Содержание

Вступление

Основным учебным пособием для изучения работы микроконтроллеров являются разнообразные отладочные стенды. Эти устройства представляют собой печатные платы с размещенными на них микроконтроллером и значительным количеством периферийных устройств различных типов. Отладочные стенды дают возможность испытания программы на реальном устройстве, а также, в некоторых случаях, отладки (debug) в режиме реального времени.

Испытания программы на реальном устройстве дают возможность оценить скорость её работы, удобство работы пользователя с ней или взаимодействие всех частей программно-аппаратного комплекса в целом. В то же время, запись программы («прошивка») – одна из наиболее трудозатратных операций, т.к. требует подключения программатора, перевода устройства в режим программирования и проведения дополнительных операций, связанных с обеспечением нормального функционирования устройства (правильный порядок включения, подготовка программатора, подготовка программного обеспечения для работы с программатором). Немаловажно также то, что внутренняя память микроконтроллеров имеет ограниченное количество циклов перезаписи, достаточное для промышленного программирования, но ощутимо малое для учебных целей.

Для сведения нагрузки на стенд к минимуму можно применять виртуальный стенд, который позволяет избежать отладки мелких проблем на реальном оборудовании и сосредоточится лишь на специфических вопросах конкретной задачи. Такой виртуальный стенд должен полностью повторять логику работы реального и отличаться только ограничениями накладываемыми средствами моделирования.

Разработка виртуальной модели учебно-отладочного стенда описана в данной дипломной работе.

1 Применение учебно-отладочных стендов

Подготовка специалистов для проектирования и эксплуатации автоматизированных систем измерений, испытаний и управления требует организации лабораторных практикумов, позволяющих изучать компоненты этих систем, приобретать соответствующие практические навыки. Огромная, непрерывно обновляющаяся номенклатура средств автоматизации и инструментария для интеграции их в системы ставит перед техническими вузами практически неразрешимые проблемы внедрения методик ускоренного обучения и постоянного совершенствования лабораторной базы. Создание современных учебных лабораторий требует значительных финансовых затрат на приобретение технических средств, поддержание их в работоспособном состоянии, разработку методических материалов.

Учебно-отладочные стенды позволяют студентам изучить и испытать работу микроконтроллерных устройств с различными видами периферии. Также могут быть промоделированы действия, выполняемые реальными системами управления и измерения, такие как преобразования аналоговых сигналов в цифровые, обратные преобразования, считывание состояний датчиков, управление исполнительными устройствами (например, двигателями), индикация состояния системы и т.д.. Помимо перечисленного выше, зачастую учебно-отладочные стенды имеют возможность предоставлять пользователям информацию о содержимом регистров микроконтроллера, внутренней и внешней памяти, выполняемой в данный момент инструкции. Этот функционал позволяет «обкатать» разрабатываемую программу перед внедрением её в производство.

Таким образом, учебно-отладочные стенды являются одним из основных наглядных пособий для изучения дисциплин «Микропроцессорные системы» и «Электроника и микросхемотехника» в высших учебных заведениях, а также важнейшим тестовым оборудованием на производстве.

2 Выбор подхода к моделированию

Кафедра АУТС факультета информатики и вычислительной техники НТУУ «КПИ» успешно применяет в учебном процессе учебно-отладочные стенды производства ЧМП «Оупен Систем». Было решено для создания виртуального учебно-отладочного стенда взять за основу стенд EV8031/AVR и плату расширения к нему EV8031/AN от данного производителя.

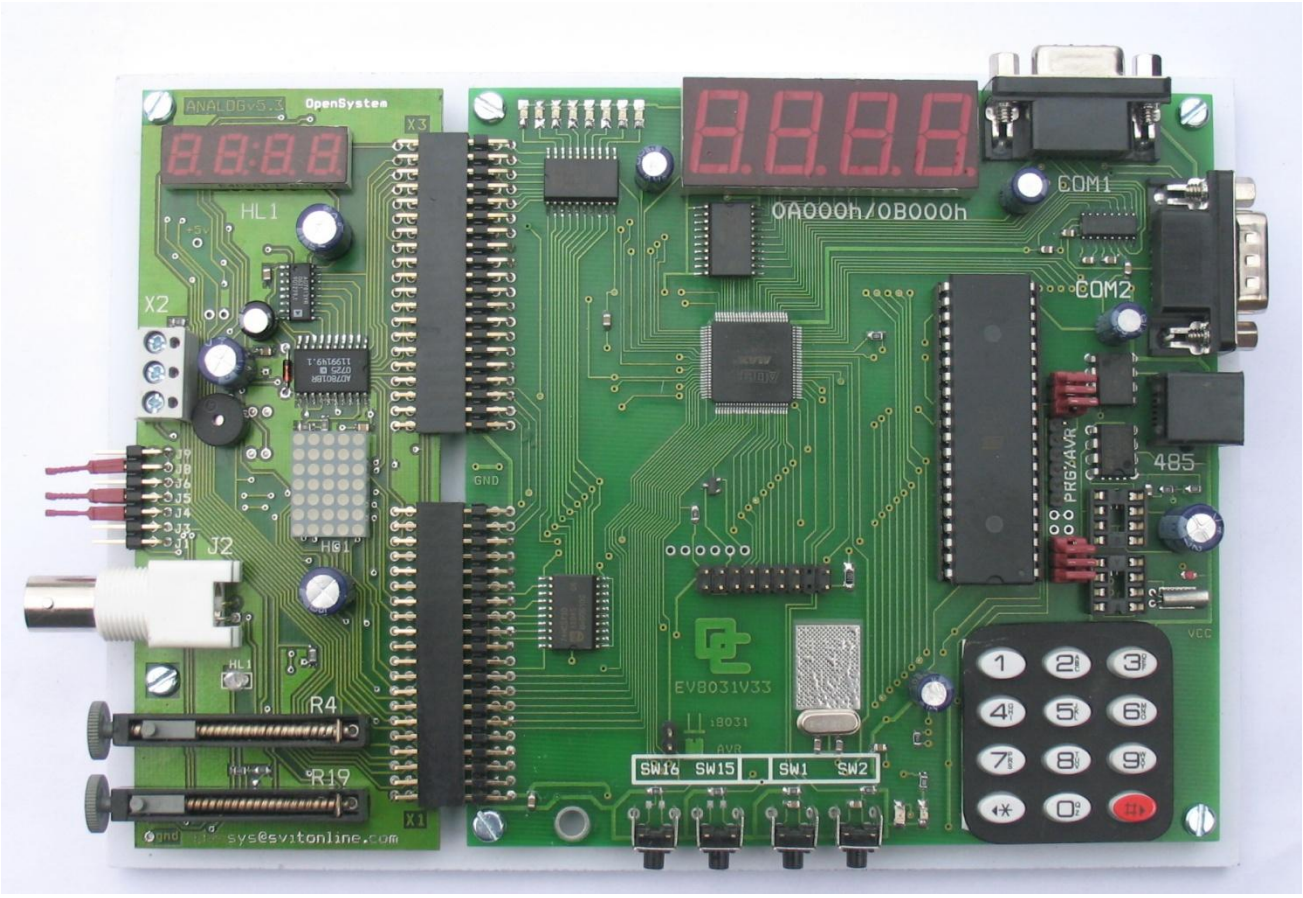


Рисунок 2.1 – Учебно-отладочный стенд EV8031/AVR

Виртуальный учебно-отладочный стенд должен полностью повторять функционал реального изделия. Это позволит студентам выполнять лабораторные работы в домашних условиях и демонстрировать их для проверки на реальном устройстве в учебной лаборатории. Такой подход решает следующие проблемы: ограниченное количество учебно-отладочных стендов, недостаток времени для написания программ под неизученное оборудование,

ненулевая вероятность поломки оборудования при выполнении программы с ошибками.

В качестве средства моделирования применяется пакет Proteus ISIS. Пакет представляет собой систему схемотехнического моделирования, базирующуюся на основе моделей электронных компонентов принятых в PSpice. Отличительной чертой пакета PROTEUS VSM является возможность моделирования работы программируемых устройств: микроконтроллеров, микропроцессоров, DSP и проч. Библиотека компонентов содержит справочные данные. Дополнительно в пакет PROTEUS VSM входит система проектирования печатных плат. Пакет Proteus состоит из двух частей, двух подпрограмм: ISIS — программа синтеза и моделирования непосредственно электронных схем и ARES — программа разработки печатных плат. Вместе с программой устанавливается набор демонстрационных проектов для ознакомления. Разработка компании Labcenter Electronics (Великобритания).

3 Учебно-отладочный стенд EV8031/AVR

Программно-аппаратный комплекс "EV8031/AVR", предназначен для применения в учебных целях по курсам программирование (язык Ассемблер, СИ), а также как средство разработки программного обеспечения для контроллеров на базе однокристальной ЭВМ серии MSC-51, либо микроконтроллеров архитектуры AVR. Стенд построен на современной элементной базе. В составе стенда имеется два стандартных RS-232C порта, последовательная Flash-память с интерфейсом I^2C, память программ и память данных по 64КБ. Наличие системного и периферийного интерфейсов позволяет использовать стенд для отладки любых систем.

Системный интерфейс содержит полную шину адреса (16 линий), шину данных, линии прерываний и сигналы управления памятью, а также цепи питания. Периферийный интерфейс состоит из микросхемы параллельного приемопередатчика (24 линии ввода-вывода), линии порта P1 однокристальной ЭВМ (8 двунаправленных линий), линии прерываний, а также линии таймеров-счетчиков и цепи питания. Наличие интерфейса RS-485, 4-х разрядная динамическая индикация, возможность расширения, установка датчиков температуры, часов реального времени, разъем программирования для процессоров AVR, разъем для подключения ЖК индикатора.

3.1 Комплектация учебно-отладочного стенда

Стенд состоит из следующих блоков:

- Микроконтроллер i8051 / ATmega8515S;
- Память данных EEPROM объемом 32 Кб (AT24C02);
- Блок статической индикации (4 разряда):
- Блок дискретной индикации (8 разрядов):
- Интерфейс RS-232 (2 канала);
- Интерфейс RS-485;

– Матричная клавиатура 3*4;
– Блок инициализации прерываний: дискретные кнопки - 2 шт. (линии INT0, INT1)
– Плата расширения "EV8031/AN” или "EV8031/AU"

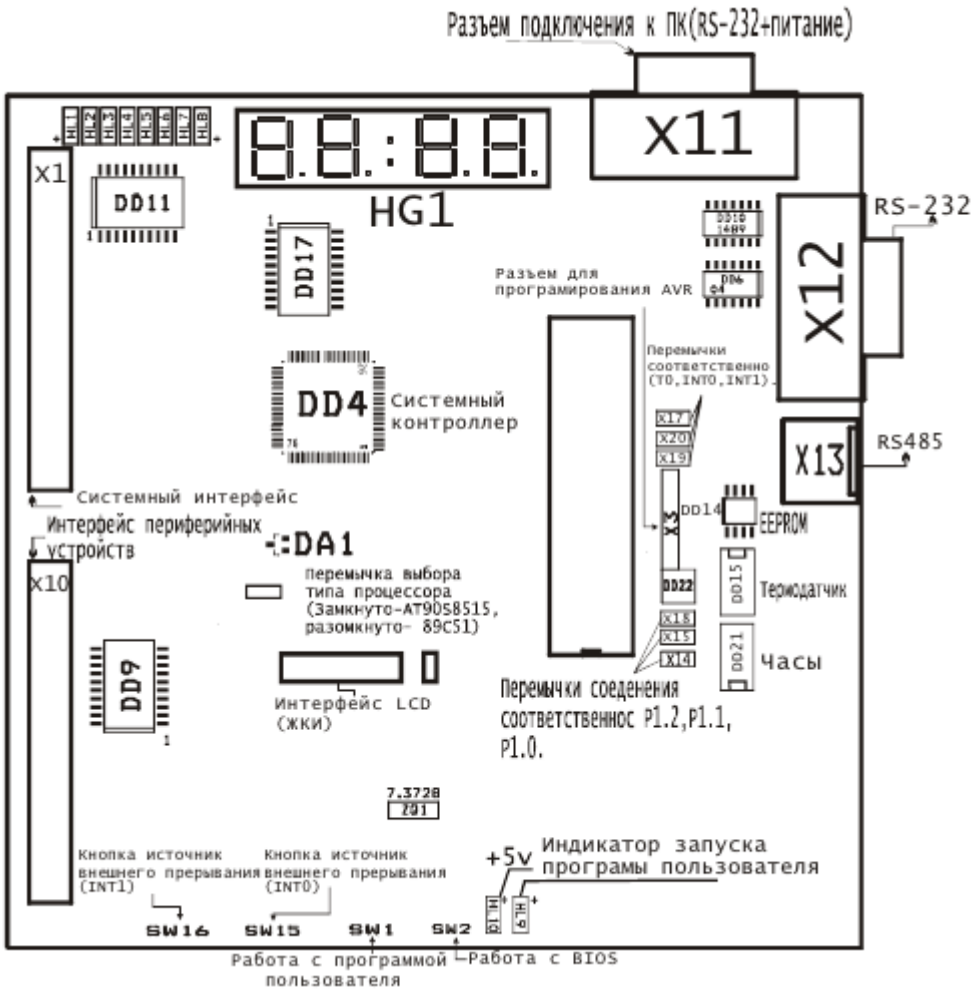


Рисунок 3.1 – Схема расположения элементов стенда

3.2 Комплектация плат расширения

Платы расширения служат для увеличения количества периферийных устройств доступных для применения на учебно-отладочном стенде. Платы расширения соединяются с EV8031/AVR через системный интерфейс и интерфейс периферийных устройств. Это позволяет менять платы в зависимости от потребности в различных ПУ.

Существует два типа плат расширения: EV8031/AN и EV8031/AU. Их комплектация приведена ниже.

Плата расширения EV8031/AN:

- ЦАП 8 разрядов (AD7801);
- динамическая 4-х разрядная индикация;
- знакосинтезирущая индикация 5x7;
- компаратор напряжения;
- резистор изменения аналогового сигнала;
- генератор с изменяемой частотой генерации;
- генератор с фиксированной частотой генерации;
- динамик;
- BNC разъем.

Плата расширения EV8031/AU:

- Динамик;
- N-кодер;
- дискретные кнопки - 2 шт.;
- вентилятор KD0504PFS2;
- датчик Холла - SS443З;
- нагревательный элемент;
- датчик температуры TMP03;
- ЦАП 8 разрядов (AD7801).

4 Принцип работы учебно-отладочного стенда EV8031/AVR

Программа загрузчик находится в Flash-памяти микроконтроллера AT89C51 либо ATmega8515S, она проводит инициализацию последовательного приемопередатчика ОЭВМ (DD1), проверяет наличие и ёмкость памяти данных.

Память ОЗУ объёмом 32К делится на две части по 16К. Одна часть для памяти программ, другая для памяти данных. В режиме загрузки вся память 32К отображается в адресное пространство как память данных.

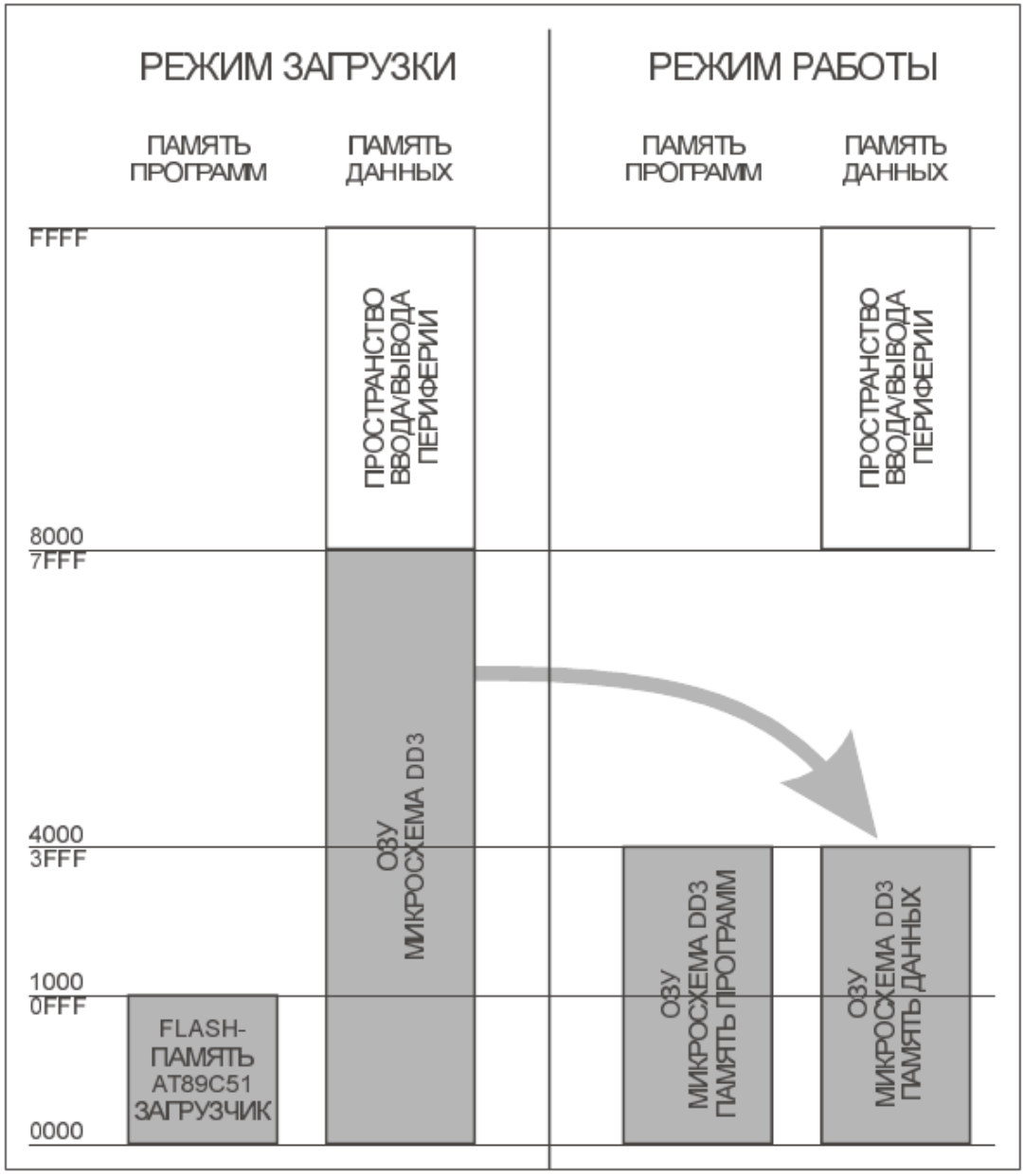


Рисунок 4.1 – Распределение памяти стенда

При поступлении данных с последовательного порта персонального компьютера в последовательный порт (разъём X11) стенда, ОЭВМ записывает их в ОЗУ отведённое под память программ. Сигналы управления – PME, WR, RD, ALE, формируемые микропроцессором и необходимые для обращения к памяти данных поступают через системный контроллер. После принятия последнего байта загрузчик формирует сигнал запуска программы записью управляющего кода в системный контроллер.

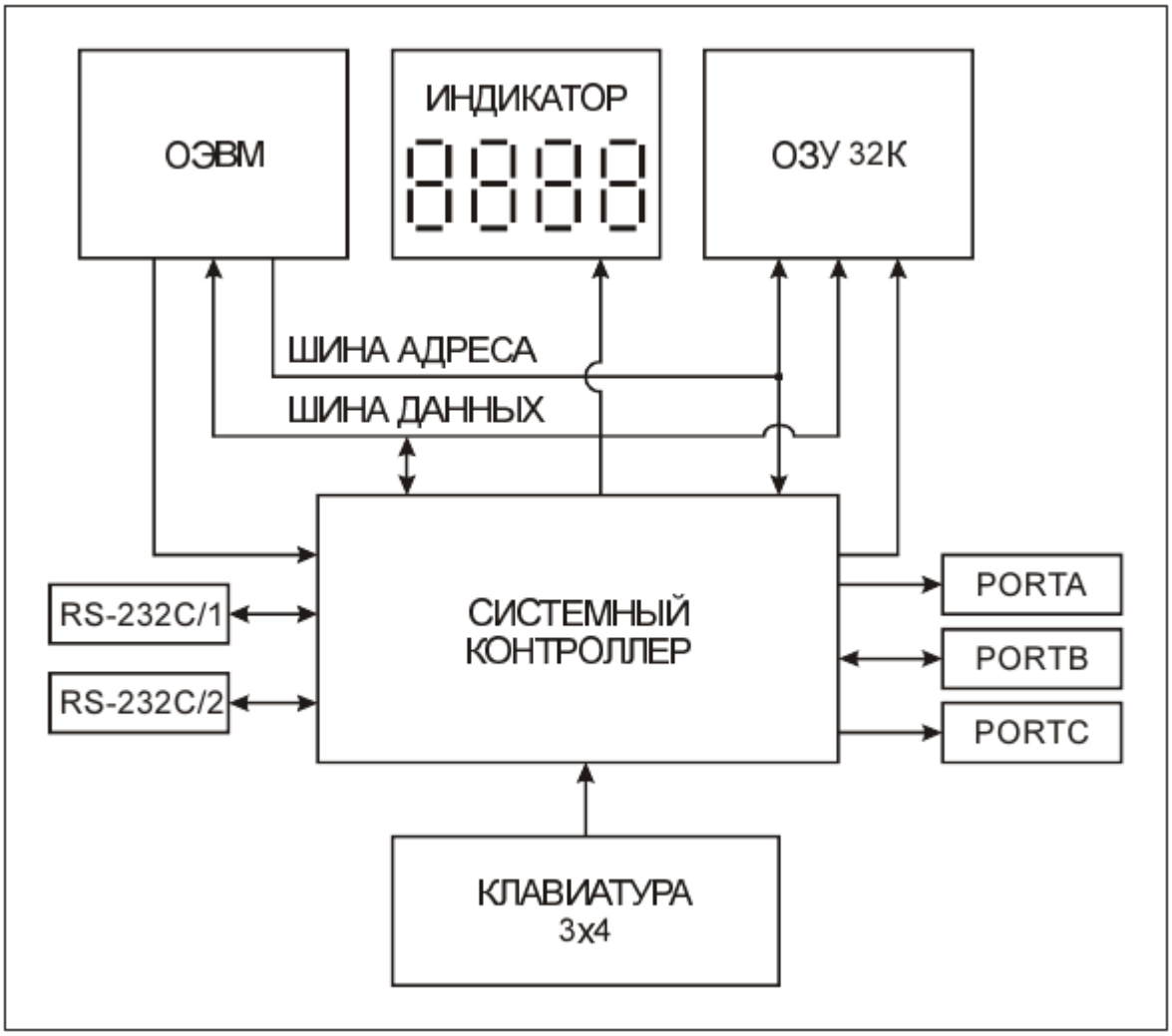


Рисунок 4.2 – Структурная схема стенда

Кнопка SW2 необходима для формирования сигнала сброса на входе RESET микроконтроллера, т.е. перевода стенда в режим загрузки и ожидания приема данных с последовательного порта. Микроконтроллер готов принимать данные в память данных.

Кнопка SW2 необходима для перезапуска программы находящейся в памяти программ (DD3). При нажатии кнопки SW1 загорается светодиод HL9. При этом возможна новая запись программы в стенд с персонального компьютера. При передаче данных с персонального компьютера в стенд, компьютер на линии RI последовательного порта формирует сигнал, который через системный контроллер сбрасывает микроконтроллер так же как и кнопка SW2.

Вся логика стенда реализована на программируемой логической микросхеме EPM7128STC100 (DD4). Системный контроллер управляет режимами работы, выработки управляющих сигналов на ОЗУ, регистры защелки, динамическим светодиодным индикатором, клавиатурой.

4.1 Организация доступа к периферийным устройствам

Адресация (обращение) микроконтроллера к периферийным устройствам стенда реализовано как адресация к ячейкам памяти в адресном пространстве от 8000H до FFFFH. Сигналы выборки периферийных устройств формируются дешифратором адреса внутри микросхемы системного контроллера DD4.

Карта адресации периферийных устройств приведена в таблице 4.1.

Таблица 4.1 – Карта портов ввода/вывода стенда

Адрес	**Цикл**	**B7**	**B6**	**B5**	**B4**	**B3**	**B2**	**B1**	**B0**	**Имя**
Порты периферийных устройств										
8xx0	Запись	[Порт A]								PA_REG
8xx1	Запись	[Порт B]								PB_REG
8xx2	Запись	[Порт C]								PC_REG
ЖКИ										
8xx4	Запись	Регистр команд ЖК индикатора								LCD_CMD
8xx5	Запись	Регистр данных ЖК индикатора								LCD_DATA

Продолжение таблицы 4.1

Адрес	**Цикл**	**B7**	**B6**	**B5**	**B4**	**B3**	**B2**	**B1**	**B0**	**Имя**
Последовательный порт										
9xxx	Чтение	CTS	DSR	DCD	RI	KL3	KL2	KL1	KL0	US_REG
Cxxx	Запись									UC_REG
Индикатор и светодиоды										
Axx0	Запись	[Регистр индикатора 0]								DISPLAY[0]
Axx1	Запись	[Регистр индикатора 1]								DISPLAY[1]
Axx2	Запись	<Зарезервировано>								DISPLAY[2]
Axx3	Запись	<Зарезервировано>								DISPLAY[3]
Axx4	Запись	DP3	DP2	DP1	DP0	BL3	BL2	BL1	BL0	DC_REG
Axx5	Запись	<Зарезервировано>								EDC_REG
Axx6	Запись	LED7	LED6	LED5	LED4	LED3	LED2	LED1	LED0	LED_REG
Управление работой										
Axx7	Запись	X	x	x	x	x	X	x	RUN	SYS_CTL
Совместимые регистры										
Bxx0	Запись	[Регистр индикатора 1]								DISPLAYB
F000	Запись									DAC

Подход с применением системного контроллера позволяет, в теории, использовать 32768 ПУ с 8-битным входом (диапазон адресов 8000H-FFFFH). Без системного контроллера количество выводов микроконтроллера определяет количество управляемых периферийных устройств.

5 Функциональные элементы учебно-отладочного стенда

5.1 Микроконтроллер ATmega8515s

В учебно-отладочном стенде используется микроконтроллер ATmega8515s. Этот микроконтроллер совместим с микроконтроллерами на основе ядра Intel 8051, поэтому EV8031/AVR позволяет устанавливать микроконтроллер AT89C51 после размыкания соответствующей перемычки.

ATmega8515s – это 8-разрядный однокристальный микроконтроллер на базе архитектуры AVR-RISC Atmel, которая обеспечивает очень быстрое выполнение программы. 130 команд, выполняемых за один цикл. Схема имеет 35 контакта ввода-вывода и содержит УСАПП, SPI-интерфейс, контрольный таймер, компаратор, два таймера/счетчика, 8-разрядный и 16-разрядный, три канала ШИМ, JTAG-интерфейс и часы реального времени с отдельным осциллятором. Он имеет функцию сброса при включении питания и три маломощные режимов. Программа хранится в памяти Flash EPROM, объёмом 8 Кб, также микроконтроллер содержит 512 байт памяти данных EEPROM, 512 байт памяти статического ОЗУ и интерфейс подключения внешней памяти.

Интересной особенностью данного микроконтроллера является возможность, кроме программирования через интерфейс SPI, осуществлять самопрограммирование из загрузчика, работающего в ядре AVR. Программа в области загрузчика может продолжать работать во время обновления памяти программ в Flash за счёт реализации в ядре AVR поддержки записи во время чтения.

Эксплуатационные характеристики микроконтроллера таковы:

- напряжение питания: от +4.5 до +5.5 В ;
- активный режим: 11 мА при 8 МГц;
- холостой режим: 5.5 мА при 8 МГц;
- ждущий режим: <1 μA;
- Тактовая частота: 0 − 16 МГц;
- Температурный диапазон: от −40 до +85 °C;

Микроконтроллеры AVR построены на гарвардской архитектуре (программа и данные находятся в разных адресных пространствах) и систему команд, близкую к идеологии RISC. Ядро AVR имеет 32 8-битных регистра общего назначения объединённых в регистровый файл. В отличие от «идеального» RISC, регистры не абсолютно независимы:

- Три «сдвоенных» 16-битных регистра-указателя X (r26:r27), Y (r28:r29) и Z (r30:r31);
- Некоторые команды работают только с регистрами r16…r31;
- Результат умножения (в тех моделях, в которых есть модуль умножения) всегда помещается в r0:r1.

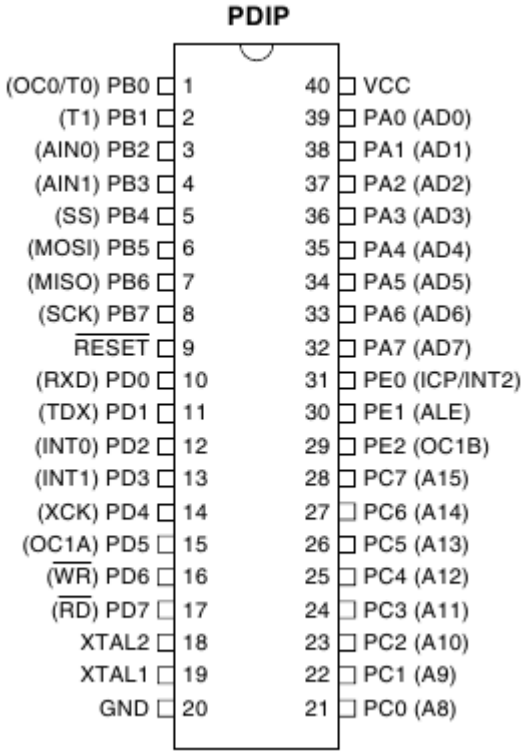


Рисунок 5.1 – Расположение контактов микроконтроллера ATmega8515s

5.2 Последовательный приёмопередатчик

Приёмопередатчик позволяет соединять учебно-отладочный стенд с любым оборудованием, имеющим интерфейс последовательной связи. Также модуль последовательной связи служит для записи в ПЗУ прошивки

микроконтроллера. EV8031/AVR содержит два модуля последовательной связи с интерфейсом RS-232C и один с интерфейсом RS-485.

Модуль последовательной связи сформирован на микросхеме приемника MC1489, передатчика 74HC04, мультиплексора канала передачи (внутри системного контроллера).

Скорость обмена по последовательному порту в режиме загрузки 9600б/с. Скорость обмена по последовательному порту в отлаживаемой программе может быть изменена.

Выбор канала последовательной передачи осуществляется сигналами CFG0, CFG1 по адресу 9001H. Установка этих битов в "логический ноль" включает порт 1, на схеме X11, этот порт имеет неполный набор сигналов (RxD, TxD, RI) и предназначен для записи программы в стенд.

Программная установка сигналов CFG0 в "0", а CFG1 в "1" формирует выборку дополнительного канала последовательной передачи данных, разъем X12. Дополнительный последовательный канал имеет полный набор сигналов интерфейса RS-232C.

5.3 Модуль статической индикации

Модуль статической индикации позволяет отображать все цифры шестнадцатеричной системы счисления. Может использоваться для индикации статуса выполняемой программы, численных значений входных сигналов, а также для изучения работы со статической индикацией в общем.

Четырехразрядный 7-сегментный светодиодный индикатор подключен к системному контроллеру, который автоматически выполняет динамическую регенерацию и декодирование двоичного кода в код семисегментного индикатора. Индикатор работает всегда, сразу после подачи питания. Контроллер индикатора содержит два восьмиразрядных регистра, содержимое которых отображается на индикаторе. Содержимое регистра с адресом 0xA000 отображается на двух левых разрядах, содержимое регистра с адресом

0xA001(0xB000) – на двух правых разрядах в шестнадцатеричной форме. Управление десятичными точками и гашением осуществляется через регистр DC_REG(0xA004). Биты DP3..DP0 управляют десятичными точками. Запись 1 в соответствующий разряд включает десятичную точку. Биты BL3..BL0 управляют гашением разрядов индикатора. Запись 1 в эти биты вызывает гашение соответствующего разряда индикатора.

5.4 Матричная клавиатура

Клавиатура может служить для выбора режима работы программы, ввода пользовательской информации и т.п. Содержит 12 клавиш: цифровые 0-9, а также специальные символы “*” и “#”.

Состояние столбца матрицы клавиатуры считывается из ячейки с базовым адресом 0x9000, биты 3..0. Соответствующий столбец выбирается нулем в разрядах адреса A2..A0. То есть, адрес 0x9006 выбирает первый столбец, адрес 0x9005 – второй столбец, адрес 0x9003 – третий столбец. Признак нажатой кнопки считывается как ноль в соответствующем разряде.

6 Описание плат расширения

Плата расширения (в комплексе с учебно-отладочным стендом) предназначена для проведения лабораторных работ связанных с аналого-цифрового и частотного преобразования, а также с обработкой дискретных сигналов. Структурная схема платы расширения приведена на рисунке

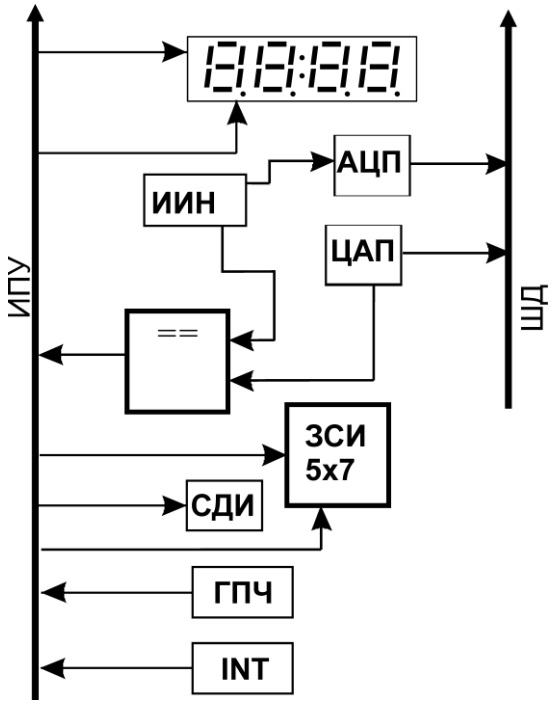


Рисунок 6.1 – Структурная схема платы расширения

8888 – 4 разрядная динамическая индикация;

ИПУ – Интерфейс периферийных устройств;

ЦАП – Цифроаналоговый преобразователь;

СДИ – Светодиодные индикаторы;

ЗСИ – Знакосинтезирующий индикатор 5х7;

ГПЧ – Генератор с изменяемой частотой генерации;

INT – Кнопки запроса прерывания;

ИИН – Источник измеряемого напряжения;

ШД – Шина данных.

Расположение элементов платы расширения показаны на рисунке ниже

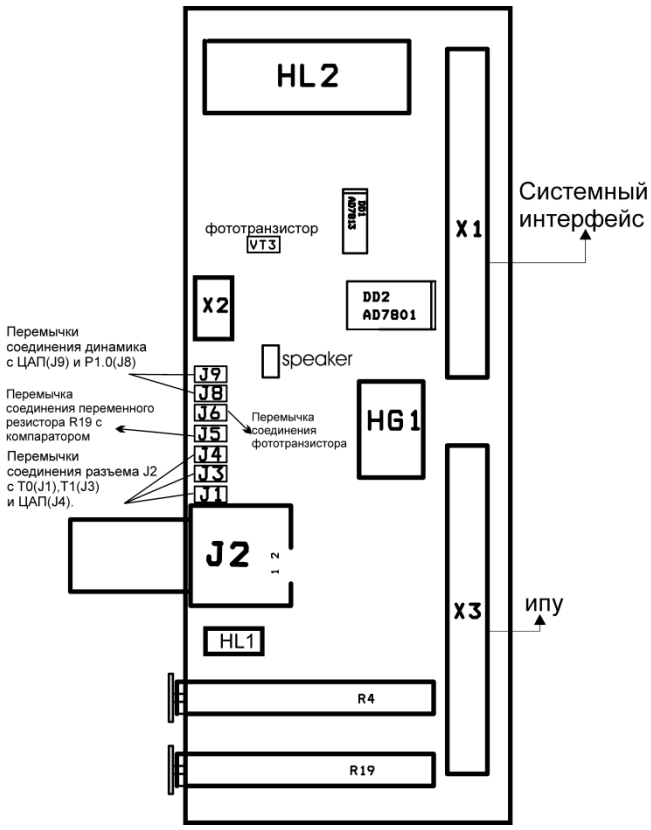


Рисунок 6.2 – Схема расположения элементов платы расширения

HG1 – знакосинтезирующий индикатор 5x7;

HL2 – 4-х разрядная динамическая индикация;

HL1 – светодиодный индикатор срабатывания компаратора;

J1 – перемычка подключения к разъёму J2 выхода генератора с постоянной частотой генерации;

J2 – разъём подключения внешних контрольно-измерительных приборов.

J3 – перемычка подключения к разъёму J2 выхода генератора с изменяемой частотой генерации;

J4 – перемычка подключения к разъёму выхода ЦАП;

J5 – подключение в качестве источника внешнего прерывания INT1 кнопки S11;

J6 – подключение в качестве источника внешнего прерывания INT1 внешнего источника, который может быть подключён через разъём JP1;

J7 – интерфейс подключения платы расширения к стенду;

J8 – подключение ко входу АЦП внешнего источника сигнала, подключенного к разъёму JP2;

J9 – подключение в качестве источника сигнала для АЦП переменного резистора R27;

R19 – переменный резистор, источник входного сигнала для АЦП;

R4 – переменный резистор, изменяет частоту генерации генератора импульсов;

6.1 Модуль цифроаналогового преобразования

Цифроаналоговый преобразователь необходим при подключении учебно-отладочному стенду аналогового оборудования. Подключение выполняется через разъём BNC, имеющийся на плате расширения. Использование ЦАП позволяет студентам научиться выдавать аналоговые сигналы необходимой формы на исполнительные устройства не имеющие цифровых интерфейсов.

ЦАП выполнен на микросхеме AD7801 DD2 (8-разрядный ЦАП). Входными сигналами для ЦАП являются линии AD7-AD0. Выходной сигнал снимается с разъема BNC. К выходу ЦАП может быть подключён компаратор и светодиод для индикации результатов сравнения.

6.2 Модуль аналого-цифрового преобразования

Аналого-цифровой преобразователь служит для подключения аналоговых устройств к учебно-отладочному стенду в качестве устройств ввода. Таким образом, студенты могут изучать методику использования аналоговых датчиков, определять параметры (частоту, амплитуду, фазу) аналоговых сигналов.

АЦП выполнен на микросхеме ЦАП AD7801, операционном усилителе LM358 DA1, используемым в качестве компаратора. Входным аналоговым сигналом для АЦП являются сигнал с переменного резистора R19. Линии AD7-AD0 (см. схему стенда) используются для формирования цифрового входного

кода. На выходе ЦАП формируется напряжение, пропорциональное входному коду. Сигнал срабатывания компаратора снимается с (DA1-2) вход ОЭВМ P1.7. Срабатывание компаратора визуально видно по загоранию светодиода HL1. Если на P1.7 "0" светодиод светится.

В расширенной комплектации стенда поставляется микросхема AD7813 – 8-разрядный АЦП.

6.3 Генераторы

Генераторы с переменной и постоянной частотой могут быть использованы как учебное пособие при изучении методов определения параметров входных аналоговых сигналов (частоту, амплитуду, фазу), определения скважности, среднего уровня сигнала и т.п.

В схеме присутствует генератор с изменяемой частотой генерации ~1-50кГц, элементы R1, R4, R5, R7, R10, R11, R15, R16, C3, VT1, DA1-1 (изменение частоты осуществляется с помощью резистора R4), и генератор с фиксированной частотой ~10кГц, элементы R19, R20, C16, DD18-1, DD18-2, DD18-3.

6.4 Вывод дискретной информации

Вывод дискретной информации осуществляется с помощью четырех разрядного семисегментного индикатора HL2 включенного по схеме динамической индикации. Управление динамической индикацией осуществляется с помощью элементов DD3 (линия данных A, B, C, D, E, F, G, DP, PB0, PB1, PB2, PB3, PB4, PB5, PB6 ,PB7) сигналы поступают с порта PB, сигналы выборки соответствующего индикатора поступают от линий PC0, PC1 порта C.

7 Пакет моделирования Proteus ISIS

Стремительное развитие систем автоматизированного проектирования (САПР) позволило сделать процесс размещения компонентов устройства на печатной плате и разводки соединений практически полностью автоматическим. В такой ситуации большую часть времени при разработке электронного оборудования стало занимать проектирование и тестирование принципиальных электрических схем.

Пакет Proteus ISIS предназначен для разработки электрических принципиальных схем и симуляции их работы в интерактивном режиме. Proteus ISIS позволяет не только создавать и симулировать схемы на основе простейших схемотехнических элементов (источников питания, резисторов, диодов, транзисторов и т.п.), но и использовать в схемах полноценные модели микроконтроллеров. Скорость симуляции приближена к реальному времени и не требует использования внешних программ.

Proteus ISIS предоставляет возможность полного контроля над внешним видом разрабатываемой схемы. Пакет поддерживает изменение толщины соединительных линий, стилей заливки, цветов и шрифтов. К тому же пользователь имеет возможность создавать собственные шаблоны компонентов и переносить их между проектами.

Общие характеристики Proteus ISIS таковы:

- Работает на Windows 98/Me/2k/XP и выше;
- Поддержка автоматической разводки соединений и размещения пересечений;
- Мощные средства для выделения групп объектов и изменения их свойств;
- Полная поддержка шинных соединений, включая объединение контактов микросхем в шинный контакт;
- Создание сметы используемых компонентов;

– Проверка правильности создаваемой схемы (отсутствие коротких замыканий, питания, заземления и т.п.);
– Экспорт списка соединений во все популярные САПР для разработки печатных плат.

Proteus ISIS обеспечивает среду разработки для Proteus VSM, интерактивного низкоуровневого симулятора. Этот продукт сочетает в себе симуляцию электрических цепей смешанного типа (с использованием и цифровых, и аналоговых элементов одновременно), различных моделей микропроцессоров и интерактивных периферийных устройств, что позволяет создавать полноценные микропроцессорные системы. Параметры симуляции, например, частота и отношение времени симуляции к реальному времени, могут быть изменены.

Proteus VSM дополнительно предоставляет следующие возможности:

– Поддержка интерактивного и графического типов симуляции;
– Использование моделей популярных микроконтроллеров, таких как PIC и серии 8051.
– Использование интерактивных моделей ПУ, включая светодиодные и ЖК индикаторы, матричные клавиатуры, терминалы с интерфейсом RS-232, большое количество моделей разнообразных переключателей, светодиодов, ламп накаливания и т.д.;
– Использование виртуальных измерительных инструментов: вольтметров, амперметров, осциллоскопов, анализаторов цифровых сигналов.
– Разработка собственных моделей и компонентов на языке C++ и других языках.

Следует упомянуть, что Proteus ISIS содержит также компиляторы исходных кодов для микроконтроллеров на основе ядра 8051, AVR и PIC и позволяет подключать внешние от сторонних производителей. Есть возможность пошаговой отладки программ с отображением содержимого

регистров микроконтроллеров, внутренней и внешней памяти данных, памяти программ. В режиме отладки все элементы схемы продолжают симуляцию.

Пакет Proteus ISIS содержит большое количество примеров схем с использованием практически всех доступных в его библиотеках компонентов, начиная с простейших аналоговых схем и заканчивая интерактивными шахматами на основе микропроцессора ARM с визуализацией на жидкокристаллическом экране.

8 Моделирование учебно-отладочного стенда

Основная проблема при разработке виртуального стенда – наличие значительного количества периферийных устройств (ПУ) при ограниченном количестве выводов микроконтроллера. В стенде EA8031/AVR эта проблема решается микросхемой DD4, выполняющей функции маршрутизатора управляющих сигналов и системного контроллера. Обращение микроконтроллера к периферийному устройству происходит, как обращение к определенному адресу внешней памяти. Адреса памяти, через которые можно получить доступ к периферийным устройствам, собраны в таблице 4.1.

Все периферийные устройства соединены с общей шиной данных. При обращении к соответствующему адресу внешней памяти регистр перед ПУ защёлкивает значение на шине данных.

Микроконтроллер ATmega8515S способен адресовать 64К байт внешней памяти. Шина данных микроконтроллера 8-битная, а для адресации всей внешней памяти необходимо 16 бит, поэтому необходимо использование защелки по схеме приведенной ниже.

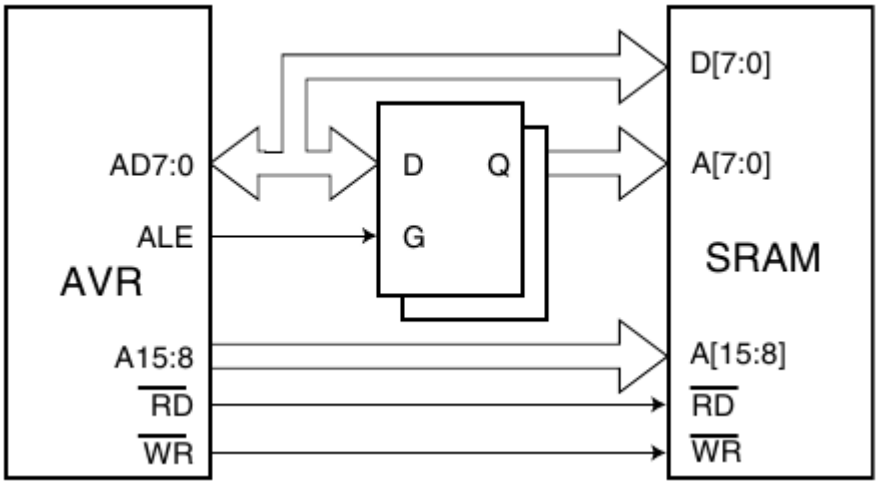


Рисунок 8.1 – Адресация внешней памяти ATmega8515S

В стенде EV8031/AVR вместо внешней памяти к микроконтроллеру подключен системный контроллер DD4. Системный контроллер формирует сигнал на линии защелкивающей значение шины данных в регистр перед ПУ.

Так как обращение ко внешней памяти происходит за три такта, системным контроллером может выступать другой микроконтроллер работающий на большей частоте либо логическая схема.

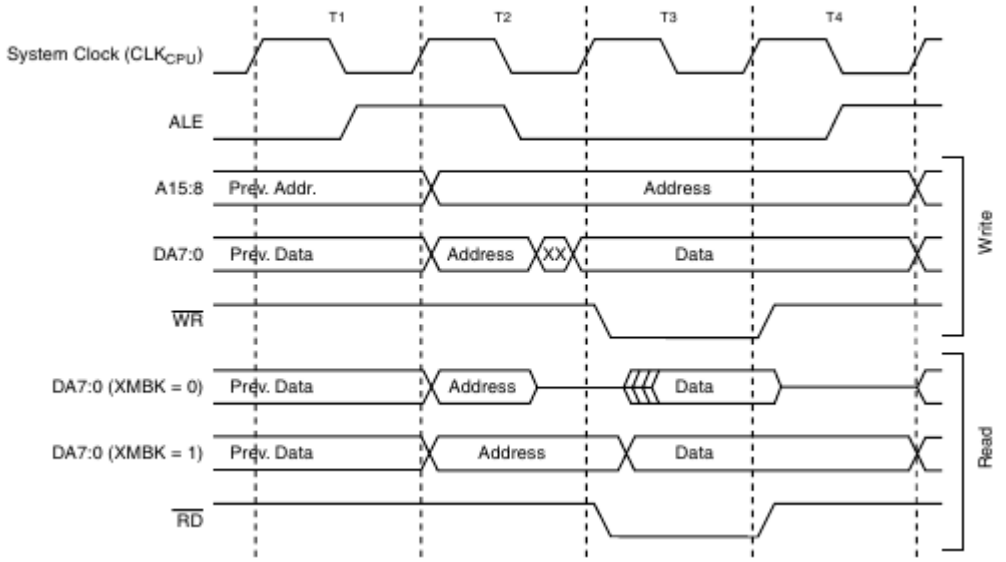


Рисунок 8.2 – Временная диаграмма обращения к внешней памяти ATmega8515S

В качестве средства моделирования применяется пакет Proteus ISIS. Proteus не позволяет создать модель с использованием двух микроконтроллеров работающих на разных тактовых частотах, т.к. это вызывает слишком большую нагрузку на ЦПУ. Поэтому необходимо использовать логическую схему.

Создание логической схемы из простейших элементов излишне перегружает дизайн, поэтому был использован подход разработчика стенда – использовать ПЛИС в качестве системного контроллера.

8.1 Программируемые логические интегральные схемы

Программируемая логическая интегральная схема (ПЛИС, англ. programmable logic device, PLD) — электронный компонент, используемый для создания цифровых интегральных схем. В отличие от обычных цифровых микросхем, логика работы ПЛИС не определяется при изготовлении, а задаётся посредством программирования (проектирования). Для программирования используются программаторы и отладочные среды, позволяющие задать желаемую структуру цифрового устройства в виде принципиальной электрической схемы или программы на специальных языках описания аппаратуры: Verilog, VHDL, AHDL и др. Альтернативой ПЛИС являются: программируемые логические контроллеры (ПЛК), базовые матричные кристаллы (БМК), требующие заводского производственного процесса для программирования; ASIC — специализированные заказные большие интегральные схемы (БИС), которые при мелкосерийном и единичном производстве существенно дороже; специализированные компьютеры, процессоры (например, цифровой сигнальный процессор) или микроконтроллеры, которые из-за программного способа реализации алгоритмов в работе медленнее ПЛИС.

Некоторые производители ПЛИС предлагают программные процессоры для своих ПЛИС, которые могут быть модифицированы под конкретную задачу, а затем встроены в ПЛИС. Тем самым обеспечивается уменьшение места на печатной плате и упрощение проектирования самой ПЛИС, за счёт быстродействия.

Существует несколько типов ПЛИС: PAL, GAL, CPLD, FPGA. В разработанном виртуальном стенде используется ПЛИС типа CPLD, рассмотрим его.

CPLD (англ. complex programmable logic device — сложные программируемые логические устройства) содержат относительно крупные программируемые логические блоки — макроячейки, соединённые с внешними

выводами и внутренними шинами. Функциональность CPLD кодируется в энергонезависимой памяти, поэтому нет необходимости их перепрограммировать при включении. Может применяться для расширения числа входов/выходов рядом с большими кристаллами, или для предобработки сигналов (например, контроллер COM-порта, USB, VGA).

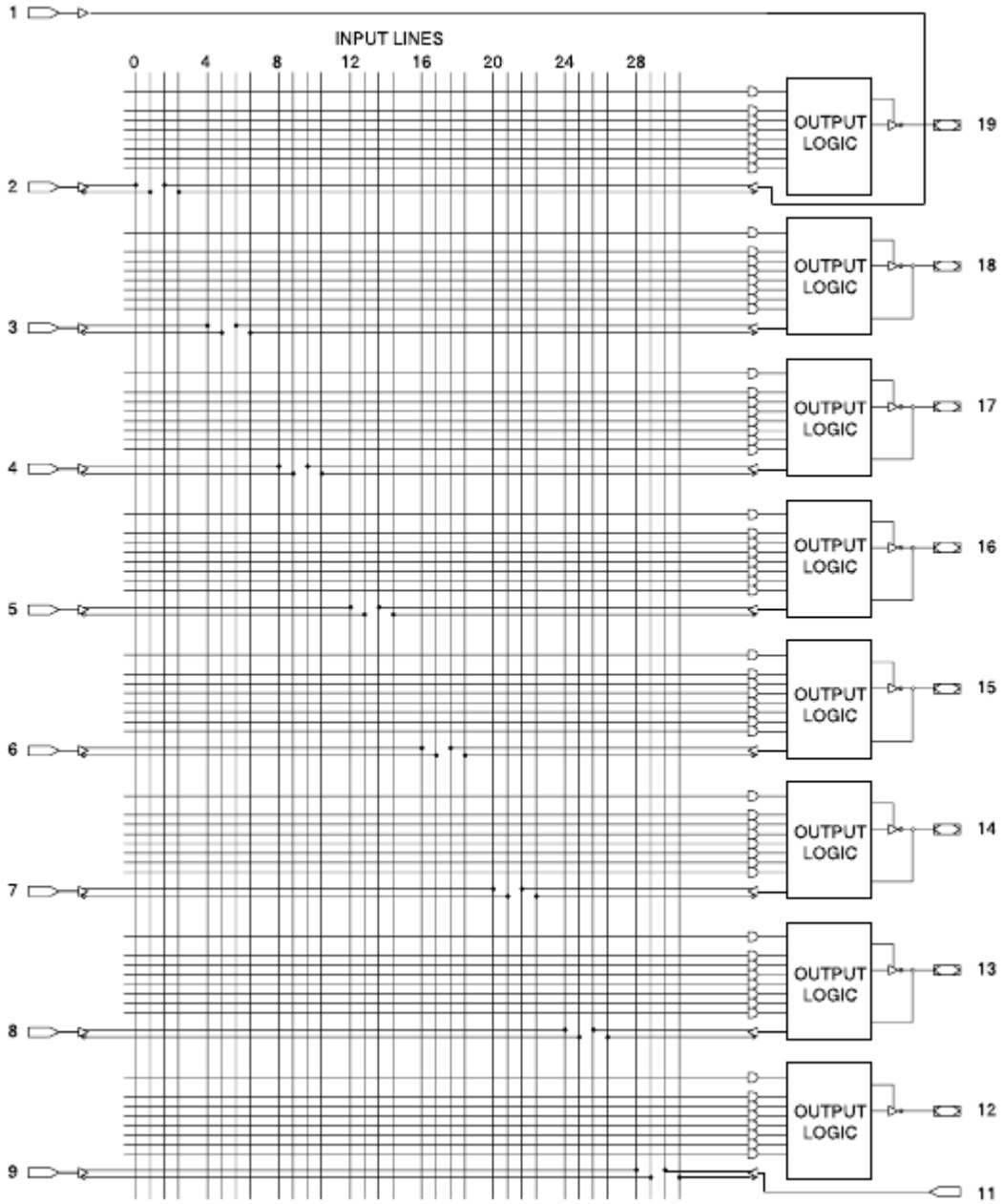


Рисунок 8.3 – Логическая структура ПЛИС ATF16V8C

Главным отличием между большим CPLD и малым FPGA до недавних пор было наличие внутренней энергонезависимой конфигурационной памяти в CPLD. Это отличие становится уже не столь значимым, поскольку ряд

последних моделей FPGA также включают такую внутреннюю память. Тем не менее, наличие такой внутренней энергонезависимой конфигурационной памяти, наряду с такой важной характеристикой, как устойчивость показателей, делают CPLD незаменимыми для современных цифровых схем в качестве устройства для начальной «загрузки ОС» схемы, перед тем, как передать управление другим микросхемам, не обладающим такой способностью. В качестве примера можно привести использование CPLD для загрузки данных конфигурации FPGA от энергонезависимой памяти.

8.1.1 Языки программирования ПЛИС

В начале существования микросхем программируемых логических устройств (ПЛУ) спецификация на них представляла собой принципиальную схему или схему конечного автомата. Позже конструкция устройства, представленная в графическом виде, была вручную преобразована в табличный эквивалент и затем сохранена в текстовом файле. Этот текстовый файл, кроме всего прочего, определял, какие плавкие перемычки должны быть подвержены пережиганию или какие наращиваемые перемычки должны быть созданы. В те времена текстовый файл напрямую вводился в специальное устройство, называемое программатором, который использовался для программирования микросхем. В дальнейшем файл начали создавать на компьютере, с помощью которого производилась загрузка и управление программатором.

При создании файла от инженера требовались знание внутренних связей устройства и формата файла, записываемого в программатор. Каждый производитель ПЛУ развивал собственные форматы файлов, которые обычно работали со своими типами устройств. Всем заинтересованным было очевидно, что такой порядок проектирования был трудоёмким, чреват ошибками и, к тому же, затруднял их поиск и исправление.

В 1980 году комиссия Объединенного инженерного совета по электронным устройствам (JEDEC), подразделение Ассоциации электронной

промышленности США, предложила стандарт форматов текстовых файлов для программирования ПЛУ. Это случилось незадолго до того, как программисты электронных устройств уже были готовы принять этот формат.

Примерно в это же время Джон Беркнер (John Birkner), человек, который разработал первые устройства PAL и стоял во главе их дальнейшего развития, создал PAL Ассемблер (PALASM). PAL Ассемблер сочетает в себе язык описания аппаратных средств (HDL – Hardware Description Language) и прикладное программное обеспечение. Как язык описания аппаратных средств PAL Ассемблер даёт инженерам-разработчикам точно описать функции принципиальной схемы в текстовом файле с исходными кодами. Файл состоял из булевых уравнений, записанных в формате «сумма-произведений». Как прикладная программа, которая на языке Фортран занимала всего лишь 6 страниц (сегодня её бы называли САПР электронных устройств) PALASM читал текстовый файл с исходными кодами и автоматически генерировал текстовый программный файл, пригодный для программирования устройства.

Появление PALASM, несомненно, было эпохальным событием для того времени, но его первоначальные версии поддерживали только устройства компании MMI (Monolithic Memories Inc.), и не поддерживали какие-либо виды минимизации или оптимизации. Для решения этих проблем в 1983 году компания Data I/O представила язык ABEL (Advanced Boolean Expression Language – расширенный язык булевых выражений). Примерно в это же время компания Assisted Technology обнародовала пакет CUPL (Common Universal tool for Programmable Logic – общие универсальные средства проектирования для программируемой логики). Оба продукта сочетали в себе язык HDL и программные приложения. Кроме того, они поддерживали конструкции конечных автоматов и алгоритмы автоматической минимизации логики, они позволяли работать с многочисленными типами ПЛУ разных производителей.

Кроме PALASM, ABEL и CUPL, которые были, несомненно, лучшими из ранних версий языка HDL, существовало много других версий, таких как AMAZE (Automated Map and Zap of Equation – автоматическое преобразование и

исправление выражений) компании Signetics. Эти простые языки и сопутствующие им средства проектирования прокладывали путь для высокоуровневых версий языка HDL, таких как Verilog или VHDL, и средств проектирования, таких как логический синтез, которые в настоящее время используются для проектирования современных заказных микросхем и устройств с использованием ПЛИС.

8.2 Разработка прошивки системного контроллера

Адресация ПУ в стенде EV8031/AVR происходит как обращение к ячейкам внешней памяти, при этом, как видно из таблицы 4.1, значение имеют только старшие четыре бита адреса и младшие три бита адреса. Остальные 8 бит могут быть произвольными.

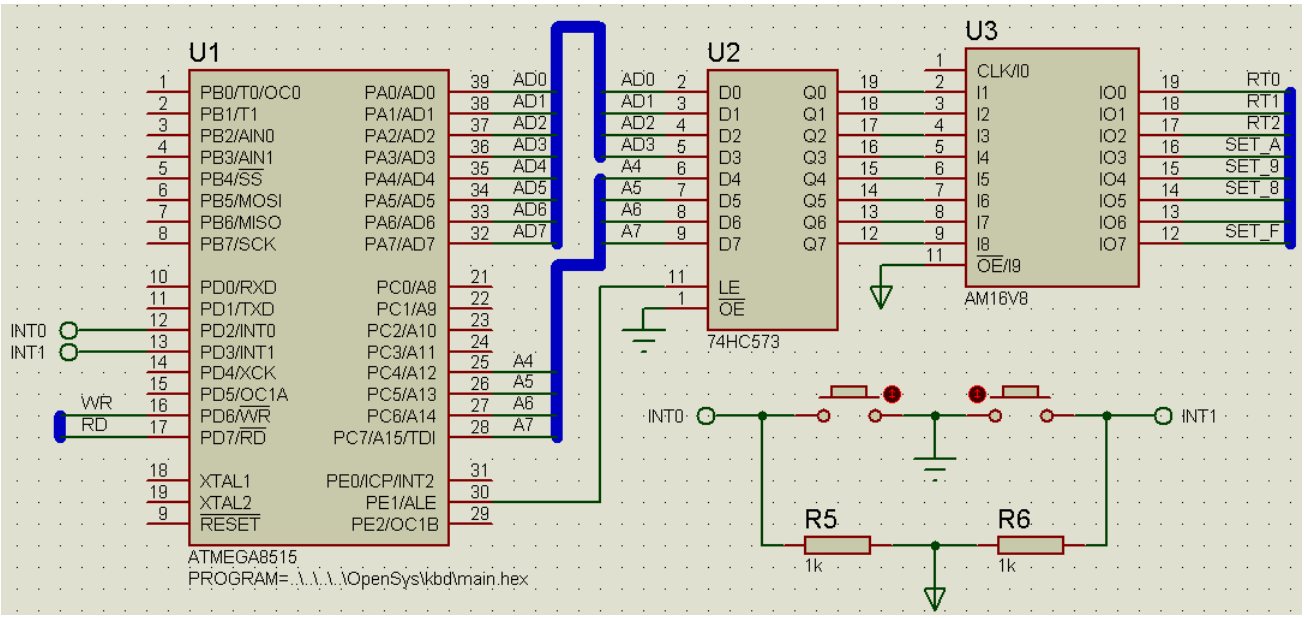


Рисунок 8.4 – Схема подключения системного контроллера

При разработке прошивки ПЛИС был использован следующий подход: всё адресное пространство периферийных устройств было разделено на блоки по 256 байт (или по 8 байт, т.к. значимы только младшие 3 бита адреса), адресуемые старшими четырьмя битами адреса. При обращении к такому блоку активируется унитарный дешифратор, преобразующий двоичный код в сигнал на линии, соответствующей входному значению. На вход дешифратора подаются младшие три бита адреса, которые передаются на общую шину. Таким образом, выходной сигнал дешифратора защёлкивает значение шины данных в регистр перед ПУ.

Прошивка ПЛИС передаёт три младших бита со входа напрямую на выход, и устанавливает логическую единицу на выходе, соответствующем старшим четырём битам входа. Так как используется ПЛИС ATF16V8C, у которой 8 входов и 8 выходов, возможно адресовать до 40 периферийных устройств.

Программа реализована на языке программирования ПЛИС – CUPL и компилируется с помощью компилятора WinCUPL. Алгоритм программы представлен в приложении А, а листинг – в приложении Б.

8.2.1 ПЛИС ATF16V8C

ATF16V8C – это высокопроизводительное программируемое логическое устройство, основанное на электрически стираемой комплементарной логике на транзисторах металл-оксид-полупроводник (EECMOS). Скорость срабатывания не превышает 5 нс, а потребление энергии в энергосберегающем режиме, контролируемом отдельным выводом, составляет около 100 мкА.

ПЛИС ATF16V8C включает в себя расширение общей архитектуры, что позволяет прямую замену устройств семейства 16R8 и большинства 20-контактных ПЛУ. Каждому из 8 выводов присвоен сомножитель. Три режима функционирования настраиваются программно и позволяют реализовать логические функции высокой сложности.

Использование ПЛИС ATF16V8C позволяет значительно снизить общее потребление энергии системы, таким образом, повышая надежность и снижая стоимость обеспечения питанием. Если контакт 4 настроен для контроля режима энергосбережения, то ток питания можно снизить до значений менее 100 мкА при подаче логической единицы на этот вход. В случае, когда режим энергосбережения не требуется, контакт 4 может быть использован как логический вход. Особая конструкция цепей поддержки входов убирает потребность в подтягивающих резисторах с сопутствующим им энергопотреблением.

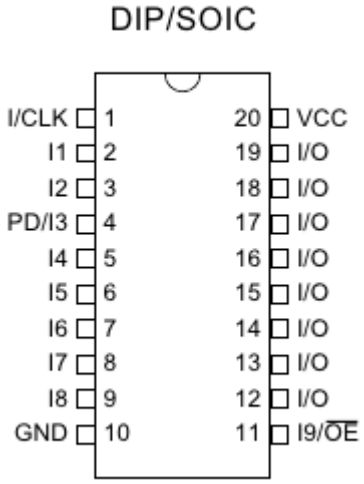


Рисунок 8.5 – Расположение контактов AT16FV8C

ПЛУ выдерживает до 100 циклов стирания/записи и содержит встроенную систему антистатической защиты. Срок хранения прошивки – до 20 лет. Иммунитет к перепадам силы тока до 100 мА.

8.2.2 Унитарный дешифратор/демультиплексор SN74LS137

Дешифраторы/демультиплексоры серии *LS137 содержат три адресных входа, восемь выходов и могут работать в режиме защёлки. При подаче на вход активации защёлки ( ) низкого уровня напряжения, устройство действует как дешифратор/демультиплексор. При перепаде напряжения на с низкого уровня в высокий, значение адресных входов (A, B и C) защёлкивается.

Дальнейшие изменения адреса игнорируются, пока уровень напряжения на остаётся высоким. Контакты управления выходами, и , позволяют контролировать выходы независимо от адресных входов и входа активации защёлки. Уровень напряжения на выходе остаётся высоким, пока на подаётся низкое напряжения, а на – высокое. Серия дешифраторов/демультиплексоров *LS137 идеально подходит для реализации помехоустойчивых декодеров в системах использующих шины. Таблица вариантов управления представлена ниже.

Таблица 8.1 – Таблица значений входов и выходов дешифраторов серии *LS137

Входы						**Выходы**							
Управления			**Адресные**										
!GL	G1	!G2	C	B	A	Y0	Y1	Y2	Y3	Y4	Y5	Y6	Y7
x	x	B	x	x	x	B	B	B	B	B	B	B	B
x	H	x	x	x	x	B	B	B	B	B	B	B	B
H	B	H	H	H	H	H	B	B	B	B	B	B	B
H	B	H	H	H	B	B	H	B	B	B	B	B	B
H	B	H	H	B	H	B	B	H	B	B	B	B	B
H	B	H	H	B	B	B	B	B	H	B	B	B	B
H	B	H	B	H	H	B	B	B	B	H	B	B	B
H	B	H	B	H	B	B	B	B	B	B	H	B	B
H	B	H	B	B	H	B	B	B	B	B	B	H	B
H	B	H	B	B	B	B	B	B	B	B	B	B	H
B	B	H	x	x	x	Выходы соответствуют защёлкнутому адресу							

B – высокий уровень напряжения

H – низкий уровень напряжения

x – уровень напряжения не имеет значения

8.3 Моделирование дискретной индикации

Дискретная индикация моделируется блоком светодиодов с общим катодом. Пакет Proteus содержит лишь модель блока светодиодов из 10 штук, поэтому два не используются.

Так как светодиоды собраны в схему с общим катодом, подача на вход одного из них высокого уровня активирует светодиод. Высокий уровень напряжения создаётся логической единицей на выходе регистра стоящего перед блоком светодиодов. Защёлкивание в него значения происходит при обращении к ячейке внешней памяти с адресом Axx6H.

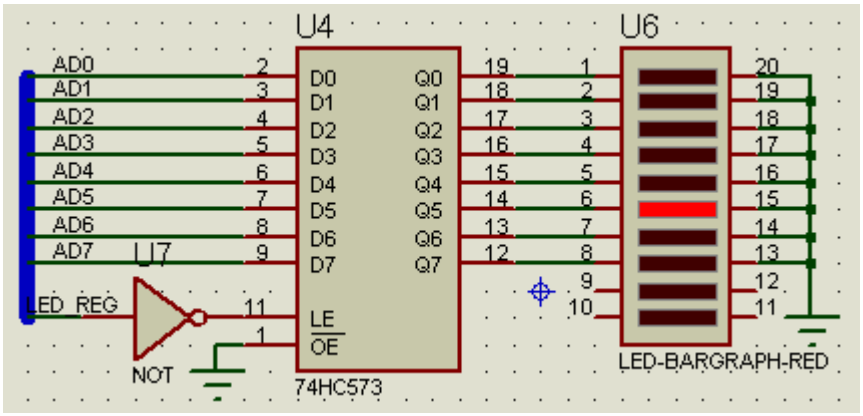


Рисунок 8.6 – Блок дискретной индикации

8.4 Моделирование статической индикации

В стенде EV8031/AVR статическая индикация реализована как динамическая, управляемая системным контроллером. В виртуальном стенде статическая индикация реализована с помощью модели 7-сегментного индикатора со встроенным дешифратором.

К регистрам, соответствующим ячейкам внешней памяти Axx0H и Axx1H, подключено по две модели 7-сегментного индикатора. В виртуальном стенде не реализовано управление включением и выключением индикаторов и десятичными точками, т.к. для этого требуется включение дополнительного

микроконтроллера для управления динамической индикацией. Дополнительный микроконтроллер значительно повысит загрузку ЦПУ во время процесса симуляции и приведет к снижению интерактивности виртуального стенда.

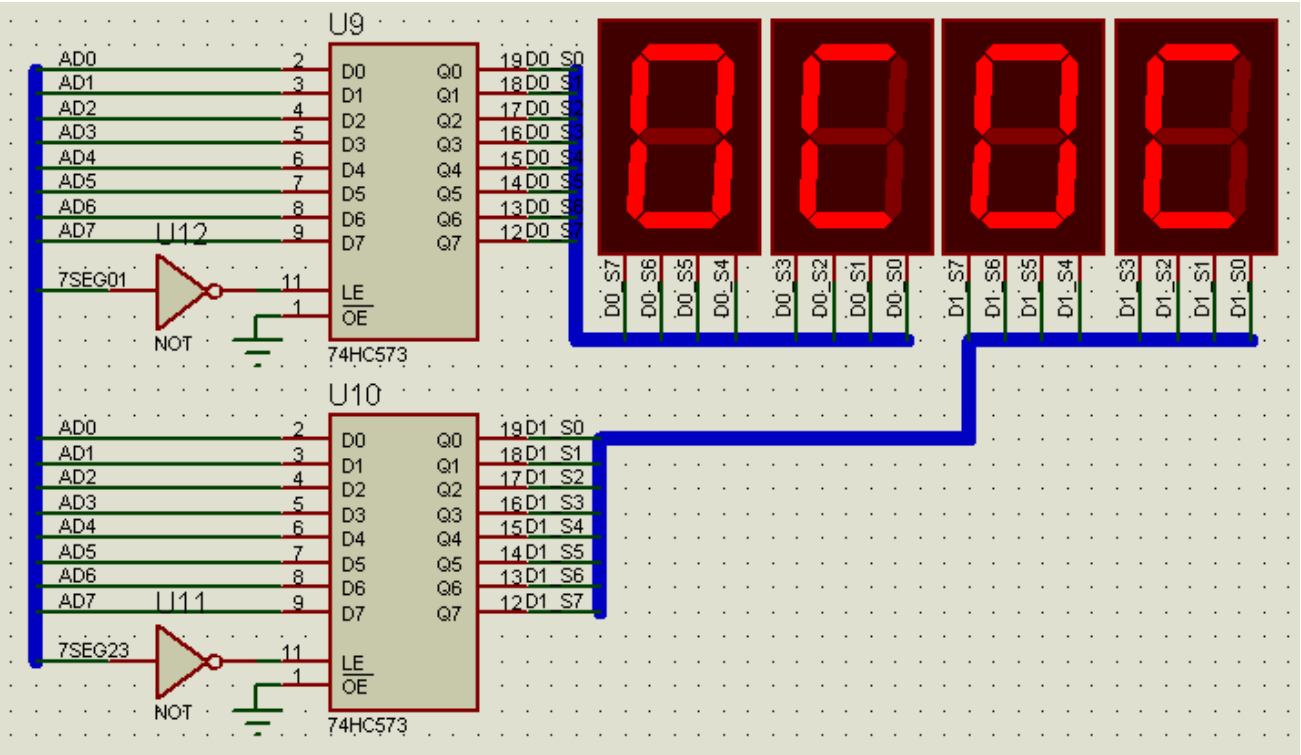


Рисунок 8.7 – Статическая индикация

7-сегментный индикатор, как говорит его название, состоит из семи элементов индикации (сегментов), включающихся и выключающихся по отдельности. Включая их в разных комбинациях, из них можно составить упрощённые изображения арабских цифр. Часто 7-сегментные индикаторы делают в курсивном начертании, что повышает читаемость.

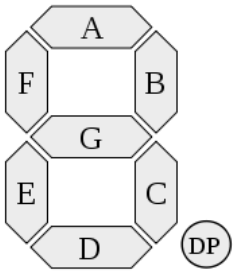


Рисунок 8.8 – Обозначение сегментов индикатора

Ниже представлена таблица значений, которые необходимо подавать на входы 7-сегментного индикатора, чтобы получить определённый символ.

Таблица 8.2 – Значение входов 7-сегментного индикатора

Символ	**Слово данных**	**a**	**b**	**c**	**d**	**e**	**f**	**g**
0	7EH	1	1	1	1	1	1	0
1	30H	0	1	1	0	0	0	0
2	6DH	1	1	0	1	1	0	1
3	79H	1	1	1	1	0	0	1
4	33H	0	1	1	0	0	1	1
5	5BH	1	0	1	1	0	1	1
6	5FH	1	0	1	1	1	1	1
7	70H	1	1	1	0	0	0	0
8	7FH	1	1	1	1	1	1	1
9	7BH	1	1	1	1	0	1	1
A	77H	1	1	1	0	1	1	1
B	1FH	0	0	1	1	1	1	1
C	4EH	1	0	0	1	1	1	0
D	3DH	0	1	1	1	1	0	1
E	4FH	1	0	0	1	1	1	1
F	47H	1	0	0	0	1	1	1

В пакете Proteus дешифратор встроен в модель 7-сегментного индикатора, поэтому достаточно подавать на его входы (4 входа) значение, которое необходимо отобразить. Таблицу 7.1 необходимо использовать при реализации динамической индикации. Модель 7-сегментного индикатора не содержит десятичной точки, в отличии от блока статической индикации EV8031/AVR, поэтому моделирование этого блока выполнено не с полным повторением функционала.

8.5 Моделирование матричной клавиатуры

При небольшом количестве кнопок и равном или большем количестве выводов МК, кнопки подключаются непосредственно к входам. На Рисунок 8.9 – схема подключения кнопки, при которой в ненажатом состоянии на вход подаётся логическая единица (напряжение 5 вольт относительно общего провода), а в нажатом - логический ноль (уровень напряжение 0 вольт относительно общего провода). Резистор "подтягивает" вход МК к единице. Это делается для того, чтобы избежать т.н. третьего состояния (состояние "Z" или просто "обрыв") на входе. При Z на входах возникают помехи – очень короткие импульсы тока, которые могут повлиять на работу системы. Помехи возникают от статического электричества, от прикосновения пальцами к проводникам, от работающих поблизости приборов. Подтяжка работает так: в ненажатом состоянии сопротивление между нулём и входом очень велико, и через резистор на входе создаётся потенциал, воспринимаемый МК как лог. 1. При нажатии картина меняется – теперь резистор – относительно бесконечное сопротивление, а на выходе – потенциал нуля.

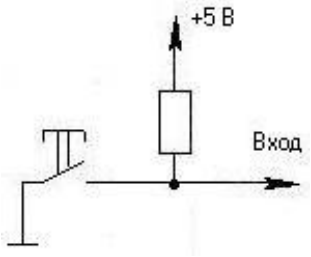


Рисунок 8.9 – Схема подключения кнопки к микроконтроллеру

Матрица кнопок позволяет использовать кнопок вдвое больше количества доступных пинов портов(если матрица квадратная). На рисунке 2 изображена матрица для клавиатуры из 11 кнопок, взятая из реального устройства. Для работы с ней используется 7 контактов порта B. Четыре контакта (PB0-PB3) программно сконфигурированы как выходы и три контакта (PB4-PB6) - как входы.

Принцип опроса матрицы таков: группы кнопок условно разбиты на "линейки" и "колонки". Сначала программно на выходах PB1-PB3 выставляются единицы, а на PB0 –ноль. При этом включена первая колонка. Если сейчас нажать в любом сочетании кнопки этой колонки, то на входах сформируется трёхбитный код, который программа сохраняет в массиве. Затем первая колонка отключается, и подключается следующая, и т.д.

Также полезно избавляться от дребезга контактов. Это не требуется тогда, когда нужно, к примеру, просто зажечь светодиод. Но необходимо, если считанный сигнал МК использует для управления какой-либо логикой. Дребезг - это механическое отскакивание металлических контактов при замыкании, при этом формируется пачка коротких импульсов, что не есть хорошо. Для формирования "чистого" фронта из аппаратных средств чаще всего используются повторители с гистерезисом. Если время исполнения программы некритично, то от дребезга можно избавиться программно

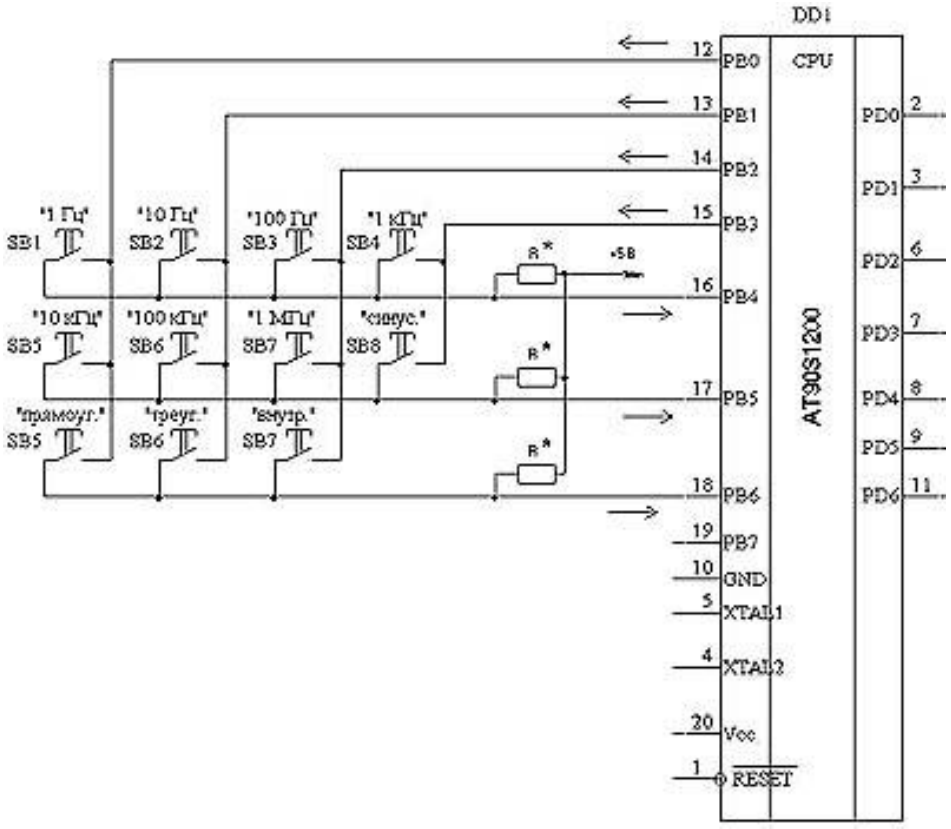
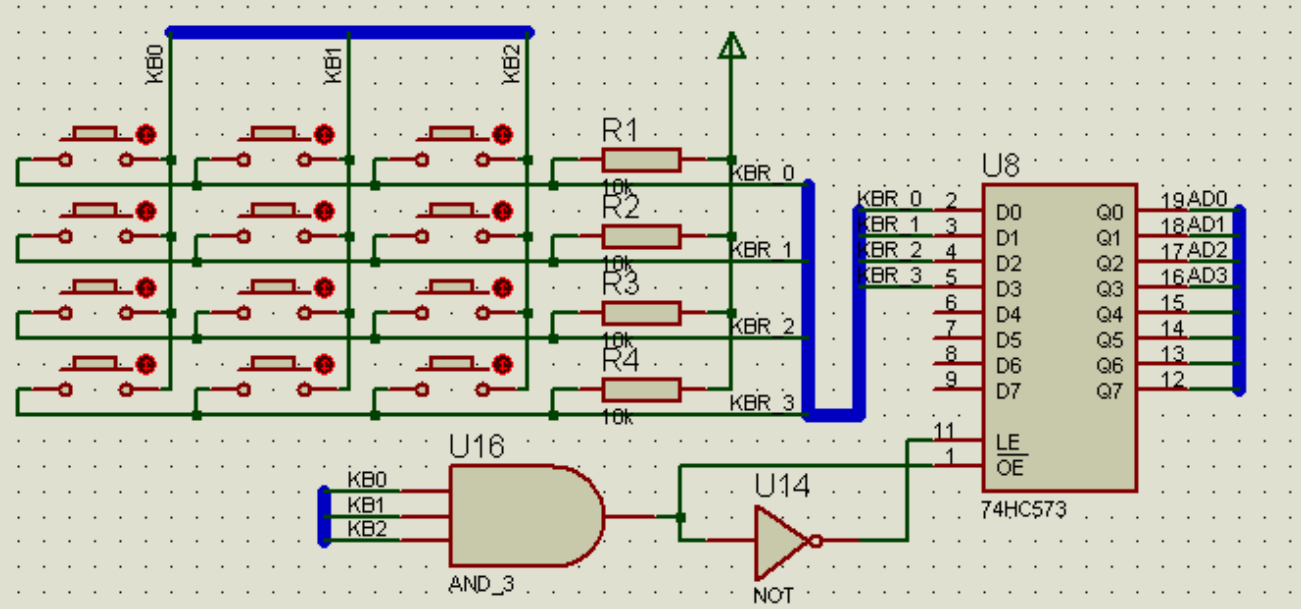


Рисунок 8.10 – Подключение матричной клавиатуры

По приведенной выше описанию реализовано моделирование матричной клавиатуры. При считывании внешней памяти по адресу 0x9006 определяется состояние первого столбца, адреса 0x9005 – второго, адреса 0x9003 – третьего столбца.



8.11 – Матричная клавиатура виртуального стенда

8.6 Блок цифроаналогового преобразования

Цифро-аналоговый преобразователь (ЦАП) — устройство для преобразования цифрового (обычно двоичного) кода в аналоговый сигнал (ток, напряжение или заряд). Цифро-аналоговые преобразователи являются интерфейсом между дискретным цифровым миром и аналоговыми сигналами.

Наиболее общие типы электронных ЦАП:

– широтно-импульсный модулятор — простейший тип ЦАП. Стабильный источник тока или напряжения периодически включается на время, пропорциональное преобразуемому цифровому коду, далее полученная импульсная последовательность фильтруется аналоговым фильтром низких частот. Такой способ часто используется для управления

скоростью электромоторов, а также становится популярным в Hi-Fi (класс аппаратуры) аудиотехнике;

– ЦАП передискретизации, такие как дельта-сигма ЦАП, основаны на изменяемой плотности импульсов. Передискретизация позволяет использовать ЦАП с меньшей разрядностью для достижения большей разрядности итогового преобразования; часто дельта-сигма ЦАП строится на основе простейшего однобитного ЦАП, который является практически линейным. На ЦАП малой разрядности поступает импульсный сигнал с модулированной плотностью импульсов (с постоянной длительностью импульса, но с изменяемой скважностью), создаваемый с использованием отрицательной обратной связи. Отрицательная обратная связь выступает в роли фильтра высоких частот для шума квантования. Большинство ЦАП большой разрядности (более 16 бит) построены на этом принципе вследствие его высокой линейности и низкой стоимости. Быстродействие дельта-сигма ЦАП достигает сотни тысяч отсчетов в секунду, разрядность — до 24 бит. Для генерации сигнала с модулированной плотностью импульсов может быть использован простой дельта-сигма модулятор первого порядка или более высокого порядка как MASH (англ. Multi stage noise SHaping). С увеличением частоты передискретизации смягчаются требования, предъявляемые к выходному фильтру низких частот и улучшается подавление шума квантования;

– ЦАП взвешивающего типа, в котором каждому биту преобразуемого двоичного кода соответствует резистор или источник тока, подключенный на общую точку суммирования. Сила тока источника (проводимость резистора) пропорциональна весу бита, которому он соответствует. Таким образом, все ненулевые биты кода суммируются с весом. Взвешивающий метод один из самых быстрых, но ему свойственна низкая точность из-за необходимости наличия набора множества различных прецизионных источников или резисторов. По этой причине взвешивающие ЦАП имеют разрядность не более восьми бит;

– ЦАП лестничного типа (цепная R-2R схема). В R-2R ЦАП значения создаются в специальной схеме, состоящей из резисторов с сопротивлениями R и 2R. Это позволяет существенно улучшить точность по сравнению с обычным взвешивающим ЦАП, так как сравнительно просто изготовить набор прецизионных элементов с одинаковыми параметрами. ЦАП типа R-2R позволяют отодвинуть ограничения по разрядности. С лазерной подгонкой резисторов на одной подложке достигается точность 20-22 бита. Основное время на преобразование тратится в операционном усилителе, поэтому он должен иметь максимальное быстродействие. Быстродействие ЦАП единицы микросекунд и ниже (т.е. наносекунды);

В стенде EV8031/AVR применяется ЦАП последнего типа. Рассмотрим его более подробно.

8.6.1 ЦАП на матрице R-2R с суммированием напряжений

ЦАП с суммированием напряжений, использует обратное включение входа и выхода матрицы R-2R. На входы a_0, a_1,a_2,…,a_{n-1} подают цифровые сигналы, которые отвечают значению i-го разряда входного двоичного кода. Если на входе i-го разряда присутствует логическая 1, то соответствующий ключ КЛ переключается в верхнее положение и опорное напряжение $U_{оп}$ через резисторы матрицы R-2R с определённым коэффициентом деления подаётся на неинвертирующий вход операционного усилителя (ОУ) DA1, где происходит суммирование напряжений.

Если на вход i-го разряда поступает логический 0, то ключ переключается в нижнее положение, и данная ветка матрицы R-2R подключается к общей шине. Так как матрица резисторов является линейной цепью, её работу можно проанализировать методом суперпозиции, то есть вклад в выходное напряжение от каждого источника (разряда) рассчитывается независимо друг от друга. Вклады от каждого разряда суммируются на неинвертирующем входе ОУ и на

выходе получаем результат в виде напряжения.

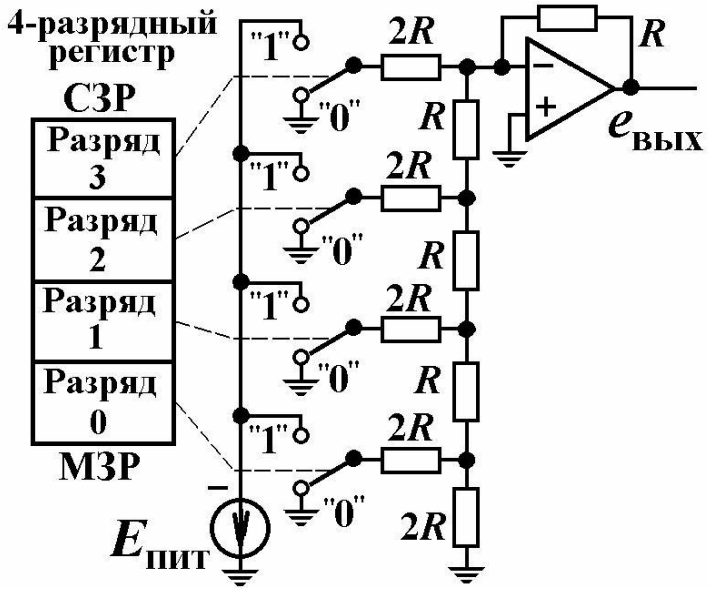


Рисунок 8.12 – ЦАП на матрице R-2R с суммированием напряжений

Выходное напряжение ЦАП на матрице R-2R с суммированием напряжений можно подсчитать по следующей формуле:

$$U_{вих} = \frac{U_{оп}}{2^n} \sum_{i=0}^{n-1} a_i \cdot 2^i \qquad (8.1)$$

Сомножитель $\sum_{i=0}^{n-1} a_i \cdot 2^i$ является десятичным эквивалентом входного двоичного кода (представляет десятичное значение входного цифрового кода).

Выход ЦАП соединен со входом операционного усилителя, работающего как компаратор. Если напряжение на выходе ЦАП превышает напряжение опорное напряжение, регулируемое потенциометром, то загорается светодиод. Принцип работы операционного усилителя рассмотрен ниже.

8.6.2 Операционные усилители

Операционный усилитель (ОУ) — усилитель постоянного тока с дифференциальным входом и, как правило, единственным выходом, имеющий

высокий коэффициент усиления. ОУ почти всегда используются в схемах с глубокой отрицательной обратной связью, которая, благодаря высокому коэффициенту усиления ОУ, полностью определяет коэффициент передачи полученной схемы.

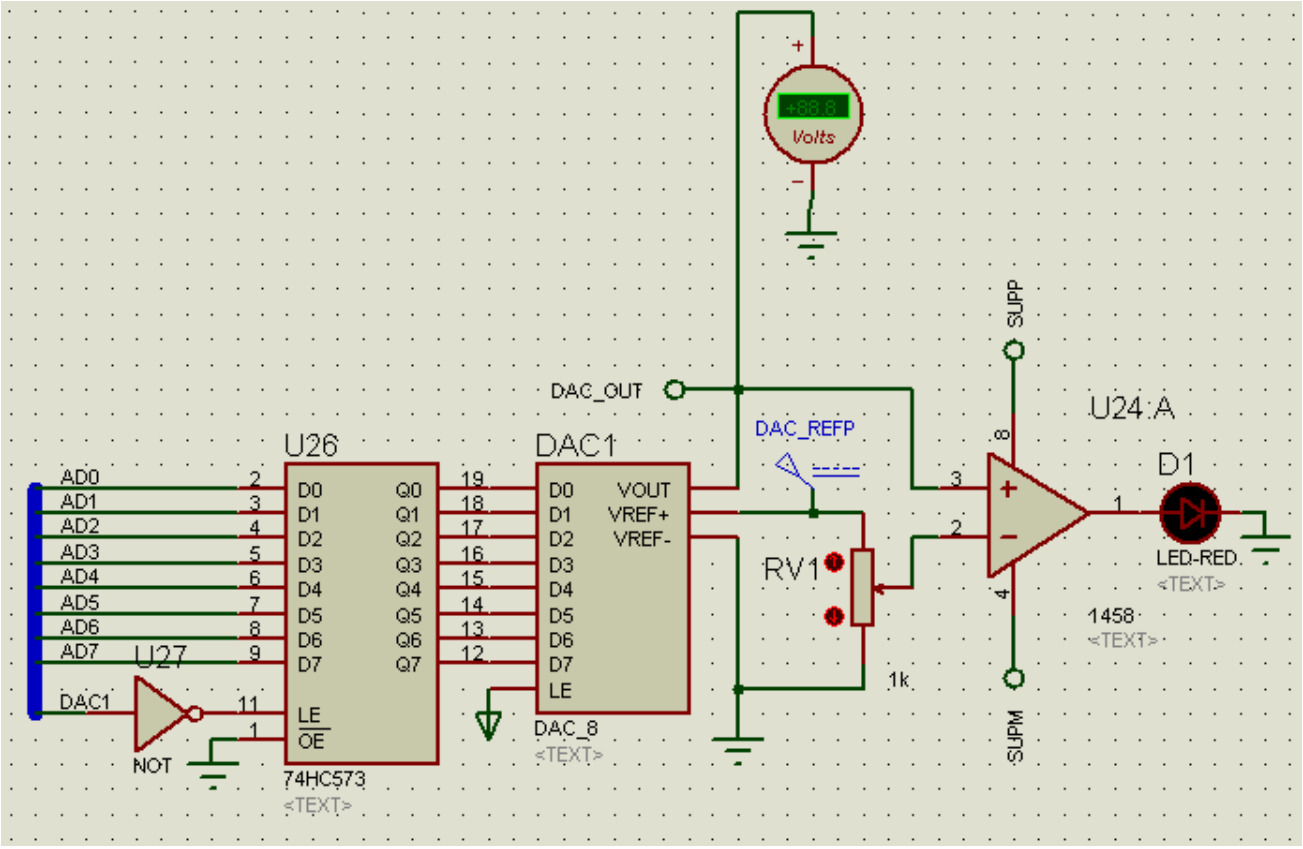


Рисунок 8.13 – ЦАП в виртуальном стенде

В настоящее время ОУ получили широкое применение как в виде отдельных чипов, так и в виде функциональных блоков в составе более сложных интегральных схем. Такая популярность обусловлена тем, что ОУ является универсальным блоком с характеристиками, близкими к идеальным, на основе которого можно построить множество различных электронных узлов.

Компаратор (аналоговых сигналов) (англ. comparator - сравнивающее устройство) — электронная схема, принимающая на свои входы два аналоговых сигнала и выдающая логическую «1», если сигнал на прямом входе («+») больше чем на инверсном входе («−»), и логический «0», если сигнал на прямом входе меньше, чем на инверсном входе.

Простейший компаратор представляет собой дифференциальный усилитель. Компаратор отличается от линейного операционного усилителя (ОУ) устройством и входного, и выходного каскадов:

– Входной каскад компаратора должен выдерживать широкий диапазон входных напряжений между инвертирующим и неинвертирующим входами, вплоть до размаха питающих напряжений, и быстро восстанавливаться при изменении знака этого напряжения. В ОУ, охваченном обратной связью, это требование некритично, так как дифференциальное входное напряжение измеряется милливольтами и микровольтами.

– Выходной каскад компаратора выполняется совместимым по уровням и токам с конкретным типом логических схем (ТТЛ, ЭСЛ и т. п.). Возможны выходные каскады на одиночном транзисторе с открытым коллектором (совместимость с ТТЛ и КМОП логикой).

При подаче эталонного напряжения на инвертирующий вход, входной сигнал подаётся на неинвертирующий вход и компаратор является неинвертирующим (повторителем, буфером).

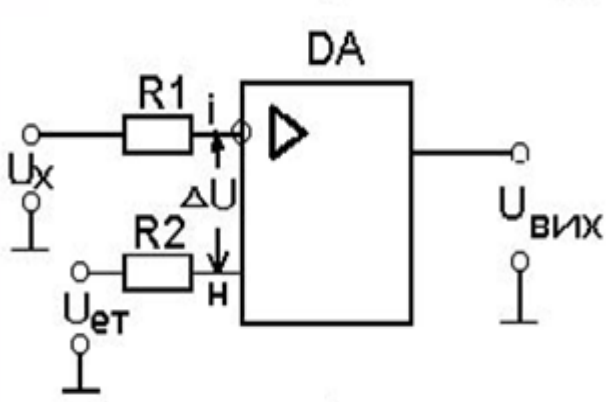


Рисунок 8.14 – Схема включения компаратора для сравнения однополярных напряжений

При подаче эталонного напряжения на неинвертирующий вход, входной сигнал подаётся на инвертирующий вход и компаратор является инвертирующим (инвертором).

8.7 Моделирование динамической индикации

Динамическая индикация – это метод отображения целостной картины через быстрое последовательное отображение отдельных элементов этой картины. Причем, «целостность» восприятия получается благодаря инерционности человеческого зрения.

Для отображения четырёхзначного числа при использовании статической индикации потребуется использовать 32 вывода (не считая общие). При этом у большинства контроллеров количество каналов ввода/вывода как раз равно 32. Выходом может быть подключение всех индикаторы параллельно. Точнее, выводы сегментов соединяют с общей шину. Раздельными остаются контакты активирующие отдельный 7-сегментный индикатор. При этом последовательно подаётся напряжение на адресные входы индикаторов, и одновременно выводится в шину данных 7-сегментный код, соответствующий индикатору, активному в данный момент.

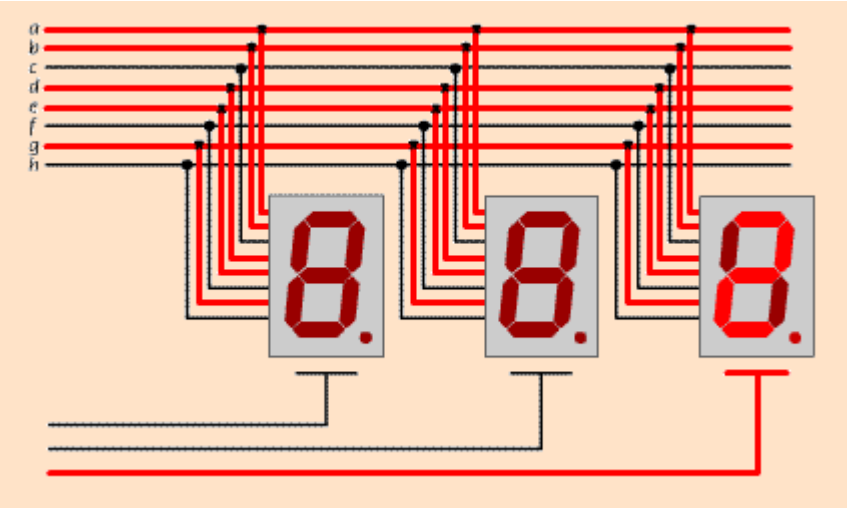


Рисунок 8.15 – Схема динамической индикации

Выбор активного индикатора осуществляется двумя младшими битами порта С (адрес 8xx2H), которые подключены к блоку динамической индикации через унитарный дешифратор 2-в-4. Выходы дешифратора инвертированы, поэтому адресные входы блока динамической индикации тоже должны быть инвертированы (задаётся в свойствах модели).

Данные для отображения передаются в порт В (адрес 8xx2H). Высокий уровень (логическая 1) зажигает сегмент, так как реализована схема с общим катодом.

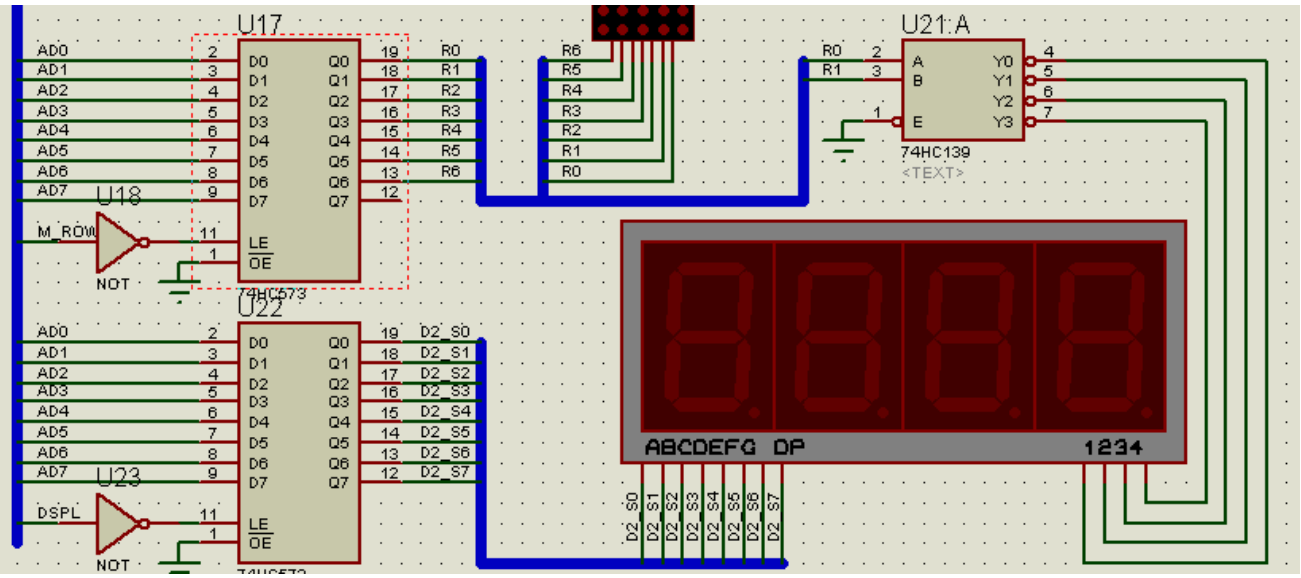


Рисунок 8.16 – Модуль динамической индикации

8.8 Моделирование знакосинтезирующей матрицы

Знакосинтезирующая матрица 5x7 реализована как совокупность 7 блоков из 5 светодиодов с общим катодом. Т.е. на столбцы выведены аноды параллельно включенных светодиодов, а на строки – катоды.

При таком подключении, чтобы зажечь определенный светодиод необходимо подать высокое напряжение (логическую 1) на соответствующий столбец и низкое напряжение на соответствующую строку.

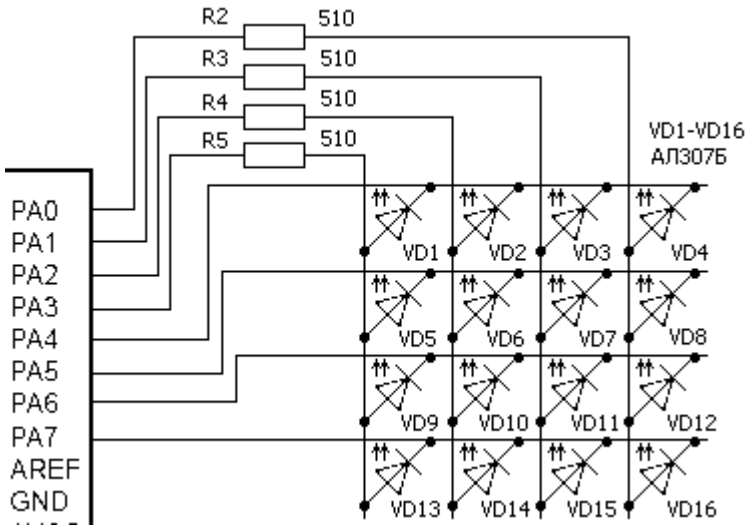


Рисунок 8.17 – Пример подключения матрицы светодиодов

При использовании виртуального стенда для управления строками используются младшие регистры порта B (адрес 8xx1H), а для управления столбцами – младшие регистры порта A (адрес 8xx0H).

Написание программ с использованием знакосинтезирующей матрицы позволит студентам научиться работать с распространенными информационными табло, создавать бегущие строки а также простейшие игры типа «тетрис», «арканоид», «змейка».

Подключение знакосинтезирующей матрицы через порты A и B имеет свои особенности. Так как через порт B управляется также и выбор активного символа в динамической индикации, необходимо следить, чтобы при одновременном использовании упомянутых периферийных устройств не возникало ситуации, в которой значение регистра на выходе порта B не обновляется. Если это произойдет, то активные символы блока динамической индикации будут непредсказуемо меняться, либо будут загораться не те узлы светодиодной матрицы, которые должны гореть. Возможным решением будет

подача на порт A значения 00H, чтобы погасить все светодиоды перед активацией динамической индикации.

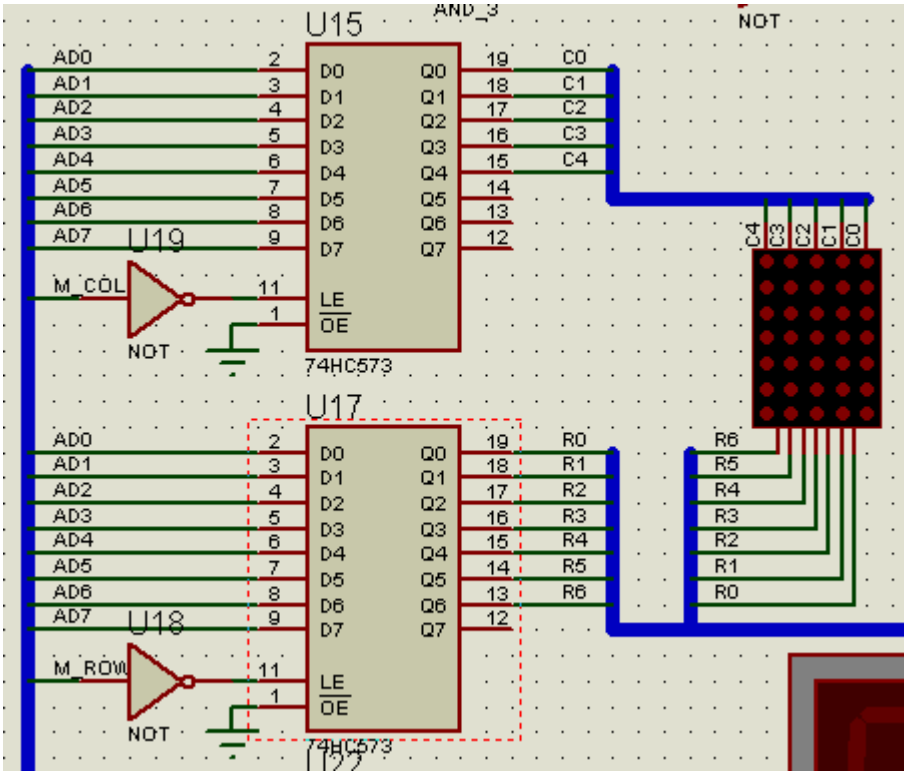


Рисунок 8.18 – Знакосинтезирующая матрица

9 Тестирование виртуального учебно-отладочного стенда

Тестирование должно показать степень соответствия разработанного виртуального учебно-отладочного стенда заявленным требованиям: полное соответствие работы программ на виртуальном и реальном стендах, выполнение программы в реальном времени на персональном компьютере с такими характеристиками:

- ЦПУ с тактовой частотой 2.0 ГГц и выше;
- Объём оперативной памяти 512 Мб и более.

9.1 Методика тестирования

Тестовая программа должна задействовать все модули, содержащиеся в виртуальном учебно-отладочном стенде. В их число входит:

- Блок статической индикации (4 разряда):
- Блок дискретной индикации (8 разрядов):
- Матричная клавиатура 3*4;
- Блок инициализации прерываний: дискретные кнопки - 2 шт.
- Знакосинтезирующая матрица 5x7
- Блок динамической индикации (4 разряда)
- Блок цифроаналогового преобразования

Были написаны тестовые программы, задействующие перечисленные выше модули. Их листинг представлен в приложениях В и Г.

Тестовые программы написаны на языках Ассемблер и С. Для компиляции используются AVRASM2 (компилятор встроенный в пакет Proteus ISIS) и avr-gcc – свободный компилятор, созданный на базе gcc, предназначенный для компиляции исходного кода программ для AVR микроконтроллеров в hex-файлы (или другие форматы).

9.2 Описание программы для тестирования учебно-отладочного стенда №1

Программа для тестирования №1 задействует такие модули учебно-отладочного стенда:

- Блок статической индикации (4 разряда):
- Блок дискретной индикации (8 разрядов):
- Матричная клавиатура 3*4;
- Знакосинтезирующая матрица 5x7
- Блок динамической индикации (4 разряда)
- Блок цифроаналогового преобразования

Программа считывает состояние матричной клавиатуры и отображает код нажатой клавиши в первых двух символах блока статической индикации. В третьем и четвёртом символах блока статической индикации выводится значение нажатой клавиши (специальным символам “*” и “#” соответствуют значения AH и BH).

На блок динамической индикации по шине данных подаётся значение FFH, что, исходя из таблицы 7.1, зажигает в активном знакоместе число 8 и десятичную точку.

Блок дискретной индикации изображает «бегущий огонёк» - поочерёдно зажигаются и гаснут все светодиоды, меняя направления движения при достижении края ряда светодиодов.

На блок цифроаналогового преобразования подаётся инвертированное значение с блока дискретной индикации. На выходе ЦАП образовывается аналоговый сигнал показанный на рисунке 8.2.

На знакосинтезирующую матрицу также подаётся значение с блока дискретной индикации. Но из-за особенностей конструкции светодиодной матрицы на ней загорается инвертированный пятикратный бегущий огонёк. Данная программа задействует все блоки виртуального учебно-отладочного стенда кроме блока инициализации прерываний. Этот блок задействован в тестовой программе №2.

9.3 Описание программы для тестирования учебно-отладочного стенда №2

Тестовая программа №2 написана на языке Ассемблер и скомпилирована с помощью встроенного в Proteus ISIS компилятора AVRASM2.

Программа задействует следующие модули учебно-отладочного стенда:

- Блок статической индикации (4 разряда):
- Блок инициализации прерываний;
- Блок дискретной индикации.

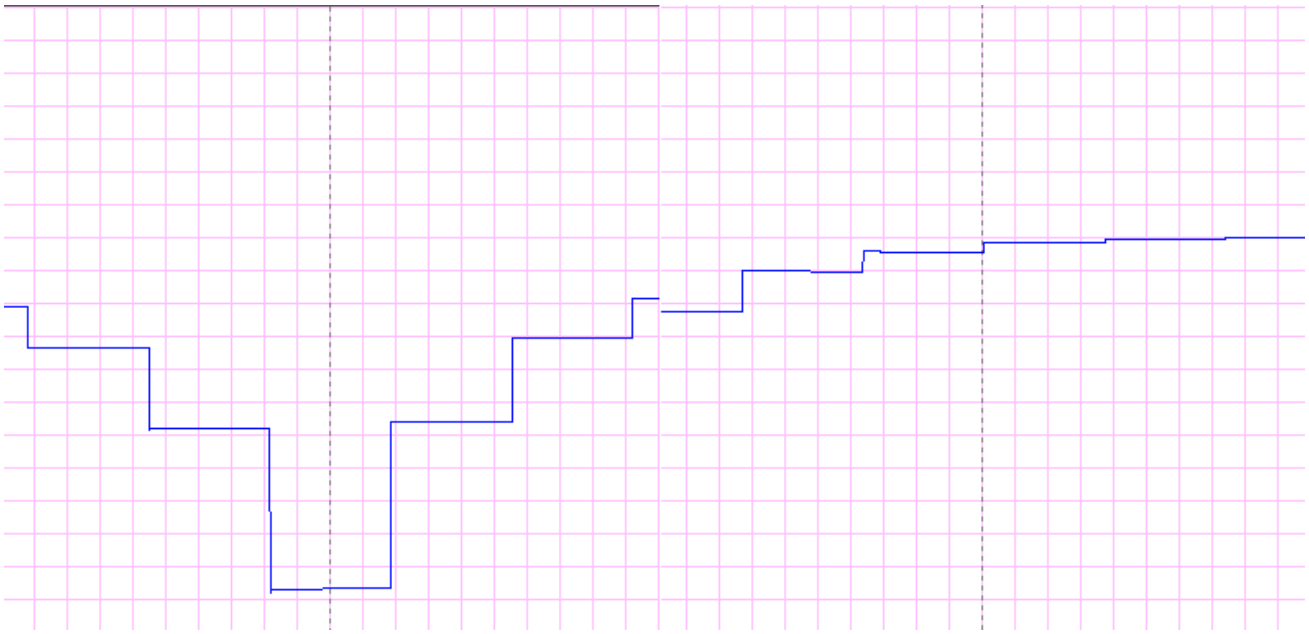


Рисунок 9.1 – Сигнал на выходе блока цифроаналогового преобразования

Программа предназначена для тестирования системы внешних прерываний микроконтроллера ATmega8515s и управления периферийными устройствами с помощью обработчиков прерываний.

При нажатии кнопки SW15 инициализируется прерывание INT0 и значение счетчика инкрементируется. При нажатии кнопки SW16 инициализируется прерывание INT1 и значение счетчика декрементируется. Текущее значение счетчика выводится на блок статической индикации, дублируясь на символах 0-1 и 2-3.

На блок дискретной индикации выводится «бегущий огонёк». При достижении последнего светодиода загорается первый.

9.4 Результаты тестирования

После запуска тестовых программ на виртуальном и реальном учебно-отладочных стендах сделан вывод, что поведение всех модулей идентично. Виртуальный стенд можно использовать в качестве модели реального учебно-отладочного стенда с высокой степенью соответствия.

Скорость работы виртуального стенда в пакете Proteus ISIS немного ниже чем у EV8031/AVR, но это заметно только при прямом сравнении.

Заключение

Разработанный виртуальный стенд отвечает заявленным требованиям и может быть использован в учебном процессе в качестве учебного пособия. Предполагается использование стенда для самостоятельной работы студентов.

Виртуальный стенд может быть дополнен путем моделирования плат расширения реального стенда и добавлением интерфейсов связи с компьютером и другими устройствами. Также, возможно дополнение стенда элементами других версий учебно-отладочного стенда (EV8031/AVR PRO) и различных плат расширения, таких как EV8031/AU.

Отличия разработанного виртуального стенда от реального минимальны, в частности, отличаются реализации системного контроллера, статической индикации и т.д. Вследствие этого невозможно отладить использование части функционала реального стенда, например, управление гашением символа и десятичной точкой в составе статической индикации.

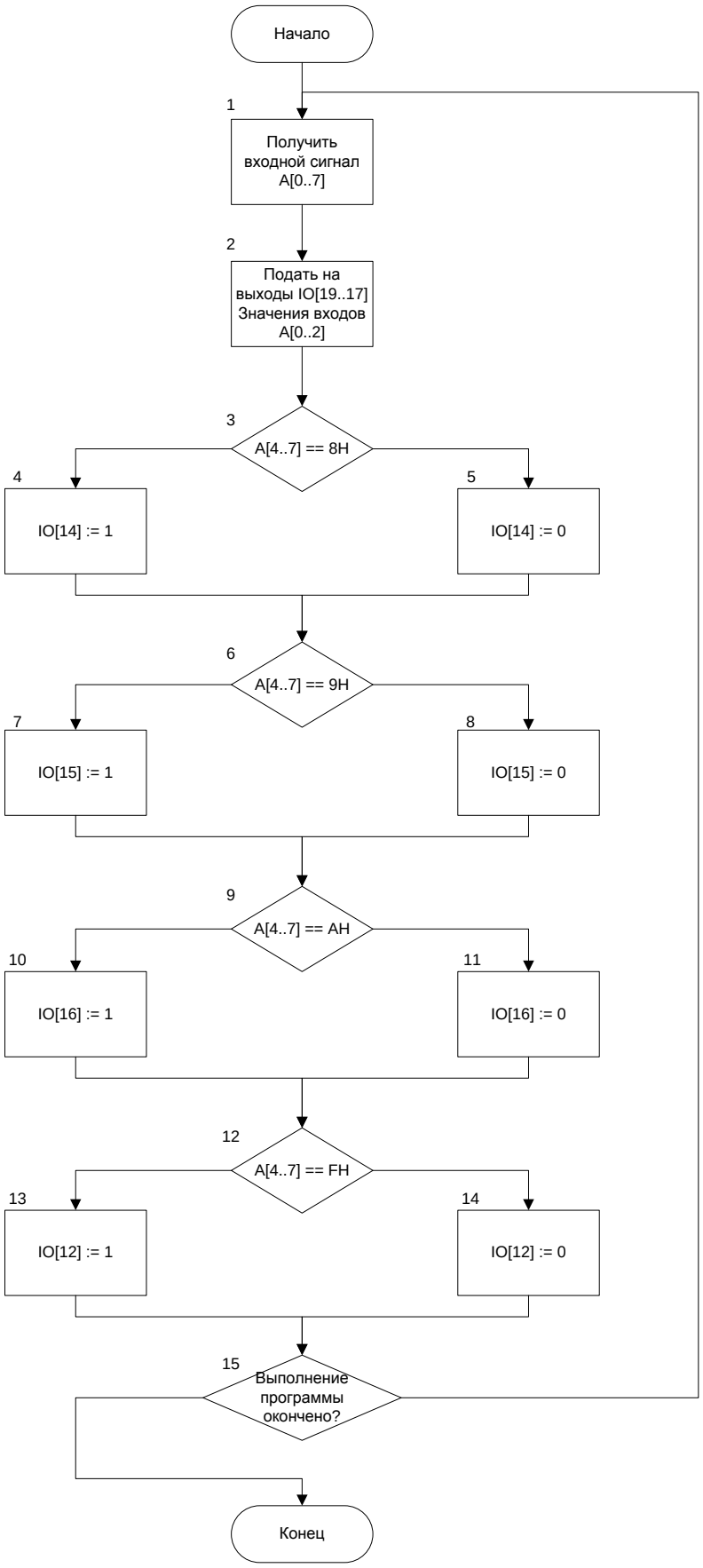
Виртуальный стенд функционирует близко к режиму реального времени, загрузка процессора не превышает 20%.

Список литературы

1. Учебно-отладочный стенд EV8031/AVR. Техническое описание. Инструкция по эксплуатации.
2. Intelligent Schematic Input System. User Manual. Issue 6.3. : Labcenter Electronics - 512 с.
3. Новацкий А.А. Конспект лекций по курсу ЕМСТ.
4. CUPL Programmer’s Reference Guide.: Logical Devices, Inc., 140 с.
5. Linden H. McClure, Ph.D. Programming the Atmel ATF16VC SPLD – 2007
6. Клайв Максфилд. Проектирование на ПЛИС. Архитектура, средства и методы. Курс молодого бойца – Москва: Издательский дом «Додэка-XXI», 2007. - 406 с.
7. ATF16V8C Data Sheet – Atmel Corporation, 1999. - 18 с.
8. Микроконтроллеры AVR семейства Mega. Руководство пользователя – Москва: Издательский дом «Додэка-XXI», 2007. - 594 с.
9. Новацкий А.А. Конспект лекций по курсу ПМПС-2.
10. ATmega8515, ATmega8515S Data Sheet : Atmel Corporation, 2002. - 225 с.
11. Динамический опрос клавиатуры. [Электронный ресурс] – Режим доступа: http://www.realcoding.net/articles/dinamicheskii-opros-klaviatury.html
12. Динамическая индикация. [Электронный ресурс] – Режим доступа: http://www.radiokot.ru/start/mcu_fpga/avr/15/
13. Параллельные ЦАП. [Электронный ресурс] – Режим доступа: http://www.limi.ru/dacs/pardacs.htm
14. Starting Atmel AVR C Programming Tutorial1. [Электронный ресурс] – Режим доступа: http://www.ermicro.com/blog/?p=49

Приложение А

Алгоритм работы системного контроллера



Приложение Б

Прошивка системного контроллера

```
1   /**  Inputs  **/
2
3   Pin [2..4]   = [al0..2];          /* lower address */
4   Pin [6..9]   = [ah0..3];          /* higher address */
5   Pin 5 = wr_rd;              /* write or read */
6
7   /**  Outputs  **/
8   Pin [17..19] = [rt0..2];       /* outputs for routing */
9   Pin 16 = a_en;
10  Pin 15 = 9_en;
11  Pin 14 = 8_en;
12  Pin 12 = f_en;
13
14  /** Declarations and Intermediate Variable Definitions **/
15  set_a = [ah0..3]:'h'A;
16  set_9 = [ah0..3]:'h'9;
17  set_8 = [ah0..3]:'h'8;
18  set_f = [ah0..3]:'h'F;
19
20  /** Logic **/
21
22  rt0 = al2;
23  rt1 = al1;
24  rt2 = al0;
25
26  a_en = set_a & wr_rd;
27  9_en = set_9 & wr_rd;
28  8_en = set_8 & wr_rd;
29  f_en = set_f & wr_rd;
```

Приложение В

Тестовая программа для учебно-отладочного стенда №1

```
1   #include <avr/io.h>
2   #include <stdlib.h>
3   #include <string.h>
4   #include <avr/pgmspace.h>
5   //#include <avr/signal.h>
6   #include <avr/interrupt.h>
7   #include <avr/wdt.h>
8   #include <avr/eeprom.h>
9   #include "bitdef.h"
10
11  unsigned char *lefti=(unsigned char *)0xa000;
12  unsigned char *righti=(unsigned char *)0xa001;
13  unsigned char *dotsi=(unsigned char *)0xA004;
14  unsigned char *LED_REG=(unsigned char *)0xa006;
15  unsigned char *pA=(unsigned char *)0x8000;
16  unsigned char *pB=(unsigned char *)0x8001;
17  unsigned char *pC=(unsigned char *)0x8002;
18  // KEYBOARD
19  unsigned char *kbc0=(unsigned char *)0x9006;
20  unsigned char *kbc1=(unsigned char *)0x9005;
21  unsigned char *kbc2=(unsigned char *)0x9003;
22  // LED MATRIX
23  unsigned char *mrow=(unsigned char *)0x8002;
24  unsigned char *mcol=(unsigned char *)0x8000;
25  // DISPLAY CONTENT
26  unsigned char *DSP_REG=(unsigned char *)0x8001;
27  // DAC REGISTER
28  unsigned char *DAC_REG=(unsigned char *)0xF001;
29
30  unsigned char KEY_CODES[] = {1,4,7,10, 2,5,8,0, 3,6,9,11, 12};
31
32  volatile unsigned int delay_msec,need_zoom=500;
```

```
33
34  #include "delays.c"
35
36  SIGNAL(SIG_OUTPUT_COMPARE1A)
37  {
38    PORTB ^= _10000000;                          // debug output
39    if (delay_msec) delay_msec--;                    // delay
40  counter
41    if (need_zoom) { need_zoom--; PORTB ^= _01000000; }  // zoom
42  processing
43  }
44
45  void hardware_init()
46  {
47    DDRB = _11000000;
48    MCUCR = _11000000;          // ext. RAM enable
49
50
51    OCR1A = 114;           // period ~1ms
52    TCCR1B = _00001011;          // CTC1, PS=64
53    TIMSK |= (1 << OCIE1A);     // OCIE1A interrupt enable
54    sei();
55
56    *pA = 0x00;
57    *pB = 0;
58    *pC = 0xff;
59
60    *LED_REG = 0;
61    *dotsi = _00000011;
62  }
63
64  int main ()
65  {
66      unsigned char keycounter;
67      unsigned int kbdstate;
68      unsigned char ledstate = _00000001;
```

```
69      unsigned char dir = _11111111;
70
71      unsigned char dac = _00000001;
72
73      hardware_init();
74
75      while (1){
76
77       kbdstate  = (~(*kbc0) & _00001111);
78       // read column 0
79       kbdstate |= (~(*kbc1) & _00001111) << 4;
80       // read column 1
81       kbdstate |= (~(*kbc2) & _00001111) << 8;
82       // read column 2
83
84       for (keycounter=0;keycounter<12;keycounter++) {
85       // keyboard state processing
86         if (kbdstate & 1) break;
87       // if LSB = 1 then break
88         kbdstate = kbdstate >> 1;
89       // shift kbdstate to right
90       }
91
92       // show keycounter in hex-code on right part
93       // of 7segment indicator
94       // if key not pressed => 0x0c
95       *righti = keycounter;
96       // show pressed key on left part of 7segment indicator
97       *lefti = KEY_CODES[keycounter];
98
99       delay(100);
100
101         *mrow = ledstate;//_10101010;
102         *mcol = _11111111;
103
104      *LED_REG = ledstate;
```

```
105             *DAC_REG = ~ledstate;
106             *DSP_REG = _11111111;
107 
108             if (dir)
109             {
110                 ledstate = (ledstate << 1);
111             } else {
112                 ledstate = (ledstate >> 1);
113             }
114 
115             if (ledstate == _10000000 || ledstate == _00000001){
116                 dir = ~dir;
117             }
118 
119         }
120 
121     }
```

Приложение Г

Тестовая программа для учебно-отладочного стенда №2

```
1   .include "m8515def.inc"
2
3   ; Register definitions.
4   .def temp = r16
5   .def int0_counter = r17
6   .def int1_counter = r18
7
8   ; Constants.
9   .equ INIT_LED_MASK = 0x01
10
11  .dseg
12
13  ; Variables.
14  counter:  .byte     1                   ; simple counter
15  led_mask: .byte     1
16
17  ; Acccess to the some devices on board is implemented through
18  read/write
19  ; from/to the external RAM. Of course we have to enable it at
20  the program
21  ; star t. For device addresses consult the board's
22  documentation.
23
24  ; Here we place some fake variables which in fact are the ports
25  to
26  ; acccess the board's peripherals.
27  .org 0xA006
28  led_reg:  .byte     1                   ; LEDs
29
30  .org 0xA000
31  left_ind: .byte     1                   ; left static 7-segment
32  LED indicator
```

```
33  right_ind:     .byte     1                    ; right static 7-
34  segment LED indicator
35
36  .cseg
37
38  .org 0x0000
39  reset:
40       rjmp      reset_handler
41
42  .org INT0addr
43       rjmp      int0_handler
44
45  .org INT1addr
46       rjmp      int1_handler
47
48  main:
49  ; Stack initialization.
50       ldi       temp, low(RAMEND)
51       out       SPL, temp
52       ldi       temp, high(RAMEND)
53       out       SPH, temp
54
55  ; Variables initialization.
56       clr       temp
57       clr       int0_counter
58       clr       int1_counter
59       sts       counter, temp            ; counter = 0
60
61       ldi       temp, INIT_LED_MASK
62       sts       led_mask, temp      ; led_mask = INIT_LED_MASK
63
64  endless_loop:
65
66       ;lds      temp, counter
67       ldi       XL, low(left_ind)
68       ldi       XH, high(left_ind)
```

```
69        st        X+, int0_counter          ; left_ind = counter
70        st        X, int0_counter           ; right_ind = counter
71
72        ;inc      temp                      ; counter++
73        sts       counter, temp
74
75        lds       temp, led_mask
76        ldi       XL, low(led_reg)
77        ldi       XH, high(led_reg)
78        st        X, temp                   ; led_reg = led_mask
79
80        lsl       temp                 ; led_mask = led_mask << 1;
81
82        tst       temp
83        brbc      1, next_iter              ; if (led_mask != 0)
84                                            ; goto next_iter
85
86        ldi       temp, INIT_LED_MASK ; led_mask = INIT_LED_MASK
87  next_iter:
88
89        sts       led_mask, temp
90
91  ; Some delay.
92        ldi       temp, 0x00
93
94  endless_loop_delay_loop:
95
96        rcall          delay
97        rcall          delay
98        rcall          delay
99
100       dec       temp
101       brne      endless_loop_delay_loop
102
103 ; Repeat once again.
104       rjmp      endless_loop
```

```
105
106  ; Simple and stupid delay function.
107  delay:
108       push      temp
109       ldi       temp, 0x00
110
111  delay_loop:
112       dec       temp
113       brne      delay_loop
114
115       pop       temp
116
117       ret
118
119  reset_handler:
120  ; Enable external RAM.
121       ldi       temp, 0xCF
122       out       MCUCR, temp
123       ldi       temp, 0xF0
124       out       GICR, temp
125  ;    sei
126       ldi       temp, 0x80
127       out       SREG,  temp
128  ; Jump to the 'main' function.
129       rjmp      main
130
131  int0_handler:
132       inc       int0_counter
133       reti
134
135  int1_handler:
136       dec       int0_counter
137       reti
```

Приложение Д

Внешний вид виртуального учебно-отладочного стенда