно-асинхронный передатчик) микроконтроллер осуществляет прием и передачу информации, представленной последовательным кодом, поэтому его часто называют также последовательным портом. С помощью этого модуля микроконтроллер может обмениваться данными с различными внешними устройствами. Блок схема модуля USART представлена на рисунке 3.

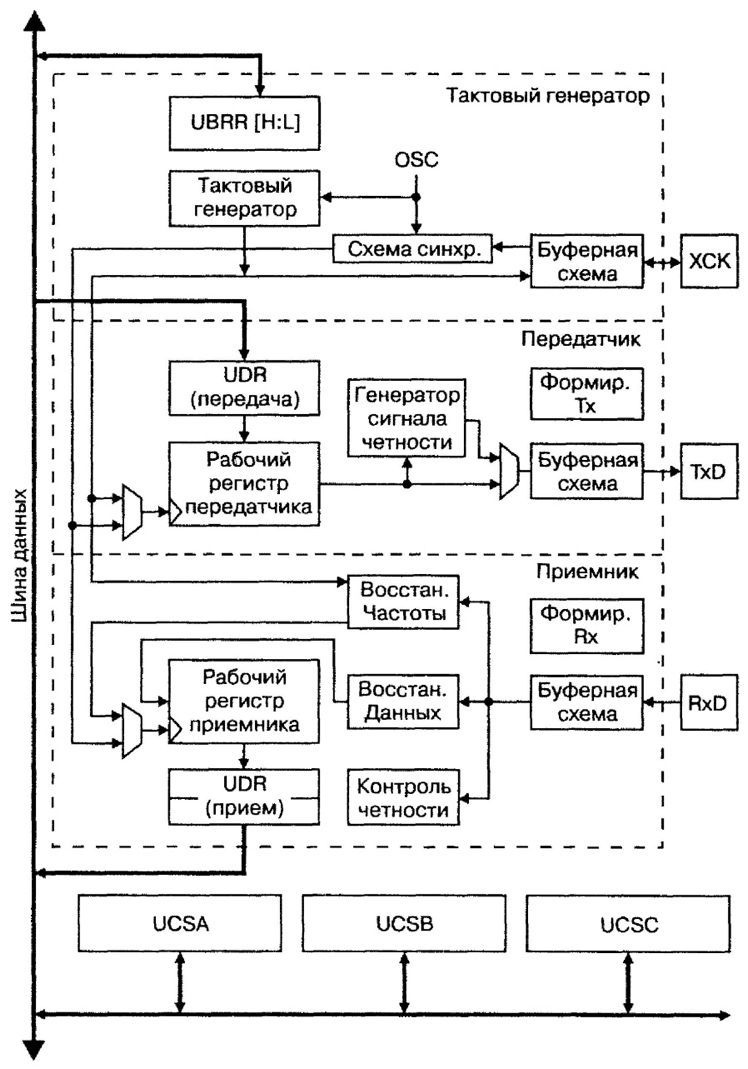


Рисунок 3 – модуль USART

Скорость передачи данных может варьироваться в широких пределах, причем высокие скорости передачи могут быть достигнуты даже при относительно низкой тактовой частоте микроконтроллера.

Для взаимодействия с программой в модуле предусмотрены 3 раздельных прерывания, запрос на которые генерируется при наступлении следующих событий: «передача завершена», «регистр данных передатчика пуст» и «прием завершен».

Интерфейс RS232 (COM-порт)

Интерфейс RS-232C предназначен для подключения к компьютеру стандартных внешних устройств (принтера, сканера, модема, мыши и др.), а также для связи компьютеров между собой. Данные в RS-232C передаются в последовательном коде побайтно. Каждый байт обрамляется стартовым и стоповыми битами. Данные могут передаваться как в одну, так и в другую сторону (дуплексный режим).

Компьютер имеет 25-контактный (DB25P) или 9-контактный (DB9P) разъем для подключения RS-232C. Скорость передачи по RS-232C может выбираться из ряда: 110, 150, 300, 600, 1200, 2400, 4800, 9600, 19200, 38400, 57600, 115200 бит/с.

Все сигналы RS-232C передаются специально выбранными уровнями, обеспечивающими высокую помехоустойчивость связи (рисунок 7.). Отметим, что данные передаются в инверсном коде (логической единице соответствует низкий уровень, логическому нулю - высокий уровень).

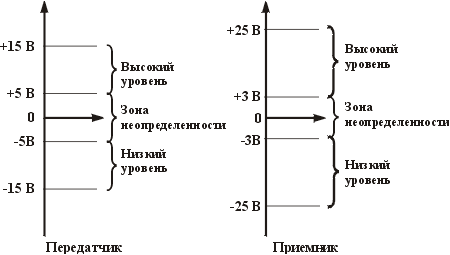


Рисунок 4 – Уровни сигналов RS-232C на передающем и принимающем концах линии связи

Для подключения произвольного УС к компьютеру через RS-232C обычно используют трех- или четырехпроводную линию связи, но можно задействовать и другие сигналы интерфейса. Для преобразования уровней сигналов с выхода модуля USART микроконтроллера к уровням интерфейса RS232 используются специальные микросхемы преобразования уровней (например, MAX232).

Протокол последовательной передачи данных

При передаче данных необходимо, чтобы имелся протокол обмена, согласно которому участвующие в обмене устройства будут интерпретировать данные. Кроме этого необходимо оговорить количество информационных бит в кадре данных, количество стоп-бит и т.д. В предлагаемой лабораторной работе принят следующий протокол обмена. Передача/прием данных осуществляется со следующими параметрами:

а) скорость обмена 19200/38400 бит/сек;

б) формат посылки 10/11 бит информации:

- старт-бит;

8 бит данных;

- бит контроля четности (может отсутствовать);

- один стоп бит.

Порядок обмена информацией.

С персонального компьютера, выполняющего роль ведущего в обмене (master), осуществляет передача кадра запроса выполнения простейшей арифметической операции. Кадр состоит из следующей последовательности:

1) Байт кода операции:

- 0x01 – сложение 2-х однобайтных двоичных чисел без знака (0 до 255);

- 0х02 – вычитание 2-х однобайтных двоичных чисел без знака (0 до 255);

- 0х03 – умножение 2-х однобайтных двоичных чисел без знака (0 до 255);

- 0х04 – деление 2-х однобайтных двоичных чисел без знака (0 до 255).

2) Байт первого операнда (1-ое слагаемое, уменьшаемое, множимое, делимое).

3) Байт второго операнда (2-ое слагаемое, вычитаемое, множитель, делитель).

4) Байт контрольной суммы с инверсией.

Микроконтроллер, ведомый (slave) принимает запрос на выполнение операции, осуществляет выполнение арифметической операции и возвращает кадр ответа выполненной операции. Кадр ответа состоит из следующей последовательности:

1) Старшего байта результата выполнения операции/(частное);

2) Младший байт результата выполнения операции/(остаток от деления);

3) Байт контрольной суммы с инверсией.

Передача запроса на вычисление и выдача ответа осуществляются асинхронно по нажатию соответствующих кнопок на экране монитора персонального компьютера и макетной платы.

Практическая часть

Постановка задачи

1. Выполните проверку работы программы в симуляторе AVR Studio. Обратите особое внимание проверке в симуляторе подпрограмм обработки прерываний Tx Complete, Data register Empty, RX Complete;

2. Выполните: программирование стенда EasyAVR5. Запустите программу AVRStend на ПК. Выполните передачу команды с ПК и прием ответа от стенда EasyAVR5. Проверьте правильность принятых и переданных данных на семисегментных индикаторах стенда EasyAVR5.

3. Доработайте программу сократив необоснованно большую паузу между нажатиями кнопки «Просмотр».

4. Проведите эксперимент по изменению и передаче команд арифметических операций и правильности их приема стендом EasyAVR5.

5. Измените скорость передачи в программе МК и программе AVRStend, проверьте и добейтесь правильной работы измененной программы.

6. Добавьте бит паритета в программах МК и AVRStend и проверьте, и добейтесь правильной работы измененной программы.

Полученный исправленный код находится в приложении 1.

Блок-схема программы

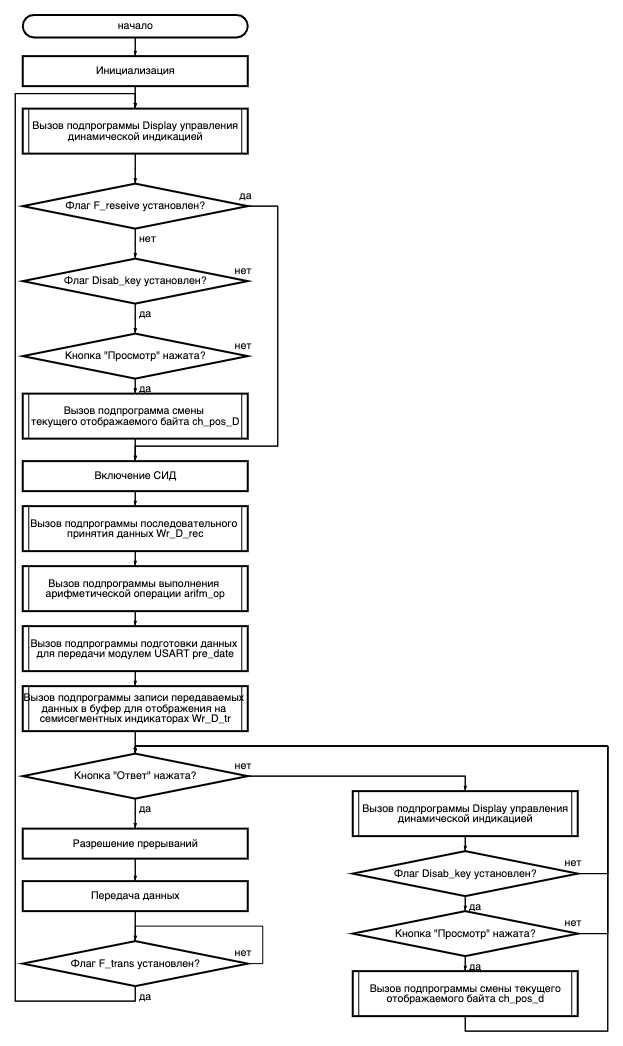


Рисунок 5 – Блок схема программы

Код программы

Код программы управления микроконтроллером:

|  |
| --- |
| ;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  ;\* Designer Mechtcherjakowa R.I.  ;\* Version: 1.0  ;\* Date 15.03.2010  ;\* Title: contr\_UART.asm  ;\* Device ATmega16  ;\* Clock frequency:Частота кв.резонатора 9,216 mHz  ;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  ; учебная  ;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  ;Программа осуществляет обмен по интерфейсу RS-232 между персональным компьютером  ;и макетной платой.В макетной плате устанавливается микроконтроллер ATmega16  ;с программируемым последовательным портом USART. Преобразование в интерфейс  ;RS-232 выполняется микросхемой Max232, установленной в макетной плате.  ;Передача/прием данных по USART осуществляется со следующими параметрами:  ;а)скорость обмена 19200бит/сек;  ;б)формат посылки 11 бит информации:  ; - старт-бит;  ; - 8 бит данных;  ; - бит контроля четности - дополнение до нечетности;  ; - один стоп бит.  ;Выводы USART(RXD) PD0 (вход),(TXD) PD1(выход)  ;При приеме кадра запроса загорается светодиод СИД "Запрос", подключенный  ;к PB7 ( выход)  ;Кадры принятых и подготовленных к ответу данных можно просмотреть на  ;семисегмент.индикаторах последовательно байт за байтом,нажимая кнопку  ;"Просмотр",подключенную к PD4( байт1, байт2, байт3, байт4,байт5,байт6,байт7,байт1..).  ;Крайний левый индикатор показываетномер номер просматриваемого байта  ;Кнопка "Ответ", инициализирующая ответную посылку микроконтр.подключена к PB7,при этом  ;СИД гаснет  ;семисегментные индикаторы подключены к PС0-PС7-выходы,семигегментный индикатор к PС0-PС7  ;PC0-a,PC1-b,PС2-c,PС3-d,PС4-e,PС5-f,PС6-g,PС7-h  ;сигналы выбора индикатора PB3-номер байта,PB2 - сотни,PB1-десятки,PB0 - единицы- выходы  ;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  ;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  .include "m16def.inc"  ; присоединение файла описаний;  .list ;включение листинга  ;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  ;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  ; Register Variables  ;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  .def temp\_L =R16  .def temp\_H =R17  ;  .def Number =R18;регистр, используемый для хранения адреса символа по таблице перекодировки, отображаемого на семисегментном индикаторе "номер байта";  .def Hundreds =R19;  .def Tens =R20  .def Ones =R21;регистры, используемые для хранения адреса символа по таблице перекодировки, отображаемого на семисегментных индикаторах  ;  .def Disp\_Numb =R22;регистр, используемый для хранения номера активного семисегментного индикатора  .def Disp\_Count =R23;регистр, используемый в качестве счетчика времени активности каждого из семисегментных индикаторов при динамической индикации  ;  .def res\_a\_op =R8;буфер хранения промежуточных результатов при выполнении арифметических операций  .def Time =R25;счетчик переполн.Т0(1024\*255/9216000\*Xotc=1cek(Xotc=35)  ;  .def Cou\_Rec =R4;счетчик принятых байт  .def Cou\_Tran =R5;счетчик переданных байт  .def c\_sumREC =R6;контр сумма прин. байт (без переноса)  .def c\_sumTRAN =R7;контр сумма переданных байт (без переноса)  .def n\_ar\_op =R24 ;регистр, используемый в качестве счетчика числа циклов суммирования при выполнении операции умножения  ;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  .def Byte\_fl =R2; байт флагов  ;-------------------  .equ Disab\_Key =0; бит запрещения опроса кнопки "Просмотр"  .equ F\_receive =1;флаг принятого запроса  .equ F\_trans =2;флаг завершения передачи  ;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  ; Constants  ;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  .equ Val\_dispCount=50;величина константы,опр. время вкл индикатора  .equ CID =7;PDB подключение СИД  .equ ENV =7;Кнопка "Ответ", подключена к PD7  ;========================================  ;Количество байт обмена с ПЭВМ  ;----------------------------------------  .equ VAL\_TR =3;константа, определяющая количество бит к кадре, передаваемом модулем USART;  .equ VAL\_REC =4;константа, определяющая количество бит к кадре, принимаемом модулем USART;  .equ VAL\_time =30;35 переполнений соответствуют 1 сек  ;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  ;Variable  ;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  .DSEG ;буферы приема и передачи хранятся в ОЗУ (SRAM)  ;  varBuf\_Rxd: .BYTE 8;буфер приема(4 байта)  ;  varBuf\_Txd: .BYTE 8;буфер передачи (3 байта)  ;  varBuf\_disp: .BYTE 8;буфер данных,выводимых на дисплей (7 байт)  ;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  .cseg  .org $0000  rjmp Init  ;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  ;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  .org INT0addr;=$002 ;External Interrupt0 Vector Address  reti;  ;----------------  .org INT1addr;=$004 ;External Interrupt1 Vector Address  reti  .org OC2addr; =$006 ;Output Compare2 Interrupt Vector Address  reti  .org OVF2addr;=$008 ;Overflow2 Interrupt Vector Address  reti  .org ICP1addr;=$00A ;Input Capture1 Interrupt Vector Address  reti  .org OC1Aaddr;=$00C ;Output Compare1A Interrupt Vector Address  rjmp Time\_OUT; Прерывание по совпадению таймера  .org OC1Baddr;=$00E ;Output Compare1B Interrupt Vector Address  reti  .org OVF1addr;=$010 ;Overflow1 Interrupt Vector Address  reti  .org OVF0addr;=$012 ;Overflow0 Interrupt Vector Address  rjmp time\_d\_k ;прерывание по переполнению таймера  .org SPIaddr; =$014 ;SPI Interrupt Vector Address  reti  .org URXCaddr;=$016 ;UART Receive Complete Interrupt Vector Address  rjmp REC\_date; прием кадра завершен UART  .org UDREaddr;=$018 ;UART Data Register Empty Interrupt Vector Address  rjmp B\_TRANS; буфер передачи пуст UART  .org UTXCaddr; =$01A ;UART Transmit Complete Interrupt Vector Address  rjmp TRANdate; Передача кадра завершена UART  .org ADCCaddr; =$01C ;ADC Interrupt Vector Address  reti  .org ERDYaddr; =$01E ;EEPROM Interrupt Vector Address  reti  .org ACIaddr; =$020 ;Analog Comparator Interrupt Vector Address  reti  .org TWIaddr; =$022 ;Irq. vector address for Two-Wire Interface  reti  .org INT2addr; =$024 ;External Interrupt2 Vector Address  reti  .org OC0addr; =$026 ;Output Compare0 Interrupt Vector Address  reti  .org SPMRaddr; =$028 ;Store Program Memory Ready Interrupt Vector Address  reti  ;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  ; Start Of Main Program  ;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  Init:  ldi temp\_L,LOW(RAMEND);выбор вершины стека  out SPL, temp\_L;Указатель стека  ldi temp\_L,HIGH(RAMEND)  out SPH,temp\_L  ;  ; ------Инициализация портов В/B  ;  ldi temp\_L,0b11111111;(PB1-PB7)-выходы,  out DDRB,temp\_L  ldi temp\_L,0b00001000;выбраны PB3-номер байта (1-й индикатор)  out PORTB,temp\_L  ;  ldi temp\_L,0b00000010;PD1-выход(Txd),(PD0,PD2-PD7)-входы  out DDRD,temp\_L  ldi temp\_L,0b11111111;Вкл подтяжка на кнопках PD0, PD2-PD7.  out PORTD,temp\_L;  ;  ldi temp\_L,0b11111111;;(PC0-PC7)-выходы  out DDRC,temp\_L  ;  ;INIT USART  ldi temp\_L,0x3B;(Частота кв. 9,216 мГц,скорость обмена 19200),U2X0=0,  ldi temp\_H,00  out UBRRL,temp\_L;задаем скорость передачи  out UBRRH,temp\_H  ldi temp\_L,(1<<RXEN)|(1<<RXCIE); разрешение прерывания по приему  out UCSRB,temp\_L  ldi temp\_L,(1<<URSEL)|(1<<UCSZ0)|(1<<UCSZ1)|(1<<UPM1)|(1<<UPM0);UCSZ2=0,UCSZ1=1,UCSZ0=1 - 8 bit;1 stop bit,  out UCSRC,temp\_L;    ; --- Инициализация таймера TCNT0  ;  ; Инициализация таймера TCNT1  ldi temp\_L,00;compare A,(COM1A1,COM1A0=00) OC1A disconnect  out TCCR1A,temp\_L;  ldi temp\_L,(1<<WGM12);WGM13=0,WGM12=1,WGM11=0,WGM10=0,режим CTC  ; No clock source,CS42,CS41,CS40=000  out TCCR1B,temp\_L;(No prescaling CS10=1  ldi temp\_H,0xD8 ; time\_out 6mcek - 0,006\*9216000=55296(D800)  ldi temp\_L,0x00 ;  out OCR1AH,temp\_H  out OCR1AL,temp\_L  ;Инициализация динамической индикации семисегментные индикаторы  ldi Disp\_Count,Val\_dispCount  clr Disp\_Numb  clr Byte\_fl  clr Cou\_Rec  clr Cou\_Tran  clr c\_sumREC  clr Time  ldi Number,1  ;Обнуление SRAM (буфер хранение переданных и принятых данных)  ldi YL,low(varBuf\_disp) ; Y register low буфер данных,выводимых на дисплей  ldi YH,high(varBuf\_disp) ; Y register high буфер данных,выводимых на дисплей  ldi temp\_L,0x00;начальная установка буфера данных,выводимых на дисплей  ldi temp\_H,VAL\_TR + VAL\_REC  LoadBdsp: st Y+,temp\_L  dec temp\_H  cpi temp\_H,0x00  brne LoadBdsp  ;  ldi YL,low(varBuf\_Rxd) ; Load Y register low буфер приема  ldi YH,high(varBuf\_Rxd) ; Load Y register high буфер приема  ldi temp\_L,0x00  ldi temp\_H,VAL\_REC  ld\_b\_r:st Y+,temp\_L;начальная установка буфер приема  dec temp\_h  cpi temp\_H,0x00  brne ld\_b\_r  ;  ldi YL,low(varBuf\_Txd); Load Y register low буфер передачи  ldi YH,high(varBuf\_Txd) ; Load Y register high буфер передачи  ldi temp\_L,0x00  ldi temp\_H,VAL\_TR  ld\_b\_t:st Y+,temp\_L;начальная установка буфер передачи  dec temp\_h  cpi temp\_H,0x00  brne ld\_b\_t  ;  ldi YL,low(varBuf\_Rxd) ; Load Y register low буфер приема  ldi YH,high(varBuf\_Rxd) ; Load Y register high буфер приема  ;  sei ;разрешаем прерывания  clr Hundreds  clr Tens  clr Ones  ;==================================================  ;начало цикла программы  ;==================================================  Start: rcall Display  sbrc Byte\_fl,F\_receive;проверка флага принятого запроса  rjmp Receive  ;Disab\_Key активен в момент формирования защитного интервала при подавлении дребезга контактов от кнопки "Просмотр"  sbrc Byte\_fl,Disab\_Key;  rjmp Start  sbic PinD,4;Проверка нажатия кнопки(проверяет состояние кнопки "Просмотр")  rjmp Start  rcall ch\_pos\_D;осуществляет смену текущего отображаемого байта  rjmp Start  Receive:  sbi PORTB,CID  rcall Wr\_D\_rec;подпрограмма записи принятых данных в буфер для отображения на семисегментных индикаторах  rcall arifm\_op;подпрограмма выполнения арифметической операции  rcall pre\_date;подпрограмма подготовки данных для передачи модулем USART  rcall Wr\_D\_tr;подпрограмма записи передаваемых данных в буфер для отображения на семисегментных индикаторах  Wait: sbis PIND,ENV;Проверка нажатия кнопки(проверяет состояние кнопки "Ответ")  rjmp Trans  rcall Display; дин. индикация  sbrc Byte\_fl,Disab\_Key;  rjmp Wait  sbic PinD,4;Проверка нажатия кнопки  rjmp Wait  rcall ch\_pos\_D;осуществляет смену текущего отображаемого байта  rjmp wait  Trans: ldi temp\_L,(1<<TXEN)|(1<<UDRIE);UCSZ2=0,UCSZ1=1,UCSZ0=1 - 8 bit  out UCSRB,temp\_L;переход к передаче через вызов прерывания UDRE  Wait\_tr: sbrs Byte\_fl,F\_trans; ожидание окончания передачи (последний байт)  rjmp Wait\_tr  clt  bld Byte\_fl,F\_trans  rjmp Start  ;==================================================  ; конец цикла программы  ;==================================================  ;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  ;Подпрограмма изменения позиции просмотра на дислее  ;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  ch\_pos\_D:  cpi Number,7;7 позиций просмотра  brne PC+2  ldi Number,0  ldi XL,low(varBuf\_disp) ; XL register low буфер данных,выводимых на дисплей  ldi XH,high(varBuf\_disp) ; XH register high буфер данных,выводимых на дисплей  ldi temp\_H,0x00  add XL,Number  adc XH,temp\_H  ld temp\_L,X  rcall digitConvert  ;  inc Number  set  bld Byte\_fl,Disab\_Key; бит запрещ. опроса кнопки "Просмотр" от дребезга  ldi temp\_L,(1<<CS02)|(1<<CS00);частота TCNT0 Clk/1024,(CS02,CS01,CS00) - 101  out TCCR0,temp\_L  in temp\_L,TIMSK;UDRIE 5 10  set  bld temp\_L,TOIE0 ;TOIE0-Timer/Counter0 Overflow Interrupt Enable  out TIMSK,temp\_L; разрешить прерывание Tov0  ret  ;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  ; Подпрограмма Display работы с дисплеем (динам. индикация)  ;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  Display: dec Disp\_Count  brne ex\_displ  ;  ldi Disp\_Count,Val\_dispCount  ;  inc Disp\_Numb  cpi Disp\_Numb,4  brne Out\_disp  clr Disp\_Numb  Out\_disp:  clr ZH  ldi ZL,18; ;указатель на Number  add ZL,Disp\_Numb  ld temp\_L,Z  ; преобразуем в семисегментный код  ldi ZL,low(TABLE\*2);загружаем адрес начала  ldi ZH,high(TABLE\*2);таблицы в памяти программ (\*2 - для байтовой  add ZL,temp\_L ;адресации)  lpm temp\_L,Z ;читаем семисегментный код значения ;  out PortC,temp\_L; передаем на индикатор  ;  in temp\_L,PINB  in temp\_H,PINB  andi temp\_L,0b00001111  andi temp\_H,0b10000000  lsr temp\_L  brcc PC+2  ldi temp\_L,0b00001000; в начало (Number)  or temp\_L,temp\_H  out PORTB,temp\_L    ;  ex\_displ: ret  ;  ;==================================================  ;------- Таблица перекодировки символов  TABLE: .db 0b00111111,0b00000110; коды "0","1"  .db 0b01011011,0b01001111; коды "2","3"  .db 0b01100110,0b01101101;;коды "4","5"  .db 0b01111101,0b00000111;;коды "6","7"  .db 0b01111111,0b01101111;;коды "8","9"  .db 0b10000000,0b00000000; коды " "," "  ;==================================================  Wr\_D\_rec: ;Подпрограмма записи принятых данных в буфер дисплея (после проверки контрольной суммы)  ;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  ldi YL,low(varBuf\_Rxd) ; Load Y register low буфер приема  ldi YH,high(varBuf\_Rxd) ; Load Y register high буфер приема  ldi XL,low(varBuf\_disp) ; XL register low буфер данных,выводимых на дисплей  ldi XH,high(varBuf\_disp)  ldi temp\_H,VAL\_REC  Loop\_wD: ld temp\_L,Y+  st X+,temp\_L  subi temp\_H,1  breq PC+2  rjmp Loop\_wD  ret  ;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  arifm\_op:;Подпрограмма выполнения арифметических операций  ;результат старший байт- temp\_H,младший байт - temp\_L  ;==================================================  clr res\_a\_op;байт промежуточных результатов  ldi YL,low(varBuf\_Rxd) ; Load Y register low буфер приема  ldi YH,high(varBuf\_Rxd) ; Load Y register high буфер приема  ld temp\_L,Y  cpi temp\_L,0x01  breq add\_1\_b  cpi temp\_L,0x02  breq sub\_1\_b  cpi temp\_L,0x03  breq mul\_1\_b  cpi temp\_L,0x04  breq div\_1\_b  ex\_ar\_op: ret  ;------------------------  ;сложение двоичных чисел  add\_1\_b: ldd temp\_L,Y+1  ldd temp\_H,Y+2  add temp\_L,temp\_H  ldi temp\_H,0x00  mov res\_a\_op,temp\_H  adc temp\_H,res\_a\_op  rjmp ex\_ar\_op  ;------------------------  ;вычитание двоичных чисел  sub\_1\_b: ldd temp\_L,Y+1  ldd temp\_H,Y+2  sub temp\_L,temp\_H  ldi temp\_H,0x00  sbci temp\_H,0x00  rjmp ex\_ar\_op  ;-------------------------  ;умножение двоичных чисел  mul\_1\_b: ldd temp\_H,Y+1;множимое  ldd temp\_L,Y+2;множитель  ldi n\_ar\_op,8;восьмиразрядное число  clr res\_a\_op  cycle\_m: sbrc temp\_L,0  add res\_a\_op,temp\_H  lsr res\_a\_op  ror temp\_L  subi n\_ar\_op,1  brne cycle\_m  mov temp\_H,res\_a\_op  rjmp ex\_ar\_op  ;-------------------------  ;деление двоичных чисел  div\_1\_b: ldd temp\_H,Y+1;делимое  ldd temp\_L,Y+2;делитель  Divide: sub temp\_H,temp\_L  brcs DoneDividing  inc res\_a\_op  rjmp Divide  DoneDividing:  neg temp\_L  sub temp\_H,temp\_L  mov temp\_L,temp\_H  mov temp\_H,res\_a\_op  rjmp ex\_ar\_op  ;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  pre\_date:;Подпрограмма подготовки данных к передаче  ;Подпрограмма pre\_date осуществляет вычисление контрольной суммы передаваемых данных и запись передаваемых данных в буфер rav\_Buf\_Txd.  ;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  ldi YL,low(varBuf\_Txd); Load Y register low буфер передачи  ldi YH,high(varBuf\_Txd) ; Load Y register high буфер передачи  clr c\_sumTRAN  add c\_sumTRAN,temp\_H  st Y+,temp\_H  add c\_sumTRAN,temp\_L  st Y+,temp\_L  com c\_sumTRAN  st Y+,c\_sumTRAN  ret  ;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  Wr\_D\_tr:;Подпрограмма записи данных передачи в буфер дисплея  ;осуществляет чтение из буфера rav\_Buf\_Txd и запись в буфер varBuf\_disp, начиная с четвертого адреса.  ;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  ldi YL,low(varBuf\_Txd); Load Y register low буфер передачи  ldi YH,high(varBuf\_Txd) ; Load Y register high буфер передачи  ldi temp\_L,VAL\_REC; смещение 4  ldi temp\_H,0x00  ldi XL,low(varBuf\_disp);XL register low буфер данных,выводимых на дисплей  ldi XH,high(varBuf\_disp)  add XL,temp\_L  adc XH,temp\_H  ldi temp\_H,VAL\_TR  rcall Loop\_wD  ret  ;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  ;Конвертирует двоичное 1-e байтное число в двоично-десятичный код  ;===================================================  digitConvert:  clr Hundreds  clr Tens  clr Ones  ;  FindHundreds:  subi temp\_L,100  brcs FindTens  inc Hundreds  rjmp FindHundreds  ;  FindTens:  subi temp\_L,-100  subi temp\_L,10  brcs FindOnes  inc Tens  rjmp FindTens+1  ;  FindOnes:  subi temp\_L,-10  mov Ones,temp\_L  ret  ;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  ;Subroutine interrupt Overflow 0  ;(предотвращение срабатываний от дребезга контактов кнопки "Просмотр")  ;Подпрограмма обработки прерывания переполнения таймера Т0  ;Т0 - 8 разр таймер. Counter0 оverflow соотвествует времени  ;за 1cек =(1024/9216000)\*256\*N\_отсчетов(переполнений)=35 переполнений  ;При каждом вызове подпрограммы осуществляется инкрементирование регистра Time.  ;Когда регистр достигнет значения VAL\_time = 35 осуществляется сброс флага Disab\_Key, чем разрешается отрос кнопки "Просмотр".  ;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  time\_d\_k: push temp\_L  in temp\_L,SREG  push temp\_L  ;  inc Time  cpi Time,VAL\_time  brne ex\_timDK  clt  bld Byte\_fl,Disab\_Key; бит разр. опроса кнопки "Просмотр" от дребезга  ldi temp\_L,0x00;(1<<CS02)|(1<<CS00);No clock source,(CS02,CS01,CS00) - 000  out TCCR0,temp\_L  in temp\_L,TIMSK  clt  bld temp\_L,TOIE0 ;TOIE0-Timer/Counter0 Overflow Interrupt Enable  out TIMSK,temp\_L; запрещение прерывание Tov0  clr Time  ;  ex\_timDK: pop temp\_L  out SREG,temp\_L  pop temp\_L  reti  ;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  ;Subroutine interrupt USART RX Complete  ;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  REC\_date: push temp\_L  push temp\_H  in temp\_L,SREG  push temp\_L;загружаем данные из temp\_L, temp\_H и SREG в СТЕК  ;  in temp\_H,UCSRA;сохраняем регистр UCSRA  rd\_UDR: in temp\_L,UDR;сохраняем в регистр принятый байт  rjmp rt\_rec; поппрограмма проверки принятого байта  ;  pop\_rec:  pop temp\_L  out SREG,temp\_L  pop temp\_H  pop temp\_L  reti  ;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  rt\_rec:  andi temp\_H,(1<<FE)|(1<<DOR)|(1<<PE); проверка флагов ошибок  breq USART2NoError  rjmp pop\_rec; если есть ошибки приема, то не запоминаем байт  USART2NoError:  st Y+,temp\_L; если нет ошибок приема, то запоминаем байт в буфер данных  inc Cou\_Rec; увеличиваем число принятых байт  mov temp\_H,Cou\_Rec  cpi temp\_H,0x01  breq rec\_1\_b; проверка на первый байт(перейдем,если первый)  cpi temp\_H,VAL\_REC  breq rec\_end; проверка на первый байт(перейдем,если последний)  add c\_sumREC,temp\_L; прибавляем в контрольную сумму принятый байт  rjmp pop\_rec  rec\_1\_b: add c\_sumREC,temp\_L; прибавляем в контрольную сумму принятый байт  ;запускается таймер Т1 для формирования времени таймаута принимаемых данных и разрешается прерывание по совпадению канала А таймера Т1  in temp\_L,TIMSK;  set  bld temp\_L,OCIE1A;OCIE1A разрешить прерывание  out TIMSK,temp\_L  ldi temp\_L,(1<<WGM12)|(1<<CS10);WGM3=0,WGM2=1,WGM1=0,WGM0=0,режим CTC  ; clkI/O/1 (No prescaling),CS2,CS1,CS0=001  out TCCR1B,temp\_L;(No prescaling CS0=1  rjmp pop\_rec  rec\_end:  com temp\_L;инверсия принятого байта  cp temp\_L,c\_sumREC; проверка контр суммы  brne ex\_rec  set  bld Byte\_fl,F\_receive;прием верных данных  ;  ex\_rec: in temp\_L,TIMSK  clt  bld temp\_L,OCIE1A;OCIE1A запретить прерывание  out TIMSK,temp\_L  ldi temp\_L,(1<<WGM12);WGM3=0,WGM2=1,WGM1=0,WGM0=0,режим CTC  ; clkI/O/1 (stop TCNT1),CS2,CS1,CS0=000  out TCCR1B,temp\_L  ldi temp\_L,0x00  ldi temp\_H,0x00  out TCNT1H,temp\_H  out TCNT1L,temp\_L  clr Cou\_Rec  clr c\_sumREC  ldi YL,low(varBuf\_Rxd) ; Load Y register low буфер приема  ldi YH,high(varBuf\_Rxd) ; Load Y register high буфер приема  rjmp pop\_rec  ;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  ;Subroutine interrupt USART Data register Empty  ;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  B\_TRANS: push temp\_L  in temp\_L,SREG  push temp\_L  ;  ldi temp\_L,(1<<TXEN)|(1<<TXCIE);разрешить прерывание TXC  out UCSRB,temp\_L  ldi YL,low(varBuf\_Txd); Load Y register low буфер передачи  ldi YH,high(varBuf\_Txd); Load Y register high буфер передачи  ld temp\_L,Y+  clt  bld Byte\_fl,F\_receive;  out UDR,temp\_L  ;  pop temp\_L  out SREG,temp\_L  pop temp\_L  reti  ;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  ;Subroutine interrupt USART, Tx Complete  ;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  TRANdate: push temp\_L  in temp\_L,SREG  push temp\_L  push temp\_H  rjmp r\_trans;Осуществляется запись байта данных из буфера rav\_Buf\_Txd в регистр передачи  ;  pop\_tran:  pop temp\_H  pop temp\_L  out SREG,temp\_L  pop temp\_L  ;  reti  ;  r\_trans: inc Cou\_Tran  mov temp\_L,Cou\_Tran  cpi temp\_L,VAL\_TR  breq end\_tr;Если значение регистра Cou\_Tran стало равно VAL\_TR, то передача прекращается  ld temp\_L,Y+  out UDR,temp\_L  rjmp pop\_tran    end\_tr: clr Cou\_Tran  ldi temp\_L,(1<<RXEN)|(1<<RXCIE);  out UCSRB,temp\_L  ldi YL,low(varBuf\_Rxd) ; Load Y register low буфер приема  ldi YH,high(varBuf\_Rxd) ; Load Y register high буфер приема  cbi PORTB,CID  set  bld Byte\_fl,F\_trans  rjmp pop\_tran  ;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  ;Subroutine interrupt OC1A  ;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  ;останавливается работа таймера Т1 и запрещается прервынае при совпадении канала А таймера Т1.  ; Затем сбрасываются регистры Cou\_Rec и c\_sumREC, что приводит к тому, что данные,  ;принятые модулем USART будут записываться с начала буфера rav\_Buf\_Rxd.  Time\_OUT: push temp\_L  push temp\_H  in temp\_L,SREG  push temp\_L  ;  ldi temp\_L,(1<<WGM12);WGM13=0,WGM12=1,WGM11=0,WGM10=0,режим CTC  ; No clock source,CS42,CS41,CS40=000  out TCCR1B,temp\_L;  in temp\_L,TIMSK  clt  bld temp\_L,OCIE1A;OCIE1A запретить прерывание  out TIMSK,temp\_L  ldi temp\_H,0x00  ldi temp\_L,0x00 ;  out TCNT1H,temp\_H;сброс счетчика  out TCNT1L,temp\_L  clr Cou\_Rec  clr c\_sumREC  ldi YL,low(varBuf\_Rxd) ; Load Y register low буфер приема  ldi YH,high(varBuf\_Rxd) ; Load Y register high буфер приема  ;  pop temp\_L  out SREG,temp\_L  pop temp\_H  pop temp\_L  reti  ;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* |