Министерство образования и науки РФ
Федеральное государственное образовательное учреждение
высшего профессионального образования
ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ СИСТЕМ
УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ (ТУСУР)
Кафедра комплексной информационной безопасности электронно-
вычислительных систем (КИБЭВС)

РЕСУРСЫ УПРАВЛЕНИЯ И ПРОГРАММНОЕ УПРАВЛЕНИЕ ПЕРИФИРИЙНЫМИ УСТРОЙСТВАМИ СТЕНДА SDK-1.1

Отчёт по производственной практике

Выполнил:
Студент гр. 772
_____ Марсюков Н.В.
« »_____2015г.
Руководитель практики от предприятия:
доцент каф. КИБЭВС
_____Торгонский Л.А.
« »_____2015г.

Томск 2015

Реферат

Отчет по преддипломной практике содержит 64 страниц, 25 рисунков, 7 источников, 8 приложений.

СТЕНД SDK-1.1, МИКРОПРОЦЕССОР ADUC842, ТРАНСЛЯТОР TASM, ОРИГИНАЛЬНАЯ ПРОГРАММА ОТЛАДКИ, ПРОГРАММИРОВАНИЕ, ПЕРИФИРИЙНЫЕ МОДУЛИ СТЕНДА

Программа производственной практики исполнена в лаборатории электроники и схемотехники кафедры КИБЭВС факультета безопасности ТУСУРа.

Цель практики: обретение навыков работы в области микропроцессорной техники по профилю профессиональной деятельности, закрепление и развитие полученных во время обучения знаний и умений на практике. окончанию практики были получены навыки работы с учебным 1.1.

Исследованы ресурсы стенда SDK 1.1 по управлению светодиодной линейкой, клавишным модулем, модулем расширения, жидкокристалличесским дисплеем, аналого-цифровым и цифроаналоговым преобразователями, звуковым излучателем. Материалы исследования представлены в виде схем, комментариев, программ управления.

Отчет выполнен в текстовом редакторе Microsoft Word 2013 и представлен на компакт-диске CD-R

Министерство образования и науки РФ

Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования

«ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ СИСТЕМ УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ» (ТУСУР)

Кафедра комплексной информационной безопасности электронно-вычислительных систем (КИБЭВС)

ЗАДАНИЕ

на производственную практику

Студенту Марсюкову Никите Вячеславовичу группы 772, факультет безопасности.

1.Тема индивидуального задания:

Исследование и управление ресурсами ПУ стенда SDK-1.1.

Тема отчёта:

Ресурсы управления и программное управление ПУ стенда SDK 1.1.

2. Цель работы: Контроль соответствия, анализ ресурсов, отладка программ управления ресурсами стенда SDK-1.3.Исходные данные к заданию.

3.1.1 Изучить организацию и структуру управление деятельностью ТУСУРа

3.1.2 Исследовать технические средства и возможности управления периферийными устройствами стенда SDK-1.1.

3.1.3 Освоить программирование и выполнить отладку программ управления ПУ стенда SDK-1.1.

3.1.4 Подготовить отчётные материалы к защите результатов практики Руководитель практики

доц. Торгонский Л.А.

_____

(подпись руководителя)

Задание принял к исполнению

Студент гр. 772

Марсюков Н.В.

_____

(подпись студента)

«___» _____ 2015 г

Содержание

1 Введение

В период с 22.06.15 по 20.07.15 была пройдена производственная технологическая практика в лаборатории «Электротехники и электроники» кафедры КИБЭВС, Факультета Безопасности ТУСУРа в учебно-лабораторном корпусе (УЛК). В процессе прохождения практики:

- ознакомился со организационной структурой вуза, местом практики в этой структуре и предоставил отчётный материал по структуре в разделе 2 отчёта;

- изучил материалы руководства по применению стенда SDK-1.1;

- изучил и представил в разделе 3 отчёта программные модели ПУ стенда SDK-1.1;

- освоил приёмы подготовки и отладки программ управления ПУ на языке программирования Ассемблер с применением программной оболочки Aldonah;

- подготовил, отладил и представил в разделе 3 отчёта программы управления резидентными периферийными устройствами стенда:

- таймерами;

- линейкой восьми светодиодов;

- матричным клавишным модулем размерности 4x4;

-жидкокристаллическим индикатором со встроенным контроллером с логикой командного управления электроникой (ЖКИ);

- цифроаналоговым преобразователем (ЦАП);

-портом расширения ENA

2 Сведения о месте прохождения практики

Местом прохождения практики была определеня лаборатория кафедры ТУСУРа – КИБЭВС (Кафедра комплексной информационной безопасности электронно-вычислительных систем).

Кафедра организована в ТУСУРе в 1971 году, как кафедра «Конструирования и производства электронно-вычислительной аппаратуры».

Кафедра была переименована 21 сентября 1999 г. в связи с открытием новой актуальной специальности «Комплексное обеспечение информационной безопасности автоматизированных систем» в кафедру «Комплексной информационной безопасности электронно-вычислительных систем».

На рисунке 2.1 представлена структурная схема управления ТУСУРом.

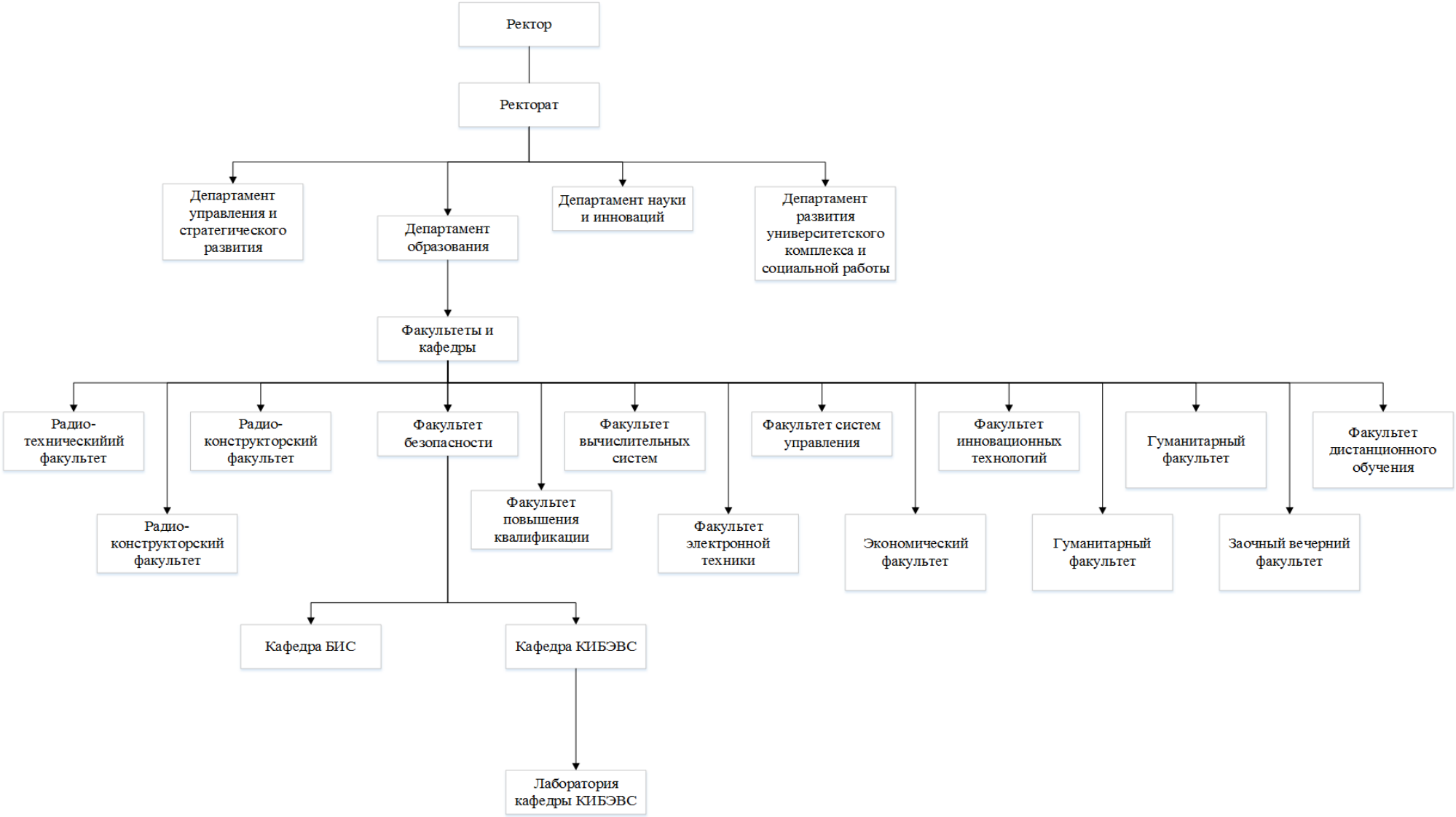


Рисунок 2.1 – Схема управления ТУСУРом.

3 Работа по индивидуальному заданию

3.1 Общие сведенья

Практика проходила учебном-лабораторном корпусе, в аудитории 404. Был получен план работы на время прохождения практики и теоритический материал , для ознакомления с работой. Был составлен план работы на время прохождения практики. В план работы входили: изучение архитектуры стенда SDK-1.1 и на основе полученных знаний, отладить программы для работы с периферийными устройствами данного стенда. Для отладки программ был выдано программное обеспечение «Aldonah».

Следуя плану, практика была закончена в соответствии со сроками прохождения практики. На время прохождения практики было предоставлено рабочее место, с необходимым комплектом оборудования. На рисунке 3.1 представлен комплект выданного оборудования. В данный комплект входит:

1)отладочный стенд SDK-1.1, с необходимыми ресурсами для работы со стендом(блок питания, кабель для загрузки программ с персонального компьютера);

2)модуль расширения стенда SDK-1.1;

3)мультиметр;

4)персональный компьютер;

5)программное обеспечение «Aldonah»;

6)Переходной кабель USB-RS232;

За имением собственного ноутбука, необходимость в выданном персональном компьютере отсутствовала и все работы проходили на ноутбуке.

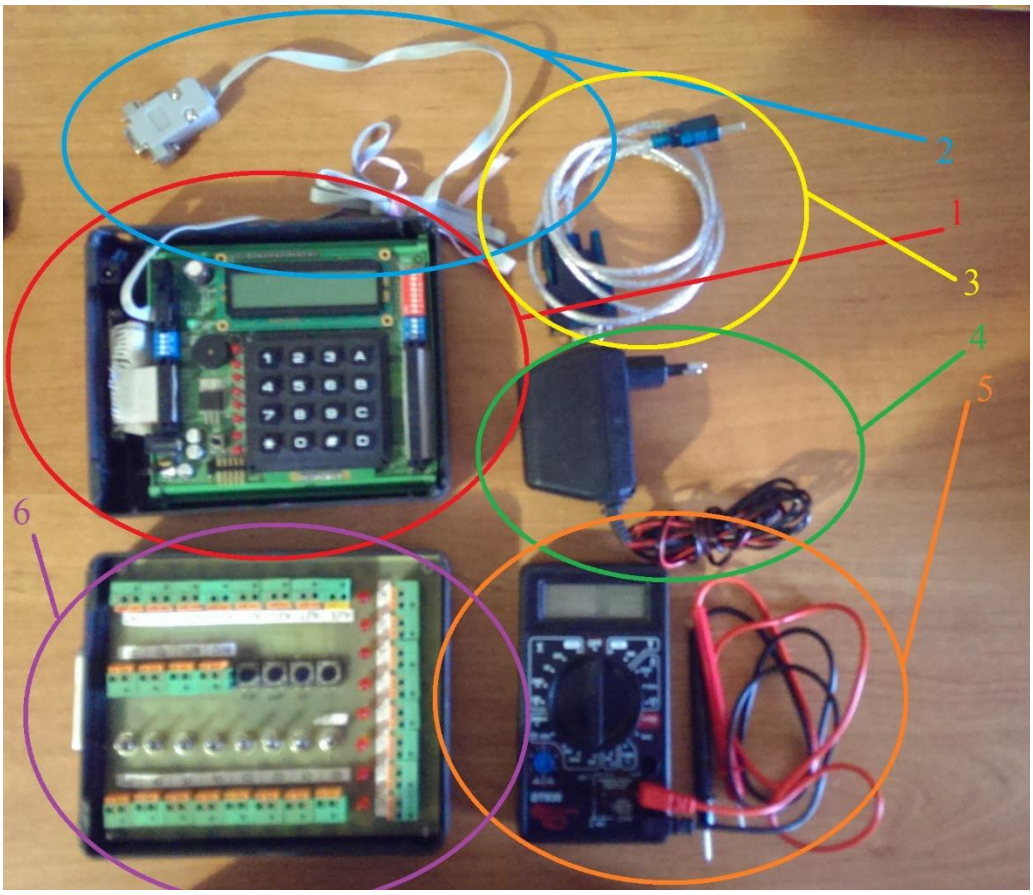


Рисунок 3.1 – Выданное оборудование

На рисунке 3.1 – под условно обозначенными цифрами располагаются соответственно:

1. учебный стенд SDK-1.1;
2. кабель дя загрузки программ;
3. переходной кабель USB-RS232;
4. блок питания для SDK-1.1;
5. мультиметр;
6. модуль расширения для SDK-1.1;

4 Архитекутура стенда SDK-1.1

4.1 Структура аппаратной части

В состав учебного стенда SDK-1.1 входят

- Микроконтроллер Aduc842;
- Внешняя E2PROM объёмом 256 байт;
- Клавиатура АК1604A-WWB Фирмы ACCORD;
- Жидкокристаллический индикатор (ЖКИ) WH1602B-YGK-CP фирмы Winstart Display;
- Часы реального времени PCF8583;
- 128K Внешней SRAM с возможностью расширения до 512K;
- Набор сигнальных светодиодов (8 шт.).

На рисунке 4.1 представлена структура аппаратной части учебного стенда SDK – 1.1.

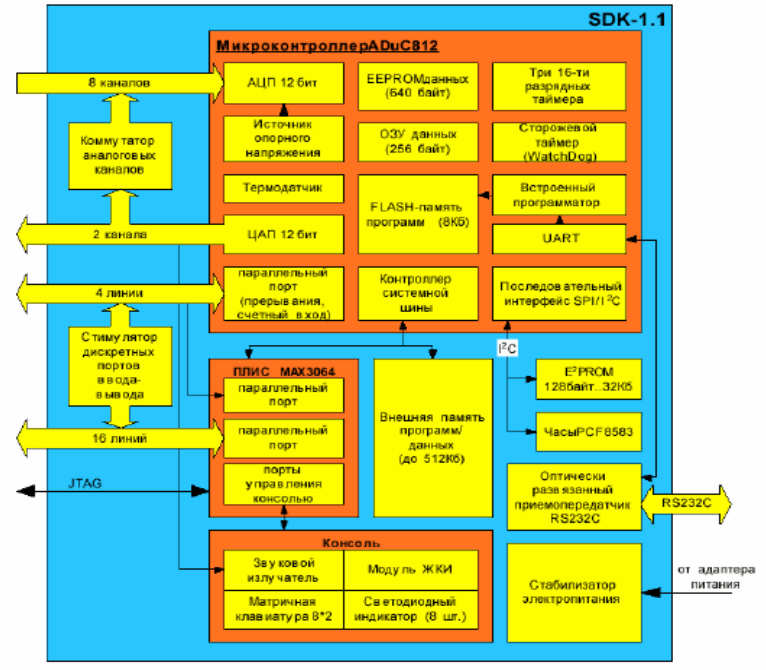


Рисунок 4.1 – Структура аппаратной части учебного стенда SDK-1.1

Распределение памяти представлено в приложении А.

Схематическое изображение стенда представлено в приложении А.

5 Интерфейсы для отладки программ

Для отладки программ используется последовательный порт стенда JDP1 (Приложение А схема изображения стенда SDK-1.1). Что бы была возможность отлаживать написанные программы на стенде SDK-1.1. Его необходимо подготовить для работы, то есть подключить в соответствии со схемой изображенной на рисунке 5.1, для этого выполняются следующие пункты:

1. переходной кабель USB-RS232 подключается стороной имеющим USB разъём к персональному компьютеру, второй стороной подключается к стороне RS-232 кабеля для загрузки программ в SDK-1.1;
2. кабель для загрузки программ в SDK-1.1. второй стороной имеющим JDP1 выход подключается к разъёму стенда JDP1;
3. стенд подключается к источнику питания через блок питания. В роли источника питания выступает розетка;

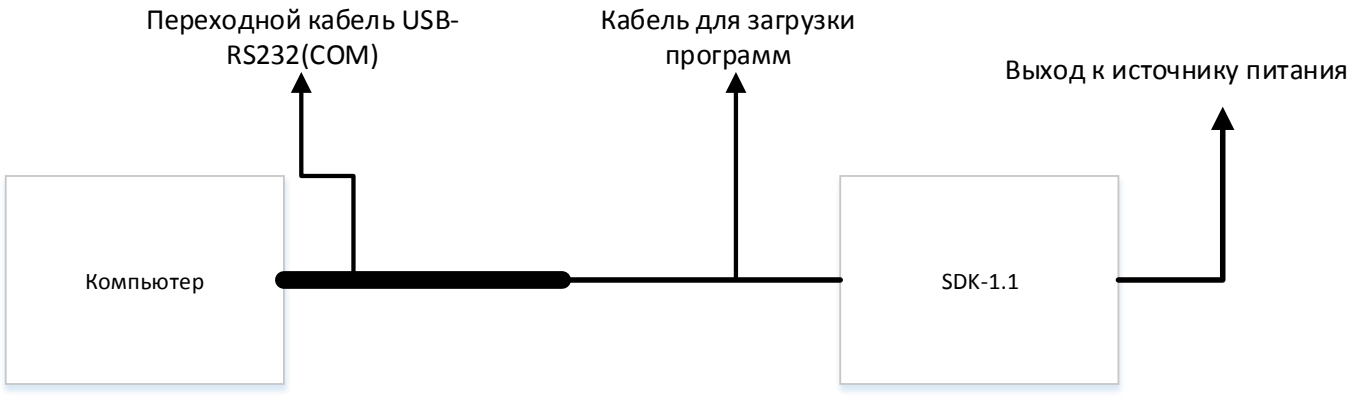


Рисунок 5.1 – схема подключения стенда SDK-1.1 к компбютеру.

Примечание: имеется несколько вариантов исполнения переходного кабеля USB-RS232, следовательно драйвера для каждого варианта кабеля разные и не всегда все драйвера работают корректно. Решение данной

проблемы - это попробовать установить более ранний драйвер для кабеля, либо обновить до более актуальной версии.

После подготовки стенда, подготавливается программа для отладки.

Для этого необходимо иметь необходимые ресурсы:

1. TASM.exe - компилятор-транслятор asm-файлов в hex-файлы с указанными в пакетном файле параметрами;
2. TASM51.TAB - табличный файл под ядро i8051;
3. Define.asm - модуль, содержащий таблицу sfr-регистров;
4. Инструментальная система Т2, для преобразования файла HEX в BIN и загрузки программ в стенд SDK-1.1;
5. Test.asm - исходный текст программы, созданный в блокноте по правилам программирования на ассемблере;

Для практического ознакомления работы со стендом можно взять готовый пример работы со светодиодами представленном в приложении Б, и перенести его в файл Test и сохранить с расширением asm. Сценарий исполнения программы приводится в разделе 6.1.2.

Все выше перечисленные файлы необходимо расположить в папке «C:\SDK-1.1»

Далее производится трансляция программы транслятором ассемблера tasm.exe в HEX –формат. Она необходима для того, чтобы преобразовать код команды, написанный на языке программирования ассемблер, в машинный код, который будет записан в постоянную память программ стенда и по которому, будет происходить выполнение программ. Для 32-х и 64-х разрядных систем необходимо использовать DosBox эмулирующий 16-разрядную систему. В окне DosBox необходимо прописать следующие команды для трансляции:

1) «mount C C:\SDK-1.1» - смонтировать директорию SDK-1.1;
2) «C:» -Выбрать директорию C:\SDK-1.1;

3)«tasm.exe –h -51 test.asm test.hex» - трансляция программы;

На рисунке 5.2. представлен результат при удачной трансляции, а на рисунке 5.3 представлен результат при не удачной трансляции.

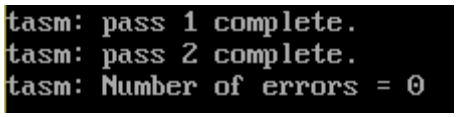


Рисунок 5.2 – Пример результата удачной трансляции.

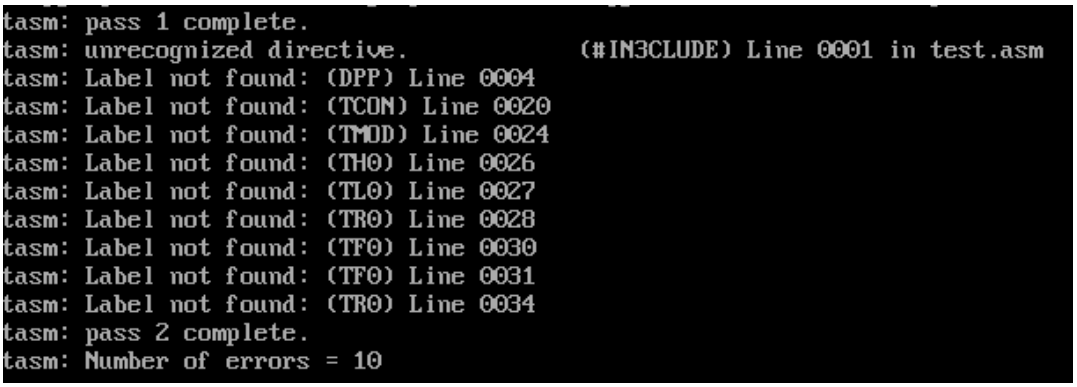


Рисунок 5.3 – Пример результат неудачной трансляции.

В папке «C:\SDK-1.1» автоматически создаются два файла: файл test.hex с содержанием представленном на рисунке 5.4, и файл test.lts где содержатся контрольные суммы трансляции.

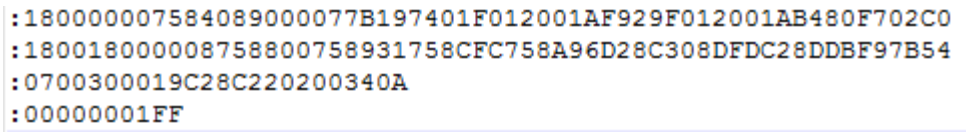


Рисунок 5.4 – Содержание файла test.hex.

Данные на рисунке 5.4 представлены в шестнадцатеричном виде, в виде строк. Каждая строка начинается с символа «:», далее идет: длинна записи (количество байтов данных), адрес по которому будут начинаться загружаться данные, тип записи (00 – данные, 01 - завершение), непосредственно данные и контрольная сумма, необходимая для проверки корректной записи данных из компьютера в память стенда.

Далее производится загрузка HEX-модуль в стенд через интерфейс RS-

232(COM) с помощью инструментальной системы T2.exe. Для этого нужно открыть командную строку и выполнить следующие пункт:

1) «cd c:\sdk-1.1» - переход в директорию с файлами;
2) «t2.exe» - запустить программу T2 , интерфейс управления программой располагается использованной в командной строке;
3) «0x0000 0x0 addhexstart test.hex» - указание начального адреса памяти размещения программы в SDK-1.1;
4) «9600 openchannel com3» - открытие соединения (номер после com может отличаться, необходимо уточнить в диспетчере устройств компьютера);
5) «loadhex test.hex» - загрузка программы в стенд SDK-1.1 (при выполнении этой команды необходимо нажать кнопку «SW2» находившуюся на стенде SDK-1.1) Далее должна произойти запись программы и по окончанию начать выполняться программа;

6)После окончания работы со стендом необходимо прописать команду «closechannel» для корректного закртия соединения со стендом.

7)Для корректного выхода из программы T2.exe служит команда «bye».

На рисунке 5.5 можно наблюдать результаты выполнения команд

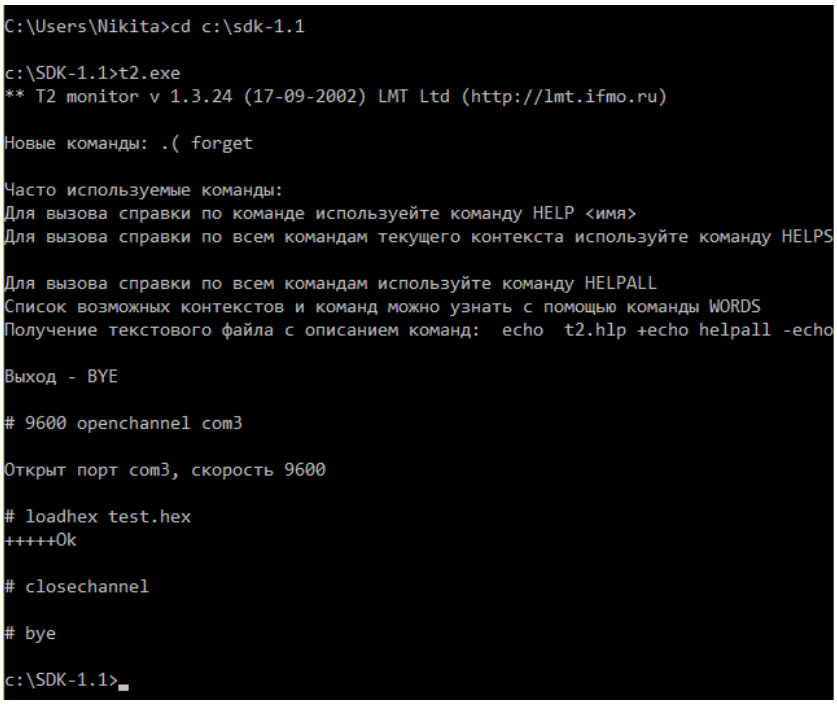


Рисунок 5.5 – Загрузка программы

Для более удобной отладки было предоставлено программное обеспечение «Aldonah.exe» разработанной Чугоевским К.И. в рамках дипломного проекта «Компьютерное управление автоматизированной установкой травления печатных плат». ПО «Aldonah» включает в себя функции: трансляции, добавления стартового, открытия канала, загрузки программы в стенд SDK-1.1.

Трансляция осуществляется автоматизированным запуском эмулятора DosBox, где происходит автоматизированный вызов транслятора tasm.exe, а добавления стартового адреса , открытия канала и загрузки программы через последовательный порт JDP1 выполняется автоматизированными запусками программы T2.exe. Следовательно программа «Aldonah» является оболочкой управления программами DosBox.exe, tasm.exe и t2.exe.

На рисунке 5.6 изображена стартовая форма взаимодействия с пользователем.

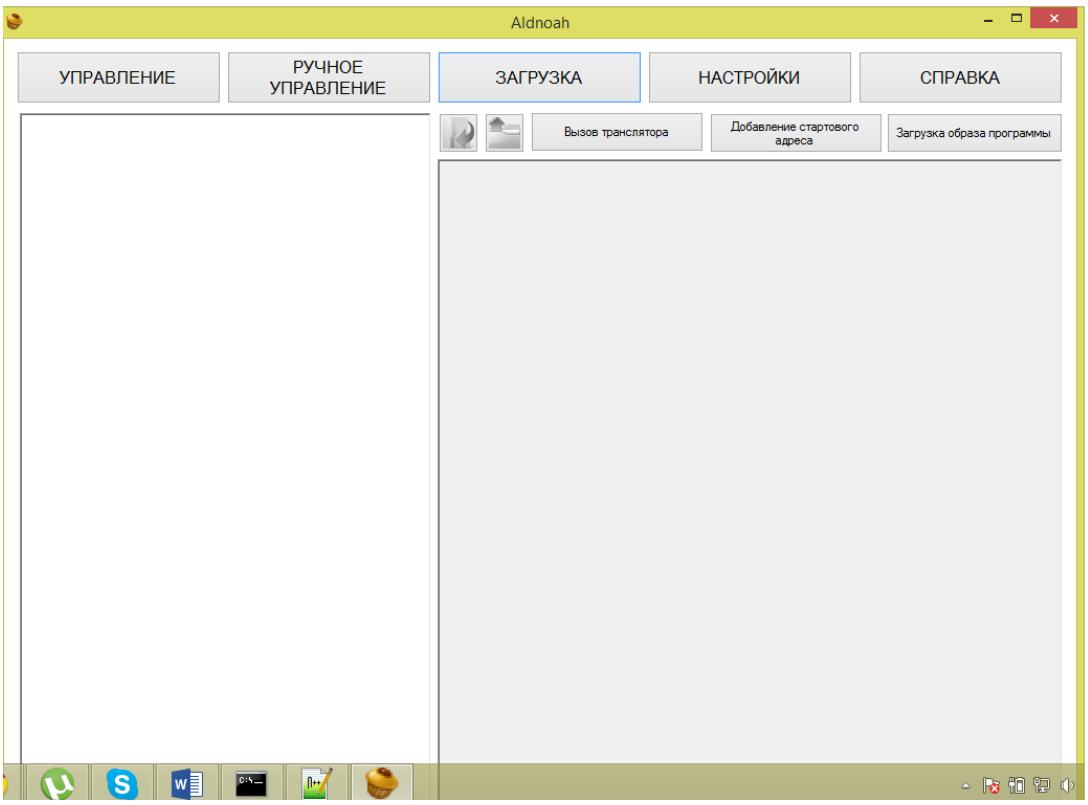


Рисунок 5.6 – Стартовое окно программы «Aldonah».

Для работы с программой «Aldonah» , необходимо выбрать использованный com порт к которому подключен стенд SDK-1.1, выбрать скорость передачи данных и стартовый адрес загрузки. Для этого необходимо перейти во вкладку «НАстройки». И Установить соответствующие параметры в соответствии с рисунком 5.7 , номер com-порта может отличаться, нужно уточнить в диспетчере устройств.

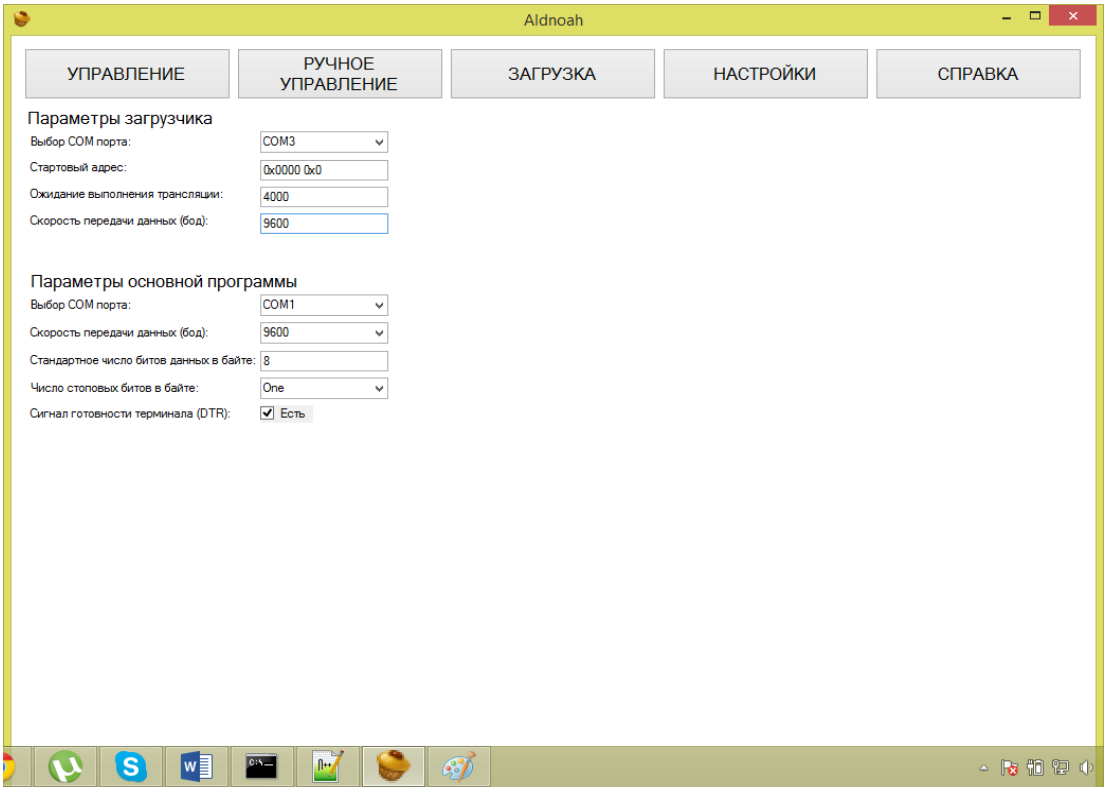


Рисунок 5.7 – окно настроек Aldonah.

После настройки программы «Aldonah» можно подгатовить код для трансляции и загрузки в стенд SDK-1.1. Для этого необходимо перейти во вкладку «Загрузки» рисунок 5.8.

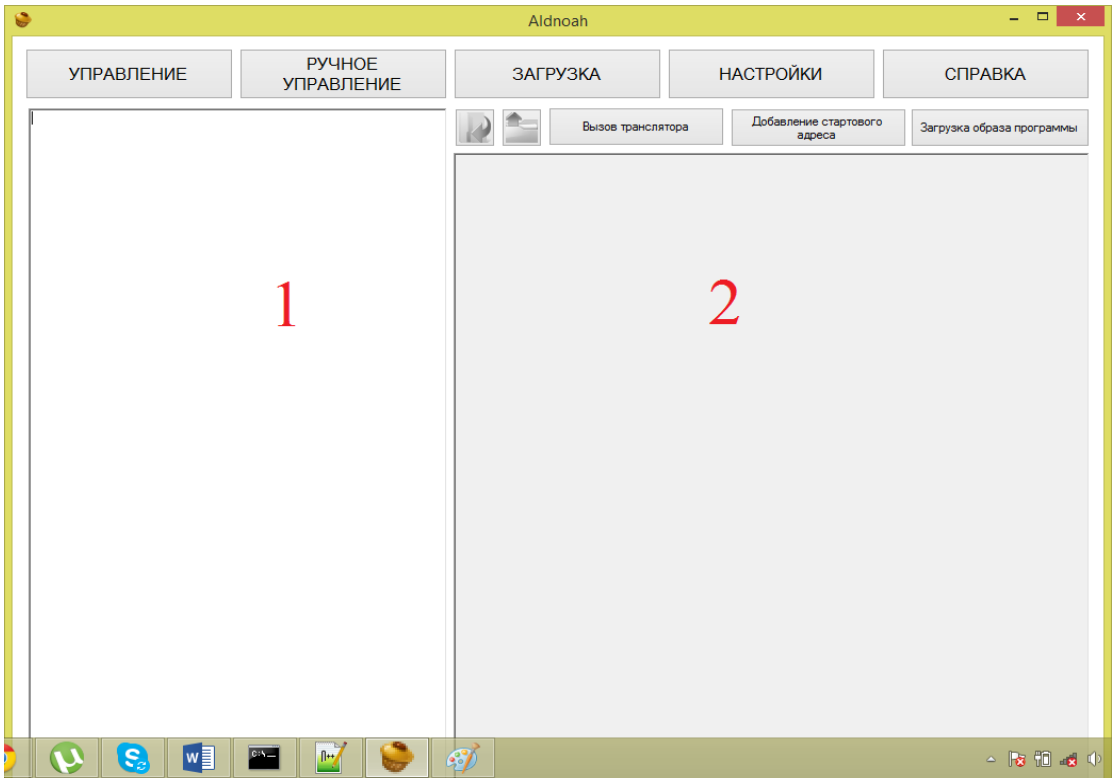


Рисунок 5.8 – Окно «Загрузка» программы Aldonah.

В окне «1» условно обозначенном на рисунке 5.8 располагается код программы написанный на языке Assembler. В окне «2» выводится контрольная сумма результата трансляции. Расположим в окне «1» пример программы работы светодиодов прикасающимся в приложении Б. Для трансляции служит кнопка «вызов транслятора». В окне «2» расположена контрольная сумма трансляции. На рисунках 5.9 и 5.10 представлены результаты удачной и не удачной трансляции соответственно.

Примечание: для опознания результат трансляции необходимо окно «2 прокрутить до конца вниз и определить кол-во ошибок»

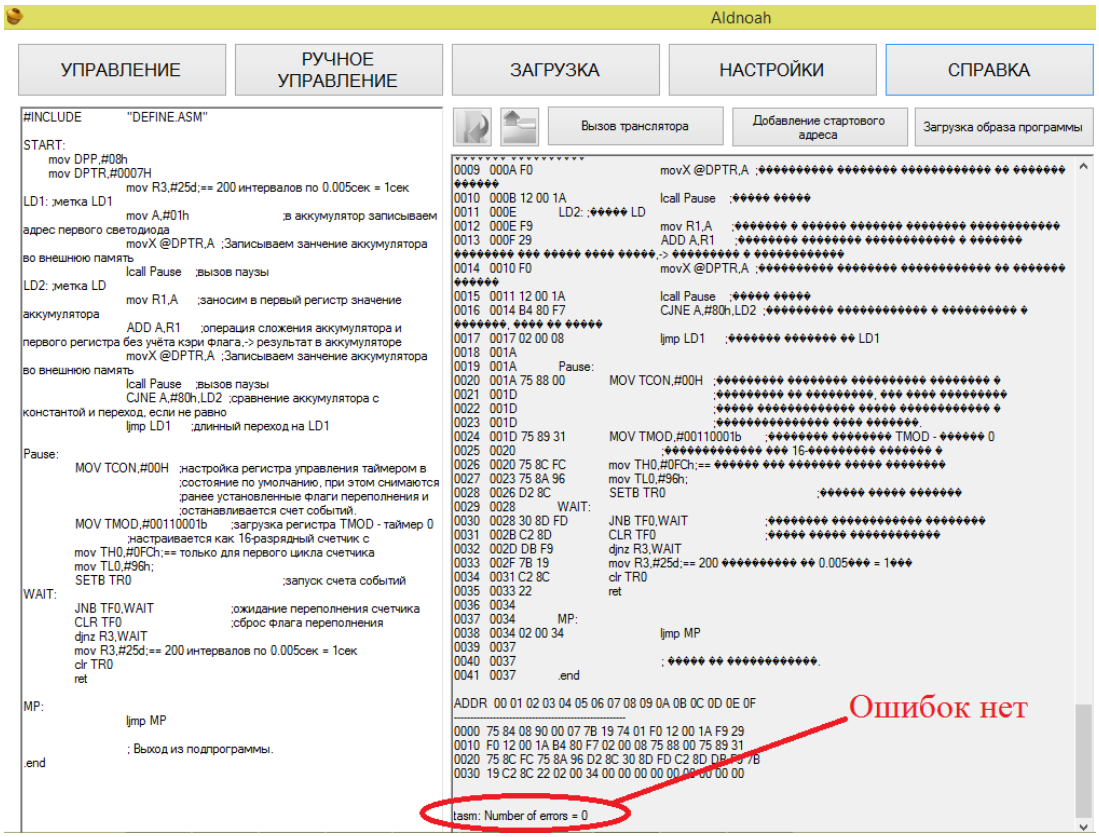


Рисунок 5.9 – Результат удачной трансляции в программе «Aldonah»

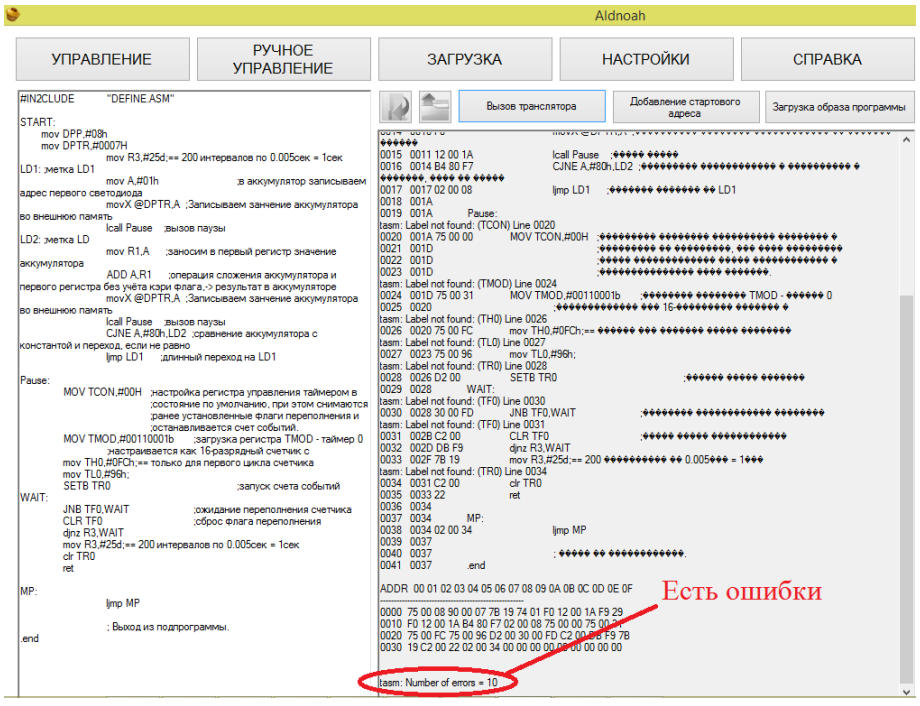


Рисунок 5.10 – Результат удачной трансляции в программе «Aldonah»

При успешной трансляции добавляется стартовый адрес нажатием на кнопку «Добавление стартового адреса» , результат добавления стартового адреса представлен на рисунке 5.11. Где оповещают какой стартовый адрес был выбран.

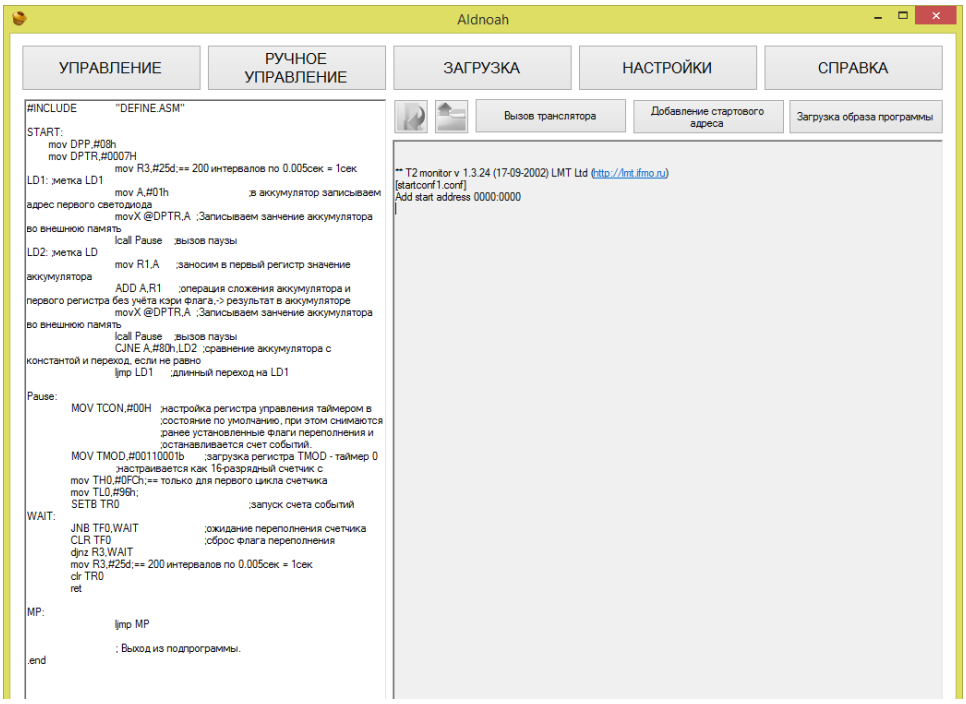


Рисунок 5.11 – Результат добавления стартового адреса.

Далее осуществляется загрузка программы нажатием на кнопку «Загрузка образа программы». Через 1-2 секунды после нажатия кнопки «загрузка образа программы» в стенде производят загрузку нажатием на кнопку «SW2» находящейся на стенде. Результат выполнения загрузки представлен на рисунке 5.12.

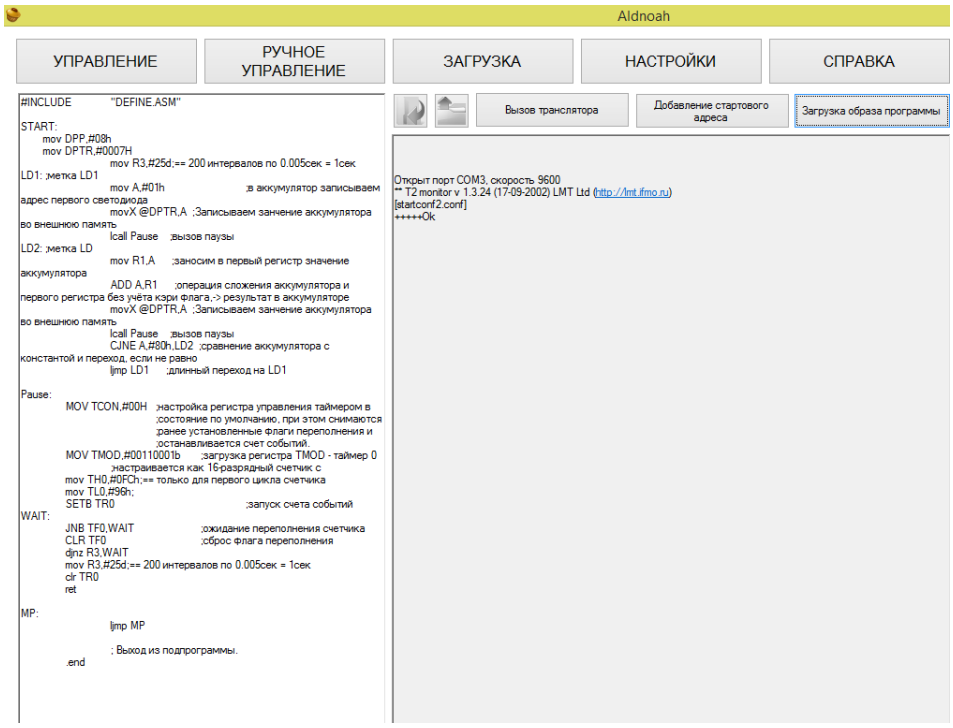


Рисунок 5.12 – Результат загрузки программы.

Примечание: при трансляции программ могут возникать ошибки связанные с особенностью текстового редактора (окно «1» рисунок 5.8). Трансляция требует что бы, код программы оканчивался символом перевода строки. Команды подключения дополнительных файлов и метки располагались в начале новой строки, а команды выполняющие операции присваивания, логические операции и т.д. должны располагаться в новой строке после табуляции.

6 Управление периферийными устройствами стенда SDK-1.1.

6.1 Средства управления линейкой светодиодов стенда

6.1.1 Схема подключения светодиодов

Светодиоды линейки подключаются через программируемую логическую интегральную схему (ПЛИС, EPM3064ATC100), которая показана на схеме ООО «ЛМТ» , Стенд учебно-лабораторный SDK1.1R4 в приложении 3 (Для более подробного изучении схемы, она предоставлена в цифровом формате на диске которой прилагается с отчётом).

Аналог схемы подключения светодиодов представлен на рисунке 6.1.

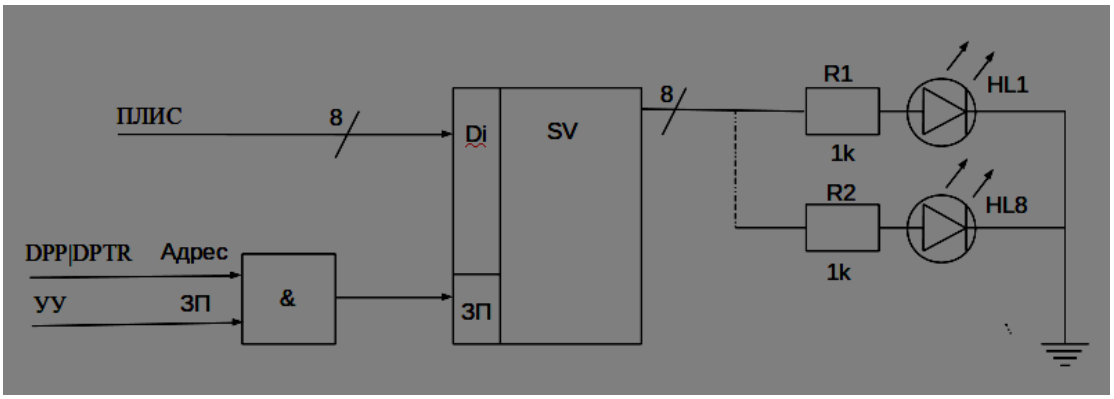


Рисунок 6.1- Схема подключения светодиодов

Регистр RGSV отображает функцию хранения при записи состояния на светодиоды (VD1-VD8). Включение светодиода осуществляется выдачей

сигнала высокого уровня (U^1). Резисторы (R1-R8) применены для ограничения тока через диод (а ток определяет яркость свечения).

Регистр RGSV предназначен для сохранения состояния, записываемого по линиям Di . Адрес регистра определен, принятым в стенде распределением адресов, и соответствует коду 080007h. Сигнал записи(ЗП) формируется при адресном обращении к регистру по записи.

6.1.2 Программное управление линейкой светодиодов

Для программного управления регистр RGSV и светодиоды отображаются через программную модель. В программной модели фиксируется адрес обращения и формат посылок.

Программная модель представлена в таблице 6.1.1

Таблица 6.1- Програмная модель RGSV

Наименование	Адрес объекта	Содержимое объекта(регистра)
Регистр RGSV	080007	Биты управления светодиодами(D0-D7).

Примечание.

1. Подача логической «1» (U^1)зажигает светодиод.
2. Линии D0-D7 соответственно управляют светодиодами VD1-VD8

Доступ к регистру RGSV, обеспечивается записью кода страницы 08h в регистр расширения DPP (84h) и кода ячейки(0007h) в двухбайтовый регистр DPTR (DPH - старший байт (00h),DPL- младший байт(07h)).

Сигнал ЗАП формируется при обращении из команды movx @DPTR, A , где в A находится адрес светодиода.

Пример программы управления светодиодами представлен в приложении Б.

Приведенная программа обеспечивает возможность загрузкой кодом регистра А, включить кодированный светодиод VD1. Меняя загрузку аккумулятора можно включать любое сочетание диода. При выполнение программы диод остаётся включенным , до записи новой программы.

Программный модуль подлежит трансляции в соответствии с разделом 4.1, загрузки в память микроконтроллера.

6.2 Управление таймером/счётчиком

Микроконтроллер ADuC842 имеет три 16-разрядных таймера-счетчика: Таймер 0, Таймер 1 и Таймер 2. Структура и режимы работы таймеров-счетчиков соответствуют общим принципам архитектуры MCS-51. Каждый таймер-счетчик содержит по два 8-битных регистра **THx** и **TLx** (x = 0, 1, и 2).

6.2.1 Таймеры — счётчики 0 и 1

Каждый таймер содержит два 8 — битных регистра .

TH0 и TL0 — старший и младший байт Таймера 0. SFR адрес — 0x8C и 0x8A соответственно.

TH1 и TL1 — старший и младший байт Таймера 0. SFR адрес — 0x8D и 0x8B соответственно.

При работе в качестве таймера содержимое Tlx инкрементируется в каждом машинном цикле. При работе в качестве счётчика содержимое Tlx инкрементируется под воздействием перехода из 1 в 0 внешнего входного сигнала, подаваемый на соответствующий (T0,T1) вход микроконтроллера. Для гарантированного прочтения входного считываемого сигнала он должен быть подан как минимум в течении одного машинного цикла. В режиме счетчика могут работать только таймер 0 и таймер 1.

Управление таймерами — счетчиками производятся при помощи регистров специального назначения (SFR) : TCON и TMOD, а текущее

значения счётчиков доступны через пары регистров Tlx/Thx, где x- номер таймера.

Регистр TMOD задает режимы работы таймеров, адрес 089h. Значение после подачи питания 00h. Регистр не имеет битовой адресации. Описание бит регистра приведено в таблице 1 приложения В .

Режим работа таймера определяется комбинациями его битов M0/M1. Они определяет по четыре режима работы для каждого из таймеров.

Для управления таймерами используется регистр TCON, назначение битов которого приведено в таблице 2 приложения В. Заметим , что четыре младших бита этого регистра предназначены не для управления таймерами, а для выбора сигнала прерывания, поступающих на выводы INT0 и INT1 микросхемы.

Регистр TCON является SFR регистром, его адрес 0x88. Значение после подачи питания 0x00. Регистр имеет битовую адресацию.

Работа таймеров в режиме 0 -для выбора данного режима следует установить биты M0=0 и M1=0. В режиме 0 таймеры работают в режиме 13-разрядного счета, при этом в регистре Tlx используется 5 разрядов , а в регистре Thx — 8 разрядов. Старшие 3 бита регистра Tlx не определены и игнорируются. Установка запускающего таймер флага Trx не очищает эти регистры. Работе таймера 0 соответствует рисунок 6.2.

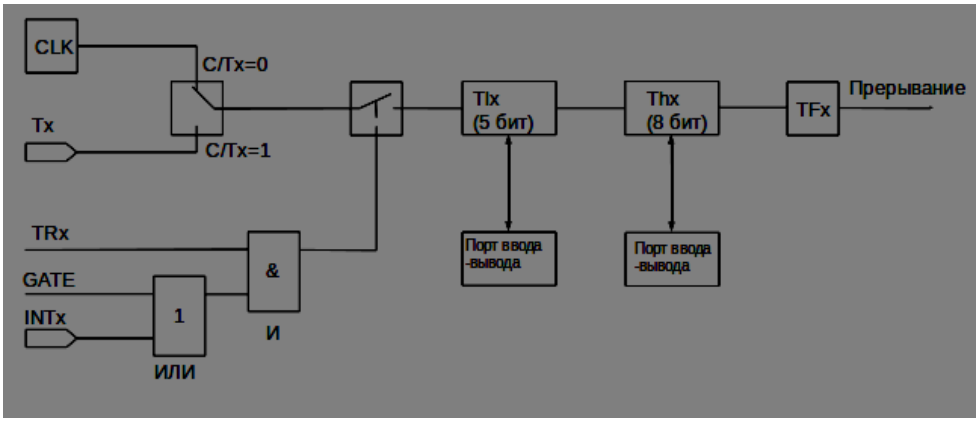


Рисунок 6.2- структурная схема работы таймеров 0 и 1 в режиме 0.

Переполнение таймера происходит , когда единицы в каждом разряде Thx и Tlx меняются нулями, при этом флаг переполнения TFx устанавливается в единицу. То есть при достижение максимального значение счета 8191 регистры Thx и Tlx переходят в значение 0с одновременной установкой флагов переполнения TFx в регистре TCON.

Работа таймеров в режиме 1 - для выбора первого режима следует установить биты M0=0 и M1=1 регистра TMOD. В режиме 1 оба таймера работают в режиме 16 разрядного счета. Режим 1 похож на режим 0 , за исключением того , что в реистрах таймера используются все 16 бит. В этом режиме Thx и Tlx также включены друг за другом. Работе таймеров в режиме 1 соответствует рисунок 6.3.

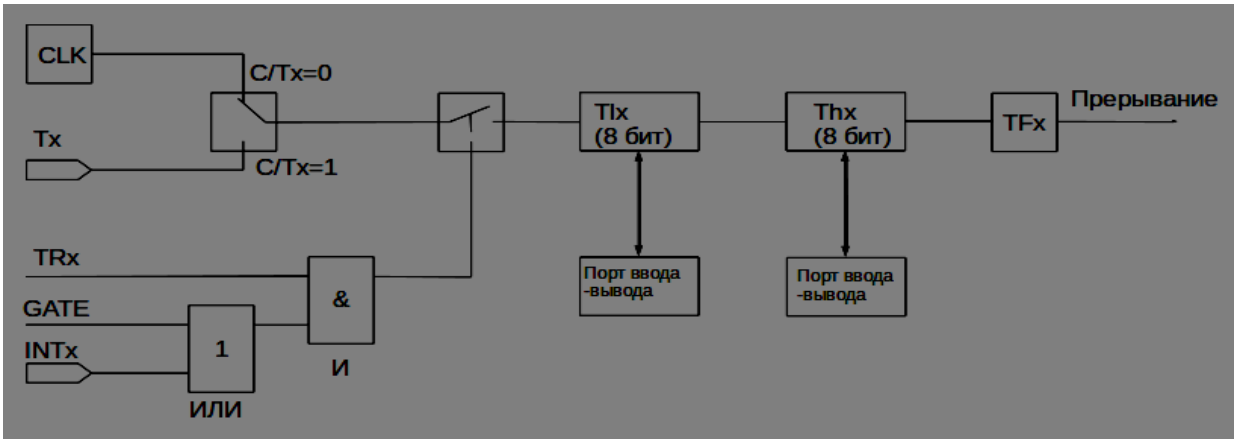


Рисунок 6.3 - структурная схема работы таймеров 0 и 1 в режиме 1.

Переполнение таймера происходит , когда единицы в каждом разряде Thx и Tlx меняются нулями, при этом флаг переполнения TFx устанавливается в единицу. То есть при достижение максимального значение счета 65535 регистры Thx и Tlx переходят в значение 0с одновременной установкой флагов переполнения TFx в регистре TCON.

Работа таймеров в режиме 2 - для выбора второго режима следует установить биты M1=1 и M0=0. В режиме 2 оба таймера работают в режиме 8-разрядного счета с автозагрузкой. Переполнение регистра Tlx не только устанавливает флаг TFx, но и загружает регистр Tlx содержимым регистра Thx, который предварительно инициализируется программно.

Работе таймеров 0 и 1 в режиме 2 соответствует рисунок 6.4.

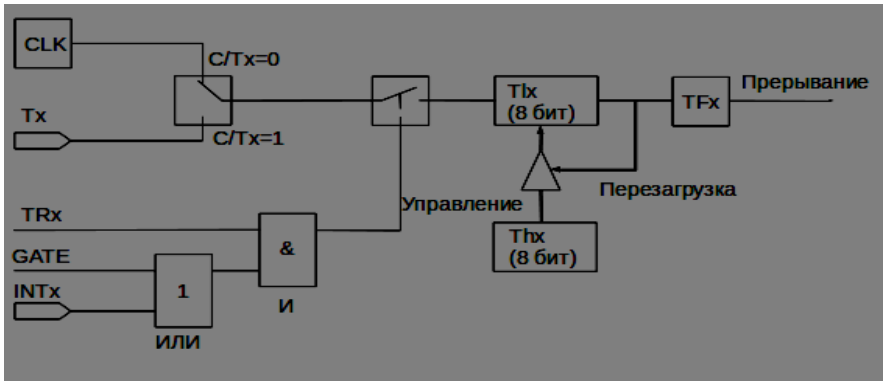


Рисунок 6.4 - структурная схема работы таймеров 0 и 1 в режиме 2.

Работа таймеров в режиме 3 - чтобы выбрать данный режим работы необходимо установить биты M1=1 и M0=1. Для таймеров 0 и 1 третий режим различается. В режиме 3 таймер 1 остановлен, то есть просто хранит свое значение. Таймер 0 в режиме 3 работает как два независимых счетчика, это показано на рисунке 6.5.

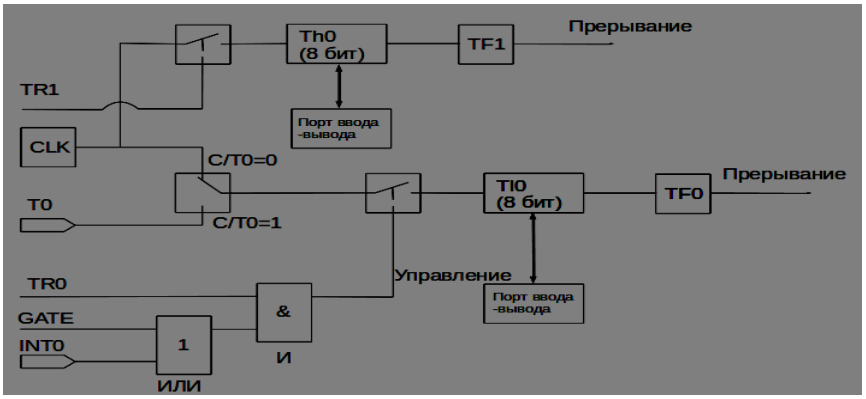


Рисунок 6.5 - структурная схема работы таймеров 0 в режиме 3

Регистр TL0 использует биты управления таймера 0: C/T0, GATE, TR0 и TF0. Регистр TH0 работает только в режиме таймера, и использует биты TR1 и TF1 таймера 1.

6.2.2 Таймер — счётчик 2

Таймер 2 , обозначаемый T/C2 , используется как 16- разрядный таймер-счетчик или как генератор частоты приема/передачи. Он управляется регистром T2CON. T2CON относится к SFR регистрам, его адрес C8H, значение после подачи питания 0x00. Регистр имеет битовую адресацию. Назначение битов этого регистра приведено в таблице 3 приложения В.

Режим работы таймера-счетчика 2 устанавливается сочетаниями битов регистра T2CON так, как это показано в таблице 4 приложения В.

Таймер 2 использует регистры захвата/перезагрузки (RCAP2L и RCAP2H) и регистры (TL2 и TH2).

TH2 и TL2 — старший и младший байт Таймера 0. SFR адрес — 0xCDh и 0xCCh соответственно.

RCAP2H и RCAP2L — старший и младший байт захвата/перезагрузки. SFR адрес — Cbh , Cah соответственно.

Примечание: в ходе прохождения производственной практики, было пропущено исследование работы таймера 2 в режиме генератора частоты приемпередатчика.

Работа таймера 2 в режиме автозагрузки - логика работы таймера 2 в режиме автозагрузки представлена на рисунке 6.6. 16 — битное значение перезагрузки записывается в регистры RCAP2L(младшая часть) и RCAP2H(старшая часть). В регистры TL2 и TH2 перед началом работы записываются те же значения. После окончания счета устанавливается флаг прерывания TF2, который вызывает перезагрузку регистров TL2 и TH2 значениями из регистров RCAP2L и RCAP2H.

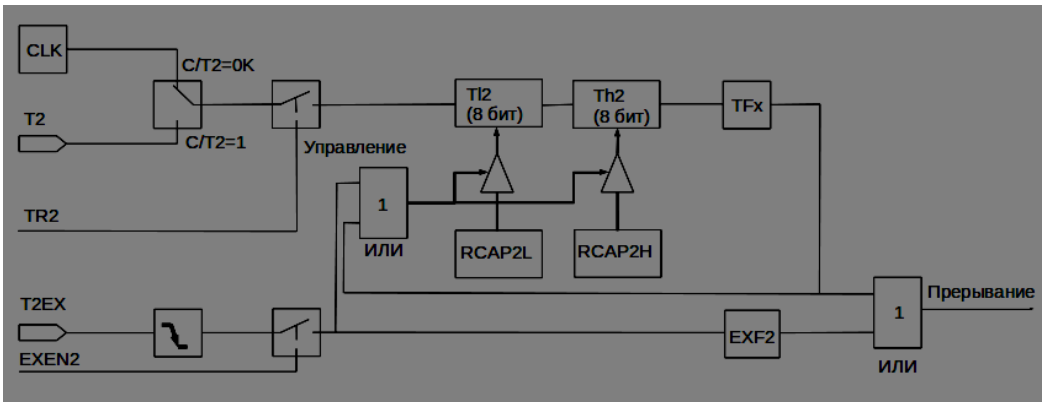


Рисунок 6.6 - структурная схема работы таймера 2 в режиме автозагрузки.

Перезагрузка может быть вызвана также переходом 1 -0 внешнего сигнала, поступающего в вывод T2EX микросхемы, при этом бит EXEN2. Должен быть установлен.

Работа таймера 2 в режиме захвата - логика работы таймера 2 в режиме захвата приведена на рисунке 6.7.

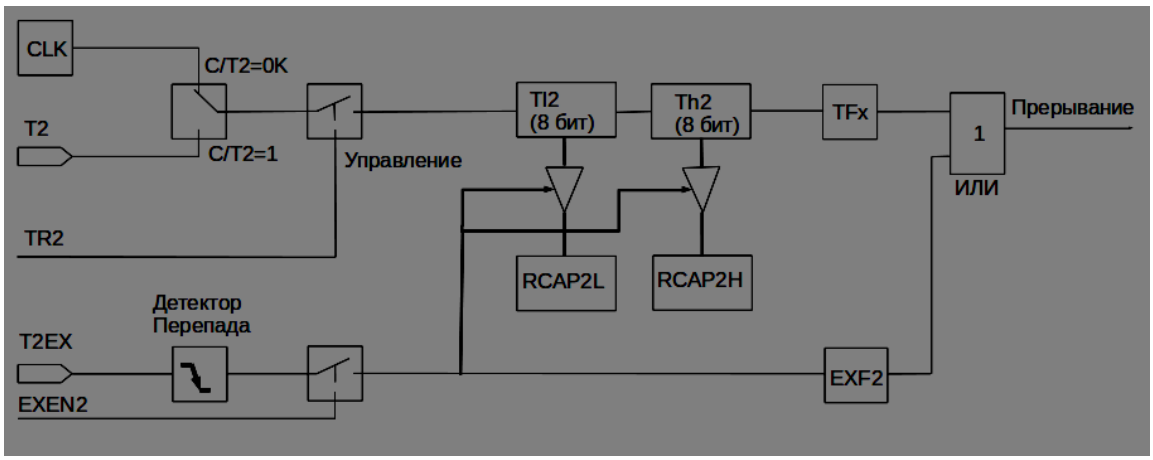


Рисунок 6.7 - структурная схема работы таймера 2 в режиме захвата.

Захват производится переходом 1 -0 внешнего сигнала, поступающего на выход T2EX микросхемы, если бит EXEN2 установлен. При этом значение регистров счетчика запоминаются в регистрах захвата RCAP2x . Одновременно в регистре T2CON устанавливается внешний флаг таймера EXF2, который может быть использован для генерирования прерывания так же, как и флаг переполнения TF2.

6.2.3 Пример программы работы таймера.

Приведенная программа в приложении В-программа для таймера, запускает таймер и по истечению времени, инвертирует состояние светодиодов (переводит состояние светодиодов из 1 в 0 и наоборот). Всё происходит в бесконечном цикле.

При установке таймера , не обходимо учитывать тактовую частоту процессора. В SDK 1.1 частота процессора имеет возможность работать на частотах от 0.131072 МГц до 16.777216 Мгц. По умолчанию процессора работает на частоте 2.097152 МГц.

В примере используется таймер 2 в режиме автозагрузки. Тактовую частоту процессора оставили по умолчанию. В таком случаем на выход таймера буду подаваться импульсы с периодом 1/2097152=0,476 мкс. Таймер 2 использует 16 - битный счетчик. Учитывая , что регистры TL2 =00h и TH2 =00h и регистры перезагрузки обнулены. То , что бы счетчик таймера переполнился , на вход таймера должно быть подано 65536 импульсов. Зная периодичность импульсов можно рассчитать время , через которое переполнится счетчик.

Умножая период импульсов на количество необходимых импульсов , получим время переполнения таймера.

0,476*10^(-6)*65536=0.031 с.

В программе используется таймер с паузой в 1 секунду. Что бы получить паузу в 1 , необходимо рассчитать количество раз переполнения таймера.

1/0.031=32.154

И так таймер должен переполнится 32 раза , что бы выдержать паузу в 1 секунду.

Так же необходимо указать, что при обработке переполнения таймера используется прерывание. Вектор прерывания от таймера 2 имеет адрес 202bh.

6.3. Средства управления ЖКИ

6.3.1 Регистры управление ЖКИ

Модуль ЖКИ встроен в контроллер (БИС) и имеет два 8 битных регистра: регистр команд (IR) и регистр данных (DR).

Регистр команд хранит коды таких операции, как очистка дисплея, перемещение курсора а также информацию об адресах памяти отображаемых данных (DDRAM) и генератора символов (CGRAM). В регистр команд можно только записывать информацию из микропроцессора.

Регистр данных временно хранит и получает данные из регистра DATA_IND , предназначенные для записи или чтения DDRAM или CGRAM. Когда адресная информация записывается в регистр команд, данные из DDRAM или CGRAM сохраняются в регистре данных.

Регистр IR имеет адрес – 080006H.

Значение после сброса xxxxx010B.

Регистр IR получает данные из регистра C_IND.

Заметим , что биты 4-7 не используются, и игнорируются.

Регистр DR имеет адрес – 080001H.

Регистр DR получает данные из регистра DATA_IND.

Назначение бит регистра Data_IND представлены в таблице 5 приложения Г.

6.3.3 Описание функций

За выполнением внутренних операций БИС следит флаг BF , если флаг равен 1 , то значит БИС занята, и следующая команда не может быть принята.

Если биты RS=0 и R/W =1, содержимое флага занятости передаётся в бит DB7. Следующая команда может быть записана только при значении флага занятости, равном 0.

Счетчик адреса (AC)

Счетчик адреса (AC) назначает адреса и DDRAM, и CGRAM.

В ЖКИ присутвует память данных (DDRAM). Эта память используется для хранения данных , выводимых на дисплей. Один символ представлен в виде 8 битного кода. Объём памяти состовляет 80x8 битов или 80 символов.

В таблице 6 приложения Г приведена схема соответсвтия между адресами DDRAM и позициями ЖКИ.

Генератор символов, встроенный в ПЗУ (CGROM)

CGROM генерирует символы размером 5×8 или 5×10 точек на основе 8-битных кодов символов.

В CGRAM пользователь может программно генерировать символы.

Можно определить 8 символов размером 5×8 точек и 4 символа размером 5×10 точек. Коды символов нужно записывать в DDRAM по адресам, отображенным в таблице. В таблице также показано, как отобразить символ, хранящийся в CGROM.

Образы символов представлены в таблице 7 приложения Г –образы символов.

Таблица команд представлена в таблице 8 приложения Г.

6.3.4.Работа с ЖКИ

Работать с ЖКИ достаточно просто, так как оперировать вам придётся всего регистрами. Необходимо помнить, что контроллер ЖКИ в SDK-1.1 подключен не напрямую к микроконтроллеру, а через расширитель портов, выполненный на

базе ПЛИС. За связь с ЖКИ, в расширителе портов отвечают первые два регистра:

1. DATA_IND отвечает за выдачу информации на шину данных (через этот регистр можно передавать команды контроллеру и данные;

2. C_IND отвечает за формирование сигналов E, R/W и RS, позволяющих регулировать обмен на шине между расширителем портов и контроллером ЖКИ.

Вся работа с ЖКИ сводится к нескольким простым вещам:

1. Первым шагом вы записываете команду или данные (коды выводимых символов) в регистр DATA_IND расширителя портов. После этого, содержимое этого регистра появляется на шине данных контроллера ЖКИ (DB0..DB7). Контроллер на эти данные естественно не реагирует, так как сигнал ‘E’ (Enable) нами еще не выставлен в активный уровень (логическая ‘1’).

2. Вторым шагом, вы должны разрешить работу с шиной с помощью сигнала ‘E’ (логическая ‘1’), выставить сигнал записи (логический ‘0’ на линии ‘W’) и указать тип регистра, с которым вы будете работать в контроллере ЖКИ на линии RS. Если вы передаёте данные, то на сигнал RS нужно подать ‘1’, если команду, то ‘0’.

6.3.5.Пример программы

При использовании ЖКИ необходимо помнить , что время выполнения команд контроллером ЖКИ не равно нулю. Следовательно необходимо устанавливать задержки после выполнения команд. Величину времени выполнения различных команд можно посмотреть в таблице 11 приложения Г.

Изначально курсор находится в позиции 00 в видеопамяти, ЖКИ способен отобразить 16 символов в верхней строке и 16 символов в нижней строке.

Приведённая программа в приложении В – программа для ЖКИ выполняет следующее в чёткой последовательности:

1)Вывод на дисплей курсора;

2)Вывод на дисплей элемента «1» в первой позиции верхней строки;

3)вывод на дисплей элемента «2» во второй позиции;

Между выполнениями выше перечисленных операции стоит пауза в размере 2 секунд.

Что бы отобразить курсор необходимо в регистр данных(DATA_IND) записать 0Eh. Что соответствует команде Дисплей ON/OFF смотрите таблицу 5.3.8. Но перед тем как записать данные в регистр DATA_IND нужно выбрать его в ПЛИС . После того как выбрали регистр DATA_IND и записали в него данные, с помощью регистра C_IND указываем ЖКИ как поступить с этими данными для этого в сначала нужно переключить использованный регистр ПЛИСы с DATA_IND на C_IND. И записать в него следующее 01h. Что соответствует RS=0,R/W=0,E=1. Что бы контроллер ЖКИ успешно обработал команду необходимо поставить паузу в размере 39 мс(в программе используется пауза в размере 2 с.).

Для вывода на дисплей какого либо элемента нужно установить курсор в видеопамяти в нужную позицию и выбрать элемент в знакогенераторе который хотим записать в данную ячейку видеопамяти. В данной программе изменения позиции курсора не использовалось , следовательно элементы должны будут записаться в последовательные ячейки памяти.

6.4.Средства управления клавиатурой

6.4.1 Схема подключения клавиатуры

Клавиатурная панель подключаются через программируемую логическую интегральную схему (ПЛИС, EPM3064ATC100) , которая показана на схеме ООО «ЛМТ», Стенд учебно-лабораторный SDK1.1R4. Аналог схемы подключения клавиатурной панели представлен на рисунке 6.8.

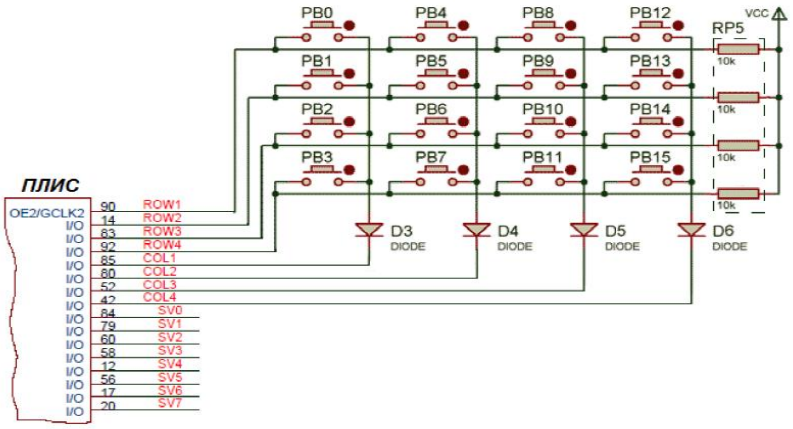


Рисунок 6.8 - Схема подключения матричной клавиатуры.

Доступ к столбцам и строкам клавиатуры организован как чтение/запись регистра ПЛИС (адрес 0x080000): младшие 4 бита соответствуют 4 столбцам (COL1…COL4), старшие 4 бита –строки (ROW1…ROW4).

На рисунке 6.9. представлена принципиальная электрическая схема матричной клавиатуры.

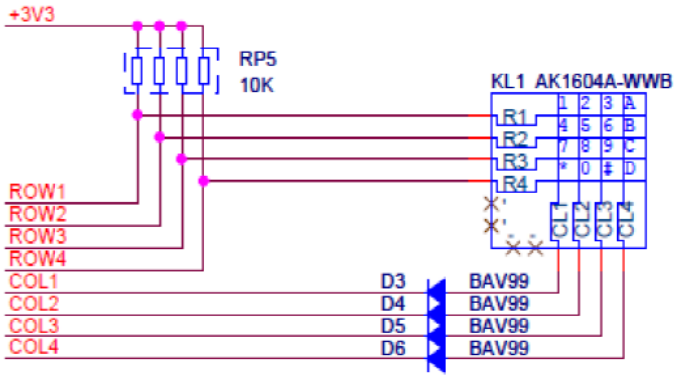


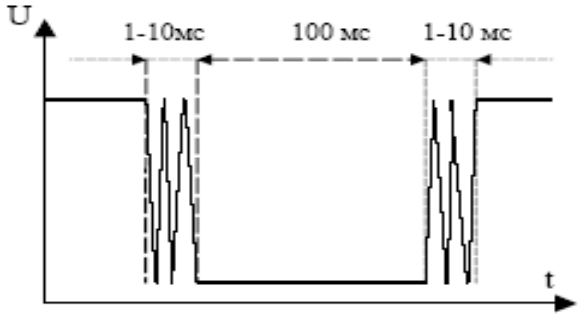
Рисунок 6.9 - Матричная клавиатура на принципиальной электрической схеме SDK1.1

Из схемы видно, что потенциал на выходных портах ROW остаётся высоким , если все ключи разомкнуты, следовательно соответствует логической единицы. Если замкнуть один из ключей, то есть нажать клавишу и на соответствующую вертикальную линию подать сигнал низкого уровня, то потенциал горизонтальной линии, с которой соединена это кнопка тоже станет низким .

Таким образом, полный опрос матричный клавиатуры включает

последовательное обнуление младших четырех битов регистра КВ с анализом на каждом шаге старших разрядов регистра.

Необходимо отметить , что в работе с клавиатурой можно наткнуться на «Эффект дребезга». То есть при нажатии на клавишу напряжение на не сразу устанавливается в 0В.А скачет в течении (0-10)мс, пока надёжно не замкнётся. Такая же ситуация возникает при отпускании клавиши, напряжение скачет , пока не установится логическая 1.



6.10- Диаграмма значения напряжения при нажатии кнопки

Поскольку процессор обладает высоким быстродействием, то он может воспринять эти скачки напряжения за несколько нажатий. Для программного устранения влияния «дребезга» используется задержка. После того, как в результате сканирования обнаружится «0» в регистре ROW, сканирование прекращается и производится задержка на некоторое время. После этого сканируется тот же столбец и, если на том же месте регистра ROW обнаружен «0», то фиксируется нажатие клавиши. После этого через некоторое время, достаточное для отпускания клавиши, еще раз проверяется тот же столбец. Если состояние линии изменилось, то фиксируется отпускание клавиши и продолжается сканирование клавиатуры. Если клавиша все еще нажата, то производится задержка на время перед повтором символа, и если состояние регистра не изменилось, то в буфер клавиатуры повторно заносится символ. После этого, пока клавиша не будет отпущена, в буфер заносится код клавиши через промежутки времени, определяемые скоростью повтора символа.

6.4.2-Пример работы с клавиатурой

Доступ к столбцам и строкам клавиатуры организован как чтение/запись регистра (КВ) ПЛИС (адрес 0x080000): младшие 4 бита соответствуют 4 столбцам (COL1…COL4), старшие 4 бита –строки (ROW1…ROW4).

В приложении Д представлен листинг программы обрабатывающий нажатия из последнего столбца, который соответствует клавишам :1,4,7,*.

6.5.Средства управления ЦАПом

Микроконтроллер ADuC842 содержит на своём кристалле два 12-битных ЦАПа. Аналог схемы подключения показан на рисунке 6.11.

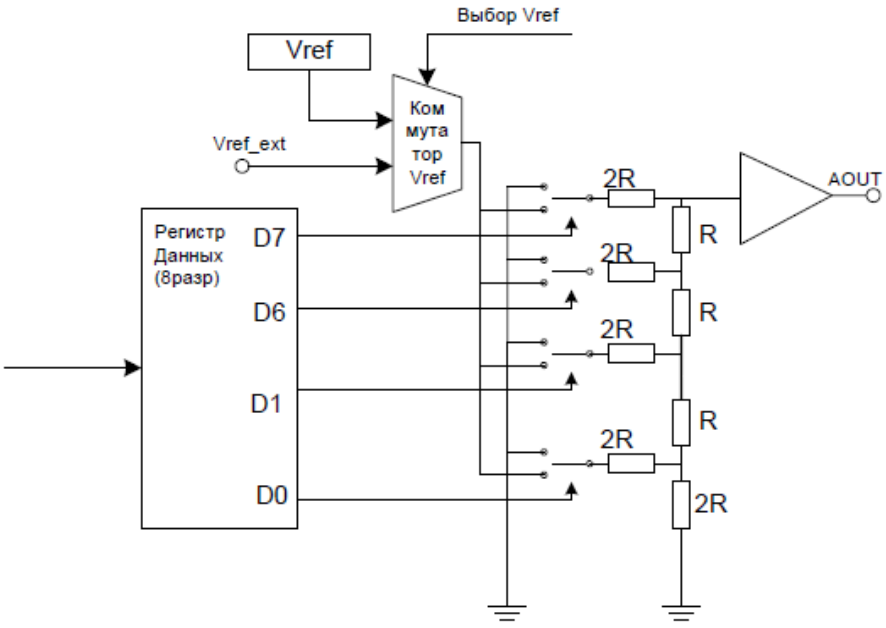


Рисунок 6.11 - Аналог схемы подключения ЦАПа.

6.5.1.Программное управление

Для управления ЦАП предназначен регистр DACCON.Данный регистр не имеет битовой адресации, при подаче питания значение по умолчанию 04h.

В таблице 9 приложения Е представлена программная модель DACCON.

Входными данными ЦАП служат регистры специального назначения: DAC0H/DAC1H, содержащие старшие 4 байта и DAC0L/DAC1L, содержащие

младшие восемь байт соответственно для нулевого и первого канала.

При восьмибитной работе байт, записанный в регистры DAC0L/DAC1L, автоматически направляется в верхнюю часть 12-битного регистра ЦАП.

Управление ЦАПом, достаточно просто.

Изначально необходимо заполнить регистры входных данных , в соответствии с режимом который будет использоваться в ЦАПе.

Далее задать параметры ЦАПов используя регистр DACCON.

Примечание: что бы ЦАП начал преобразовывать цифровой сигнал в аналоговый, нужно установить бить SYNC.

В приложении Е представлена программа , демонстрирующая работу ЦАПа.

В демонстрационной программе выбирается 0-ой ЦАП, 12 –битный режим, диапазон от 0 до 2,5 В. на вход которого подаётся 2.5 В , что соответствует коду 0FFFh.

6.6 Регистр управления параллельным портом ENA

Для управления параллельным портом, предназначен регистр ENA , который относится к ПЛИС. Адрес регистра ENA 080004h, значение при включении стенда 0000000h. Назначение битов регистра ENA приведено в таблице 10 приложения Е.

Регистр данных параллельного порта EXT_LO позволяет считывать и записывать биты 0..7 параллельного порта. Для того чтобы из регистра попали на выход , необходимо установить бит EN_LO в логическую «1»(смотрите назначение битов регистра ENA).Для чтения данных необходимо установить этот битв логический «0». Адрес регистра EXT_LO 080002h. Значение после сброса 00h.

Таблица 6.2-регистр данных параллельного порта EXT_LO.

7	6	5	4	3	2	1	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
D7	D6	D5	D4	D3	D2	D1	D0

Регистр данных параллельного порта EXT_HI позволяет считывать и записывать биты 0..7 параллельного порта. Для того чтобы из регистра попали на выход , необходимо установить бит EN_HI в логическую «1»(смотрите назначение битов регистра ENA).Для чтения данных необходимо установить этот битв логический «0». Адрес регистра EXT_HI 080003h. Значение после сброса 00h.

Таблица 6.3-регистр данных параллельного порта EXT_HI.

7	6	5	4	3	2	1	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
D7	D6	D5	D4	D3	D2	D1	D0

6.6.1.Программное управление параллельным портом

В приложении Ж представлена демонстрационная программа считывания и записи данных в порт. Для работы программы необходимо подключить к порту устройство расширяющее возможности SDK.

Программа считывает состояние тумблеров и приводит зажигает соответствующий светодиод на расширенном устройстве.

Заключение

По окончанию практики, был получен опыт работы с контроллерами и с периферийными устройствами. Были продемонстрированы и использованы на практике полученные знания во время обучения в университете. Ознакомлен с работой чтения принципиальных электрических схем,

применения полученных знания в программном коде.

Список использованной литературы

1) Положение об организации и проведении практик студентов, обучающихся в ТУСУРе, утверждено первым проректором, проректором по УР 20.11.2014 г.

2) . Торгонский Л.А. Производственная практика (Электронный ресурс): методические указания/ Л.А.Торгонский; Томский государственный университет систем управления и радиоэлектроники (Томск). - Электрон. текстовые дан. - Томск :2015 - 32 с.

3) Образовательный стандарт ТУСУР (ОС).Общие требования и правила оформаления.01 – 2013. Томск 2014.– 57с.

4)Технология и автоматизация производства радиоэлектронной аппаратуры: Учебник для вузов/ И. П. Бушминский, О.Ш. Даутов, А. П. Достанко и др.; Под ред. А.П. Достанко , Ш.М. Чабдарова. – М.: Радио и связь, 1989. – 624 с.: ил.

5) Учебный стенд , руководство пользователя – «ЛМТ» ,2001г – 99стр..

6)Datasheets Aduc841,842,843 –AnalogDevices , 2003г-88стр.

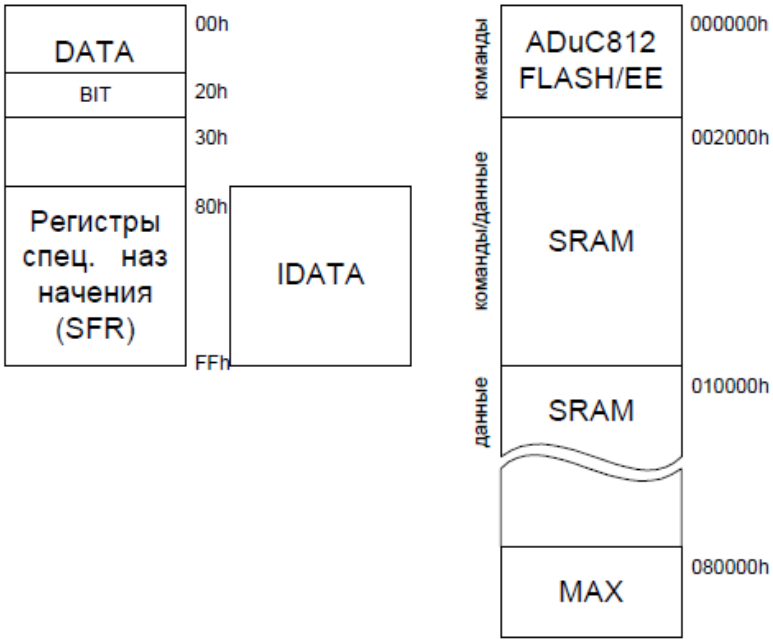
7)Чугоевский К.И. Компьютерное управление автоматизированной установкой травления печатных плат./Дипломная работа, Тусур, КИБЭВС, Томск ,2015.-90стр.

Приложение А

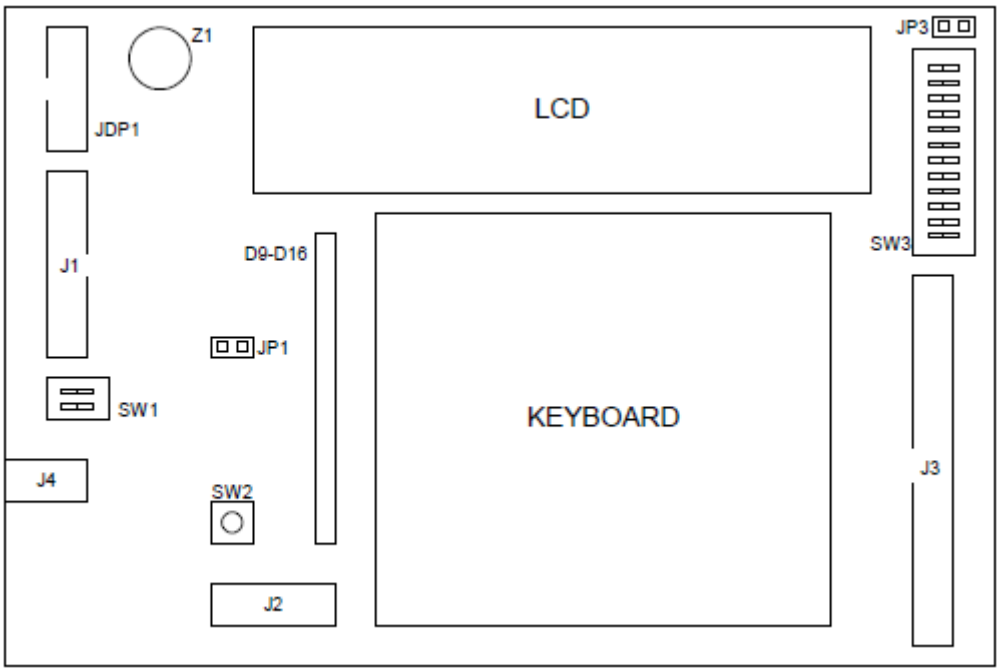
(Обязательное)

Ресурсы стенда SDK-1.1.

Распределение памяти в стенде SDK-1.1.



Схематическое изображение стенда SDK-1.1



Приложение Б

(Обязательное)

Управление линейкой светодиодов

Программа для светодиодов

```
#INCLUDE "DEFINE.ASM"  ; подключение таблицы имён и кодов SFR,
;применёных в программе
START:
 mov DPP,#08h     ;настройка страницы ПЛИС
 mov DPTR,#0007H         ;выбор ячейки(регистра) в старнице
 mov A,#01h                  ;в аккумулятор записываем адрес первого светодиода
movX  @DPTR,A  ;Записываем занчение аккумулятора во внешнюю память
loop:
    ljmp loop
.end
```

Приложение В

(Обязательное)

Управление таймерами

Таблица 1 – Описание бит регистра TMOD

Номер бита	Имя бита	Описание
7	GATE	Бит управления Тамером 1. При Gate=1 для работы необходимо условие TR1=1 и INT1=1. При Gate =0 Таймер 1 работает всегда, когда TR1=1.
6	C/T1	Бит выбора событий для Таймера 1. При C/T1=1 он работает как счётчик (вход с внешнего вывода T1), при C/T1=0 — как таймер (вход с внутреннего генератора).
5	M1	M1, M0 биты определяют режим работы Таймера 1 0 0 13 разрядный счётчик 0 1 16 разрядный счетчик 1 0 8- битный таймер-счетчик с автозагрузкой 1 1 Таймер-счетчик 1 остановлен.
4	M0	
3	GATE	Бит управления Тамером 0. При Gate=1 для работы необходимо условие TR0=1 и INT1=1. При Gate =0 Таймер 1 работает всегда, когда TR0=1.
2	C/T0	Бит выбора событий для Таймера 0. При C/T0=1 он работает как счётчик (вход с внешнего вывода T1), при C/T1=0 — как таймер (вход с внутреннего генератора).
1	M1	M1, M0 биты определяют режим работы Таймера 0 0 0 13 разрядный счётчик 0 1 16 разрядный счетчик 1 0 8- битный таймер-счетчик с автозагрузкой 1 1 Таймер-счетчик 1 остановлен.
0	M0	

Таблица 2 – Описание бит регистра TCON

Номер бита	Имя бита	описание
7	TF1	Флаг переполнения таймера 1. Устанавливаются аппаратно. Сбрасывается аппаратно при обслуживании прерывания.
6	TR1	Бит управления (запуска) таймера 1. Устанавливается / сбрасывается программно для запуска/останова таймера.
5	TF0	Флаг переполнения таймера 0. Устанавливаются аппаратно. Сбрасывается аппаратно при обслуживании прерывания.
4	TR0	Бит управления (запуска) таймера 0. Устанавливается / сбрасывается программно для запуска/останова таймера.
3	IE1	Флаг внешнего прерывания 1. Устанавливается аппаратно при срезе сигнала INT1 или если сигнал равен нулю, в зависимости от бита IT1. Сбрасывается аппаратно.
2	IT1	Флаг внешнего прерывания 1. Если установлен, прерывание 1 возникает при переходе 1-0 на выходе INT1. Если сброшен прерывание 0 возникает при низком уровне INT1.
1	IE0	Флаг внешнего прерывания 0. Устанавливается аппаратно при срезе сигнала INT0 или если сигнал равен нулю, в зависимости от бита IT0. Сбрасывается аппаратно.
0	IT0	Флаг внешнего прерывания 0. Если установлен, прерывание 0 возникает при переходе 1-0 на выходе INT0. Если сброшен прерывание 0 возникает при низком уровне INT0.

Таблица 3– Описание бит регистра T2CON

Номер бита	Имя бита	описание
7	TF2	Флаг переполнения. Устанавливается аппаратно, сбрасывается программно. Не устанавливается, если либо RCLK , либо TCLK установлены в 1.
6	EXF2	Внешний флаг таймера 2. Устанавливается аппаратно при защелкивании информации в регистрах захвата или при перезагрузке вследствие перехода 1-0 на выводе P1.1(T2EX) при EXEN2=1. Сбрасывается программно.
5	RCLK	Разрешение тактовых сигналов приема. Если установлен, таймер 2 используется для тактирования приема в режимах 1 и 3.
4	TCLK	Разрешение тактовых сигналов передачи. Если установлен , таймер 2 используется для тактирования передачи в режимах 1 и 3.
3	EXEN2	Разрешение внешнего сигнала T2EX. Устанавливается для разрешения захвата или перезагрузки вследствие перехода 1-0 на выводе P1.1(T2EX).
2	TR2	Бит запуска/останова таймера 2.
1	CNT2	Выбор режима работы. Если установлен, таймер 2 работает как счетчик сигналов на выводе T2. Если сброшен , таймер 2 работает как таймер.
0	CAP2	Выбор режима захвата/перезагрузки. Если установлен , разрешен захват по переходу 1- 0 на выводе P1.1 при EXEN2=1. Если сброшен , разрешается перезагрузка по переполнению или переходу 1 -0 на выводе P1.1 при EXEN2 =1. Бит игнорируется, если либо RCLK, либо TCLK установлены в 1.

Таблица 4-режимы работы таймера 2

RCLK или TCLK	CAP2	TR2	Режим
0	0	1	16-разрядный таймер-счетчик с перезагрузкой(автозагрузки).
0	1	1	16-разрядный таймер-счетчик с захватом.
1	X	1	Генератор частоты приемпередатчика
X	X	0	Отключен

Программа для таймера

```
#INCLUDE "DEFINE.ASM"
START:
;инициализация светодиодов
        mov DPP,#08h
        mov DPTR,#0007H
        mov A,#0FFh
;инициализация таймера
        lcall Init_Timer2
;переход в функцию с бесконечным циклом
        ljmp main
;---------------------------------------
```

```
        .ORG 202bh;== с этого адреса переход к подпрограмме обработчику
прерывания
        ljmp T2_int;== по переполнению от "таймера 2" - T2_int


;---------------------------------------


;---------------------------------------
main:
                movX @DPTR,A
                ljmp main
;---------------------------------------


Init_Timer2:
        mov R3,#32d;== 32 раза перполнения счетчика
        mov RCAP2H,#00h
        mov RCAP2L,#00h
        mov TH2,#00h
        mov TL2,#00h
        setb ET2;== разрешить прерывания от событий таймера2
        setb EA;== снять запрет со ВСЕХ прерываний
        setb TR2;== запустить счетчик-таймер2
        RET  ;== возврат из подпрограммы
```

```
;-----------------------------------------------


        ;================================================

T2_int:

        clr TF2;== обязательный сброс флага переполнения TF2

        djnz R3,vyhod;оператор проверки количества раз переполнения
счетчика

        mov R3,#32d;== секунда прошла – в счетчик прерываний снова 32



        xrl A,#0ffh

vyhod:

        RETI          ;== возврат из подпрограммы обработчика прерывания

;================================================

;-----------------------------------------------

MP:

                ljmp MP

                .end
```

Приложение Г

(Обязательное)

Управление ЖКИ

Таблица 5 – назначение бит регистра DATA_IND

Биты	Поле	Описание
0..7	D0..D7	Регистр DATA END позволяет устанавливать данные на шине данных ЖКИ и считывать их оттуда. Для организации взаимодействия с ЖКИ (формирования временных диаграмм чтения и записи) необходимо использование регистра C END.

Таблица 6 – Соответсвия между адресами DDRAM и позициями ЖКИ

Позиция на ЖКИ	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Адрес в DDRAM (верхнаяя строка)	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
Адрес в DDRAM (нижняя строка)	40	41	42	43	44	45	46	47	48	49	4A	4B	4C	4D	4E	4F

Таблица 7 - Образы символов

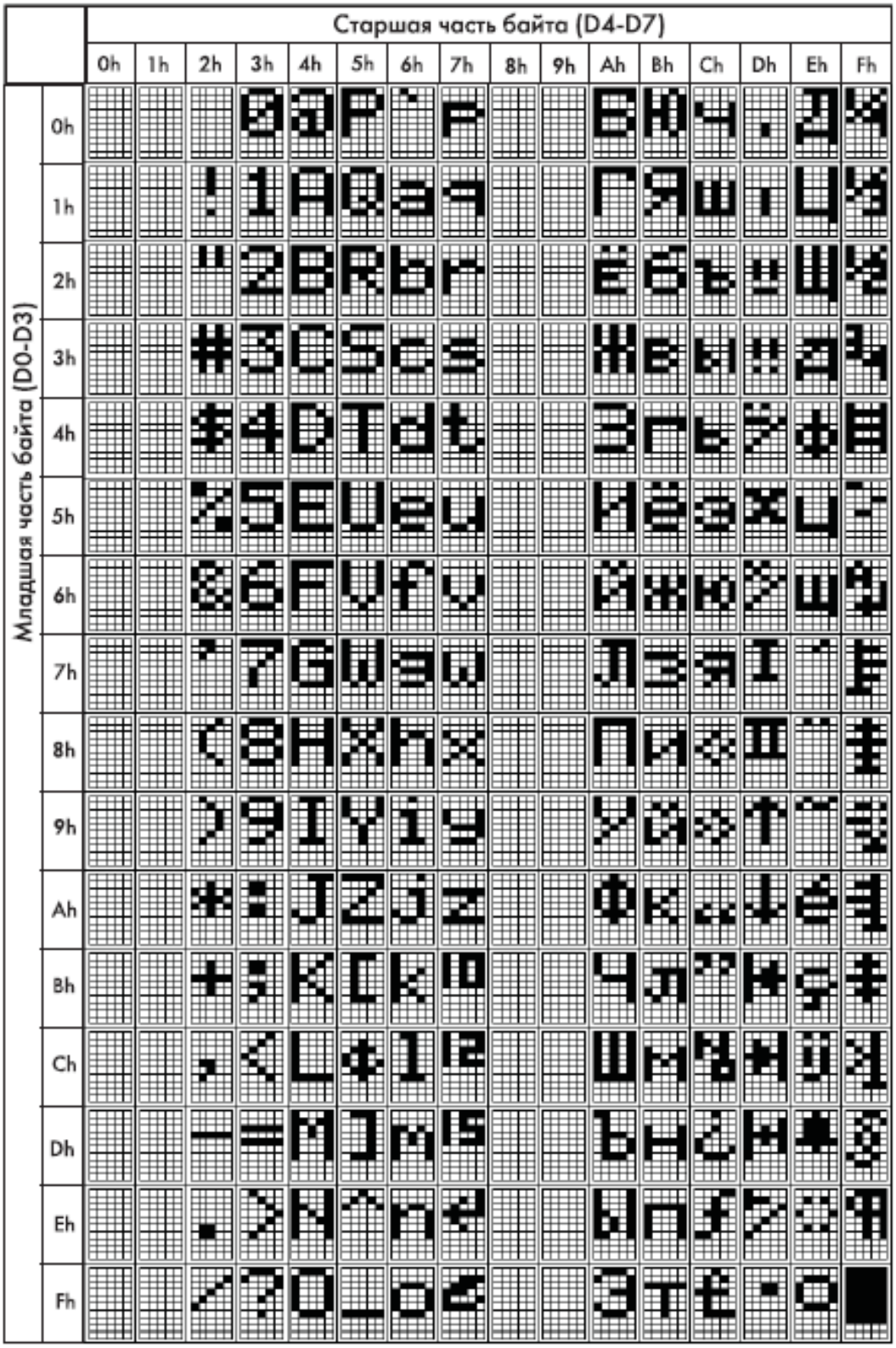


Таблица 8 – таблица команд

Команда	Код операции										Описание	Время выполнения(fosc =270Кгц)
	RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0		
Очистка экрана	0	0	0	0	0	0	0	0	0	1	Запись «00H» в DDRAM и установка адреса DDRAM на «00H» из AC.	1.53мс
Возврат в начало строки	0	0	0	0	0	0	0	0	1	*	Установка адреса DDRAM на «00H» из AC и возврат курсора начало строки, если он был смещен. Содержимое DDRAM не меняется.	1.53мс
Начальные установки	0	0	0	0	0	0	0	1	I/D	SH	Задает направление перемещения курсора и разрешает сдвиг сразу всех символов.	39мс
Дисплей ON/OFF	0	0	0	0	0	0	1	D	C	B	Устанавливает/отключает биты, отвечающие за режим дисплея (D), отображение курсора (C) , мерцание курсора (B).	39мс
Передвижение курсора по экрану	0	0	0	0	0	1	S/C	R/L	*	*	Установка бита движения курсора и смещения всех символов , указание напраления смещения без изменения данных в DDRAM	39мс
Функции установки	0	0	0	0	1	DL	N	F	*	*	Установка длины данных(DL:8/4-бит) , количества строк на дисплее(N:2-строки или 1) и размер символов (F:5x11 точек/ 5x8 точек).	39мс
Установка адреса CGRAM	0	0	0	1	AC5	AC4	AC3	AC2	AC1	AC0	Установка адреса CGRAM в счетчик адреса.	39мс
Установка адреса DDRAM	0	0	1	AC6	AC5	AC4	AC3	AC2	AC1	AC0	Установка адреса DDRAM в счетчик адреса.	39мс

Чтение флага занятости и адреса	1	0	BF	AC6	AC5	AC4	AC3	AC2	AC1	AC0	Прочитав флаг заняточти, можно определить занят контроллер выполнением внутренних операций. Также можно прочесть содержимое счетчика адреса.	0мс
Запись данных в память	1	0	D7	D6	D5	D4	D3	D2	D1	D0	Запись данных во внутреннюю память (DDRAM/CGRAM).	43мс
Чтение данных из памяти	1	1	D7	D6	D5	D4	D3	D2	D1	D0	Чтение данных из внутренней памяти (DDRAM/CGRAM).	43мкс

*-не имеет значение.

Листинг -Программа для ЖКИ

```
#INCLUDE "DEFINE.ASM" ; подключение таблицы имён и кодов SFR,
;применённых в программе

start:

        mov DPP,#08h ;настройка страницы ПЛИС

main:

;----------------включение экрвна с высвечиванием и без мерцания символа----------
--


        MOV DPTR,#0001H        ;выбор регистра данных DATA_IND

        mov A,#0Eh ;запись данных в аккумулятор

        movx @DPTR,A ;Записываем данные из аккумулятора  в быранный
регистр ПЛИСа


        MOV DPTR,#0006H    ;выбор регистра команд C_IND
```

```
        mov A,#01h          ;RS=0,RW=0,E=1:запись в аккумулятор данных
        movx @DPTR,A                ;Запись данных в из аккумулятора в  регистр
команд


        mov A,#00h
        movx @DPTR,A                ;RS=0,RW=0,E=0   стираем данные из регистра
команд
        lcall Pause                 ;ставим паузу в размере 2 секунды


;----------------------------высвечивание символа 1 в 4 позиции-----------------------
                        ;-------------------------------------------------------------------------------
        MOV DPTR,#0001H         ;выбор регистра данных DATA_IND
        mov A,#31h                  ;запись данных в аккумулятор
        movx @DPTR,A                ;Записываем данные из аккумулятора  в
быранный регистр ПЛИСа
        MOV DPTR,#0006H  ;выбор регистра команд C_IND
        mov A,#05h          ;запись в аккумулятор данных
        movx @DPTR,A                ;Запись данных в из аккумулятора в  регистр
команд(RS=1,RW=0,E=1 )


        mov A,#04h                  ;RS=1,RW=0,E=0   стираем данные из регистра
команд
        movx @DPTR,A                ;RS=0,RW=0,E=0
```

```
        lcall Pause                     ;ставим паузу в размере 2 секунды


;----------------------высвечивание символа 2 во 2-ой позиции-----------------------------
---

;все пречисленные ниже действия идентичный действию высвечиния 1 в позиции
1 , за исключением

;выбора позиции в видеопамяти и выбора элемента в знакогенераторе


                        ;-------------------------------------------------------------------------

        MOV DPTR,#0001H

        mov A,#32h

        movx @DPTR,A


        MOV DPTR,#0006H

        mov A,#05h

        movx @DPTR,A


        mov A,#04h

        movx @DPTR,A


        lcall Pause


;-------------------------------------------------------------------------
```

```
MP:
        ljmp MP
;-------------------------------------------------------------------------------


;функция паузы
Pause:
        MOV TCON,#00H ;настройка регистра управления таймером в
                      ;состояние по умолчанию, при этом снимаются
                      ;ранее установленные флаги переполнения и
                      ;останавливается счет событий.
        MOV TMOD,#00110001b         ;загрузка регистра TMOD - таймер 0
              ;настраивается как 16-разрядный счетчик с
        mov TH0,#00h;== только для первого цикла счетчика
        mov TL0,#00h;
        SETB TR0                    ;запуск счета событий
        mov R3,#64d;== 32 интервалов по 0.03125сек ~ 1сек
WAIT:
        JNB TF0,WAIT                ;ожидание переполнения счетчика
        CLR TF0                     ;сброс флага переполнения
        djnz R3,WAIT
        mov R3,#64d;== 32 интервалов по 0.03125сек ~ 1сек
        clr TR0
```

```
ret

.end
```

Приложение Д

(Обязательное)

Управление клавиатурой

Программа для клавиатуры

```
#INCLUDE "DEFINE.ASM"   ; подключение таблицы имён и кодов SFR,
;применённых в программе

.ORG 0000h

AJMP MAIN          ; переход к метке MAIN

MAIN:

mov R0,#00h                ;обнуление регистра R0

mov DPP,#08h               ;выбор страницы ргеистра ПЛИС

MOV DPTR,#0000H        ;ригистр клавиатуры

MOV A,#0eh        ; на последнем столбце нулевой потенциал

movX @DPTR,A


MOV DPTR,#0000H        ;ригистр клавиатуры

movX A,@DPTR

mov R1,A


;--------Анализ нажатых кнопок---------

;Если на регистре клавиатуры - код EEh, то

; нажета клавиша "1";

;если код DEh, то клавиша "4";
```

```
;если код BEh, то клавиша "7";
;если код 7Eh, то клавиша "*";

CJNE R1,#0eeh,EMPTY0 ;Сравнить непосредственные данные
   ;и A, при несовпадении перейти на адрес "EMPTY0"
MOV R0,#01h
EMPTY0:

CJNE R1,#0deh,EMPTY1 ;Сравнить непосредственные данные
   ;и A, при несовпадении перейти на адрес "EMPTY1"
MOV R0,#04h
EMPTY1:

CJNE R1,#0beh,EMPTY2 ;Сравнить непосредственные данные
   ;и A, при несовпадении перейти на адрес "EMPTY2"
MOV R0,#07h
EMPTY2:

CJNE R1,#07eh,EMPTY3 ;Сравнить непосредственные данные
   ;и A, при несовпадении перейти на адрес "EMPTY3"
MOV R0,#0FFh
```

```
EMPTY3:


; вывод результата на светодиоды
        MOV DPTR,#0007H
        mov A,R0
        movX @DPTR,A
        LJMP MAIN


.END
```

Приложение Е

(Обязательное)

Управление ЦАПами

Таблица 9 – программная модель регистра DACCON

Расположение бит	Мнемоника	Описание
DACCON7	MODE	Бит устанавливает режим работы обоих ЦАП. Если =1, то 8-ми битный. Если = 0, то 12-битный
DACCON6	RNG1	Бит выбора диапазона ЦАП1. Если =1, то диапазон 0..5В Если = 1, то диапазон 0..2,5В
DACCON5	RNG0	Бит выбора диапазона ЦАП0. Если =1, то диапазон 0..5В Если = 1, то диапазон 0..2,5В
DACCON4	CLR1	Бит очистки ЦАП1. Если =1, то выход ЦАП1 соответствует коду. Если = 0, то выход равен 0В
DACCON3	CLR0	Бит очистки ЦАП0. Если =1, то выход ЦАП0 соответствует коду. Если = 0, то выход равен 0В
DACCON2	SYNC	Бит синхронизации ЦАП0/1.
DACCON1	PD1	Бит включения ЦАП1. Если =1, то ЦАП1 включён Если = 0, то ЦАП1 выключен
DACCON0	PD0	Бит включения ЦАП0. Если =1, то ЦАП1 включён Если = 0, то ЦАП1 выключен

Таблица 10 -назначение битов регистра ENA.

Бит	Мнемоника	Описание
0	EN_LO	В полной конфигурации бит EN_LO нужен для управления младшими 8 разрядами (биты 0..7) 16-разрядного порта ввода-вывода. Если записать в EN_LO логический «0», то порт ввода-вывода переводится в Z-состояние и появляется возможность чтения данных из EXT_LO. При записи в данный бит логической «1» порт переключается на вывод и данные, записанные в регистр EXT_LO, попадают на выход порта ввода-вывода.
1	EN_HI	В полной конфигурации бит EN_HI нужен для управления старшими 8 разрядами (биты 8..15) 16-разрядного порта ввода-вывода. Если записать в EN_HI логический «0», то порт ввода-вывода переводится в Z-состояние и появляется возможность чтения данных из EXT_HI. При записи в данный бит логической «1» порт переключается на вывод и данные, записанные в регистр EXT_HI, попадают на выход порта ввода-вывода.
2..4	EPMSND0-EPMSND2	Выход звукового ЦАП. Задает уровень напряжения на динамике. Позволяет формировать звуковые сигналы различной тональности и громкости.
5	INT0	При записи логического «0» в этот бит на вход INT0 ADuC812 также попадает логический «0». Бит можно использовать для формирования внешнего прерывания для микроконтроллера.
6	KB	В полной конфигурации при записи логического «0» прерывание от клавиатуры запрещается. Если бит установлен в «1», то прерывание от клавиатуры разрешено. В упрощенной конфигурации бит KB всегда равен нулю, т.е. прерывание клавиатуры запрещено.

Демонстрационная программа управления ЦАПом

```
#INCLUDE "DEFINE.ASM"   ; подключение таблицы имён и кодов SFR,
;применённых в программе

START:
```

```
MOV         DAC0H,#0fh ; старшие биты
MOV         DAC0L,#0ffh ; младшие биты
mov DACCON,#00011101b ; ЦАП-0,12-бит, 2,5В.
sjmp START
.end
```

Приложение Ж

(Обязательное)

Управление портом расширения ENA

```
#INCLUDE "DEFINE.ASM"     ; подключение таблицы имён и кодов SFR,
;применённых в программе


START:
        mov DPP,#08h      ;выбор страницы регистра ПЛИС
        mov DPTR,#0004H       ;выбор регистра ENA
        mov A,#02h
        movX @DPTR,A  ;задаём параметры, что бы появилась возможность
считывать данные из EXT_LO и запись в EXT_HI
loop
        mov DPTR,#0002H   ;выбор регистра данных EXT_LO
        movX A,@DPTR          ;запись данных в аккумулятор из регистра
EXT_LO


        mov DPTR,#0003H               ;выбор регистра EXT_HI
        movX @DPTR,A          ;запись данных из аккумулятора в регистр
EXT_HI


        sjmp loop
        .end
```

Приложение 3

(Обязательное)

Принципиальная схема стенда SDK-1.1