ВВЕДЕНИЕ В ПРОГРАММИРОВАНИЕ МИКРОКОНТРОЛЛЕРОВ

Методические материалы по учебной дисциплине
«Программирование микропроцессорных систем»
для специальности
1-31 04 02 Радиофизика

2019 г.

СОСТАВИТЕЛИ:

П.П. Коржуков, старший преподаватель кафедры системного анализа и компьютерного моделирования Белорусского государственного университета.

Текст представлен в авторской редакции.

РЕКОМЕНДОВАНА К РАЗМЕЩЕНИЮ В ЭЛЕКТРОННОЙ БИБЛИОТЕКЕ БГУ:

Кафедрой системного анализа и компьютерного моделирования Белорусского государственного университета
(протокол № 8 от 22 января 2019 г.);

1 МИКРОПРОЦЕССОРНЫЕ СИСТЕМЫ

1.1 Общие сведения

Часть важных требований, которые потребители предъявляют к вычислительным и управляющим блокам сложных электронных устройств и приборов, можно сформулировать следующим образом:

- низкая стоимость изделия;
- высокая надежность в работе;
- высокая степень миниатюризации в исполнении;
- малое энергопотребление в основных режимах работы;
- работоспособность в жестких условиях эксплуатации;
- достаточная производительность для выполнения требуемых функций в полном объеме.

Даже этот короткий перечень представляет собой набор довольно противоречивых условий. Выполнение всех этих условий одновременно весьма и весьма затруднительно. Поэтому развитие и совершенствование интегральной техники БИС и СБИС пошло по пути специализации для узко ориентированного набора требований, и в настоящее время количество различных моделей микропроцессорных систем чрезвычайно велико. Микропроцессора, универсального для всех случаев в жизни, нет.

Микропроцессор (МП) – это самостоятельное или составное электронное цифровое устройство, управляемое по программе и предназначенное для обработки цифровой информации и управления процессом этой обработки, построенное на одной или нескольких интегральных схемах с высокой степенью интеграции электронных элементов.

МП – это электронная система, иначе микропроцессорная система (англ. Microprocessor System). Состоит из многих электронных блоков, т.е. подсистем. Деление на блоки не жесткое, и определяется разработчиком. Таких подсистем в МП достаточно много. Важнейшие из них перечислены ниже в следующих абзацах.

- Выполнение всех арифметических и логических операций над числовой и символьной информацией осуществляет **арифметико-логическое устройство** (АЛУ, англ. CPU).
- Слаженное взаимодействие различных подсистем МП и координацию потоков данных обеспечивает **устройство управления**.
- Микропроцессорная **память** предназначена для кратковременного хранения, записи и выдачи информации, используемой в вычислениях непосредственно в ближайшие такты работы АЛУ. Микропроцессорная память строится на **регистрах** и используется для обеспечения высокого быстродействия.

- Интерфейсная система МП предназначена для связи с другими устройствами. **Порты ввода-вывода** – это аппаратура сопряжения.

По своему основному применению выделяют три основные категории микропроцессоров:

Микропроцессор	**Микроконтроллер**	**Сигнальный процессор**
Универсальный МП.	Однокристальная микро-ЭВМ или МК (англ. MCU).	Цифровой сигнальный процессор (ЦСП, англ. DSP).

Универсальные микропроцессоры ориентированы, в первую очередь, на высокую производительность вычислений при обработке цифровой и символьной информации. На их основе строятся компьютеры и различные варианты специальных вычислителей.

Микрокомпьютер – это устройство обработки произвольных данных, содержащее один или несколько микропроцессоров, БИС постоянной и оперативной памяти, БИС управления вводом и выводом информации и некоторые другие микросхемы. Микрокомпьютер такого состава иногда называют «голым» из-за отсутствия в нем периферийных устройств.

Понятие микрокомпьютера со временем менялось. С конца 1970-х годов до конца 1980-х годов этот термин обозначал компьютер, построенный на основе МП. В эти же годы имел хождение синоним **микро-ЭВМ**. Обе трактовки вышли из употребления в 1990-е годы. Тогда это понятие было вытеснено термином **персональный компьютер** (ПК).

Начиная с 2010-х годов, термин микрокомпьютер в связи с общей миниатюризацией употребляется в новом значении. Уже вошли в употребление одноплатные вычислители размером с банковскую карту и сопоставимые по производительности с персональными компьютерами предыдущих поколений. Пример – Raspberry Pi, представленный на рисунке 1. Также микрокомпьютерами иногда называют сами микроконтроллеры.

На плате размещена контактная панель для сопряжения с дополнительной периферией. Присутствуют обычные интерфейсы для ПК-периферии: USB, Fast Ethernet, видеовыход HDMI. Для подачи внешнего питания – миниджек.

Рисунок 1 – Микрокомпьютер Raspberry Pi, 2011 г

Микрокомпьютер с небольшими вычислительными ресурсами и упрощенной системой команд, ориентированный не на производство вычислений, а на выполнение процедур логического управления различным оборудованием, называют программируемым микроконтроллером или просто микроконтроллером (МК).

Микроконтроллер – это БИС, которая заключает в себе все составляющие компьютера: процессор, оперативную и постоянную память, порты данных и прочее, что расширяет его функциональность.

Термин микроконтроллер (англ. Micro Controller Unit, MCU) в англоязычных литературных источниках встречается в виде аббревиатур: **MCU** для microcontroller unit, или **UC** для µ-controller.

В отличие от универсальных МП к управляющим контроллерам, как правило, не предъявляются высокие требования к производительности и программной совместимости.

Микросхема МК во время своего существования выступает в двух своих представлениях. В период программирования МК – это **микросхема ПЗУ**. В рабочем режиме МК – это в первую очередь **микросхема МП**, а во вторую – все его периферийные устройства.

Производство микроконтроллеров ежегодно во много раз превышает производство МП, и потребность в них все время не снижается. На 2009 год по данным Gartner Grup мировой рейтинг по объёму продаж микроконтроллеров выглядит следующим образом:

- На первом месте с существенным отрывом стоит японская компания Renesas Electronics;
- На втором месте – Freescale;
- На третьем месте – Samsung;
- На четвертом месте идут Microchip и Texas Instruments;
- Далее все остальные производители.

Для противостояния действиям технического шпионажа микроконтроллеры имеют программируемые ячейки защиты информации. В первых микроконтроллерах это были плавкие перемычки. На профессиональном жаргоне в продолжение традиции сейчас название этих ячеек защиты – **FUSE биты** или просто фьюзы (англ. fuse – плавить). Запись и чтение этих ячеек возможны только в режиме программирования. Определенная комбинация этих ячеек блокирует запись в память программ и энергонезависимую память данных. Блокируется также возможность чтения ПЗУ (сейчас Flash памяти), после чего узнать содержимое оригинальной программы МК становится невозможным.

Использование ячеек защиты противодействует и конкуренции на рынке готовых изделий, сдерживая ее.

Главное правило при работе с фьюзами – это не торопиться их выставлять. Их устанавливают только тогда, когда точно уверены в правильности своих действий.

Микроконтроллеры скромно прячутся внутри самых разнообразных устройств, в том числе и в компьютерах. Без микроконтроллеров не могло бы работать ни одно из периферийных устройств компьютера, таких как клавиатура, мышь, дисковод, принтер, сканер и т.д. МК применяются преимущественно во встроенных системах, в автомобилях, в приборах, в станках, в массовой домашней технике (микроволновках, стиральных машинах, музыкальных центрах и т.п.) и домашней автоматике, в игрушках.

В общем, где нужна не вычислительная мощность процессора, а, скорее, баланс между ценой и достаточной функциональностью, применяется микроконтроллер. Именно поэтому самые старые типы микроконтроллеров еще до сих пользуются спросом. Они многое могут: от автоматического открывания дверей и включения полива газонов до интеграции в систему «Умный дом».

Микроконтроллер компании Intel i8051 – невероятный долгожитель. Он – самый популярный микроконтроллер для встраиваемых систем за всю их историю. И хотя еще в 1996 году i8051 был заменен Intel на более совершенную модель, его клоны по сей день продолжают выпускаться десятками компаний. Семейство 8-разрядных микроконтроллеров i8051, разработанное корпорацией Intel, в наше время практически стало промышленным стандартом.

Цифровой сигнальный процессор отличается тем, что в его архитектуре предусмотрен накапливающий аккумулятор. В этом случае совмещаются операции умножить и сложить. Такая система команд ориентирована на эффективное решение задач цифровой обработки сигналов (ЦОС), а именно фильтрации и спектрального анализа, т.е. вычисления преобразования Фурье.

Микропроцессорная система (МПС) – это функционально законченное электронное изделие, которое состоит из одного или нескольких устройств, главным образом из микропроцессора и/или микроконтроллера.

Общее для всех МП или МПС то, что работают они довольно однообразно, все они работают под управлением программы и требуют программирования для конкретного применения.

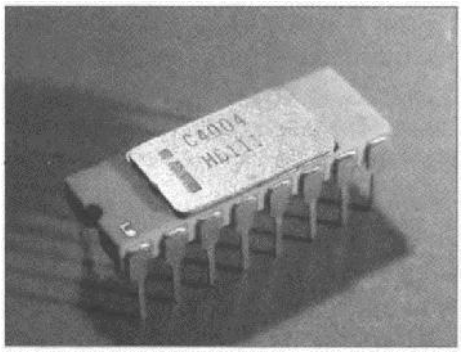
1.2 Исторический экскурс

Исторически появились практически одновременно три проекта по созданию микропроцессора. Эти проекты:

- Central Air Data Computer (CADC) в компании Garrett AiResearch (1968);
- TMS 1000 в компании Texas Instruments (1971);
- 4004 в компании Intel (1971).

Компании Texas Instruments (TI) и Intel практически одновременно и независимо создали первый микропроцессор. В 1971 году компания Texas Instruments разработала однокристальный микрокомпьютер. В 1973 году получила патент на однокристальный микропроцессор, основным разработчиком которого был сотрудник TI Гари Бун.

1970 г. – Маршиан Эдвард Хофф из фирмы Intel разработал первый микропроцессор Intel-4004. **15 ноября 1971 г.** – начало новой эры в электронике, поставка в продажу первого в мире микропроцессора Intel-4004. На рисунке 2 представлена эта легендарная БИС.



Металлокерамический корпус DIP для МП. 16 выводов.

Рисунок 2 – Микропроцессор Intel i4004

МП Intel-4004 размером менее 3 см был производительнее гигантской машины ENIAC. Но работал он гораздо медленнее и мог обрабатывать одновременно только 4 бита информации. В это время процессоры больших ЭВМ обрабатывали 16 или 32 бита одновременно. Однако стоил первый МП в десятки тысяч раз дешевле. Его основные характеристики:

- 4-разрядный процессор с классической архитектурой ЭВМ гарвардского типа;
- p-канальная МОП-технология с проектными нормами 10 мкм;
- 2300 транзисторов;
- тактовая частота 750 кГц при длительности цикла команд 10,8 мкс;
- адресный стек (счетчик команд и три регистра стека типа LIFO);
- блок 16 РОНов 4-разрядных (регистры сверхоперативной памяти или регистровый файл – РФ), можно использовать как восемь 8-разрядных;
- 4-разрядное параллельное АЛУ;
- аккумулятор, регистр команд с дешифратором команд и схемой управления;

- схема связи с внешними устройствами;
- 4-разрядная ШД.

Память команд достигала 4 кб. Для сравнения: объем ЗУ мини-ЭВМ в начале 70-х годов редко превышал 16 кб.

Три регистра стека обеспечивали три уровня вложения подпрограмм.

Пластмассовый или металлокерамический корпус типа DIP (Dual In-line Package) с 16 выводами.

В систему команд входило всего 46 инструкций.

Располагал весьма ограниченными средствами ввода/вывода.

В системе команд отсутствовали операции логической обработки данных (И, ИЛИ, ИСКЛЮЧАЮЩЕЕ ИЛИ), их приходилось реализовывать с помощью специальных подпрограмм.

МП i4004 не имел возможности останова (команды HALT) и обработки прерываний.

Цикл команды процессора состоял из 8 тактов задающего генератора.

Была мультиплексированная ША (шина адреса)/ШД (шина данных), 12-разрядный адрес передавался по 4 разряда.

1 апреля 1972 г. начались поставки первого 8-разрядного МП i8008. МП изготавливался по p-канальной МОП-технологии с проектными нормами 10 мкм. Содержал 3500 транзисторов. Работал на частоте 500 кГц при длительности машинного цикла 20 мкс, что составляло 10 периодов задающего генератора. Архитектура ЭВМ принстонского типа. Для памяти допускал применение комбинации ПЗУ и ОЗУ.

Число РОН – 8, из них два регистра использовались для хранения адреса при косвенной адресации памяти. Сократилась длительность машинного цикла с 8 до 5 состояний. Для синхронизации работы с медленными устройствами был введен сигнал готовности READY.

Система команд насчитывала 65 инструкций. МП мог адресовать память объемом 16 кб. Производительность, по сравнению с четырехразрядными МП, возросла в 2,3 раза. В среднем для сопряжения процессора с памятью и устройствами ввода/вывода требовалось около 20 схем средней степени интеграции.

1 апреля 1974 – МП Intel 8080. Технология n-МОП (n-MOS) с проектными нормами 6 мкм. Содержал 6 тыс. транзисторов. Тактовая частота процессора доведена до 2 МГц, а длительность цикла команд составила 2 мкс. Объем адресуемой памяти увеличен до 64 кб. Используя 40-выводной корпус удалось разделить ША и ШД. Общее число микросхем для построения системы в минимальной конфигурации сократилось до 6.

В РФ введен указатель стека, активно используемый при обработке прерываний, а также два программно недоступных регистра для внутрен-

них пересылок. Блок РОНов реализован на микросхемах статической памяти. Аккумулятор вывели из РФ и включили в состав АЛУ, что упростило схему управления внутренней шиной. Новое – использование многоуровневой системы прерываний по вектору, что позволило довести общее число источников прерываний до 256 (до появления БИС контроллеров прерываний схема формирования векторов прерываний требовала применения до 10 дополнительных чипов средней интеграции). В i8080 появился механизм прямого доступа в память (ПДП) (как ранее в универсальных ЭВМ IBM System 360 и др.). ПДП открыл зеленую улицу для применения в микро-ЭВМ таких сложных устройств, как накопители на магнитных дисках и лентах дисплеи на ЭЛТ, которые и превратили микро-ЭВМ в полноценную вычислительную систему. Традицией компании, начиная с первого кристалла, стал выпуск не отдельного чипа ЦП, а семейства БИС, рассчитанных на совместное использование.

В **1976 г.** фирма Intel создала первый МК – i8048. Помимо ЦП, в его состав входила память программ, память данных, восьми битный таймер и 27 линий ввода/вывода. Адресуемая память команд – 4 кб. В системе команд 96 основных инструкций.

1980 г. – рождение МК i8051. Создан на высокоуровневой n-МОП технологии (HMOS). В настоящее время среди всех 8-разрядных микроконтроллеров семейство MCS-51 является несомненным чемпионом по количеству разновидностей и количеству компаний, выпускающих его модификации.

Удачный набор периферийных устройств, возможность гибкого выбора внешней или внутренней программной памяти и приемлемая цена обеспечили этому микроконтроллеру успех на рынке. С точки зрения технологии микроконтроллер i8051 являлся для своего времени очень сложным изделием – в кристалле было использовано 128 тыс. транзисторов, что в 4 раза превышало количество транзисторов в 16-разрядном микропроцессоре i8086.

На сегодняшний день существует более 200 модифицированных вариантов микроконтроллеров семейства i8051, выпускаемых почти 50-ю компаниями. Эти модификации включают в себя кристаллы с широчайшим спектром периферии: от простых 20-выводных устройств с одним таймером и 1K программной памяти до сложнейших 100-выводных кристаллов с 10-разрядными АЦП, массивами таймеров-счетчиков, аппаратными 16-разрядными умножителями и 64K программной памяти на кристалле. Каждый год появляются все новые варианты представителей этого семейства.

Основными направлениями развития являются:

- увеличение быстродействия за счет повышение тактовой частоты и переработки архитектуры;
- снижение напряжения питания и энергопотребления;
- увеличение объема ОЗУ и FLASH памяти на кристалле с возможностью внутрисхемного программирования;
- введение в состав периферии микроконтроллера сложных устройств типа системы управления приводами;
- поддержка интерфейсов I^2C и SPI;
- поддержка интерфейсов CAN и USB.

Основными производителями клонов 51-го семейства в мире являются фирмы NXP (Philips), Infineon (Siemens), Intel, Acer Labs, Actel, Altium, Microchip (Atmel), Analog Device, Cast Ins, Chipcon, Dallas, Digital Core Design, Hynix Semiconductor, Maxim, Temic, Oki, AMD, MHS, Gold Star, Winbond, Silicon Systems, Silicon Laboratories (Cygnal) и ряд других.

Примером современного микроконтроллера, разработанного в компании Atmel, является МК **AT89C51RC**. Это – мало мощный, высокоэффективный 8-битный микроконтроллер CMOS с 32 КБ flash программируемой ROM и 512 байтами RAM. Он использует **Microchip** высокоплотную энергонезависимую технологию памяти и совместим с набором команд и схемой контактов промышленных стандартов 80C51 и 80C52. Память flash на кристалле позволяет пользователю запрограммировать программный код обычным программатором энергонезависимой памяти.

Это универсальное 8-битное микропроцессорное устройство с flash памятью обеспечивает высоко гибкое и экономичное решение для многих встроенных приложений управления. Вместо DPTR имеет два 16-разрядных регистра указателя данных для упрощения доступа к внутренней и внешней памяти данных. Добавлены сторожевой таймер и усовершенствованный таймер/счетчик T2 с регистром захвата.

Исполнения МК в разных корпусах показаны на рисунке 3.

Корпус на 44 вывода 44J – PLCC (Plastic J-leaded Chip Carrier)

Корпус на 40 выводов 40P6 – PDIP (Plastic Dual Inline Package)

Корпус на 44 вывода 44A – TQFP (Thin Plastic Gull Wing Quad Flatpack)

Рисунок 3 – Современный микроконтроллер AT89C51RC

1.3 Встроенная система

Наряду с термином встроенная система (англ. embedded system) равноценно применяется термин встраиваемая система.

Встраиваемая система – специализированная микропроцессорная система управления или контроля, или мониторинга состояния, которая работает, будучи встроенной непосредственно во внешнее устройство, которым она управляет.

Среди первых встроенных систем можно выделить компьютер D-17, который управлял американской межконтинентальной баллистической ракетой Minuteman I. Примерами встроенных систем в повседневной жизни могут служить банкоматы, авиационное компьютерное оборудование, телекоммуникационное оборудование, мобильные телефоны и т.д..

Авиационная и космическая промышленность всегда ставили жесткие требования к весу и габаритам используемых устройств. В основе построения современных встроенных систем часто используются микроконтроллеры, одноплатные или однокристальные микрокомпьютеры, специализированные или универсальные микропроцессоры, ПЛИС.

Встроенные компьютерные системы имеют малые геометрические размеры и отличаются низким энергопотреблением. Часто процессорное ядро и другие интегральные схемы всей системы совмещаются на одном кристалле, образуя единое целое (англ. SOC – System-on-a-Chip, произносится как «эс-оу-си»). Термины-синонимы – **система на кристалле** (СнК), **однокристальная система**.

1.3.1 Пример из RFID-системы

Некоторые встроенные системы в наше время находят массовое применение. Примером тому могут служить , например, устройства **радиочастотной идентификации** (англ. **RFID** – Radio Frequency IDentification,).

Часть небольшую по габаритам, но весьма существенную, любой RFID-системы представляет простой **транспондер**, он же RFID-метка, иногда также применяется термин RFID-тег. Один из вариантов транспондера показан на рисунке 4. Это один из примеров массовой реализации встроенной системы.

Метка стандарта EPC Class 1 Generation 2

Рисунок 4 – RFID-метка (транспондер)

Наиболее известной реализацией RFID-системы стало семейство карт **Mifare**. На базе этого стандарта создана технология беспроводной связи NFC (Near Field Communication) для двустороннего обмена сообщениями и данными между гаджетами.

Для оплаты проезда на транспорте используются RFID карты (иначе смарт-карты), те же транспондеры. Смарт-карта представляет из себя колебательный контур с катушкой на пластине карты и конденсатором C_1, параллельно которому включена микросхема микроконтроллера. Резонансная частота – 13,56 МГц, что соответствует стандарту ISO 14443. В общем и целом принцип работы транспондера поясняет рисунок 5.

В широко распространенных смарт-картах, таких как MIFARE DESFire, SmartMX, SmartMX2 и Smart eID, микроконтроллер построен на основе распространенного ядра 80C51.

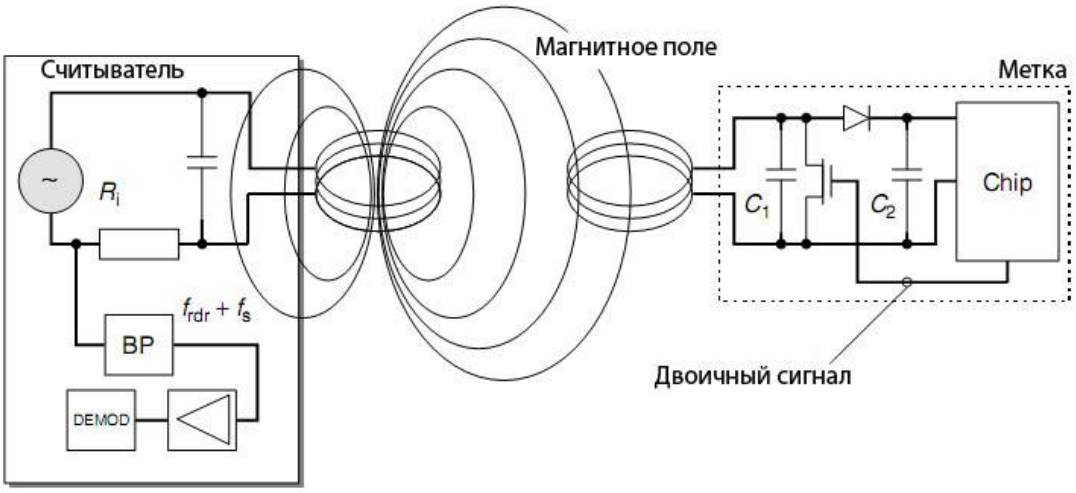
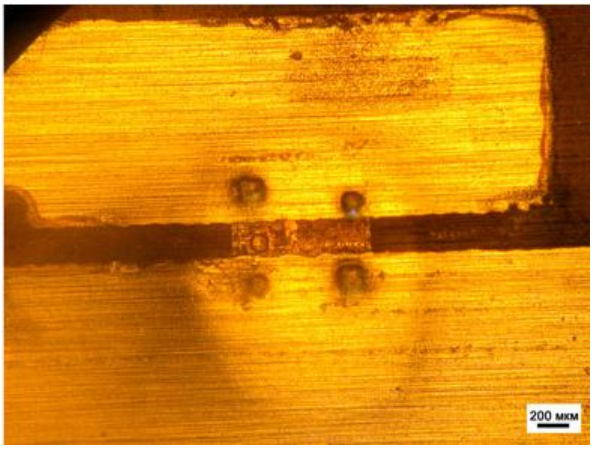


Рисунок 5 – Устройство и работа RFID-системы

Этот пример – случай пассивной метки. Антенна нужна для приема электромагнитных волн передатчика, в роли которого выступает считыватель, превращения их:

- в сигнал,
- в электроэнергию для питания самого микроконтроллера,
- и передачи ответного сигнала.

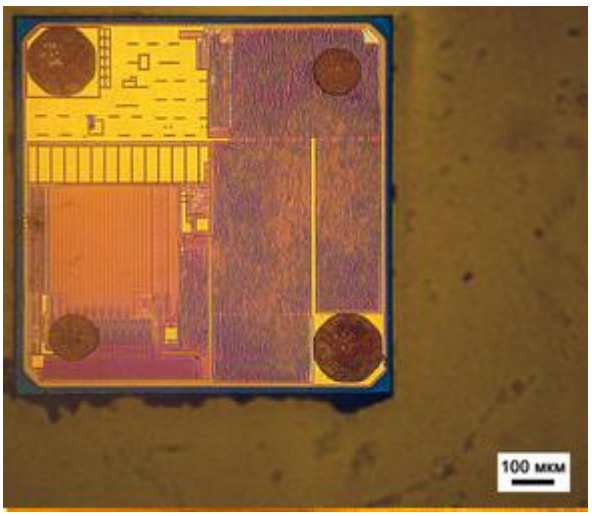
Крепление антенны к кристаллу микроконтроллера приведено на рисунке 6. А рисунок 7 представляет фотографию самого микроконтроллера.



Микросхема имеет 4 контакта. Проводники антенны крепятся либо парными контактами, либо одиночными.

Рисунок 6 – Микрофотография антенны с микросхемой RFID карты

На рисунке 7 слева внизу отчетливо показан модуль EEPROM памяти, так как он занимает около трети поверхности чипа и имеет регулярную структуру. ПЗУ программ занимает тоже не меньшую по площади часть поверхности чипа. Вычислительному ядру микроконтроллера остается совсем малая часть.



Фокус оптического микроскопа на верхнем слое металлизации

Рисунок 7 – Микрофотография микросхемы RFID карты

Это хорошо подтверждает известный закон Мура, который утверждает, что число транзисторов на интегральной микросхеме удваивается каждые 18 месяцев. Ограничения связаны с преодолением тепловыделения. С начала 1970-х годов рост мощности микропроцессоров существенно опережает рост объема памяти на кристалле.

Программы для встроенных систем массового производства должны быть оптимизированы в первую очередь по размеру кода.

1.4 Архитектуры микропроцессоров

Архитектур МП можно выделить несколько, но широко распространены два типа архитектур микропроцессоров. Эти названия связаны с двумя университетами США: Гарвардским и Принстонским.

• МП с **гарвардской архитектурой**.

Гарвардская архитектура была разработана Говардом Эйкеном в конце 1930-х годов в Гарвардском университете. В этой архитектуре выделяются две отличительные особенности:

- память машинных инструкций и память данных представляют собой разные физические устройства;
- шины данных для канала инструкций и канала данных также физически разделены.

• МП с **фон Неймановской** архитектурой (принстонской).

В архитектуре фон Неймана используется принцип хранимой программы вместе с данными.

Архитектура фон Неймана более экономно расходует аппаратные ресурсы, а гарвардская позволяет реализовать большее быстродействие.

Управление в МП может быть **схемное** и **микропрограммное**.

Все микропроцессоры можно разделить на группы по **архитектуре набора команд**. Таких архитектур много. Четыре важные из них, которые часто используются, перечислены ниже.

• МП типа **CISC** (Complex Instruction Set Computing) с полным набором команд.

• МП типа **VLIW** (Very Long Instruction Word) с очень длинными машинными командами.

Это архитектура МП с несколькими вычислительными устройствами. Здесь одна инструкция процессора содержит несколько операций, которые должны выполняться параллельно. Можно сказать, что это «видимое программисту» микропрограммное управление. В этом случае машинный код представляет собой лишь немного свёрнутый микрокод для непосредственного управления аппаратурой.

• МП типа **RISC** (Reduced Instruction Set Computing) с сокращенным набором команд.

• МП типа **MISC** (Minimum Instruction Set Computing) с минимальным набором команд и весьма высоким быстродействием (в настоящее время эти модели находятся в стадии разработки).

1.5 Общее представление о семействе MCS-51

Чрезвычайно популярный в течение 1980-х и начале 1990-х годов микроконтроллер Intel 8051 в пластиковом корпусе представлен на рисунке 8. На корпусе микросхемы в качестве фирменного знака стоит латинская буква «**i**». По этой причине в литературных источниках все изделия компании Intel сопровождаются этой буквой в записи устройства, например, i8051. Официальное же название 8051-семейства микроконтроллеров компании Intel – MCS 51.

Представитель семейства Intel MCS 51

Рисунок 8 – Микроконтроллер Intel P8051

Аббревиатура MCS относится к категории товарных знаков и торговых марок компании Intel.

MCS является аббревиатурой от английской фразы Micro-Control Sys tem. Эта аббревиатура дополнительно используется для обозначения мик-

ропроцессорной системы Intel (Intel Microprocessor System) и для обозначения микрокомпьютерного набора (Micro-Computer Set).

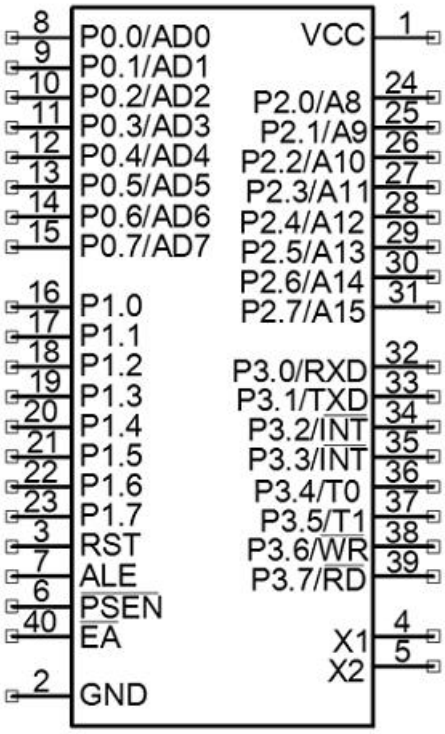
В родное семейство Intel MCS-51 входят следующие устройства:

- 8031-8-Bit Control-Oriented Microcontroller
- 8032-8-Bit Control-Oriented Microcontroller
- 8044-High Performance 8-Bit Microcontroller
- 8344-High Performance 8-Bit Microcontroller
- 8744-High Performance 8-Bit Microcontroller
- 8051-8-Bit Control-Oriented Microcontroller
- 8052-8-Bit Control-Oriented Microcontroller
- 8054-8-Bit Control-Oriented Microcontroller
- 8058-8-Bit Control-Oriented Microcontroller
- 8351-8-Bit Control-Oriented Microcontroller
- 8352-8-Bit Control-Oriented Microcontroller
- 8354-8-Bit Control-Oriented Microcontroller
- 8358-8-Bit Control-Oriented Microcontroller
- 8751-8-Bit Control-Oriented Microcontroller
- 8752-8-Bit Control-Oriented Microcontroller
- 8754-8-Bit Control-Oriented Microcontroller
- 8758-8-Bit Control-Oriented Microcontroller
- 8775 Microcontroller Peripheral I/O Port Expander W/32Kx8 EPROM
- 80151-8-Bit Control-Oriented Microcontroller
- 83151-8-Bit Control-Oriented Microcontroller
- 87151-8-Bit Control-Oriented Microcontroller
- 80152-8-Bit Control-Oriented Microcontroller
- 83152-8-Bit Control-Oriented Microcontroller
- 85154-8-Bit Piggyback Microcontroller
- 80251-8-Bit Control-Oriented Microcontroller
- 83251-8-Bit Control-Oriented Microcontroller
- 87251-8-Bit Control-Oriented Microcontroller

Назначение выводов микроконтроллера i8051, которое приведено на рисунке 9, дает первое общее представление о самом микроконтроллере и его возможностях. Связь с внешним аппаратным окружением осуществляется через 32 цифровые линии четырех портов.

В микроконтроллере заложена возможность дополнительно подключать внешнюю память данных и программ.

Основные обозначения, приведенные на рисунке 9, расшифровываются следующим образом:



ADx – данные / младший адрес
Ax – старший адрес

Рисунок 9 – Распиновка МК 8051 в DIP корпусе на 40 ножек

– GND (Vss) – потенциал общего провода ("земли");
– Vcc – основное напряжение литания +5 В;
– X1 (XTAL1), X2 (XTAL2) – выводы для подключения кварцевого резонатора;
– RST (RESET) – вход общего сброса микроконтроллера;
– PSEN – разрешение внешней памяти программ; выдается только при обращении к внешнему ПЗУ;
– ALE – строб адреса внешней памяти;
– EA – отключение внутренней программной памяти; уровень 0 на этом входе заставляет микроконтроллер выполнять программу только из внешнего ПЗУ; игнорируя внутреннее (если последнее имеется);
– RD – чтение внешней памяти данных;
– WR – запись во внешнюю память данных;
– P0 – восьми битный двунаправленный порт ввода-вывода информации: при работе с внешними ОЗУ и ПЗУ по линиям порта в режиме временного мультиплексирования выдается адрес внешней памяти, после чего осуществляется передача или прием данных;
– P1 – восьми битный квазидвунаправленный порт ввода/вывода: каждый разряд порта может быть запрограммирован как на ввод, так и на вывод информации, независимо от состояния других разрядов;

– P2 – восьми битный квазидвунаправленный порт, аналогичный P1; кроме того, выводы этого порта используются для выдачи адресной информации при обращении к внешней памяти программ или данных (если используется 16-битовая адресация последней);
– P3 – восьми битный квазидвунаправленный порт, аналогичный. P1; кроме того, выводы этого порта могут выполнять ряд альтернативных функций, которые используются при работе таймеров, порта последовательного ввода-вывода, контроллера прерываний, и внешней памяти программ и данных;
– INT – аппаратное прерывание (2 внешних);
– RXD – прием последовательных данных;
– TXD – передача последовательных данных;
– T0 и T1 – входы таймеров/счетчиков.

Когда числа представляются несколькими байтами, имеет существенное значение, в каком порядке эти байты записываются в памяти компьютера. Это влияет на реализацию **многозначной арифметики**. У 8-разрядных микропроцессоров многозначная арифметика реализуется для двухбайтовых и более чисел. Тип хранения числовых данных у семейства MCS-51 – от старшего к младшему (англ. **big-endian** – большим концом).

Большие числа записывается в позиционной системе счисления по основанию 256. Так десятичное число 67305985 в шестнадцатеричной системе счисления записывается как 4030201h в виде последовательности 4 байт. В этом примере младшее значимое число 1 – байт 01h – записывается по самому старшему адресу в памяти. Другими словами, число в своей записи кончается старшим адресом, отсюда и название. Рассматривая дамп памяти с данными, этот порядок соответствует привычному порядку записи арабских цифр слева направо.

Противоположный порядок хранения – от младшего к старшему (англ. little-endian – малым концом). Этот тип хранения принят для персональных компьютеров с процессорами семейства x86. Такой порядок соответствует логичному представлению более значимых чисел по старшему адресу: число в своей записи кончается младшим адресом.

Функциональная схема МК основоположника семейства MCS-51 приведена на рисунке 10. В основе построения устройства MCS-51 лежит шинная архитектура.

Главную основу образует внутренняя двунаправленная 8-битная шина, которая связывает между собой все основные узлы и устройства. Ядро системы включает резидентную память программ (RPM), резидентную память данных (RDM), арифметико-логическое устройство (ALU), блок регистров специальных функций (SFR), устройство управления (CU) и

порты ввода/вывода (P0-P3). Дополнив ядро системы конкретным набором периферийных устройств, получается конкретный вариант микроконтроллера. Управление всей периферией осуществляется через соответствующие регистры специальных функций.

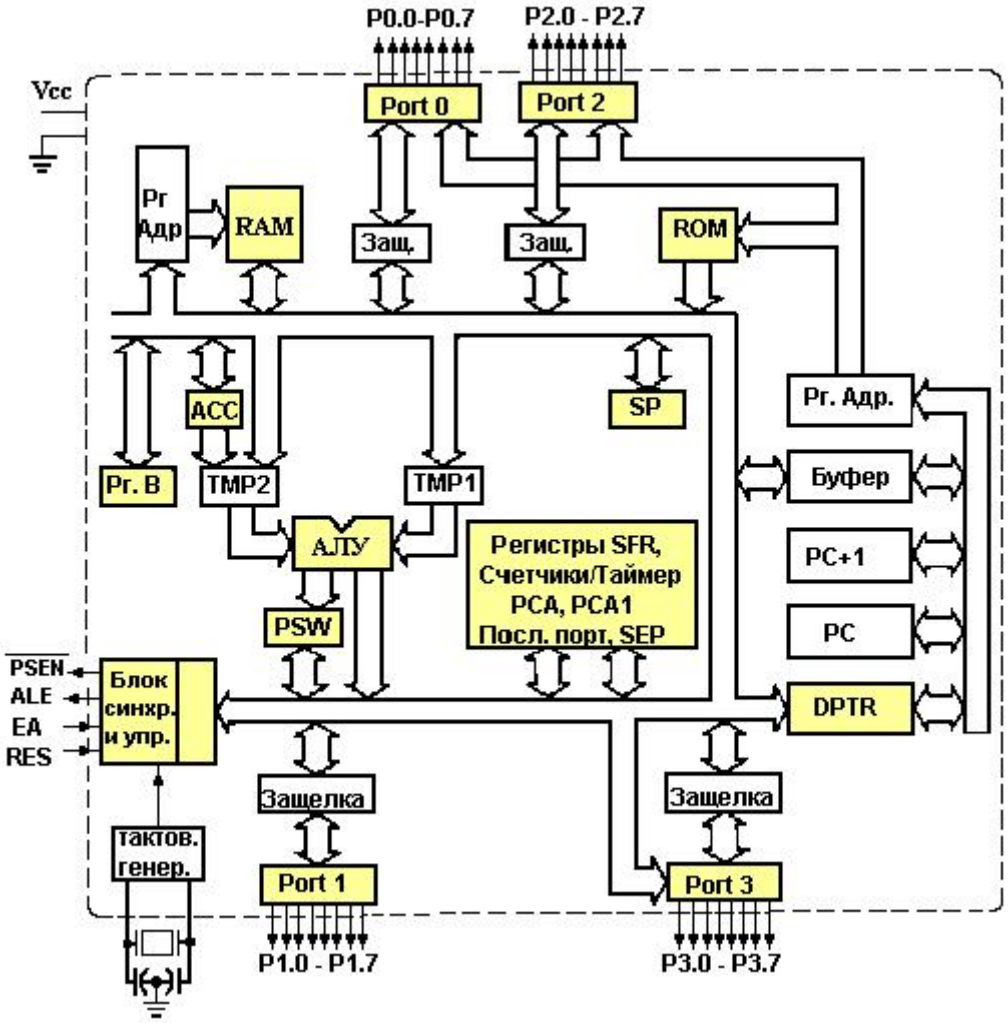


Рисунок 10 – Функциональная схема МК семейства MCS-51

В таком построении микроконтроллер оптимизирован для работы с внутренней памятью. Разработан по гарвардской архитектуре.

Программно доступные узлы микроконтроллера на рисунке 10 отмечены желтым цветом.

2 ОСОБЕННОСТИ РАЗРАБОТКИ И ФУНКЦИОНИРОВАНИЯ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ДЛЯ ВСТРАИВАЕМЫХ МИКРОПРОЦЕССОРНЫХ СИСТЕМ

2.1 Характерные черты микроконтроллеров

Основные характерные черты микроконтроллеров, которые предопределяют всю специфику разработки цифровых устройств на микроконтроллерах, разработки их системного и прикладного программного обеспечения и использования, следующие:

1) незначительная емкость адресуемой управляемой памяти относительно универсальных МП;
2) незначительный объем памяти, как правило, присутствует на самом кристалле в различных вариациях;
3) физическое и логическое разделение памяти программ (ПЗУ) и памяти данных (ОЗУ) – гарвардская архитектура;
4) упрощенная и ориентированная на задачи управления система команд, есть битовые операции;
5) простые методы адресации данных и команд (переходов);
6) специфическая организация ввода/вывода информации.

Мы живем а пространстве и во времени. Для МПС роль пространства отождествляется с **памятью**, ее количеством, а роль времени – с **быстродействием** АЛУ. В качестве примера базовая архитектура памяти семейства MCS-51 приведена на рисунке 11.

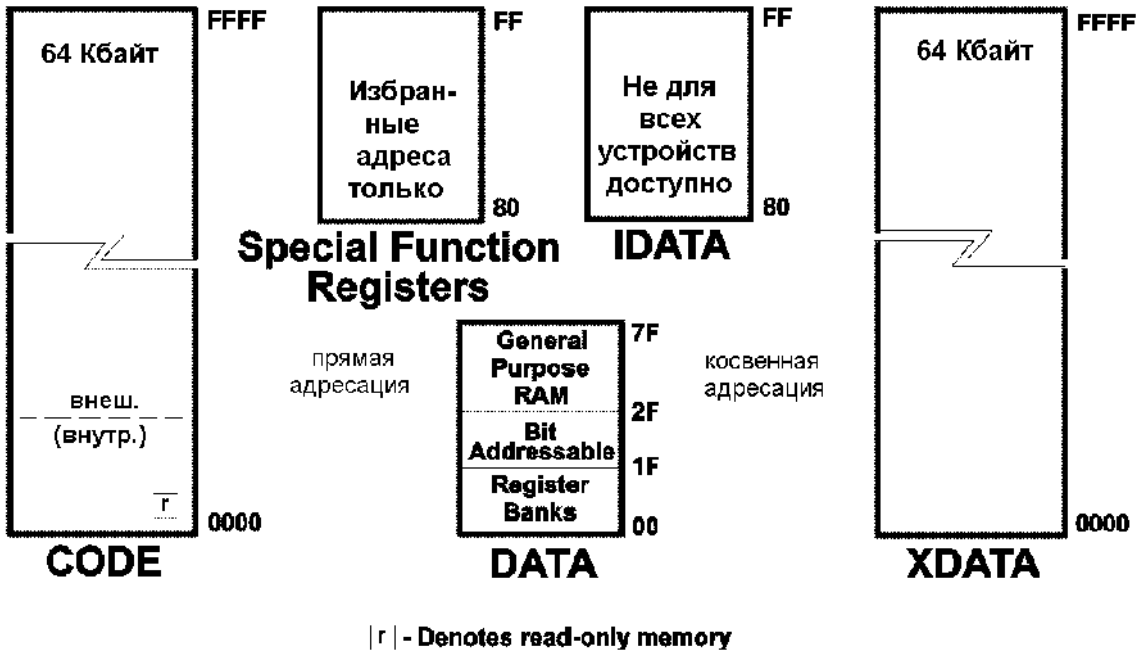


Рисунок 11 – Базовая структура памяти семейства MCS-51

Быстродействие микроконтроллера можно регулировать частотой тактового генератора МК.

Специфика МК отражается на декомпозиции алгоритма работы устройства, на работе Ассемблера и компиляторов, на работе компоновщика, на функционировании отладочных средств и т.д. Примечательная особенность MCS-51 – есть команды пересылки память-память без помощи регистров. Узкое место – реентерантные процедуры. Узкое место – сильно ограниченный стек, так как он организуется в одном из блоков данных. Самомодифицирующийся код программы возможен только с помощью дополнительной внешней логики. Реализуется различный доступ к данным одного формата, находящимся в разных типах памяти. И т.д.

Эти особенности предопределяют область использования микроконтроллеров как правило в качестве **специализированных вычислителей**, включенных в контур управления объектом или процессом.

Структурная организация, набор команд и аппаратурно-программные средства ввода/вывода информации микроконтроллеров лучше всего приспособлены для **решения задач управления и регулирования** в приборах, устройствах и системах автоматики, а не для решения задач обработки больших объёмов данных.

Микроконтроллеры не являются машинами классического "фон Неймановского" типа, так как физическая и логическая разделенность памяти программ и памяти данных исключает возможность модификации и/или замены (перегрузки) прикладных программ микроконтроллеров во время работы, что сильно затрудняет их использование в качестве универсальных средств обработки данных. Не отрабатывается принцип «хранимой программы», что существенно **затрудняет отладку**. В разной памяти одни и те же действия требуют разного количества машинных циклов на исполнение.

В устройствах управления объектами, иначе контроллерах, на основе МК аппаратурные средства и программное обеспечение существуют в форме **неделимого аппаратурно-программного комплекса**. При проектировании контроллеров приходится решать одну из самых сложных задач разработки, а именно задачу **оптимального распределения функций контроллера между аппаратурными средствами и программным обеспечением**. Решение этой задачи осложняется тем, что взаимосвязь и взаимовлияние аппаратурных средств и программного обеспечения в микропроцессорной технике претерпевают динамичные изменения.

Если в начале развития МП-техники определяющим было правило, в соответствии с которым аппаратурные средства обеспечивают производи-

тельность, а программное обеспечение – дешевизну изделия, то в настоящее время это правило нуждается в серьезной корректировке. Так как МК представляет собой стандартный массовый (относительно недорогой) логический блок, конкретное назначение которого определяет пользователь с помощью программного обеспечения, то с ростом степени интеграции и, следовательно, функционально-логических возможностей МК резко понижается стоимость изделия в пересчете на выполняемую функцию, что в конечном итоге и обеспечивает достижение высоких технико-экономических показателей изделий на МК. При этом затраты на разработку программного обеспечения изделия в 2–10 раз превышают (за время жизни изделия) затраты на приобретение и изготовление аппаратурных средств.

2.2 Языки программирования

Исторически первый и основной язык программирования МК – **Ассемблер**. С ростом размера программного кода программирование на языке Ассемблера усложняется. Поэтому для всех микроконтроллеров существуют компиляторы для языка **Си**. Язык Си – язык системного программирования, и он позволяет создавать произвольные и сложные иерархические структуры данных. Программу легче всего сверстать вначале на языке Си, а потом оптимизировать отдельные критические фрагменты на языке Ассемблера.

Для 32-разрядных архитектур используется язык C++.

Для программирования MCS-51 применяются также и другие языки высокого уровня. За долгую жизнь семейства MCS-51 встречались такие, как Форт (Forth), Бейсик, Паскаль, PL/M, Modula-2 и Python. Однако эти языки не получили такого широкого распространения, как Си и Ассемблер. Они выступают, скорее всего, как экзотика.

Применительно к семейству MCS-51 для программного обеспечения имеются существенные наработки, проверенные временем:

- Большой багаж накопленных знаний;
- Много наработанных программ;
- Много отлаженных библиотек;
- Много разработанных инструментальных средств.

2.3 Архитектура

В широком смысле слова под **архитектурой** МП (или МК) понимается его программная модель, т.е. все его программно-видимые свойства, которые доступны программисту. Иногда, для уточнения, используется термин программная архитектура. Под **микроархитектурой** понимается внутренняя реализация этой программной модели. Для одной архитек-

туры разными фирмами в разных поколениях применяются существенно различные микроархитектурные реализации. Клон MCS-51 насчитывает более 200 производных.

Специфика инструментальных средств для разработки прикладного программного обеспечения (ПО) для МК в том, что они поддерживают как правило несколько архитектур. Так инструментальные средства KEIL поддерживают 4 следующих архитектуры клона MCS-51: классическую структуру памяти 8051, расширения 8051 структуры памяти, универсальную структуру памяти 80C51MX процессоров Philips и структуру памяти 251 для Intel/Temic 251.

Компания Keil выпустила первый компилятор С, разработанный с нуля специально для микроконтроллера 8051. Компания Keil – Keil Elektronik GmbH в Германии и Keil Software, Inc. в Техасе. В 2005 году компания Arm Limited приобрела Keil

2.3.1 Архитектура памяти

Память программ (ПП). Программная память на кристалле – резидентная память программ (РПП) – и внешняя память программ в адресном пространстве до 64 Кбайт – **CODE** (C). Обычно только читаемая, используется для исполнительного кода и констант. В расширенных архитектурах до 16 Мбайт – ECODE и HCONST для констант в сегменте кода.

Память данных (ПД). Резидентная память данных (РПД) – внутренняя память на кристалле до 256 байт. Прямо адресуемая часть памяти – **DATA** (D). Часть памяти с доступом до бита – **BIT** (B). Косвенно адресуемая память через @R0 или @R1 – **IDATA** (I). У процессоров Philips и 251 эта область расширена до 64 Кбайт – EDATA. Внешняя память данных в адресном пространстве до 64 Кбайт – **XDATA** (X). Нулевая страница внешней памяти, косвенно адресуемая, размером в 256 байт – **PDATA**. В расширенных архитектурах память данных до 16 Мбайт – HDATA.

Сводка основных типов памяти данных и программ семейства MCS-51 представлена в таблице 1.

Таблица 1 – Типы памяти MCS-51

Тип памяти	Адресный диапазон	Описание
DATA	D:00 – D:7F	Прямо адресуемая память данных
BIT	D:20 – D:2F	Битовая память младших 128 бит
IDATA	I:00 – I:FF	Косвенно адресуемая память данных
XDATA	X:0000 – X:FFFF	Внешняя память данных
PDATA	X:00 – X:FF	0 страница внешней памяти
CODE	C:0000 – C:FFFF	Программная память
BANK 0 … BANK 31	B0:0000 – B0:FFFF B31:0000 – B31:FFFF	Дополнительные переключаемые банки памяти программ

Префиксы D, I, X, B, C, B0 … B31, приведенные в таблице 1, используются в отладчиках для задания адреса. Указанные типы памяти в языках программирования – это класс памяти, иначе тип хранения.

Структура памяти MCS-51, ставшая за долгие годы классической, приведена на рисунке 12.

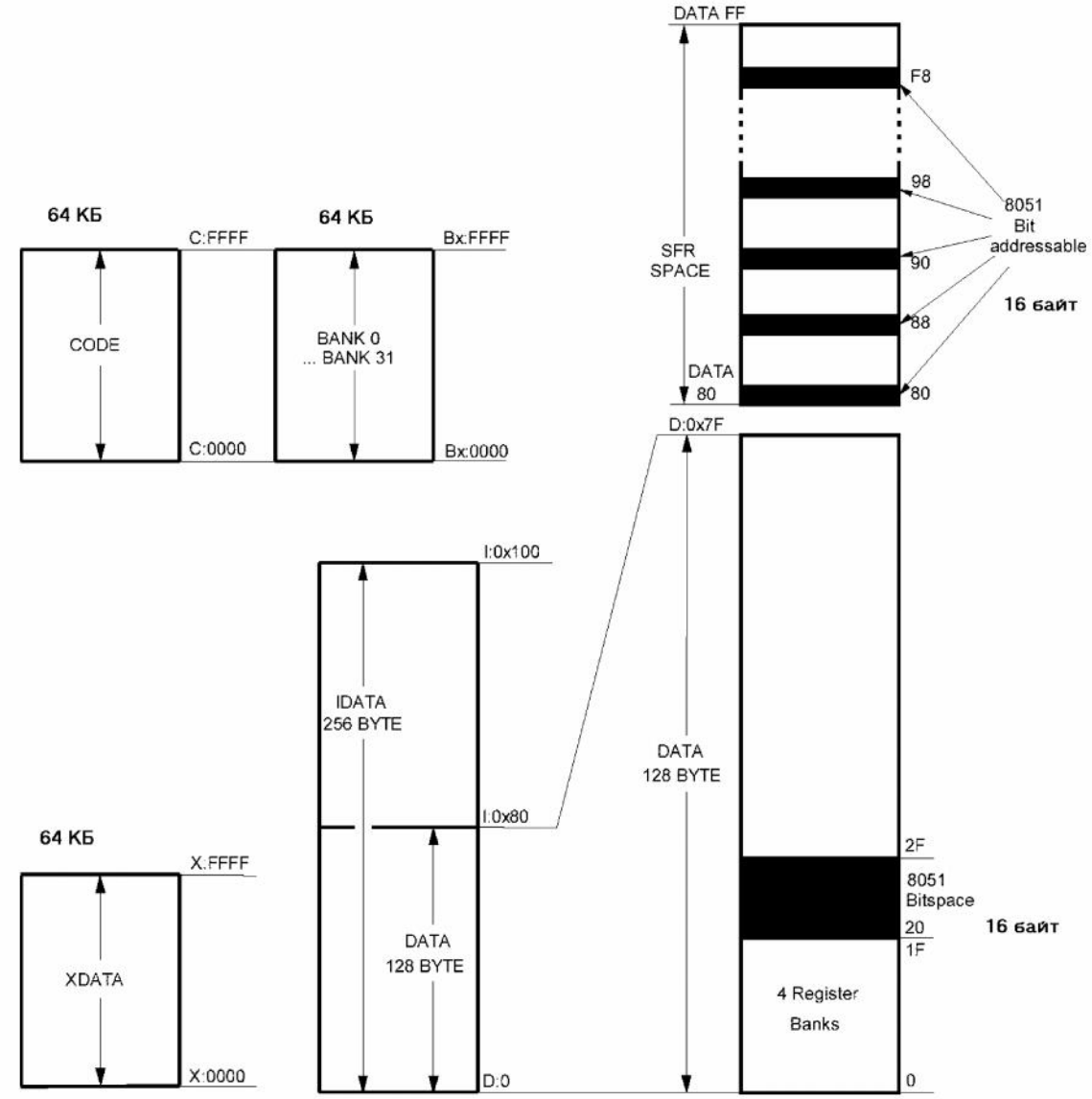


Рисунок 12 – Классическая структура памяти 8051

Организация программной памяти MCS-51 такова, что, если не хватает 64 Кбайт программной памяти, ее расширяют с помощью дополнительных переключаемых банков памяти максимум по 64 Кбайт. Так 32 таких банка дают дополнительно 2 Мбайт памяти программ. Подключение переключаемых банков требует дополнительной аппаратной поддержки.

Структура памяти расширения семейства MCS-51, MCS-251 и их производных приведена на рисунках 13 – 15.

16 Мбайт адресного пространства дает 24-разрядный адрес. Вводится дополнительный байт для адресации.

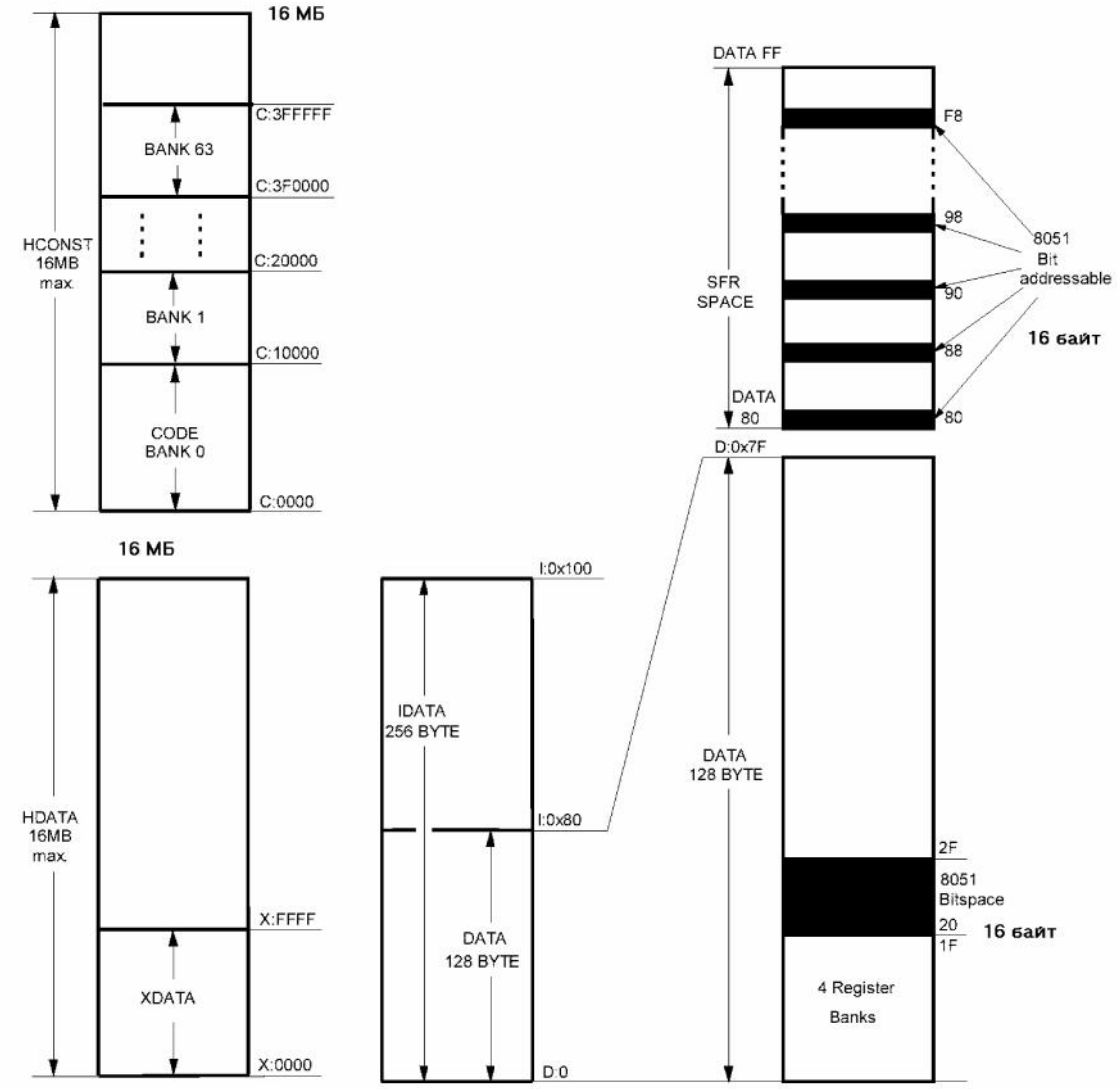


Рисунок 13 – Расширения структура памяти 8051

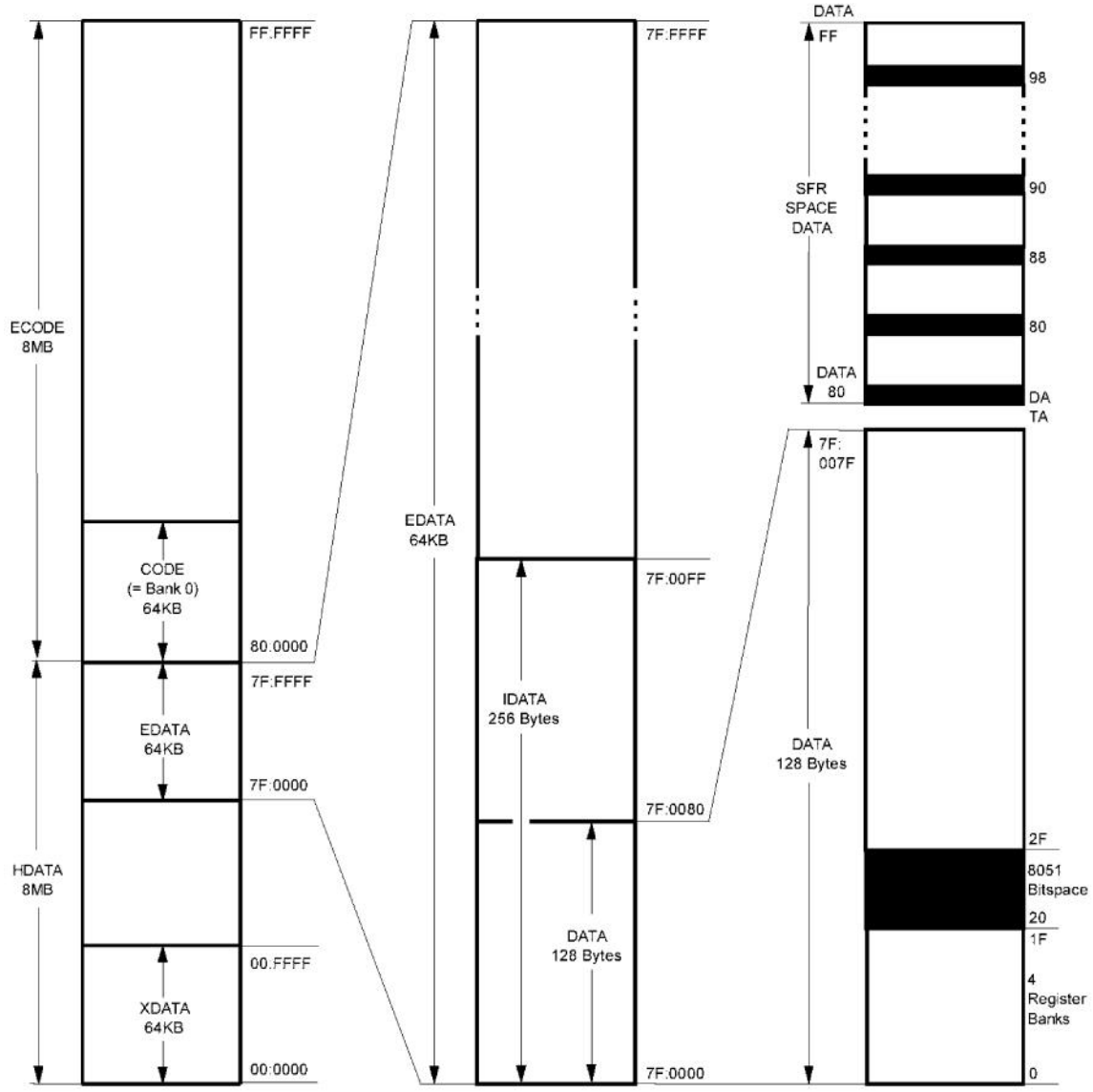


Рисунок 14 – Структура памяти 80C51MX

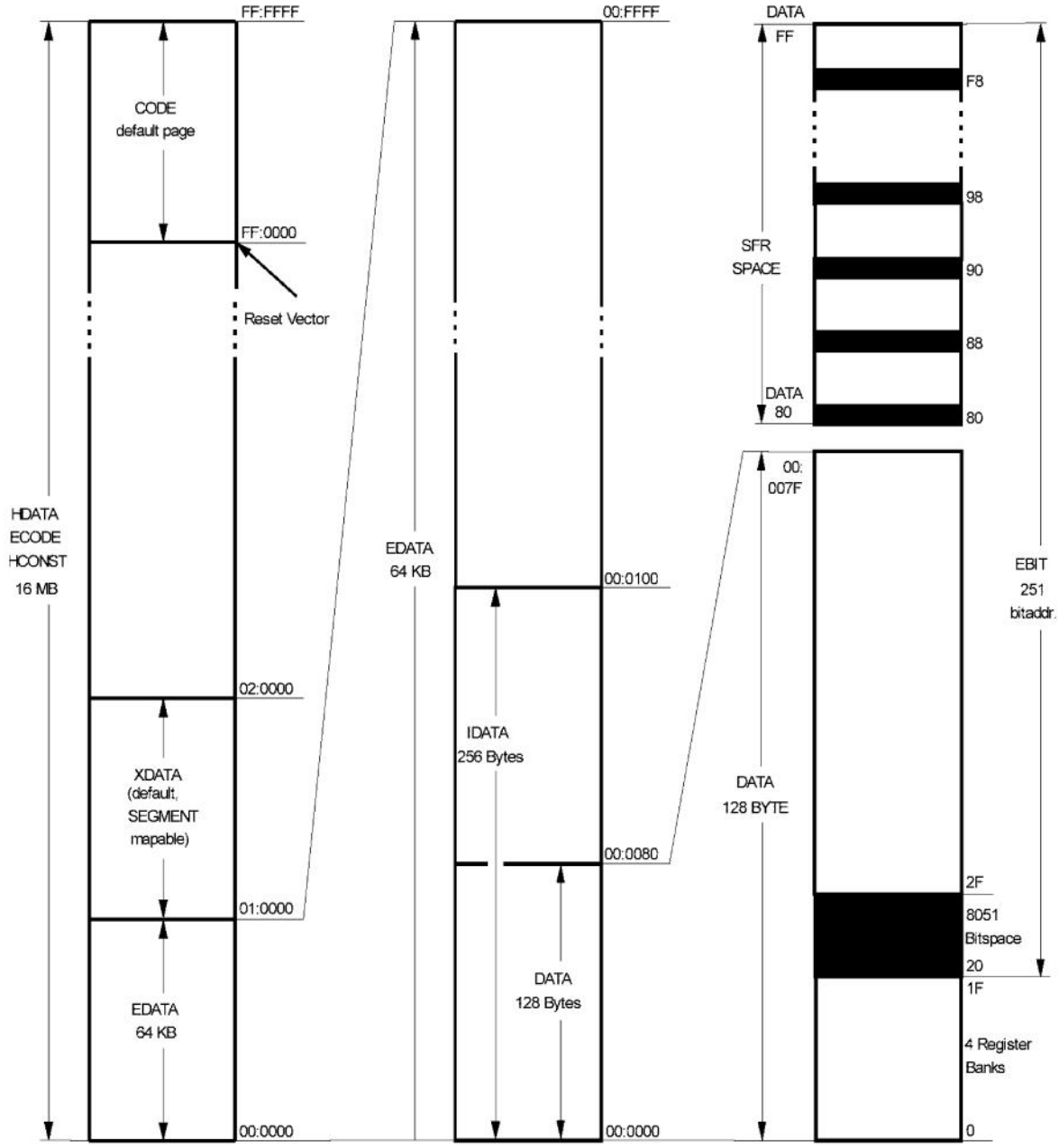


Рисунок 15 – Структура памяти MCS-251

2.3.2 Регистровая память

Регистровая память в МП по-другому называется сверхоперативным запоминающим устройством (СОЗУ).

В распоряжении пользователя имеются 8 **регистров общего назначения** (РОН) **R0–R7** одного из четырёх возможных банков. Банки регистров находятся в начале внутренней памяти данных и занимают 32 байта. 0-ой банк – адреса от 00h до 07h. 1-ый банк – адреса от 08h до 0Fh. 2-ой банк – адреса от 10h до 17h. 3-ий банк – адреса от 18h до 1Fh. Текущий рабочий

банк регистров задаётся в слове состояния программы (ССП) – по англ. PSW. Механизм переключения банков регистров упрощает обслуживание прерываний. Это своеобразная поддержка многозадачности.

2.3.3 Регистры специальных функций

Регистры специальных функций (РСФ) – по англ. SFR. Используются для управления всеми периферийными устройствами микроконтроллера. Доступ к этим регистрам осуществляется по прямой адресации.

Перечень РСФ семейства MCS-51 приведен в таблице 2.

Таблица 2 – Блок регистров специальных функций

Обозначение	Наименование	Адрес
* ACC	Аккумулятор	0E0H
* B	Регистр-расширитель аккумулятора	0F0H
* PSW	Слово состояния программы	0D0H
SP	Регистр-указатель стека	81H
DPTR	Регистр-указатель данных (DPH)	83H
	(DPL)	82H
* P0	Порт 0	80H
* P1	Порт 1	90H
* P2	Порт 2	0A0H
* P3	Порт 3	0B0H
* IP	Регистр приоритетов	0B8H
* IE	Регистр маски прерываний	0A8H
TM0D	Регистр режима таймера/счетчика	89H
* TC0N	Регистр управления/статуса таймера	88H
TH0	Таймер 0 (старший байт)	8CH
TL0	Таймер 0 (младший байт)	8AH
TH1	Таймер 1 (старший байт)	8DH
TL1	Таймер 1 (младший байт)	8BH
* SCON	Регистр управления приемопередатчиком	98H
SBUF	Буфер приемопередатчика	99H
PCON	Регистр управления мощностью	87H

Регистры с символическими именами IP, IE, TMOD, TCON, SCON и PCON и т.п. используются для фиксации и программного изменения управляющих бит и бит состояния схемы прерывания, таймера/счетчика, приемопередатчика последовательного порта и для управления мощностью электропитания у представителей семейства MCS-51.

Регистры, имена которых отмечены знаком (*) в таблице 2, допускают адресацию отдельных бит. Все их адреса кратны 8. Младшие три бита адреса дают адрес бита в текущем байте. Это старшая группа битовой памяти в диапазоне 128 бит.

МК MCS-51 называют битовый процессор.

2.3.4 Машинные инструкции семейства MCS-51

Система команд содержит 111 базовых команд 8-разрядного микроконтроллера, число которых увеличивается за счёт различных методов адресации до полного набора 255. Это классический набор инструкций, и он выполняется для всех представителей семейства. Производные МК семейства MCS-51 и Philips расширяют свою функциональность с помощью кода операции **A5h**.

Команды имеют длину от одного до трех байт. По формату все команды сгруппированы в 13 типов команд. Первый байт команды всегда содержит код операции (КОП). Второй и третий байты команды, когда они присутствуют, задают либо адреса операндов, либо их непосредственные значения.

Когда микроконтроллер выполняет сугубо вычислительную часть задачи, то на собственно вычисления в лучшем случае тратится порядка 30% времени. Остальное время занимает подготовка данных, а именно операции пересылки операндов. Поэтому команды пересылки данных – самые быстрые в системе команд.

В зависимости от способа адресации и места расположения операнда выделяется девять типов операндов, между которыми происходит информационный обмен. Схема возможных путей пересылки данных показана на рисунке 16. Отличительная особенность – пересылки данных могут выполняться без участия аккумулятора. В системе команд есть даже пересылка память память.

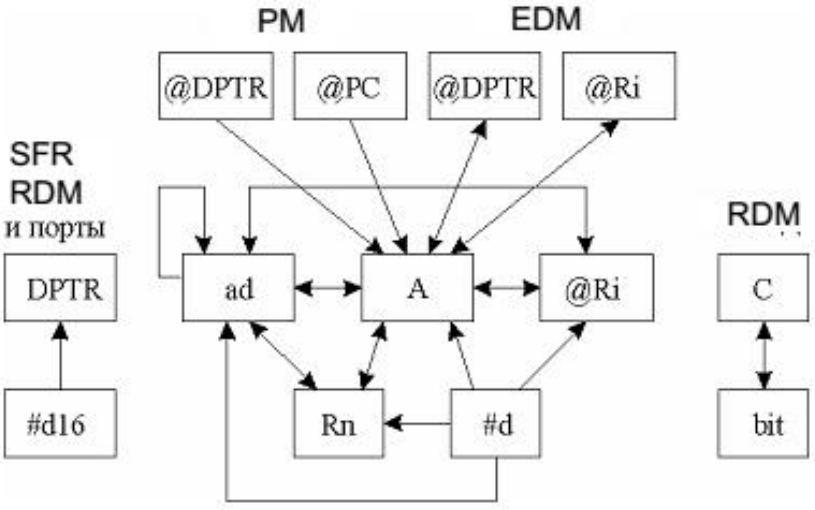


Рисунок 16 – Схема операций пересылки данных

Список всех команд для семейства MCS-51 приведен в таблице 3.

В таблице 3 команды обозначены кодом операции (КОП). Второй и третий байты, если нужны, определяются операндами команды.

Таблица 3 – КОП и команды семейства MCS-51

Binary Mode	**x0**	**x1**	**x2**	**x3**	**x4**	**x5**	**x6 – x7**	**x8 – xF**
Source Mode	**x0**	**x1**	**x2**	**x3**	**x4**	**x5**	**A5x6- A5x7**	**A5x8- A5xF**
0x	NOP	AJMP adr11	LJMP adr16	RR A	INC A	INC dir	INC @Ri	INC Rn
1x	JBC bit,rel	ACALL adr11	LCALL adr16	RRC A	DEC A	DEC dir	DEC @Ri	DEC Rn
2x	JB bit,rel	AJMP adr11	RET	RL A	ADD A,#data	ADD A,dir	ADD A,@Ri	ADD A,Rn
3x	JNB bit,rel	ACALL adr11	RETI	RLC A	ADDC A,#data	ADDC A,dir	ADDC A,@Ri	ADDC A,Rn
4x	JC rel	AJMP adr11	ORL dir,A	ORL dir,#data	ORL A,#data	ORL A,dir	ORL A,@Ri	ORL A,Rn
5x	JNC rel	ACALL adr11	ANL dir,A	ANL dir,#data	ANL A,#data	ANL A,dir	ANL A,@Ri	ANL A,Rn
6x	JZ rel	AJMP adr11	XRL dir,A	XRL dir,#data	XRL A,#data	XRL A,dir	XRL A,@Ri	XRL A,Rn
7x	JNZ rel	ACALL adr11	ORL c,bit	JMP @A+DPTR	MOV A,#data	MOV dir,#data	MOV @Ri,#data	MOV Rn,#data
8x	SJMP rel	AJMP adr11	ANL C,bit	MOVC A,@A+PC	DIV AB	MOV dir,dir	MOV dir,@Ri	MOV dir,Rn
9x	MOV DPTR,#d16	ACALL adr11	MOV bit,c	MOVC A,@A+DPTR	SUBB A,#data	SUBB A,dir	SUBB A,@Ri	SUBB A,Rn
Ax	ORL C,/bit	AJMP adr11	MOV C,bit	INC DPTR	MUL AB	OP-CODE PREFIX	MOV @Ri,dir	MOV Rn,dir
Bx	ANL C,/bit	ACALL adr11	CPL bit	CPL C	CJNE A,#d8,rel	CJNE A,dir,rel	CJNE @Ri,#d8,rel	CJNE Rn,#d8,rel
Cx	PUSH dir	AJMP adr11	CLR bit	CLR C	SWAP A	XCH A,dir	XCH A,@Ri	XCH A,Rn
Dx	POP dir	ACALL adr11	SETB bit	SETB C	DA A	DJNZ dir,rel	XCHD A,@Ri	DJNZ Rn,rel
Ex	MOVX A,@DPTR	AJMP adr11	MOVX A,@Ri		CLR A	MOV A,dir	MOV A,@Ri	MOV A,Rn
Fx	MOV @DPTR,A	ACALL adr11	MOVX @Ri,A		CPL A	MOV dir,A	MOV @Ri,A	MOV Rn,A

Обозначения, используемые при описании команд:

- Rn (n = 0, 1,..., 7) – регистр общего назначения в выбранном банке
- регистров;
- @Ri(i= 0, 1) – регистр общего назначения в выбранном банке регистров, используемый в качестве регистра косвенного адреса;
- ad – адрес прямо адресуемого байта;
- ads – адрес прямо адресуемого байта-источника;
- add – адрес прямо адресуемого байта-получателя;

- ad11 – 11-разрядный абсолютный адрес перехода;
- ad16 – 16-разрядный абсолютный адрес перехода;
- rel – относительный адрес перехода;
- #d – непосредственный операнд;
- #d16 – непосредственный операнд(2 байта);
- bit – адрес прямо адресуемого бита;
- /bit – инверсия прямо адресуемого бита;
- A – аккумулятор;
- PC – счетчик команд;
- DPTR – регистр указатель данных;
- ( ) – содержимое ячейки памяти или регистра.

Расширения семейства MCS-51 являются надмножествами указанного семейства. Присутствует совместимость вверх на уровне кодов. Исходный текст компилируется любым компилятором. Высший представитель исполняет программу более производительнее, чем исходный 8051.

MCS-251 работает в двух режимах: **бинарный режим** 8051 и родной **исходный режим**. В бинарном режиме дополнительные инструкции идут с префиксом A5h. В исходном режиме действует расширенный набор инструкций, а базовый набор инструкций 8051 используется с префиксом A5h. Какой набор инструкций, базовый или расширенный, в программе преобладает, тот режим лучше и использовать для сокращения кода программы. Расширенный набор инструкций представлен в таблице 4.

Таблица 4 – Расширенные инструкции семейства MCS-251

Binary Mode	**A5x8**	**A5x9**	**A5xA**	**A5xB**	**A5xC**	**A5xD**	**A5xE**	**A5xF**
Source Mode	**x8**	**x9**	**xA**	**xB**	**xC**	**xD**	**xE**	**xF**
0	JSLE rel	MOV Rm, @WRj+dis	MOVZ WRj,Rm	INC Rm/WRj/ Drk,#short MOV reg,ind			SRA reg	
1	JSG rel	MOV @WRj+dis,Rm	MOVS WRj,Rm	DEC Rm/WRj/ Drk,#short MOV ind,reg			SRL reg	
2	JLE rel	MOV Rm,@DRk+dis			ADD Rm,Rm	ADD WRj,WRj	ADD reg,op2	ADD DRk,DRk
3	JG rel	MOV @DRk+dis,Rm					SLL reg	
4	JSL rel	MOV WRj,@WRj+dis			ORL Rm,Rm	ORL WRj,WRj	ORL reg,op2	
5	JSGE rel	MOV @WRj+dis,WRj			ANL Rm,Rm	ANL WRj,WRj	ANL reg,op2	
6	JE rel	MOV WRj,@DRk+dis			XRL Rm,Rm	XRL WRj,WRj	XRL reg,op2	
7	JNE rel	MOV @Drk+dis,WRj	MOV op1,reg		MOV Rm,Rm	MOV WRj,WRj	MOV reg,op2	MOV DRk,DRk
8		LJMP @WRj EJMP @DRk	EJMP addr24		DIV Rm,Rm	DIV WRj,WRj		
9		LCALL @WR ECALL @DRk	ECALL addr24		SUB Rm,Rm	SUB WRj,WRj	SUB reg,op2	SUB DRk,DRk
A		BIT instructions	ERET		MUL Rm,Rm	MUL WRj,WRj		
B		TRAP			CMP Rm,Rm	CMP WRj,WRj	CMP reg,op2	CMP DRk,DRk
C			PUSH op1					
D			POP op1					
E								
F								

Дополнительные инструкции Philips 80C51MX. Расширенные команды вызываются через префикс A5. Это расширение включает набор инструкций для адресации памяти в пределах единого 16 Мбайт адресного пространства CODE и XDATA и для доступа к дополнительным SFR регистрам. Указатель стека 16-разрядный для адресации данных в памяти EDATA размером до 64 Кбайт. Указатель PR0 составлен из регистров R1, R2 и R3, а указатель PR1 – из регистров R5, R6 и R7. Список расширения приведен в таблице 5.

Таблица 5 – Расширенные инструкции Philips 80C51MX

	A5 x0	**A5 x1**	**A5 x2**	**A5 x3**	**A5 x4**	**A5 x5**	**A5 x6 – A5 x7**	**A5 x8 – A5 xF**	**A5 x8 – A5 xF**
A5 0x			EJMP adr23			INC esfr			
A5 1x	JBC es-bit,rel		ECALL adr23			DEC esfr			
A5 2x	JB esbit,rel					ADD A,esfr			
A5 3x	JNB es-bit,rel					ADDC A,esfr			
A5 4x			ORL esfr,A	ORL esfr,#data		ORL A,esfr		EMOV A,@PR0+d2	EMOV A,@PR1+d2
A5 5x			ANL esfr,A	ANL esfr,#data		ANL A,esfr		EMOV @PR0+d2,A	EMOV @PR1+d2,A
A5 6x			XRL esfr,A	XRL esfr,#data		XRL A,esfr		ADD PR0,#data2	ADD PR1,#data2
A5 7x			ORL c,esbit	EJMP @A+EPTR		MOV dir,#data			
A5 8x			ANL C,esbit			MOV esfr,esfr	MOV esfr,@Ri	MOV esfr,Rn	
A5 9x	MOV EPTR,#d23		MOV esbit,c	MOVC A,@A+EPTR		SUBB A,esfr			
A5 Ax	ORL C,/es-bit		MOV C,esbit	INC EPTR			MOV @Ri,esfr	MOV Rn,esfr	
A5 Bx	ANL C,/es-bit		CPL es-bit			CJNE A,esfr,rel			
A5 Cx	PUSH esfr		CLR es-bit			XCH A,esfr			
A5 Dx	POP esfr		SETB esbit			DJNZ esfr,rel			
A5 Ex	MOVX A,@EPTR					MOV A,esfr			
A5 Fx	MOV @EPTR,A					MOV esfr,A			

Архитектура семейства **16-разрядных** микроконтроллеров **Philips XA** в системе команд не содержит подмножество семейства 80C51. Программы совместимы на уровне исходного текста ассемблера. Точнее, требуется трансляция исходного текста программы систем 80C51 в исходный код 80C51XA.

3 ЭТАПЫ ПРОЕКТИРОВАНИЯ МИКРОПРОЦЕССОРНОГО КОНТРОЛЛЕРА

МК-система управления. Типовая структура МК-системы управления состоит из:

1. объекта управления,

2. микроконтроллера,
3. аппаратуры их взаимной связи.

Средства взаимной связи включают систему сбора информации и набор исполнительных устройств. В количественном измерении аппаратных затрат эти компоненты могут быть представлены в различных соотношениях.

Микроконтроллер путем периодического опроса через порты ввода осведомительных слов (ОС) генерирует в соответствии с алгоритмом управления последовательности управляющих слов (УС) через порты вывода. Осведомительные слова это сигналы состояния объекта (СС), сформированные датчиками объекта управления, и флаги. Выходные сигналы датчиков вследствие их различной физической природы могут потребовать промежуточного преобразования на аналого-цифровых преобразователях (АЦП) или на схемах формирователей сигналов (ФС), которые чаще всего выполняют функции гальванической развязки и формирования уровней двоичных сигналов стандарта ТТЛ.

Микроконтроллер с требуемой периодичностью обновляет управляющие слова на своих выходных портах. Некоторая часть управляющего слова интерпретируется как совокупность **прямых двоичных сигналов управления** (СУ), которые через схемы формирователей сигналов (усилители мощности, реле, оптроны и т.п.) поступают на исполнительные механизмы (ИМ) и устройства индикации. Другая часть управляющего слова представляет собой **упакованные двоичные коды**, которые через цифро-аналоговые преобразователи (ЦАП) воздействуют на исполнительные механизмы аналогового типа. Если объект управления использует цифровые датчики и цифровые исполнительные механизмы, то наличие ЦАП и АЦП в системе необязательно.

Законы функционирования типовой системы управления всецело определяются прикладной программой, размещаемой в резидентной памяти программ МК. Иными словами **специализация контроллера** типовой структуры на решение задачи управления конкретным объектом осуществляется путем разработки 1) прикладных программ МК и 2) аппаратуры связи МК с датчиками и исполнительными механизмами объекта.

3.1 цикл разработки

Весь цикл разработки контроллеров рассматривается как последовательность **трех фаз проектирования**. Фазы проектирования могут проводится итерационно по мере возникновения уточнений. Эти фазы:

1) Анализ задачи и выбор (и/или разработка) комплекта аппаратных средств контроллера;

2) Разработки системного и прикладного программного обеспечения;
3) Объединение всех аппаратных средств и программного обеспечения в макетном образце контроллера и его отладка, проведенная автономно и комплексно.

Аппаратные средства. МК представляет собой логический автомат с высокой степенью детерминированности, который допускает очень немного вариантов его системного включения. Типовой состав аппаратурных средств ядра любой МК-системы включает МК, ППЗУ, ОЗУ, интерфейсные БИС, схемы синхронизации и системного управления. Два пути: **[1]** строить котроллер из указанных компонентов изначально и самостоятельно; **[2]** типовой состав аппаратуры оформляется конструктивно в виде **одноплатных универсальных программируемых контроллеров**, которые предназначены для встраивания в контур управления объектом или процессом. 0) Такой контроллер как БИС можно установить на своей плате. 1) На печатной плате такой МК-системы могут быть представлены гнезда для установки БИС пользователя. 2) На некоторых моделях таких плат имеется еще и так называемое "монтажное поле пользователя", на котором он имеет возможность монтировать свои специфические схемы, такие как оптронные развязки, АЦП, ЦАП, реле и т.п. 3) Кроме того, на плате МК-системы может быть размещен источник электропитания. 4) На плате такой МК-системы могут быть реализованы интерфейсы аппаратуры связи.

Появление однокристальных МК иллюстрирует тот факт, что все более сложные функционально насыщенные части аппаратурных средств контроллеров в процессе интеграции переходят из разряда подсистем в разряд **комплектующих изделий**. Так как эти комплектующие изделия являются сложно организованными приборами, функционирующими под управлением программ, то удельный вес прикладного программного обеспечения МК-систем имеет устойчивую тенденцию к увеличению, а удельный вес аппаратурных средств – к снижению.

3.2 Разработка прикладного программного обеспечения

Ориентация на разработку прикладных программ для МК-систем силами программирующих профессионалов-разработчиков (а не профессиональных программистов) получает распространение потому, что в условиях быстро дешевеющей памяти изменились стиль и технология разработки программ. Экономят теперь уже не память МК-системы, а время разработчика программного обеспечения, т.е. сокращают сроки разработки изделия.

Фаза разработки программного обеспечения. Это – фаза получения прикладных программ в машинном коде. В свою очередь, эта фаза разбивается на два существенно различных этапа, которые могут проводиться итерационно:

1) "от постановки задачи к исходной программе";

2) "от исходной программы к исполнительному коду".

Проведение нескольких итераций этих этапов друг за другом скорее приводит к успеху.

3.2.1 Второй этап разработки

Второй этап разработки – самый лёгкий. Этап "от исходной программы к исполнительному коду" представлен на рисунке 17 и имеет целью получение машинных кодов прикладных программ, работающих в МК. Этот этап разработки прикладного программного обеспечения легко поддается формализации и поддержан всей мощью системного программного обеспечения МК, направленного на автоматизацию процесса получения прикладных программ. В состав средств системного программного обеспечения входят трансляторы с различных алгоритмических языков высокого уровня, ассемблеры, редакторы текстов, программы-отладчики, программы-документаторы и т.д. Наличие всех этих системных средств придает инженерной работе на этом этапе проектирования контроллеров характер ремесла, а не инженерного творчества. Так как в конечном изделии (контроллере) имеются только "голый" МК и средства его сопряжения с объектом, то выполнять отладку разрабатываемого прикладного программного обеспечения на нем невозможно (из-за отсутствия средств ввода, вывода, ОЗУ большой емкости и операционной системы), и, следовательно, разработчик вынужден обращаться к средствам вычислительной техники для выполнения всех формализуемых стадий разработки: трансляции, редактирования, отладки, загрузки объектных кодов в программируемую постоянную память МК. Попутно отметим, что системные средства автоматизации разработки прикладных программ МК на этапе "от исходной программы к исполнительному коду" широко распространены и существуют как в среде операционных систем ряда специализированных микро-ЭВМ, так и присутствуют в операционных системах Windows персональных компьютеров. Производители HI-TECH Software, IAR Systems, Keil Elektronik GmbH, TASKING, RAISONANCE S. A. и др.

3.2.2 Первый этап разработки

Первый этап разработки. Совсем по-другому выглядит инженерный труд на этапе разработки программного обеспечения "от постановки задачи к исходной программе", так как он практически не поддается формализации и, следовательно, не может быть автоматизирован. Проектная работа здесь носит творческий характер, изобилует решениями, имеющими "волевую" или "вкусовую" окраску, и решениями, продиктованными конъюнктурными соображениями. В силу перечисленных обстоятельств именно на этапе проектирования "от постановки задачи к исходной программе" разработчик сталкивается с наибольшим количеством трудностей. Качество получаемого прикладного программного обеспечения контроллера всецело зависит от уровня проектных решений, принятых на этапе разработки "от постановки задачи к исходной программе". Уровень проектных решений в свою очередь из-за отсутствия теории проектирования программируемых контроллеров определяется только опытом, квалификацией и интуицией разработчика. Систематический подход к процессу разработки прикладных программ для контроллеров обеспечивает достижение хороших результатов даже начинающими разработчиками.

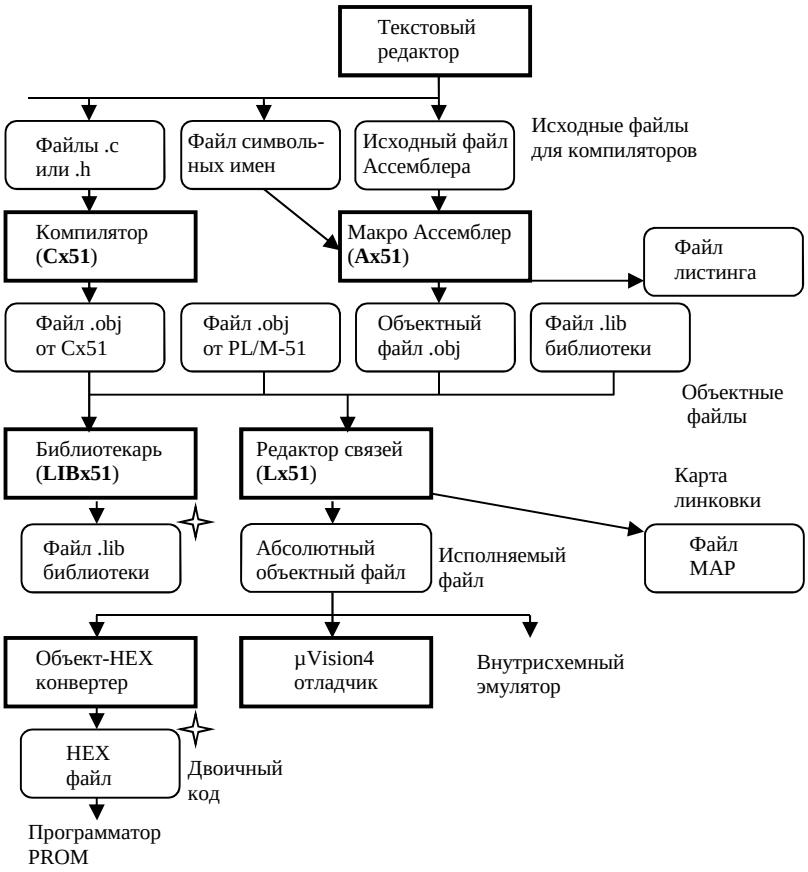


Рисунок 17 – Этап получения исполнительного кода МК

4. ТЕХНОЛОГИИ РАЗРАБОТКИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ МИКРОПРОЦЕССОРНЫХ СИСТЕМ

4.1. Методика

Если задача на разработку прикладной программы для микроконтроллера поставлена, то для получения текста исходной программы необходимо выполнить ряд последовательных действий.

1. Подробно описать задачу.
2. Проанализировать задачу.
3. Выполнить инженерную интерпретацию задачи, желательно с привлечением того или иного аппарата формализации (граф автомата, сети Петри, матрицы состояний и связности и т.п.).
4. Разработать общую схему алгоритма работы контроллера.
5. Разработать детализированные схемы отдельных процедур, выделенных на основе модульного принципа составления программ.
6. Детально проработать интерфейс контроллера и внести исправления в общую и детализированные схемы алгоритмов.
7. Распределить рабочие регистры и память.
8. Сформировать текст исходной программы.

В результате работы по трем первым пунктам данного перечня получают так называемую **функциональную спецификацию прикладной программы**, в которой основное внимание уделяется детализации способов формирования входной и выходной информации.

На языке схем алгоритмов разработчик описывает метод, выбранный им для решения поставленной задачи. Довольно часто бывает, что одна и та же задача может быть решена различными методами. Способ решения задачи, выбранный на этапе её инженерной интерпретации, на основе которого формируется схема, определяет не только качество разрабатываемой прикладной программы, но и качественные показатели конечного изделия.

Алгоритм есть точно определенная процедура, предписывающая контроллеру однозначно определенные действия по преобразованию исходных данных в обработанные выходные данные. Поэтому разработка схемы требует предельной точности и однозначности используемой атрибутики: символических имен переменных, констант, подпрограмм (модулей), символических адресов таблиц, портов ввода вывода и т.п. Основное

внимание при разработке следует уделить тому разделу функциональной спецификации прикладной программы, в котором приводится описание аппаратуры сопряжения с объектом управления. Это описание должно быть детализировано вплоть до электрических и временных характеристик каждого входного и выходного сигнала или устройства.

Успех разработки прикладной программы заключается в использовании метода декомпозиции, при котором вся задача последовательно разделяется на меньшие функциональные модули. Каждый из модулей можно анализировать, разрабатывать и отлаживать отдельно от других. При выполнении прикладной программы в микроконтроллере управление без всяких двусмысленностей передается от одного функционального модуля к другому. Схема связности этих функциональных модулей, каждый из которых реализует некоторую процедуру, образует общую схему алгоритма прикладной программы. Язык графических образов схемы алгоритма можно использовать на любом уровне детализации описания модулей вплоть до того, что каждому оператору схемы будет соответствовать единственная команда микроконтроллера.

Структурное программирование есть процесс построения прикладной программы из набора программных модулей, каждый из которых реализует определенную процедуру обработки данных. Программные модули должны иметь только одну точку входа и одну точку выхода. Только в этом случае отдельные модули можно разрабатывать и отлаживать независимо, а затем объединять в законченную прикладную программу с минимальными проблемами их взаимосвязи.

Источником подавляющего большинства ошибок программирования является использование модулей, имеющих один вход и несколько выходов. При необходимости организации множественных ветвлений в программе декомпозицию задачи выполняют таким образом, чтобы каждый функциональный модуль имел только один вход и один выход. Для этого условные операторы (имеющие два выхода) или включают внутрь модуля, объединяя их с операторами обработки, или выносят в систему межмодульных связей, формируя тем самым схему алгоритма более высокого ранга.

В международном стандарте на программный продукт **HIPO** (Hierarchy-Input-Process-Output) (“хай-по”) декларируется аналогичный подход к разработке прикладных программ.

Разработка схемы алгоритма функционального модуля программы имеет **ярко выраженный итеративный характер**, т.е. требует многократных проб, прежде чем возникает уверенность, что алгоритм реализа-

ции процедуры правильный и завершённый. Вне зависимости от функционального назначения процедуры при разработке её схемы необходимо придерживаться следующей очередности работы.

1. Определить, что должен делать модуль.
2. Определить способы получения модулем исходных данных (от датчиков через порты ввода, из таблиц в памяти или через рабочие регистры).
3. Определить необходимость какой-либо предварительной обработки введенных исходных данных (маскирование, сдвиг, масштабирование, перекодировка).
4. Определить метод преобразования входных данных в требуемые выходные. Используя операторы процедур и условные операторы принятия решения, отобразить на языке схемы алгоритма выбранный метод.
5. Определить способы выдачи из модуля обработанных данных (передать в память, в вызывавшую программу или в порты вывода).
6. Определить необходимость какой-либо вторичной обработки выводимых данных (изменение формата, перекодирование, масштабирование, маскирование).
7. Вернуться к пункту 1 настоящего перечня и проанализировать полученный результат. Выполнить итеративную корректировку схемы алгоритма с целью сделать её простой, логичной, стройной и обладающей чётким графическим образом.
8. Проверить работоспособность алгоритма на бумаге путем подстановки в него действительных данных. Убедиться в его сходимости и результативности.
9. Рассмотреть предельные случаи и попытаться определить граничные значения информационных объектов, с которыми оперирует алгоритм, за пределами которых он теряет свойства конечности, сходимости или результативности. Особое внимание при этом следует уделить анализу возможных ситуаций переполнения разрядной сетки, изменения знака результата операции, деления на переменную, которая может принять нулевое значение.
10. Провести мысленный эксперимент по определению работоспособности алгоритма в реальном масштабе времени, когда стохастические события, происходящие в объекте управления, могут оказать влияние на работу алгоритма. При этом самому тщательному анализу следует подвергнуть реакцию алгоритма на возможные прерывания с целью определения критических операторов, которые необходимо защитить от прерываний. Кроме того, в ходе этого

мысленного эксперимента следует проанализировать логику алгоритма с целью определения таких последовательностей операторов, при выполнении которых микроконтроллер может “не заметить” кратковременных событий в объекте управления. При обнаружении таких ситуаций в логику следует внести коррективы.

Практика разработки программного обеспечения показала, что последовательное использование описанной поэтапной процедуры, составляющей основу метода структурного программирования, позволяет уверенно получать работоспособные прикладные программы.

Преобразование разработанной схемы алгоритма в исходный текст программы дело несложное. Но прежде чем приступить к написанию программы необходимо специфицировать память и выбрать язык программирования.

Спецификация памяти и рабочих регистров заключается в определении адреса первой команды прикладной программы, действительных начальных адресов стека, таблиц данных, буферных зон передачи параметров между процедурами, подпрограмм обслуживания прерываний и т.д. При этом следует помнить, что в микроконтроллерах память программ и память данных физически и логически разделены.

Диапазон **языков** написания исходного текста прикладной программы простирается от машинного кода до почти естественного языка. В машинном коде или на языке ассемблера программировать труднее, чем на алгоритмическом языке высокого уровня, но зато получается более короткий код программы, требуется меньшая ёмкость памяти программы и выполняется такая программа быстрее.

Объектные коды, полученные путем трансляции исходных программ, написанных на алгоритмическом языке высокого уровня, занимают в памяти больше места и требуют большего времени на исполнение. Выбор языковых средств составления исходных программ в каждом конкретном случае зависит от характеристик прикладной задачи, опыта программиста и допустимых затрат на разработку.

Сверстать программу на языке высокого уровня значительно проще. Получив работающий вариант программы, его можно уже оптимизировать на ассемблере. (5-ый признак сложной системы).

Огромное большинство прикладных задач управления объектами вследствие того, что они должны решаться в реальном времени, предъявляет столь высокие требования по быстродействию, что для их решения основным языковым средством написания прикладных программ еще долгие годы будет оставаться язык ассемблера. Это положение о преиму-

щественном использовании языка ассемблера подкрепляется и тем обстоятельством, что однокристальные микроконтроллеры имеют ограниченный объем резидентной памяти программ и, следовательно, критичны к длине прикладных программ.

4.2. Процедуры и подпрограммы

Общее представление.

При разработке микроконтроллерных систем могут быть использованы два способа организации прикладных программ: **монолитный** и **модульный**. При первом способе вся прикладная программа разрабатывается как единое целое. Пригоден для небольших программ. При втором она строится из отдельных программных блоков, каждый из которых реализует некоторую процедуру обработки данных или управления. Взаимосвязь блоков определяется разработчиком при монтаже из этих блоков законченной прикладной программы.

Отдельные фрагменты прикладной программы могут быть получены в виде линейной последовательности блоков, другие (многократно используемые) обычно оформляются в виде **подпрограмм**, к которым прикладная программа, называемая основной, имеет возможность обратиться по мере необходимости. Подпрограмма должна обладать следующими свойствами: 1) выполнять законченную процедуру обработки данных, 2) иметь только один вход и один выход и 3) не обладать эффектом последействия, при котором текущее выполнение подпрограммы оказывало бы влияние на её последующие выполнения.

Вызов подпрограммы. Обращение к подпрограмме осуществляется по команде вызова **CALL MARK**, где MARK – символическое имя процедуры. Имя процедуры используется в качестве метки, отмечающей одну из команд (чаще всего первую) подпрограммы. Для Intel 8051 мнемоническое значение CALL является обобщенным и транслируется в одну из команд **ACALL** или **LCALL** в зависимости от адресного расстояния вызываемой подпрограммы.

По команде CALL в стеке сохраняется значение счётчика команд, и возврат из подпрограммы осуществляется в то место основной программы, откуда был осуществлен вызов (к команде основной программы, следующей за командой CALL). Для этого любая подпрограмма должна заканчиваться командой возврата **RET**, осуществляющей восстановление содержимого программного счётчика из стека.

Достаточно часто возникает необходимость такой организации вычислительного процесса, при которой подпрограмма вызывает другую подпрограмму, та в свою очередь вызывает следующую и т.д. Этот процесс

называется **вложением подпрограмм**. Число подпрограмм, которые могут быть вызваны таким образом (глубина вложенности подпрограмм), ограничивается только ёмкостью стека.

Сохранение параметров основной программы. Иногда при обращении к подпрограмме возникает необходимость сохранить не только адрес возврата в основную программу, но и содержимое отдельных рабочих регистров. Удобным способом для этого является **переключение банка регистров**. Например, если основная программа использует банк регистров 0, то подпрограмма может использовать банк регистров 1. Однако переключение банка регистров не обеспечивает сохранение содержимого аккумулятора, что приводит к необходимости создавать в одном из рабочих регистров или в памяти копию аккумулятора.

Передача параметров. Для успешной работы любой подпрограммы необходимо однозначно определить способ передачи в неё исходных данных и способ вывода результата её работы. Подпрограмма, которой требуется дополнительная информация в виде параметров её настройки или операндов, называется **параметризуемой**.

Получили распространение четыре способа передачи параметров: через память, через регистры общего назначения, через регистр признаков и через стек.

При передаче входных параметров **через память** основная программа обязательно содержит команды загрузки некоторых ячеек памяти, а подпрограмма – команды считывания из этих ячеек. При передаче входных параметров подпрограмма должна загрузить некоторые ячейки памяти, а основная программа – считать.

Передача параметров **через регистры** осуществляется аналогичным образом.

Третий способ передачи параметров – **через регистр признаков** – удобно использовать при передаче выходных параметров (например, в подпрограммах сравнения чисел). В этом случае подпрограмма должна установить (или сбросить) соответствующие признаки, а основная программа – проанализировать их значение. Intel 8051 обладает большими возможностями для передачи параметров через признаки. В нём имеется 128 флагов пользователя, доступных для модификации и анализа.

Способ передачи **через стек** позволяет использовать в качестве параметра содержимое счётчика команд.

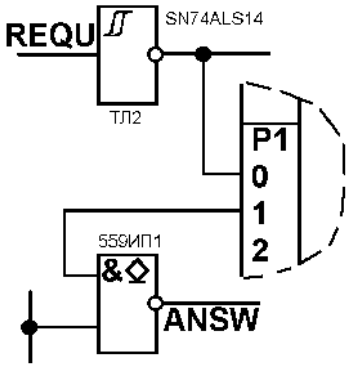
Использование процедур, оформленных в виде подпрограмм, при разработке программного обеспечения имеет ряд достоинств. **1)** Прежде всего относительно простые модули, выделенные из сложной программы,

могут программироваться несколькими разработчиками с целью сокращения времени проектирования. **2)** Еще более важным является то, что любая подпрограмма допускает автономную отладку. Это, как правило, многократно сокращает время отладки всего прикладного программного обеспечения. **3)** И, наконец, механизм использования подпрограмм уменьшает длину прикладной программы, что имеет своим следствием уменьшение требующейся ёмкости памяти программ.

4) Существенным является и то обстоятельство, что отлаженные процедуры организуются разработчиками в библиотеки параметризуемых подпрограмм и могут быть многократно использованы в проектной работе. Библиотека подпрограмм должна строиться на основе соглашения о едином способе обмена параметрами.

4.3. Простой пример

Часть задачи, например системы сигнализации, приведена на рисунке 18. Здесь на простой запрос в системе нужно сгенерировать ответ. В качестве запроса выступает положительный импульс, поступающий по линии REQU. Ответом является периодическая последовательность ANSW положительных импульсов заданной скважности. Для начального решения параметр скважности точно выдерживать не обязательно.



Вход запроса – REQU
Выход ответа – ANSW

Рисунок 18 – Фрагмент системы отклика

Для устранения помех запрос принимается на триггер Шмитта. Поэтому в программе ловить необходимо отрицательный импульс на входе 0 порта P1. Ответ генерируем на выходе 1 порта P1 и передаём его в линию через мощный передатчик с открытым коллектором (ОК), а нагрузка на приёмном конце. Следовательно программно генерируем последовательность отрицательных импульсов. Период задаём с помощью таймера T0. В силу сравнительно большого периода (например 256 мкс) для исключения непроизводительных расходов процессорного времени задействуем систему прерываний от таймера T0.

Воспользуемся интегрированной средой разработки **µVision 5** компании Keil с набором инструментов C51.

Решение приведём на машинном языке, языке Ассемблера и на языке высокого уровня Си.

4.3.1 Создание проекта

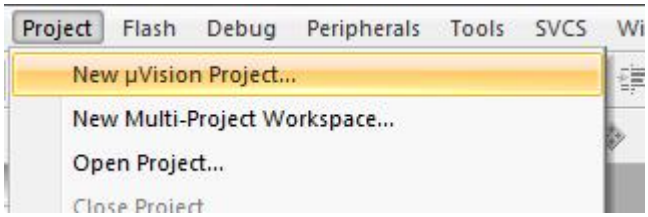
Интегрированную среду разработки можно вызвать разными способами, один из которых – просто щелкнуть мышкой пиктограмму программы, изображенную на рисунке 19.

Ярлык на рабочем столе

Рисунок 19 – Пиктограмма среды разработки µVision 5 Keil

При любом способе запуска среды изначально откроется основное окно интегрированной среды разработки с дочерними окнами, панелями инструментов и меню, где проводится вся работа.

Начинать надо с создания нового проекта. Команда меню **Project – New µVision Project (Проект – Новый µVision Проект)** создает новый проект, как показано на рисунке 20. Появится стандартный диалог выбора папки и задания имени проекта. В ответ выберите пустую папку и введите название проекта, например *Mash1*. Создается файл проекта с этим именем, расширение файла – *.**UVPROJ**. Рекомендуемая практика, чтобы использовать отдельный каталог для каждого проекта.



Выпадающее меню

Рисунок 20 – Создание нового проекта

Затем среда самостоятельно открывает диалоговое окно **Select Device for Target (Выбор Устройства для Цели)**, пример которого представлен на рисунке 21. Этот диалог пропустить нельзя ни при каких условиях. Здесь задаётся конкретное устройство – микроконтроллер – для своего приложения. Этот выбор определяет существенные параметры настройки инструментов, такие как средства управления компилятором, компоновщиком и симулятором, расположение памяти для компоновщика и алгоритмы программирования Flash памяти.

Верхнее поле ввода на закладке позволяет выбрать базу данных устройств. По умолчанию это поле – **Software Packs** (**Программные пакеты**) – для инструментов C51 неактивно, и используется заранее подготовленный набор имеющихся устройств. Из готового набора устройств нужно выбирать в левом окне иерархического списка, спадающего вниз. Чтобы упростить процесс выбора, можно в поле ввода **Search (Поиск)** задать общее название семейства, например 8032, как на рисунке 21.

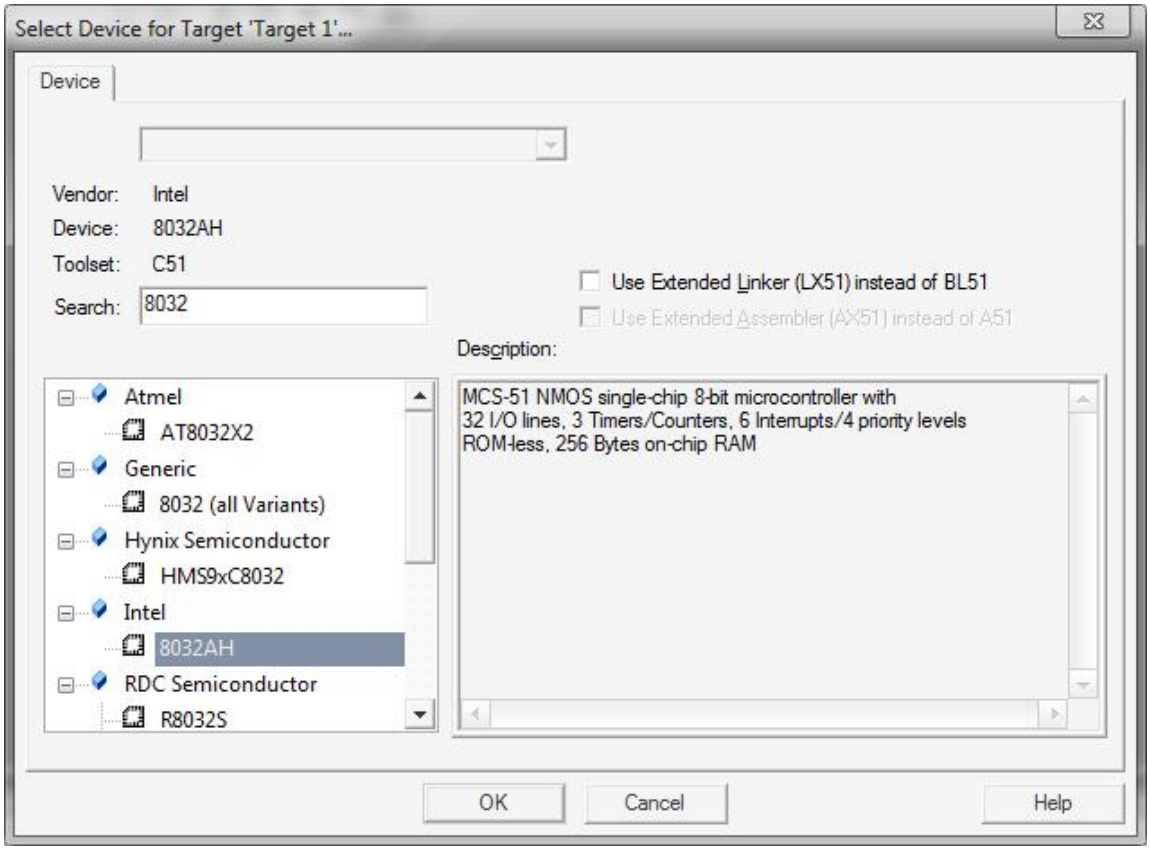


Рисунок 21 – Выбор микроконтроллера для проекта

Программы для встроенных систем требуют некоторого кода инициализации микроконтроллера, который должен соответствовать конфигурации разрабатываемых аппаратных средств. Код запуска, поставляемый со средой µVision, конфигурирует устройство микроконтроллера и инициализирует систему времени выполнения компилятора – модуль C0.

Для большинства случаев достаточно скопировать предлагаемый код запуска в проект. Это требуется почти для всех проектов. Исключением могут быть проекты библиотек и дополнительные проекты. Файл, содержащий код запуска, должен быть скопирован в папку проекта, потому что могут потребоваться его настройка и модификация, чтобы соответствовать целевым аппаратным средствам. С этой целью среда выводит окно

сообщения выбора файла конфигурации – **Copy … and Add the Startup Code (Скопировать … и добавить код запуска)**, которое показано на рисунке 22.

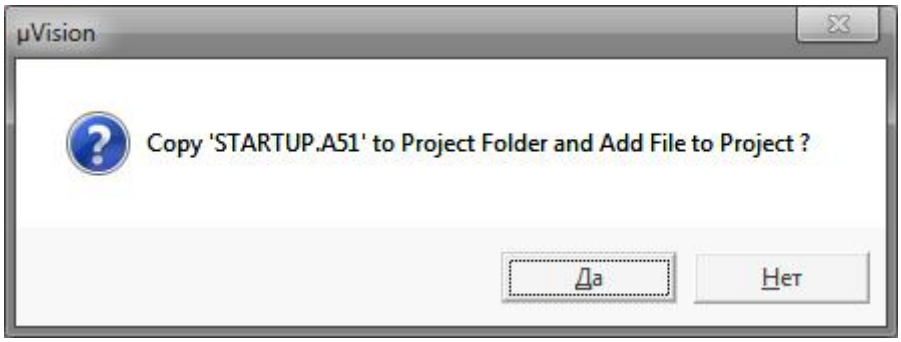


Рисунок 22 – Окно сообщения выбора файла конфигурации

Поэтому, в общем случае, ответьте на этот вопрос нажатием кнопки **Да (Yes)**. В папке проекта появится файл **STARTUP.A51**.

В первом проекте на машинном языке файл конфигурации не нужен, и мы нажимаем кнопку **Нет**. В итоге будет создан пустой проект, приведенный на рисунке 23. В нём имя цели проекта переименовано на Mashina0.

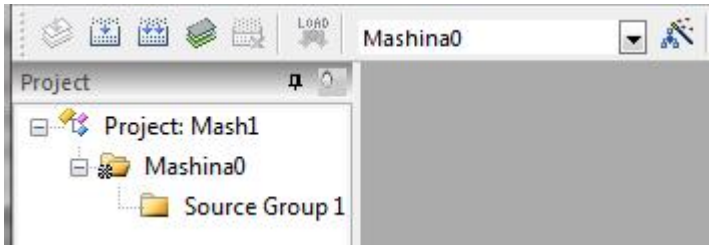


Рисунок 23 – Пустой проект

Следующий шаг в работе над проектом – настроить инструментальные средства среды. **Опции инструментов (Tool options)** формируют целевые аппаратные средства и среду разработки. Варианты зависят от выбранного устройства и набора инструментальных средств, отобранного для применения. Все эти параметры настройки предназначаются для одной цели проекта. Можно определить несколько целей в проекте для различного назначения, например, одна цель для тестирования, а другая для конечного применения.

Кнопка панели инструментов – – или команда меню **Project – Options for Target (Проект – Возможности для Цели)** открывают диалоговое окно настроек, приведенное на рисунке 24. Здесь присутствует много закладок настройки каждого инструмента или выборов проекта.

Например, закладка **Target (Цель)** позволяет определять все соответствующие параметры целевых аппаратных средств и компонентов на мик-

росхеме микроконтроллера. Первая закладка **Device (Устройство)** служит для выбора микроконтроллера, и работа с ней уже обсуждалась выше при создании проекта. Тем не менее, с её помощью можно сменить микроконтроллер в любой момент времени.

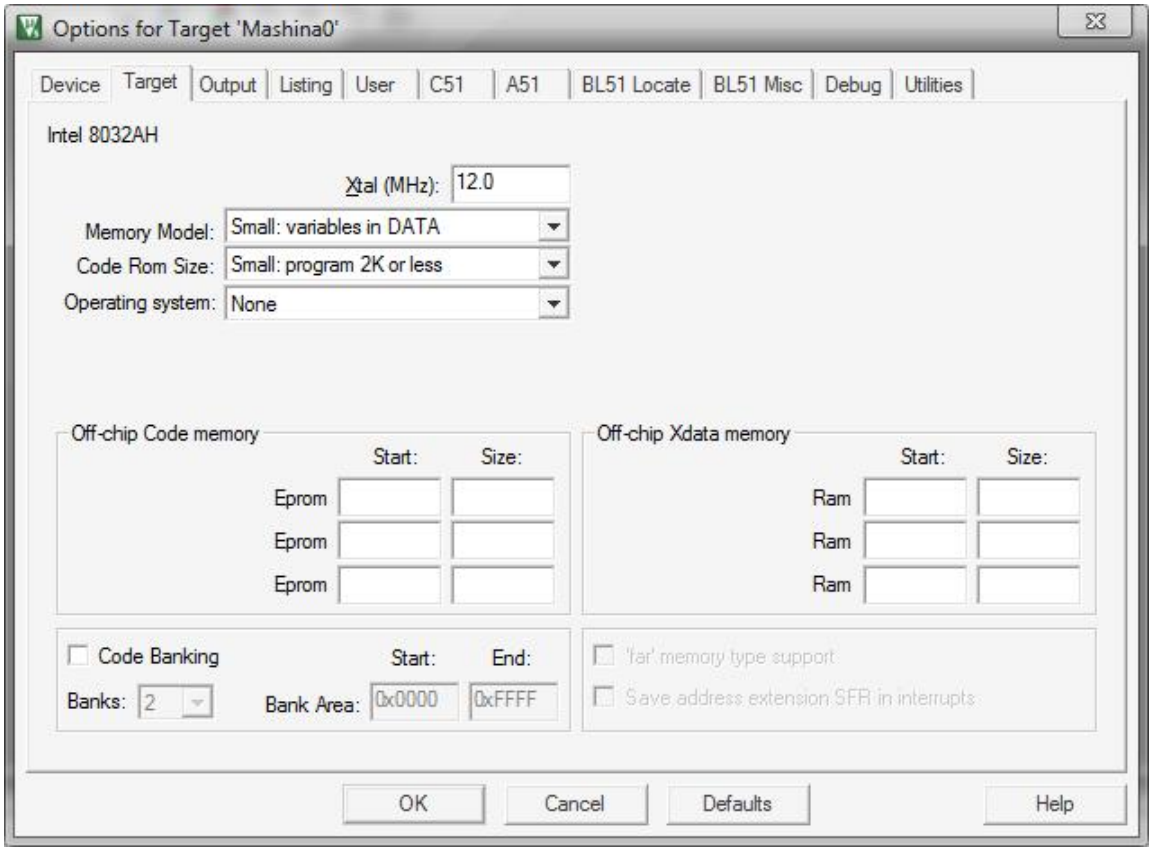


Рисунок 24 – Диалоги настроек проекта

Можно устанавливать отдельные параметры для выбранных пунктов проекта, которые отличаются от параметров настройки, определенных для всей цели проекта.

В проекте на машинном языке незначительной правки заслуживает закладка **Debug (Отладка)**, представленная на рисунке 25.

Диалоговое окно **Debug (Отладка)** определяет варианты, которые применяются, когда начнется сеанс отладки. Экран разделяется на возможности для программного симулятора и для аппаратных средств отладки.

Работая на машинном языке, здесь надо убрать флажок (снять галочку) выбора переключателя **Load Application at Startup (Загрузка Приложения при запуске)**. Установленный флажок заставляет среду загружать объектный файл в отладчик µVision. Но в пустом проекте на машинном языке нет ни исходных файлов, ни объектных.

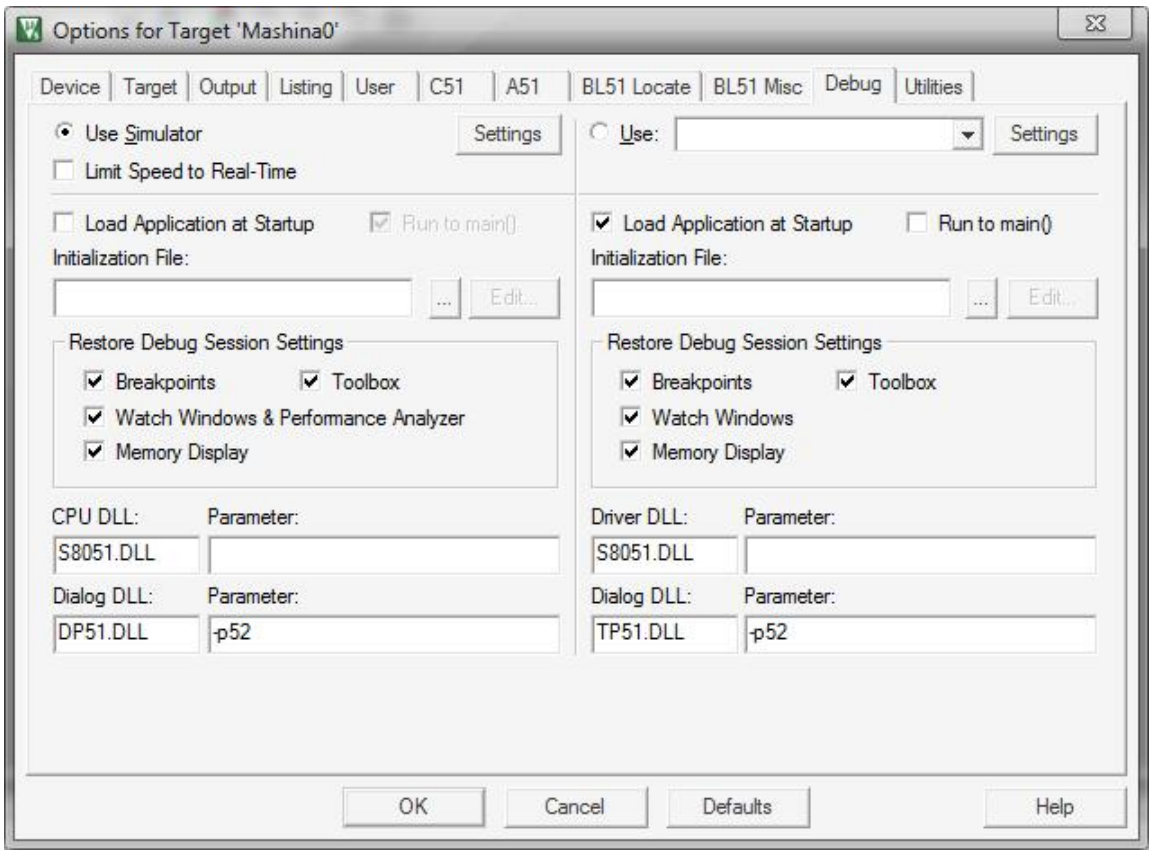


Рисунок 25 – Закладка диалога настроек отладки

Для входа в отладчик нужно нажать кнопку панели инструментов или выполнить команду меню **Debug – Start/Stop Debug Session (Отладка – Начало/Остановка Сеанса Отладки)**. Среда перейдет из режима программирования в режим отладки.

Программирование на машинном языке осуществляется в отладчике-симуляторе и отображается в окне дизассемблера. Есть два входа во встроенный ассемблер: командная строка отладчика и локальный диалог. Третий способ – воспользоваться редактором формул. Рабочее окно отладчика приведено на рисунке 26.

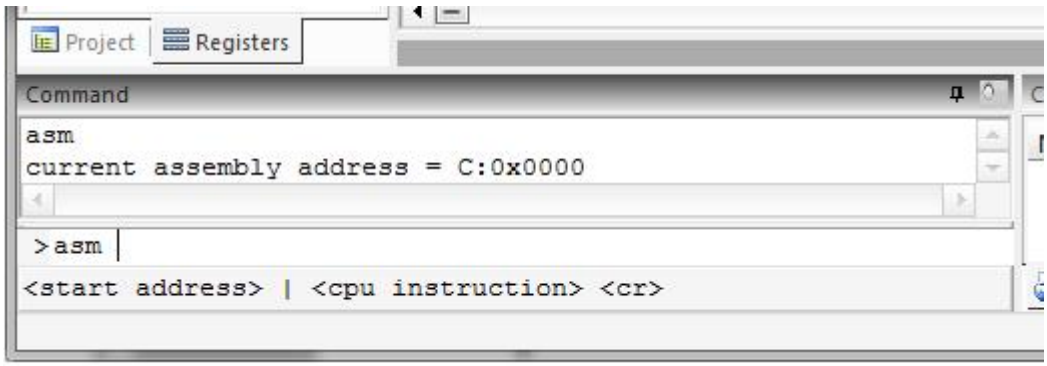


Рисунок 26 – Окно команд и вывода отладчика

Работа с командами отладки подробно описана в пособии [1].

Для модификации фрагмента кода удобнее использовать локальное диалоговое окно встроенного Ассемблера отладчика, которое приведено на рисунке 27.

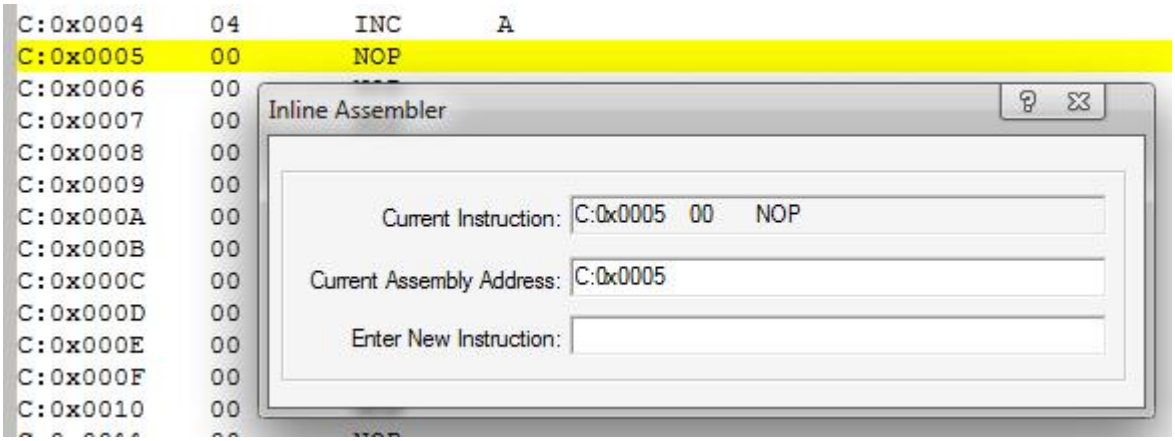


Рисунок 27 – Окно встроенного Ассемблера отладчика

Команда меню **Debug – Inline Assembly (Отладка – Встроенный Ассемблер)** открывает диалог для оперативного изменения текущей команды во время выполнения отладки. Модифицировать можно последовательный набор нескольких команд. Диалог работает только с наборами инструментальных средств C51 и C251. Этим способом вводятся и несколько фрагментов кода, меняя при этом адрес начала.

Окно Дизассемблера показывается вместе с окном Встроенного Ассемблера (Inline Assembler) и прокручивается к Текущему Адресу Ассемблера (Current Assembler Address), который выделяется цветом фона (по умолчанию отмечен желтой полосой).

Поле ввода с запретом записи **Current Instruction (Текущая команда)** показывает текущую команду вместе с адресом, кодом операции и мнемоникой. Эту команду в данный момент можно изменить или вообще покинуть встроенный Ассемблер.

Поле ввода **Current Assembly Address (Текущий адрес Ассамблирования)**. Здесь устанавливается новый адрес машинной команды для начала модификации кода. Нужно изменить прежнее числовое значение и нажать клавишу **Enter (Ввод)** или просто покинуть поле ввода, чтобы закончить изменение. Текущая команда, отмеченная желтой полосой фона, прокрутится к соответствующему адресу

Поле ввода **Enter New Instruction (Ввод Новой Инструкции)**. Здесь нужно напечатать новую машинную инструкцию, которая заменит старую инструкцию, показанную в поле Текущая команда (Current Instruction). Нажмите клавишу **Ввод (Enter)**, чтобы закончить ввод команды. Область

поля Ввод Новой Инструкции (Enter New Instruction) очистится для следующего ввода, а Текущая команда (Current Instruction) перейдет к следующей команде.

Результирующая программа появляется в окне дизассемблера, которое включается командой меню **View – Disassembly Window (Представление – Окно Дизассемблера).** Диалог встроенного ассемблера вызывается также из локального меню. Правая кнопка мыши всегда вызывает контекстное меню.

4.3.2 Машинный язык

Ниже приведен список команд отладчика.

```
// start комментарии допустимы в отладчике
asm 0
asm sjmp 30h

// main только генерировать периодический ответ
asm 30h
asm orl tmod,#1
asm mov tl0,#0
asm mov th0,#0ffh
asm orl ie,#82h
asm setb tr0
asm jmp 3eh

// interrupt T0
asm 0bh
asm push psw
asm mov psw,#8
asm push acc
asm clr tr0
asm mov a,tl0
asm add a,#7
asm mov tl0,a
asm mov a,th0
asm addc a,#0ffh
asm mov th0,a
asm setb tr0
asm clr p1.1
asm setb p1.1
asm pop acc
asm pop psw
asm reti

e char c:0=80h,2eh
e char c:3eh=80h,0feh
```

На машинном языке следует задать поиск запроса, длительность импульса ответа и поддержку отладки, что демонстрируется в следующих трёх столбцах:

```
// main program                  // Два варианта              // button start кнопка
// с поиском запроса             // длины импульсов           // для отладки
asm 30h                          … … …                       //define button
asm orl tmod,#1                  asm clr p1.1                "start","p1.0=0"
asm mov tl0,#0                   asm nop                     define button
asm mov th0,#0ffh                … … …                       "start","port1 &=
asm orl ie,#82h                  asm mov r7,#5               0xFE"
//asm jb p1.0,3ch                asm djnz r7,22h
asm nop                          asm setb p1.1               // start
asm nop                          … … …                       asm 0
asm nop                                                      asm jmp 30h
asm setb tr0                                                 … … …
asm jmp 41h

e char c:41h=80h,0feh
```

4.3.3 Язык Ассемблера

Программа генерации отклика на языке Ассемблера в соответствии с принципом модульности состоит из трёх самостоятельных модулей:

```
; головной модуль для генератора с timer0        Модуль 1
$NOMOD51
clock   EQU     0FF00H
        PUBLIC  clock

TMOD    DATA    089H
TL0     DATA    08AH
TH0     DATA    08CH
IE      DATA    0A8H
TR0     BIT     088H.4
requ    BIT     090H.0

        PUBLIC  main
        NAME    MAINPRG
;=======================
; сегмент: головная процедура
MYMAIN  SEGMENT CODE
        RSEG    MYMAIN
;********************** main ****************************
main:
        ORL     TMOD,#01H
        MOV     TL0,#LOW( clock )
        MOV     TH0,#HIGH( clock )
        MOV     IE,#082H
```

```
WaitRequ:
        JB      requ,WaitRequ
        SETB    TR0
Loop:
        SJMP    $
        END
```

Модуль 2

```
; модуль обработчика прерываний для генератора с timer0
$NOMOD51

TL0     DATA    08AH
TH0     DATA    08CH
PSW     DATA    0D0H
ACC     DATA    0E0H
TR0     BIT     088H.4
answ    BIT     090H.1

        EXTRN   NUMBER ( clock )
        NAME    INTERRUPTT0PRG
;        PUBLIC  int_timer0
        USING   0
;=======================
; сегмент: обработчик прерывания
        CSEG    AT      0BH
int_timer0:
        PUSH    ACC
        PUSH    PSW
        CLR     TR0
        MOV     A,TL0
        ADD     A,#LOW( clock + 7 )
        MOV     TL0,A
        MOV     A,TH0
        ADDC    A,#HIGH( clock + 7 )
        MOV     TH0,A
        SETB    TR0
; выдать ответ
        CLR     answ
        SETB    answ
; завершение
        POP     PSW
        POP     ACC
        RETI
        END
```

Модуль 3

```
; стартовый модуль для генератора с timer0
$NOMOD51
        EXTRN   CODE ( main )
```

```
        PUBLIC  stack
        NAME    STARTT0PRG
;=======================
; сегмент: стартовый код
        CSEG    AT      0
        SJMP    main
; без очистки внутренней памяти!
;=======================
; сегмент: рабочий стек
MYSTACK SEGMENT IDATA
        RSEG    MYSTACK
stack:
        DS      5
        END
```

4.3.4 Язык высокого уровня Си

Исходные тексты имеют следующее содержимое:

```
#pragma CODE

// определение регистров SFR
#include <reg51.h>
// глобальные константы и переменные
//const unsigned int clock = 0xFF00; // переменная!!
#define clock 0xFF00
sbit requ = P1^0;      // запрос
sbit answ = P1^1;     // ответ

//static void int_timer0( void ) interrupt 1 {
static void int_timer0( void ) interrupt 1 using 1 {
  TR0 = 0;
  TL0 += clock + 7; // внимание 0!
  if( CY ) TH0++;
  TH0 += ( clock + 7 )>> 8;
  TR0 = 1;
  answ = 0;
  answ = 1;
}

//************** main *****************
void main ( void ) {
// начальная работа
// . . . . .
// интересующий нас фрагмент
  TMOD |= 1;
  TL0 = ( unsigned char )clock;
  TH0 = ( unsigned char )( clock >> 8 );
  IE = 0x82; // включить систему прерываний
```

```
  while( requ ) ;
  TR0 = 1; // запустить таймер
  while( 1 ); // ждать
}
```

5 ПРИМЕР СИСТЕМЫ ДОЗИРОВАННОГО НАГРЕВА БИООБЪЕКТОВ

Система включает ВЧ установку общего нагрева в объёме, систему водяного нагрева и охлаждения, систему воздушного нагрева и охлаждения, контроллер термометрии, контроллер ВЧ мощности, управляющую ЭВМ и т.д. – целый ряд мелких подсистем и устройств.

Решаем часть задачи управления – **контроллер термометрии** (КТ). КТ должен – 1). измерять температуру с помощью полупроводниковых датчиков не менее, чем с 8 точек съёма, с точностью не хуже 0.05 град. Цельсия; 2). управлять нагревом воды для водяной системы; 3). управлять нагревом воздуха для воздушной системы; 4). поддерживать смену датчиков, т.е. обеспечивать калибровку с заданной точностью; 5). выбирать датчики для конкретного сеанса, мобильно переключать их; 6). взаимодействовать по интерфейсу RS-232 с управляющей ЭВМ, поддерживая помехозащищённый алгоритм со скоростью передачи не ниже 57600 бод; 7). разные скорости измерения и передачи температурных значений; 8). менять алгоритм взаимодействия с управляющей ЭВМ; и т.д. и т.п..

Состав контроллера КТ. **1)** МК Intel 80C31 – небольшая резидентная память данных, нет резидентной памяти программ, поддерживает интерфейс RS-232 [6] и [7]. **2)** Два блока памяти программ [8]. Первый блок содержит монитор загрузки в ПЗУ. Второй блок содержит собственно управляющую программу в ОЗУ. Размер блоков 2 Кб. Предусмотрено расширение до 8 Кб для каждого. [8]. **3)** Один блок памяти данных ОЗУ 2 Кб, расширяем до 8 Кб. Это резерв системы для будущих расширений [8]. Можно ввести дополнительную статистическую обработку для повышения точности измерения температуры с возможным изменением метода измерения температуры вплоть до оптических. **4)** Аналоговый коммутатор на 8 каналов [1] и [5]. Для управления коммутатором используется регистр номера датчика (РНД). **5)** Измерительный усилитель [4] и [5]. Должен обеспечивать балансировку для всех сменных датчиков. Напряжение балансировки получается с ЦАП. Для управления ЦАП нужен регистр кода баланса (РКБ) и РНД. **6)** АЦП 12-разрядный, типа AD1674, поддерживающий интерфейс с 8- и 16-разрядными микропроцессорами. Результат преобразования выдаёт 1 или 2 байтами для регулирования скорости пересылки и точности измерения температуры [7]. **7)** Регистр управления

нагревом воды и воздуха (РНВВ) [2] и [3]. **8)** Устройство управления с дешифратором адреса внешней памяти через 8 Кб. Все регистры отображены на память (XDATA). И т.д… средства гальванической развязки на оптронах, система автономного питания.

Необходимо разработать **систему команд управления** для помехоустойчивого взаимодействия с управляющей ЭВМ (протокол обмена). На языке Си сверстать программу, а на ассемблере оптимизировать.

Структурная схема контроллера термометрии приведена на рисунке 28.

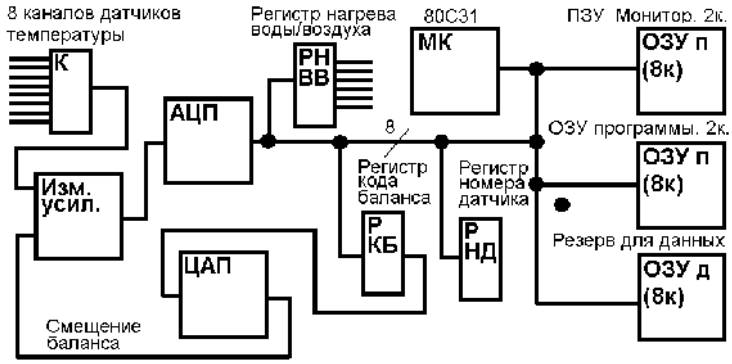


Рисунок 28 – Структурная схема контроллера термометрии

Память монитора предназначена для загрузки программы в память программы.

Пример программы 1.

```
;$CASE          ;для AX51 A251

$SET (TYPERPOGR = 1)
$DATE (01/Jan/00)
$DEBUG
;$ERRORPRINT (IZM.ERR)
$ERRORPRINT    ;на консоль
$PAGELENGTH (5000)
$NOMOD51
;$INCDIR (H:\Keil6.1\C51\ASM)
$INCLUDE ( reg51.inc )          ;#include <reg51.inc>
$TITLE (IZMERITEL TEMPERATURY Program N2)
$GEN

        NAME    IzmeritelTemperatury
;       IZMTEMP.ASM
; Загружаемая часть монитора измерителя температуры,входящего
в состав
; установки для проведения сеансов общей гипертермии (80С31)
;
```

```
     DSEG AT 0H
DATAIN    DATA 0H        ;начало сегмента
;-----------------------------------------
;       Адреса внутренних переменных
;
FLAGDT  DATA    20H            ;регистр флагов подключ. дат-
чиков температуры
KREGUN  DATA    21H            ;копия регистра управления
нагревом
KODUPR  DATA    22H            ;код управления, поступивший
от ПЭВМ
FLAGUPR DATA    28H            ;регистр флагов управления
RREGFL  DATA    40H            ;рабочий регистр флагов дат-
чиков
NOMDAT  DATA    41H            ;номер датчика температуры
MPFLAG  DATA    42H            ;местоположение устанав. или
сбрасыв. флага

     BSEG AT 20H
BITDAT    BIT  0H        ;начало сегмента
;-----------------------------------------

;       Символьные имена отдельных бит
;
FLREKT  BIT     FLAGDT.0       ;флаг ректальн. датчика ("1"-
дат. подключен)
FLKOGN  BIT     FLAGDT.1       ;флаг кожного датчика
FLVOD   BIT     FLAGDT.2       ;флаг водяного датчика
NAGRVOD BIT     KREGUN.0       ;включение нагрева воды ("1"-
включен)
OHLVOD  BIT     KREGUN.1       ;включение охлаждения воды
NAGRVOZ BIT     KREGUN.2       ;включение нагрева воздуха
SOSTDAT BIT     KODUPR.6       ;состояние датчика ("0"-ис-
ключ.,"1"-включ.)
PREDRU  BIT     FLAGUPR.0      ;бит предыдущего режима
управл. контроллером
PROGRET BIT     FLAGUPR.1      ;флаг "Контроллер прогрет"
(лог."1")
BALANS  BIT     FLAGUPR.2      ;флаг "Балансировка выпол-
нена" (лог."1")
ZATRKB  BIT     FLAGUPR.3      ;флаг "Затребован КБ"
(лог."1")
RAZRYAD BIT     FLAGUPR.4      ;текущ. разрядн. АЦП ("0"-12
бит,"1"-8 бит)
TIPKOM  BIT     ACC.7          ;тип команды ("0"-первый,"1"-
второй)
;-----------------------------------------
;       Символьные имена бит портов 1 и 3
```

```
;
GOTACP  BIT     P3.2            ;готовность АЦП ("0"-конец
преобразования)
REGUPR  BIT     P1.3            ;режим управления ("0"-
авт.,"1"-ручн.)
RAZRACP BIT     P1.6            ;разрядность АЦП ("0"-12
бит,"1"-8 бит)
REGACP  BIT     P1.7            ;режим работы АЦП ("0"-преоб-
раз.,"1"-чтение)

;----------------------------------------
;       Адреса  портов  контроллера
;
     XSEG AT 4000H
EXTDATA XDATA  4000H
$NOCOND
$IF (TYPERPOGR = 0)
  AREGKB  EQU     4000H         ;адреса регистров кода ба-
лансировки,
  AREGND  EQU     6000H         ;       номера датчика тем-
пературы и
  AREGUN  EQU     8000H         ;       управления нагревом
воды и воздуха,
  ADRACP  EQU     0A000H        ;адрес АЦП
$ELSE
          ORG     4000H
  AREGKB: DS      1               ;адреса регистров кода ба-
лансировки,
          ORG     6000H
  AREGND: DS      1               ;       номера датчика
температуры и
          ORG     8000H
  AREGUN: DS      1           ;       управления нагревом
воды и воздуха,
          ORG     0A000H
  ADRACP: DS      1             ;адрес АЦП
$ENDIF
$COND
$EJECT

     CSEG AT 0H
CODE0     CODE 0H        ;начало сегмента
;----------------------------------------
;       Адреса внешних символов
;
;PUSK    EQU     30H
     ORG     0000H
;----------------------------------------
```

```
;       Отладочная часть
;
        SJMP    PUSK
        ORG     0030H
PUSK:   LJMP    2000H           ;адрес перезапуска контрол-
лера

;-----------------------------------------
;       Коды сообщений и запросов котроллера,пересылаемые
управляющей ПЭВМ
;
NEPROGR EQU     0               ;контроллер не прогрет
DANBAL  EQU     1               ;передать данные для ба-
лансировки
OTKACP  EQU     2               ;отказ АЦП
OPROSDT EQU     3               ;выполняется опрос датчиков
температуры
AVTREG  EQU     4               ;измерение температуры под
управлением ПЭВМ
RUCHREG EQU     5               ;ручное измерение температуры
KBREKT  EQU     6               ;нет КБ ректального,
KBKOGN  EQU     7               ;       кожного,
KBVOD   EQU     8               ;       водяного датчиков
температуры,
REZKOM  EQU     9               ;резервная команда
;
; PRIZND+(NOMDAT) – передать КБ указанного датчика
;
; PRIZKT1+(NOMDAT) – принять 8-миразряд. код температуры ука-
занного датчика
; <код температуры> – 7...0 двоичные разряды
;
; PRIZKT2+(NOMDAT) – принять 12-тиразряд. код температуры
указанного датчика
; <ст.байт кода темп.> – 11...4 двоичные разряды
; <мл.байт кода темп.> – ст. тетрада – 3...0 разряды,мл.
тетрада – 0
;
; PRIZSK – принять 3 байта состояния контроллера
; (FLAGUPR) – содержимое регистров флагов управления,
; (FLAGDT) –    флагов датчиков температуры и
; (KREGUN) –    управления нагревом
;
;       Коды команд от управляющей ПЭВМ
;
PEREZAP EQU     0               ;перезапуск контроллера
OTMPROG EQU     1               ;отмена прогрева контроллера
SOSTOYA EQU     2               ;выдать состояние контроллера
```

```
KONBAL  EQU     3               ;конец балансировки
VKLNVOD EQU     4               ;включить нагрев воды
VYKNVOD EQU     5               ;выключить нагрев воды
VKLOVOD EQU     6               ;включить охлаждение воды
VYKOVOD EQU     7               ;выключить охлаждение воды
VKLNVOZ EQU     8               ;включить нагрев воздуха
VYKNVOZ EQU     9               ;выключить нагрев воздуха
RAZR8B  EQU     10              ;установить разрядность 8 бит
RAZR12B EQU     11              ;установить разрядность 12
бит
;
; 80H+<номер датчика> – исключить указанный датчик из цикла
опроса
;
; C0H+<номер датчика> – включить указанный датчик в цикл
опроса
; <КБ датчика>
;
; Примечания. 1) <номер датчика> – 3 двоичных разряда
;             2) <КБ датчика> передается после получения за-
проса
;                "Передать КБ указанного датчика"
;
;       Значения остальных символьных имен
;
NADRKB  EQU     44H             ;начальный адрес хранения ко-
дов балансировки
PRIZND  EQU     80H             ;признаки запроса КБ указ.
датчика температ.,
PRIZKT1 EQU     90H             ; выдачи 8-миразрядного или
PRIZKT2 EQU     0A0H            ;  12-тиразряд. кода темпера-
туры указ. датч.,
PRIZSK  EQU     0AH             ; выдачи состояния контрол-
лера

     CSEG AT 2000H
CODE1     CODE 2000H          ;начало сегмента
     ORG     2000H
;-----------------------------------------
;       Начальная настройка контроллера
;

     MOV     SP,#07H
     CLR     A
     MOV     IE,A            ;запрещение прерываний
     MOV     R7,A            ;обнуление счетчика цикла опроса
датчиков,
     ACALL   ZREGUN          ; регистра управления нагревом
```

```
     MOV     KREGUN,A        ; и его копии в ОЗУ
     MOV     FLAGDT,A        ;сброс флагов подключен. датчи-
ков температ.,
     CLR     PROGRET         ; флагов "Контоллер прогрет",
     CLR     BALANS          ;        "Датчики отбалансиро-
ваны" и
     CLR     ZATRKB          ;        "Затребован КБ",
     CLR     RI              ;а также флагов готовности при-
емника и
     CLR     TI              ; передатчика последовательных
данных
     SETB    RAZRYAD         ;установка флага 8-мибитной раз-
рядности АЦП
     SETB    REN             ;разрешение приема последова-
тельных данных
     ACALL   ZAPRU           ;запоминание режима управления и
     JNB     PREDRU,AVTUPR1  ;переход,если автоматический
;----------------------------------------
;       Ручное управление измерением температуры
;
     MOV     R4,#30          ;установка 10-тиминут. прогрева
контроллера
     MOV     R5,#200
RUCHUPR:MOV     A,#RUCHREG      ;выдача сообщения:
     ACALL   SOOBSH          ; "Ручное измерение температуры"
L0:     JNB     RI,L3           ;переход,если нет данных от
ПЭВМ,
     ACALL   PRIEM           ;иначе прием команды
     CJNE    A,#PEREZAP,L1   ; "Перезапуск контроллера"? –
нет,продолжить
     LJMP    PUSK            ;да,переход
L1:     CJNE    A,#OTMPROG,RUCHUPR;"Отмена прогрева кон-
трол."? – нет,переход
L2:     SETB    PROGRET         ;да,установка флага "Контрол-
лер прогрет"
L3:     JB      PROGRET,L5      ;переход,если контроллер про-
грет,
     ACALL   ZADER2          ;иначе временная задержка на
0,1с
     DJNZ    R5,L5
     DJNZ    R4,L4
     SJMP    L2              ;переход,если время прогрева ис-
текло
L4:     MOV     R5,#200
L5:     ACALL   IZMRU           ;режим управления изменился?
     JNC     L0              ;нет,переход
     ACALL   ZAPRU           ;да,запоминание нового положения
тумблера
```

```
     SJMP    AVTUPR2         ; "Режим управления" и переход
;----------------------------------------
;               Автоматическое управление измерением темпера-
туры
;
;       Прогрев контроллера в течение 10 минут
;
AVTUPR1:MOV     R4,#30          ;установка 10-тиминут. про-
грева контроллера
     MOV     R5,#200
AVTUPR2:MOV     A,#AVTREG       ;выдача сообщения:
     ACALL   SOOBSH          ; "Измерение температуры под
управлен. ПЭВМ"
     JB      PROGRET,BALANDT ;переход,если контроллер прогрет
или
L6:     JNB     RI,L9           ; нет данных от ПЭВМ,иначе
     ACALL   PRIEM           ;прием команды
     CJNE    A,#PEREZAP,L7   ;"Перезапуск контроллера"? –
нет,продолжить
     LJMP    PUSK            ;да,переход
L7:     CJNE    A,#OTMPROG,L8   ;"Отмена прогрева" – нет,про-
должить
     SJMP    L10             ;да,переход
L8:     MOV     A,#NEPROGR      ;выдача сообщения:
     ACALL   SOOBSH          ; "Контроллер не прогрет"
L9:     ACALL   ZADER2          ;временная задержка на 0,1 с
     DJNZ    R5,L12
     DJNZ    R4,L11          ;если время прогрева истекло
L10:    SETB    PROGRET         ; установка флага "Контроллер
прогрет"
     SJMP    BALANDT
L11:    MOV     R5,#200
L12:    ACALL   IZMRU           ;режим управления изменился ?
     JNC     L6              ;нет,переход
L13:    ACALL   ZAPRU           ;да,запоминание нового поло-
жения тублера
     SJMP    RUCHUPR         ; "Режим управления" и переход
;----------------------------------------
;       Балансировка датчиков температуры
;
BALANDT:JNB     BALANS,M0       ;переход,если балансировка
выполнена или
     AJMP    IZMTEMP
M0:     JNB     ZATRKB,POVTCB   ; не принят КБ,
     AJMP    PRIEMKB
POVTCB: MOV     A,#DANBAL       ;иначе  выдача сообщения:
     ACALL   SOOBSH          ; "Передать данные для баланси-
ровки"
```

```
M1:     MOV     R5,#10          ;установка 1с-ожидания по-
ступления данных
M2:     ACALL   IZMRU           ;режим управления изменился ?
     JNC     M3              ;нет,продолжить
     SJMP    L13             ;да,переход к ручному управлению
M3:     JB      RI,M4           ;переход,если данные посту-
пили,
     ACALL   ZADER2          ;иначе 0,1с-задержка и
     DJNZ    R5,M2           ;переход,если тайм-аут не окон-
чился,
     SJMP    POVTCB          ;иначе повторить запрос данных
M4:     ACALL   PRIEM           ;прием байта,поступившего от
ПЭВМ
     JB      TIPKOM,KOM2T    ;переход,если команда 2-го типа,
     CJNE    A,#PEREZAP,M5   ;иначе "Перезапуск контроллера"?
     LJMP    PUSK
M5:     CJNE    A,#SOSTOYA,M6   ; "Выдать состояние контрол-
лера"?
     ACALL   SOSTKON
     SJMP    POVTCB
M6:     CJNE    A,#KONBAL,POVTCB; "Конец балансировки"?
     JB      FLREKT,M7       ;продолжить,если флаг ректаль-
ного установлен,
     MOV     A,#KBREKT       ;иначе выдача сообщуния:
     ACALL   SOOBSH          ; "Нет КБ ректального датчика"
     SJMP    M1
M7:     JB      FLKOGN,M8       ;продолжить,если флаг кожного
установлен,
     MOV     A,#KBKOGN       ;иначе выдача сообщения:
     ACALL   SOOBSH          ; "Нет КБ кожного датчика"
     SJMP    M1
M8:     JB      FLVOD,M9        ;продолжить,если флаг водя-
ного установлен,
     MOV     A,#KBVOD        ;иначе выдача сообщения:
     ACALL   SOOBSH          ; "Нет КБ водяного датчика"
     SJMP    M1
M9:     SETB    BALANS          ;установка флага "Баланси-
ровка выполнена" и
     AJMP    IZMTEMP         ;переход к измерению температуры
KOM2T:  ANL     A,#00111000B    ;команда балансировки ?
     JZ      M10             ;да,продолжить
     SJMP    POVTCB          ;нет,переход
M10:    MOV     A,KODUPR        ;формирование номера дат-
чика,к которому
     ANL     A,#00000111B    ; относится эта команда,
     MOV     NOMDAT,A
     ADD     A,#NADRKB       ; адреса ячейки хранения его КБ
и
```

```
     MOV     R0,A
     MOV     R1,NOMDAT       ; местоположения флага этого
датчика
     INC     R1              ;  в регистре FLAGDT
     MOV     A,#01H
M11:    DJNZ    R1,M12
     SJMP    M13
M12:    RL      A
     SJMP    M11
M13:    MOV     MPFLAG,A
     JB      SOSTDAT,M14     ;переход,если датчик включается
в систему,
     CPL     A               ;иначе сброс его флага в реги-
стре FLAGDT и
     ANL     FLAGDT,A
     AJMP    POVTCB          ;переход к следующему циклу ба-
лансировки
M14:    SETB    ZATRKB          ;установка флага "Затребован
КБ" и
PRIEMKB:MOV     A,#PRIZND       ;выдача сообщения:
     ADD     A,NOMDAT        ; "Передать КБ указанного дат-
чика"
     ACALL   SOOBSH
     MOV     R5,#10          ;установка 1с-ожидания поступле-
ния данных
M15:    ACALL   IZMRU           ;режим управления изменился ?
     JNC     M16             ;нет,продолжить
     AJMP    L13             ;да,переход к ручному управлению
M16:    JB      RI,M17          ;ожидание поступления байта
данных
     ACALL   ZADER2
     DJNZ    R5,M15
     SJMP    PRIEMKB
M17:    ACALL   PRIEM           ;и его прием,
     MOV     @R0,A           ;пересылка КБ в ячейку хранения,
     MOV     A,MPFLAG        ;установка флага,свидетельствую-
щего о его
     ORL     FLAGDT,A        ; поступлении,
     CLR     ZATRKB          ;сброс флага "Затребован КБ" и
     AJMP    POVTCB          ;переход к следующему циклу ба-
лансировки
;----------------------------------------
;       Измерение температуры и управление нагревом воды и
воздуха
;
IZMTEMP:MOV     NOMDAT,#0       ;;+
        ACALL   KOMUTAT         ;;+
     MOV  R6,#3          ;;+ задержка после коммутации
```

```
DP2:    ACALL   ZADER1          ;;+
     DJNZ    R6,DP2          ;;+
        CJNE    R7,#00H,N0      ;переход,если предыдущий цикл
опроса датчиков

NOVCIKL:MOV     RREGFL,FLAGDT   ; температуры не закон-
чен,иначе подготовка
     MOV     NOMDAT,R7       ; следующего цикла
     MOV     R7,#8
N0:     JB      RAZRYAD,N1      ;установка
     CLR     RAZRACP         ; 12-тибитной или
     SJMP    N2
N1:     SETB    RAZRACP         ; 8-мибитной разрядности АЦП
N2:     MOV     A,RREGFL        ;определение номера следую-
щего
        CLR     C               ; опрашиваемого датчика тем-
пературы и
N3:     RRC     A
     JC      N4
     INC     NOMDAT
     DJNZ    R7,N3
DP0: MOV  R6,#1          ;задержка между пакетами
DP1:    ACALL   ZADER1
     DJNZ    R6,DP1
     SJMP    NOVCIKL
N4:     MOV     RREGFL,A
        ACALL   KOMUTAT
     ACALL   ZADER1          ;задержка на время установл.
сигнала темпер.
     ACALL     ZADER1
     CLR     REGACP          ;перевод АЦП в режим "Преобразо-
вание" и
     ACALL     PUSKACP   ; его пуск
     MOV     R6,#4           ;задержка на время преобразова-
ния
N5:     DJNZ    R6,N5
     JNB     GOTACP,N6       ;данные АЦП готовы ? – да,пере-
ход
     MOV     A,#OTKACP       ;нет,выдача сообщения:

     ACALL   SOOBSH          ; "Отказ АЦП" и
     SJMP    N10             ;переход к проверке поступл. ко-
манды от ПЭВМ
N6:     ACALL   IZMRU           ;режим управления изменился ?
     JNC     N7              ;нет,переход
     INC     NOMDAT          ;да,корректировка номера датчика
температуры,
```

```
     DEC     R7              ; а также счетчика цикла опроса
и
     AJMP    L13             ;переход к ручному измерению
температуры
N7:     SETB    REGACP          ;перевод АЦП в режим "Чтение"
     JB      RAZRYAD,N8      ;формирование и
     MOV     A,#PRIZKT2
     SJMP    N9
N8:     MOV     A,#PRIZKT1
N9:     ADD     A,NOMDAT
        INC     NOMDAT          ;;+ формирование номера сле-
дующего датчика
        MOV     R6,A
        MOV     R5,NOMDAT
        MOV     A,RREGFL
        CJNE    A,#0,DP3
        MOV     NOMDAT,#0
DP3:    ACALL   KOMUTAT
        MOV     NOMDAT,R5
        MOV     A,R6
     ACALL   SOOBSH          ;выдача сообщения:
     CLR     RAZRACP         ; "Код температуры"
     ACALL   CHTACP          ;чтение и
     ACALL   SOOBSH          ;передача старшего байта кода
температуры,
     JB      RAZRYAD,N10     ;а в случае 12-тибитной разряд-
ности АЦП,
     SETB    RAZRACP         ; также его младшего байта
     ACALL   CHTACP
     ACALL   SOOBSH
N10:    JB      RI,VYPKOM       ;переход,если поступили дан-
ные от ПЭВМ,иначе
SLEDDAT:DJNZ    R7,N0           ;продолж. цикла опроса,если
он не закончился,
     SJMP    DP0
;       ACALL   ZADER1
;       AJMP    NOVCIKL         ;иначе переход к следующему
циклу
VYPKOM: ACALL   PRIEM           ;прием команды
     JNB     TIPKOM,N11      ;продолжить,если команда 1-го
типа,иначе
     CLR     BALANS          ;сброс флага "Балансировка вы-
полнена",
     MOV     R7,#00H         ; а также счетчика цикла опроса
и
     AJMP    KOM2T           ;переход к процедуре баланси-
ровки
```

```
N11:    CJNE    A,#PEREZAP,N12  ;команда "Перезапуск контрол-
лера" ?
     LJMP    PUSK            ;да,переход
N12:    CJNE    A,#OTMPROG,N13  ; "Отмена прогрева" ?
     SJMP    N14
N13:    CJNE    A,#KONBAL,N15   ; "Конец балансировки" ?
N14:    MOV     A,#OPROSDT      ;выдача сообщения:
     ACALL   SOOBSH          ; "Опрос датчиков температуры"
     SJMP    SLEDDAT
N15:    CJNE    A,#VKLNVOD,N16  ; "Включение нагрева воды" ?
     SETB    NAGRVOD
     SJMP    N21
N16:    CJNE    A,#VYKNVOD,N17  ; "Выключение нагрева воды" ?
     CLR     NAGRVOD
     SJMP    N21
N17:    CJNE    A,#VKLOVOD,N18  ; "Включение охлаждения воды"
?
     SETB    OHLVOD
     SJMP    N21
N18:    CJNE    A,#VYKOVOD,N19  ; "Выключение охлаждения
воды" ?
     CLR     OHLVOD
     SJMP    N21
N19:    CJNE    A,#VKLNVOZ,N20  ; "Включение нагрева воздуха"
?
     SETB    NAGRVOZ
     SJMP    N21
N20:    CJNE    A,#VYKNVOZ,N22  ; "Выключение нагрева воз-
духа" ?
     CLR     NAGRVOZ
N21:    MOV     A,KREGUN        ;соответствующая коррекция
содержимого
     ACALL   ZREGUN          ; регистра управления нагревом
     SJMP    SLEDDAT
N22:    CJNE    A,#SOSTOYA,N23  ; "Выдать состояние контрол-
лера" ?
     ACALL   SOSTKON
     SJMP    SLEDDAT
N23:    CJNE    A,#RAZR8B,N24   ; "Установить разрядность 8
бит" ?
     SETB    RAZRYAD
     SJMP    SLEDDAT
N24:    CJNE    A,#RAZR12B,N25  ; "Установить разрядность 12
бит" ?
     CLR     RAZRYAD
     SJMP    SLEDDAT
N25:    MOV     A,#REZKOM       ;выдача сообщения:
     ACALL   SOOBSH          ; "Резервная команда"
```

```
     SJMP    SLEDDAT
;-----------------------------------------
;               Подпрограммы
;-----------------------------------------
;       Запись в регистры управления и пуск АЦП
;
ZREGKB: MOV     DPTR,#AREGKB
     SJMP    LZ0
ZREGND: MOV     DPTR,#AREGND
     SJMP    LZ0
ZREGUN: MOV     DPTR,#AREGUN
     SJMP    LZ0
PUSKACP:MOV     DPTR,#ADRACP
LZ0:    MOVX    @DPTR,A
     RET
;-----------------------------------------
;       Запоминание режима управления
;
ZAPRU:  MOV     C,REGUPR
     MOV     PREDRU,C
     RET
;-----------------------------------------
;       Выдача сообщения на ПЭВМ
;
SOOBSH: MOV     SBUF,A
SL0:    JNB     TI,SL0
     CLR     TI
     RET
;-----------------------------------------
;       Прием данных от ПЭВМ
;
PRIEM:  MOV     A,SBUF
     MOV     KODUPR,A
     CLR     RI
     RET
;-----------------------------------------
;       Проверка изменения режима управления
;
IZMRU:  MOV     C,PREDRU
     JNB     REGUPR,LI0
     CPL     C
LI0:    RET
;-----------------------------------------
;       Временная задержка на 1мс или 0,1с
;
ZADER1: MOV     R2,#2
     SJMP    LZ1
ZADER2: MOV     R2,#200
```

```
LZ1:    MOV     R3,229
LZ2:    DJNZ    R3,LZ2
     DJNZ    R2,LZ1
     RET
;-----------------------------------------
;       Чтение данных АЦП
;
CHTACP: MOV     DPTR,#ADRACP
     MOVX    A,@DPTR
     RET
;-----------------------------------------
;       Выдача состояния контроллера
;
SOSTKON:MOV     A,#PRIZSK
     ACALL   SOOBSH
     MOV     A,FLAGUPR
     ACALL   SOOBSH
     MOV     A,FLAGDT
     ACALL   SOOBSH
     MOV     A,KREGUN
     ACALL   SOOBSH
     RET
;-----------------------------------------
;       Коммутация датчика
;
KOMUTAT:MOV     A,#NADRKB       ; адреса хранения его кода
балансировки
     ADD     A,NOMDAT
     MOV     R0,A            ; занесение в соответстующие ре-
гистры
     MOV     A,@R0
     ACALL   ZREGKB          ; кода балансировки датчика тем-
пературы и
     MOV     A,NOMDAT
     ACALL   ZREGND          ; его номера
        RET
;
        END
```

6 ЭТАП ОТ ИСХОДНОЙ ПРОГРАММЫ К ИСПОЛНИТЕЛЬНОМУ КОДУ

Все интегрированные среды разработки ПО решают общий комплекс задач, одинаковый для всех МПС. С точки зрения стратегии работы они работают одинаково. Тактические решения разные. Сильно отличаются дизайном окон и комплектом инструментария.

На простом примере по методу “Быстрый старт” изучаются этапы технологии разработки и отладки программ, основные приёмы работы на примере среды µVision2.

5.1. Введение в µVision2.

µVision2 фирмы Keil – интегрированная среда разработки программного обеспечения для однокристальных микроконтроллеров семейства Intel 8051 и его клонов. Она включает в себя всё, что нужно для создания, редактирования, компиляции, трансляции, компоновки, загрузки и отладки программ: 1) – стандартный интерфейс Windows, 2) – полнофункциональный редактор исходных текстов с выделением синтаксических элементов цветом, 3) – организатор проекта, 4) – транслятор с языка С, 5) – ассемблер, 6) – отладчик, 7) – встроенную справочную систему.

Среда разработки подобна Visual C++ Microsoft и Borland C++ для Windows. Пользователи, знакомые с любым из этих изделий, будут чувствовать себя в µVision2, как дома.

1. Первый этап разработки программы – запись её исходного текста на каком-либо языке программирования.

2. Затем производится компиляция или трансляция его в коды из системы команд микроконтроллера, используя транслятор или ассемблер. Трансляторы и ассемблеры – прикладные программы, которые интерпретируют текстовый файл, содержащий исходный текст программы, и создают объектные файлы, содержащие объектный код.

3. Компановка объектных модулей в исполнительный файл с помощью редактора связей (компановщика).

4. После компоновки объектных модулей наступает этап отладки программы, устранения ошибок, оптимизации и тестирования программы.

Интегрированная среда разработки µVision2 объединяет все этапы разработки прикладной программы в единый рекурсивный процесс, когда в любой момент времени возможен быстрый возврат к любому предыдущему этапу. Для этого µVision2 имеет следующие компоненты.

Макроассемблер A51

Ассемблер A51 совместим с ASM51 Intel для всего семейства микроконтроллеров Intel 8051. Ассемблер 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 Intel 8051.

A51 транслирует исходный файл ассемблера в перемещаемый объектный модуль. При отладке или при включенной опции “Include debugging information” этот объектный файл будет содержать полную символическую информацию для отладчика/имитатора или внутрисхемного эмулятора.

Оптимизирующий кросс-компилятор C51

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

C51 – полная реализация стандарта ANSI (Американского национального института стандартов), насколько это возможно для архитектуры Intel 8051. C51 генерирует код для всего семейства микроконтроллеров Intel 8051. Транслятор сочетает гибкость программирования на языке С с эффективностью кода и быстродействием ассемблера.

Использование языка высокого уровня С имеет следующие преимущества над программированием на ассемблере: 1) – глубокого знания системы команд процессора не требуется, элементарное знание архитектуры Intel 8051 желательно, но не необходимо; 2) – распределение регистров и способы адресации управляются полностью транслятором; 3) – лучшая читаемость программы, используются ключевые слова и функции, которые более свойственны человеческой мысли; 4) – время разработки программ и их отладки значительно короче в сравнении с программированием на ассемблере; 5) – библиотечные файлы содержат много стандартных подпрограмм, которые могут быть включены в прикладную программу; 6) – 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 транслятором C51, ассемблером A51, транслятором PL/M-51 Intel и ассемблером ASM51 Intel.

Компоновщик автоматически выбирает соответствующие библиотеки поддержки и связывает 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.

Отладчик/симулятор

Отладчик исходных текстов используется с транслятором C51, ассемблером A51, транслятором PL/M-51 Intel и ассемблером ASM51 Intel. Отладчик/симулятор позволяет моделировать большинство особенностей Intel 8051 без наличия аппаратных средств. Можно использовать его для проверки и отладки прикладной программы прежде, чем будут изготовлены аппаратные средства. При этом моделируется широкое разнообразие периферийных устройств, включая последовательный порт, внешний ввод – вывод и таймеры.

5.2. Быстрый старт

“Быстрый старт” – это обычный приём разработчиков современных программных средств. Цель состоит в том, чтобы, не углубляясь пока в подробности, дать новичку или достаточно опытному пользователю первое представление о программном средстве, дать возможность быстро получить конкретный результат. Полное представление, знания и умения появятся позже в процессе работы и изучения справочных материалов.

В качестве примера используется простейшая программа, с которой начинают изучение языков программирования многие поколения студентов. “Hello World” – программа из папки …\Examples\Hello\, которая выдаёт в последовательный порт (UART) микроконтроллера строку символов “Hello World” (“Привет Мир”). Весь исходный текст программы следующий:

Пример программы 2.

```
/***************************/
/* YOUR FIRST 8051 PROGRAM */
/***************************/
#include <reg51.h>    /* special function register declarations  */
                      /* for the intended 8051 derivative        */
#include <stdio.h>    /* prototype declarations for I/O functions*/

/****************/
/* main program */
/****************/
void main (void)  {          /* execution starts here after stack init */
   SCON  = 0x50;             /* SCON: mode 1, 8-bit UART, enable rcvr  */
   TMOD |= 0x20;             /* TMOD: timer 1, mode 2, 8-bit reload    */
   TH1 = 0xf3;               /* TH1:  reload value for 2400 baud       */
   TR1 = 1;                  /* TR1:  timer 1 run                      */
   TI  = 1;                  /* TI:   set TI to send first char of UART*/
   printf ("Hello World\n"); /* the 'printf' function call             */
```

```
   while (1) {                /* An embedded program does not stop and  */
   ;  /* ... */               /* never returns.  We've used an endless  */
   }                          /* loop.  You may wish to put in your own */
}                             /* code were we've printed the dots (...) */
```

5.2.1. Общий план работы

Для создания приложения в среде проектирования µVision2 необходимо выполнить следующие этапы:

1)- Запустить интегрированную среду µVision2, создать файл проекта, задать тип микроконтроллера, используемого в проекте.

2)- Создать новые исходные файлы программ на соответствующих языках программирования и включить их в проект в соответствующие группы.

3)- Дополнить проект конфигурационными файлами и настроить последние для заданной модели МК.

4)- Настроить опции проекта, рабочих групп и отдельных файлов, если в этом есть необходимость.

5)- Построить исполнительный файл проекта, провести его тестирование и отладку.

6)- Создать файл в формате HEX для программатора, и зашить программу в ПЗУ для контроллера.

5.2.2. Программа "Hello World"

Прежде чем начать разработку проекта, желательно создать свою личную папку. В этой папке находится всего лишь один файл с текстом данной программы – hello.c. Хотя, это можно сделать и позже.

Запуск µVision2 и создание файла проекта. µVision2 запускается из стартового меню Windows подобно остальным приложениям. Если необходимо запустить программу из командной строки, её синтаксис имеет вид:

*uv2 [commands] [projectfile]***,**

где projectfile – имя файла проекта с расширением [.Uv2], а необязательные команды управляют загрузкой.

Любая новая работа в µVision2, как и во всех современных интегрированных средах, начинается с создания нового файла проекта. Файл проекта содержит имена всех исходных файлов, связанных с проектом, а также установки компиляции, трансляции и связывания файлов, чтобы генерировать выполняемую программу.

Типичный вид окна интегрированной среды разработки программных средств для семейства MCS-51 µVision2

Окна µVision2

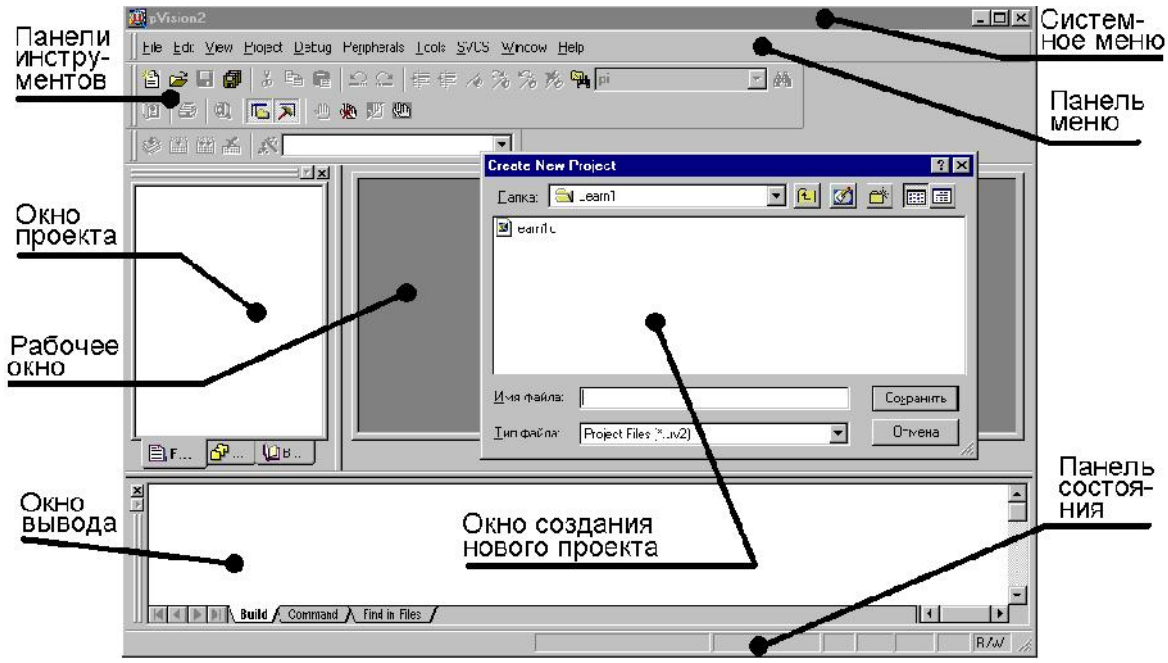


Рис. 11.

Тип устройства

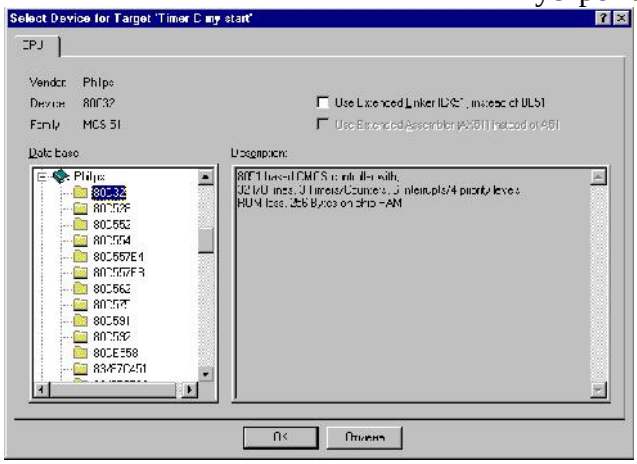


Рис. 12.

Для того чтобы создать новый файл проекта, выберите из меню **Project – New Project …** . Откроется стандартное диалоговое окно для создания файлов – Create Project (рис. 11). Используйте кнопки иконки, чтобы войти в свою папку. Найдите свою папку и нажмите кнопку [OK]. На этом этапе также можно создать свою папку с помощью иконки **Create New Folder** и в ней указать свой проект.

Теперь обязательно задать тип микроконтроллера. Он определяет важные параметры всех инструментов интегрированной среды. Тип задаётся

с помощью меню **Project – Select Device for Target**, что показано на рис. 12. Для нового проекта запрашивается автоматически.

Тип микроконтроллера очень важен для линковки. Тип описывает параметры, которые чаще всего придётся вносить в соответствующие файлы конфигурации в проекте.

Среда содержит базу данных руководящих технических материалов для предлагаемых типов МК. Доступ к базе осуществляется через окно проекта, закладку **Books**. Эту базу данных можно пополнять и самому для нужных МК.

Теперь можно наполнять проект содержимым. Используется меню **Project – Targets, Groups, Files …** , которое даёт диалог, приведенный на рис 13. Этот диалог можно получить и в локальном меню окна проекта на закладке **Files**, предварительно выделив нужную тему. Так наполняется проект из готовых файлов.

Если готового файла нет, его следует создать посредством меню **File – New**. В рабочей части окна откроется окно для текстового ввода. Набрав программу, её сохраняют с меню **File – Save As …** .

Диалог **Targets, Groups, Files …** позволяет создать в одном проекте несколько целей, в каждой цели несколько групп определённой принадлежности, а в каждой группе несколько файлов. Цели впоследствии можно добавлять и удалять; тоже самое можно делать и с группами и с файлами.

Каждая тема в проекте на своём уровне имеет свой набор управляющих опций, которые управляют всеми инструментами при создании целевых файлов проекта.

Окно целей, групп и файлов

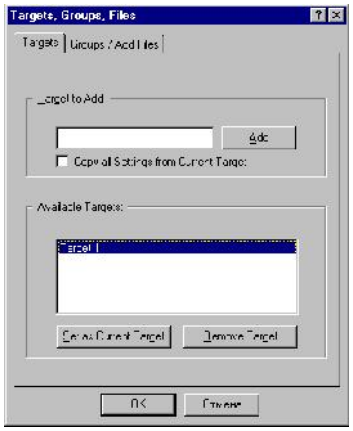


Рис. 13.

Диалог опций цели 1

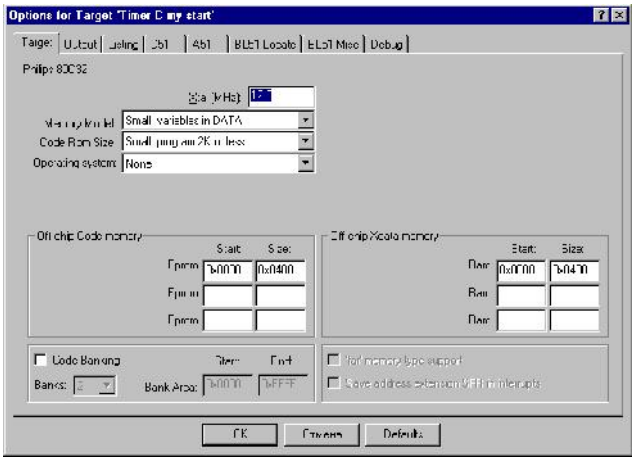


Рис. 14.

Для выходного файла задаётся имя файла и его каталог при необходимости. В выходной файл можно включить информацию для отладки и для базы имён программы. Кроме того, оговаривается создание файла для программатора. Файлы можно также включить в собственную библиотеку.

Здесь также задаются действия, выполняемые средой после построения проектного файла. Это может быть старт отладки. Кроме того можно задать две другие программы.

Закладка **Listing** управляет листингами компиляторов, задействованных на данном уровне тем проекта или даже исходного файла.

Наполнив проект содержимым, необходимо установить соответствующие опции. Диалог установки опций меняется от уровня темы и самый полный для уровня темы цели. Меню **Options for Target** … открывает этот диалог, который приведен на рис. 14.

Закладка **Target** задаёт частоту тактирования МК, модель памяти, конфигурацию внутренних блоков МК, конфигурацию внешней памяти и модель банков памяти для огромных многофазовфх (оверлейных) программ, если необходимо.

Рис. 15 показывает вторую закладку **Output**, которая определяет выходные файлы для цели проекта.

Диалог опций цели 1

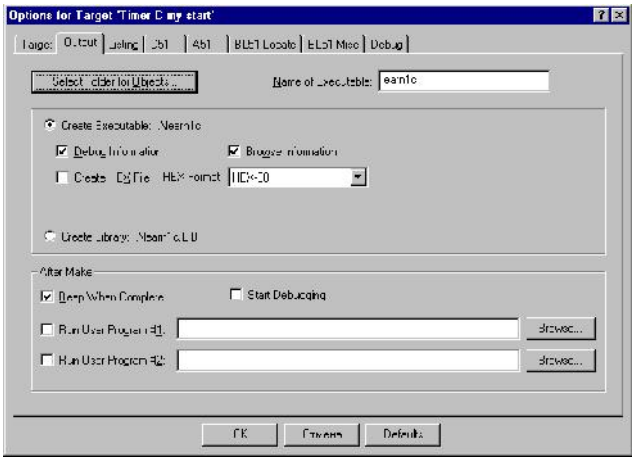


Рис. 15.

Опции компилятора С

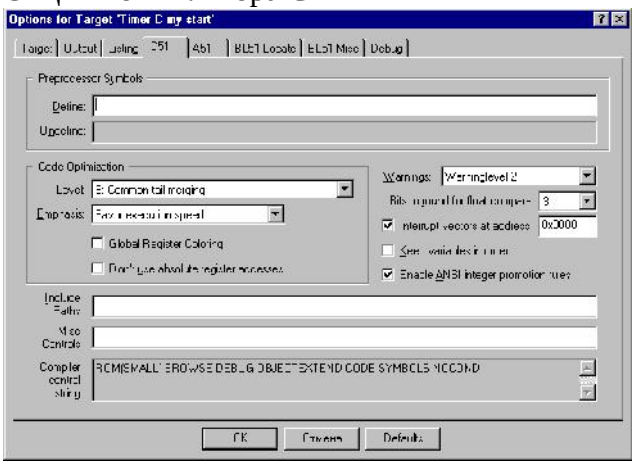


Рис. 16.

Основные опции для компиляторов задаются на своих закладках, которые приведены на рис. 16 и 17.

Опции Ассемблера

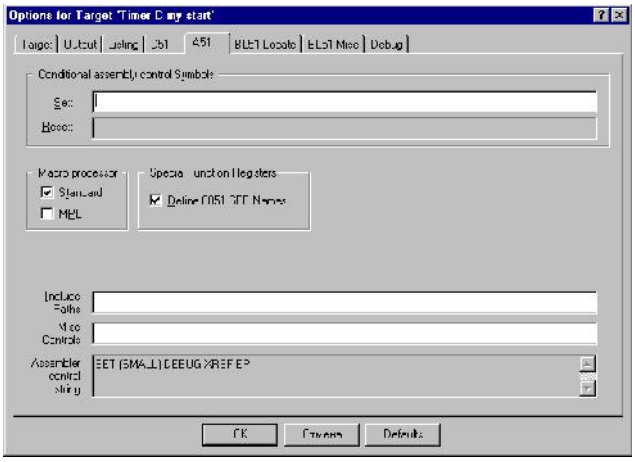


Рис. 17.

5.3. Введение в ProView.

ProView фирмы Franklin Software Inc. (Keil Software в США и Канаде) – интегрированная среда разработки программного обеспечения для однокристальных микроконтроллеров семейства Intel 8051 и его клонов.

На простом примере по методу “Быстрый старт” изучаются этапы технологии разработки и отладки программ,

основные приёмы работы на примере среды ProView.

Прежде чем начать разработку проекта, создайте свою личную папку. В этой папке находится всего лишь один файл с текстом данной программы – hello.c.

Запуск ProView и создание файла проекта.

ProView запускается из стартового меню Windows подобно остальным приложениям (рис. 18). Если необходимо запустить программу из командной строки, её синтаксис имеет вид: PV32 [projectfile], где projectfile – имя файла проекта с расширением [.PRJ].

Запуск программы

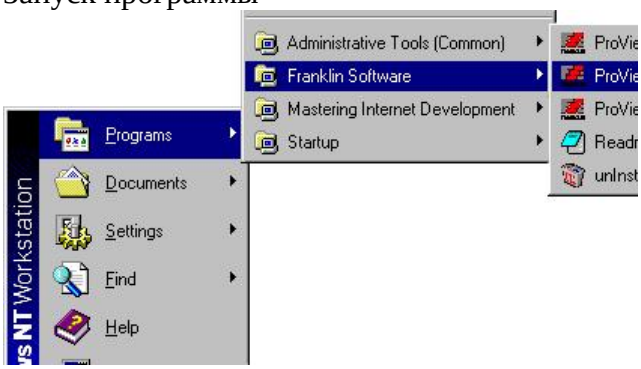


Рис. 18.

Диалоговое окно New Project

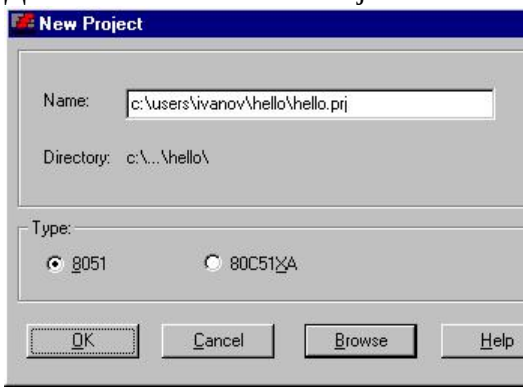


Рис. 19.

Любая новая работа в ProView, как и во всех современных компиляторах, начинается с создания нового файла проекта. Файл проекта содержит имена всех исходных файлов, связанных с проектом, а также установки компиляции, трансляции и связывания файлов, чтобы генерировать выполняемую программу.

Для того чтобы создать новый файл проекта, выберите New из меню Project. Откроется диалоговое окно New Project (рис. 19). Используйте кнопку Browse, чтобы войти в свою папку. Найдите свою папку и нажмите кнопку [OK]. Затем выберите “8051” как тип проекта.

Когда менеджер проекта открывает файл проекта, окно проекта показывает включенные исходные файлы. В данном случае пока нет никаких исходных файлов. Имеется только один исходный файл, который необходимо подключить – hello.c.

Добавка файла с исходным текстом и его редактирование

Диалоговое окно Add File

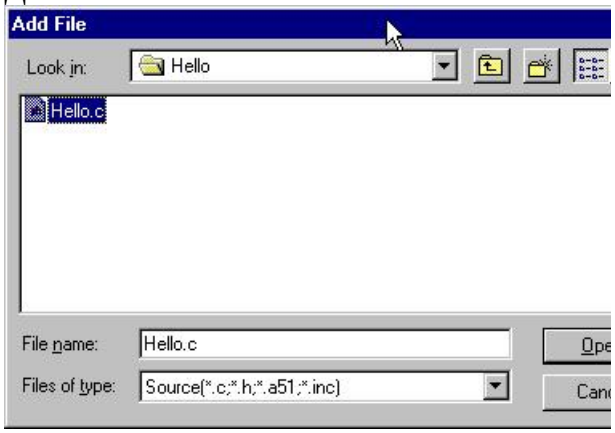


Рис. 20.

Теперь можно добавить hello.c к проекту. Выберите Add file из меню Project. Откроется диалоговое окно Add File (рис. 20). Выберите hello.c из списка.

Наш проект имеет только один исходный файл. В дальнейшем Ваши проекты, возможно, будут состоять из множества исходных файлов. Диалог Add File позволит Вам выбрать и добавить несколько файлов сразу. Для этого используют комбинацию клавиши [CTRL] и указателя мыши. Когда Вы нажмёте [Open],

исходные файлы будут добавлены к проекту в выбранном порядке.

Диалоговое окно Project

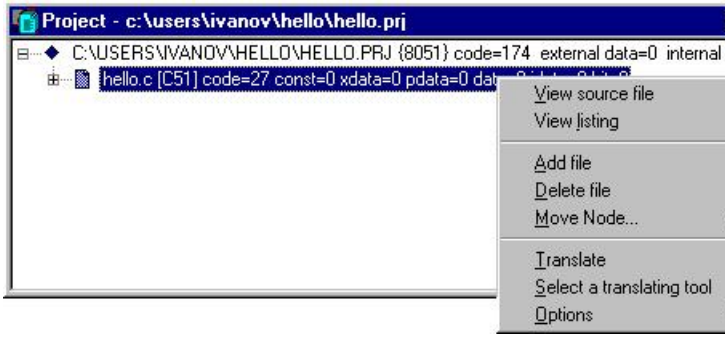
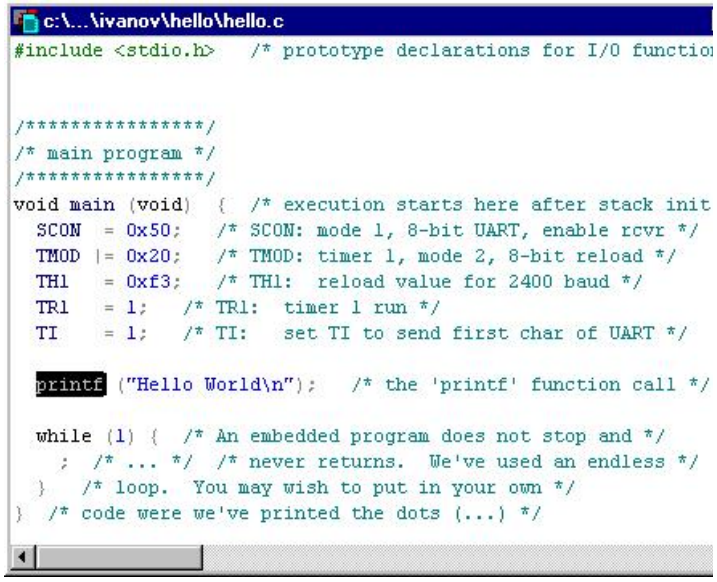


Рис. 21.

Теперь можно редактировать текст из файла hello.c. Выберите hello.c из окна Project (рис. 21). Нажмите его правой кнопкой мыши и выберите View source file, или просто дважды щёлкните мышью для того, чтобы просматривать файл в окне редактирования.

Окно редактирования



ProView загружает и показывает содержание hello.c в окне, где можно редактировать файл. Окно редактирования (рис. 22) – полнофункциональный редактор исходного текста, предлагающий такие возможности, как высвечивание синтаксических элементов и контекстный поиск. Если выбрать “printf” и нажать клавишу [F1], ProView откроет систему справки и перейдёт к разделу справки о “printf”.

Рис. 22.

Компиляция и компоновка.

Этот процесс компилирует, связывает hello.c с библиотеками и создает абсолютный объектный модуль, который мы сможем проверить в отладчике WinSim.

Окно сообщений

Рис. 23.

Выберите Make из меню Project. ProView отображает окно, показывая текущее состояние процесса. Когда процесс компиляции закончится, в окне Message (рис. 23) отображается

Рис. 1. Окно сообщений

сообщение завершения. Если были обнаружены какие-нибудь ошибки, о них сообщается здесь же.

Тестирование и отладка.

Выполним отладку программы. Если проект новый, откроется диалоговое окно Debug Options (рис. 24), где Вы можете изменять установки отладчика. В дальнейшем можно установить опции отладчика, выбрав Debug из меню Options. Наш проект использует значения по умолчанию.

Выберите Start из меню Debug.

Окно диалога опций отладчика

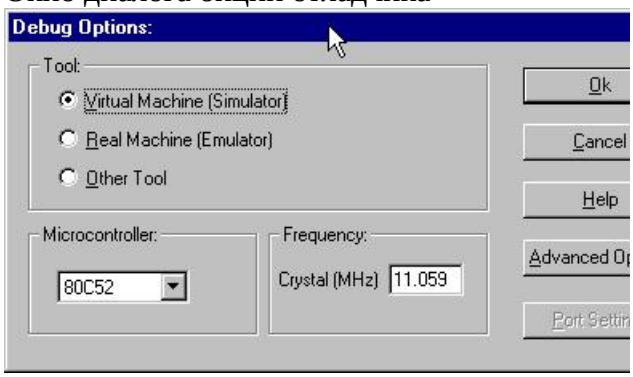


Рис. 24.

Окно последовательного порта

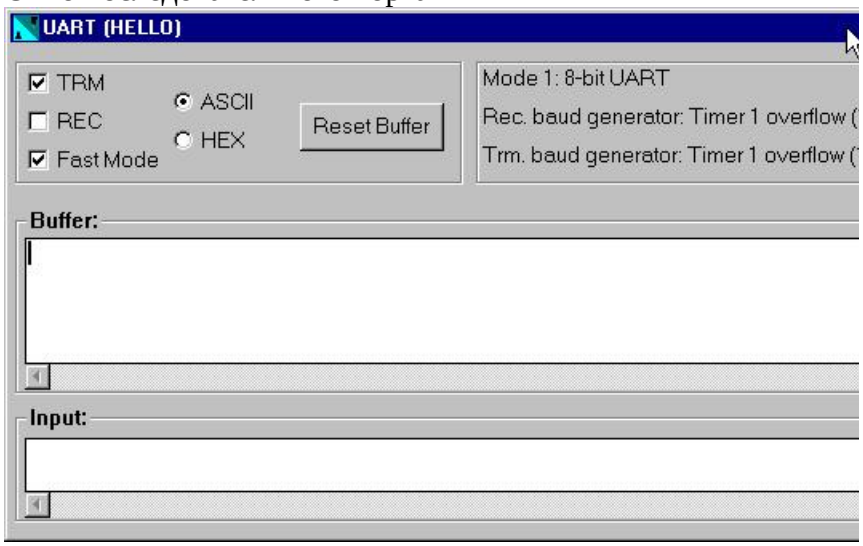


Рис. 25.

Выберите Hardware (аппаратные средства) из меню View. Выберите UART, откроется окно последовательного порта (рис. 25). В дальнейшем при работе программы здесь можно будет увидеть всё, что выводит микро-

контроллер в последовательный порт.

Выберите Run из меню Debug или нажмите кнопку

Run

Экран отладчика при выполнении программы

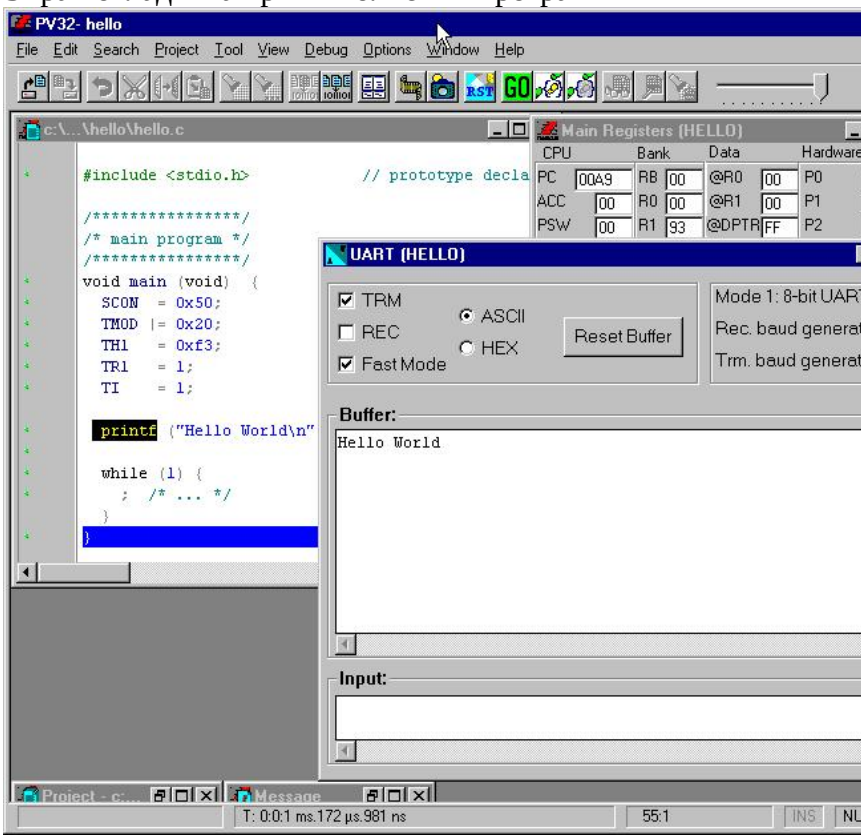


Рис. 26.

Рис. 26 показывает, как выглядит экран отладчика WinSim при выполнении программы. Обратите внимание, что в окно UART выведен текст “Hello World”.

При выводе символов в порт начинается выполнение бесконечного цикла. Вы можете остановить выполнение программы, выбрав Stop из меню Debug. С помощью регулятора

при нажатой кнопке

Animate

на панели инструментов можно менять скорость работы отладчика.
Строка состояния показывает текущее реальное время.

Пошаговый режим и выход из отладчика

Вы можете использовать отладчик, чтобы перемещаться по программе. Выберите Reset из меню Debug (эта команда сбросит моделируемый процессор) и выберите Step Into и Step Over из меню Debug.

Команды Step позволяют “шагать” по каждой строке исходного текста. Текущая команда высвечивается на каждом шаге. Step Into позволяет войти в вызываемую функцию, Step Over – перешагнуть через неё, не входя во внутрь.

Проделайте эти операции.

Для завершения работы с отладчиком в любой момент времени Вы можете выбрать Terminate из меню Debug и возвратиться в режим редактирования.

Окно кода

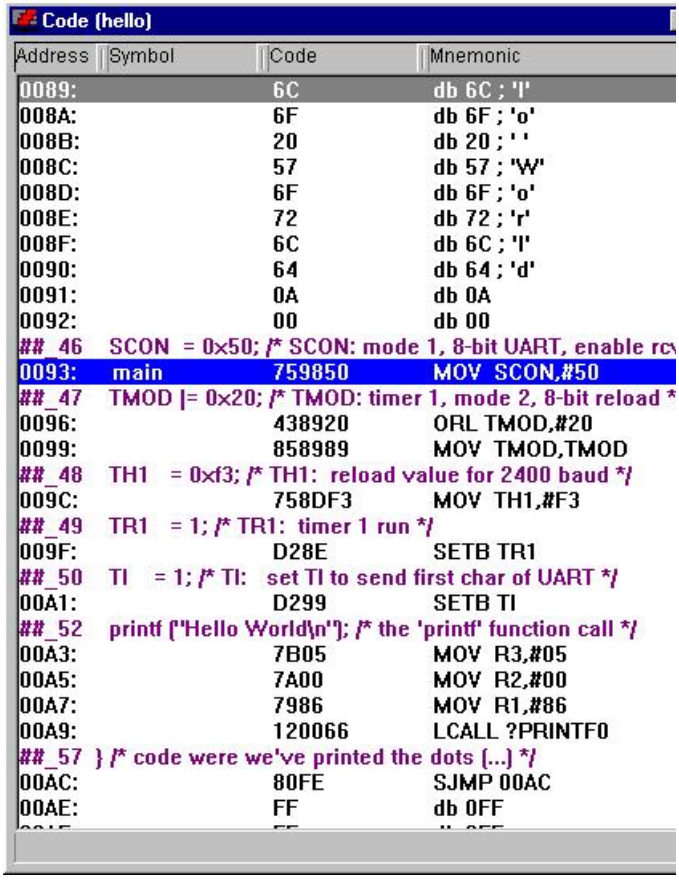


Рис. 27.

Следующий шаг

Обратите внимание, что в режиме отладки на экране

видны ещё два окна. Первое – окно кода (рис. 27), где в пошаговом режиме параллельно с исходным текстом на языке С идёт трассировка текста на ассемблере.

Прокрутите окно кода и изучите ассемблерный аналог исходного текста. С символов “##” начинаются строки, с помощью которых легко сопоставить ассемблерный текст и текст на языке С. Обратите внимание на то, сколько кода пришлось бы написать, если проектировать программу на ассемблере.

Диалог опций проекта

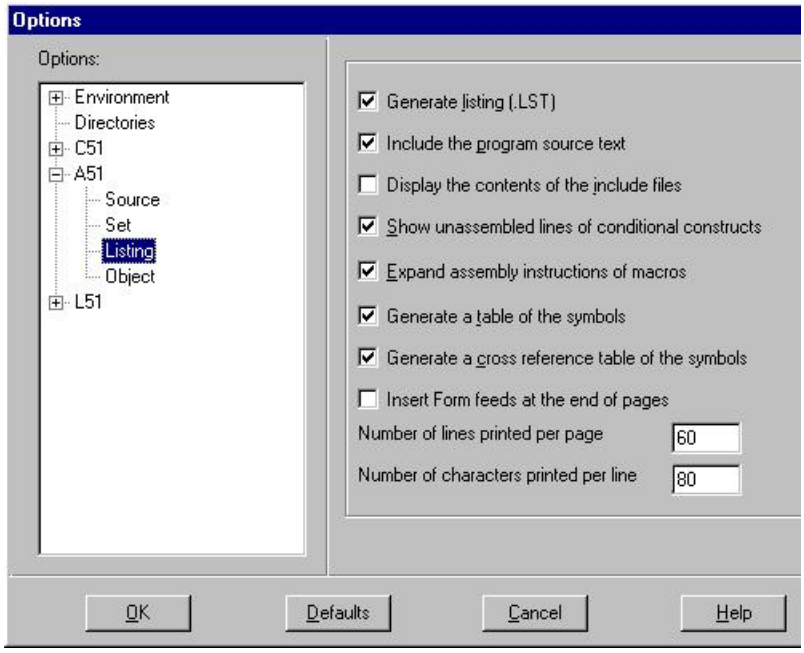


Рис. 28.

Ассемблерный аналог текста сохраняется в файле hello.lst, если в опциях проекта (Project из меню Options) отмечено Generate Listing (рис. 28). Здесь же можно указать, какую информацию включать в листинг.

Изучите смысл других опций проекта в разделах Environment, C51, A51, L51. Откройте файл листинга (рис. 29) с помощью View listing из меню View.

Изучите и постарайтесь понять содержание разделов файла листинга.

Второе окно, которое присутствует на экране во время отладки, – Main Registers (рис. 30).

В этом окне постоянно отображается текущее состояние всех программно-доступных регистров микроконтроллера. Более того, содержимое регистров можно менять во время отладки.

С помощью пункта Data dump из меню View можно посмотреть содержимое памяти различного типа в режиме отладки. Попробуйте это сделать.

Окно файла листинга

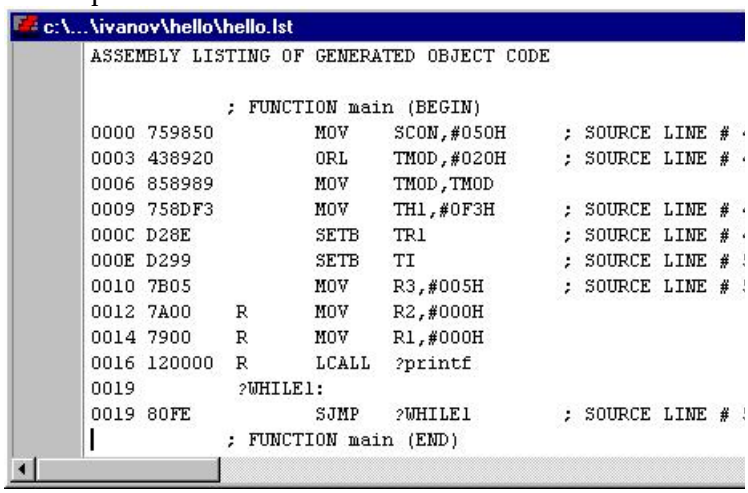


Рис. 29.

Окно регистров

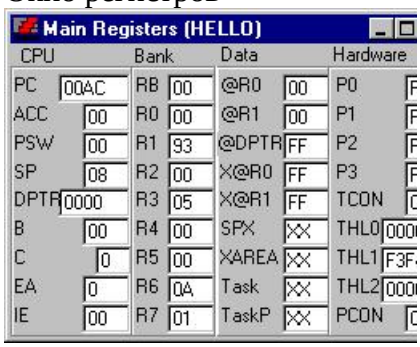


Рис. 30.

III ИНСТРУМЕНТАЛЬНЫЕ СРЕДСТВА ДЛЯ ПРОГРАММИРОВАНИЯ И ОТЛАДКИ МК-СИСТЕМ

1. Дополнительные аппаратные средства.

Программаторы ПЗУ предназначены для записи исполнительного кода программы в различные варианты ИС ПЗУ. Программные средства обслуживания программаторов имеют интуитивно понятный интерфейс взаимодействия с оператором и воспринимают несколько десятков форматов представления исполнительного файла.

2. Средства диагностики неисправностей.

В этом разделе рассматриваются наиболее характерные методы и классы приборов для диагностики неисправностей МК систем. Из всех видов приборов можно выделить пять основных классов: 1). – **логические анализаторы**; 2). – **сравнивающие тестеры**; 3). – **внутрисхемные эмуляторы**; 4). – **сигнатурные анализаторы**; 5). – **тестеры с управляемым пробником**. Эта классификация условна; для каждого класса характерно наличие различных модификаций методов и вариантов реализации.

Логические анализаторы.

Для них характерны следующие особенности: 1) – возможность наблюдения и обработки сигналов одновременно по нескольким каналам; 2) – нормирование входных сигналов по логическим уровням; 3) – возможность реализации различных вариантов запуска развертки; 4) – возможность регистрации предельно коротких по сравнению с периодом повторения сигнала импульсных помех; 5) – возможность сравнения полученных данных с опорными. Отдельные из перечисленных возможностей могут отсутствовать у конкретного логического анализатора.

Из всего многообразия логических анализаторов можно выделить **три группы**: [1] – анализаторы логических состояний; [2] – анализаторы логических временных диаграмм; [3] – микропроцессорные анализаторы. Такие анализаторы отличаются способами визуального наблюдения, числом входных каналов и степенью эффективности при проверке цифровых схем с обычной логикой и схем на основе микропроцессоров.

Для **анализаторов логических состояний** характерно двоичное представление данных. Они рассчитаны на работу с тактируемыми логическими схемами, в которых изменение логических состояний происходит по фронту синхросигнала, вырабатываемого схемой. Логический анализатор лишь отображает это изменение в двоичном представлении. Наиболее типичными представителями такого класса являются анализаторы логических состояний 1600A, 1607A фирмы Hewlett-Packard и т.п..

Логическими анализаторами невозможно обнаружить кратковременные импульсные помехи, разновременность прихода сигналов и перемещающиеся нарушения синхронизации. Этих недостатков лишены **анализаторы логических временных диаграмм**. В таких анализаторах стробирование данных при передаче в память осуществляется с более высокой частотой, чем тактовая частота проверяемой схемы, благодаря чему, можно получить более точную информацию о сигналах на входных шинах. Анализаторы особенно полезны для тестирования асинхронных схем. Примером таких приборов могут служить анализаторы 8200, 9100D, 920 фирмы Gould/Biomation.

Микропроцессорные анализаторы разрабатываются специально для работы с микропроцессорами и имеют большие возможности визуального отображения и большее число входных каналов. Некоторые из анализаторов имеют также диалоговые схемы, что позволяет оператору управлять работой микропроцессора и контролировать результаты. Примером для этого класса логических анализаторов являются модели 1610A, 1611A фирмы Hewlett-Packard, MPA-1 фирмы Motorola.

Поиск неисправностей с ипользованием логических анализаторов проводит оператор имеющий хорошее представление о работе проверяемого

устройства, умеющий интерпретировать достаточно большие массивы выходных данных и владеющий методами отыскания неисправностей. Автоматизированный поиск неисправностей не представляется возможным.

Сравнивающие тестеры.

Для сравнивающих тестеров характерно автоматическое сравнение сигналов исследуемой системы с опорными сигналами, снимаемыми с заведомо исправной системы, с выдачей результата типа "годен/не годен". К таким приборам относится цифровой испытательный осциллограф DTO-1 фирмы Gould/Biomation. Он объединяет фактически три прибора: сравнивающий тестер, логический анализатор и запоминающий осциллограф. Логический анализатор имеет один канал. Наличие запоминающего осциллографа позволяет записывать до восьми каналов путем последовательного перемещения пробника. Опорные сигналы исправной работы системы хранятся на магнитной ленте в кассетном запоминающем устройстве, и воспроизводятся в процессе испытаний. Прибор сравнивает два логических сигнала, отображает их на экране ЭЛТ и выдает результат "годен/не годен". Многофункциональность тестера DTO-1 делает его полезным при контроле различных систем.

Тестеры с управляемым пробником.

Эти приборы снимают вопрос высокой квалификации оператора. В них используется автоматизация измерений, характерная для больших испытательных систем, но реализованная в относительно портативном приборе. Примером явлется тестер PSP, разработанный фирмой Omnicomp и реализованный фирмой Gen Red как модель 2225. Тестер выполнен на базе микропроцессора и включает в себя средства для проверки "годен/не годен" и средства для автоматической диагностики. Пробник, управляемый микропроцессором, буквенно-цифровая индикация позволяют быстро изолировать неисправные компоненты даже малоопытному оператору. Для таких приборов характерно наличие программирования испытаний, высокоскоростной диагностической проверки печатных плат. В приборе, кроме программы испытаний, должны хранится "образ" схемы, т.е. описание соединений компонентов испытуемой платы, и таблица ожидаемых реакций для каждого узла. Тестеры с управляемым пробником значительно упрощают процесс поиска неисправностей. Однако такие приборы сложны и дорогостоящи, требуют больших программных затрат перед работой по диагностике.

В приведенных выше диагностических приборах сравнение длинных последовательностей в контролируемых точках с опорными последовательностями всегда выполняется автоматически либо оператором. Поиск

неисправностей связан со скоростью анализа этих последовательностей. Поиск неисправностей можно сделать более эффективным, если укоротить контролируемые последовательности. Для этой цели предлагается использовать метод переходного счета либо метод сигнатурного анализа ошибок.

В методе цифрового счета осуществляется подсчет числа переходов цифрового сигнала из одного логического состояния в другое. При подсчете переходов предполагается нулевое значение начального состояния счетчика. Счетчик производит суммирование по заданному модулю. Метод переходного счета при заданной длине контролируемой последовательности дает худшие вероятностные характеристики обнаружения ошибок в сравнении с методом сигнатурного анализа.

При одном ошибочном символе входной последовательности сигнатурный анализ обеспечивает 100% обнаружение ошибки. При нескольких ошибочных символах во входной последовательности метод обеспечивает 99,998% обнаружения ошибки.

Сигнатурные анализаторы.

Сигнатурные анализаторы – наиболее дешевые и простые в применении диагностические приборы, позволяющие выполнять высококачественную и быструю проверку цифровых устройств. Они способны определять ошибку во входной последовательности произвольной длины, независимо от положения ошибок во входной последовательности.

Сигнатурный анализатор преобразует длинные последовательности двоичных сигналов, поступающие от испытываемого изделия, в короткие четырехзначные шестнадцатиричные ключевые слова -"синатуры". Кодирование осуществляется по законам циклических кодов, исправляющих ошибки. Измеренные таким образом значения сигнатур оператор сравнивает с эталонными значениями. Если в двоичной последовательности выявляется ошибка, ее прослеживают по схеме обратным ходом, просматривая вентили и запоминающие элементы, пока не удается обнаружить элемент с правильными входными, но ошибочными выходными сигналами.

Вслед за фирмой Hewlett-Packard идеи сигнатурного анализа начали использовать фирмы Millenium Systems, Paratronics, Kurz-Kasch, Phoenix Digital, Zehntel и др..

Внутрисхемные эмуляторы.

Все рассмотренные выше классы диагностической аппаратуры являются пассивными приборами, т.е. приборы, которые только принимают сигналы от исследуемого устройства. Внутрисхемные эмуляторы являются активными диагностическими приборами. Их использование весьма эффективно при отработке и проверке микропроцессорных схем.

В эмуляторах применяется внешняя по отношению к проверяемому устройству система для имитации микропроцессора. Они обеспечивают диагностику отказов при функционировании в реальном времени. Использование сменных узлов позволяет легко адаптировать прибор к конкретному типу микропроцессора.

Внутрисхемные эмуляторы не позволяют диагностировать источники неисправностей с точностью до компонента, как в случае сигнатурного анализа, при этом для обслуживания требуется персонал высокой квалификации.

Фирма Millenium Systems объединила внутрисхемную эмуляцию и сигнатурный анализ в приборе mSA,что позволило создать чрезвычайно мощное средство для диагностики цифровых микропроцессорных систем. С помощью внутрисхемной эмуляции анализатор вырабатывает цифровые коды, благодаря чему имеется возможность проверять (диагностировать) фактически неработающие схемы. Единственно, что необходимо для осуществления диагностики – функционирование генератора тактовых импульсов проверяемой системы. Микропроцессор анализатора подключается с помощью кабеля эмуляции вместо микропроцессора проверяемой системы. Контрольный микропроцессор функционирует по программе испытаний, хранящейся в ПЗУ анализатора, в соответствии с программой проверяемой системы.

Сочетание сигнатурного анализа с внутрисхемной эмуляцией обеспечивает более широкие возможности, чем любой отдельно взятый метод. Однако такие анализаторы сложны и дорогостоящи.

ЛИТЕРАТУРА

1. *Коржуков П. П.* Программирование микропроцессорных систем: пособие для студентов спец. 1-31 04 02 «Радиофизика», 1-31 04 03 «Физическая электроника». – Мн.: БГУ, 2011. – 156 с.

2. *Сташин В.В., Урусов А.В., Мологонцева О.Ф.* Проектирование цифровых устройств на однокристальных микроконтроллерах / В.В. Сташин, А.В. Урусов, О.Ф. Мологонцева. – М.: Энергоатомиздат, 1990. – 224 с.
3. *Боборыкин А.В., Липовецкий Г.П., Литвинский Г.В. и др.* Однокристальные микроЭВМ / А.В. Боборыкин и др. – М.: МИКАП, 1994. – 400 с.

ЛАБОРАТОРНАЯ РАБОТА

Введение в программирование микроконтроллеров
(2 ч)

Часть 1

1. Изучите лекционный материал: изучите особенности однокристальных микроконтроллеров; ознакомьтесь с архитектурой микроконтроллеров семейства Intel 8051; ознакомьтесь с системой команд микроконтроллеров семейства Intel 8051.

2. На примере по методу “Быстрый старт” изучите этапы технологии разработки и отладки программ для МК семейства MCS-51. Освойте основные приёмы работы в интегрированной среде разработки ПО.

3. Определите временные интервалы в тактах и секундах выполнения основных блоков кода программы из раздела “Быстрый старт”. Меняйте тактовую частоту осциллятора МК.

4. Для углублённого изучения возможностей интегрированной среды разработки программного обеспечения и её компонентов самостоятельно изучите содержание и смысл всех пунктов меню, кнопок инструментальной панели, окон и настроек. Для этого воспользуйтесь встроенной справочной системой, которая вызывается через меню Help. Эти знания потребуются при выполнении следующих лабораторных работ.

Часть 2. Машинный язык

1. Реализовать преобразование 8-разрядного числа в знаковое слово (2-байта). Использовать свой месяц и день рождения. Положительное число – (7*М + Д). Отрицательное число – (Д – 30 – 7*М). Определите временные интервалы в тактах и секундах выполнения кода программы.

2. Те же числа преобразовать в знаковое двойное слово (4-байта) и определить временные интервалы.

3. Реализовать программу генератора периодического отклика. Скважность сигнала равна 2*(номер в списке группы). Определите временные

интервалы в тактах и секундах выполнения основных блоков кода программы. Оцените точность периода и длительность импульса.

4. Поварьируйте различными исполнениями программы генератора периодического отклика.

Дополнительная часть

Часть 3. Язык Ассемблера. (Начало лабораторных работ № 3, 4)

1. Реализовать программу генератора периодического отклика. Модульное программирование. Скважность сигнала равна 2*(номер в списке группы). Определите временные интервалы в тактах и секундах выполнения основных блоков кода программы. Оцените точность периода и длительность импульса.

2. Поварьируйте различными исполнениями программы генератора периодического отклика.

Часть 4. Язык Си. (Начало лабораторных работ № 2, 4)

1. Реализовать программу генератора периодического отклика. Модульное программирование. Скважность сигнала равна 2*(номер в списке группы). Определите временные интервалы в тактах и секундах выполнения основных блоков кода программы. Оцените точность периода и длительность импульса.

2. Поварьируйте различными исполнениями программы генератора периодического отклика.