Содержание

Введение

1. Анализ поставленной задачи

1.1 Обоснование достаточности аппаратных и программа ресурсов

1.2 Доопределение набора аппаратных средств для реализации устройства

* 1. Распределение функций устройства между узлами микроконтроллера

2. Проектирование принципиальной схемы устройства

* 1. Схема включения микроконтроллера
  2. Формирование тактовых импульсов
  3. Схема сброса
  4. Схема входных и выходных устройств
  5. Схема стабилизатора напряжения

3. Проектирование программного обеспечения микроконтроллера

* 1. Проектирование функции инициализации микроконтроллера
  2. Проектирование процедур обработки прерываний
  3. Проектирование процедур ввода информации
  4. Проектирование процедур вывода информации
  5. Проектирование процедур управления периферийны! устройствами
  6. Проектирование процедуры main()

4. Листинг программы

Приложение 1. Схема электрическая принципиальная

Приложение 2. Чертеж печатной платы (вид сверху)

Приложение 3. Чертеж печатной платы (вид снизу)

Ведение

Широко распространенное семейство микроконтроллеров MCS51, выпускаемое целым рядом фирм-производителей (Intel, Philips, Temic, OKI, Siemens и др.), уже являлось де-факто промышленным стандартом для 8-разрядных систем и прекрасно подходило для использования в широком классе задач, особенно если выбирались кристаллы с дополнительными встроенными периферийными устройствами и повышенной тактовой частотой. Но эти микроконтроллеры обладали значительным энергопотреблением. Тогда, если необходимо было получить высокую производительность кристалла при фиксированном энергопотреблении или, наоборот, снизить последнее не теряя производительности, внимание разработчика, как правило, останавливалось на микросхемах Dallas Semiconductor, Microchip или Hitachi. Широко развитые линии PIC-контроллеров фирмы Microchip и микроконтроллеров Н8/300 фирмы Hitachi обеспечивают достаточно высокую производительность при небольшом энергопотреблении. Эффективность работы микроконтроллеров Dallas Semiconductor, имеющих базовую архитектуру MCS51, в среднем превышает стандартную в 2,5 - 3 раза. Появившиеся в последнее время новые процессорные платформы MSP430 фирмы Texas Instruments и ХЕ8000 фирмы Xeraics также заслуживают самого пристального внимания, особенно если основным критерием для конечного приложения является минимальное энергопотребление.

Окончательный выбор разработчиком той или иной микропроцессорной платформы для реализации своей задачи зависит от большого числа разнообразных факторов, включая экономические. Но обычно первостепенным условием остается получение максимально выгодного соотношения "цена - производительность энергопотребление", определяемого сложностью решаемой задачи. Видимо, это обстоятельство и послужило толчком к разработке в середине 1990-х нового 8-разрядного микроконтроллера.

AVR одно из самых интересных направлений, развиваемых корпорацией Atmel. Они представляют собой мощный инструмент для создания современных высокопроизводительных и экономичных многоцелевых контроллеров. На настоящий момент соотношение "цена - производительность - энергопотребление" для AVR является одним из лучших на мировом рынке 8-разрядных микроконтроллеров. Объемы продаж AVR в мире удваиваются ежегодно. В геометрической прогрессии растет число сторонних фирм, разрабатывающих и выпускающих разнообразные программные и аппаратные средства поддержки разработок для них. Можно считать, что AVR постепенно становится еще одним индустриальным стандартом среди 8-разрядных микроконтроллеров общего назначения.

Области применения AVR очень широки. Для семейства "tiny" -это интеллектуальные автомобильные датчики различного назначения, игрушки, игровые приставки, материнские платы персональных компьютеров, контроллеры защиты доступа в мобильных телефонах, зарядные устройства, детекторы дыма и пламени, бытовая техника, разнообразные инфракрасные пульты дистанционного управления. Для семейства "classic" - это модемы различных типов, современные зарядные устройства, изделия класса Smart Cards и устройства чтения для них, спутниковые навигационные системы для определения местоположения автомобилей на трассе, сложная бытовая техника, пульты дистанционного управления, сетевые карты, материнские платы компьютеров, сотовые телефоны нового поколения а также различные и разнообразные промышленные системы контроля и управления. Для "mega" AVR - это аналоговые (NMT, ETACS, AMPS) и цифровые (GSM, CDMA) мобильные телефоны, принтеры и ключевые контроллеры для них, контроллеры аппаратов факсимильной связи и ксероксов, контроллеры современных дисковых накопителей, CD-ROM и т.д.

В данной работе приводится пример использования AVR типа ATmega16 для построения устройства удаленной индикации, принимающее сигналы по протоколу связи RS-485 и отображающее соответствующую информацию на графическом ЖКИ дисплее фирмы Bolyrain.

1. Анализ поставленной задачи

Постановка задачи; спроектировать удаленное устройство индикации на основе 8-битного AVR микроконтроллера типа ATmega16 с питанием данного устройства от источника питания на 10 V. Требуется обеспечить прием данных по протоколу RS-485 на скорости 9600 бит в секунду, с размером посылки данных 8 бит, проверкой на четность и одним стоповым битом. Данные должны отображаться на графическом LCD-дисплее фирмы Bolymin.

1.1 Обоснование достаточности аппаратных средств и программных ресурсов

Предложенный для решения задачи микроконтроллер ATmegal6 обладает следующими характеристиками:

1. напряжение питания+5 V
2. размер памяти программ 16 К
3. размер EEPROM512 В
4. размер внутренней SRAM 1 К
5. порты ввода/вывода4x8 bit
6. четыре таймера счетчика
7. программируемый последовательный УСАПП

Этих свойств микроконтроллера вполне достаточно для обеспечения взаимодействия с графическим LCD-дисплеем и протоколом обмена данными RS-485, поскольку большой объем памяти программ позволяет обеспечить логику работы всех аппаратных средств микроконтроллера и управление LCD-дисплеем. Кроме того, подключение внешнего источника тактовых импульсов позволяет обеспечить скорость обмена данными до 1 миллиона бит в секунду.

1.2 Доопределение набора аппаратных средств

Для организации канала связи по протоколу RS-485 необходимо использование устройств, отвечающих требованиям этого протокола. Возможное решение - использование схемы МАХ485, которая работает от одного источника питания +5 V, и его выходное сопротивление становится высоким в диапазоне синфазного сигнала от -7 до +12 V при подаче и при выключении питания. Передатчик имеет максимальное время задержки 50 пз и время нарастания и спада менее 80 ns. Это позволяет получить скорость передачи данных до 4 Mbaud.

Поскольку разрабатываемое устройство питается от источника напряжения +10 V, то для обеспечения питания микросхем устройства необходимо использование стабилизатора напряжения, который можно реализовать на микросхеме LM2574 (понижающий импульсный стабилизатор напряжения).

1.3 Распределение функций устройства между узлами микроконтроллера

Разрабатываемое удаленное устройство индикации должно выполнять следующие две главные функции: обеспечение приема данных по каналу связи и индикация обработанных данных на LCD-дисплей.

Вполне логично в качестве приемника использовать встроенный в контроллер программируемый последовательный универсальный синхронно-асинхронный приеме-передатчик (УСАПП). При этом будут задействованы выводы PDO (RxD) и PD1 (TxD), которые подключаются к соответствующим выводам микросхемы МАХ485. Кроме того, для управления микросхемой МАХ485 необходимо подключить также сигналы разрешения приема и передачи данных (сигнал разрешения приема - инверсный), Но поскольку в нашем устройстве не предусмотрена возможность одновременно принимать и передавать данные, представляется удобным использование общего сигнала с одного из выводов контроллера для управления приемом и передачей (вывод порта С РС5). Более того, в частном случае наше устройство не будет передавать данные на внешние устройства, поэтому как один из вариантов может быть использовано просто подключение разрешающих выводов микросхемы МАХ485 к общему постоянному сигналу низкого уровня, что запретит микросхеме передачу данных и она будет все время использоваться как приемник.

Управление LCD-дисплеем осуществляется с помощью восьми линий, по которым передаются данные, и пяти линий, по которым передаются сигналы управления (чтение/запись данных, запись команды и т.п.}. Поэтому для управления дисплеем мы выделим порт А контроллера для передачи сигналов данных и линии РСО - РС4 порта С для передачи сигналов управления.

Все остальные устройства контроллера в нашем (простейшем) случае остаются незадействованными.

2. Проектирование принципиальной схемы устройства

2.1 Схема включения микроконтроллера

Микроконтроллер AVR типа ATmegal6 имеет напряжение питания +5 V, которое подводится к выводу VCC. Так как питание всего нашего устройства +10 V, то питание к микроконтроллеру должно подводится через понижающий импульсный стабилизатор. Кроме этого, микроконтроллер имеет еще два вывода для питания порта А (или АЦП), один из которых (AVCC) должен быть подключен к напряжению питания даже если порт А не используется. При если используется АЦП, то это питание должно подключатся через фильтр низких частот. Второй вывод (AREF) используется для подачи напряжения смещения на АЦП.

В нашем устройстве АЦП не используется, поэтому к выводам VCC и AVCC можно подключить предварительно стабилизированное питание +5 V от стабилизатора, а вывод AREF подключить к общей шине земли.

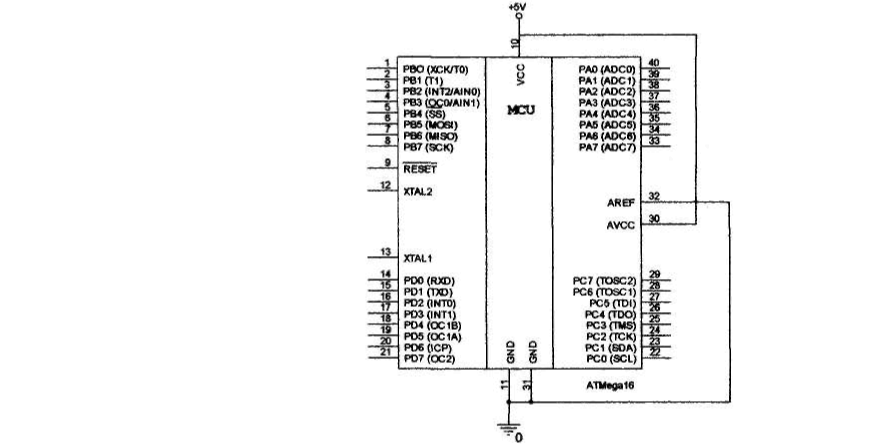
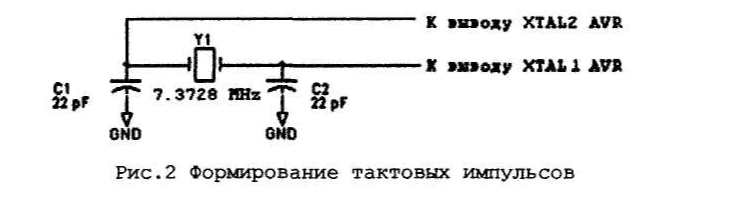


Рис.1 Схема включения микроконтроллера

2.2 Формирование тактовых импульсов

Тактовые импульсы для работы микроконтроллера можно формировать с помощью либо встроенного генератора импульсов, либо подключая внешний генератор на кварцевом резонаторе. Внутренний генератор тактовых импульсов в нашем случае не обеспечит необходимую стабильность для работы с интерфейсом RS-485, поэтому мы будем использовать внешний кварцевый генератор на 7.3728 MHz. Для работы на такой частоте разработчики фирмы ATMEL советуют использовать два дополнительных конденсатора емкостью 22 pF, включенные по следующей схеме:



2.3 Схема сброса

Схема сброса должна формировать импульс логического нуля для подачи его на инверсный вывод сброса микроконтроллера RESET. Это импульс должен формироваться при, например, нажатии на кнопку сброса устройства или при переключении ключа. Кроме этого, этот сигнал сброса должен подаваться и на вывод сброса LCD - дисплея.

Для такой схемы подходит включение транзистора, показанное на рис.3. Когда ключ открыт, транзистор находится в закрытом состоянии и на входы RESET AVR и RST LCD-дисплея поступает сигнал логической единицы. Когда ключ замыкается, транзистор открывается и на входы сброса устройств поступает сигнал логического нуля.

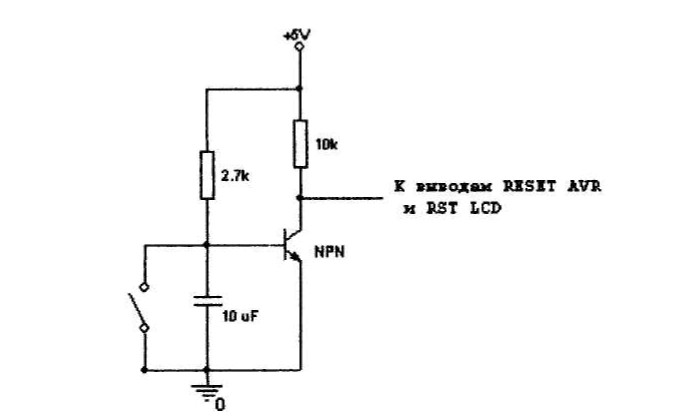


Рис.3 Схема формирования сигнала сброса

2.4 Схемы входных и выходных устройств

Входным устройством в нашем проекте является микросхема обеспечения связи по протоколу передачи данных RS-485 МАХ 485 со следующими электрическими характеристиками:

Из этих параметров видно, что микросхема МАХ485 согласуется с микроконтроллером ATmegal6.

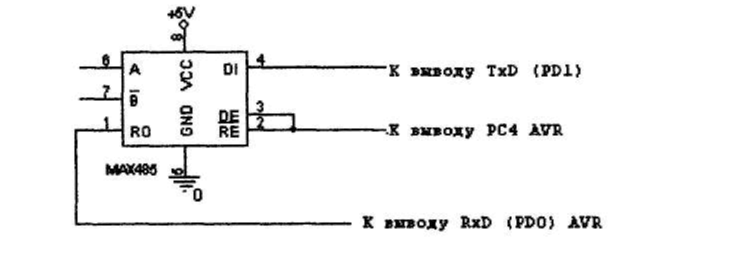


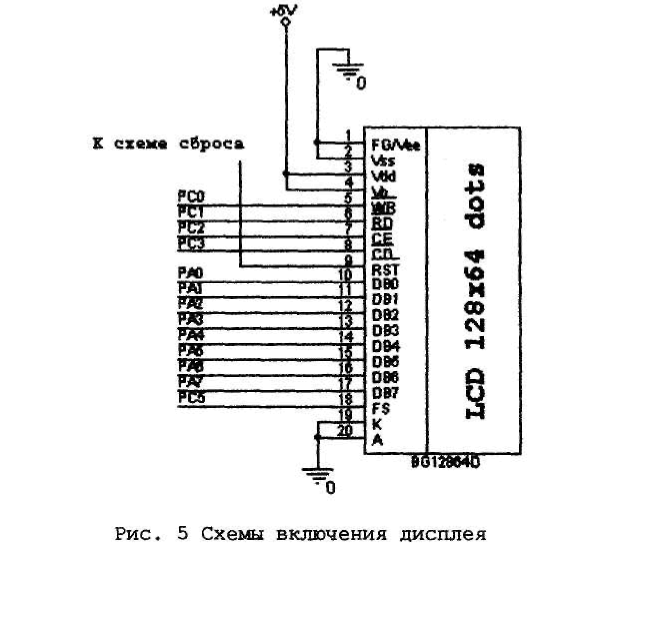
Рис. 4 Подключение микросхемы MAX4S5

Выходное устройство жидкокристаллический графический дисплей BG12864D фирмы Bolymin со встроенным контроллером Т6963С. Этот дисплей обладает следующими характеристиками:

• Механические характеристики

• Назначение выводов

В соответствии с этими параметрами схема подключения LCD-дисплея к микроконтроллеру будет следующей:



2.5 Схема стабилизатора напряжения

В качестве стабилизатора напряжения в нашем устройстве используется импульсный понижающий стабилизатор LM2574, который обладает следующими характеристиками:

1. входное напряжение - до 60 V {для HV версий)
2. выходное напряжение - 3.3 V, 5 V, 12 V, 15V
3. выходной ток - 0.5 А

Схема включения стабилизатора для преобразования +10 V - +5 V приведена на рис.6:

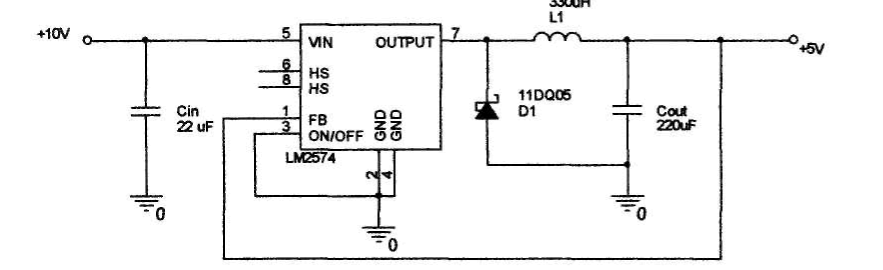


Рис. 6 (Схема стабилизатора напряжения

3. Проектирование программного обеспечения микроконтроллера

3.1 Проектирование функции инициализации микроконтроллера

Процедура инициализации микроконтроллера должна состоять из процедур или операторов инициализации всех узлов самого микроконтроллера и всех периферийных устройств, и установить все начальные значения для их регистров. Таким образом, мы должны проинициализировать следующие узлы устройства - порт А, порт С, УСАПП, таймер 0 и LCD-дисплей.

Порты А и С в начале работы устройства работают только на вывод данных, поэтому при их настройке необходимо в соответствующие регистры DDRx записать значение 0, тем самым настроив все их выводы на передачу данных. Для этого используются две процедуры;

void InitPortAWrite(void){

DDRA = Oxff; }

void InitPortCWrite(void){ DDRC = Oxff; PORTC = 0x30;

Инициализация таймера О проходит по следующему алгоритму - устанавливается делитель частоты на 1024 записью значения 5 в регистр TCCRO. Затем разрешается прерывание этого таймера и устанавливается его начальное значение:

void InitTimer(void)

i

\_disable\_interrupt() ;

TCCRO = 5;// установить делитель частоты 1024

TIMSK |= (1 « TOIEO); // разрешить прерывания таймера

TCNTO = TmrO\_Reload;

enable\_interrupt{);

]

Инициализация УСАПП работает следующим образом - в регистр

UBRR записывается значение, которое соответствует заданной скорости передачи данных для соответствующей частоты работы микроконтроллера.

УСАПП и параметры кадра данных. Кроме этого, т.к. прием и обработка данных в программе происходит через кольцевой буфер, то в процедуре инициализации необходимо провести начальные установки для головы и хвост буфера - обнулить их.

void USART\_Init( unsigned int baudrate )

i

unsigned char x;

UBRROH = (unsigned char) (baudrate»8) ; UBRROL = (unsigned char) baudrate;

UCSRB = ({1 « RXCIE) j <1«ЮС£Н) ) ;

UCSRC = (1«URSEL) | (3«UCSZO) | (1«UPM1);

x = 0;

USART\_RxTail = x; USARTJRxHead = x;

}

3.2 Проектирование процедур обработки прерываний

В процессе работы нашего устройства могут возникнуть два прерывания - от таймера 0 и от УСАПП. Прерывание от таймера О обрабатывается очень просто: перезагружается начальное значение и флаг его срабатывания устанавливается в 1.

pragma vector=TIMERO\_OVF\_vect

interrupt void TIMERO\_OVF\_interrupt(void)

{

TCNT0=TmrO\_Reload; TmrOFlag = 1;

}

Прерывание от УСАПП говорит о том, что прием пакета данных был закончен и принят в буферный регистр UDR. Обработчик этого прерывания должен принять эти данные и поместить в кольцевой буфер. Для этого должен быть рассчитан новый индекс буфера (указатель головы) и если этот индекс вдруг стал равен указателю хвоста, это говорит о том, что произошла ошибка и буфер приемника переполнился.

^pragma vector=USART\_RXC\_\_vect

^interrupt void USART\_RX\_interrupt{ void )

{

unsigned char data; unsigned char tmphead;

data = UDR;

tmphead = ( USART\_RxHead + 1 );

USART\_RxHead = tmphead; /\* Сохранить новый индекс V

if < tmphead == USART\_RxTail )

(

/\* Ошибка! Буффер приемника переполнен \*/

)

USART\_RxBuf[tmphead] = data; /\* Сохранить полученные данные в буффере \*/

}

3.3 Проектирование процедур ввода информации

Ввод информации в разрабатываемое устройство осуществляется через УСАПП по протоколу RS-4 85. Как уже отмечалось, прием данных в программе происходит по прерыванию от УСАПП, обработчик которого помещает принятый байт в Оуфер приемника. В главной программе, для того, чтобы можно было анализировать этот буфер и читать данные уже непосредственно из него, необходима процедура, которая будет доставать данные из буфера таким образом, чтобы первыми поступали байты, попавшие в буфер раньше всех. Это делает процедура USART\_Receive(), которая сначала ждет поступления данных в буфер, а затем по одному байту достает их оттуда.

BYTE USART\_Receive( void )

1

unsigned char tmptail;

while ( USARTJixHead == USART\_RxTail )

;

tmptail = USART\_RxTail + 1; USART\_RxTail = tmptail; return USART\_RxBuf[tmptail];

f

3.4 Проектирование процедур вывода информации

Вывод информации в нашем устройстве осуществляется на LCD-дисплей. Основной процедурой, которая отображает строки поступивших данных на дисплее - это процедура AutoWriteMode(), которая по сути управляет дисплеем, выводя на него последовательно символы, хранящиеся в глобальном массиве OutString[], в режиме автозаписи, При этом нам необходимо только менять адрес позиции, в которую выводятся данные.

WORD AutoWriteMode(WORD Address, BYTE NumBytes)

{

SetAddressPointer(Address); SendCommanct{SET\_DATA\_AUTO\_HRITE\_COMMAND);

forfint i = 0; i < NumBytes; i++) {

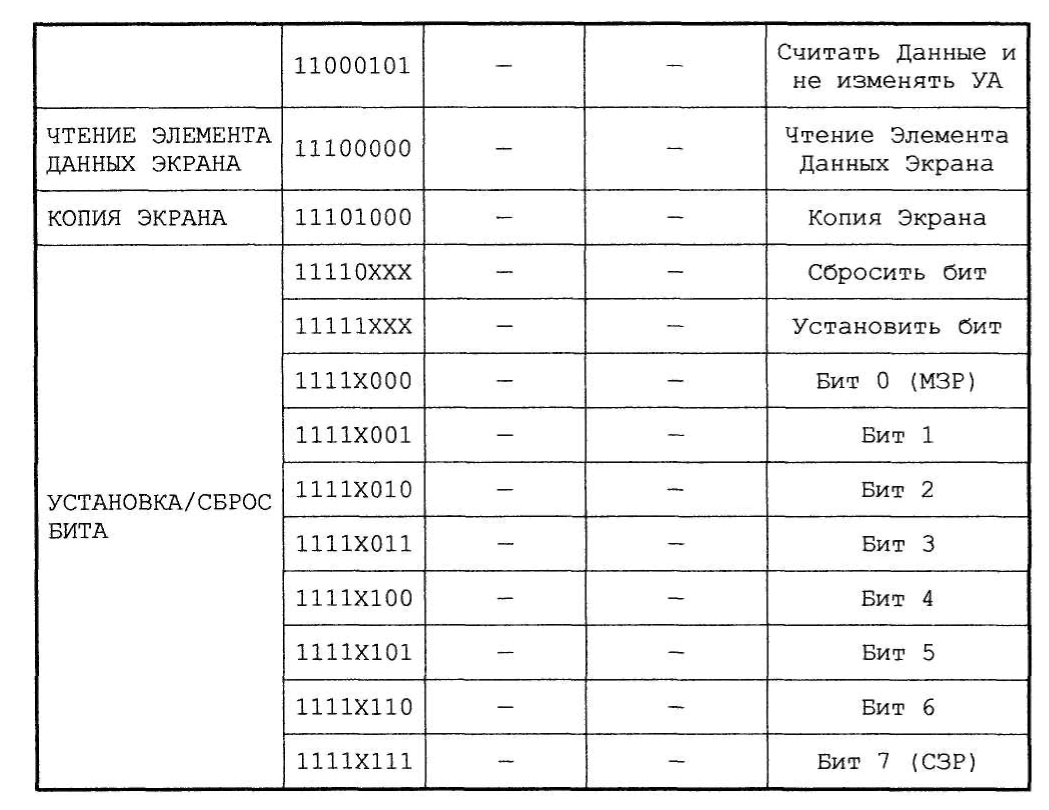
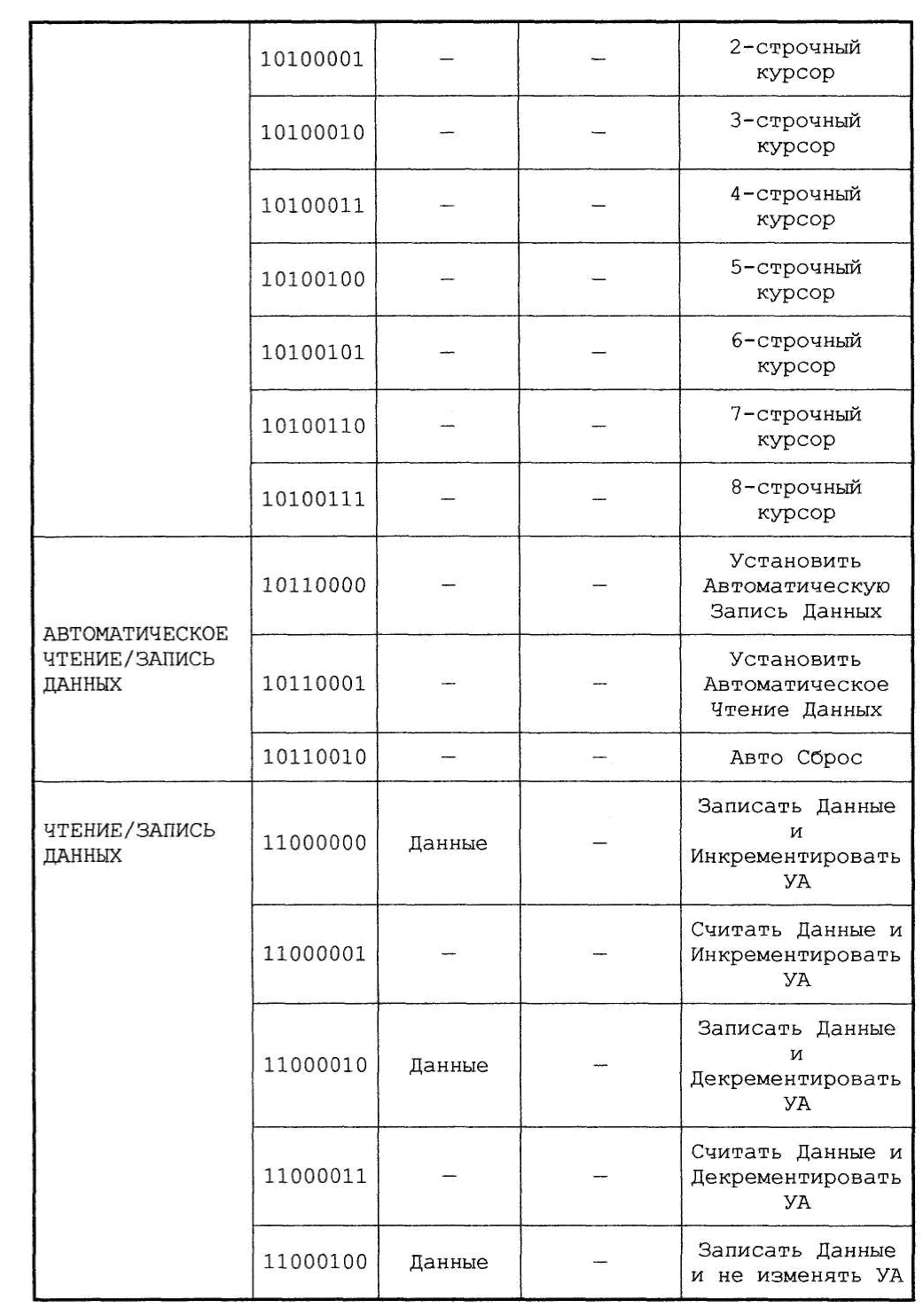
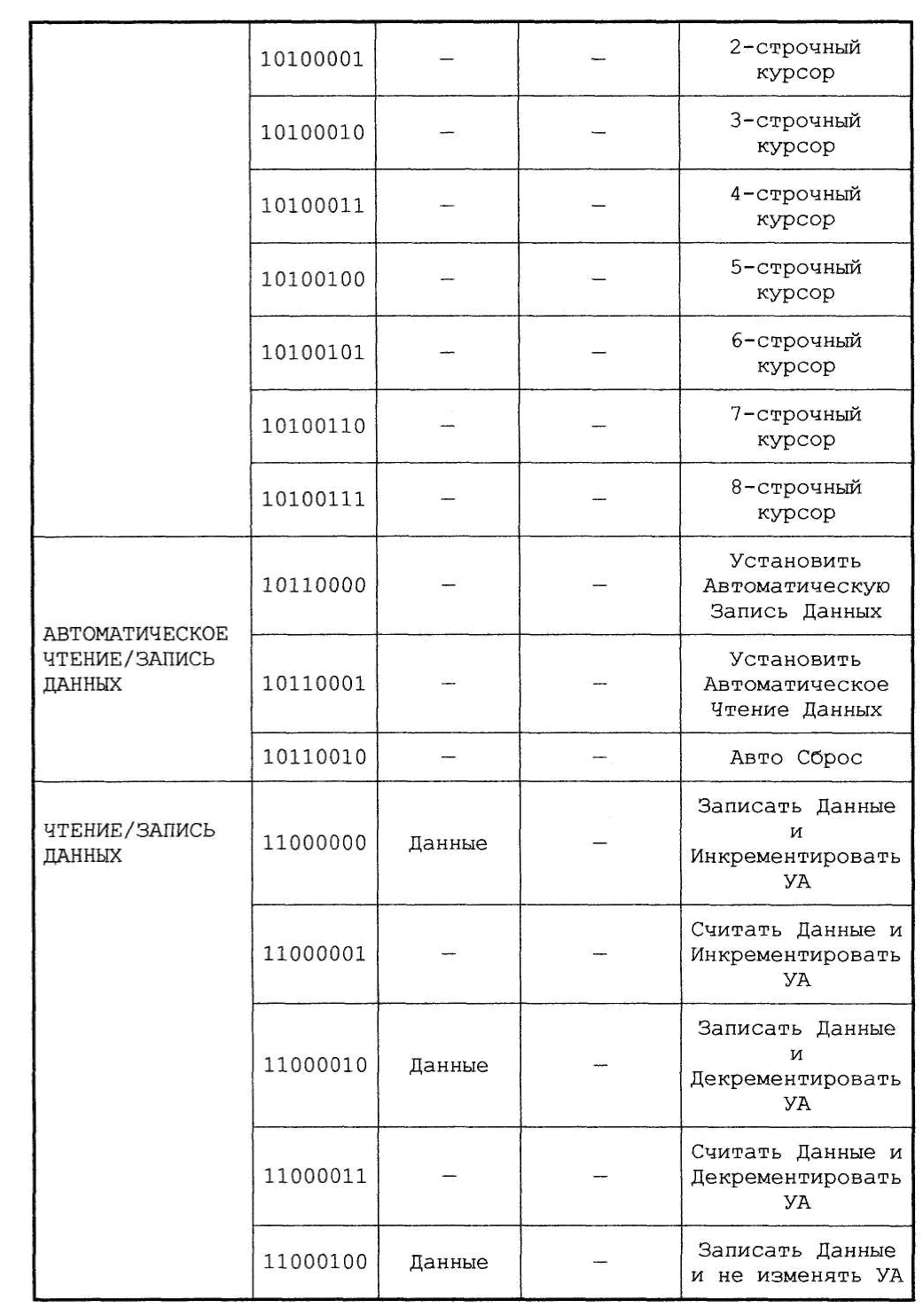
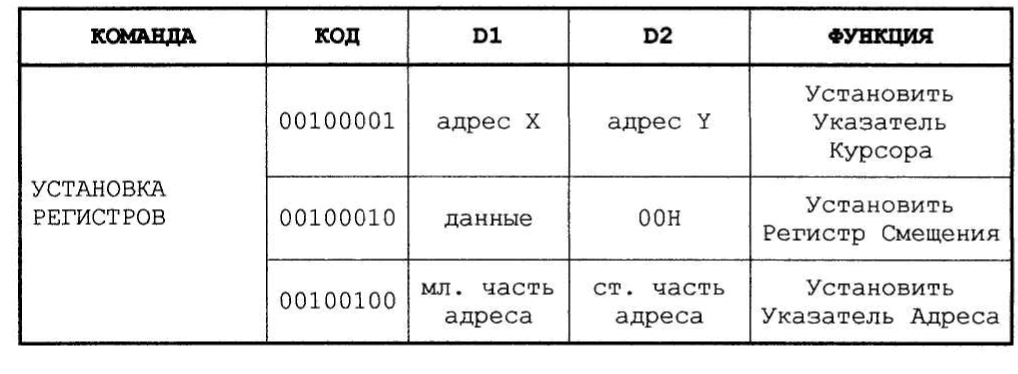
AutoWrite(OutString[i]);

Address++; }

SendCommand(AUTO\_RESET\_COMMAND); return Address; }

3.5 Проектирование процедур управления периферийными устройствами

Все процедуры управления LCD-дисплеем осуществляются согласно системе команд встроенного контроллера Т6963:



3.6 Проектирование процедуры main()

Процедура main{) работает следующим образом после инициализации все узлов AVR и периферийных устройств, разрешается выполнение всех прерываний. После этого программа ждет срабатывания таймера, который настроен таким образом, что он немного чаще, чем может происходить прерывание от УСАПП. Сигналом того, что таймер 0 сработал служит факт установки глобальной переменной TmrOFlag в 1, что делается обработчиком прерываний таймера.

Затем, все время, пока кольцевой буфер не будет пуст, происходит считывание данных из него. Считанный байт помещается в строку-массив OutString[], и увеличивается счетчик принятых байт.

После этого проверяется, а не был ли последний принятый из буфера байт признаком окончания строки. Если да, то в режиме автозаписи эта строка выводится на дисплей (за исключением последнего символа конца строки). После этого строка обнуляется и счетчик принятых байт устанавливается в ноль.

В любом случае происходит сброс флага таймера в ноль и происходит ожидание следующего срабатывания таймера.

4. Листинг программы

Файл макроопределений my\_header.h:

^define BYTE unsigned char ttdefine WORD unsigned int

ttdefine READ\_STATUS\_COMMAND 0x39

ttdefine DATA\_WRITE\_COMMAND 0x32

tfdefine COMMANDjmTE\_COMMAND ОхЗА

// Установка регистров

^define SET\_CURSOR\_COMMAKD 0x21

tfdefine SET\_OFFSET\_COMMAND 0x22

#define SET\_ADDRESS\_COMMAND 0x24

// Установка контрольного слова

#define SET\_TEXT\_HOME\_ADDRESS\_COMMAND 0x^0

fldefine SET\_TEXT\_AREA\_COMMAHD0x41

#define SET\_GR№HIC\_HOME\_ADDRESS\_CO№1AND 0x42

#define SET\_GRAPHIC\_AREA\_CCMMAND0x43

// Константы установки режима

^define OR\_MODE0x80

tfdefine EXOR\_MODE0x81

^define ANDJMODE0x83

#define TEXT\_ATTRIBUTE\_MODE0x84

^define INTERNAL\_CG\_ROM\_MODE 0x80

tfdefine EXTERNAL\_CG\_RAM\_\_MODE 0x88

// Константы режима дисплея

#define DISPLAY\_OFF0x90

^define CURSOR\_ON\_BLIHK\_OFF0x92

^define CURSOR\_ON\_BLINK\_OK0x93

#define TEXT\_ON\_GRAPHIC\_OFF0x94

#define TEXT\_\_OFF\_GRAPHIC\_ON0x98

^define TEXT\_ON\_GRAPHIC\_OnOX9C

// Размер курсора

#define ONE\_LINE OxAO

tfdefine TWO\_LINE OxAl

^define THREE\_LINEOxA2

ttdefine FOUR\_LINE ОхАЗ

^define FIVE\_LINE OxA4

^define SIX\_LINE OxA5

^define SEVEN\_LINEOxA6

^define EIGHT\_LINEOxA7

// Автоматическое чтение/данных

tfdefine SET\_DATA\_AUTO\_WRITE\_COMMAND OxBO

^define SET\_DATA\_AUTO\_READ\_COMMAND OxBl

^define AUTO\_RESET\_COMMANDOxB2

//

#define SCREEN\_PEEK\_CCMMANDOxEO

#define SCREEN\_COPY\_CC»1MANDOxE8

// Установка/сброс Оитов

tfdefine SET\_BIT OxFO

^define RESET\_BIT OxF8

^define BIT\_0 OxFO

#define BIT\_1 OxFl

ttdefine BIT\_2 OxF2

^define BIT\_3 OxF3

idefine BIT\_4 OxF4

#define BIT\_5 OxFS

^define BIT\_6 OxF6

^define BIT\_7OxF7

Файл главкой программы kurs.c;

/\* Includes \*/ ^include <iom!28.h> ^include <ina90.h>

Sinclude "my\_header.h"

^define USART\_RX\_BUFFER\_SI2E 128

itdefine TmrOReload 4

static unsigned char TmrOFlag;

Static unsigned char USART\_RxBuf[USART\_RX\_BUFFER\_SIZE];

static volatile unsigned char USART\_RxHead; static volatile unsigned char USART\_RxTail;

static BYTE OutString[128];

// Процедуры инициализации устройств

void InitAVR(void);

void USART\_Init( unsigned int baudrate );

void InitLCD(void);

void InitTimer(void);

void InitPortARead(void);

void InitPortAWrite(void)?

void InitPortCWrite(void);

// Процедуры для работы с УСАПП и LCD

unsigned char USART\_Receive( void );

void Data«rite<BYTE Data);

void CommandWrite(BYTE Command);

void ReadStatus(void);

void ReadStatus2(void);

void SendCoimand{BYTE Command);

void SendlByteCommand(BYTE Data, BYTE Command);

void Send2ByteCommand(WORD Data, BYTE Command);

void SetCursorPointer(WORD Position)

void SetAddressPointer(WORD Address)

void ByteWriteToRam(WORD AddressPointer, BYTE Data)

void AutoWrite(BYTE Data);

void mainf void )

f

BYTE HumBytes = 0;

InitAVR();

\_SEI(); /\* Разрешить прерывания \*/

while (!)

{

if(TmrOFlag) // Произошло срабатывание таймера

{

BYTE rec = 0;

while(DatalnReceiveBuffer() != 0) // Пока буффер не пуст

{

rec = USART\_Receive(); // Приняли байт из буффера

OutString[NumBytes] = Rebuild(rec); // Записали его в строку NumBytes-n-;// Увеличить счетчик принятых байт

if (rec == OxOD)// Конец строки - ?

f

/\* Выводим строку на LCD в режиме AutoWrite\*/

AutoWriteModefO, —NumBytes)

for(int i = 0; i <= NumBytes; i++) OutStringU] = 0;

NumBytes = 0; }

\

TmrOFlag = 0;

} )

}

/\* Процедура инициализации USART \*/ void USART\_Init( unsigned int baudrate ) t

unsigned char x;

/\* Установить частоту \*/

UBRROH = (unsigned char) (baudrate»8) ;

UBRROL = (unsigned char) baudrate;

/\* Включить приемник UART \*/ UCSRB = { (1 « RXCIE) | (1«RXEH) ) ;

У/Формат приема данных: 8 бит данных, 1 стоп-бит, проверка на четность UCSRC = (1«URSEL) I (3«UCSZO) | (1«UPM1) ;

/\* Обнулить буффер приемника \*/ х = 0;

USARTJbcTail = х; USART\_RxHead = х;

)

/\* Процедура чтения из приемника \*/ BYTE USART\_Receive( void }

{

unsigned char tmptail;

while ( USART\_RxHead == USART\_RxTail ) /\* Ждем поступления данных \*/ ;

tmptail = USART\_RxTail + 1; /\* Рассчет индекса буффера \*/ USART\_RxTail = tmptail;/\* Сохранить новели индекс \*/

return USART\_RxBuf[tmptail]; /\* Вернуть данные \*/

>

unsigned char DatalnReceiveBuffer( void )

{

return ( USART\_RxHead != USART\_RxTail ); /\* Возвращает О если буффер пуст \*/

}

/\* Обработчик прерывания от приемника \*/ #pragma vector=USART\_RXC\_vect \_interrupt void USART\_RX\_interrupt( void ) {

unsigned char data;

unsigned char tmphead;

/\* Прочесть полученные данные \*/

data = UDR;

/\* Рассчет нового индекса \*/

tmphead = ( USARTJRxHead + 1 };

USART\_RxHead = tmphead; /\* Сохранить новый индекс V

if ( tmphead == USART\_RxTail ) (

/\* Ошибка! Буффер приемника переполнен \*/

}

USART\_RxBuf[tmphead] = data; /\* Сохранить полученные данные в буффере \*/ }

// Процедура инициализации таймера

void JnitTimer(void)

{

disable\_interrupt();

TCCRO = 5;// timerO counts elk/1024

TIMSK |= M « TOIEO); //Enable TimerO Interrupt TCWTO = TmrO\_Reload; TmrOFlag = 0;

enable\_interrupt(};

}

/\* Обработчик прерывания от таймера \*/ tfpragma vector=TIMERO\_OVF\_vect

interrupt void TIMERO\_OVF\_interrupt(void)

(

TCNTO=TmrO\_Reload;

TmrOFlag =1; }

void InitPortARead(void) {

DDRA = 0;

PORTA = Oxff; }

void InitPortAWrite(void) {

DDRA = Oxff; }

void InitPortCWrite(void) I

DDRC = Oxff;

PORTC = 0x30; }

void InitLCD(void) {

int tmp = 0;

Send2ByteComraand(tm.p, ET\_TEXT\_HOME\_ADDRESS\_COMMAND);

tmp = 0x14;

Send2ByteCommand(tmp, SET\_TEXT\_AREA\_COMMAND);

tmp = 0x80;

CoromandWrite(INTERNAL\_CG\_ROM\_MODE);

CommandWrite{TEXT\_ON\_GRAPHIC\_OFF);

I

void InitAVR(void)

I

InitPortAWriteO ;

InitPortCWriteO ;

InitLCDO;

InitTimer();

USART\_Init( 47 };//Установить частоту 9,600 используя 1.3728MHz кристалл

}

// Процедуры для работы с LCD-дисплеем void DataWrite(BYTE Data)

t

PINC = DATA\_WRITE\_COMMAND;

PIНА ~ Data; }

void CommandWrite(BYTE Command)

f

PINC = CCMMRND\_WRITE\_COMMAND;

Р1ЫА = Command; J

void ReadStatus(void) {

BYTE tempFlag = 0; BYTE stat; InitPortAReadO ; while(tempFlag != 1) (

PINC = READ\_STATUS\_COMMAND;

Stat = PIMH.;

if((stat & 0x03) == 0x03) tempFlag = 1; }

InitPortAWriteO }

void ReadStatusS(void)

{

BYTE tempFlag = 0; BYTE stat; InitPortAReadO; while(tempFlag != 1) {

PINC = READ\_STATUS\_COMMAND;

stat = PINA;

if((stat b 0x08) == 0x08) tempFlag = 1; }

InitPortAWriteO }

void SendCommandfBYTE Command)

i

ReadStatus{); WriteCommand(Command); }

void SendlByteCommandfBYTE Data, BYTE Command) {

ReadStatus();

DataWrite(Data);

ReadStatus{);

CommandWrite(Command); }

void Send2ByteCommand{WORD Data, BYTE Command) {

ReadStatus();

DataWrite((BYTE)Data);

ReadStatus();

DataHrite((BYTE)(Data » 8));

ReadStatus ();

CommandWrite(Command); }

// Position: младший байт - координата X (от OOh до 4Fh)

//старший байт - координата Y (от OOh до IFh)

void SetCursorPointer(WORD Position)

{

Send2ByteComnand(Positon, SET\_CURSOR\_COMMAND);

>

// Address: младший байт

//старший байт

void SetAddressPointer(WORD Address)

{

Serid2BytesCommand[Address, S£T\_ADDRESS\_COMMAND);

}

void ByteWriteToRamfWORD AddressPointer, BYTE Data)

(

SetAddressPointer(AddressPointer);

ReadStatus();

DataWrite (Data) ,-

ReadStatus();

DataWrite(DATAJWRITEjLNC\_ADP); }

void AutoWrite(BYTE Data) (

ReadStatus2();

PINA = Data;

}

// Массив символов OutString должен быть объявлен как

// глобальный массив типа BYTE

WORD AutoWriteModefWORD Address, BYTE NumBytes)

I

SetAddressPointer(Address);

SendCommand(SET\_DATA\_AUTO\_WRITE\_COMMRND);

for(int i = 0; i < NumBytes; i++) {

AutoWrite(OutString[i]);

Address++; }

SendCommand (AUTO\_RESET\_CCMMAND) ; return Address;

}

