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 **ИНФОРМАТИКА И СИСТЕМЫ УПРАВЛЕНИЯ**

КАФЕДРА **КОМПЬЮТЕРНЫЕ СИСТЕМЫ И СЕТИ (ИУ6)**

НАПРАВЛЕНИЕ ПОДГОТОВКИ 09.03.01 **Информатика и вычислительная техника**

РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

К КУРСОВОЙ РАБОТЕ

по дисциплине «Микропроцессорные системы»

НА ТЕМУ:

Контроллер электронной очереди

Студент	ИУ6-73Б		М.Д. Жиркова
	(Группа)	(Подпись, дата)	(И.О. Фамилия)
Руководитель			И.Б. Трамов
		(Подпись, дата)	(И.О. Фамилия)

2024 г.

ЛИСТ ЗАДАНИЯ

РЕФЕРАТ

РПЗ 56 страниц, 24 рисунков, 18 таблиц, 13 источников, 3 приложения.

МИКРОКОНТРОЛЛЕР, СИСТЕМА, ОЧЕРЕДЬ, ПРИЕМ

Объектом разработки является контроллер электронной очереди с возможностью добавления и удаления человека из очереди.

Цель работы – создание функционального устройства ограниченной сложности, модель устройства и разработка необходимой документации на объект разработки.

Поставленная цель достигается посредством использования Proteus 8.

В процессе работы над курсовым проектом решаются следующие задачи: выбор МК и драйвера обмена данных, создание функциональной и принципиальной схем системы, расчет потребляемой мощности устройства, разработка программы МК, а также написание сопутствующей документации.

Содержание

ОПРЕДЕЛЕНИЯ, ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ

МК – микроконтроллер.

ТЗ – техническое задание.

Proteus 8 — пакет программ для автоматизированного проектирования (САПР) электронных схем.

USART – Universal Synchronous-Asynchronous receiver/transmitter – последовательный универсальный синхронный/асинхронный приемопередатчик.

SPI – Serial Peripheral Interface – интерфейс для связи МК с другими внешними устройствами.

Введение

В данной работе производится разработка контроллера электронной очереди.

В процессе выполнения работы проведен анализ технического задания, создана концепция устройства, разработаны электрические схемы, построена управляющая программа для МК, выполнено интерактивное моделирование устройства.

Система состоит из МК, устройства для вывода звукового сигнала и дисплея для вывода текстовой информации. В ней реализован таймер для удаления человека из очереди, кнопка добавления человека в очередь, кнопка удаления человека из очереди и кнопка сброса работы МК. Также данные об очереди из МК можно загружать на ПЭВМ.

Актуальность разрабатываемой контроллера электронной очереди, приближенной по функциям к реальности, заключается в том, что электронные очереди используются во множестве организаций, и без них было бы сложно контролировать потоки людей и скорость и порядок их обслуживания. Не все очереди имеют таймер удаления человека из очереди, которые контролируют длительность обслуживания человека и устраняют потенциал того, что обслуживающий персонал забудет удалить человека из очереди.

1 Конструкторская часть

1.1 Анализ требований и принцип работы системы

Исходя из требований, изложенных в техническом задании, можно сделать вывод, что задачей работы устройства является контроль электронной очереди.

Контроллер очереди имеет 3 кнопки, влияющие на его функционал. Их функционалы в таблице 1.

Таблица 1 – Работа кнопок

Номер	Описание
1-ая кнопка	Увеличение номера на 1, добавление номера в массив очереди и увеличение индекса последнего номера на 1.
2-ая кнопка	Сброс таймера, и если индекс последнего номера больше 0, сдвиг всех номеров влево и уменьшение индекса последнего номера на 1.
3-ья кнопка	Перезагрузка МК.

Также пешеход может вызвать функционал 2-й кнопки без ее нажатия, если пройдет время, указанное для таймера МК. Возможно два случая:

– Сброс таймера, сдвиг всех номеров влево и уменьшение индекса последнего номера на 1;

– Сброс таймера.

Настройка некоторых параметров предусмотрена в коде, описание данных параметров показано в таблице 2.

Таблица 2 – Настраиваемые параметры

Порядок ввода	Описание
1-ое	Время в миллисекундах работы таймера для удаления человека из очереди.

Продолжение таблицы 2

2-ое	Максимальный размер очереди, ограничивающий максимальное количество хранимых номеров.
3-ее	Максимальный номер, который может быть причислен человеку.

1.2 Проектирование функциональной схемы

1.2.1 Микроконтроллер ATmega8

Основным элементом разрабатываемого устройства является микроконтроллер (МК). Существует множество семейств МК, для разработки выберем из тех, что являются основными [1]:

- 8051 – это 8-битное семейство МК, разработанное компанией Intel.
- PIC – это серия МК, разработанная компанией Microchip;
- AVR – это серия МК разработанная компанией Atmel;
- ARM – одним из семейств процессоров на базе архитектуры RISC, разработанным компанией Advanced RISC Machines.

Сравнение семейств показано в таблице 3.

Таблица 3 – Сравнение семейств МК

Критерий	8051	PIC	AVR	ARM
Разрядность	8 бит	8/16/32 бит	8/32 бит	32 бит, иногда 64 бит
Интерфейсы	UART, USART, SPI, I2C	PIC, UART, USART, LIN, CAN, Ethernet, SPI, I2S	UART, USART, SPI, I2C, иногда CAN, USB, Ethernet	UART, USART, LIN, I2C, SPI, CAN, USB, Ethernet, I2S, DSP, SAI, IrDA
Скорость	12 тактов на инстру-кцию	4 такта на инструкцию	1 такт на инструкцию	1 такт на инструкцию

Продолжение таблицы 3

Память	ROM, SRAM, FLASH	SRAM, FLASH	Flash, SRAM, EEPROM	Flash, SDRAM, EEPROM
Энергопо-требление	Среднее	Низкое	Низкое	Низкое
Объем FLASH памяти	До 128 Кб	До 512 Кб	До 256 Кб	До 192 Кб

Было выбрано семейство AVR, так как в модели счетчика монет важна скорость выполнения операций при регистрации падения монеты, что могут предоставить МК серии AVR, а также они имеют больший объем памяти по сравнению с семейством ARM. А также с МК серии AVR уже был опыт работы, что упростит разработку системы.

AVR в свою очередь делятся на семейства:

1. TinyAVR, имеющие следующие характеристики:
   - Flash-память до 16 Кбайт;
   - RAM до 512 байт;
   - ROM до 512 байт;
   - число пинов (ножек) ввода-вывода 4–18;
   - небольшой набор периферии.
2. MegaAVR, имеющие следующие характеристики:
   - FLASH до 256 Кбайт;
   - RAM до 16 Кбайт;
   - ROM до 4 Кбайт;
   - число пинов ввода-вывода 23–86;
   - расширенная система команд (ассемблер и C) и периферии.
3. XMEGA AVR, имеющие следующие характеристики:

- FLASH до 384 Кбайт;
- RAM до 32 Кбайт;
- ROM до 4 Кбайт;
- четырехканальный контроллер DMA (для быстрой работы с памятью и вводом/выводом).

Выберем подсемейство MegaAVR, так как оно имеет больше пинов и объем памяти, чем TinyAVR, а также поддерживает C. XMEGA AVR не был выбран так как они лучше по характеристикам, чем необходимо, то есть весь их потенциал не будет раскрыт.

В подсемействе MegaAVR семейства AVR был выбран МК – ATmega8, обладающий всем необходимым функционалом для реализации проекта:

- интерфейс SPI для программирования МК;
- интерфейс UART для обмена данными;
- 1 Кбайт ОЗУ;
- два 8-разрядных счетчика, один 16-разрядный;
- 8 Кбайт FLASH памяти;
- частота работы до 16 Мгц;

Это экономичный 8-разрядный микроконтроллер, основанный на AVR RISC архитектуре. ATmega8 обеспечивает производительность 1 миллион операций в секунду на 1 МГц синхронизации за счет выполнения большинства из 130 инструкций за один машинный цикл и позволяет оптимизировать потребление энергии за счет изменения частоты синхронизации. Структурная схема МК показана на рисунке 1 и УГО на рисунке 2 [2].

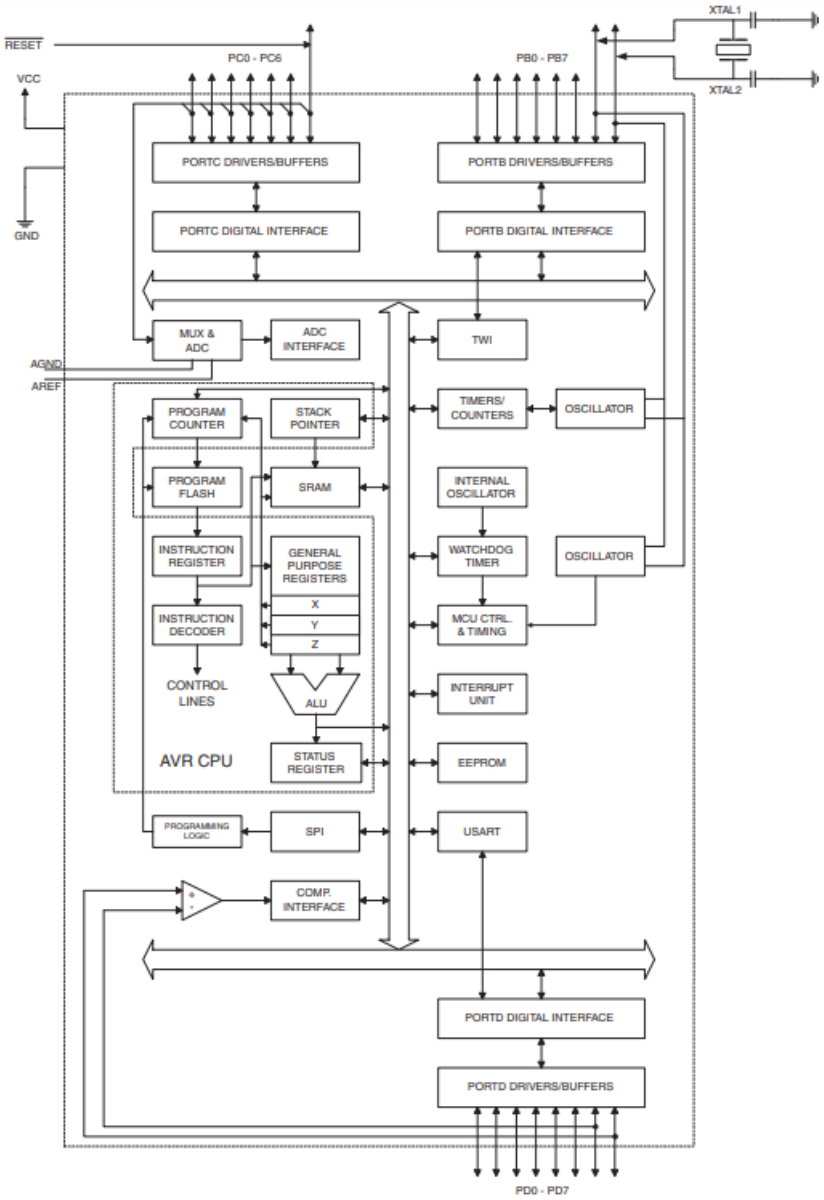

Рисунок 1 – Структурная схема МК ATmega8

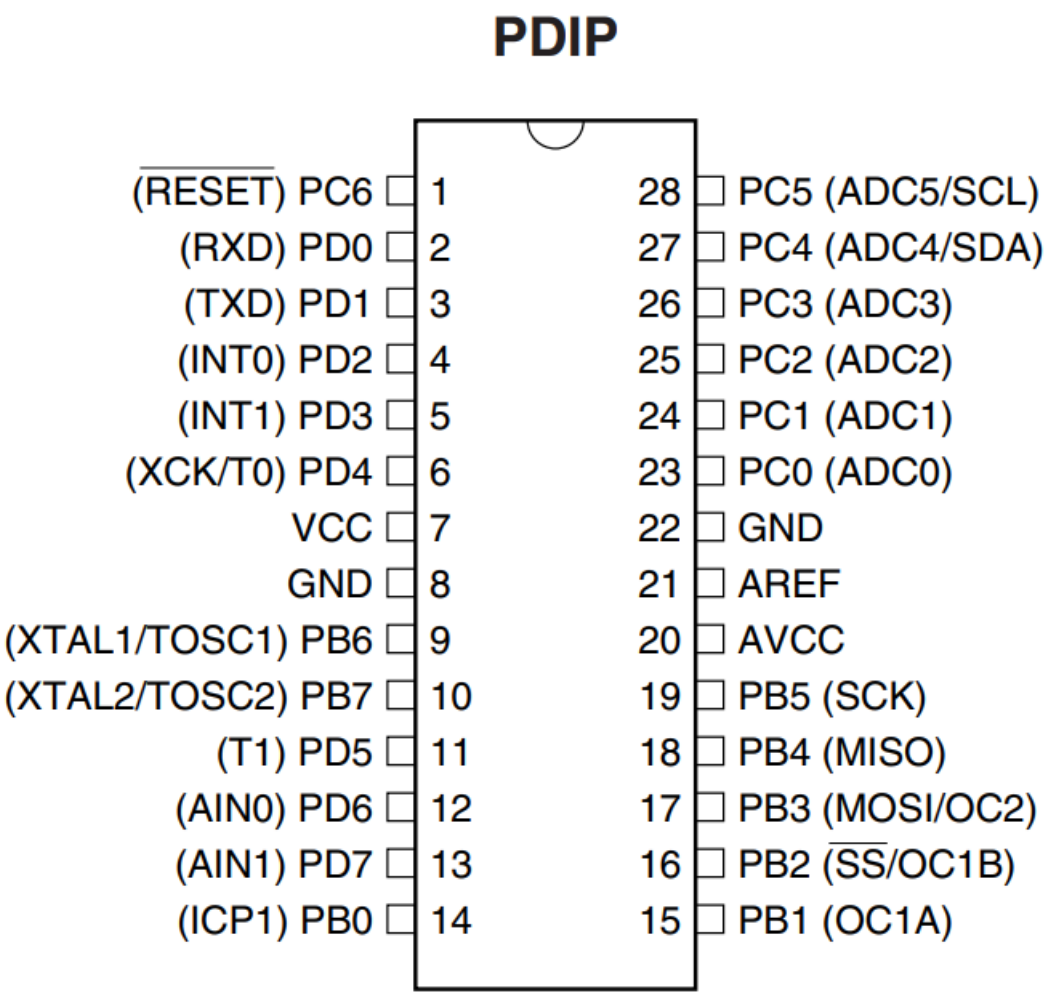

Рисунок 2 – УГО МК ATmega8

Он обладает следующими характеристиками:

1. RISC архитектура:

– мощная система команд с 130 инструкциями, большинство из которых выполняются за один машинный цикл;

– 32 восьмиразрядных рабочих регистров общего назначения;

– производительность до 16 миллиона операций в секунду при частоте 16 МГц;

– встроенное умножение за 2 цикла.

2. Энергонезависимые память программ и данных:

– 8 Кб внутрисхемно-программируемой flash-памяти с возможностью самозаписи;

– возможность внутрисхемного программирования программой во встроенном секторе начальной загрузки;

– циклы записи/чтения: 10.000 Flash/100.000 EEPROM

– возможность считывания во время записи;

– 512 байт ЭППЗУ (EEPROM);

– 1 Кбайт внутреннего статического ОЗУ;

– программирование битов защиты программного обеспечения.

3. Периферийные устройства:

– два 8-разрядных таймера-счетчика с отдельным предделителем и режимом компаратора;

– один 16-разрядный таймер-счетчик с отдельным предделителем, режимом компаратора и режимом захвата;

– три канала ШИМ (широтно-импульсная модуляция);

– программируемый последовательный UART (устройство синхронной или асинхронной приемопередачи);

– последовательный интерфейс SPI с режимами главный и подчиненный;

– программируемый сторожевой таймер с отдельным встроенным генератором;

– битоориентированный двух-канальный последовательных интерфейса

– встроенный аналоговый компаратор.

4. Специальные функции микроконтроллера:

– сброс при подаче питания и программируемый супервизор питания;

– встроенный калиброванный RC-генератор;

– внутренние и внешние источники запросов на прерывание;

– три режима управления энергопотреблением: холостой ход (Idle), пониженное потребление (Power-down), сохранение энергии

(Power-save), режим пониженного шума ADC (ADC noise reduction), и дежурный (Standby).

5. Ввод-вывод:
   - 23 программируемых линий ввода-вывода;
   - 28 регистров ввода/вывода.
6. Напряжение питания: 4.5 – 5.5В.
7. Рабочая частота: 1 – 16 МГц.

1.2.1.1 Используемые элементы

Для функционирования контроллера электронной очереди на МК ATmega8 задействованы не все элементы его архитектуры. Выделим и опишем те, что используются во время функционирования схемы [4].

Порты C, B, D – назначения каждого из них описано в разделе 1.2.1.2.

Указатель стека – используется для работы со стеком, при вызове подпрограмм. В коде они присутствуют.

SRAM – статическая память МК, где хранятся объявленные переменные.

Регистры общего назначения – предназначены для хранения операндов арифметико-логических операций, а также адресов или отдельных компонентов адресов ячеек памяти.

АЛУ – блок процессора, который под управлением устройства управления служит для выполнения арифметических и логических преобразований над данными.

SREG – регистр состояния, содержит набор флагов, показывающих текущее состояние работы микроконтроллера.

SPI – интерфейс для связи МК с другими внешними устройствами. В проекте используется только для прошивки МК.

Счетчик команд – используется для указания следующей команды выполняемой программы.

Память Flash – память МК, в котором хранится загруженная в него программа.

Регистры команды – содержит исполняемую в текущий момент (или следующий) команду, то есть команду, адресуемую счетчиком команд.

Дешифратор команд – блок, выделяющий код операции и операнды команды, а затем вызывающий микропрограмму, которая исполняет данную команду.

Сигналы управления – синхронизируют обработку данных.

Логика программирования – устанавливает логику того, как программа будет вшита в МК.

Таймеры/счетчики – включает в себя 8-разрядные таймеры T0, T2 и 16-разрядный таймер T1. Во время работы данной программы используется только T2.

Управление синхронизацией и сбросом (MCU CTRL. & Timing) – в этом блоке обрабатываются тактовые сигналы и принимается сигнал сброса.

Модуль прерываний – контроллер прерываний обрабатывает внешние прерывания и прерывания от периферийных устройств МК (таймеров, портов ввода/вывода). В проекте используются прерывания как внутренние так и внешние.

UART – через этот интерфейс в МК передается информация из ПЭВМ.

Генератор – генератор тактовых импульсов. Необходим для синхронизации работы МК.

1.2.1.2 Распределение портов

МК ATmega8 содержит пять портов – B, C, и D. Опишем назначение тех, что используются в данной системе для ее функционирования.

Порт B:

- PB0 – отправка 0 бита данных LCD дисплея;
- PB1 – отправка 1 бита данных LCD дисплея;
- PB2 – отправка 2 бита данных LCD дисплея;

Порт C:

- PC0 – отвечает за подачу звукового сигнала с помощью зуммера;

- PC1 – отвечает за управление пином RS LCD дисплея;
- PC2 – отвечает за управление пином RW LCD дисплея;
- PC3 – отвечает за управление пином E LCD дисплея;
- PC4 – отправка 6 бита данных LCD дисплея;
- PC5 – отправка 7 бита данных LCD дисплея;
- PC6 – кнопка, отвечающая за RESET микроконтроллера.

Порт D:

- PD0 – отвечает за вывод вводимой информации для проверки корректности ввода;
- PD1 – отвечает за принятие вводимой информации;
- PD2 – кнопка, отвечающая за добавление человека в очередь;
- PD3 – кнопка, отвечающая за удаление человека из очереди;
- PD5 – отправка 3 бита данных LCD дисплея;
- PD6 – отправка 4 бита данных LCD дисплея;
- PD7 – отправка 5 бита данных LCD дисплея.

1.2.1.3 Организация памяти

Схема организации памяти МК ATmega8 показана на рисунке 3.

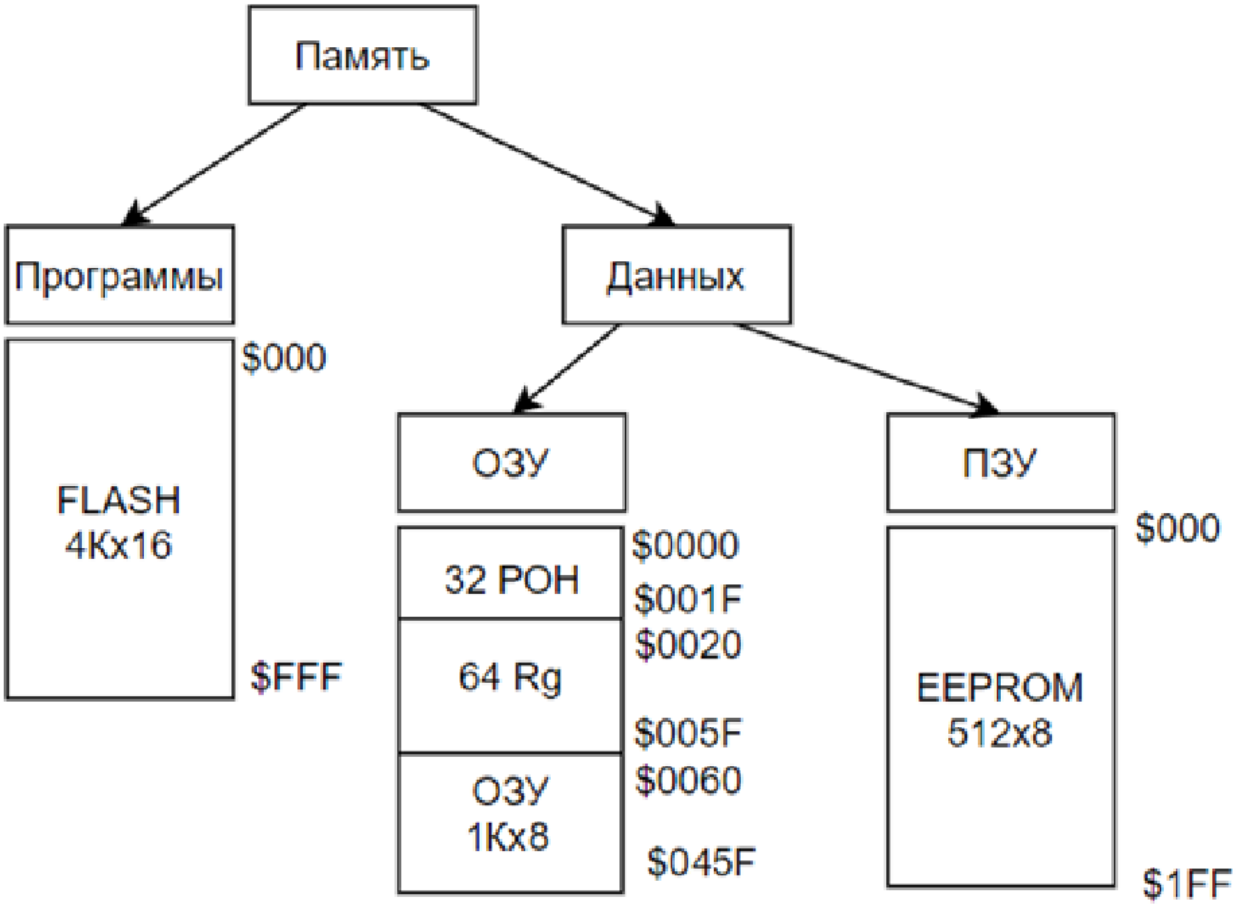

Рисунок 3 – Организация памяти МК ATmega8

Память программы предназначена для хранения последовательности команд, управляющих функционированием микроконтроллера, и имеет 16-ти битную организацию.

Все AVR имеют Flash-память программ, в выбранном МК ее объем 4Кх16, то есть длина команды 16 разрядов. Информацию в flash-память можно мгновенно стереть и записать. Заносится с помощью программатора.

Память данных делиться на три части:

1. Регистровая память – 32 регистра общего назначения и служебные регистры ввода/вывода.

2. Оперативная память (ОЗУ) – МК ATmega8 имеет объем внутреннего SRAM 1 Кбайт (с адреса $0060 до $045F). Число циклов чтения и записи в RAM не ограничено, но при отключении питания вся информация теряется.

3. Энергонезависимая память (EEPROM) – эта память доступна МК в ходе выполнения, для хранения промежуточных результатов. В МК ATmega8 ее объем 512 байт. Также в нее могут быть загружены данные через программатор.

1.2.2 Передача данных в ПЭВМ

Передача данных в ПЭВМ происходит через драйвер MAX232. MAX232 – интегральная схема, преобразующая сигналы последовательного порта RS-232 в цифровые сигналы.

RS-232 – стандарт физического уровня для синхронного и асинхронного интерфейса (USART и UART). Обеспечивает передачу данных и некоторых специальных сигналов между терминалом и устройством приема. Сигнал, поступающий от интерфейса RS-232, через преобразователь передается в микроконтроллер на вход RxD.

К внешнему устройству MAX232 подключен через разъем DB-9.

Внутреннее изображение MAX232 показано на рисунке 4. Назначение пинов описано в таблице 4 [5].

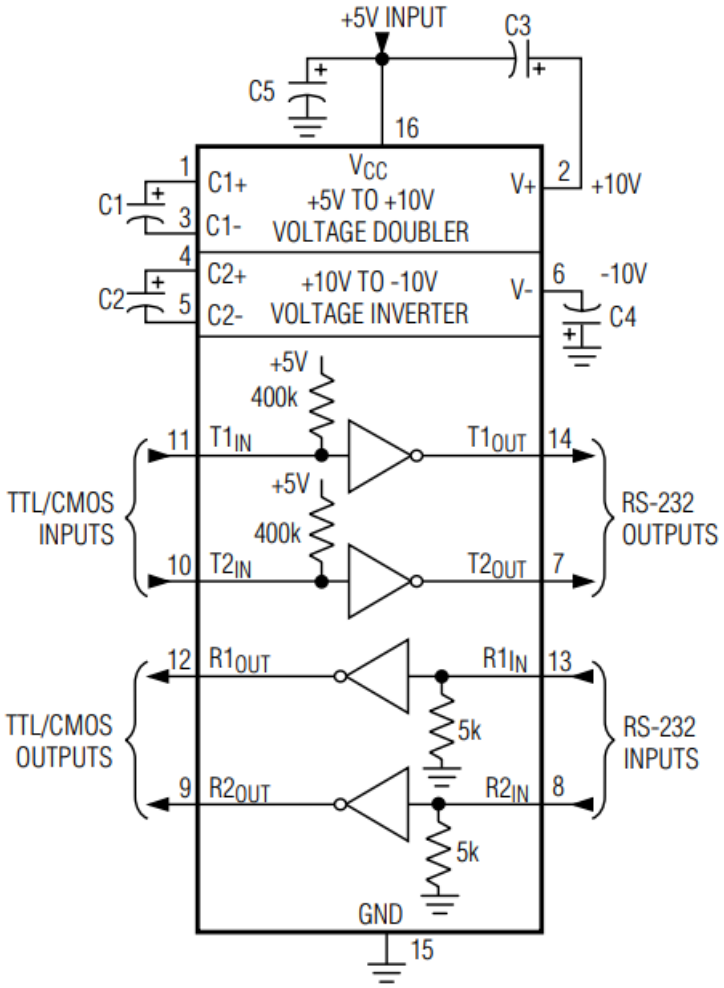

Рисунок 4 – Преобразователь MAX232

Таблица 4 - Назначение пинов MAX232

Номер	Имя	Тип	Описание
1	C1+	–	Положительный вывод C1 для подключения конденсатора
2	V+	O	Выход положительного заряда для накопительного конденсатора
3	C1-	–	Отрицательный вывод C1 для подключения конденсатора
4	C2+	–	Положительный вывод C2 для подключения конденсатора
5	C2-	–	Отрицательный вывод C2 для подключения конденсатора

Продолжение таблицы 4

6	V-	O	Выход отрицательного заряда для накопительного конденсатора
7, 14	T2OUT, T1OUT	O	Вывод данных по линии RS232
8, 13	R2IN, R1IN	I	Ввод данных по линии RS232
9, 12	R2OUT, R1OUT	O	Вывод логических данных
10, 11	T2IN, T1IN	I	Ввод логических данных
15	GND	–	Земля
16	Vcc	–	Напряжение питания, подключение к внешнему источнику питания 5 В

Когда микросхема MAX232 получает на вход логический "0" от внешнего устройства, она преобразует его в напряжение от +5 до +15В, а когда получает логическую "1" - преобразует ее в напряжение от -5 до -15В, и по тому же принципу выполняет обратные преобразования от RS-232 к внешнему устройству.

1.2.3 Настройка канала передачи

Для передачи информации из МК используется последовательный интерфейс UART, для корректной работы необходимо настроить регистры управления UCSRA, UCSRB и UCSRC [6].

UCSRA. Биты регистра UCSRA показаны в таблице 5.

Таблица 5 – биты регистра UCSRA

Номер	7	6	5	4	3	2	1	0
Название	RXC	TxC	UDRE	FE	DOR	PE	U2X	MPCM
Доступ	R	R/W	R	R	R	R	R	R

Во время работы системы используются только биты RXC и UDRE.

TXC – флаг завершения передачи. Устанавливается в 1, когда передача данных завершена, что позволяет сигнализировать о том, что данные успешно отправлены.

UDRE – флаг опустошения регистра данных. Устанавливается в 1, если буфер передатчика пуст. Используется при отправке данных из МК: когда UDRE == 1, значит, регистр UDR пуст и в него можно загружать новые данные для передачи.

UCSRB. Биты регистра UCSRB показаны в таблице 6.

Таблица 6 – биты регистра UCSRB

Номер	7	6	5	4	3	2	1	0
Название	RXCIE	TXCIE	UDRIE	RXEN	TXEN	UCSZ	RXB8	TXB8
Доступ	R	R/W	R/W	R/W	R/W	R/W	R	R/W

Во время работы системы, для управления приемом и передачей информации, будут использоваться только биты TXEN и RXEN.

TXEN – разрешение передачи. Если бит сбрасывается во время передачи, то передатчик выключается только после завершения текущей передачи.

RXEN – разрешение приема, включающий работу приемника.

UCSRC. Биты регистра UCSRC показаны в таблице 7.

Таблица 7 – биты регистра UCSRC

Номер	7	6	5	4	3	2	1	0
Название	URSEL	UMSEL	UPM1	UPM0	USBS	UCSZ1	UCSZ0	UCPOL
Доступ	R	R/W	R/W	R/W	R/W	R/W	R	R/W

Во время работы системы будут использоваться только биты URSEL, UCSZ0 и UCSZ1.

URSEL – позволяет выбрать между регистрами UCSRC и UBRRH, находящимися в одном адресном пространстве. При значении URSEL = 1 данные записываются в UCSRC, при значении URSEL = 0 – в UBRRH.

UCSZ0 и UCSZ1 – устанавливают формат посылки данных. При значении каждого из этих битов равным 1 обеспечивается 8-битная посылка.

Скорость передачи определяется выражением:

$$BAUD = \frac{f_{osc}}{16(UBRR + 1)}$$

где BAUD — скорость передачи (бод);

f_{osc} — тактовая частота микроконтроллера (Гц);

UBRR — содержимое регистров UBRR0H, UBRR0L контроллера скорости передачи.

Зададим скорость 9600 бод. Получится:

$$UBRR = \frac{f_{osc}}{16 * BAUD} - 1 = \frac{8 * 10^6}{16 * 9600} - 1 = 51_{10}$$

UBRRL будет принимать значение 51.

1.2.4 Настройка внешних прерываний

Была задействована системная функции ISR – прерывания внешних прерываний на INT0 и INT1, и прерывания таймера. Функции вызываются, нажимаются кнопки, подключенные пину PD2, PD3 и таймер T2, работающий в режиме CTC. Для работы внешних прерываний необходимо настроить регистры MCUCR и GICR [7].

Биты регистра MCUCR показаны в таблице 8.

Таблица 8 – биты регистра MCUCR

Номер	7	6	5	4	3	2	1	0
Название	SM2	SE	SM1	SM0	ISC11	ISC10	ISC01	ISC00
Доступ	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Во время работы системы, для управления приемом и передачей информации, будут использоваться только биты ISC01 и ISC11. Принципы

работы прерываний в зависимости от битов ISCx1 и ISCx0 представлены на таблице 9.

Таблица 9 – значение битов ISCx1 и ISCx0

ISCx1	ISCx0	Описание
0	0	Низкий уровень INTx генерирует запрос на прерывание
0	1	Любое изменение логическое состояния в INTx генерирует запрос на прерывание
1	0	Спадающий фронт INTx генерирует запрос на прерывание
1	1	Нарастающий фронт INTx генерирует запрос на прерывание

Биты регистра GICR показаны в таблице 10.

Таблица 10 – биты регистра GICR

Номер	7	6	5	4	3	2	1	0
Название	INT1	INT0	-	-	-	-	IVSEL	IVCE
Доступ	R/W	R/W	R	R	R	R	R/W	R/W

Во время работы системы, для управления приемом и передачей информации, будут использоваться только биты INT1 и INT0.

INT1 – бит разрешения прерывания INT1. Если устанавливается в 1, то прерывание разрешено, если в 0, то запрещено.

INT0 – бит разрешения прерывания INT0. Если устанавливается в 1, то прерывание разрешено, если в 0, то запрещено.

1.2.5 Настройка таймера

Для того, чтобы настроить таймер T2 используем регистр TCCR2 [8].

Биты регистра TCCR2 показаны в таблице 11.

Таблица 11 – биты регистра TCCR2

Номер	7	6	5	4	3	2	1	0

Продолжение таблицы 11

Название	FOC 2	WGM2 0	COM2 1	COM2 0	WGM2 1	CS2 2	CS2 1	CS2 0
Доступ	W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Во время работы системы будут использоваться только биты WGM21, CS22, CS21 и CS20. Принципы работы прерываний в зависимости от битов CS22, CS21 и CS20 представлены на таблице 12.

Таблица 12 – значение битов CS22, CS21 и CS20

CS22	CS21	CS20	Описание
0	0	0	Таймер/счетчик остановлен
0	0	1	$Clk_{T2S}/1$
0	1	0	$Clk_{T2S}/8$
0	1	1	$Clk_{T2S}/32$
1	0	0	$Clk_{T2S}/64$
1	0	1	$Clk_{T2S}/128$
1	1	0	$Clk_{T2S}/256$
1	1	1	$Clk_{T2S}/1024$

Принцип работы таймера в зависимости от битов битов WGM20 и WGM21 представлены на таблице 13.

Таблица 13 – значение битов WGM20 и WGM21

WGM20	WGM21	Описание
0	0	Нормальный режим
0	1	Генерация ШИМ сигнала с фазовой коррекцией.
1	0	Сброс таймера при совпадении
1	1	Генерация ШИМ сигнала с быстрым обновлением выходного сигнала.

Биты регистра TIMSK показаны в таблице 14.

Таблица 14 – биты регистра TIMSK

Номер	7	6	5	4	3	2	1	0
Название	OCIE2	TOIE2	TICIE1	OCIE1A	OCIE1B	TOIE1	-	TOIE0
Доступ	W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Во время работы системы будет использоваться только бит OCIE2.

OCIE2 – флаг разрешения прерывания по совпадению таймера/счетчика2. Если устанавливается в 1, тогда разрешается прерывание по совпадению содержимого регистра сравнения (OCR2) и состояния таймера/ счетчика2, если устанавливается в 0, то оно запрещается.

МК работает при частоте 2 МГц, а предделитель равен 1024. Таймер работает по сравнению и OCR2 = 195, получается, таймер срабатывает каждую 0,1 секунду.

В начале работы системы создаем переменную timer_counter, отвечающую за время таймера в секундах, и обнуляем ее. Каждый раз, когда T2 становится равным OCR2, значение timer_counter увеличивается на 1. Затем создаем переменную TIME и даем ей значение в милисекундах, это будет значение, при достижении которого с помощью timer_counter, срабатывает приглашение человека из очереди.

1.2.6 LCD-дисплей

Информация о людях в очереди также отображается на ЖК-дисплее LM016L.

Жидкокристаллический дисплей (ЖК-дисплей) представляет собой тип плоскоэкранного дисплея, использующий жидкие кристаллы для визуализации данных.

Основные компоненты LCD-дисплея включают:

- Жидкокристаллическая матрица – прямоугольная сетка пикселей или символов;

- Подсветка (если предусмотрена) – светодиоды, освещающие экран;
- Контроллер дисплея — отвечает за преобразование команды микроконтроллера в управляющие сигналы для работы матрицы;
- Контактная панель — контакты для передачи данных и сигналов управления.

Контроллер дисплея выполняет следующие функции: принятие команд и данных, управление матрицей дисплея, буферизация данных, формирование сигналов для управления ЖК-элементами и синхронизация работы. В его состав входят:

- DDRAM — оперативная память объемом 32 байта, где хранятся символы для отображения;
- CGROM — постоянная память, содержащая стандартный набор символов (буквы, цифры, специальные знаки);
- CGRAM — оперативная память для создания до 8 пользовательских символов, например, логотипов.

Схема LM016L показана на рисунке 5. Назначение пинов описано в таблице 15.

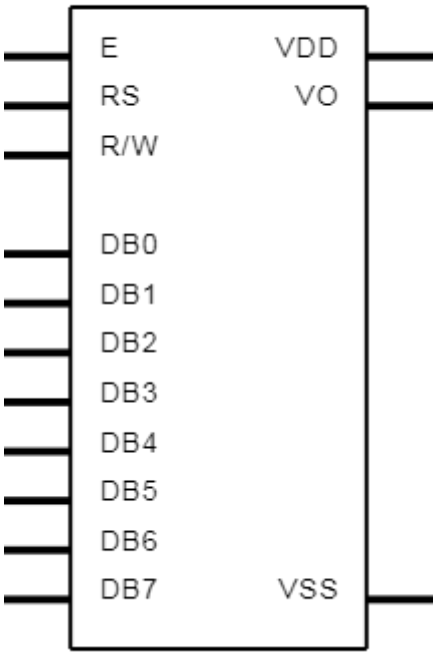

Рисунок 5 – Схема LM016L

Таблица 15 – Назначение пинов LM016L

Номер	Имя	Тип	Описание
1	E	I	Разрешающий сигнал
2	RS	I	Бит команд/данных
3	R/W	I	Бит чтения/записи
4	DB0	I	Бит шины данных 1
5	DB1	I	Бит шины данных 2
6	DB2	I	Бит шины данных 3
7	DB3	I	Бит шины данных 4
8	DB4	I	Бит шины данных 5
9	DB5	I	Бит шины данных 6
10	DB6	I	Бит шины данных 7
11	DB7	I	Бит шины данных 8
12	VDD	-	Подключается к питанию +5V
13	V0	-	Ответственен за контрастность
14	VSS	-	Подключается к земле

При работе с LCD-дисплеем устанавливаются биты чтения или записи в зависимости от выполняемого действия, биты команды или данных в зависимости от типа передаваемой информации, а также непосредственно передаются сами биты данных [9].

1.2.7 Устройство для подачи звукового сигнала

Для подачи звукового сигнала используется зуммер модели MCKPX-G1203UB-K4065, который является экономичным и подходящим решением для выполнения поставленной задачи.

Работа магнитного зуммера основана на взаимодействии электромагнитного поля катушки и постоянного магнита. Это взаимодействие приводит в движение мембрану, вызывая звуковые колебания. Магнитный

зуммер функционирует с переменным током, частота которого определяет тон звука.

При прохождении переменного тока через катушку создается переменное магнитное поле. При изменении направления тока магнитное поле катушки также меняет свое направление. Взаимодействие постоянного магнитного поля и переменного поля катушки вызывает циклическое притягивание и отталкивание мембраны. Эти движения заставляют мембрану вибрировать с частотой, соответствующей частоте подаваемого сигнала. Вибрация мембраны создает звуковые волны, распространяющиеся в окружающей среде, частота которых соответствует частоте электрического сигнала [10].

Назначение пинов зуммера MCKPX-G1203UB-K4065 представлено в таблице 16.

Таблица 16 – Пины зуммера MCKPX-G1203UB-K4065.

Номер	Имя	Описание
1	GND	Подключается к земле
2	Signal	Подключается к микроконтроллеру.

1.2.8 Сброс

В микроконтроллерах AVR встроена схема сброса, где сигнал RESET уже подтянут внутренним резистором на 100 кОм к Vcc. Для реализации ручного сброса выход RESET подключают к питанию через резистор на 10 кОм и конденсатор емкостью 100 мкФ. При подаче питания конденсатор изначально разряжен, поэтому напряжение на выводе RESET близко к нулю, и микроконтроллер остается в состоянии сброса. Со временем конденсатор заряжается через резистор, напряжение на RESET повышается до логической единицы, и микроконтроллер начинает работу.

1.2.9 Построение функциональной схемы

На основе всех вышеописанных сведений была спроектирована функциональная схема разрабатываемой системы, показанная в приложении Б [11, 12].

1.3 Проектирование принципиальной схемы

1.3.1 Разъем программатора

Для программирования МК используется программатор, для его подключения необходим специальный разъем. Будет использован разъем IDC-06MS. Подключение программатора осуществляется при помощи интерфейса SPI, под что на МК ATmega8 задействован порт PB.

Он имеет следующие разъемы для подключения к МК:

- MISO – для передачи данных от микроконтроллера в программатор;
- SCK – тактовый сигнал;
- MOSI – для передачи данных от программатора в микроконтроллер;
- Reset – сигналом на RESET программатор вводит контроллер в режим программирования.

1.3.2 Подключение цепи питания

Для работы схемы, на нее необходимо подать напряжение, для этого будет использован блок питания ARDV-15-5B, имеющий следующие технические характеристики:

- выходной ток: 3 А;
- выходное напряжение: 5 В;
- входное напряжение: 100 – 240 В;
- пусковой ток: 40 А.

Диаметр центрального проводника – 2 мм и диаметр Jack-а – 2 мм.

Для подключения питания к схеме, будет использовано гнездо питания DS-201, имеющий диаметр центрального проводника – 2 мм и диаметр Jack-а – 2 мм.

1.3.3 Расчет сопротивления резисторов

Номинал резистора определяется на основе максимального тока, протекающего через цепь. Максимальный ток, который может подать цифровой пин ATmega8, составляет 40 мА (0,04 А). Согласно закону Ома, при напряжении 5 В сопротивление рассчитывается следующим образом: $R = \frac{\mathrm{U}}{I} = \frac{5\mathrm{B}}{0{,}04\mathrm{A}}$ = 125 Ом.

Таким образом, минимально допустимое сопротивление резистора составляет 125 Ом. Однако, чтобы подтягивающий или стягивающий резистор не оказывал значительного влияния на другие части цепи при замыкании, рекомендуется выбирать значительно большее сопротивление. В данном случае предпочтительным будет номинал 10 кОм.

1.3.4 Расчет потребляемой мощности

Потребляемая мощность – это мощность, потребляемая интегральной схемой, которая работает в заданном режиме соответствующего источника питания.

Чтобы рассчитать суммарную мощность, рассчитаем мощность каждого элемента. На все микросхемы подается напряжение +5В. Мощность, потребляемая один устройством, в статическом режиме, рассчитывается формулой:

$$P = U * I$$

где U – напряжение питания (В);

I – ток потребления микросхемы (мА).

Расчет потребляемого напряжения для каждой микросхемы показан в таблице 17.

Таблица 17 – Потребляемая мощность

Микросхема	Ток потребления, мА	Потребляемая мощность, мВт	Количе-ство устройств	Суммарная потребляемая мощность, мВт
ATmega8	12	60	1	60

Продолжение таблицы 17

MAX232	8	40	1	40
MCKPX-G1203UB-K4065	30	150	1	150
LM016L	3	15	1	15

Потребление тока ATmega8 показано на рисунке 6. Подсчет был проведен из расчета температуры в 25 градусов Цельсия.

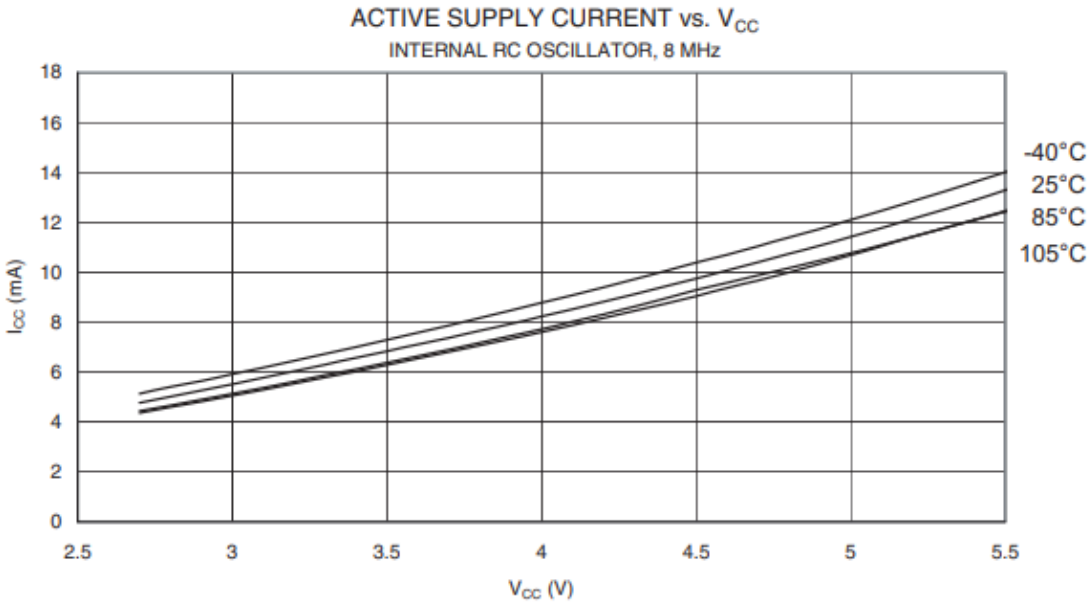

Рисунок 6 – График потребления при активном использовании.

Также в схеме используются 3 резистора CF-1 с номиналом 10 кОм.

$P_{суммарная} = P_{ATmega8} + P_{MAX232} + P_{LM016L} + P_{MCKPX-G1203UB-K4065} = 60 + 40 + 15 + 150 + 250*3 = 1015$ мВт

Суммарная потребляемая мощность системы равна 1015 мВт.

1.3.5 Построение принципиальной схемы

На основе всех вышеописанных сведений была спроектирована принципиальная схема разрабатываемой системы, показанная в приложении Б [11, 12].

1.4 Алгоритмы работы системы

1.4.1 Функция Main

Работа программы начинается с функции main, из которой последовательно вызываются все остальные функции.

Сначала выполняется вызов функции External_Interrupts_Init, которая отвечает за инициализацию внешних прерываний. Затем вызывается функция LCD_Init, настраивающая порты LCD-дисплея и сам дисплей. После этого запускается функция USART_Init(9600), которая инициализирует USART с установленной скоростью передачи 9600 бит/с. Далее инициализируется таймер через функцию Timer2_Init.

Следующим шагом разрешаются прерывания с помощью функции sei. Затем выполняются вызовы функций LCD_String("Queue is empty"), выводящей на дисплей строку "Queue is empty", и USART_SendString("Queue is empty;\r\n"), передающей через USART строку "Queue is empty;\r\n". После этого запускается функция Button_Init, которая инициализирует пины для кнопок.

В течение работы микроконтроллера происходит проверка размера очереди и сброс таймера, если она оказывается пустой. Схема алгоритма показана на рисунке 7.

Рисунок 7 – Функция Main

1.4.2 Используемые при работе подпрограммы

В начале работы программы инициализируются внешние прерывания. Происходит настройка регистров, необходимых для работы с внешними прерываниями, описанных в пункте 1.2.4. Схема алгоритма показана на рисунке 8.

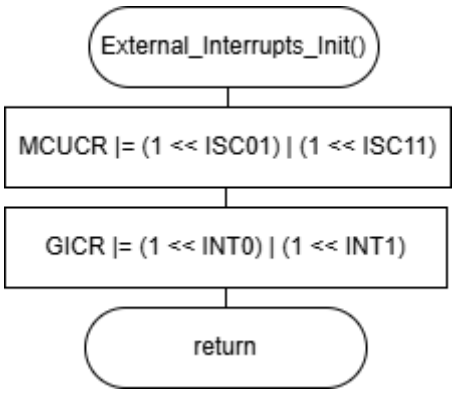

Рисунок 8 – Схема алгоритма функции External_Interrupts_Init

Рассмотрим инициализацию портов LCD дисплея. Сначала происходит активация PC1-3 для передачи команд, затем активация PB0-2, PD5-7 и PC4-5 для передачи команд. После задержки происходит настройка LCD дисплея: выбирается 8-битный режим работы в 2 строки, включается дисплей и выключается курсор, включается автоматическое перемещение курсора вправо, происходит очистка экрана и установка курсора в начальную позицию. Схема алгоритма показана на рисунке 9.

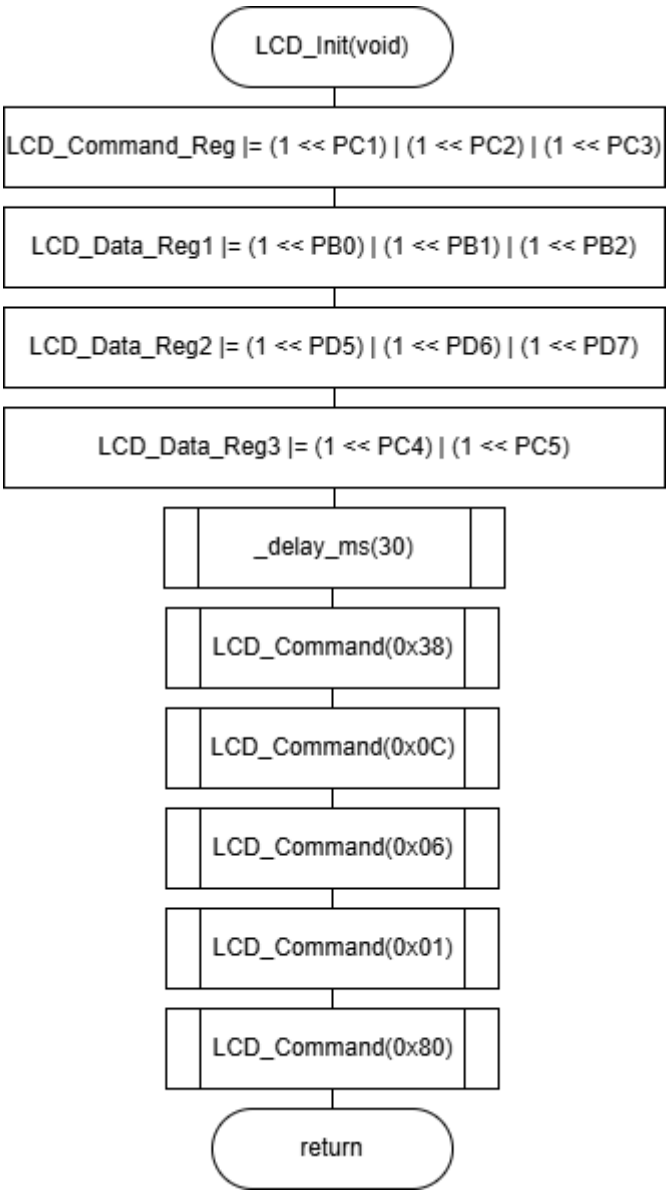

Рисунок 9 – Схема алгоритма подпрограммы LCD_Init

Рассмотрим передачу команд на LCD дисплей. Передаем на пины PB0-2, PD5-7 и PC4-5 части команды, переключаем LCD-дисплей в режим команды и записи и посылаем сигнал активации. Затем после завершаем сигнал активации. Схема алгоритма показана на рисунке 10.

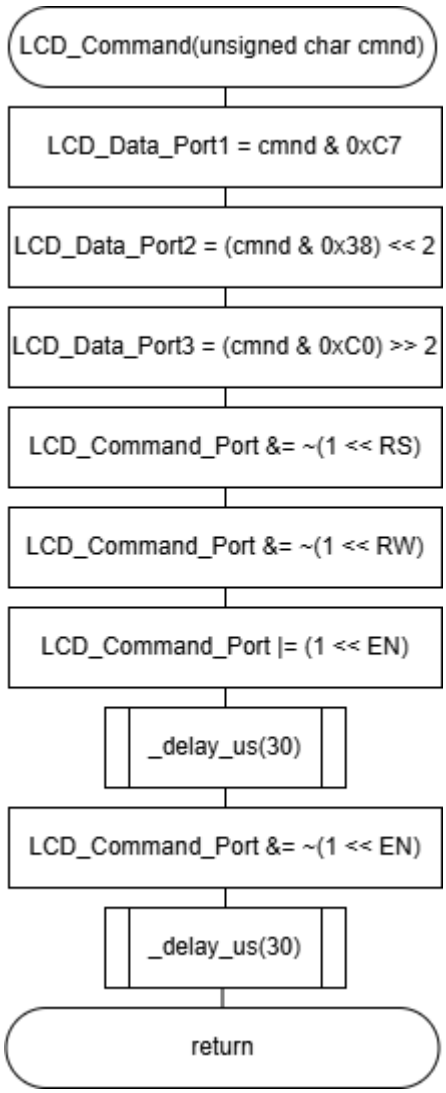

Рисунок 10 – Схема алгоритма подпрограммы LCD_Command

Рассмотрим инициализацию USART. Происходит настройка регистров, необходимых для работы с USART разобранных в пункте 1.2.3. Схема алгоритма показана на рисунке 11.

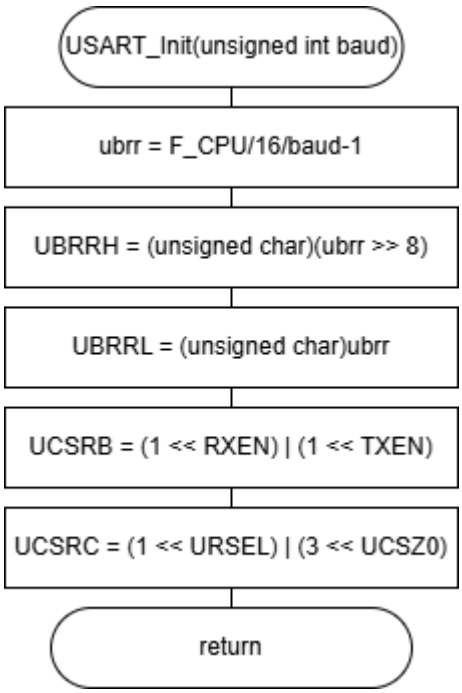

Рисунок 11 – Схема алгоритма функции USART_Init

Рассмотрим инициализацию таймера Timer2. Происходит настройка всех регистров, необходимых для работы с Timer2 разобранных в пункте 1.2.5. Схема алгоритма показана на рисунке 12.

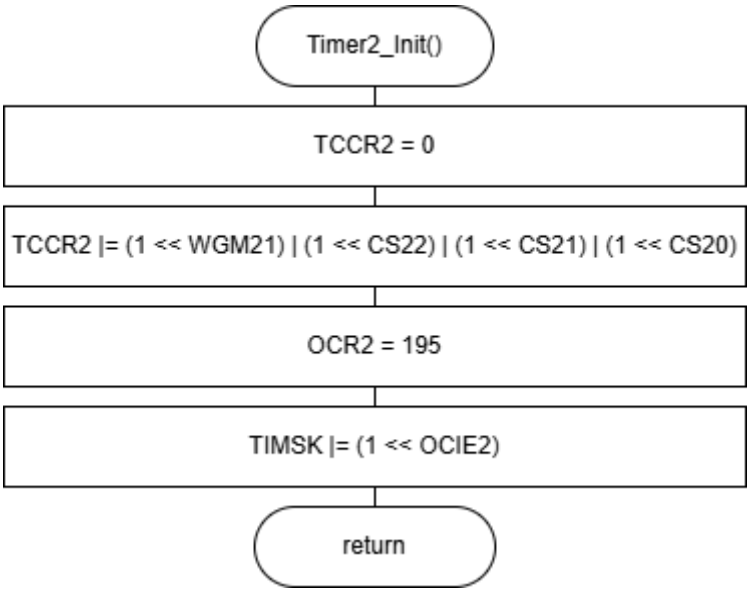

Рисунок 12 – Схема алгоритма функции Timer2_Init

Рассмотрим передачу строки на LCD дисплей. Передаем на пины PB0-2, PD5-7 и PC4-5 части строки, переключаем LCD-дисплей в режим данных и записи и посылаем сигнал активации. Затем после завершаем сигнал активации. Схема алгоритма показана на рисунке 13.

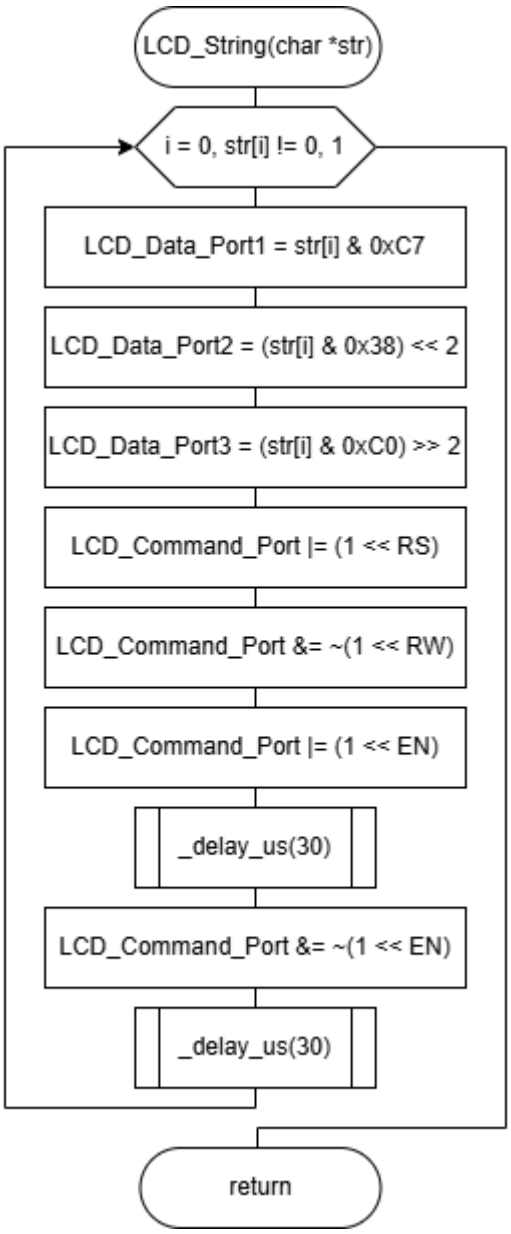

Рисунок 13 – Схема алгоритма функции LCD_String

Рассмотрим передачу строки на USART, где посимвольно передается строка до тех пор, пока она не окажется пустой. Схема алгоритма показана на рисунке 14.

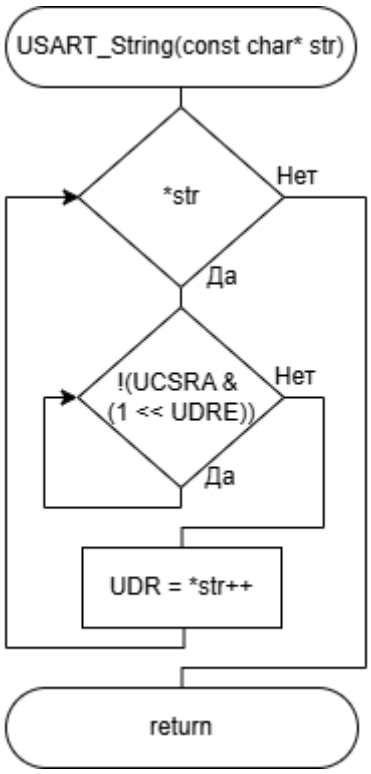

Рисунок 14 – Схема алгоритма функции USART_String

Рассмотрим инициализацию кнопок. Настраиваем пин PC0 на выход и пины PD2-3 на вход. Схема алгоритма показана на рисунке 15.

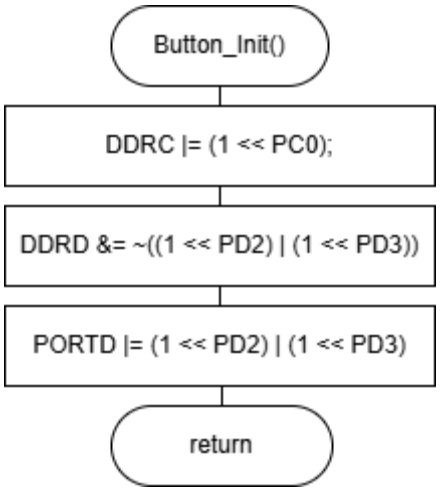

Рисунок 15 – Схема алгоритма функции Button_Init

Рассмотрим прерывание при нажатии кнопки добавления человека в очередь. Если размер очереди меньше максимального и равен нулю, то номер добавленного человека будет равен сохраненному значению, если номер не равен нулю, то проверяется номер предпоследнего человека. Если он равен максимальному настраиваемому номеру, то номер добавляемого человека сбрасывается и становится 1, иначе ему передается номер предпоследнего

человека, увеличенный на 1. После этого размер очереди увеличивается на 1 и изменения отображаются на дисплее. Схема алгоритма показана на рисунке 16.

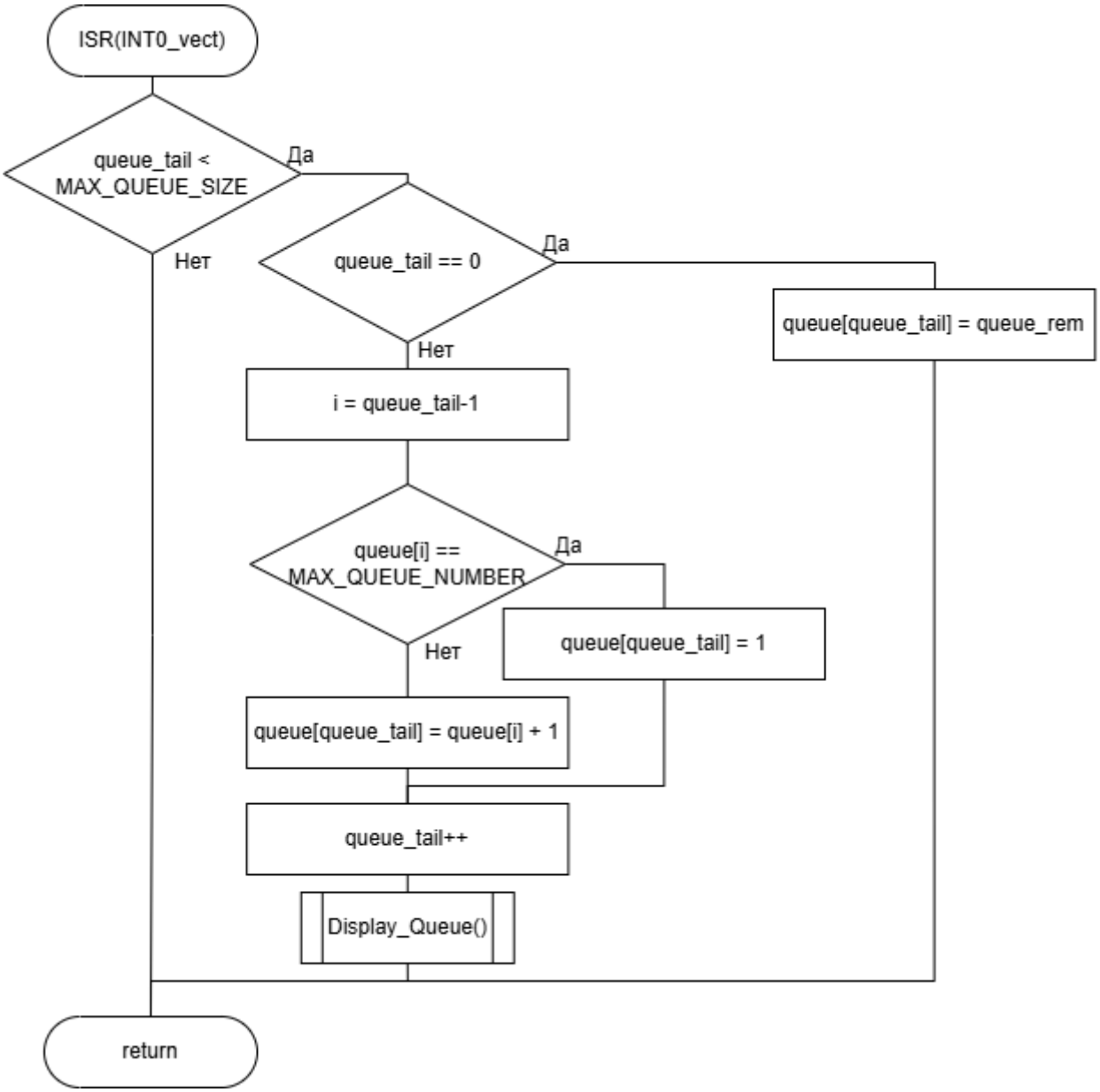

Рисунок 16 – Схема алгоритма функции ISR(INT0_vect)

Рассмотрим отображение данных об очереди на LCD-дисплее. Помимо вывода данных на экран происходит проверка полноценного вывода номера, так как в каждую из 2 строк максимально возможно вывести 16 символов. Для этого в переменную line_length передается кол-во символов в неизменяемой части строки. Затем происходит очищение дисплея и перенос курсора в начало. Если размер очереди больше нуля, то на дисплей

выводится строка "Next:", а на USART - "Queue:". Далее для каждого номера в очереди происходит проверка того, что при выводе его на дисплей сумма выводимых символов будет не больше 16. Если это так, то он выводится на дисплей. Вывод номера на USART происходит в любом случае, так же как и увеличение переменной line_length на длину номера. После этого происходит перенос курсора на следующую строку и вывод на дисплей строки "Last: " и последнего номера в очереди. Если размер очереди будет равен максимальному, то в конце строки на дисплей и USART выводится " (FULL)". Далее на USART происходит переход на следующую строку. Если размер очереди не больше 0, вывести на дисплей и USART строку "Queue is empty" и на USART сделать переход на следующую строку. Схема алгоритма показана на рисунке 17.

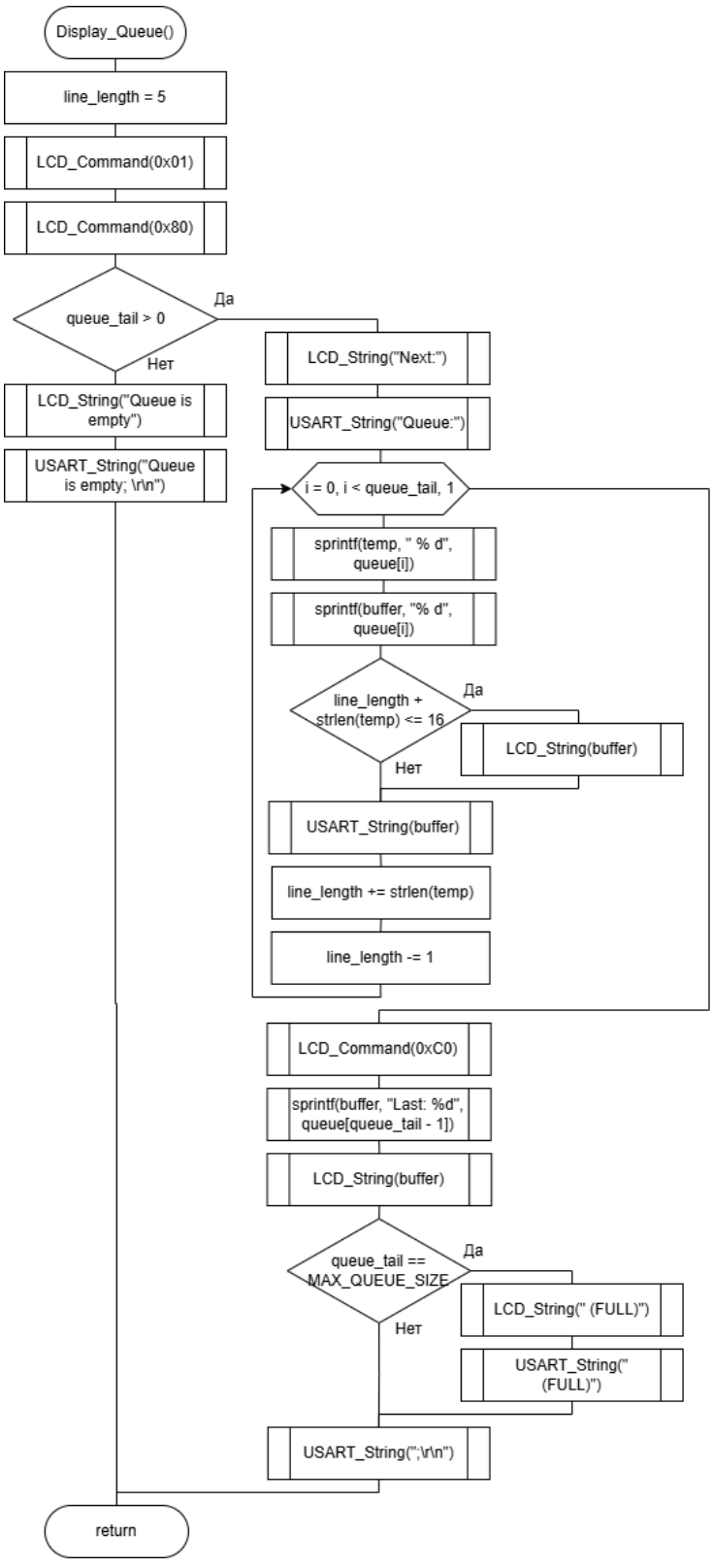

Рисунок 17 – Схема алгоритма функции Display_Queue

Рассмотрим прерывание при нажатии кнопки удаления человека из очереди, вызывающее функцию удаления человека из очереди. Схема алгоритма показана на рисунке 18.

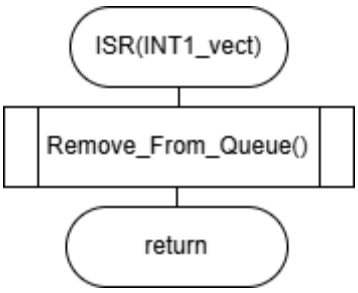

Рисунок 18 – Схема алгоритма функции ISR(INT1_vect)

Рассмотрим процесс удаления человека из очереди. Сначала сбрасывается значение timer_counter. Если размер очереди больше 0 и равен 1 передаем в queue_rem значение последнего номера в очереди, увеличенное на 1. Далее происходит сдвиг всех номеров в очереди влево, уменьшение размера очереди на 1 и вывод изменений на дисплей. Также происходит подача 1 на пин PC0, подавая сигнал на зуммер. После задержки подача сигнала прекращается. Схема алгоритма показана на рисунке 19.

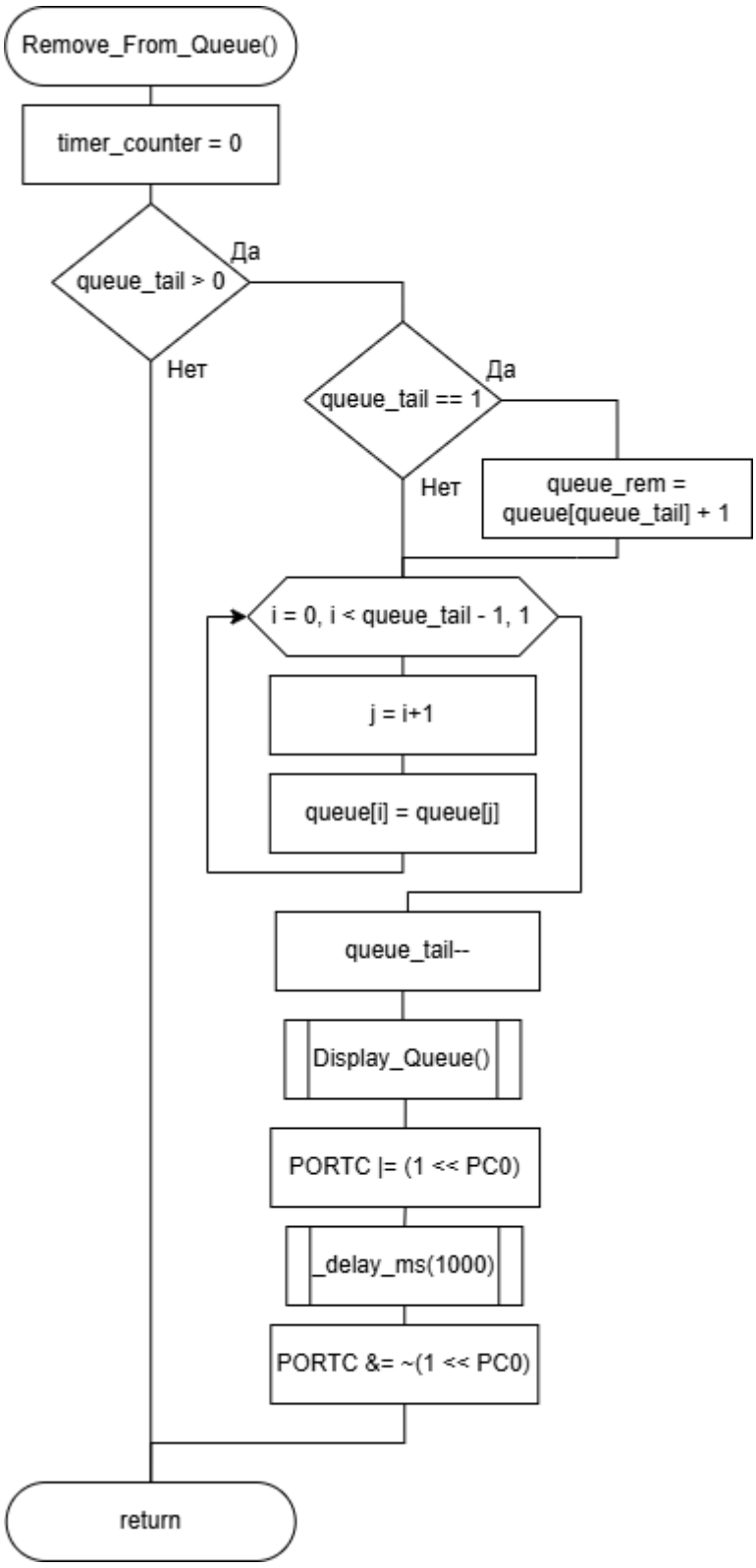

Рисунок 19 – Схема алгоритма функции Remove_From_Queue

Рассмотрим прерывание при совпадении значения таймера со хначением OCR2. Увеличивается на единицу значение seconds_counter, означающее прошедшее время в миллисекундах. Когда его значение будет

равно настраиваемому значению TIME вызовется функция удаления человека из очереди. Схема алгоритма показана на рисунке 20.

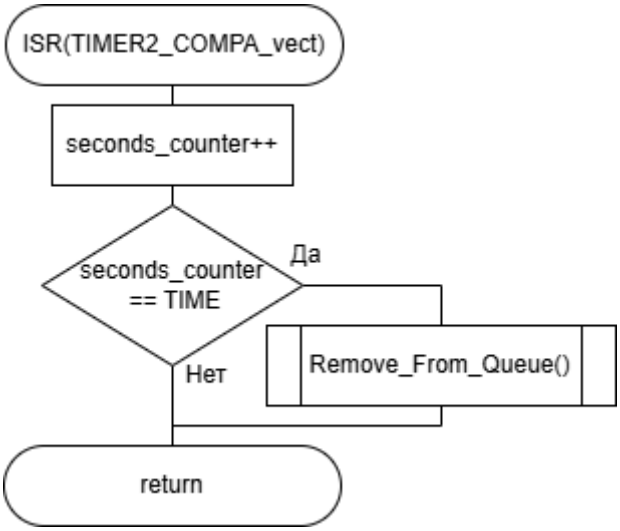

Рисунок 20 – Схема алгоритма функции ISR(TIMER2_COMPA_vect)

На основе составленных схем алгоритмов был написан код, который показан в Приложении А (Текст программы).

2 Технологическая часть

Для реализации работы контроллера электронной очереди была написана программа на языке Си, после загруженная в МК. Симуляция проводилась в программе Proteus 8.

2.1 Отладка и тестирование программы

Программа была отлажена с использованием приложения Proteus 8. Это приложение предназначено для выполнения различных видов моделирования аналоговых и цифровых устройств. В ней наглядно было видно, как ведут себя кнопки, LCD дисплей и USART, то есть как МК выполняет заложенный в него заранее написанный алгоритм.

При написании кода были использованы следующие библиотеки:

- avr/io.h – это библиотека ввода/вывода, которая объяснит компилятору какие порты ввода/вывода есть у микроконтроллера, как они обозначены и на что они способны;
- util/delay.h – это библиотека, позволяющая вызывать задержки (delay). При вызове _delay_ms в качестве параметра передается время в миллисекундах. В программе используется после чтения данных, перед их выводом, чтобы МК успел обработать полученную информацию;
- inttypes.h — это библиотека языка, которая предоставляет расширенные типы данных и функции для работы с целочисленными значениями.
- avr/interrupt.h – это библиотека, обеспечивающая доступ к функциям и макросам для управления прерываниями на микроконтроллере AVR.
- stdio.h – эта библиотека, используемая для обработки строк и форматирования данных. Она особенно полезна, когда нужно преобразовать данные в текстовый формат, например, для отображения на LCD или отправки через USART.
- string.h – это библиотека, которая предоставляет набор функций для работы с строками и массивами символов.

После компиляции создается файл с расширением “.hex”, объем которого равен 2,9 килобайта – столько занимает скомпилированная программа.

По итогу отладки и тестирования, результатом стала функционирующая модель контроллера электронной очереди, работающая в соответствии с ТЗ. Симуляция системы описана в разделе 2.3.

2.2 Симуляция работы системы

Для имитации реальных условий была использована программа Proteus 8. Схема модели контроллера электронной очереди показана на рисунке 21. Работа схемы, в данном случае нажатие кнопки, означающей добавление человека в очередь, 4 раза, нажатие кнопки принятия человека из очереди, и ожидания в течении 10 секунд показана на рисунке 22.

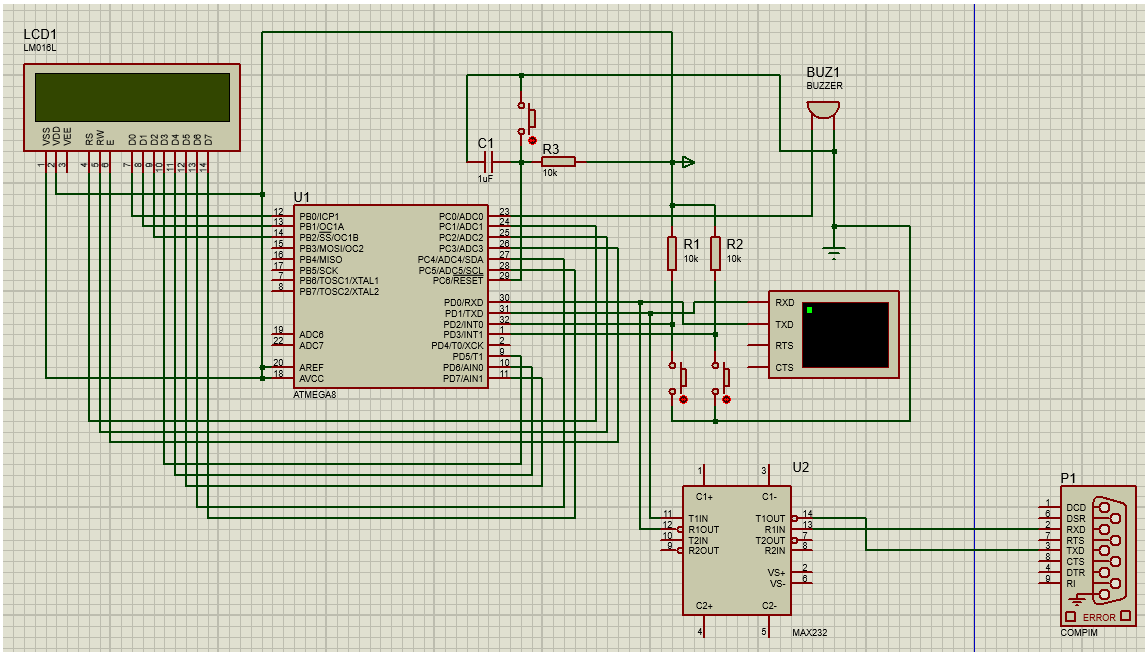

Рисунок 21 – Модель схемы

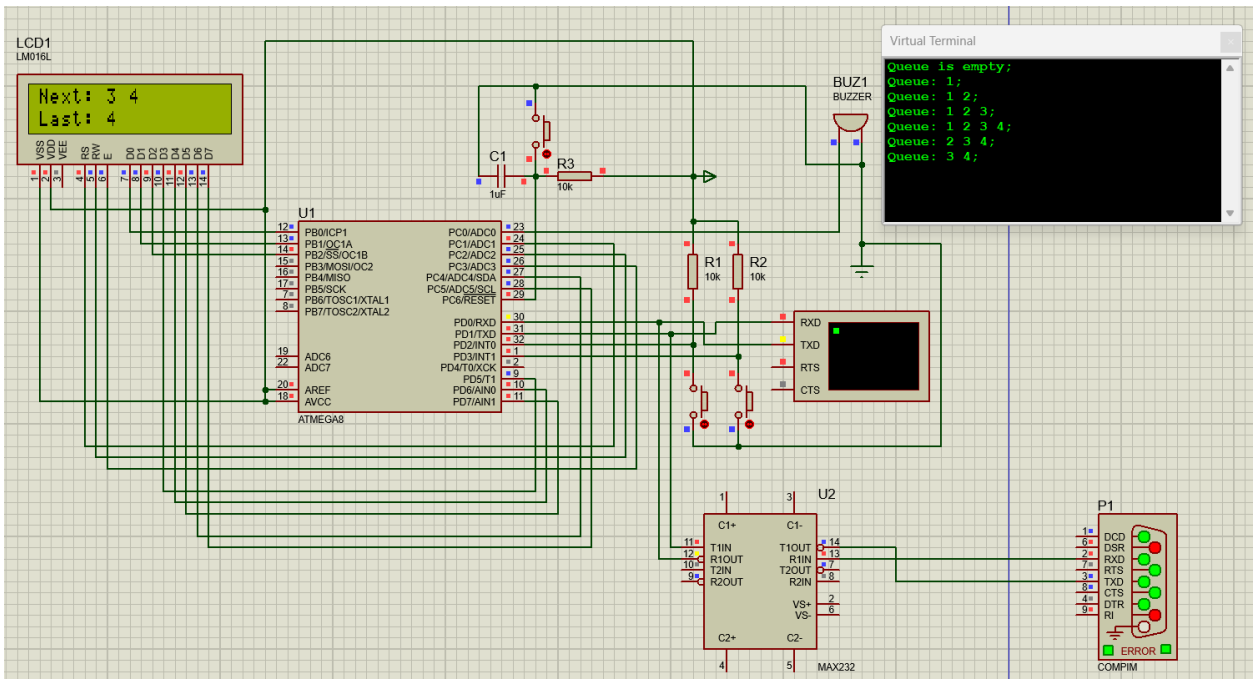

Рисунок 22 – Работа схемы

Модель в Proteus 8 не отличается от принципиальной схемы.

Для моделирования ввода данных с ПЭВМ используется инструмент системы – Virtual Terminal. Он позволяет эмулировать простейший терминал, который дает возможность передавать и получать данные по портам RxD и TxD через интерфейс UART.

При запуске системы активируется дисплей с информацией о том, что очередь пуста и последующем выводом информации о ближайших номерах в очереди и номере последнего вошедшего в очередь человека в очереди на экран, что показано на рисунке 23. После чего появляется возможность симулировать выход и вход в очередь посредством нажатия соответствующей кнопки или симулировать выход с помощью ожидания определенного количества времени без нажатия на кнопку выхода.

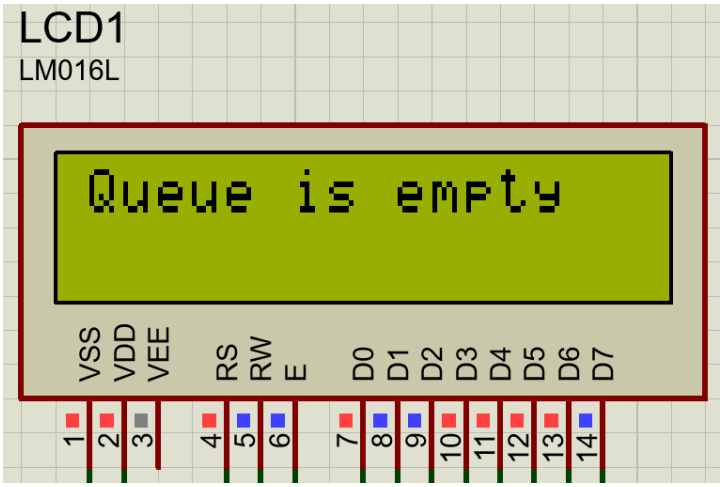

Рисунок 23 – Строка на LCD дисплее

Для проверки корректности работы драйвера MAX232 снимем показатели с осциллографа, они показаны на рисунке 24. На рисунке 24 видно передачу данных с МК в MAX232 – желтый отрезок и данные, вышедшие из MAX232 – зеленый отрезок.

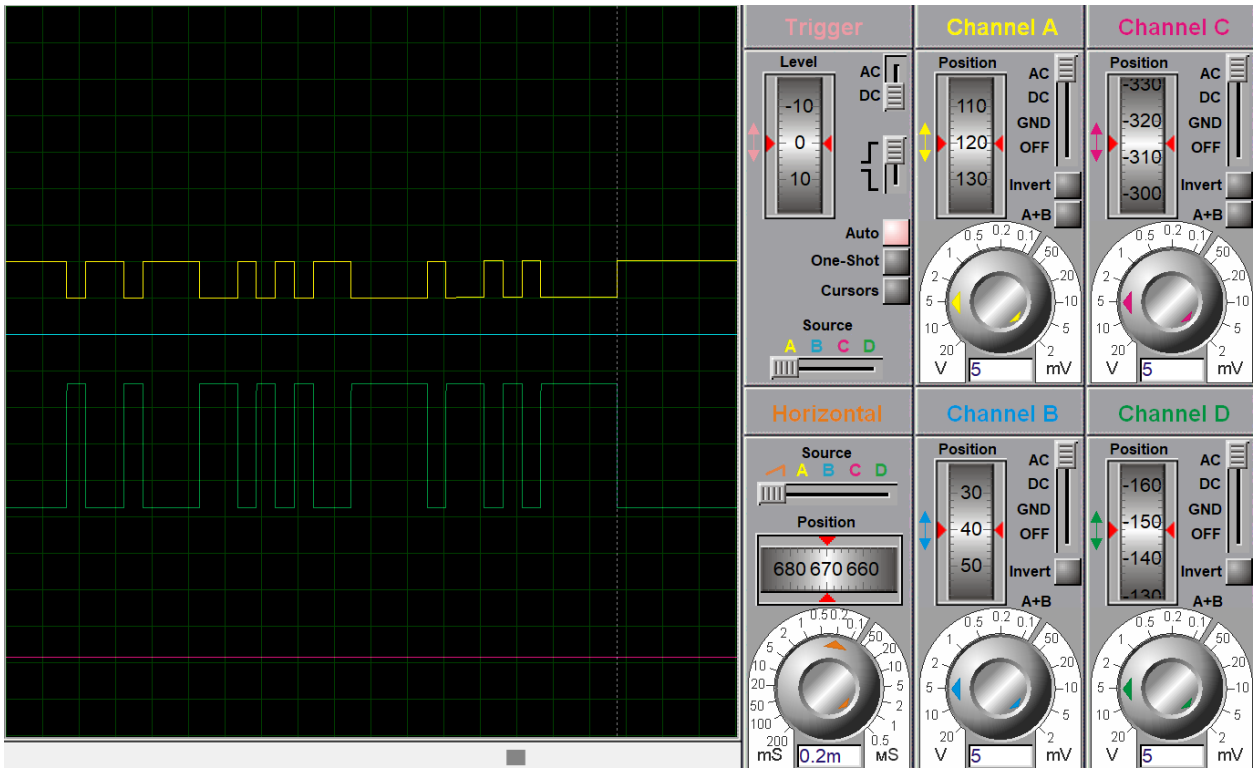

Рисунок 24 – Показатели осциллографа

RS232 основан на TTL-логике, то есть нулевому биту соответствует нулевой уровень напряжения, а единице уровень в +5 В. Стандарт RS232 использует более высокий уровень напряжения, до 15 В, и единице

соответствует -15 В, а нулю +15 В. Поэтому выходной – зеленый сигнал выглядит инвертированным относительно входного желтого.

2.3 Способы программирования МК

После написания и тестирования кода в программе, в котором все – это виртуальная модель, идет этап загрузки файла (с расширением hex – бинарный файл) в микроконтроллер. Это может выполнятся следующими способами [13]:

- внутрисхемное программирование (ISP – In-System Programming);
- параллельное высоковольтное программирование;
- через JTAG;
- через Bootloader;
- Pinboard II.

Для прошивки был выбран метод внутрисхемного программирования через интерфейс SPI, так как этот способ является простым, популярным и уже был опробован на практике. Процесс программирования микроконтроллера осуществляется с использованием программатора и специального разъема.

Прошивка проходит по интерфейсу SPI, для работы программатора нужно 4 контакта и питание:

- MISO – Master-Input/Slave-Output – линия передачи данных от программатора к микроконтроллеру;
- MOSI – Master-Output/Slave-Input – линия передачи данных от микроконтроллера к программатору;
- SCK – тактовые импульсы интерфейса SPI, для синхронизации передачи данных;
- RESET – линия сброса микроконтроллера, которая переводит его в режим программирования;
- GND – земля;
- VCC – питание.

Для работы через SPI одно устройство должно быть назначено ведущим, а остальные – ведомыми. Ведущее устройство управляет выбором ведомого и инициирует передачу данных.

Инструкции, передаваемые по интерфейсу, могут иметь разное назначение. Примеры таких инструкций представлены в таблице 18.

Таблица 18 – Коллекция инструкций при программировании по интерфейсу SPI.

Инструкция	Формат инструкции				Операция
	Byte1	Byte2	Byte3	Byte4	
Programming Enable	1010 1100	0101 0011	xxxx xxxx	xxxx xxxx	Разрешение программирования
Chip Erase	1010 1100	100x xxxx	xxxx xxxx	xxxx xxxx	Стирание памяти EEPROM и Flash
Read Program Memory	0010 H000	0000 aaaa	bbbb bbbb	oooo oooo	Чтение данных из памяти программ
Load Program Memory Page	0100 H000	0000 xxxx	xxxb bbbb	iiii iiii	Запись данных в страницу памяти программ
Read EEPROM Memory	1010 0000	00xx xxxa	bbbb bbbb	oooo oooo	Чтение данных из памяти EEPROM
Write EEPROM Memory	1100 0000	00xx xxxa	bbbb bbbb	iiii iiii	Запись данных в память EEPROM
Write Fuse Bits	1010 1100	1010 0000	xxxx xxxx	iiii iiii	Для программирования “Fuse” битов. Запись
Write Fuse High Bits	1010 1100	1010 1000	xxxx xxxx	iiii iiii	Для программирования “Fuse” битов. Запись

Продолжение таблицы 18

Read Fuse Bits	0101 0000	0000 0000	xxxx xxxx	oooo oooo	Для программирования “Fuse” битов. Чтение
Read Fuse High Bits	0101 1000	0000 1000	xxxx xxxx	oooo oooo	Для программирования “Fuse” битов. Чтение

Буквы в байтах означают следующее: a - старшие биты адреса; b - младшие биты адреса; H – младший байт, если = 0, и старший байт, если = 1; o - данные на выход; i - данные на вход; x - не имеет значения.

Программатор передает инструкции микроконтроллеру в определенной последовательности в зависимости от выполняемого действия и получает результат. В случае с разработанной системой процесс начинается с выполнения команды стирания памяти. Затем выполняется запись данных в память микроконтроллера, включая как старшие, так и младшие байты. В завершение выполняются команды чтения данных для проверки правильности операций. Настройка Fuse-битов также включает последовательное выполнение инструкций записи,

Обмен данными через интерфейс SPI осуществляется в полнодуплексном режиме, при котором за каждый такт передается по одному биту в обоих направлениях. На возрастающем фронте сигнала SCK ведомое устройство считывает очередной бит с линии MOSI, а на спадающем фронте передает следующий бит на линию MISO.

В микроконтроллер загружается бинарный файл с расширением “.hex”, содержащий скомпилированную программу.

Заключение

В результате выполнения курсовой работы был создан проект – контроллер электронной очереди. Система работает на основе МК серии AVR – ATmega8. Устройство разработано в соответствии с ТЗ.

В процессе работы над курсовой работой была разработана схема электрическая функциональная и принципиальная, перечень элементов, а также расчетно-пояснительная записка. Исходный код программы, написанный на языке С, отлажен и протестирован при помощи симулятора Proteus 8.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Микроконтроллеры 8051, PIC, AVR и ARM: отличия и особенности [Электронный ресурс]. – URL: http://digitrode.ru/computing-devices/mcu_cpu/1253-mikrokontrollery-8051-pic-avr-i-arm-otlichiya-i-osobennosti.html (дата обращения: 20.09.2024)

2. ATmega8 Datasheet [Электронный ресурс]. – URL: https://www.alldatasheet.com/datasheet-pdf/view/80247/ATMEL/ATMEGA8.html (дата обращения: 20.09.2024)

4. Хартов В.Я. Микроконтроллеры AVR. Практикум для начинающих: учебное пособие. – 2-е изд., испр. и доп. – М.: Издательство: МГТУ им. Н.Э. Баумана, 2012. – 280с.

5. MAX232 Dual EIA-232 Drivers and Receivers datasheet [Электронный ресурс]. – URL: https://www.ti.com/lit/ds/symlink/max232.pdf (дата обращения: 20.09.2024)

6. AVR. USART. Связь МК с ПК. [Электронный ресурс]. – URL: https://narodstream.ru/avr-urok-14-usart-svyaz-mk-s-pk-chast-1/ (дата обращения: 20.09.2024)

7. Использование внешних прерываний в AVR. [Электронный ресурс]. – URL: https://chipenable.ru/index.php/programming-avr/105-uchebnyy-kurs-ispolzovanie-vneshnih-preryvaniy-v-avr.html (дата обращения: 20.09.2024)

8. Timer/Counter for AVR. [Электронный ресурс]. – URL: https://cxem.net/mc/mc388.php (дата обращения: 20.09.2024)

9. LM016L Datasheet [Электронный ресурс]. – URL: https://digsys.upc.edu/csd/chips/classic/LM016L_Hitachi.pdf (дата обращения: 20.09.2024)

10. МСКРХ-G1203UB-K4065 Datasheet [Электронный ресурс]. – URL: https://www.tedss.com/DataSheets/2099/2099003066.pdf?srsltid=AfmBOoqWcjdkdLzxbdwmagNz1OHVjzPZhkv0kLaYYCcj3q9ca6w_NhWC (дата обращения: 20.09.2024)

11. ГОСТ 2.743-91 ЕСКД ОБОЗНАЧЕНИЯ БУКВЕННО-ЦИФРОВЫЕ В ЭЛЕКТРИЧЕСКИХ СХЕМАХ [Электронный ресурс]. – URL: https://docs.cntd.ru/document/1200001985 (дата обращения: 20.09.2024)

12. ГОСТ 2.721-74 ЕСКД ОБОЗНАЧЕНИЯ УСЛОВНЫЕ ГРАФИЧЕСКИЕ В СХЕМАХ [Электронный ресурс]. – URL: https://docs.cntd.ru/document/1200007058 (дата обращения: 20.09.2024)

13. AVR910: Внутрисистемное программирование [Электронный ресурс]. – URL: https://sin-bad.narod.ru/isp.htm (дата обращения: 20.09.2024)

Приложение А

Текст программы

Объем исполняемого кода – 227 строк.

```
#include <inttypes.h>
#include <avr/io.h>
#include <avr/interrupt.h>
#include <util/delay.h>
#include <stdio.h>
#include <string.h>

#define F_CPU 2000000UL

#define LCD_Data_Reg1 DDRB
#define LCD_Data_Reg2 DDRD
#define LCD_Data_Reg3 DDRC

#define LCD_Data_Port1 PORTB
#define LCD_Data_Port2 PORTD
#define LCD_Data_Port3 PORTC

#define LCD_Command_Reg DDRC
#define LCD_Command_Port PORTC

#define RS PC1
#define RW PC2
#define EN PC3

#define TIME 100 // Предел таймера в 0.1с

#define MAX_QUEUE_SIZE 10 // Максимальный размер очереди
#define MAX_QUEUE_NUMBER 99 // Максимальный номер в очереди

volatile uint16_t timer_counter = 0;  // Счетчик таймера

volatile uint8_t queue[MAX_QUEUE_SIZE];
```

```
volatile uint8_t queue_tail = 0;
volatile uint8_t queue_rem = 1;

// Функции USART
void USART_Init(unsigned int baud) {
   unsigned int ubrr = F_CPU/16/baud-1;
   UBRRH = (unsigned char)(ubrr >> 8);
   UBRRL = (unsigned char)ubrr;
   UCSRB = (1 << RXEN) | (1 << TXEN);
   UCSRC = (1 << URSEL) | (3 << UCSZ0);
}

void USART_String(const char* str) {
   while (*str) {
      while (!(UCSRA & (1 << UDRE)));
      UDR = *str++;
   }
}

// Функция отправки команды на LCD
void LCD_Command(unsigned char cmnd) {
   LCD_Data_Port1 = cmnd & 0xC7;
   LCD_Data_Port2 = (cmnd & 0x38) << 2;
   LCD_Data_Port3 = (cmnd & 0xC0) >> 2;

   LCD_Command_Port &= ~(1 << RS);
   LCD_Command_Port &= ~(1 << RW);
   LCD_Command_Port |= (1 << EN);

   _delay_us(30);
   LCD_Command_Port &= ~(1 << EN);
   _delay_ms(30);
}

/////// Инициализация LCD
```

```
void LCD_Init(void) {
   LCD_Command_Reg |= (1 << PC1) | (1 << PC2) | (1 << PC3);
   LCD_Data_Reg1 |= (1 << PB0) | (1 << PB1) | (1 << PB2);
   LCD_Data_Reg2 |= (1 << PD5) | (1 << PD6) | (1 << PD7);
   LCD_Data_Reg3 |= (1 << PC4) | (1 << PC5);

   _delay_ms(30);
   LCD_Command(0x38); // 8-битный режим работы, 2 строки
   LCD_Command(0x0C); // Включает дисплей, выключает курсор
   LCD_Command(0x06); // Автоматическое перемещение курсора
вправо
   LCD_Command(0x01); // Очистка экрана
   LCD_Command(0x80); // Установка курсора на начальную позицию
}

// Функция вывода строки на LCD
void LCD_String(char *str) {
   uint8_t i;
   for (i = 0; str[i] != 0; i++) {
      LCD_Data_Port1 = str[i] & 0xC7;
      LCD_Data_Port2 = (str[i] & 0x38) << 2;
      LCD_Data_Port3 = (str[i] & 0xC0) >> 2;

      LCD_Command_Port |= (1 << RS);
      LCD_Command_Port &= ~(1 << RW);
      LCD_Command_Port |= (1 << EN);

      _delay_us(30);
      LCD_Command_Port &= ~(1 << EN);
      _delay_ms(30);
   }
}

// Функция отображения очереди на LCD и UART
void Display_Queue(void) {
```

```
char buffer[50];
uint8_t line_length = 5;
uint8_t i;
LCD_Command(0x01); // Очистка экрана
LCD_Command(0x80); // Установка курсора на начальную позицию

if (queue_tail > 0) {
   LCD_String("Next:");
   USART_String("Queue:");

   for (i = 0; i < queue_tail; i++) {
   char temp[4];
   sprintf(temp, " % d", queue[i]);
   sprintf(buffer, "% d", queue[i]);

   if (line_length + strlen(temp) <= 16)
      LCD_String(buffer);

   USART_String(buffer);
   line_length += strlen(temp);
   line_length -= 1;
   }

   LCD_Command(0xC0);
   sprintf(buffer, "Last: %d", queue[queue_tail - 1]);
   LCD_String(buffer);

   if (queue_tail == MAX_QUEUE_SIZE) {
   LCD_String(" (FULL)");
   USART_String(" (FULL)");
   }
   USART_String(";\r\n");

} else {
   LCD_String("Queue is empty");
```

```
      USART_String("Queue is empty; \r\n");
   }
}

/////// Инициализация пинов для кнопок
void Button_Init() {
   DDRC |= (1 << PC0);
   DDRD &= ~((1 << PD2) | (1 << PD3));
   PORTD |= (1 << PD2) | (1 << PD3);
}

/////// Инициализация внешних прерываний
void External_Interrupts_Init() {
   MCUCR |= (1 << ISC01) | (1 << ISC11);
   GICR |= (1 << INT0) | (1 << INT1);
}

/////// Инициализация таймера, Режим CTC, т.к. переполнение
получается за 0,13с
void Timer2_Init() {
   TCCR2 = 0;                          // Очистить регистр
конфигурации таймера
   TCCR2 |= (1 << WGM21) | (1 << CS22) | (1 << CS21) | (1 <<
CS20); // Режим CTC, делитель 1024
   OCR2 = 195;                         // Значение для 100 мс
(F_CPU / 1024 / 10ms - 1)
   TIMSK |= (1 << OCIE2);              // Включить прерывание по
совпадению OCR0A
}

// Удаление человека из очереди
void Remove_From_Queue() {
   timer_counter = 0;
   if (queue_tail > 0) {
      uint8_t i;
```

```
      if (queue_tail == 1)
      queue_rem = queue[queue_tail] + 1;
      for (i = 0; i < queue_tail - 1; i++) {
      uint8_t j = i+1;
      queue[i] = queue[j];
      }
      queue_tail--;
      Display_Queue();
      PORTC |= (1 << PC0);
      _delay_ms(1000);
      PORTC &= ~(1 << PC0);
   }
}

// Настройка кнопки для добавления человека в очередь
ISR(INT0_vect) {
   if (queue_tail < MAX_QUEUE_SIZE) {
      if (queue_tail == 0)
      queue[queue_tail] = queue_rem;
      else {
      uint8_t i = queue_tail-1;
      if (queue[i] == MAX_QUEUE_NUMBER)
         queue[queue_tail] = 1;
      else
         queue[queue_tail] = queue[i] + 1;
      }
      queue_tail++;
      Display_Queue();
   }
}

// Настройка кнопки для удаления человека из очереди
ISR(INT1_vect) {
   Remove_From_Queue();
}
```

```
// Настройка таймера для удаления человека из очереди
ISR(TIMER2_COMP_vect) {
   timer_counter++;
   if (timer_counter == TIME)
      Remove_From_Queue();
}

int main(void) {
   External_Interrupts_Init();
   LCD_Init();
   USART_Init(9600);

   Timer2_Init();
   sei();

   LCD_String("Queue is empty");
   USART_String("Queue is empty;\r\n");

   Button_Init();

   while (1) {
      if (queue_tail <= 0)
      timer_counter = 0;
   }
}
```

Приложение Б

Графическая часть

На 2 листах

Электрическая схема функциональная

Электрическая схема принципиальная

Приложение В

Перечень элементов

На 3 листах