МИКРОПРОЦЕССОРЫ В СИСТЕМАХ КОНТРОЛЯ

♦ ИЗДАТЕЛЬСТВО ТГТУ ♦

Министерство образования и науки Российской Федерации

Государственное образовательное учреждение
высшего профессионального образования
"Тамбовский государственный технический университет"

А.Е. Бояринов

МИКРОПРОЦЕССОРЫ В СИСТЕМАХ КОНТРОЛЯ

Методические указания

Тамбов
Издательство ТГТУ
2005

УДК 681.3.06(07)
ББК ←973.26-04я73-2
Б869

Р е ц е н з е н т

Кандидат технических наук, доцент
И.А. Дьяков

Бояринов, А.Е.
Б86 Микропроцессоры в системах контроля : методические
9 указания / А.Е. Бояринов. Тамбов : Изд-во Тамб. гос. техн. ун-та, 2005. 44 с.

Представлены методические указания для выполнения лабораторных работ по курсам "Микропроцессоры в системах контроля", "Микропроцессорная техника".

Предназначены студентам специальностей 200503, 220301 всех форм обучения.

УДК 681.3.06(07)
ББК ←973.26-04я73-2

Учебное издание

БОЯРИНОВ Алексей Евгеньевич

МИКРОПРОЦЕССОРЫ В СИСТЕМАХ КОНТРОЛЯ

Методические указания

Редактор Е.С. М о р д а с о в а

Инженер по компьютерному макетированию М.Н. Р ы ж к о в а

Подписано к печати 11.07.2005
Формат 60 × 84/16. Бумага офсетная. Печать офсетная.
Гарнитура Times New Roman. Объем: 2,56 усл. печ. л.; 2,44 уч.-изд. л.
Тираж 100 экз. С. 502^{M}

Издательско-полиграфический центр
Тамбовского государственного технического университета
392000, Тамбов, Советская, 106, к. 14

ИЗУЧЕНИЕ ЗАПОМИНАЮЩИХ УСТРОЙСТВ МИКРОПРОЦЕССОРНЫХ СИСТЕМ

Цель работы:

1 Изучить существующие виды полупроводниковых запоминающих устройств.
2 Изучить устройство постоянных и оперативных запоминающих устройств.
3 Познакомиться с работой запоминающих устройств на практике.
4 Получить навыки практической работы с программатором ПЗУ.

Методические указания

Память микропроцессорных (МП) систем часто разделяют на так называемую внутреннюю, которая выполняется на основе полупроводниковых БИС, и внешнюю – в виде магнитных дисков, лент или оптических носителей, обеспечивающих долговременное хранение большого объема информации. МП-системе для работы с внешней памятью требуются дополнительные аппаратные (например, дисковод и контроллер дисковода) и программные средства (программы-драйверы). Таким образом микропроцессор не может самостоятельно пользоваться внешней памятью. Внутренняя память системы может быть непосредственно доступна для микропроцессора. Мы будем рассматривать только полупроводниковые запоминающие устройства (ЗУ), используемые для организации внутренней памяти МП-систем.

Память (или ЗУ) состоит из огромного числа элементов памяти, каждый из которых может находиться в одном из двух состояний, кодируемых двоичной цифрой 1 или 0. ***Элемент памяти*** представляет собой область, где хранится бит информации. Элементы памяти ЗУ группируются в ***слова информации***, т.е. такие порции информации, которые могут одновременно пересылаться между ЗУ и МП и (или) обрабатываться последним. Область ЗУ, где хранится слово информации, называется ***ячейка памяти***. Структура ЗУ, состоящего из n ячеек, каждая из которых хранит слово из m битов, представлена на рис. 1.

Поиск нужного слова ЗУ можно производить либо по его адресу (адресные ЗУ), либо по его частичному содержанию (ассоциативные ЗУ). В адресных ЗУ обращение к ячейкам памяти производится по их физическим координатам, задаваемым двоичным кодом – адресом. Они бывают с произвольным обращением (выборкой), т.е. допускают любой порядок следования адресов, и с последовательным обращением, где выборка ячеек памяти возможна только в определенном порядке возрастания или убывания адресов. Архитектура МП ориентирована, в первую очередь, на использование адресных ЗУ с произвольной выборкой.

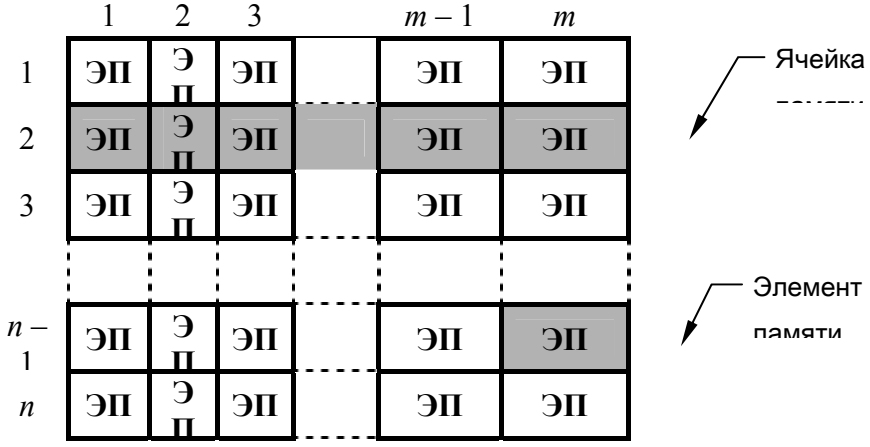


Рис. 1 Структура запоминающего устройства

Информационная емкость (или просто емкость). Емкость ЗУ выражается в количестве битов (б), байтов (Б) или слов, состоящих из определенного числа битов. Так как эта емкость может быть очень велика (до 10^{12} бит), то обычно используют более крупные единицы, образованные присоединением приставок к вышеперечисленным единицам: кило- (К), мега- (М) или гига- (Г). При этом надо учитывать, что в системах передачи и обработки информации приставки: К, М и Г, соответственно равны: 2^{10}

=
= 1024, 2^{20} = 1 048 676 и 2^{30} = 1 073 741 824.

В настоящее время наибольшее распространение получили полупроводниковые ЗУ, построенные на БИС (СБИС), каждая из которых содержит большое число элементов памяти. Эти элементы обычно объединяются в ячейки размером 1, 4 или 8 бит. Так БИС, содержащие 2К (2048) элементов памяти, можно изготавливать для хранения 2К 1-битовых слов, 512 4-битовых слов или 256 8-битовых слов (2К×1, 512×4 или 256×8).

Полупроводниковые ЗУ подразделяются на энергонезависимые – постоянные запоминающие устройства (ПЗУ) и энергозависимые – оперативные запоминающие устройства (ОЗУ). На рис. 2 представлены типы полупроводниковых ЗУ, применяемые в МП технике.

Постоянные запоминающие устройства

Благодаря энергонезависимости ПЗУ применяются для хранения инициализирующих и управляющих программ, различных таблиц констант и т.д. МП в процессе работы может только считывать (извлекать) информацию из ПЗУ, но не в состоянии изменить его содержимое.

Процесс занесения информации в ПЗУ называется программированием и производится, как правило, вне МП-системы, в которой предполагается их использовать. Для этого служат программаторы, выполняемые в виде автономных или периферийных устройств ЭВМ, в которых производится подготовка и хранение на внешних носителях записываемой в ПЗУ информации.

Исключением являются ПЗУ, выполненные по Flash-технологии, которая позволяет осуществлять общее стирание и запись информации в ячейки памяти непосредственно микропроцессором. Это свойство приближает его к ОЗУ, но в отличие от них Flash ПЗУ обладает энергонезависимостью.

На рис. 3 приведен пример внутренней структуры ПЗУ с организацией 256 ячеек по 8 бит (256×8). Блоки и сигналы, предназначенные для программирования ПЗУ, на схеме условно не показаны. Для адресации ячеек памяти служат 8 входных линий A0-A7. По ним можно задавать до $2^n = 2^8 = 256$ различных адресов в двоичном коде. Дешифратор адреса преобразует двоичный адрес в позиционный код для выбора одной из 256 строк матрицы элементов памяти (ЭП). С выходов выбранных ЭП на линии считывания вырабатываются сигналы 0 или 1. Таким образом, на выходы данных D0–D7, соединенные с вертикальными линиями считывания через усилители, поступит код, соответствующий хранящейся в адресуемой ячейке памяти информации.

Управляющие входы CS (выбор кристалла) и OE (разрешение по выходу) являются инверсными, т.е. активный уровень – логический нуль. Вход CS управляет общим выбором микросхемы, т.е. при подаче нуля (единицы) разрешается (запрещается) дешифрация адреса и выбор ячейки памяти. У выбранного ПЗУ с помощью входа OE производится активизация выходных буферов-усилителей. При отсутствии сигнала CS или OE выходы D0–D7 находятся в отключенном (высокоимпедансном) состоя-нии – в так называемом третьем состоянии.

Существует несколько разновидностей ПЗУ (рис. 2), которые раз-личаются принципом программирования, а также технологией изготов-ления.

Масочно-программируемые ПЗУ. Информация заносится в них в процессе изготовления, обычно на финишном его участке, и не может быть впоследствии изменена. В серийном производстве эти БИС относительно дешевы. Однако каждая "прошивка", т.е. заносимый в ПЗУ массив информации, требует соответствующей дорогостоящей технологической подготовки производства – индивидуальной маски (фотошаблона). Поэтому данный тип ПЗУ рентабельно применять в уже отлаженных изделиях, выпускаемых большими партиями.

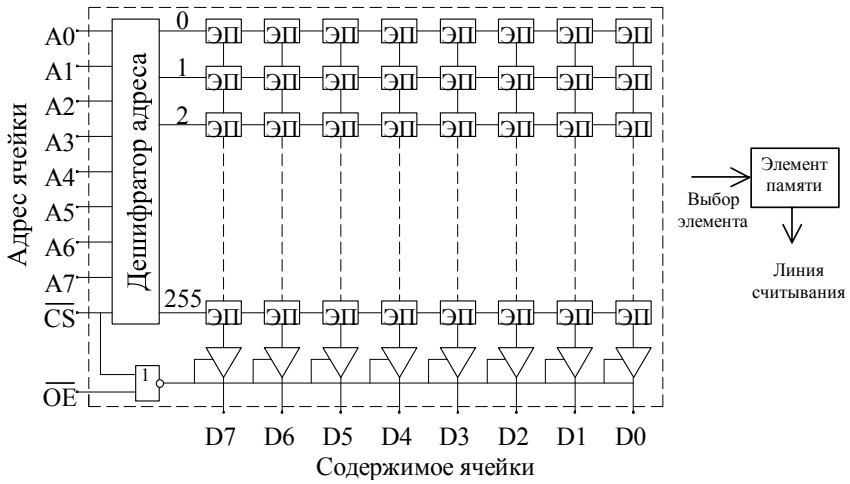


Рис. 3 Структура ПЗУ с произвольной выборкой

Программируемые ПЗУ (ППЗУ) с плавкими перемычками поступают к потребителю в первоначальном незапрограммированном состоянии, соответствующем 0 или 1 во всех элементах памяти (ЭП). В режиме программирования нужную информацию записывают в ППЗУ путем пережигания перемычек, играющих роль ЭП, электрическим током с помощью программатора. В дальнейшем изменение информации, занесенной в ППЗУ, возможно только путем пережигания перемычек, оставшихся после предыдущего программирования.

Репрограммируемые ПЗУ (РПЗУ) с ***ультрафиолетовым стиранием*** информации в настоящее время наиболее широко используются в МП-системах. В этих БИС каждый бит хранимой информации отображается состоянием соответствующего МОП-транзистора с плавающим затвором, представляющего собой конденсатор. Заряжая и разряжая его, производят запись и стирание информации. Накопленный заряд в таких конденсаторах может сохраняться очень долго (более 10 лет), за счет высокого качества изолирующего слоя.

Незапрограммированная микросхема РПЗУ с ультрафиолетовым стиранием имеет на выходах по всем адресам уровень логической единицы.

Для записи в требуемые разряды логического нуля на соответствующие выводы данных подается уровень 0, а на остальные – 1. Можно производить коррекцию ранее записанной информации, изменяя состояние
1 на 0 (но не наоборот).

Для стирания информации в течение 30…60 мин облучают кристалл БИС сквозь прозрачное окно в корпусе ультрафиолетовым излучением люминесцентной лампы, которое увеличивает ток утечки в изолирующем слое, приводя к рассасыванию хранимого на плавающих затворах заряда.

Число циклов перезаписи лежит обычно в пределах 10...100 (для различных типов), так как по мере перепрограммирования постепенно ухудшаются диэлектрические свойства изолирующего слоя.

РПЗУ с электрическим стиранием позволяет производить как запись, так и стирание информации с помощью электрических сигналов. Благодаря этому, появляется возможность изменять содержимое постоянной памяти непосредственно в МП-системе, если там предусмотрены устройства формирования сигналов стирания и программирования. Особенностью таких РПЗУ является блочное стирание, т.е. невозможность стирания информации в отдельных ячейках памяти.

Достоинством РПЗУ с электрическим стиранием является не только удобство и высокая скорость перезаписи информации, но и значительное допустимое число циклов перезаписи. Современные технологии гарантируется не менее 100000 циклов.

Flash память является разновидностью РПЗУ с электрическим стиранием. В настоящее время Flash ПЗУ широко используются для организации программной памяти микроконтроллеров. Допускается не менее 1000 циклов перезаписи.

На рис. 4 представлены примеры графического обозначения ПЗУ на принципиальных схемах. Функциональное назначение микросхемы указывается в середине. Аббревиатура ROM (Read Only Memory – память только для чтения) используется для ПЗУ. Слева расположены выводы входных сигналов адреса и управления, справа – выходы данных, подачи питания (Ucc, GND) и напряжения программирования (U_{PR}). Знак инверсии на вхо-

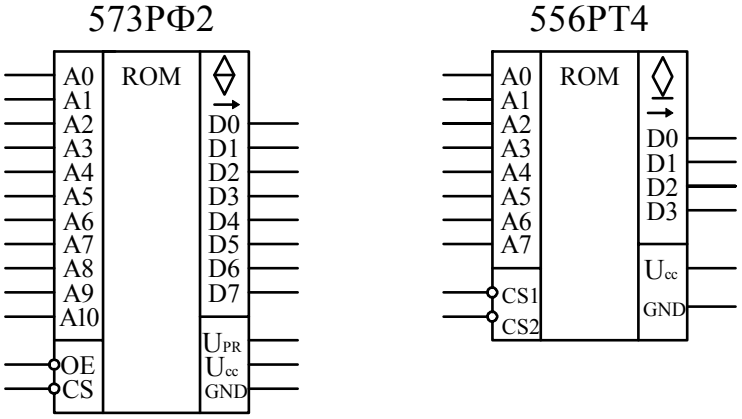


Рис. 4 Условное графическое обозначение ПЗУ

дах управления показывает, что включение режима производится логическим уровнем 0, а выключение – 1. Для обозначения типов выходов данных используются следующие символы: – выход с тремя состояниями;

– выход с открытым коллектором; – направление передачи информации.

Оперативные запоминающие устройства

ОЗУ служит для хранения временной и изменяемой информации, например отлаживаемых программ и промежуточных данных пользователя. Его главное преимущество перед ПЗУ – возможность записи и чтения информации непосредственно микропроцессором. При этом не требуется предварительное стирание содержимого ячеек памяти. Время записи в ОЗУ получается самым минимальным среди других типов ЗУ, а количество операций записи не ограничено.

ОЗУ в зависимости от структуры элементов памяти подразделяются на статические и динамические.

Элементы памяти в ***статических ОЗУ*** строятся на основе статических многотранзисторных триггерных цепей. Статические ОЗУ проигрывают в 4 – 8 раз по информационной емкости на единицу объема кристалла ***динамическим ОЗУ***, в которых запоминающий элемент выполняется однотранзисторным. Информация в таком элементе хранится в виде заряда на запоминающем конденсаторе. В режиме хранения информации необходимо периодически производить регенерацию заряда для компенсации естественных утечек. Регенерация производится чтением содержимого каждой ячейки ОЗУ с периодом не более 1...2 мс.

Регенерация может производиться микропроцессором программно, например, с помощью специальных прерываний. Но чаще она реализуется аппаратно с помощью специального контроллера регенерации.

Благодаря высокой плотности размещения, динамические структуры используют для создания микросхем ОЗУ наибольшего объема.

Хотя статические ОЗУ имеют меньший объем памяти – они не требуют постоянной регенерации. Это позволяет упростить аппаратные и программные средства МП-систем. Кроме того, статические ОЗУ потребляют гораздо меньше энергии, чем динамические и могут применяться в устройствах, работающих от автономных источников (аккумуляторов или батарей).

В настоящее время разрабатываются новые типы ОЗУ, обеспечивающие энергонезависимое хранение информации. Примером такого устройства являются статические ОЗУ фирмы "FRAMTON", элементы памяти которых выполнены на основе сегнетоэлектриков. Подобные ОЗУ, не уступая ПЗУ во времени хранения информации, превосходят их по быстродействию и количеству операций записи. Однако широкое их применение сдерживают высокая стоимость и недостаточная емкость.

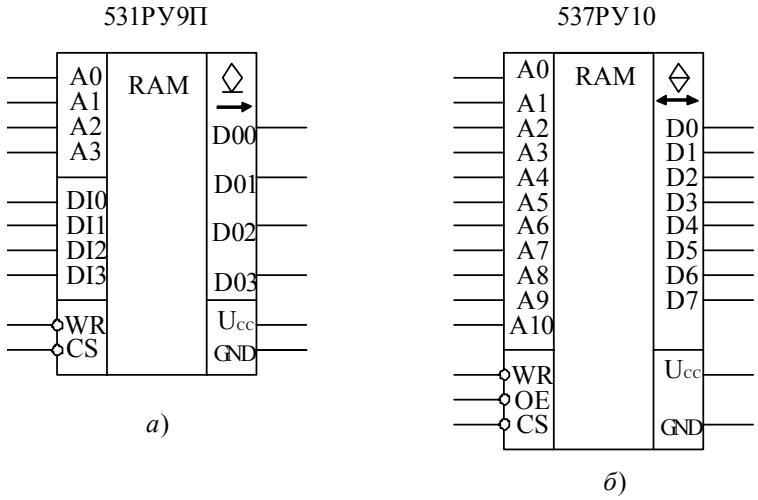


Рис. 5 Условное графическое обозначение ОЗУ

Внутренняя структура статического ОЗУ близка к схеме, приведенной на рис. 3. Отличия заключаются в том, что ЭП являются триггерами и имеется устройство управляющее их переключением по внешним сигналам разрешения записи и входных данных.

На рис. 5 представлены примеры графического обозначения ОЗУ на принципиальных схемах. Для них используется аббревиатура RAM (Random asess memory – память с произвольной выборкой).

Существует несколько вариантов организации выводов данных ОЗУ. На рис. 5, *а* показано условное графическое обозначение ОЗУ с раздельными входами (DI0-DI3) и выходами (DO0-DO3) данных, а на рис. 5, *б* – ОЗУ с совмещенными входами/выходами (D0-D7) данных. Знак "↔" свидетельствует о том, что выводы данных являются двунаправленными. Вход WR служит для управления записью информации. Так как он имеет знак инверсии, запись производится нулевым уровнем сигнала.

Программаторы

Программаторы служат для занесения информации в программируемые и репрограммируемые ПЗУ. Программаторы выполняют либо в виде автономных устройств, либо на базе компьютеров.

Автономные программаторы имеют ограниченные функциональные возможности и применяются в основном для копирования информации с ПЗУ-оригинала.

Более совершенными являются программаторы, построенные на базе компьютеров. Возможны два варианта их подключения: через стандартный интерфейс (COMPORT, LPT, USB); через системную шину компьютера. В первом случае программатор является внешним блоком компьютера, во втором – его внутренним модулем.

В лаборатории имеются все рассмотренные типы программаторов.

Первый программатор является автономным устройством для программирования ППЗУ с плавкими перемычками типа 155РЕ3 (32×8), 556РТ4 (256×4), 556РТ5 (512×8) и др. В состав программатора входит ОЗУ емкостью 1КБ и организацией 1024 ячеек по 8 разрядов.

Программатор позволяет в ручном режиме осуществлять предварительную подготовку (ввод и редактирование) данных в ОЗУ. При наличии ПЗУ-оригинала можно автоматически копировать из нее информацию в ОЗУ. Программирование ПЗУ происходит также автоматически по оригиналу информации, хранящейся в ОЗУ. При этом производится контроль правильности записи по каждому адресу.

Второй программатор выполнен на базе компьютера. Он представляет собой встраиваемый модуль и внешний блок с розетками для установки ПЗУ. Имеется специальное программное обеспечение для управления процессом ввода редактирования, хранения данных и программирования ПЗУ. Поэтому данный программатор обеспечивает большие функциональные возможности и широкую номенклатуру программируемых ПЗУ.

Третий программатор так же выполнен на базе компьютера, но подключается к нему через COMPORT. Он предназначен для Flash ПЗУ, которые не требуют повышенного напряжения при программировании.

Порядок выполнения работы

Работа с ОЗУ

В состав лабораторного стенда входит ОЗУ емкостью 1КБ и организацией 1024 ячеек по 8 разрядов. Для работы с ним тумблер режима устанавливается в соответствующее положение "ОЗУ".

Адресация ячеек ОЗУ может осуществляться последовательно и произвольно. При произвольном доступе можно по любому адресу извлечь интересующую нас информацию и также изменить ее. Для этого с помощью тумблеров "A0"..."A9" задается необходимый адрес в двоичном коде. При нажатии кнопки "∇" этот адрес отправляется на ОЗУ. Двоичный код текущего адреса можно проконтролировать с помощью светодиодов, расположенных под тумблерами "A0"..."A9". Данные, хранящиеся по заданному адресу, отображают светодиоды "D0"..."D7" так же в двоичном коде.

При последовательном доступе возможен выбор ячеек, адреса которых задаются только последовательно в порядке увеличения или уменьшения. Клавиша "▷" служит для увеличения, а клавиша "◁" – для уменьшения текущего (отображаемого светодиодами "A0"..."A8") адреса. Кнопка "Сброс" позволяет установить начальный адрес.

З а д а н и е 1 Просмотрите содержимое ячеек ОЗУ, имеющих следующие адреса: 00h-07h; 0Fh-15h; E8h-EDh[1]. Результат представьте в виде следующей таблицы:

Адрес ячейки		Содержимое ячейки	
h-код	b-код[2]	b-код	h-код

Для изменения информации новые данные задаются тумблерами "D0"..."D7", а их ввод производится при нажатии кнопки "Запись". Правильность записанных данных можно проконтролировать с помощью светодиодов "D0"..."D7".

З а д а н и е 2 Запишите в ОЗУ информацию в соответствии со следующей таблицей:

Адрес, h-код	Данные, h-код	Адрес, h-код	Данные, h-код
00	01	08	FE
01	02	09	FD
02	04	0A	FB
03	08	0B	F7
04	10	0C	EF
05	20	0D	DF
06	40	0E	BF
07	80	0F	7F

Проверьте правильность ввода информации путем повторного просмотра данных.

Если отключить лабораторный стенд от сети и включить вновь, то записанная информация разрушится, т.е. ОЗУ не может хранить информацию при отсутствии питания.

З а д а н и е 3 Выключите тумблер питания стенда на несколько секунд. Убедитесь, что вся ранее введенная вами информация отсутствует.

Работа с ПЗУ

[1] Обозначение h соответствует шестнадцатеричной системе.

[2] Обозначение b соответствует двоичной системе.

Для работы с ПЗУ тумблер режима устанавливается в положение "ПЗУ". На лицевой панели лабораторного стенда имеется розетка для подключения микросхем РПЗУ с УФ-стиранием типа К573РФ2.

Вставьте в розетку стенда микросхему ПЗУ. При этом соблюдайте правильность соединения по "меткам", обозначающим первый вывод ПЗУ и первый контакт розетки. Процесс просмотра содержимого ПЗУ аналогичен работе с ОЗУ.

З а д а н и е 4 Проверьте содержимое ячеек ПЗУ в области адресов 00h-18h. Используя кодировку ASCII представьте хранящуюся информацию в виде символов. Результаты оформите в виде следующей таблицы:

Адрес ячейки h-код	Данные b-код	Данные h-код	Символы ASCII

З а д а н и е 5 Поместите ПЗУ в программатор, считайте его содержимое и сравните данные с полученными ранее.

З а д а н и е 6 Запрограммируйте ПЗУ в соответствии с данными, предоставленными преподавателем.

Контрольные вопросы

1 Что собой представляет внутренняя память МП-систем?
2 Что такое элемент памяти?
3 Какой объем информации называется словом?
4 Что такое ячейка памяти?
5 Какие существуют способы поиска данных в ЗУ?
6 Какие существуют способы адресации ЗУ?
7 Как различают ЗУ по времени хранения данных?
8 Для чего применяют ПЗУ в МП-системах?
9 Для чего применяют ОЗУ в МП-системах?
10 Какую внутреннюю структуру имеют ЗУ?
11 Как различают ПЗУ по способу программирования?
12 Как различают РПЗУ по способу стирания?
13 Что такое программатор и какие бывают варианты его исполнения?
14 Какие существуют типы ОЗУ?
15 Как осуществляется регенерация динамических ОЗУ?
16 Какие существуют варианты организации ввода-вывода данных ОЗУ?

Лабораторная работа 2

ИЗУЧЕНИЕ ТЕХНОЛОГИИ РАЗРАБОТКИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ДЛЯ МИКРОКОНТРОЛЛЕРОВ

Методические указания

Для создания программного обеспечения микропроцессорных систем широко используются средства вычислительной техники, в том числе персональные компьютеры, и специальные программы, позволяющие разработчику выполнить весь цикл проектирования, включая отладку целевой программы. Рассмотрим технологию разработки программного обеспечения микроконтроллеров на примере использования продукта американской фирмы Keil Software Inc называемого µVision2.

µVision2 фирмы Keil Software Inc. – интегрированная среда разработки программного обеспечения для однокристальных микроконтроллеров семейства MCS51. Keil µVision2 имеет стандартный интер-

фейс Windows и включает в себя все, что нужно для создания, редактирования, компиляции, трансляции, компоновки, загрузки и отладки программ:

- организатор проекта;
- полнофункциональный редактор исходных текстов с выделением синтаксических элементов цветом;
- компилятор языка Си;
- макроассемблер;
- библиотеку стандартных функций;
- компоновщик;
- отладчик;
- операционную систему реального времени.

Первый этап разработки программного обеспечения – создание и настройка проекта под конкретный тип микроконтроллера.

На следующем этапе осуществляется запись исходного текста программы на каком-либо языке программирования. После этого производится его трансляция в коды команд микроконтроллера с использованием компилятора C51 или ассемблера A51. Компиляторы и ассемблеры – прикладные программы, которые преобразуют исходный текст программы в объектный модуль, представляющий собой перемещаемый программный код с относительной адресацией. Объектные и библиотечные модули с помощью программы компоновщика L51 объединяются в исполняемый программный код, размещаемый по абсолютным адресам.

После компоновки объектных модулей наступает этап отладки программы, устранения ошибок, оптимизации и тестирования программы. В составе среды Keil μVision2 имеются мощные средства отладки, позволяющие симулировать работу микроконтроллера в режиме выполнения программы, наблюдать за содержимым регистров, памяти и контролировать работу всех устройств. Это позволяет исправить практически все ошибки и получить рабочую версию программы до создания самого устройства. Для загрузки готовой программы в память микроконтроллера обычно используют выходной файл в формате HEX, получаемый с помощью программы-конвертора OHx51.

На рис. 6 схематически представлен процесс создания программного обеспечения для микроконтроллеров. В составе Keil μVision2 имеются следующие основные компоненты.

Макроассемблер A51

Ассемблер A51 транслирует символическую мнемонику в перемещаемый объектный код, имеющий высокое быстродействие и малый размер. Макросредства ускоряют разработку и экономят время, поскольку общие последовательности могут быть разработаны только один раз. Ассемблер поддерживает символический доступ ко всем элементам микроконтроллера и перестраивает конфигурацию для каждой разновидности MCS51.

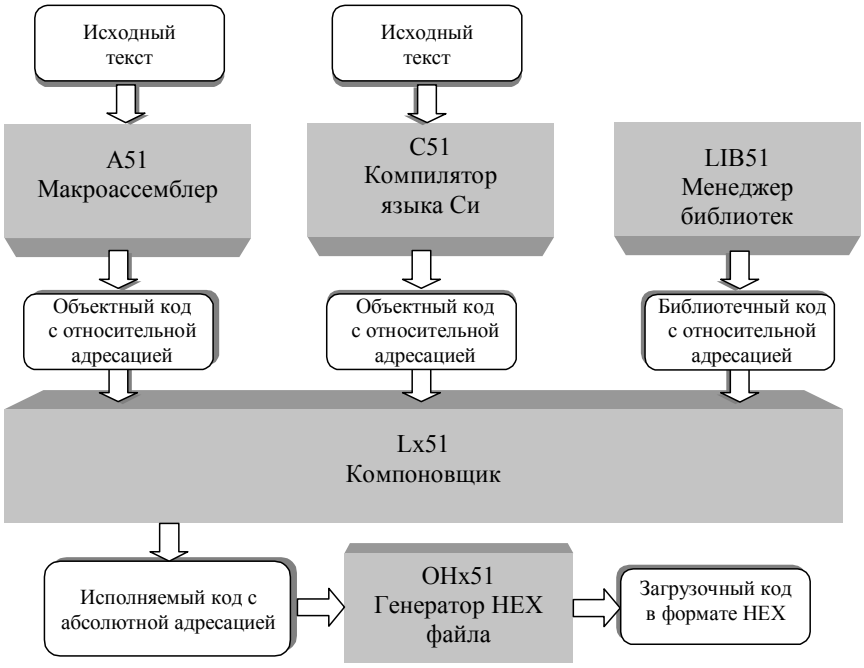


Рис. 6 Схема процесса создания программ для микроконтроллеров

При отладке или при включенной опции "Debug information" объектный файл будет содержать полную символическую информацию для отладчика.

Оптимизирующий кросс-компилятор C51

Язык Си – универсальный язык программирования, который обеспечивает эффективность кода, элементы структурного программирования и имеет богатый набор операторов. Универсальность, отсутствие ограничений реализации делают язык Си удобным и эффективным средством программирования для широкого разнообразия задач. Множество прикладных программ может быть написано легче и эффективнее на языке Си, чем на других более специализированных языках.

C51 – полная реализация стандарта ANSI (Американского национального института стандартов), насколько это возможно для архитектуры MCS51 и генерирует код для всего семейства этих микроконтроллеров. Компилятор сочетает гибкость программирования на языке высокого уровня с эффективностью кода и быстродействием ассемблера.

Использование языка высокого уровня Си имеет следующие преимущества над программированием на ассемблере:

- глубокого знания системы команд процессора не требуется, элементарное знание архитектуры микроконтроллера желательно, но не необходимо;
- распределение регистров и способы адресации управляются полностью компилятором;
- лучшая читаемость программы, используются ключевые слова и функции, которые более свойственны человеческой мысли;
- время разработки программ и их отладки значительно короче в сравнении с программированием на ассемблере;
- библиотечные файлы содержат много стандартных подпрограмм, которые могут быть включены в прикладную программу;
- существующие программы могут многократно использоваться в новых программах, используя модульные методы программирования.

Компоновщик L51

Компоновщик объединяет один или несколько объектных модулей в одну исполняемую программу. Компоновщик размещает внешние и общие ссылки, назначает абсолютные адреса перемещаемым сегментам программ. Он может обрабатывать объектные модули, созданные компилятором C51 и ассемблером A51. Компоновщик автоматически выбирает соответствующие библиотеки поддержки и связыва-

ет только требуемые модули из библиотек. Установки по умолчанию для L51 выбраны так, чтобы они подходили для большинства прикладных программ, но можно определить и заказные установки.

Порядок работы в среде Keil µVision2

Запуск Keil µVision2 и создание файла проекта

Keil µVision2 запускается из стартового меню Windows подобно остальным приложениям, либо с помощь ярлыка вынесенного на "Рабочий стол" компьютера.

На рис. 7 представлен общий вид на экране компьютера среды Keil µVision2. В верхней части экрана находится панель, включающая следующие меню:

File – для работы с файлами;
Edit – для редактирования файлов;
View – для управления режимом отображения;
Project – для настройки параметров проекта и управления его обработкой;
Debug – для отладки программ;
Peripherals – для контроля периферийных устройств при отладке программ;
Tools – для подключения и использования дополнительных программ;

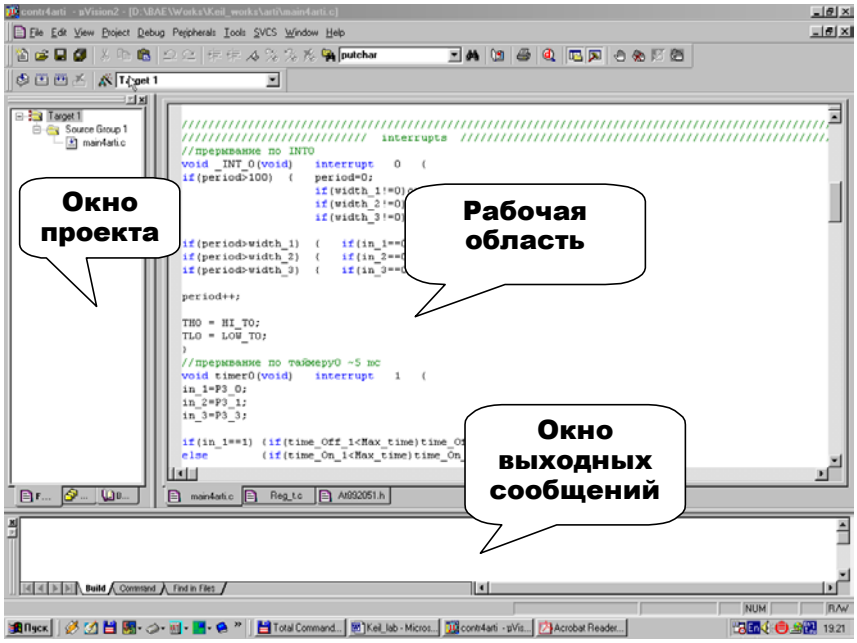


Рис. 7 Вид среды Keil µVision2

SVCS – для контроля версии программы;
Window – для выбора способа расположения окон;
Help – для получения справочной информации.

Разработка программного обеспечения начинается с установки параметров интегрированной среды Keil µVision2. Совокупность настроек среды, исходных файлов программ, стандартных библиотек и библиотечных модулей для решения конкретной задачи называется Project (проект). Созданный проект может быть сохранен и в дальнейшем использован для дальнейшей работы. Для разработки программы сразу для нескольких типов микроконтроллеров в проекте можно создавать несколько целевых задач, называемых Target. Каждая из них имеет общие для всех исходные файлы, однако могут различаться настройками среды.

Создание нового проекта

Для создания нового проекта выбираем команду New Project… из меню Project. При этом на экран выводится окно выбора устройства (рис. 3), в котором следует выбрать соответствующий тип микроконтроллера.

В базе поддерживаемые устройства сгруппированы в папки по фирмам-производителям.

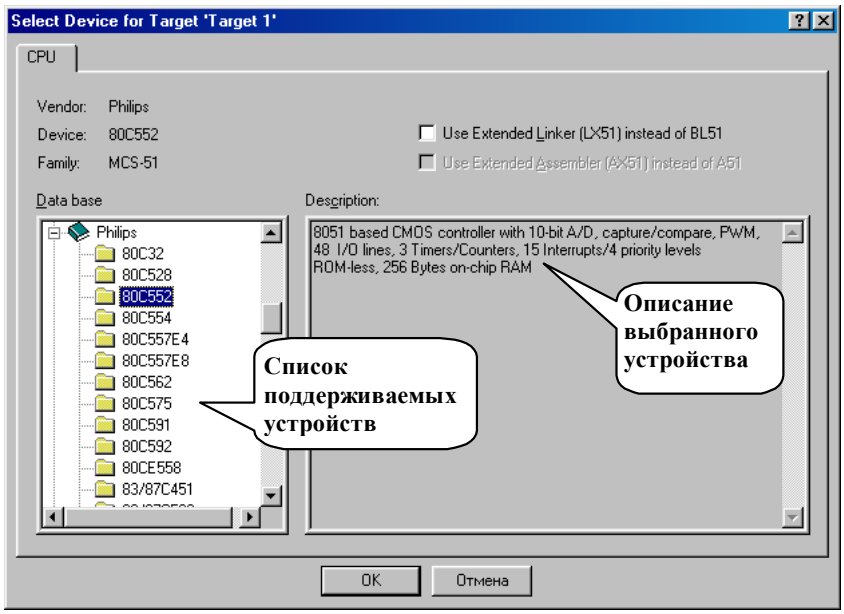


Рис. 8 Окно выбора устройства

Выбрав нужный микроконтроллер (целевая задача), которой по умолчанию присваивается имя Target 1. В составе целевой задачи автоматически создается так называемая Group 1 (группа) – папка с именем Source Group 1. В эту папку включают исходные файлы проекта. На рис. 4 показано как будет выглядеть структура на закладке Files проекта в Project Window (окно проекта).

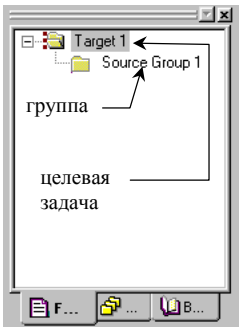


Рис. 4 Окно проекта

В группу можно включать исходные файлы в виде текста на языке Ассемблер и (или) Си, а также библиотечные и объектные файлы. При необходимости в проекте создают дополнительные группы, чтобы разделить файлы по видам или установить для них индивидуальные настройки.

Создание и сохранение нового файла

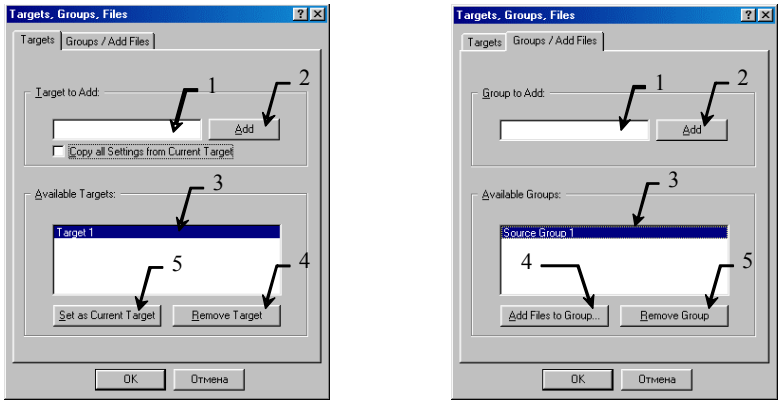
Для создания нового файла в меню File нужно выбрать пункт New или нажать кнопку  панели File Toolbar. При этом откроется окно нового файла, которому по умолчанию присваивается имя Text1. После ввода текстовой информации файл необходимо сохранить через меню File пункт Save As…

Организация структуры проекта

Организация структуры проекта производится через меню Project командой Targets, Groups, Files или через окно проекта Project Window по нажатию правой кнопки мыши на нужном элементе. При этом открывается список команд, одна из которых Targets, Groups, Files. Выбор этой команды открывает окно с двумя закладками: Targets и Groups/Files. Первая закладка (рис. 5, *а*) служит для добавления/удаления и определения текущей целевой задачи. С помощью второй закладки (рис. 5, *б*) для всех целевых задач добавляют/удаляют группы, а также в выбранную группу добавляют файлы.

Добавление новой целевой задачи

Для добавления новой целевой задачи нужно ввести ее имя в строку (рис. 5, *а*, указатель 1), а затем нажать кнопку Add (рис. 5, *а*, указатель 2). При этом имя отобразится в списке Available Targets (рис. 5, *а*, указатель 3).



а) *б*)

Рис. 5 Окно для настройки структуры проекта

Задание активной целевой задачи

В проекте может быть активна только одна Target, которая выбирается из списка и отмечается нажатием кнопки Set as Current Target (рис. 5, *а*, указатель 5). Другой способ активизации целевой задачи заключается в выборе ее из списка Select Target панели Build Toolbar (рис. 8, указатель 1).

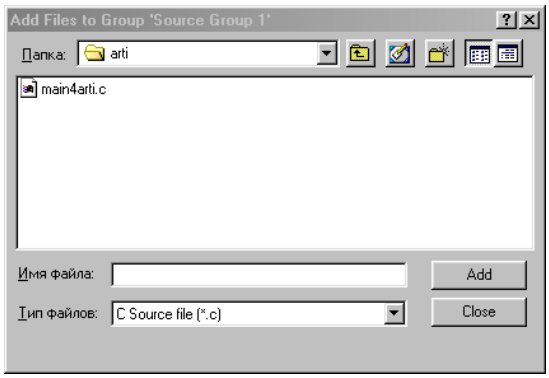


Рис. 6

Удаление целевой задачи

Для удаления ненужных целевых задач нужно выделить левой кнопкой мыши соответствующее имя в списке Available Targets (рис. 5, *а*, указатель 3), а затем нажать кнопку Remove Target (рис. 5, *а*, указатель 4).

Добавление новых групп

Для добавления новой группы нужно ввести ее имя в строку (рис. 5, *б*, указатель 1), а затем нажать кнопку Add (рис. 5, *б*, указатель 2). При этом имя новой группы отобразится в списке Available Group (рис. 5, *б*, указатель 3).

Удаление групп

Для удаления ненужных групп нужно выделить левой кнопкой мыши имя соответствующей группы в списке Availadle Group (рис. 5, *б*, указатель 1), а затем нажать кнопку Remove Group (рис. 5, *б*, указатель 5).

Включение файлов в группу

Для включения файлов в группу нужно выделить левой кнопкой мыши имя группы (рис. 5, *б*, указатель 1), а затем нажать кнопку Add Files to Group… (рис. 5, *б*, указатель 4). При этом откроется окно для

добавления файлов (рис. 6), в котором нужно выбрать тип файла, указать путь к нему и нажать кнопку Add. Добавление не закрывает окно, что позволяет оперативно подключить несколько файлов. Закрытие производится кнопкой Close. Результат выполнения ваших действий можно проконтролировать в окне Project Window (рис. 4).

Удаление файлов из группы

Для удаления файла из группы (рис. 7) нажать на нем правой кнопкой мыши. При этом откроется список команд, в котором следует выбрать команду Remove File ‘имя файла’.

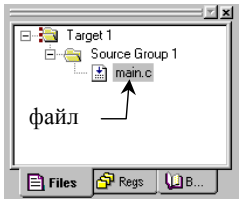


Рис. 7

Настройка параметров проекта

Для целевой задачи устанавливают:

- тип микроконтроллера;
- модель памяти данных;
- размер программной памяти;
- формат выходного файла;
- форматы файлов отчетов;
- параметры компиляции;
- параметры компоновки и размещения кода;
- параметры отладчика.

ДЛЯ ГРУПП И ФАЙЛОВ УСТАНАВЛИВАЮТ:

- свойства файлов;
- параметры компиляции.

Настройка параметров групп и файлов может быть произведена индивидуально либо может быть общей и соответствовать параметрам целевой задачи.

Выбор типа микроконтроллера

Предварительно необходимо выделить в окне Project Window целевую задачу (рис. 4). Выбор типа микроконтроллера для активной целевой задачи осуществляется командой Select Device for Target ‘имя’ из меню Project либо из окна, вызываемого щелчком правой кнопки мыши на имени целевой задачи в Project Window. При этом на экран выводится окно выбора устройства (рис. 3), в котором следует выбрать соответствующий тип микроконтроллера.

Настройка параметров целевой задачи

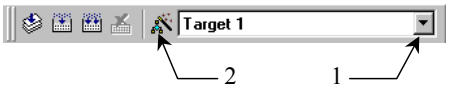


Рис. 8 Вид панели Build Toolbar

Для открытия окна настройки параметров целевой задачи можно воспользоваться кнопкой Options for Target панели Build Toolbar (рис. 8, указатель 2).

Окно настройки параметров (рис. 9) имеет 8 закладок, на каждой из которых можно устанавливать соответствующие значения для выбранной целевой задачи.

Закладка Target (рис. 9) предназначена для следующего.

1 Установка значения Xtal – тактовой частоты микроконтроллера в Мгц. Этот параметр используется при отладке программы в режиме симуляции в среде Keil µVision2.

2 Выбор Memory Model – модели памяти для хранения данных. Возможные варианты:

- Small (переменные размещаются во внутренней памяти);
- Compact (переменные размещаются в пределах одной страницы внешней памяти);
- Large (переменные размещаются во внешней памяти).

3 Выбор Rom Code Size – размера программной памяти. Возможные варианты:

- Small (объем программы не превышает 2 КБ);
- Compact (объем программы не превышает 64 КБ, а подпрограмм – 2 КБ);
- Large (объем программы и подпрограмм не превышает 64 КБ).

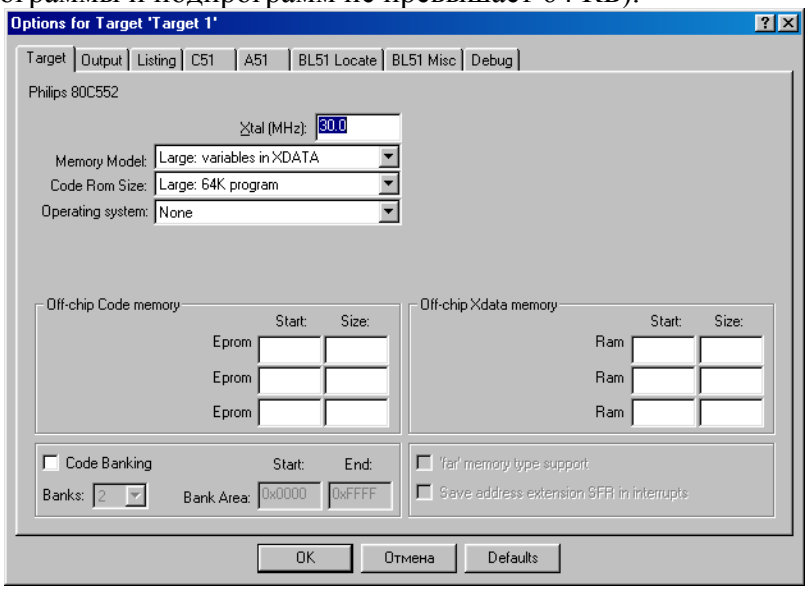


Рис. 9 Вид окна Options for Target (закладка Target)

4 Использование операционной системы (ОС) и выбор ее версии:

- None – без ОС;
- RTX-51 Full – выбор полной версии многозадачной ОС реального времени для MCS-51;
- RTX-51Tiny – выбор минимизированной версии ОС для микропроцессорной системы без внешней памяти.

5 Определение Off-chip Code memory – областей адресов внешнего ПЗУ программ. Указывается начальный адрес и размер для трех областей памяти.

6 Определение Off-chip Xdata memory – областей адресов внешнего ОЗУ данных. Указывается начальный адрес и размер для трех областей памяти.

Закладка Output (рис. 10) предназначена для следующего.

1 Выбор папки для размещения выходных файлов и их имени (указатель 1). По умолчанию имена файлов совпадают с названием проекта и размещаются в той же папке, что и сам проект.

2 Выбор типа создаваемых файлов. Нажатие кнопки Create Executable позволяет получать исполняемые файлы. При этом установка флажка:

- Debug Information добавляет в выходной объектный код информацию необходимую для отладки программы;

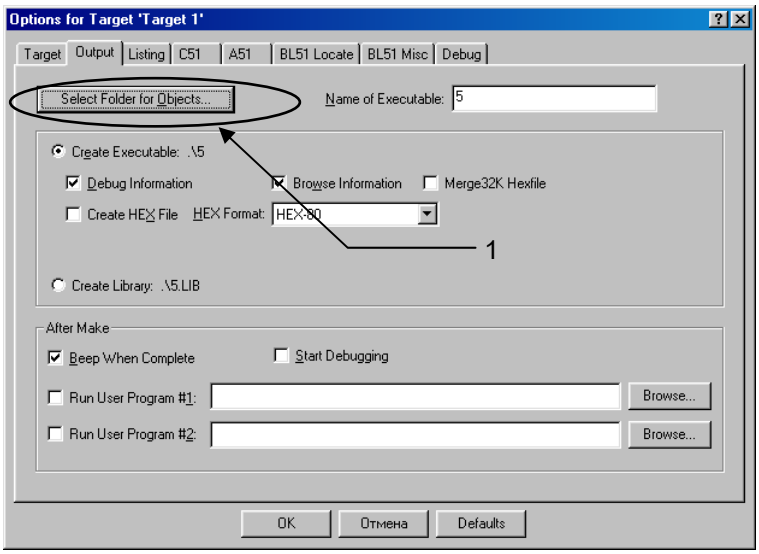


Рис. 10 Вид окна Options for Target (закладка Output)

- Browse Information делает возможным информационный поиск в программе;
- Create Hex File позволяет получать файл в формате Intel Hex для загрузки в микроконтроллер.

Нажатие кнопки Create Library позволяет получать библиотечные файлы.

3 Назначение выполнения действий после обработки проекта:

- Подача звукового сигнала;
- Запуск отладчика;
- Запуск программ по желанию пользователя.

Закладка Listing (рис. 11) предназначена для настройки создания отчетов о выполнении компиляции и компоновки проекта.

Закладка C51 (рис. 12) предназначена для настройки компилятора языка Си. Позволяет осуществлять следующее.

1 Ввод символов препроцессора языка Си.

2 Настройку оптимизации получаемого программного кода:

- Level – устанавливает уровень (степень) оптимизации;
- Emphasis – выбирает акцент между скоростью работы и объемом программы.

3 Оптимизацию использования регистров микроконтроллера.

4 Определение типа и степени строгости предупреждений компилятора.

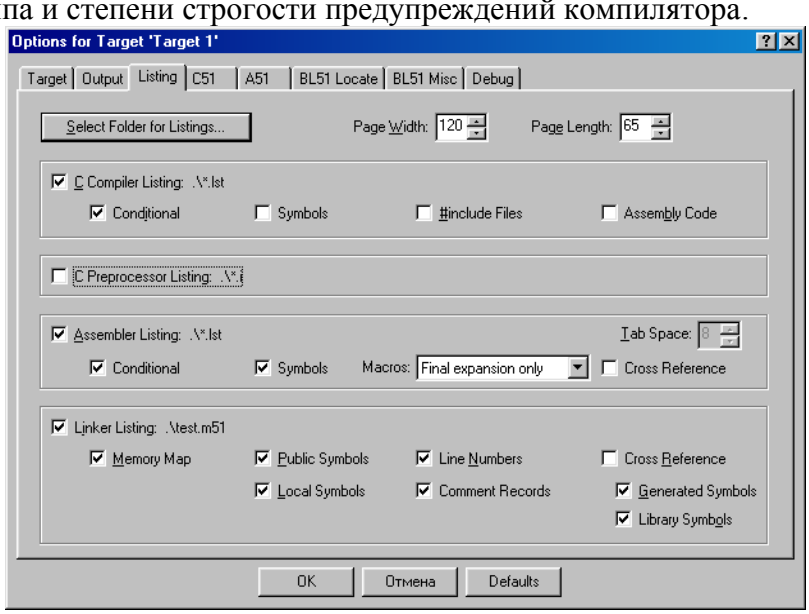


Рис. 11 Вид окна Options for Target (закладка Listing)

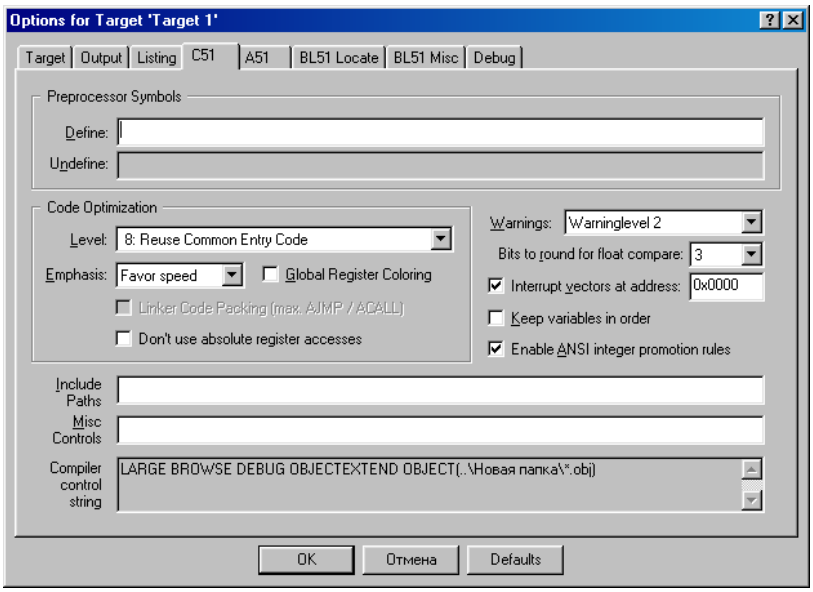


Рис. 12 Вид окна Options for Target (закладка C51)

5 Устанавливает число округляемых бит при сравнении чисел типа float (с плавающей точкой).

6 Устанавливает начальный адрес вектора прерывания. Для этого нужно ввести адрес и поставить флажок, разрешающий смещение вектора.

7 Задание порядка размещения переменных в соответствии с последовательностью их объявления.

8 Разрешение приведения к типу integer переменных меньшего размера, для обеспечения совместимости с ANSI.

9 Ввод дополнительного пути.

10 Ввод специальных директив для компилятора C51.

Все установленные параметры автоматически вносятся в Compiler control string.

Закладка A51 (рис. 13) предназначена для настройки компилятора языка ассемблер.

Позволяет осуществлять следующее.

1 Вводить символы и их значения для управления условным ассемблированием программ.

2 Выбирать тип макропроцессора.

- Standard – разрешает использование стандартных макрокоманд;
- MPL – разрешает использование расширенного языка макрокоманд.

3 Разрешать/запрещать предопределение зарезервированных имен регистров специальных функций.

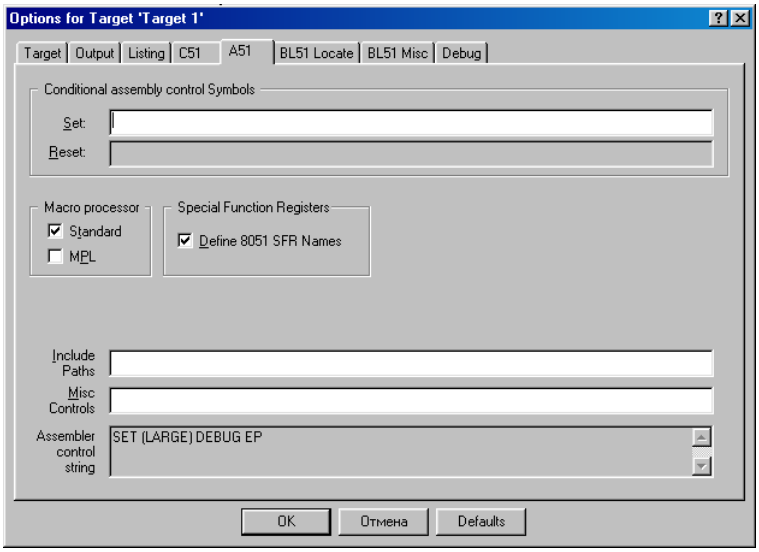


Рис. 13 Вид окна Options for Target (закладка A51)

4 Указывать пути к INCLUDE файлам.

5 Вводить прочие директивы для управления ассемблированием.

Все установленные параметры автоматически вносятся в Assembler control string.

Закладка BL51 Locate (рис. 14) предназначена для настройки компоновщика.

Позволяет осуществлять следующее.

1 Задавать области программной памяти и внешней памяти данных либо использовать установки с закладки Options for Target.

2 Определить области размещения и последовательность сегментов следующих классов памяти:

- Code – программной памяти;
- Xdata – внешней памяти данных;
- Pdata – внешней страничной памяти;
- Bit – побитно адресуемой памяти;
- Data – внутренней памяти данных;
- Idata – внутренней косвенно адресуемой памяти данных;
- Stack – стека.

Все установленные параметры автоматически вносятся в Linker control string.

Закладка BL51 Misc (рис. 15) предназначена для установки дополнительных настроек компоновщика.

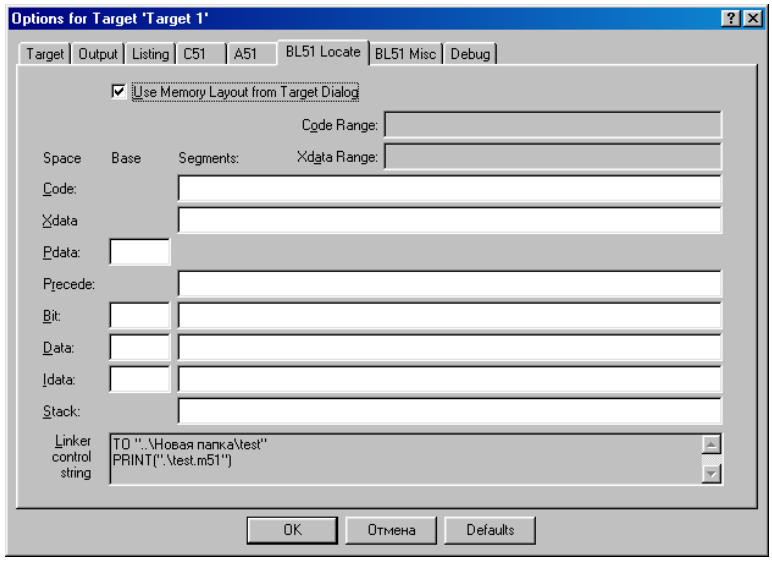


Рис. 14 Вид окна Options for Target (закладка BL51 Locate)

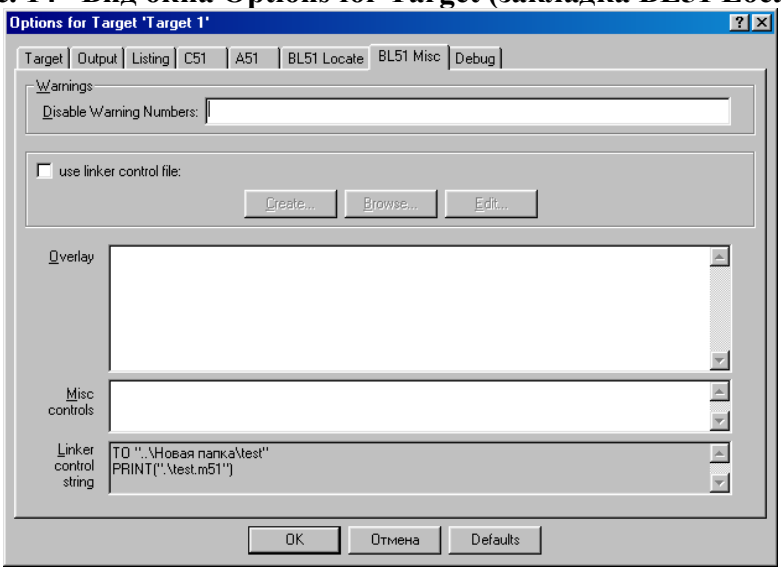


Рис. 15 Вид окна Options for Target (закладка BL51 Misc)

Позволяет осуществлять следующее.

1 Запрещать выдавать определенные предупреждения.

2 Использовать файл для управления компоновкой.

3 Разрешать перекрытие сегментов данных.

4 Вводить прочие управляющие директивы компоновщика.

Все установленные параметры автоматически вносятся в Linker control string.

Закладка Debug (рис. 16) предназначена для настройки отладчика.

Позволяет осуществлять следующее.

1 Выбирать способ отладки:

- Use Simulator – отладка программы производится на компьютере, который имитирует работу микроконтроллера;
- Use Keil Monitor-51 Driver – отладка программы производится под управлением компьютера, подключенному к микроконтроллеру через последовательный интерфейс. Так же может быть выбран и другой тип драйвера для внутрисхемной отладки программы.

2 Автоматически загружать приложение при запуске отладчика, если выбрано.

3 Устанавливать начало отладки программы с метки main, если выбрано Go till main ().

4 Определять в сроке Initialization File файл отладочных функций для имитации внешних сигналов.

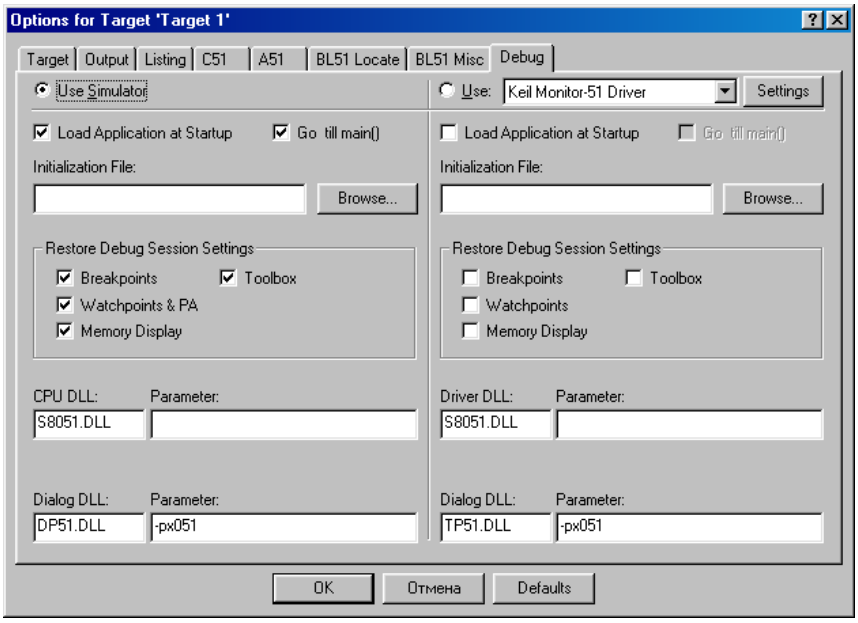


Рис. 16 Вид окна Options for Target (закладка Debug)

5 Восстанавливать установки предыдущего сеанса отладки, отмечая нужные пункты Restore Debug Session Setting:

- Breakpoints – восстанавливать точки останова программы;
- Watchpoints & PA – восстанавливать точки наблюдения и анализатор выполнения программы;
- Memory Display – восстанавливать дисплей памяти;
- Toolbox – восстанавливать кнопки управления.

Настройка шрифтов проекта

Для установки стиля, размера и цвета штифтов, используемых в данном проекте, следует выбрать в меню View пункт Options. При этом откроется окно Options, в котором на закладке Colors & Fonts (рис. 17) необходимо указать параметры шрифтов, используемых в различных окнах данного проекта. Сначала выбирают окно (рис. 17, указатель 1), затем вид текста в этом окне (рис. 17, указатель 2) и устанавливают тип штифта Font его размер Size и цвет Color (рис. 17, указатель 3).

Для сохранения установленных шрифтов следует нажать кнопку OK.

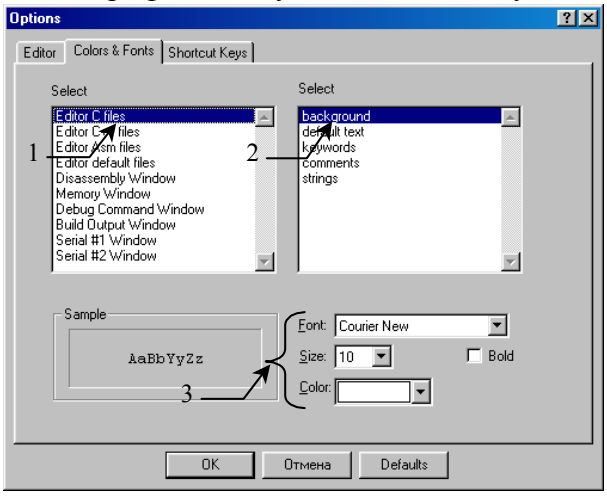


Рис. 17 Вид закладки Colors & Fonts окна View – Options

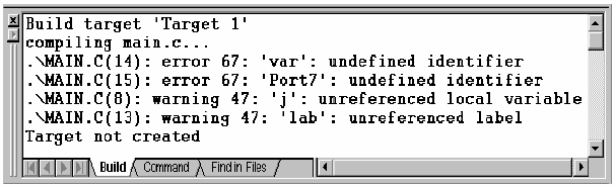
Обработка проекта

Процесс обработки включает компиляцию (трансляцию) исходных файлов проекта, их компоновку и, если разрешено в установках, создание Hex файла, предназначенного для загрузки в программную память микропроцессорной системы. Для выполнения этого служат команды из меню Project:

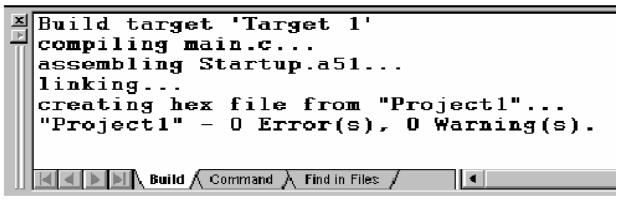
- Translate – компиляция исходного файла проекта;
- Build Target – компиляция только измененных файлов проекта и их компоновка;
- Rebuild All Target Files – компиляция всех исходных файлов проекта и их компоновка.

Для удобства работы можно пользоваться кнопками панели Build Toolbar (рис. 9) для ввода соответствующих команд: – Translate; – Build Target; – Rebuild All Target Files.

Результаты обработки проекта отображаются в окне выходных сообщений на закладке Build (рис. 18). Здесь выводятся сообщения об ошибках компиляции и компоновки, а также предупреждения. Двойной щелчок мыши на сообщение об ошибке компиляции позволяет получить подсказку о ее местонахождении в исходном тексте. Возникающие ошибки не позволяют создать загружаемый файл, и выводится сообщение Target not created (рис. 18, *а*). Если ошибки отсутствуют и в установках разрешено создание Hex файла, то выводится сообщение creating hex file from (рис. 18, *б*).



а)



б)

Рис. 18 Вид закладки Build окна выходных сообщений

Отладка программы

Прежде чем готовая программа будет загружена в микроконтроллер, вы можете ее опробовать и отладить в среде Keil µVision2. Один из способов отладки – Simulator удобен там, что для него не требуется целевая микропроцессорная система, так как отладка программы производится на компьютере, который имитирует работу микроконтроллера.

Рис. 19 Вид панели Debug Toolbar

Для отладки предварительно откомпилированной программы выбирают команду Start/Stop Debug Session из меню Debug или используют кнопку . При этом активизируется панель Debug Toolbar (рис. 19), а в окне проекта открывается закладка Regs (рис. 20) для контроля содержимого внутренних регистров микроконтроллера.

Рассмотрим назначение кнопок панели Debug Toolbar.

– Сброс микроконтроллера

– Запуск программы на выполнение

– Остановка выполнения программы

– Выполнение одного шага программы для входа в функцию

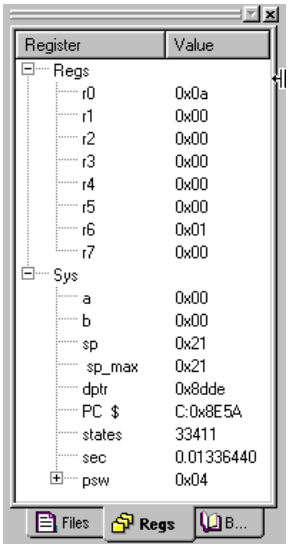


Рис. 20 Закладка Regs

– Выполнение одного шага программы для перехода через функцию

– Выполнение одного шага программы для выхода из функции

– Выполнение программы до линии, отмеченной курсором

– Показать следующую выполняемую команду

– Вкл./выключение записи выполнения программы

– Включение просмотра ретроспективы выполнения программы

– Вкл./выключение окна дизассемблера

– Вкл./выключение окна наблюдения за переменными и стеком

– Вкл./выключение окна оценки выполненных команд из их общего количества

– Вкл./выключение окна последовательного порта

– Вкл./выключение окна просмотра содержимого памяти

– Вкл./выключение анализатора времени выполнения функций программы

– Вкл./выключение окна Toolbox.

Отладка программы начинается с запуска программы на выполнение. Последовательность выполнения команд можно контролировать в окне исходного текста (рис. 21, *а*) или в окне дизассемблера (рис. 21, *б*), в котором кроме исходного текста программы отображается содержимое программной памяти и команды на языке ассемблер.

А) *Б*)

РИС. 21 ОКНО ИСХОДНОГО ТЕКСТА (*А*) И ОКНО ДИЗАССЕМБЛЕРА (*Б*) В РЕЖИМЕ ОТЛАДКИ

Рис. 22

В пошаговом режиме выполнения программы информация во всех окнах автоматически обновляется на каждом шаге. Изменения в регистрах микроконтроллера выделяются синим цветом на закладке Regs (рис. 21). В режиме непрерывного выполнения программы обновление окон производится нажатием кнопки Update Window, которая находится в окне Toolbox (рис. 22), открываемом кнопкой .

В исходном тексте программы и дизассемблере можно устанавливать Breakpoint – точки прерывания, в которых происходит остановка выполнения программы. Управление точками прерывания осуществляется соответствующими командами из меню Debug. Для некоторых из них на панель File Toolbar вынесены кнопки.

– поставить/удалить точку прерывания

– удалить все точки прерывания

– разрешить/запретить точки прерывания

– запретить все точки прерывания

Результат работы вашей программы можно наблюдать в соответствующих окнах.

- Изменения содержимого системных регистров микроконтроллера контролируется в окне проекта на закладке Regs (рис. 21).
- Изменения содержимого памяти контролируется в окне Memory Window, которое включается кнопкой .
- Изменение значений переменных контролируется в окне Watch & Call Stack Window, которое

включается кнопкой .

Имеется возможность проверки работы программы с такими устройствами микроконтроллера, как система прерываний, порты ввода/вывода, последовательные порты, таймеры/счетчики, АЦП и др. Для этого нужно из пункта меню Peripherals выбрать соответствующую команду.

Кроме того, вы можете создавать отладочные функции, которые генерируют внешние прерывания, периодически обновляют сигналы на цифровых и аналоговых входах и подают данные на вход последовательного порта.

Загрузка программы в память микроконтроллера

Выходной Hex-файл, получаемый при обработке проекта содержит информацию, которая предназначена для загрузки в целевой микроконтроллер. В структуре Hex-файла имеются данные и адреса их размещения, поэтому он может быть использован для загрузки в программатор или эмулятор ПЗУ.

Ваш лабораторный микроконтроллер EB552 снабжен эмулятором ПЗУ объемом 32 Кб. Загрузка Hex-файла в эмулятор ПЗУ осуществляется с компьютера через последовательный интерфейс RS-232. Для этого необходимо специальным кабелем подключить лабораторный микроконтроллер к COM-порту компьютера.

Процесс загрузки производится под управлением командного файла Load.bat. Пример командного файла:

```
mode COM1 96,N,8,1
copy filename.hex com1:
```

Первая строка настраивает режим работы порта COM1 компьютера, а вторая копирует Hex-файл в этот порт.

Вам необходимо поместить файл Load.bat в директорию своего проекта, открыть его и изменить имя Hex-файла на то, которое введено в Options for Target на закладке Output (рис. 10, указатель 1).

Для удобства работы можно настроить запуск командного файла Load.bat непосредственно из среды Keil µVision2. При этом создается инструмент пользователя с помощью команды Customize Tools Menu из меню Tools. Данная команда открывает окно (рис. 23), в котором с помощью кнопки в поле (рис. 23, указатель 1) вводится название нового инструмента (например, Загрузка). Затем нужно в поле Command указать имя исполняемого файла и путь к нему. Проще это сделать в режиме обзора, нажав кнопку ... (рис. 23, указатель 2) и выбрав соответствующий файл. Когда в поле Command появится путь к файлу Load.bat следует нажать кнопку OK.

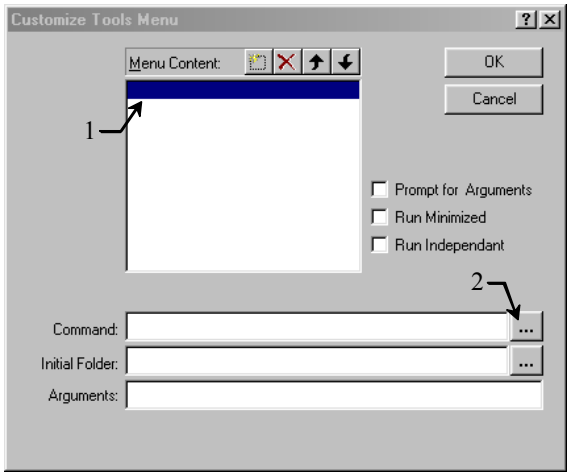


Рис. 23 Окно команды Customize Tools Menu

Теперь в меню Tools появился новая команда – инструмент пользователя Загрузка. Выбор этой команды запускает командный файл Load.bat, в результате чего происходит загрузка вашей программы через COM-порт в целевой микроконтроллер.

Используемое оборудование

1 Персональный компьютер IBM PC.
2 Лабораторный микроконтроллер EB552.
3 Блок питания для микроконтроллера.
4 Кабель для подключения микроконтроллера к компьютеру.

Подготовка к выполнению работы

1 Включите персональный компьютер, введите свой пароль и загрузите операционную систему.
2 Соедините кабелем COM-порт компьютера с последовательным портом лабораторного микроконтроллера.
3 Подключите кабельный разъем блока питания к микроконтроллеру.
4 Включите блок питания в розетку 220В.
5 Убедитесь, что на индикаторе микроконтроллера появилось сообщение о готовности к работе.

Порядок выполнения работы

1 Запустите программу Keil µVision2.
2 Создайте новый проект для микроконтроллера 80C552 фирмы Philips.
3 Скопируйте из директории **I:\микропроцессоры\Keil\Examples for EB552** в директорию, где находится ваш проект следующие файлы:

Hello.c – исходный текст демонстрационной программы на языке Си;
Startup.a51 – системная программа на языке Ассемблер;
Stdio1.h – заголовочный файл для функций ввода-вывода, модифицированный для отображения информации на индикаторе.

4 Включите в проект файлы **Hello.c** и **Startup.a51**.
5 Настройте проект под лабораторный микроконтроллер EB552 с эмулятором ПЗУ.
З а к л а д к а Target
- Memory Model – Small
- Rom Code Size – Large
- Off-chip Code memory: Start 0x8000; Size 0x8000
- Off-chip Xdata memory: Start 0x0000; Size 0x8000

З а к л а д к а Output
- Create Executable
- Create Hex File: Hex-80

З а к л а д к а C51
- Level: 8
- Emphasis: Favor speed
- Начальный адрес вектора прерывания: 0x8000

З а к л а д к а Debug
- Use Simulator
- Load Application at Startup
- Go till main ()

6 Настройте шрифт в окне проекта Editor C files.
7 Ознакомьтесь с текстом демонстрационной программы.
8 Обработайте проект с помощью команды Rebuild All Target Files и проконтролируйте результат ее выполнения в окне выходных сообщений.
9 Запустите отладчик и проверьте работу программы в пошаговом режиме.
10 Создайте инструмент пользователя для загрузки Hex-файла.
11 Загрузите программу в эмулятор ПЗУ лабораторного микроконтроллера и проверьте ее работу.

Контрольные вопросы

1 Что такое µVision2?
2 Какие функции выполняет интегрированная среда µVision2?
3 Какие компоненты входят в состав интегрированной среды µVision2?
4 Что такое компилятор?
5 Что такое компоновщик?
6 Поясните процесс разработки программного обеспечения в среде µVision2?
7 На каких языках можно писать программы в среде µVision2?
8 Сравните эти языки по их возможностям и удобству использования.
9 Что такое проект в среде µVision2?
10 Как организуется структура проекта?
11 Как производится настройка параметров проекта?
12 Как производится обработка проекта?
13 Какие средства отладки предоставляет разработчику программ среда µVision2?
14 Что такое точка прерывания?
15 Как производится загрузка программного кода в целевой микроконтроллер?
16 Что такое эмулятор ПЗУ?

Лабораторная работа 3

ИЗУЧЕНИЕ УСТРОЙСТВА И ПРОГРАММИРОВАНИЕ КЛАВИАТУРЫ МИКРОПРОЦЕССОРНЫХ СИСТЕМ

Цель работы:
1 Изучить принципы организации клавиатуры в МП-системах.
2 Познакомиться с устройством матричной клавиатуры.
3 Изучить способ динамического опроса клавиатуры.
4 Получить навыки практической реализации клавиатуры в МП-системе.

Методические указания

Для ручного ввода информации в МП-системы используются различные коммутационные элементы: переключатели, кнопки, клавиши. На их основе строятся целые блоки клавиатур, которые могут включать более ста кнопок.

Способы подключения клавиатуры

Простейшая клавиатура может состоять из нескольких кнопок, подключенных к такому же количеству входов порта ввода МП-системы. Если это 8-ми разрядный порт, то к нему можно подключить 8 кнопок, как это показано на рис. 1.

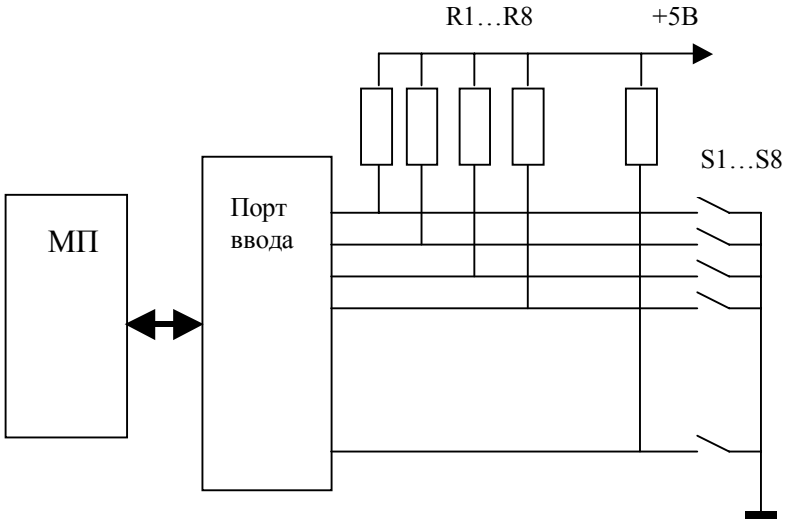


Рис. 1 Схема подключения простейшей клавиатуры

В исходном состоянии (кнопки не нажаты) на всех входах порта присутствует логический уровень 1 благодаря резисторам R1…R8, соединенным с напряжением питания +5 В. При нажатии любой кнопки происходит замыкание соответствующего входа порта с корпусом схемы и на нем появляется логический уровень 0. Для того чтобы МП-система могла определить факт нажатия одной или нескольких кнопок клавиатуры ей необходимо осуществить ввод информации с данного порта. Если получены все единицы (код FFh), то это означает, что ни одна кнопка не нажата. Если один или несколько бит содержат логические 0 – следовательно, произошло нажатие соответствующих кнопок. Для идентификации нажатых кнопок необходима дополнительная программная обработка вводимой информации.

Можно повысить эффективность использования линий портов, если применить матричную организацию и динамический способ опроса (сканирование) кнопок клавиатуры. Для построения матрицы проводники располагают в виде сетки, в узлах которой размещают кнопки (рис. 2). При нажатии кнопки замыкаются соответствующие линии строк и столбцов.

К выводам порта МП-системы подключаются строки и столбцы матрицы клавиатуры. Для определения факта нажатия и идентификации нажатой кнопки МП-системы осуществляет сканирование столбцов позиционным кодом и ввод информации со строк. При этом нажатая клавиша определяется программой по номерам строки и столбца. Такой способ опроса клавиатуры называется динамическим.

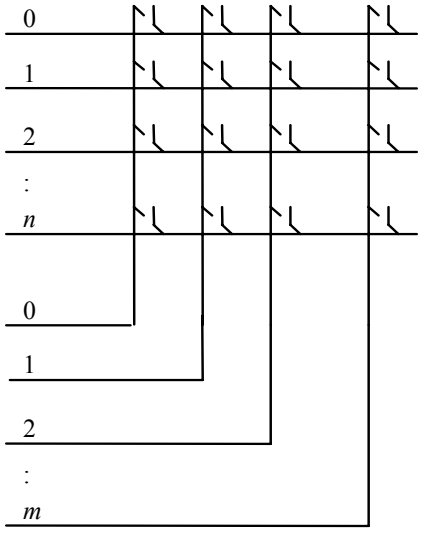


Рис. 2 Схема включения кнопок в матрицу

Организация динамического опроса клавиатуры

В качестве примера рассмотрим подключение к микроконтроллеру 80C51 матрицы клавиатуры 4×3 кнопки. К микроконтроллеру DD1 подключено 12 кнопок S1 – S12 через порт P1 (рис. 3, *а*). Сканирование столбцов производится 4–6 разрядами порта, которые используются как выходы. Ввод информации со строк производится 0–3 разрядами того же порта, которые используются как входы.

В процессе сканирования на один из выходов P1.4, P1.5, P1.6 поочередно выводится логический 0 (рис. 3, *б*). На входах порта P1.0 – P1.3 изначально поддерживается уровень логической 1 за счет "подтяжки" внутренними резисторами к источнику питания. Следовательно, если ни одна кнопка не нажата, то при вводе с порта P1 на этих входах будут получены 1. Если же одна из кнопок будет нажата, то через замкнутый контакт логический 0 выводимый при сканировании на соответствующий столбец матрицы попадет на соответствующую строку. В результате при вводе с порта P1 будет получен логический 0 на входе подключенному к этой строке.

Координаты нажатой кнопки определяются номером сканируемого столбца и номером строки, с которой получен логический 0. На рис. 3, *б* показаны сигналы на линиях порта P1 при нажатии кнопок во время сканировании клавиатуры. Нажатие первой кнопки Кн.1 вызывает появление 0 на P1.0 при сканировании столбца подключенному к P1.4. Следовательно, ее координаты: строка № 1; столбец № 1. По аналогии, другие нажатые кнопки будут иметь координаты:

Кн.2 – строка № 3; столбец № 2;

Кн.3 – строка № 2; столбец № 1;

Кн.4 – строка № 4; столбец № 1.

Алгоритм сканирования матрицы и отображения кода нажатой клавиши на индикаторе поясняет блок-схема алгоритма, представленная на рис. 4.

Сначала в блоке 1 производится настройка индикатора для отображения информации. Далее алгоритм представляет собой циклический процесс, в котором выполняются блоки 2 – 8.

В блоке 2 инициализируется переменная SCAN для опроса первого столбца матрицы.

Затем в блоке 3 производится вывод значения переменной SCAN в порт P1 для сканирования матрицы.

В блоке 4 осуществляется ввод с порта P1 и присвоение этого значения переменной KEY.

В блоке 5 выполняется проверка факта нажатия кнопки в опрашиваемом столбце. Если ни одна кнопка не нажата, то переменная KEY в двоичных разрядах 0 – 3 будет содержать логические 1 и проверка даст отрицательный результат. Если же хотя бы одна кнопка нажата, то переменная KEY в соответствующем двоичном разряде будет содержать логический 0, следовательно, проверка даст положительный результат.

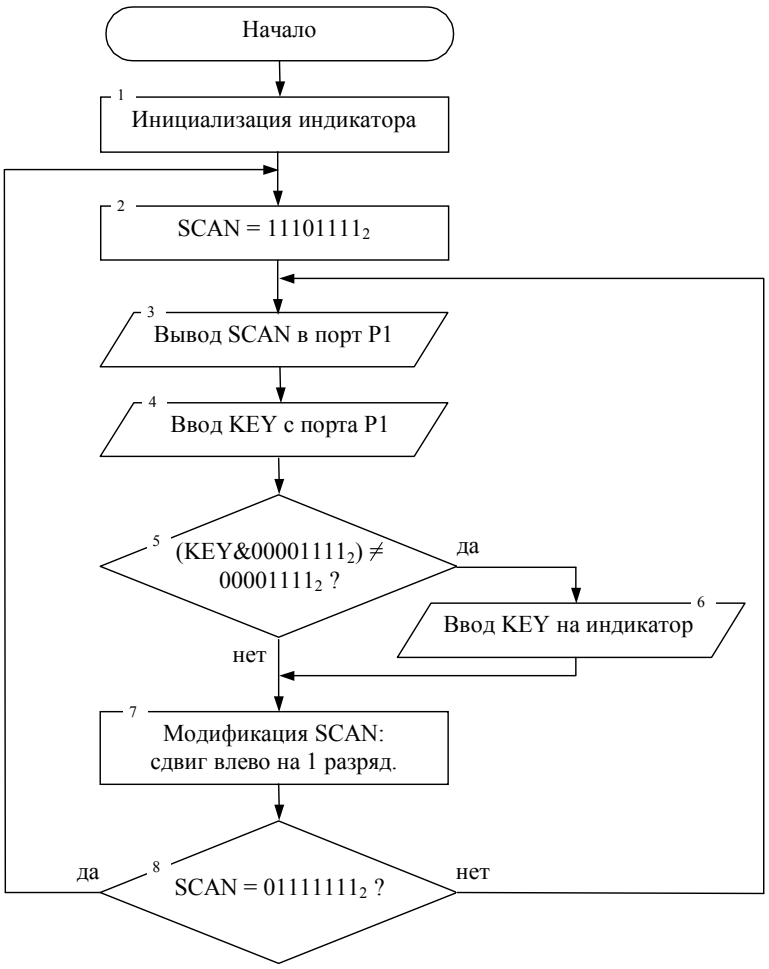


Рис. 4 Блок-схема алгоритма динамического опроса клавиатуры

При обнаружении нажатой кнопки выполняется блок 6, в котором производится вывод кода этой кнопки на индикатор. В противном случае блок 6 пропускается, и программа переходит к выполнению блока 7.

Блок 7 осуществляет модификацию переменной SCAN путем сдвига влево для сканирования следующего столбца.

В блоке 8 производится проверка окончания сканирования последнего столбца. Если условие выполняется, то программа переходит к блоку 2, и процесс начинается с опроса первого столбца. Если нет, то программа переходит к блоку 3, и процесс продолжается опросом следующего столбца.

Устройство лабораторного стенда

На рис. 5 представлена функциональная схема лабораторной установки для изучения динамического способа опроса клавиатуры. Матричная клавиатура подключена к порту P4 микроконтроллера 80C552. Индикатор позволяет отображать буквенно-цифровые символы в поле 2×24 знакоместа. Регистры индикатора отображены в область внешней памяти микроконтроллера. Для работы с индикатором можно использовать процедуры-функции wrc( ); wrd( ), приведенные в примерах программ на языке Си.

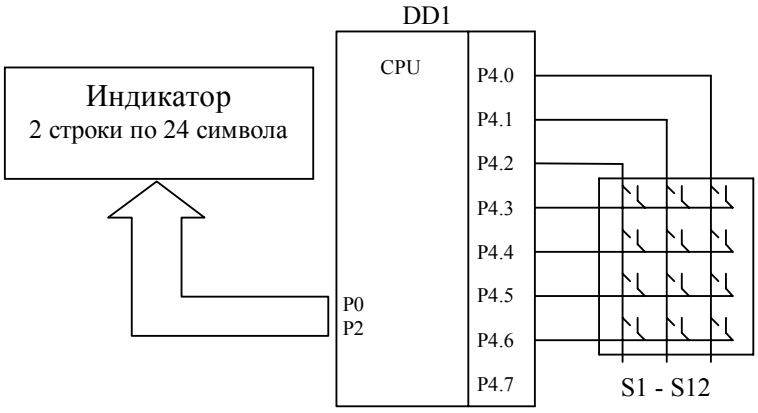


Рис. 5 Функциональная схема лабораторного стенда

Задание для практической работы

1 Изучить схему лабораторной установки.

2 Составить программу на языке СИ для динамического опроса клавиатуры. Программа должна выводить на индикатор код нажатой клавиши в шестнадцатеричной и двоичной системах.

Формат отображения:

	K	**E**	**Y**		**H**	**E**	**X**		**C**	**O**	**D**	**E**	**:**		**X**	**X**				
	K	**E**	**Y**		**B**	**I**	**N**		**C**	**O**	**D**	**E**	**:**		**B**	**B**	**B**	**B**	**B**	**B**	**B**	**B**

где XX – шестнадцатеричное значение; BBBBBBBB – двоичное значение.

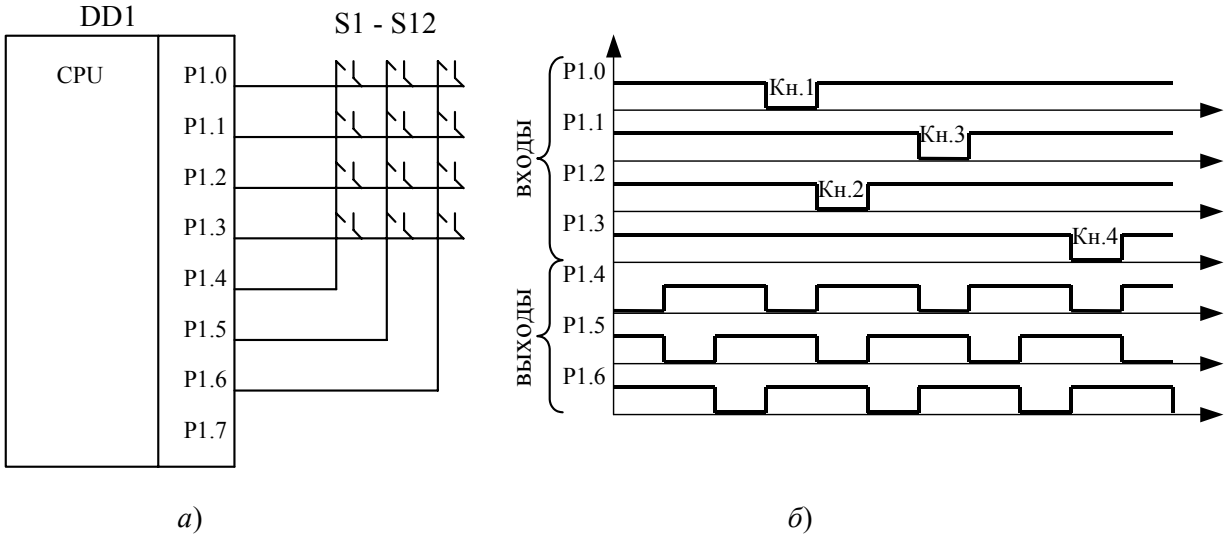
3 Выполнить обработку программы в интегрированной среде Keil и получить Hex-файл исполняемой программы.

4 Загрузить программу в эмулятор ПЗУ лабораторного микроконтроллера и произвести тестирование. При необходимости произвести доработку программы и повторную проверку. Окончательный результат показать преподавателю.

Контрольные вопросы и задания

1 Для чего служит клавиатура в МП-системах?
2 Как подключается клавиатура в МП-системе?
3 Какие существуют способы опроса клавиатуры?
4 Объясните принцип динамического опроса клавиатуры.
5 В чем преимущество динамического опроса клавиатуры?
6 Какой код используется для сканирования матрицы, объясните почему?
7 Поясните схему лабораторного стенда.
8 Объясните работу своей программы.
9 Что следует изменить в программе, если будет использоваться матрицу 8 × 8?

Рис. 2 Классификация запоминающих устройств



а) *б*)

Рис. 3 Подключение матрицы клавиатуры к микроконтроллеру 80C51