Реферат

Расчетно-пояснительная записка страниц 32, рисунков 17, таблиц 5, источников 9, приложений 4.

Ключевые слова: МИКРОКОНТРОЛЛЕР, ATMEGA32, ЭЛЕКТРОННАЯ ОЧЕРЕДЬ, СЕТЬ, RS-485, АППАРАТАНАЯ СИСТЕМА.

Объектом разработки данной курсовой работы является проект аппаратная система «Электронная очередь», предназначенная для организации электронной очереди, в которой участвуют пользователи, администраторы и устройства, отображающие состояние очереди.

Целью разработки являлось проектирование функционально законченной системы, построенной на микроконтроллерах, удовлетворяющей заданным техническим заданием требованиям, и разработка необходимой конструкторской и технической документации на проектируемую систему. При проектировании были решены следующие задачи: анализ объекта разработки на функциональном уровне, разработка функциональной схемы системы, выбор элементной базы для реализации объекта, расчет электрических параметров.

В ходе работы разработаны алгоритмы функционирования системы, функциональная и электрическая принципиальная схемы, написаны коды функционирования, соответствующие разработанным алгоритмам.

Для написания кодов был использован язык СИ AVR, работа осуществлялась в среде WinAVR. Для проверки работоспособности разработанная схема была промоделирована в среде Proteus ISIS 8.

Результатом проектирования является комплект конструкторской документации для изготовления аппаратной системы «Электронная очередь» и модель этой системы.

ОПРЕДЕЛЕНИЯ, ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ

АСЭО – аппаратная система «Электронная очередь»,

МК – микроконтроллер.

Содержание

Введение

Настоящий курсовой проект распространяется на разработку аппаратной системы «Электронная очередь» (АСЭО), предназначенной для организации электронной очереди, в которой участвуют пользователи, администраторы и устройства, отображающие состояние очереди.

Задача организации электронной очереди является актуальной во многих областях, связанных с обслуживанием клиентов. Хорошо организованная электронная очередь позволяет уменьшить загруженность потока клиентов и время ожидания в очереди.

Разработка устройства состоит из трех основных частей:

- аппаратной части – коммутации микроконтроллеров (МК) и принципов взаимодействия МК и средств отображения информации – дисплеев, и средств ввода – кнопок;
- программной части — написание программ для МК;
- тестирование системы посредством симуляции в среде Proteus ISIS 8.

Во время выполнения курсовой работы необходимо построить функциональную и принципиальную схемы, разработать алгоритмы работы системы, на основе которых программируется МК, а также рассчитать потребляемую мощность.

1 Анализ требований

Система должна состоять как минимум из трех модулей:

1) модуль приема сигналов от пользователей об их желании встать в очередь;
2) модуль приема сигналов от администратора о готовности принять первого в очереди клиента;
3) модуль отображения состояния очереди.

Полученное задание можно решить при помощи системы, состоящей из следующих крупных элементов:

1) МК, связанный с дисплеем, отображающим номер нового клиента в очереди, и кнопкой, принимающей сигналы от пользователей;
2) МК, связанный с дисплеем, отображающим номер первого в очереди клиента, и кнопкой, принимающей сигналы от администратора;
3) МК, связанный с дисплеем, отображающем состояние очереди.

МК должны быть связаны между собой, чтобы обмениваться информаций. По заданию связь МК должна быть организована с помощью интерфейса по стандарту RS-485.

Для определенности положим, что в очереди может быть не более 99 клиентов: от 1 до 99. Соответственно, для вывода номера клиента, потребуется 2-разрядные дисплеи.

2 Проектирование функциональной схемы системы

По заданию необходимо использовать МК семейства AVR. Для реализации проектируемой системы был выбран МК ATMega32, имеющий память SRAM 2Кб и Flash 32Кб.

МК, отражающий текущее состояние очереди, должен выводить очередь на дисплей. Для определенности будем использовать два 2-разрядных дисплея, чтобы выводить номера первого и второго пользователей в очереди. Выводить информацию на дисплеи будем по параллельной шине. Дисплеи выберем 7-сегментные, соответственно, ширина шины будет равна 7. Также, для каждого дисплея необходимы 2 управляющих сигнала, определяющих на какой разряд выводить информацию с шины. Для первого дисплея забронируем 7 пинов порта A, для второго 7 пинов порта B, управляющие сигналы будем подавать с 4 пинов порта C.

МК, принимающие сигналы от пользователей и администраторов, должны выводить номер нового пользователя или номер следующего пользователя и принимать сигналы от пользователей или от администратора, для чего необходимы кнопки. Для вывода номера пользователя будем использовать аналогичные дисплеям МК, отражающего очередь, дисплеи. Забронируем под них 7 пинов порта A, а под управляющие сигналы забронируем 2 пина порта B. Принимать сигналы от кнопок будем с помощью 3-его пина порта B.

Так как все три МК должны посылать друг другу информацию и принимать информацию друг от друга, необходимо задействовать протокол UART на всех МК, что обязывает забронировать 0 и 1 пины порта D под использование UART: 0 пин порта D – RX (прием данных), 1 пин порта D – TX (отправка данных).

При подключении RX и TX одного МК к TX и RX двух других МК получается ситуация, при которой RX второго МК подключен к RX третьего МК, с выходами TX аналогично. Такая ситуация создает ошибку работы UART, что обязывает к организации более сложной системы приема и передачи данных. Необходимо использовать приемо-передатчик MAX485, который способен пе-

реключать в режим передачи или приема в зависимости от управляющих сигналов TE (разрешение передачи) и RE (разрешение приема) и преобразует ТТЛ сигналы в сигналы по стандарту RS-485. Таким образом на участке с сигналами RS-485 будет присутствовать информация, отправленная одним из МК и будет приниматься двумя другими МК. Для организации управляющих сигналов TE/RE задействуем 3 пин порта D на всех МК.

Разработанная функциональная схема представлена в приложении А.

3 Проектирование принципиальной схемы устройства

3.1 Выбор и реализация разъемов

Выбранная реализация принципиальной схемы устройства подразумевает наличие 5 разъемов:

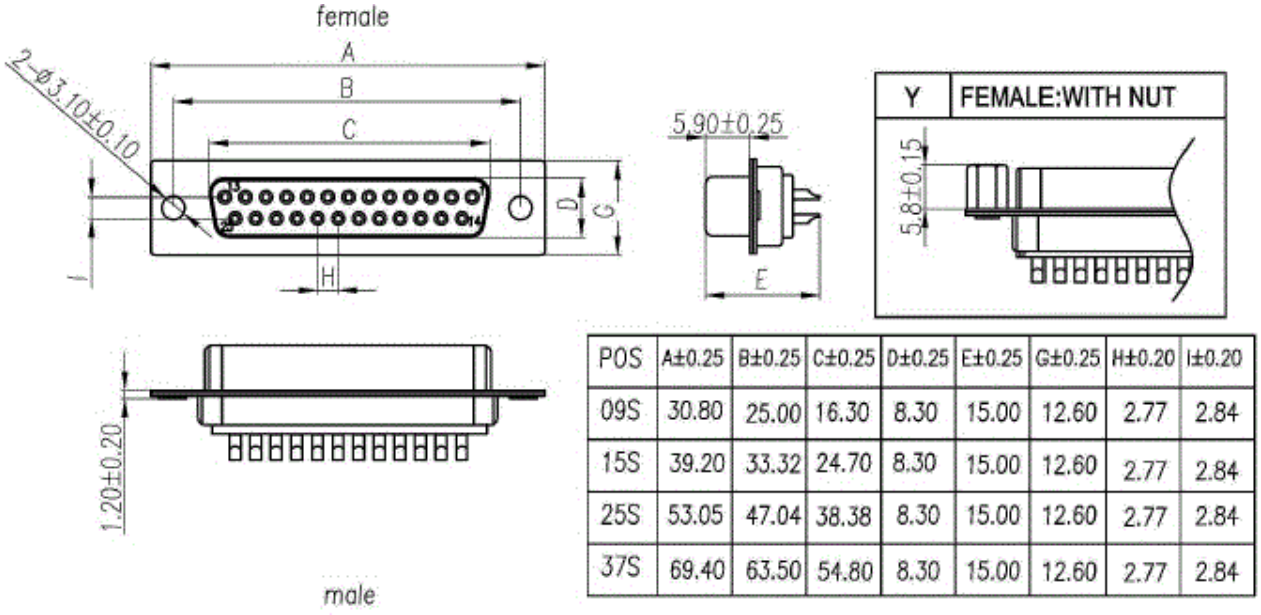
- разъем (вилка XP1 и розетка XS1), отвечающий за соединение с источником питания;
- разъем (вилка XP2 и розетка XS2), отвечающий за соединение модуля A1, отображающего состояние очереди, и модуля A2, принимающего сигналы от пользователей;
- разъем (вилка XP3 и розетка XS1), отвечающий за соединение модуля A2, принимающего сигналы от пользователей, и модуля A1, отображающего состояние очереди;
- разъем (вилка XP4 и розетка XS2), отвечающий за соединение модуля A2, принимающего сигналы от пользователей, и модуля A3, принимающего сигналы от администратора;
- разъем (вилка XP5 и розетка XS1), отвечающий за соединение модуля A3, принимающего сигналы от администратора, и модуля A2, принимающего сигналы от пользователей.

Соединение с источником питания предполагает 2 контакта: питание системы +5 В и 0 В GND. Для реализации данного разъема выберем разъем типа DE-9 семейства D-sub.

Соединения между модулями, предполагает 4 контакта: прямой сигнал A и инвертированный сигнал B по стандарту RS-485, питание системы +5 В и 0 В GND. Для реализации данного разъема также выберем разъемы типа DE-9 семейства D-sub.

Для всех разъемов выбраны разъемы из семейства D-sub. Разъемы семейства D-sub содержат два или более параллельных рядов штырьковых контактов или контактных гнёзд, обычно окружённых металлическим экраном в форме латинской буквы «D», который обеспечивает механическое крепление соедине-

ния, предохраняет от неправильной ориентации при подключении и экранирует от электромагнитных помех. Для крепления на плату на внутренней части разъемов имеются цилиндрические контакты, которые непосредственно впаиваются в плату. Фиксация разъема на плате осуществляется с помощью специальных фиксаторов: гайки с винтом [1]. Характеристики разъема представлены на рисунке 1.



POS	A±0.25	B±0.25	C±0.25	D±0.25	E±0.25	G±0.25	H±0.20	I±0.20
09S	30.80	25.00	16.30	8.30	15.00	12.60	2.77	2.84
15S	39.20	33.32	24.70	8.30	15.00	12.60	2.77	2.84
25S	53.05	47.04	38.38	8.30	15.00	12.60	2.77	2.84
37S	69.40	63.50	54.80	8.30	15.00	12.60	2.77	2.84

Рисунок 1 – Характеристики разъемов семейства D-sub

Соответствия между контактами разъемов и контактов схемы разрабатываемого устройства для разъема XP1-XS1 приведены в таблице 1, для разъемов XP2-XS2, XP3-XS1, XP4-XS2 и XP5-XS1 в таблице 2.

Таблица 1 – Подключение контактов к разъему XP1-XS1

Номер контакта в разъеме	**Назначение контакта**	**Обозначение**
1	Питание системы +5В	VCC
2..9	0 В	GND

Таблица 2 – Подключение контактов к разъемам XP2-XS2, XP3-XS1, XP4-XS2 и XP5-XS1

Номер контакта в разъеме	Назначение контакта	Обозначение
1	Инвертированный сигнал по стандарту RS-485	B
2	Прямой сигнал по стандарту RS-485	A
3	Питание системы +5В	VCC
4..9	0 В	GND

3.2 Выбор конденсаторов для питания микросхем

Для устранения ВЧ-помех при использовании интегральных микросхем (ИМС), которые могут возникать при наличии свободных выводов ИМС, применяют шунтирующие конденсаторы небольшой емкости, подключенные к входу, питающему ИМС. Для шунтирования ИМС был выбран керамический конденсатор емкостью 100 нФ GRM21B5C1H223JA01L.

3.3 Реализация высокочастотного генератора импульсов

Для реализации высокочастотного генератора импульсов использован кварцевый резонатор 8MHz HC-49US. Схема его подключения показана на рисунке 2.

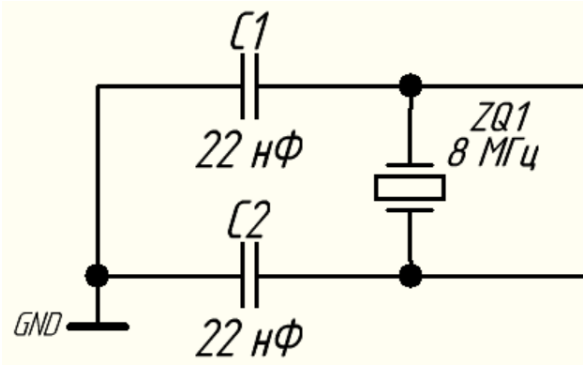


Рисунок 2 – Схема подключения кварцевого резонатора

Используемые для подавления помех в этой схеме конденсаторы – GRM188R71C104KA01D.

3.4 Реализация дисплеев

В устройствах использованы дисплеи BL-D56F-22S-X – 2-разрядные 7-сегментные дисплеи с управляющими сигналами для выбора разряда, на который будет транслироваться информация с информационнах входов. Схема подключения светодиодов в дисплее представлена на рисунке 3.

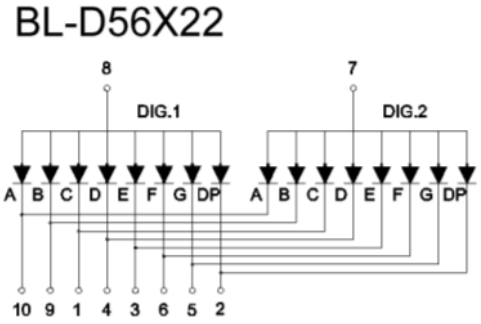


Рисунок 3 – Схема подключения светодиодов в дисплее

Для вывода информации используются информационные входы A, B, C, D, E, F, G, DP (пины 10, 9, 1, 4, 3, 6, 5 и 2 соответственно). Для выбора разряда используются входы D1 и D2 [2]. На информационные входы подадим сигналы от МК, кроме входа DP, который заземлим, так как использовать его не будем. На входы D1 и D2 подадим управляющие сигналы от МК.

Дисплей выглядит так, как показано на рисунке 4. В таблице 2 приведено соответствие отображаемой цифры сигналам на информационных входах. В таблице 3 приведено соответствие комбинаций управляющих сигналов состояниям дисплея.

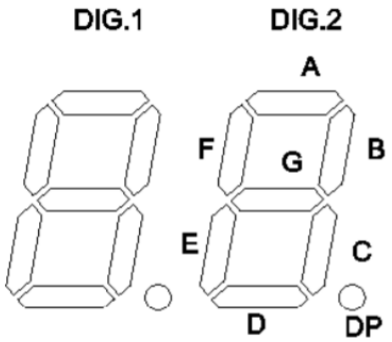


Рисунок 4 – Дисплей

Таблица 2 – Соответствие цифр сигналам на информационных входах дисплея

Цифра	Сигналы на входах (A,B,C,D,E,F,G соответственно)
0	00111111
1	00000110
2	01011011
3	01001111
4	01100110
5	01101101
6	01111101
7	00000111
8	01111111
9	01101111

Таблица 3 – Соответствие управляющих сигналов состояниям дисплея

D1	D2	Состояние дисплея
0	0	Горят оба разряда
0	1	Горит первый разряд
1	0	Горит второй разряд
1	1	Оба разряда не горят

3.5 Организация ввода информации

Для получения сигналов от пользователей и администратора необходимы кнопки. Будем использовать кнопки 1-1437565-6 (FSM1LP). На один контакт кнопки подадим 0В, а другой контакт свяжем с МК.

3.6 Подключение микроконтроллеров

Схема МК ATMega32 представлена на рисунке 5 [3]. Подключение МК выполним в соответствие с функциональной схемой системы. Также подадим на входы VCC и AVCC (пины 10 и 30 соответственно) 5В, а на входы GND (пины 11 и 31) 0В. Ко входам XTAL1 и XTAL2 (пины 13 и 12) подадим сигналы от высокочастотного генератора сигналов. Ко входу $\overline{R}$ (сброс) подключим сигнал RESET из разъема XS2, предварительно установив подтягивающий резистор R1 на 10 кОм и параллельно к нему конденсатор C5 на 100 нФ. На свободные входы подадим 0В.

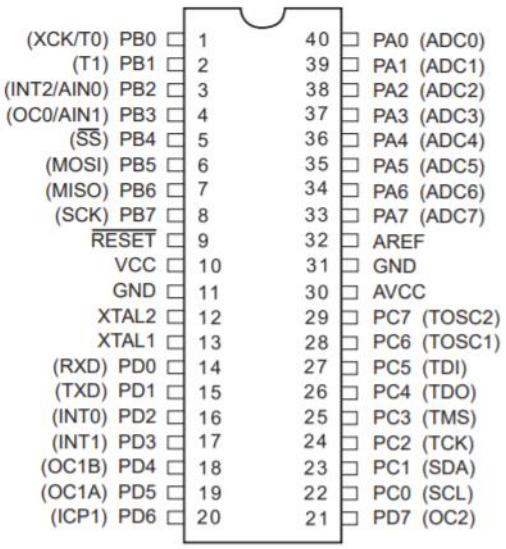


Рисунок 5 – Микроконтроллер ATMega32

3.7 Организация обмена данными между микроконтроллерами

Как уже было решено на этапе проектирования функциональной схемы системы, для обмена данными будут использоваться ИМС MAX485. Схема данной ИМС представлена на рисунке 6.

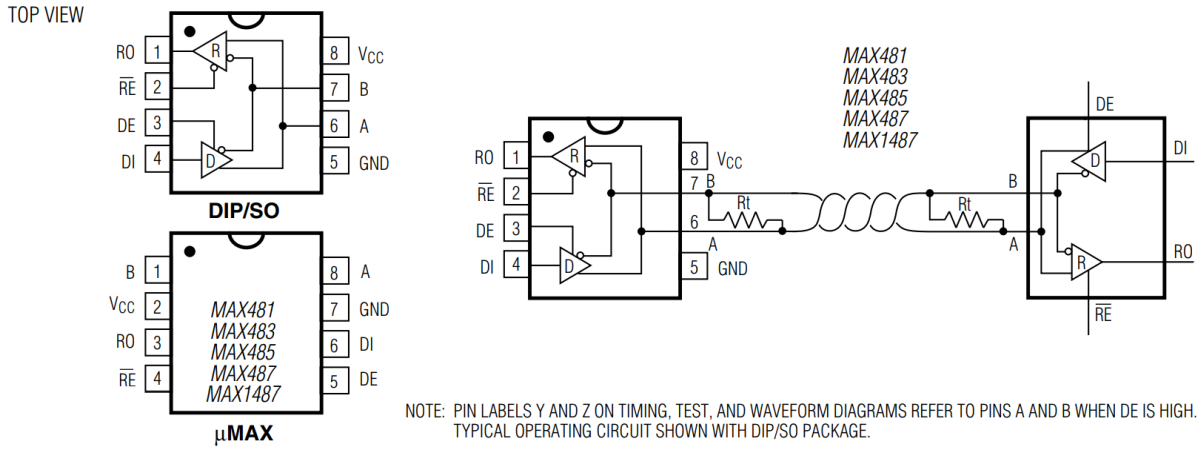


Рисунок 6 – ИМС MAX485

Данная ИМС может работать в двух режимах: приемник и передатчик. Выбор режима осуществляется с помощью входов $\overline{RE}$ (receiver enable) и DE (driver enable). Входы A и B – входы/выходы сигнала по стандарту RS-485. Выход RO (receiver output) – выход приемника – преобразованный ТТЛ-совместимый сигнал. Вход DI (driver input) – вход передатчика. VCC – питание, GND – 0 В [4].

К входам $\overline{RE}$ и DE подключим PD2 МК, к RO подключим PD0 (RX UART), к DI подключим PD1 (TX UART). Входы/выходы A и B подключим к разъемам XS1.

На выходах разъемов XP1, XP2 и XP3 имеем сигналы по стандарту RS-485. Соединим эти разъемы. Во избежание потери данных в случае, если МК-приемник еще не готов принимать данные, а МК-передатчик уже отправил их, подключим внутри цепи резистор R1 на 120 Ом.

Такая организация обмена данными между МК позволяет расширять систему добавляя в нее новые модули.

3.8 Реализация принципиальной схемы и перечня элементов

В соответствие с выявленными требованиями была реализована принципиальная схема системы, представленная в приложении Б. К принципиальной схеме был составлен перечень используемых элементов, представленный в приложении В.

4 Расчет потребляемой мощности

Все ИМС питаются от напряжения VCC +5 В. Для оценки потребляемой мощности применим формулу:

$$P = U_{\text{п}} * I_{\text{потр}}$$

$U_{\text{п}}$ — напряжение питания, $I_{\text{потр}}$ — ток, потребляемый ИМС. Рассчитаем статическую мощность, потребляемую спроектированным устройством, и проведем расчеты в таблице 4 [3, 4, 7].

Таблица 4 – Расчет потребляемой мощности

ИМС	**Количество**	**$I_{\text{потр}}$, мА**	**P, мВт**
ATMega32	3	15	225
MAX485	3	0.9	13.5
ADM707	3	0.25	3.75
		Итого:	242.25

Таким образом, суммарная потребляемая мощность равна 242.25 мВт.

4.1 Организация питания системы

Для предотвращения поломки устройства при скачках напряжения установим самовосстанавливающийся предохранитель и стабилитрон у разъема, питающего устройство. Ток в цепи равен полной потребляемой мощности, деленой на напряжение в цепи:

$$I = \frac{P}{U_{\text{п}}} = \frac{242.25 \text{ мВт}}{5 \text{ В}} = 48.45 \text{ мА}$$

Учитывая, что ток в цепи равен 48.45 мА, был установлен самовосстанавливающийся предохранитель MF-MSMF010 на 100 мА и стабилитрон 1N5339BRLG. При силе тока более 100 мА предохранитель сработает, что приведет к прекращению протекания тока через цепь. При напряжении более 7 В стабилитрон будет пропускать через себя большой ток на общую точку 0 В, что спровоцирует срабатывание предохранителя менее, чем за 10 секунд [5, 6].

Также была установлена ИМС ADM707 для генерации сигнала запуска устройства. Входы данной ИМС: $\overline{\text{MR}}$ (вход для подключения кнопки сброса

устройства), PFI (вход проверки питания), VCC (питание), GND (0 В). Выходы данной ИМС: RESET (сигнал включения устройства), $\overline{\text{RESET}}$ (инвертированный сигнал запуска устройства), $\overline{\text{PFO}}$ (выход проверки питания). На рисунке 7 показана схема данной ИМС [7].

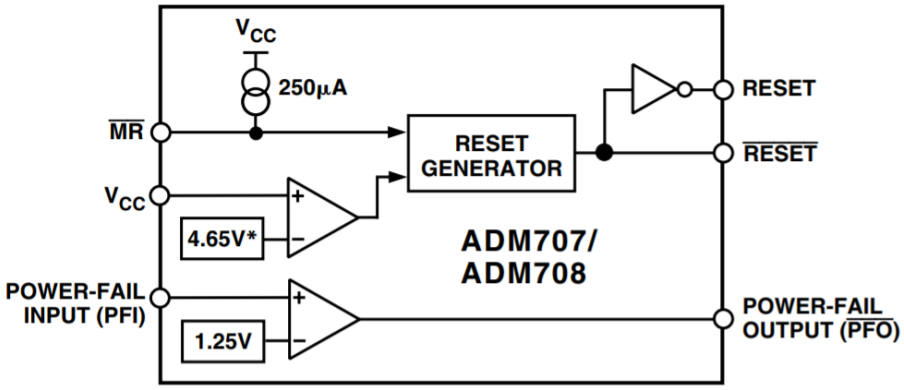


Рисунок 7 – Схема ИМС ADM707

На входы PFI и выход $\overline{\text{PFO}}$ подадим 0 В, так как не будем пользоваться функцией проверки питания. На вход $\overline{\text{MR}}$ также подадим 0 В, так как кнопка сброса не предусмотрена. Выход RESET соединим с общей точкой 0 В, так как не будем его использовать. Питание подадим на вход VCC, подадим 0 В на вход GND и подключим вход $\overline{\text{RESET}}$ к МК.

5 Разработка программ для микроконтроллеров

При выбранной конфигурации системы необходимо организовать общение МК следующим образом: главный МК с некоторой периодичностью посылает запросы данных двум другим МК, а они отвечают на данные запросы. Такой подход необходим по причине того, что, если каждый МК будет посылать свои данные тогда, когда он сам это решит, может возникнуть ситуация, что на участке цепи RS-485 окажется два пакета полезных данных, которые исказят друг друга, и полезная информация не дойдет до ее приемника.

Выберем в качестве главного микроконтроллера – мастера – микроконтроллер, отвечающий за отображение состояния очереди. Его удобно выбрать, так как ему необходимо общаться с двумя другими МК – слейвами, а им между собой общаться необязательно.

Также необходимо разработать схему общения МК, при которой слейвы будут распознавать определенный запрос от мастера и отвечать только на такие запросы, другие запросы должны игнорироваться, в том числе и те пакеты данных, которые отправляет другой слейв, ведь слейв будет переключаться в режим передатчика только в тех случаях, когда ему необходимо отправить данные, в остальных случаях он будет принимать все отправленные данные.

Для распознавания слейвами кому из них предназначен отправленный мастером запрос зарезервируем один из битов в пакете данных под флаг принадлежности слейву. Пусть этот бит будет равен 0, если запрос предназначен для МК, принимающего сигналы от администратора, и равен 1, если запрос предназначен для МК, принимающего сигналы от пользователей. Аналогично и с ответом мастеру от слейва: если флаг равен 0, то этот ответ от МК, принимающего сигналы от администратора, и 1, если это ответ от МК, принимающего сигналы от пользователей.

Полезные данные в пакете будут представлять из себя номер нового пользователя либо пользователя, следующего в очереди. Номер пользователя принимает значения от 1 до 99, соответственно нам будет достаточно 2^7=128

уникальных значений, для идентифицирования пользователя, то есть под полезные данные выделим 7 бит в пакете.

Таким образом размер пакета данных равен 8 бит и представляет из себя последовательность, описанную в таблице 5.

Таблица 5 – Пакет данных, отправляемый МК

Номер бита	7	6	5	4	3	2	1	0
Значение	Флаг принадлежности слейву	Полезная информация						

Разобравшись с архитектурой общения МК, была разработана схема взаимодействия МК, представленная на рисунке 8.

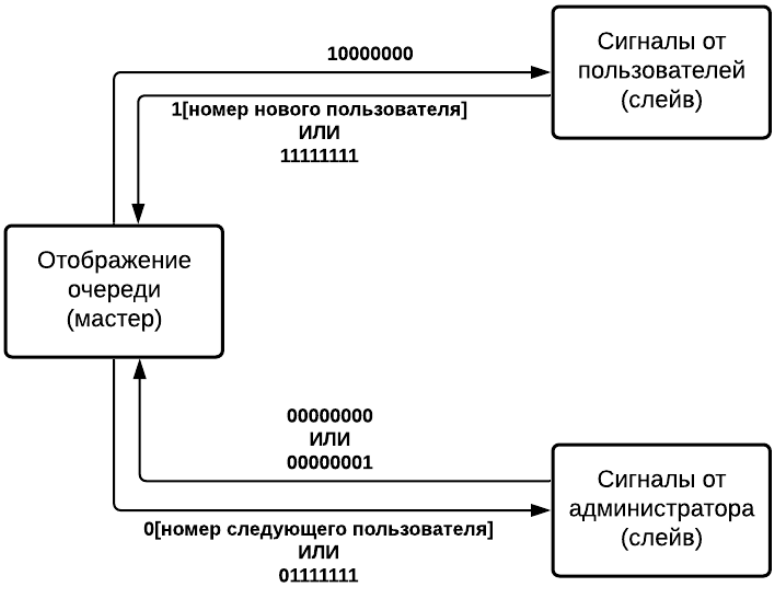


Рисунок 8 – Схема взаимодействия МК

Общение мастера со слейвом, принимающим сигналы от пользователей:

1) мастер отсылает запрос данных – 10000000;
2) слейв отвечает номером нового пользователя в очереди (с флагом принадлежности 1), если таковой появился, либо пакетом 11111111, если новый пользователь еще не пришел.

Общение мастера со слейвом, принимающим сигналы от администратора:

1) мастер посылает запрос данных в виде номера следующего в очереди пользователя (с флагом принадлежности 0), если очередь не пуста, либо пакетом 01111111, если очередь пуста;

2) слейв отвечает пакетом 00000000, если администратор еще не вызвал нового пользователя и 1 – в обратном случае.

Для программирования МК были использованы следующие источники:

– Хартов В.Я. – Микропроцессорные системы [8];
– электронный ресурс – Программирование МК AVR [9].

5.1 Разработка программы для мастера

5.1.1 Инициализация

Мастер отображает состояние очереди, а значит необходимо включить вывод на 7 пинах портов A и B и на 4 пинах порта C, в соответствие с принципиальной схемой (приложение Б).

На 2 и 1 пинах порта D необходимо включить вывод для управляющего сигнала MAX485 и TX UART соответственно.

На пинах 2 и 0 порта D необходимо включить подтягивающие резисторы для управляющего сигнала MAX485 и RX UART соответственно.

UART необходимо настроить так:

– включить скорость передачи BAUD = 9600 Бод (это повысит надежность передачи данных) при тактовой частоте работы МК равной f_{clk} = 8МГц:
$UBRR = \frac{f_{clk}}{16*BAUD} - 1 = \frac{8000000}{16*9600} - 1 \approx 51$ – содержимое регистра контроллера скорости передачи;
– разрешить прием и передачу данных в регистре UCSRB;
– установить бит URSEL регистра UCSRC в 1;
– включить 8-битные посылки данных (UCSZ1 = 1 и UCSZ0 = 1).

5.1.2 Глобальные переменные

Будут использоваться следующие глобальные переменные:

– queue – массив с номерами пользователей в очереди;
– qlen – длина очереди;
– digits – соответствие цифр и занчений на информационных входах дисплея (в соответствие с таблицей 1).

5.1.3 Код программы

Была разработана программа работы мастера, представленная в приложении Г на листинге 1. Обобщенная схема алгоритма представлена на рисунке 9.

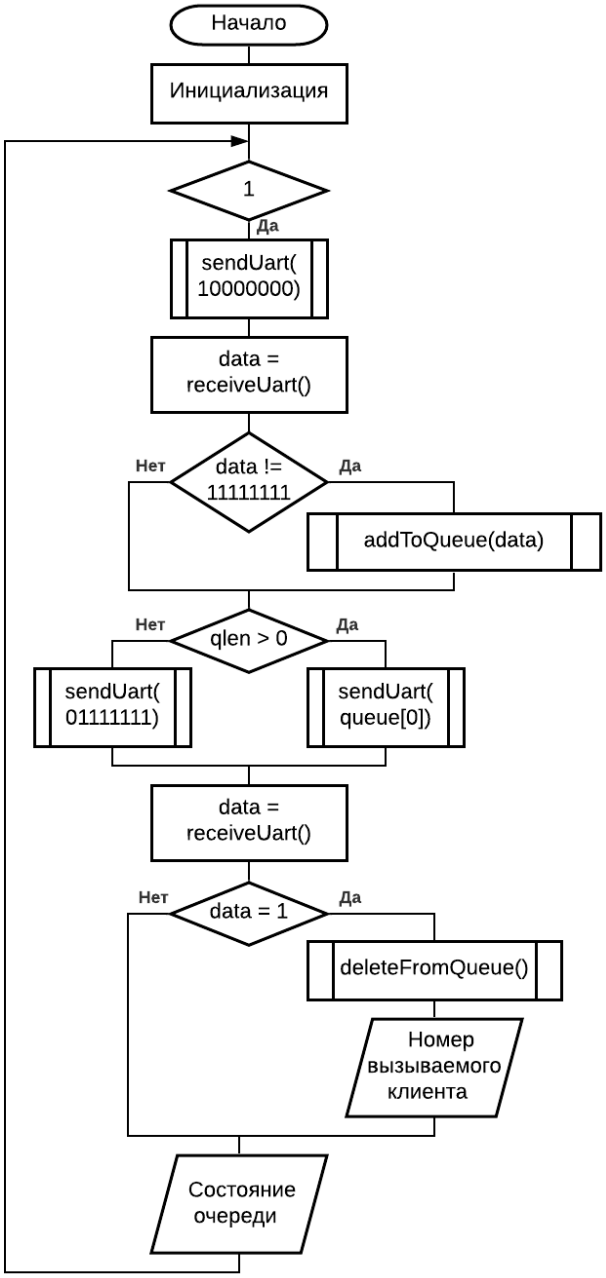


Рисунок 9 – Обобщенная схема алгоритма работы мастера

Были разработаны подпрограммы:

- sendUart – отправка пакета данных по UART;
- recieveUart – получение данных по UART;
- addToQueue – добавление пользователя в очередь;
- goToMK2 – общение с МК, принимающем сигналы от пользователей;

– deleteFromQueue – удаление пользователя из очереди;
– goToMK3 – общение с МК, принимающим сигналы от администратора;
– outputQueue – вывод состояния очереди на дисплеи.

5.2 Разработка программы для слейва (пользователи)

5.2.1 Инициализация

Этот слейв принимает сигналы от пользователей, а значит необходимо включить вывод на 7 пинах порта A и на 2 пинах порта B и ввод на третьем пине порта B, в соответствие с принципиальной схемой (приложение Б).

Порт D и UART должны быть настроены аналогично с мастером, но еще необходимо разрешить прерывания по приему UART (бит RXCIE регистра UCSRB) и глобально разрешить прерывания.

5.2.2 Глобальные переменные

Будут использоваться следующие глобальные переменные:

– counter – количество новых клиентов;
– send – количество отправленных новых клиентов.

5.2.3 Код программы

Была разработана программа работы слейва, принимающего сигналы от пользователей, представленная в приложении Г на листинге 2. Обобщенная схема алгоритма представлена на рисунке 10.

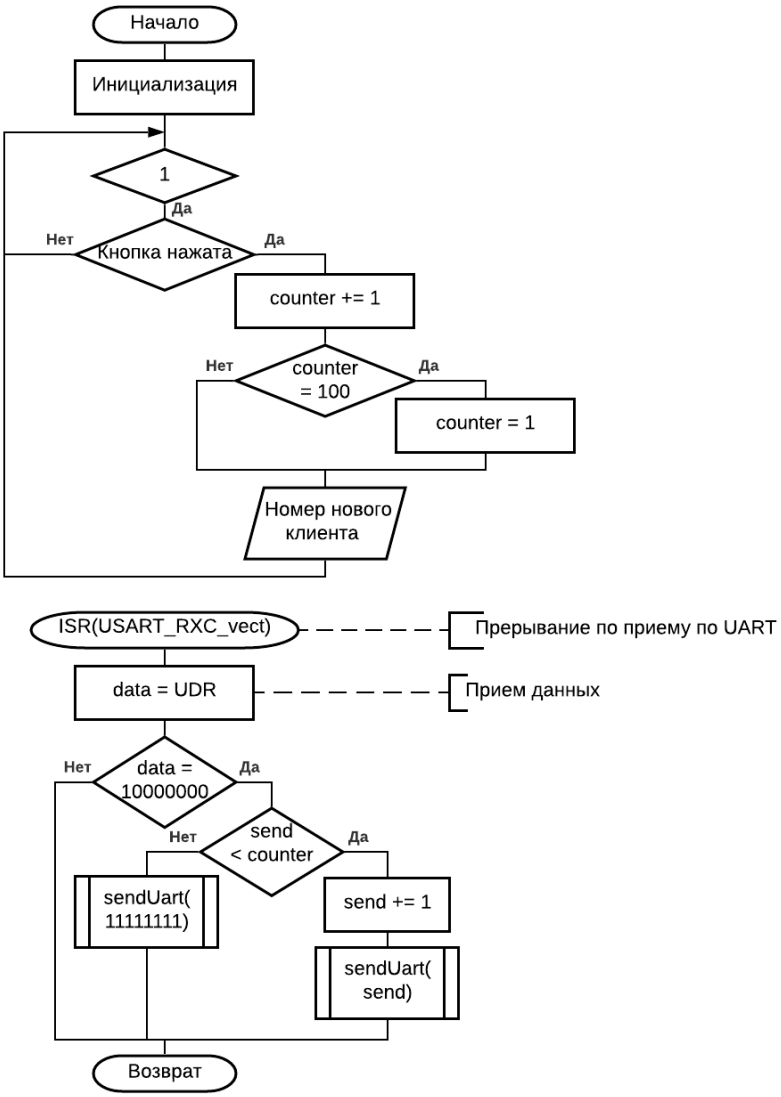


Рисунок 10 – Обобщенная схема алгоритма работы слейва, принимающего сигналы от пользователей

Были разработаны подпрограммы:

- sendUart – отправка пакета данных по UART;
- ISR(USART_RXC_vect) – прерывание по получению данных по UART;
- checkButton – проверка нажатия на кнопку.

5.3 Разработка программы для слейва (администраторы)

5.3.1 Инициализация

Этот слейв принимает сигналы от администратора. Он должен быть настроен аналогично со слейвом, принимающем сигналы от пользователей, в соответствие с принципиальной схемой (приложение Б).

5.3.2 Глобальные переменные

Будут использоваться следующие глобальные переменные:

– pressed – флаг нажатия на кнопку;
– emptyQueue – флаг пустоты очереди.

5.3.3 Код программы

Была разработана программа работы слейва, принимающего сигналы от администратора, представленная в приложении Г на листинге 3. Обобщенная схема алгоритма представлена на рисунке 11.

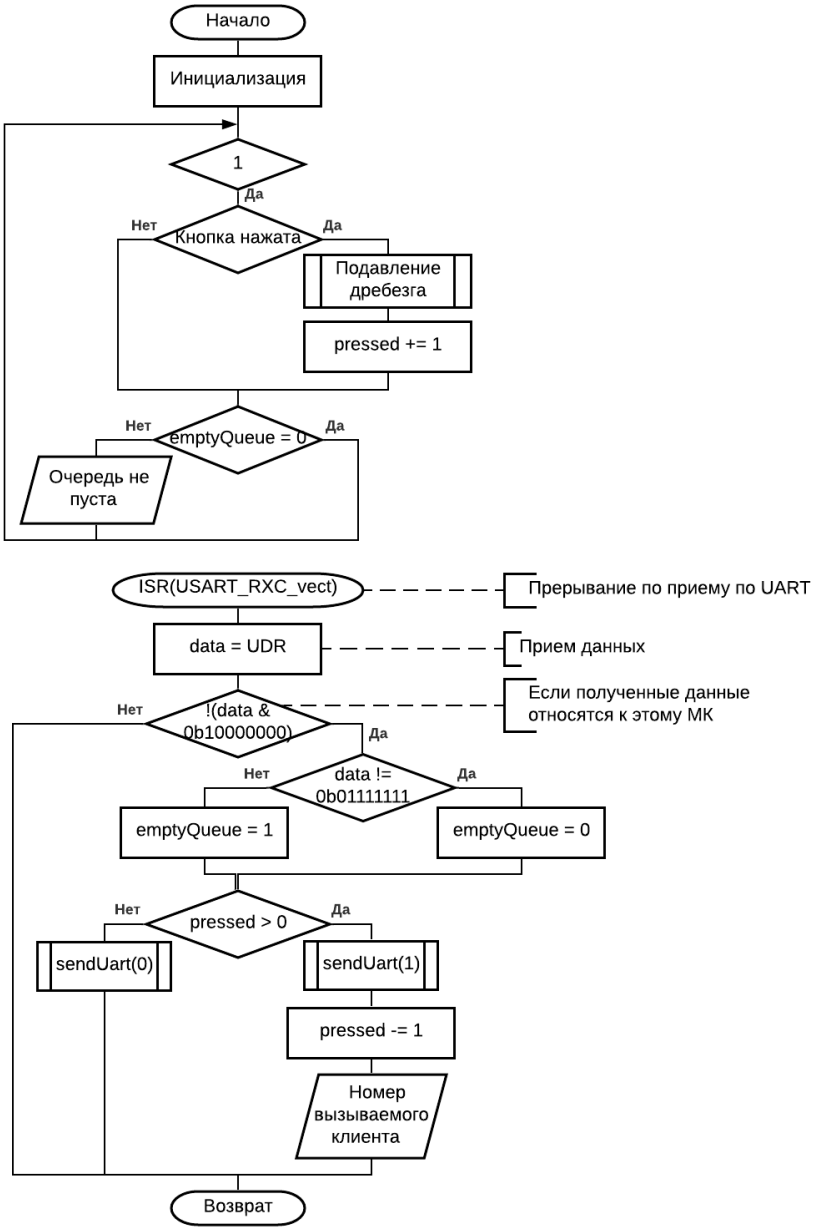


Рисунок 11 – Обобщенная схема алгоритма работы слейва, принимающего сигналы от администратора

Были разработаны подпрограммы:

- sendUart – отправка пакета данных по UART;
- ISR(USART_RXC_vect) – прерывание по получению данных по UART;
- checkButton – проверка нажатия на кнопку;
- outputNotEmptyQueue – вывод сигнала о том, что очередь не пуста.

6 Моделирование системы

При моделировании была использована среда Proteus ISIS 8. Схема, собранная в среде Proteus ISIS 8 приведена на рисунке 12.

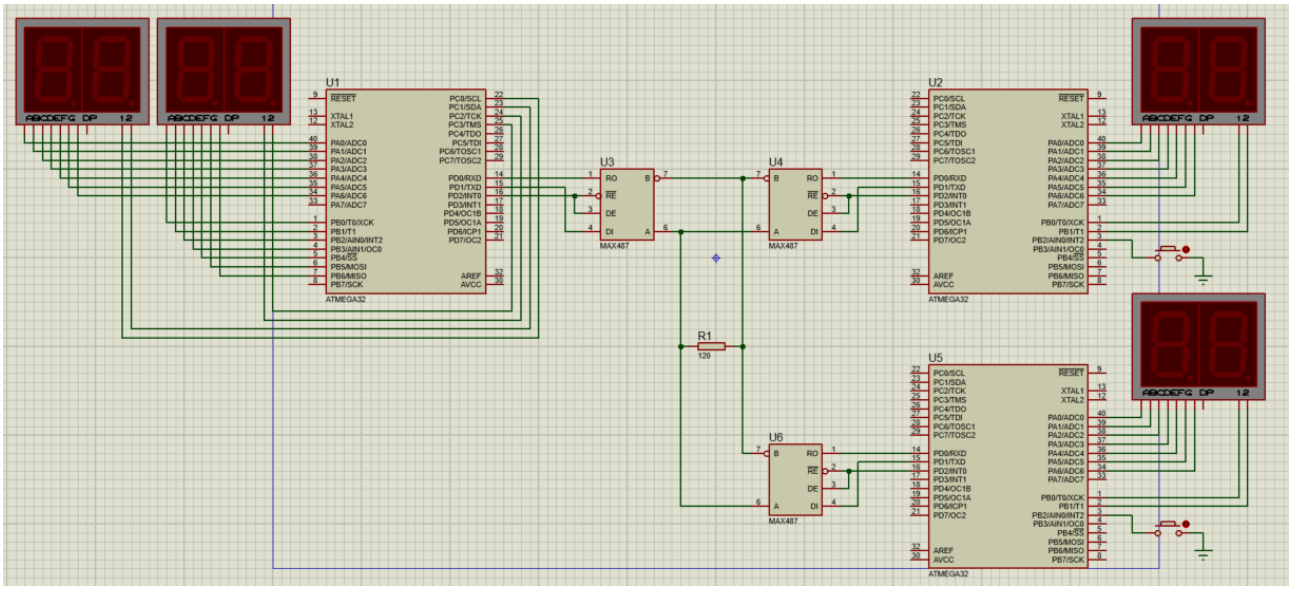


Рисунок 12 – Схема в среде Proteus ISIS 8

Все МК в среде Proteus ISIS 8 настраиваются на частоту 8 МГц, что показано на рисунке 13.

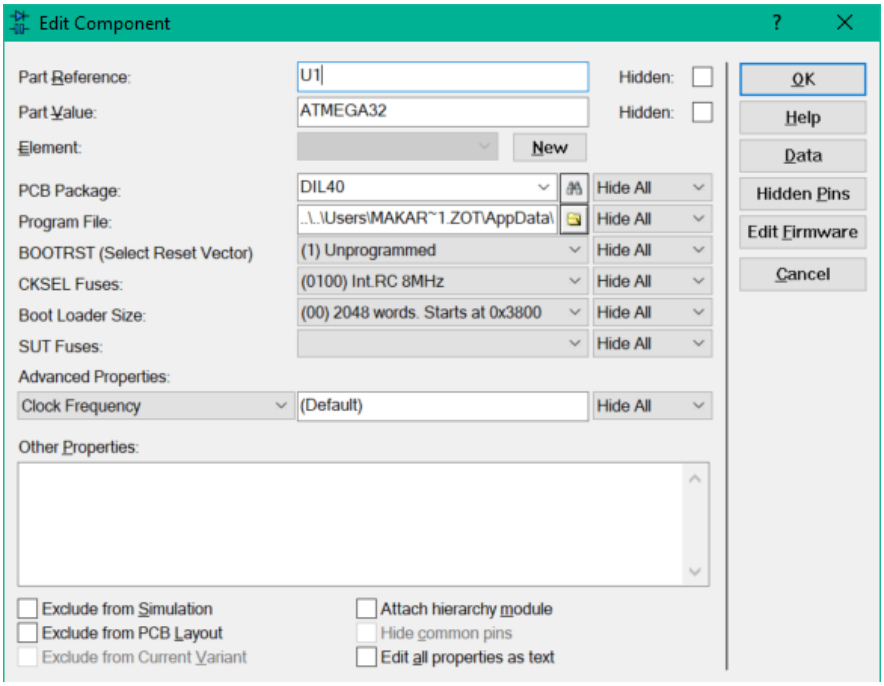


Рисунок 13 – Настройка МК в Proteus ISIS 8

7 Тестирование системы

При запуске симуляции в среде Proteus ISIS 8 дисплеи ничего не отображают (рисунок 12).

Протестируем обе функции системы:

1) добавление пользователя в очередь посредством нажатия на кнопку, подающую сигнал о добавлении нового пользователя в очередь;
2) удаление пользователя из очереди посредством нажатия на кнопку, подающую сигнал о вызове пользователя администратором.

7.1 Добавление пользователя в очередь

При нажатии на кнопку, подающую сигнал о добавлении нового клиента в очередь, номер этого пользователя (№1) мигает на дисплее, подключенном к МК, принимающем сигналы от пользователей, и появляется на первом дисплее, отображающем состояние очереди, а также на дисплее, подключенном к МК, принимающем сигналы от администратора, появляется сигнал о том, что очередь не пуста. При повторном нажатии происходит аналогичная ситуация, только номер этого пользователя (№2) появляется на втором дисплее, отображающем состояние очереди (рисунок 14).

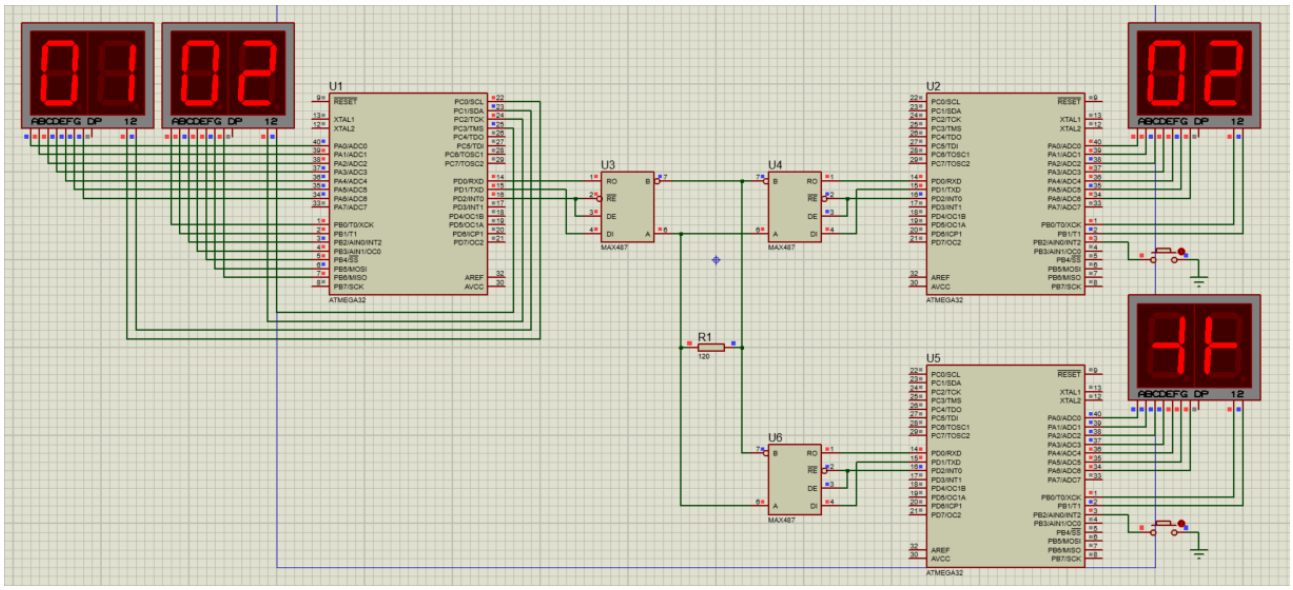


Рисунок 14 – Добавление пользователя в очередь

Оба пользователя успешно добавлены в очередь, администратор знает, что очередь не пуста. Тест прошел успешно.

7.2 Удаление пользователя из очереди

При нажатии на кнопку, подающую сигнал о вызове пользователя администратором, номер первого в очереди пользователя (№2) мигает на дисплее, отображающем очередь, и на дисплее, подключенном к МК, принимающем сигналы от администратора (рисунок 15), после чего пользователь удаляется из очереди (рисунок 16).

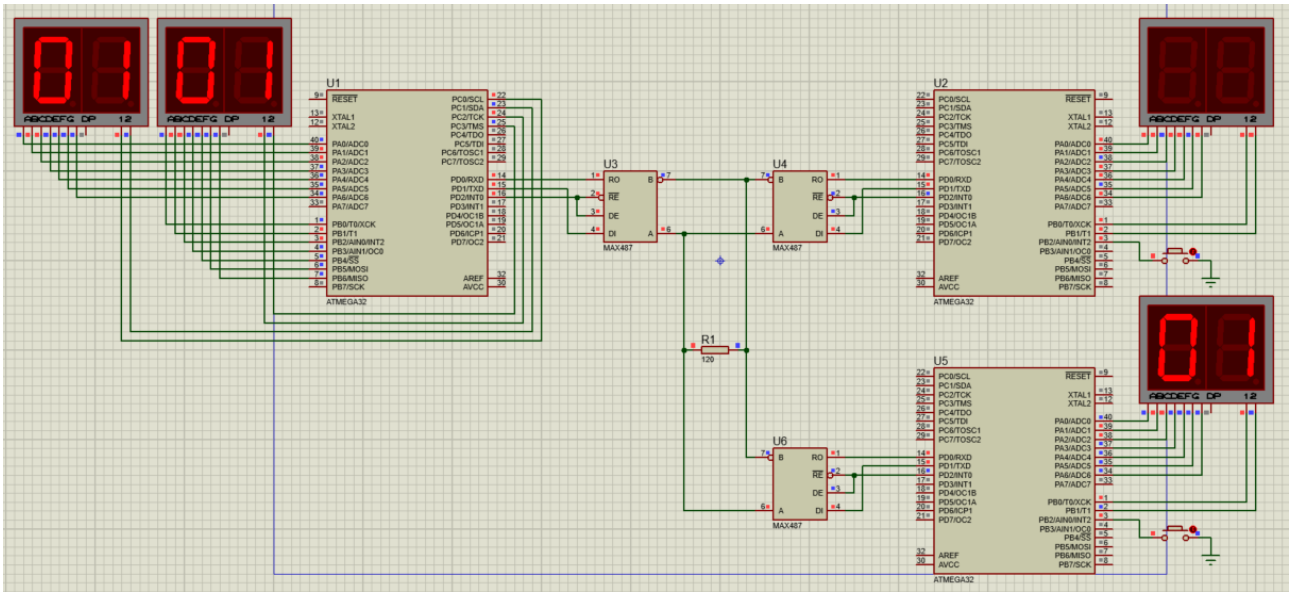


Рисунок 15 – Вызов пользователя администратором

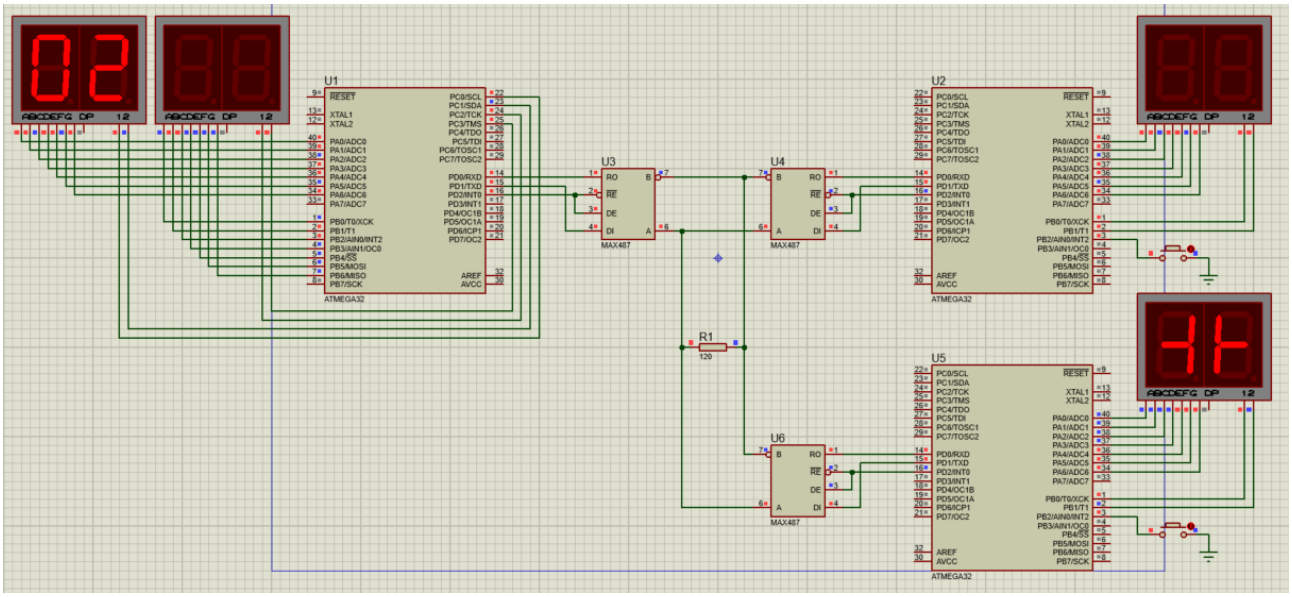


Рисунок 16 – Пользователь удален из очереди

При повторном нажатии на кнопку вызова пользователя, система приходит в исходное состояние – все дисплеи не горят (рисунок 17), но новый поль-

зователь в очереди получит номер 3. После 99 пользователя новому пользователю выдается номер 1.

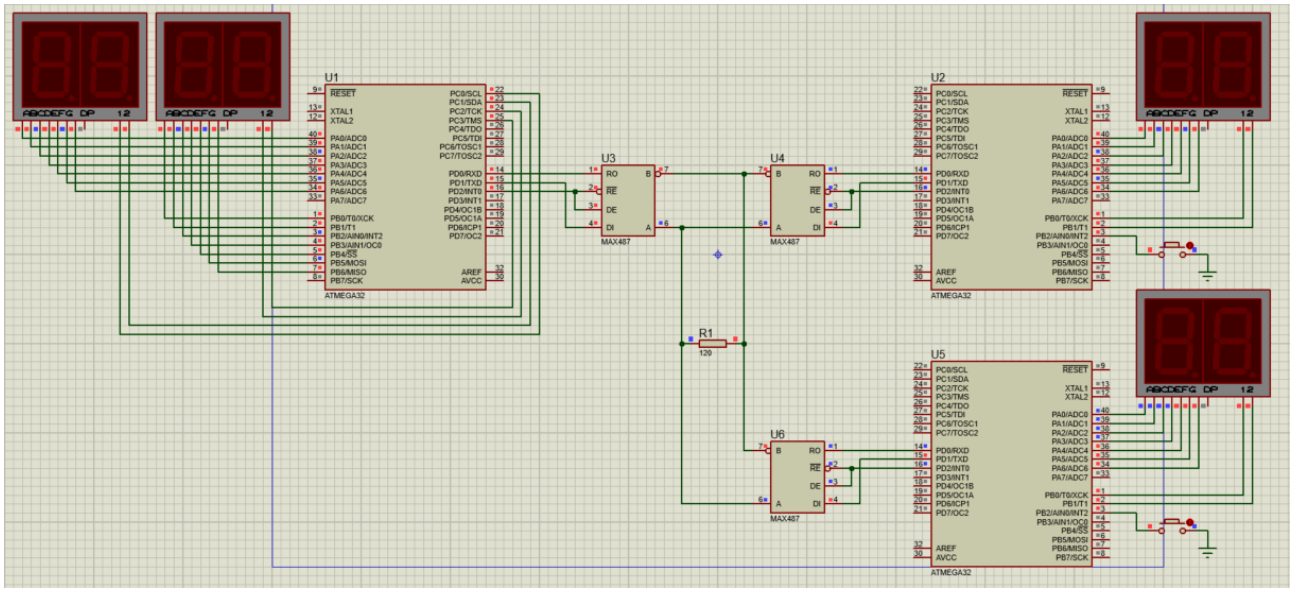


Рисунок 17 – Все пользователи удалены из очереди

Оба пользователя успешно удалены из очереди, администратор знает, что очередь пуста. Тест прошел успешно.

Заключение

В ходе выполнения данной курсовой работы была разработана система, предназначенная для организации электронной очереди, в которой участвуют пользователи, администраторы и устройства, отображающие состояние очереди.

Программный код системы был написан на языке Си процессоров семейства AVR, для чего была использована среда разработки WinAVR. Было произведено моделирование системы в среде Proteus ISIS 8, и выполнена ее симуляция.

Был подготовлен набор документации на объект разработки, состоящий из расчетно-пояснительной записки, функциональной схемы, принципиальной схемы, листинга программного кода, перечня элементов, использованных в системе.

Для развития разработанной системы может быть сделано большое количество доработок. Система легко расширяется: можно добавить в нее новые модули, отображающие состояние очереди, модули, принимающие сигналы от пользователей, или модули, принимающие сигналы от администратора. Также несложно увеличить максимальное количество пользователей в очереди и можно настроить МК на работу с более технологичными дисплеями.

Список литературы

1. D-sub [Электронный ресурс] URL: https://ru.wikipedia.org/wiki/D-sub (дата обращения: 12.12.2021)
2. LED NUMERIC DISPLAY, 2 DIGIT BL-D56X-22 [Электронный ресурс] URL: https://www.marthel.pl/katalog/Betlux/BL-D56E-22.pdf (дата обращения: 12.12.2021)
3. ATmega32-16AU Datasheet [Электронный ресурс] URL: https://static.chipdip.ru/lib/059/DOC000059770.pdf (дата обращения: 12.12.2021)
4. MAX481/MAX483/MAX485/MAX487–MAX491/MAX1487 Datasheet [Электронный ресурс] URL: http://www.farnell.com/datasheets/1700964.pdf (дата обращения: 12.12.2021)
5. MF-MSMF Series - PTC Resettable Fuses [Электронный ресурс] URL: https://static.chipdip.ru/lib/245/DOC000245515.pdf (дата обращения: 12.12.2021)
6. 1N53 Series [Электронный ресурс] URL: https://eu.mouser.com/datasheet/2/308/1N5333B_D-2309169.pdf (дата обращения: 12.12.2021)
7. ADM707 Datasheet [Электронный ресурс] URL: https://static.chipdip.ru/lib/143/DOC000143996.pdf (дата обращения: 12.12.2021)
8. Микропроцессорные системы: учеб. пособие для студ. учреждений высш. образования / В.Я. Хартов, – 2-е изд., испр. и доп. – М.: Издательский центр «Академия», 2014, 368 с. – (Сер. Бакалавриат). – ISBN 978-5-4468-0440-5;
9. Программирование МК AVR [Электронный ресурс] URL: https://narodstream.ru/programmirovanie-mk-avr/ (дата обращения: 12.12.2021)

Приложение А

Схема электрическая функциональная

Листов 1

Приложение Б

Схема электрическая принципиальная

Листов 1

Приложение В

Спецификация (перечень) используемых элементов

Листов 2

Приложение Г

Исходный код программ

Листинг 1 – Код программы МК, отображающего состояние очереди

```
#include <avr/io.h>
#include <util/delay.h>

volatile unsigned char queue[100], //очередь
                       qlen = 0; // длина очереди
const unsigned char digits[] = { 0b00111111,
                                 0b00000110,
                                 0b01011011,
                                 0b01001111,
                                 0b01100110,
                                 0b01101101,
                                 0b01111101,
                                 0b00000111,
                                 0b01111111,
                                 0b01101111 }; // соответствие цифр и их обозначения на
дисплее

void initUART(void) // инициализация UART
{
    UBRRL=51; //скорость 9600 Бод при f=8 МГц
    UCSRB=(1<<RXEN)|(1<<TXEN); //разрешение приема и передачи
    UCSRC=(1<<URSEL)|(1<<UCSZ1)|(1<<UCSZ0); //8-битные посылки
}

void sendUart(unsigned char c) // отправка по UART
{
    PORTD |= (1<<PD2); // включение передачи
    UDR = c;
    while(!(UCSRA&(1<<UDRE))) {}; // ожидание опустошения UDR
    while(!(UCSRA&(1<<TXC))) {};  // ожидание отправки
    PORTD &= ~(1<<PD2); // включение приема
}

unsigned char recieveUart(void) // прием по UART
{
    while(!(UCSRA&(1<<RXC))) {}; // ожидание приема
    PORTD |= (1<<PD2); // включение передачи
    return UDR;
}

void addToQueue(unsigned char a) // добавление пользователя в очередь
{
    queue[qlen] = a;
    qlen++;
}

void goToMK2(void) // общение с МК2
{
    sendUart(0b10000000);
    unsigned char data = recieveUart();
    if (data != 0b11111111)
        addToQueue(data&0b01111111);
}

unsigned char deleteFromQueue(void) // удаление человека из очереди
{
    unsigned char rett = queue[0];
    unsigned char i = 0;
    while (queue[i] != 0)
    {
        queue[i] = queue[i+1];
        i++;
    }
```

```
    qlen--;
    return rett;
}

void goToMK3(void) // общение с МК3
{
    unsigned char i, j, data;

    if (qlen > 0) // если очередь не пуста
        sendUart(queue[0]); // отправка номера следующего в очереди человека
    else
        sendUart(0b01111111); // отправка сигнала, что очередь пуста

    data = recieveUart();

    if (data == 1)
    {
        unsigned char a = deleteFromQueue();
        for (j = 0; j < 3; j++) // вывод вызываемого пользователя
        {
            for (i = 0; i < 50; i++)
            {
                PORTC = 0b00001010;
                PORTA = digits[a / 10];
                PORTB = digits[a / 10];
                _delay_ms(10);
                PORTC = 0b00000101;
                PORTA = digits[a % 10];
                PORTB = digits[a % 10];
                _delay_ms(10);
            }
            PORTC = 0b00001111;
            _delay_ms(500);
        }
    }
}

void outputQueue(void) // вывод очереди
{
    if (qlen > 0)
    {
        if (qlen > 1)
        {
            PORTC = 0b00001010;
            PORTA = digits[queue[0] / 10];
            PORTB = digits[queue[1] / 10];
        }
        else
        {
            PORTC = 0b00001110;
            PORTA = digits[queue[0] / 10];
        }
        _delay_ms(10);
        if (qlen > 1)
        {
            PORTC = 0b00000101;
            PORTA = digits[queue[0] % 10];
            PORTB = digits[queue[1] % 10];
        }
        else
        {
            PORTC = 0b00001101;
            PORTA = digits[queue[0] % 10];
        }
        _delay_ms(10);
    }
}

int main()
{
    initUART();
```

```
    DDRA  = 0b01111111; // включение вывода для 1 дисплея
    DDRB  = 0b01111111; // включение вывода для 2 дисплея
    DDRC  = 0b00001111; // включение вывода для управляющих сигналов дисплеев
    DDRD  = 0b00000110; // включение вывода на портах 2 (управление MAX485) и 1 (TX)
    PORTD = 0b00000101; // включение подтягивающего резистора на порте 0 (RX) и режима
отправки для MAX485

    while(1)
    {
        goToMK2();
        goToMK3();
        outputQueue();
    }
}
```

Листинг 2 – Код программы МК, принимающего сигналы от пользователей

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

volatile unsigned char counter = 0, // количество новых клиентов
                       send = 0;    // количество отправленных новых клиентов

void initUART(void) // инициализация UART
{
    UBRRL=51; //скорость 9600 Бод при f=8 МГц
    UCSRB=(1<<RXCIE)|(1<<RXEN)|(1<<TXEN); //разрешение прерывания по приему, приема и
передачи
    UCSRC=(1<<URSEL)|(1<<UCSZ1)|(1<<UCSZ0); //8-битные посылки
}

void sendUart(unsigned char c) // отправка по UART
{
    PORTD |= (1<<PD2); // включение передачи
    UDR = c;
    while(!(UCSRA&(1<<UDRE))) {}; // ожидание опустошения UDR
    while(!(UCSRA&(1<<TXC))) {};  // ожидание отправки
    PORTD &= ~(1<<PD2); // включение приема
}

ISR(USART_RXC_vect) // прерывание по приему по UART
{
    unsigned char data = UDR;
    if (data==0b10000000) // если полученные данные относятся к этому МК
    {
        if (send < counter)
        {
            send++;
            sendUart(send|0b10000000); // добавить пользователя в очередь
        }
        else
            sendUart(0b11111111); // не добавлять никого в очередь
    }
}

void checkButton(unsigned char button_pin) // проверка нажатия на кнопку
{
    unsigned char i, j;
    const unsigned char digits[] = { 0b00111111,
                                     0b00000110,
                                     0b01011011,
                                     0b01001111,
                                     0b01100110,
                                     0b01101101,
                                     0b01111101,
                                     0b00000111,
                                     0b01111111,
                                     0b01101111 }; // соответствие цифр и их обозначения
```

```
на дисплее
    if (!(PINB&button_pin)) // если кнопка нажата
    {
        counter++;
        if (counter == 100) // если счетчик новых клиентов переполнился
            counter = 1;
        for (j = 0; j < 3; j++) // вывод номера нового клиента
        {
            for (i = 0; i < 50; i++)
            {
                PORTB = 0b00000110;
                PORTA = digits[counter / 10];
                _delay_ms(10);
                PORTB = 0b00000101;
                PORTA = digits[counter % 10];
                _delay_ms(10);
            }
            PORTB = 0b00000111;
            _delay_ms(500);
        }
    }
}

int main()
{
    sei(); // глобальное разрешение прерываний

    initUART();
    DDRA  = 0b01111111; // включение вывода для дисплея
    DDRB  = 0b00000011; // включение вывода для управляющих сигналов дисплея
    PORTB = 0b00000111; // включение подтягивающего резистора на порте 2 (кнопка) и вы-
ключение управляющих сигналов дисплея
    DDRD  = 0b00000110; // включение вывода на портах 2 (управление MAX485) и 1 (TX)
    PORTD = 0b00000001; // включение подтягивающего резистора на порте 0 (RX) и режима
приема для MAX485

    while(1)
        checkButton(1<<PD2);
}
```

Листинг 3 – Код программы МК, принимающего сигналы от администратора

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

volatile char pressed = 0,    // флаг нажатия на кнопку
              emptyQueue = 1; // флаг пустоты очереди

void initUART(void) // инициализация UART
{
    UBRRL=51; //скорость 9600 Бод при f=8 МГц
    UCSRB=(1<<RXCIE)|(1<<RXEN)|(1<<TXEN); //разрешение прерывания по приему, приема и
передачи
    UCSRC=(1<<URSEL)|(1<<UCSZ1)|(1<<UCSZ0); //8-битные посылки
}

void sendUart(unsigned char c) // отправка по UART
{
    PORTD |= (1<<PD2); // включение передачи
    UDR = c;
    while(!(UCSRA&(1<<UDRE))) {}; // ожидание опустошения UDR
    while(!(UCSRA&(1<<TXC))) {};  // ожидание отправки
    PORTD &= ~(1<<PD2); // включение приема
}

ISR(USART_RXC_vect) // прерывание по приему по UART
{
    unsigned char i, j, data = UDR;
```

```
    const unsigned char digits[] = { 0b00111111,
                                     0b00000110,
                                     0b01011011,
                                     0b01001111,
                                     0b01100110,
                                     0b01101101,
                                     0b01111101,
                                     0b00000111,
                                     0b01111111,
                                     0b01101111 }; // соответствие цифр и их обозначения
на дисплее
    if (!(data&0b10000000)) // если полученные данные относятся к этому МК
    {
        if (data != 0b01111111)
            emptyQueue = 0;
        else
            emptyQueue = 1;

        if (data == 0b01111111 && pressed > 0) // если очередь пуста, но кнопка была
нажата
        {
            sendUart(0);
            pressed--;
        }
        else if (pressed > 0) // если кнопка была нажата
        {
            sendUart(1);
            pressed--;
            for (j = 0; j < 3; j++) // вывод номера вызываемого пользователя
            {
                for (i = 0; i < 50; i++)
                {
                    PORTB = 0b00000110;
                    PORTA = digits[data / 10];
                    _delay_ms(10);
                    PORTB = 0b00000101;
                    PORTA = digits[data % 10];
                    _delay_ms(10);
                }
            PORTB = 0b00000111;
            _delay_ms(500);
            }
        }
        else
            sendUart(0);
    }
}

void checkButton(unsigned char button_pin) // проверка нажатия на кнопку
{
    if (!(PINB&button_pin))
    {
        while (!(PINB&button_pin));
        pressed++;
    }
}

void outputNotEmptyQueue(void) // вывод сигнала о том, что очередь не пуста
{
    PORTB = 0b00000110;
    PORTA = 0b01000110;
    _delay_ms(10);
    PORTB = 0b00000101;
    PORTA = 0b01110000;
    _delay_ms(10);
    PORTB = 0b00000111;
}

int main()
{
    sei(); // глобальное разрешение прерываний
```

```
    initUART();
    DDRA  = 0b01111111; // включение вывода для дисплея
    DDRB  = 0b00000011; // включение вывода для управляющих сигналов дисплея
    PORTB = 0b00000111; // включение подтягивающего резистора на порте 2 (кнопка) и выключение управляющих сигналов дисплея
    DDRD  = 0b00000110; // включение вывода на портах 2 (управление MAX485) и 1 (TX)
    PORTD = 0b00000001; // включение подтягивающего резистора на порте 0 (RX) и режима приема для MAX485

    while(1)
    {
        checkButton(1<<PD2);
        if (emptyQueue == 0)
            outputNotEmptyQueue();
    }
}
```