Министерство образования Республики Беларусь
Учреждение образования
«Белорусский государственный университет
информатики и радиоэлектроники»

Кафедра электронных вычислительных машин

А.А. Петровский

МИКРОПРОЦЕССОРНЫЕ СРЕДСТВА И СИСТЕМЫ

Лабораторный практикум
для студентов специальности I-40 02 01
«Вычислительные машины, системы и сети»
всех форм обучения

Минск 2006

УДК 004.9 (075.8)
ББК 32.973 я 73
П 30

Рецензент:
доц. кафедры ЭВС БГУИР,
канд. тех. наук А.Е. Аношенко

Петровский А.А.

П 30 Микропроцессорные средства и системы: Лаб. практикум для студ. спец. I-40 02 01 «Вычислительные машины, системы и сети» всех форм обуч. / А.А. Петровский. – Мн.: БГУИР, 2006.–51с.: ил.

ISBN 985-444-968-8

Содержит описание использования 8-разрядного микроконтроллера MC68HC11 фирмы Motorola, алгоритмы программирования и стирания внутреннего репрограммируемого ПЗУ. Приводятся введение в использование современного аппаратного комплекса DSK5510 на базе 16-разрядного микропроцессора фирмы Texas Instruments совместно с интегрированной средой разработки CCS и способы снижения энергопотребления микропроцессором. Показаны примеры выполнения лабораторных работ.

УДК 004.9 (075.8)
ББК 32.973 я 73

ISBN 985-444-968-8

1. ЛАБОРАТОРНАЯ РАБОТА «ОСНОВЫ ПРОГРАММИРОВАНИЯ 8-РАЗРЯДНЫХ МИКРОПРОЦЕССОРОВ СЕМЕЙСТВА MC68HC11»

В данной лабораторной работе предлагается изучить комплекс программно-аппаратных средств, предназначенный для отладки программ и схемотехники микроконтроллеров семейства MC68HC11 фирмы «Motorola». Особенности структуры, программирования и функционирования представителя этого семейства - микроконтроллер MC68HC11A1FN. Структурная схема комплекса для отладки программ и схемотехнических устройств на базе МК семейства MC68HC11 приведена на рис. 1.1.

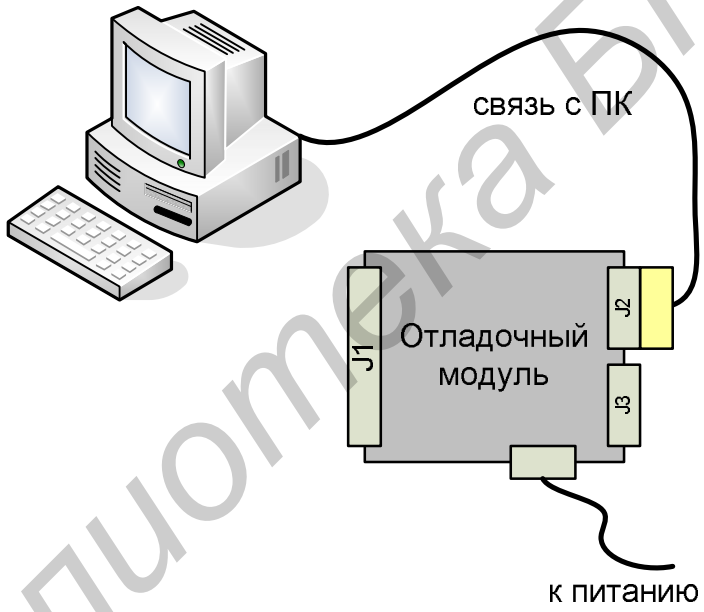


Рис. 1.1. Структура комплекса

Как видно из рис. 1.1, состав комплекса входят:

- терминал, в качестве которого будет выступать персональный компьютер, обеспечивающий ввод директив и данных, а также отображение сообщений и различного рода информации;
- учебно-отладочный модуль (ОМ), позволяющий загружать и выполнять программы в ЭВМ, выполнять различные операции с памятью и регистрами ЭВМ, а также выполнять многие другие функции, подробное описание которых приводится ниже;

- связь между терминалом и ОМ осуществляется по последовательному интерфейсу RS-232. Управление диалогом осуществляет специальная программа-монитор BUFFALO, записанная в ПЗУ ОМ, которая выводит на экран терминала сообщения, воспринимает директивы с клавиатуры терминала и производит необходимые действия.

Целью данной лабораторной работы является практическое ознакомление с перечисленными выше программно-аппаратными средствами.

1.1. Отладочный модуль M68HC11EVB

ОМ предназначен для отладки программного и аппаратного обеспечения устройств на базе ОЭВМ семейства MC68HC11 и позволяет выполнять следующие функции:

- записывать в память программы в мнемоническом виде с помощью встроенного ассемблера, а также загружать в память объектные коды программ, подготовленных на персональном компьютере, в формате S-records;
- просматривать и изменять содержимое памяти и регистров ОЭВМ;
- выполнять программы, находящиеся в памяти, как в реальном масштабе времени, так и по шагам команд или с использованием точек останова;
- программировать встроенное РПЗУ ОЭВМ, записывая туда отлаженную с помощью ОМ программу;
- осуществлять взаимодействие в диалоговом режиме с персональным компьютером типа IBM PC.

Внешний вид отладочного модуля и расположение переключателей и кнопок управления показан на рис. 1.2.

Как видно из рис. 1.2 на плате расположены два последовательных порта. Для их подключения служат коннектоы P2 и P3. Первый порт через разъем P2 подсоединен к термилалу и преобразует информацию, записываемую в него блоком ЭВМ, в последовательный код с уровнями сигналов по стандарту RS-232 и выдает, полученный код на терминал для отображения на экране. Таким способом на экране формируются сообщения. Приемная часть последовательного интерфейса производит обратные преобразования, в результате чего происходит считывание блоком ЭВМ кода нажатой на клавиатуре терминала клавиши и выполнение соответствующих действий. Второй порт последовательного интерфейса служит для подключения сервера и производит аналогичные действия.

Отладочный модуль может быть включен в разрабатываемую систему на этапе отладки вместо реальной ЭВМ. Для соединения ОМ и отлаживаемой системы служит порт расширения (разъем P1), дублирующий выводы ЭВМ.

Чтобы пользователь не ощущал действия эмуляции, на плате ОМ установлена специальная БИС, буферизующая этот порт.

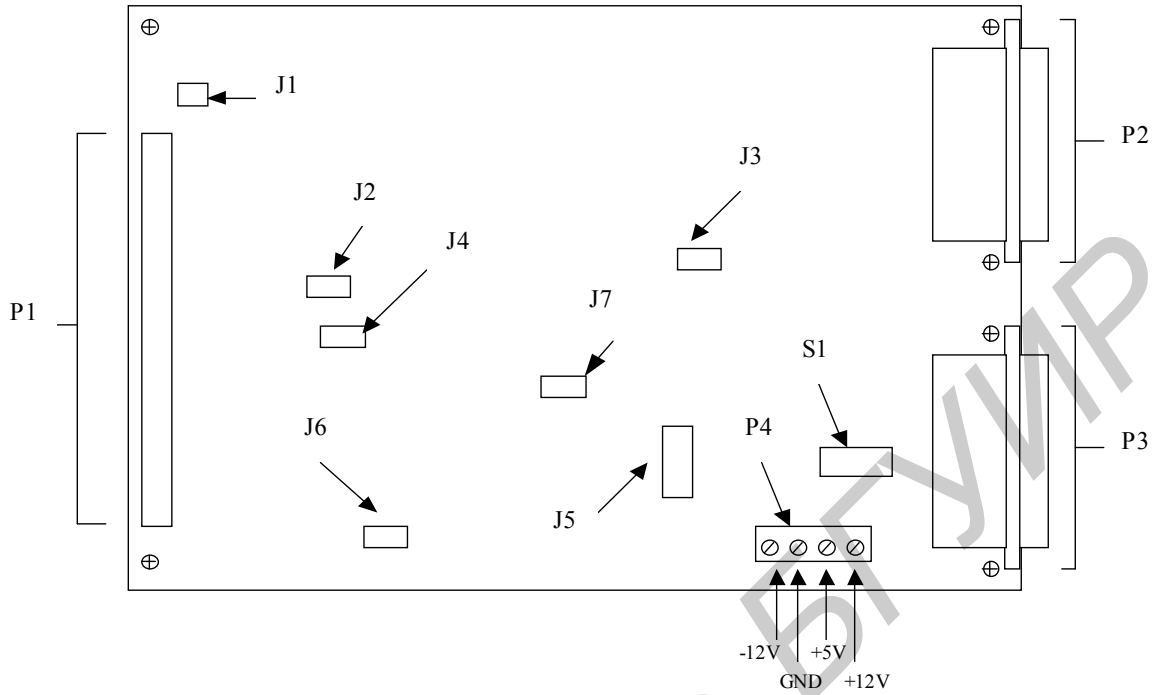


Рис. 1.2. Общий вид отладочного модуля:
P1 — разъем расширения дублирующий выводы ОЭВМ; P2 — разъем для подключения терминала; P3 — разъем для подключения сервера;P4 — разъем для подключения питания; J1 — выбор источника сигнала RESET; J2 — выбор источника синхронизации; J3 — установки памяти; J4 — включение или выключение командной строки монитора; J5 — установка скорости обмена через порт терминала; J6 — выключение сигнала RX для хост порта; J7 — используется производителем; S1 — кнопка сброса

1.2. Структура и режимы функционирования микроконтроллера

Микроконтроллер MC68HC11A8 (рис. 1.3) содержит:

- 8-разрядный процессор;
- внутреннюю память объемом 9152 байт, которая включает: ПЗУ емкостью 8 Кбайт; электрически перепрограммируемое ПЗУ емкостью 512 байт; служебное ПЗУ емкостью 192 байта; ОЗУ емкостью 256 байт;
- блок программирования (БПР);
- пять параллельных 8-разрядных портов A, B, C, D, E с блоком квитирования обмена (БКО) для портов B, C;
- синхронный и асинхронный последовательные порты (СПП, АПП), реализуемые в порту D;

- 16-разрядный таймер и блок счетчика импульсов (БСИ), включены в состав порта А;
- блок контроля функционирования (БКФ);
- генератор тактовых импульсов (ГТИ);
- 8-разрядный аналого-цифровой преобразователь (АЦП), совмещенный с портом Е.

Процессор выполняет обработку 8 и 16-разрядных операндов и реализует набор из 108 команд. Он содержит два 8-разрядных аккумулятора А и В, которые при выполнении ряда команд используются как 16-разрядный регистр D (A:B), два 16-разрядных индексных регистра X и Y, регистр условий CCR и 16-разрядные регистр-указатель стека SP и программный счетчик PC. В состав процессора входят также служебные регистры CONFIG, OPTION, HPRIO, INIT, TEST1, определяющие конфигурацию и режим работы микроконтроллера.

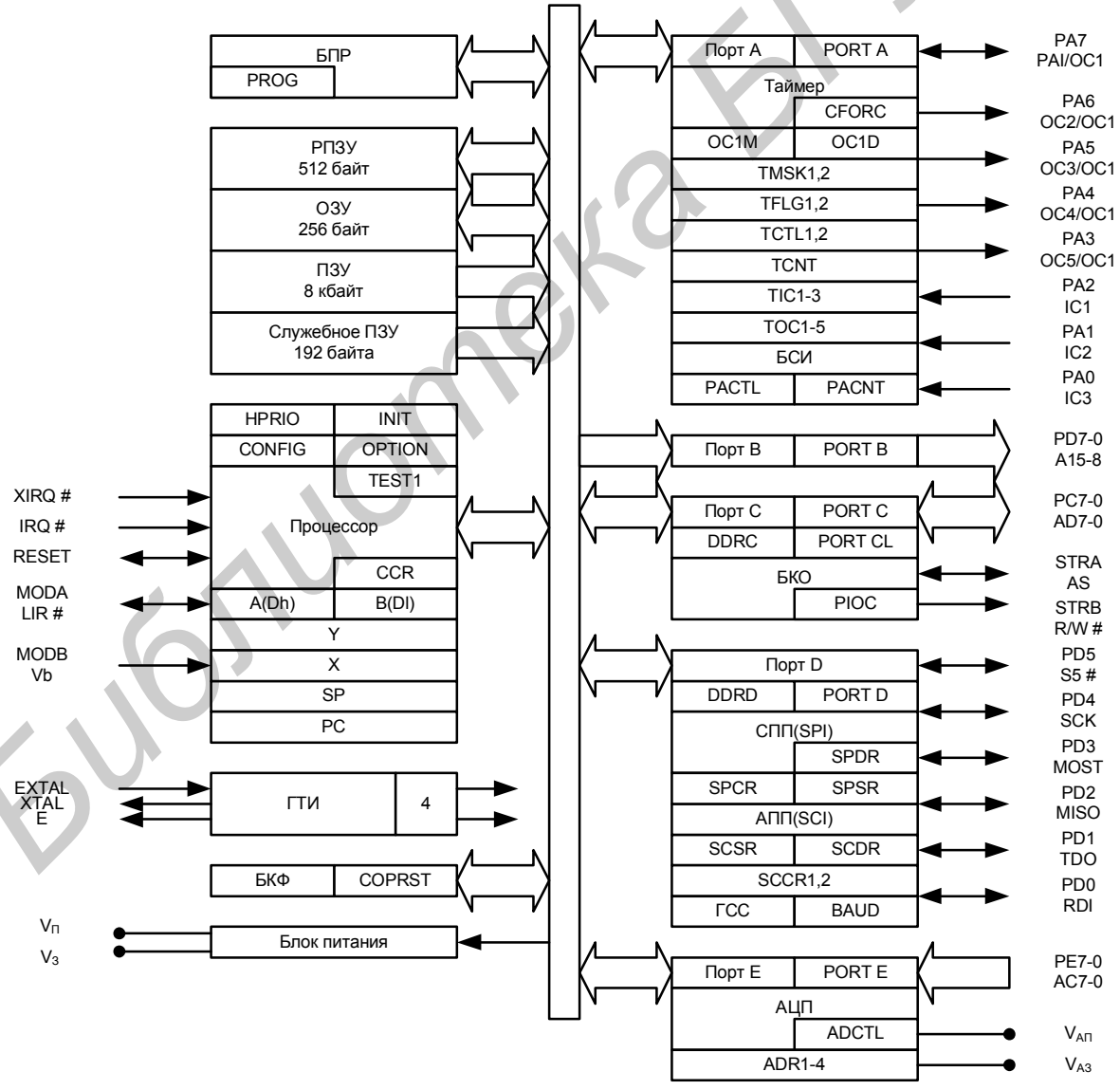


Рис. 1.3. Структура микроконтроллера MC68HC11A8

Микроконтроллер может использоваться в одном из четырех режимов: однокристальном, расширенном, загрузки или тестирования.

Режимы задаются значениями внешних сигналов, поступающих на входы MODB, MODA при начальной установке (табл. 1.1). Рабочие режимы обеспечивают функционирование микроконтроллера с использованием только внутренней памяти (однокристальный режим) или с подключением внешней памяти к портам B, C (расширенный режим). Специальные режимы реализуют загрузку внутреннего ОЗУ из внешнего источника через асинхронный последовательный порт (режим загрузки) под управлением специальной программы, выбираемой из служебного ПЗУ при обращении к адресам $BF40-$BFFF, или тестирование микроконтроллера, которое выполняется заводом-изготовителем (режим тестирования).

Таблица 1.1

Режимы использования микроконтроллера MC68HC11A8

Режимы использования	Внешние выводы		Содержимое регистра HPRIO				
	MODB	MODA	RBOOT	SMOD	MDA	IRV	PSEL 3-0
	Рабочие						
Однокристальный	1	0	0	0	0	0	0110
Расширенный	1	1	0	0	1	0	0110
	Специальные						
Загрузка	0	0	1	1	0	1	0110
Тестирование	0	1	0	1	1	1	0110

Значения сигналов на входах MODA, MODB воспринимаются микроконтроллером только в процессе начальной установки. При последующей работе микроконтроллера на вывод MODA выдается сигнал LIR# = 0 при выборке очередной команды программы, что позволяет осуществлять внешний контроль за работой микроконтроллера в процессе отладки системы.

Регистр HPRIO (адрес $103C)

7	6	5	4	3 0
RBOOT	SMOD	MDA	IRV	PSEL3-0

содержит биты PSEL3-0, которые определяют приоритет обслуживания запросов прерывания. Старшие биты в HPRIO имеют следующее назначение:

RBOOT — разрешение чтения служебного ПЗУ; при RBOOT = 1 разрешается выборка служебного ПЗУ при обращении к адресам $BF00-$BFFF

для выборки специальной программы, выполняющей загрузку ОЗУ из внешнего источника через последовательный порт АПП; при RBOOT = 0 выборка из этого ПЗУ запрещена, и данные адреса могут использоваться для обращения к внешней памяти;

SMOD, MDA — определяют режим использования микроконтроллера (см. табл. 1.1) после начальной установки;

IRV — разрешение выдачи данных на внешние выводы порта С при обращении к внутренней памяти, если IRV = 1; используется в режиме тестирования или отладки.

Регистра INIT (адрес $103D)

7 4	3 0
RAM3-0	REG3-0

определяет старшие четыре разряда адреса (номер страницы) размещения ОЗУ (биты RAM3-0) и блока внутренних регистров (биты REG3-0). При начальной установке микроконтроллера биты данного регистра принимают значения RAM3-0 = 0000 (обращение к странице 0), REG3-0 = 0001 (обращение к странице 1).

Регистр CONFIG (адрес $ 103F)

7	6	5	4	3	2	1	0
0	0	0	0	NOSEC	NOCOP	ROMON	EEON

определяет конфигурацию микроконтроллера, разрешая или запрещая использование внутренних ПЗУ, РПЗУ и блока контроля функционирования. Биты этого регистра имеют следующее назначение:

NOSEC — защита внутренней памяти от внешнего считывания; устанавливается при значении бита NOSEC = 0 и разрешает работу микроконтроллера только в однокристальном режиме или режиме загрузки, которые не обеспечивают выдачу содержимого внутренней памяти на внешние выводы; при NOSEC = 0 в регистре HPRIO устанавливается значение бита MDA = 0 независимо от сигнала на входе MODA в процессе начальной установки;

NOCOP — включение схемы контроля выполнения программы, входящей в состав БКФ, при значении бита NOCOP = 0;

ROMON — включение внутреннего ПЗУ при значении бита ROMON = 1; при ROMON = 0 можно использовать адреса $0000-$1FFF для обращения к внешней памяти;

EEON — включение внутреннего РПЗУ при значении бита EEON = 1; при EEON = 0 внутреннее РПЗУ отключается и соответствующее адресное пространство может использоваться для обращения к внутреннему ПЗУ или внешней памяти.

Регистр OPTION (адрес $1031)

7	6	5	4	3	2	1 0
ADPU	CSEL	IRQE	DLY	CME	0	CR 1-0

содержит биты, определяющие функционирование отдельных блоков микроконтроллера:

ADPU — включает питание аналого-цифрового преобразователя при установке ADPU = 1;

CSEL — разрешает при CSEL = 1 использование внутреннего RC-генератора для периодического заряда емкостей в АЦП и РПЗУ; при CSEL = 0 заряд емкостей производится тактовыми импульсами микроконтроллера;

IRQE — задает значение внешнего сигнала запроса прерывания IRQ#: при IRQE = 0 запросом является подача низкого уровня 0 на вход IRQ#, при IRQE = 1 запросом служит поступление отрицательного перепада потенциала на этот вход;

DLY — вводит при DLY = 1 значительную задержку начала функционирования микроконтроллера (около 4000 тактов) после выхода из режима останова, чтобы обеспечить установку заданной рабочей частоты ГТИ; при DLY = 0 задержка функционирования составляет всего четыре такта;

CME — разрешает при CME = 1 функционирование схемы контроля тактовой частоты микроконтроллера, входящей в состав БКФ;

CR1-0 — определяют коэффициент деления частоты при работе схемы контроля выполнения программы, входящей в состав БКФ.

Содержимое регистров HPRIO, INIT, OPTION записывается непосредственно после начальной установки микроконтроллера и затем может быть только считано. Исключение составляют только биты PSEL3-0 в регистре HPRIO и биты ADPU, CSEL в регистре OPTION, которые могут изменяться в процессе работы микроконтроллера путем записи в эти регистры. Содержимое регистра CONFIG программируется таким же образом, как и содержимое внутреннего РПЗУ, и может изменяться только в процессе репрограммирования. Так обеспечивается его сохранение при отключении питания.

1.3. Память микроконтроллера

Микроконтроллер адресует до 64 Кбайт памяти (адреса $0000-$FFFF), которые делятся на 16 страниц по 4 Кбайт. Полный объем памяти реализуется в расширенном режиме использования микроконтроллера, когда к выводам портов B, C подключается внешняя память. В однокристальном режиме используется только внутренняя память микроконтроллера: ПЗУ, РПЗУ, ОЗУ, а порты B, C служат для обмена данными с внешними устройствами. Две

последние страницы (адреса $E000-$FFFF) занимают ячейки внутреннего ПЗУ, содержимое которого программируется в процессе изготовления микроконтроллера (масочное ПЗУ) по заказу пользователя. Репрограммируемое ПЗУ, электрически стираемое и записываемое с помощью блока программирования (БПР), размещается в адресном пространстве $B600-$B7FF .

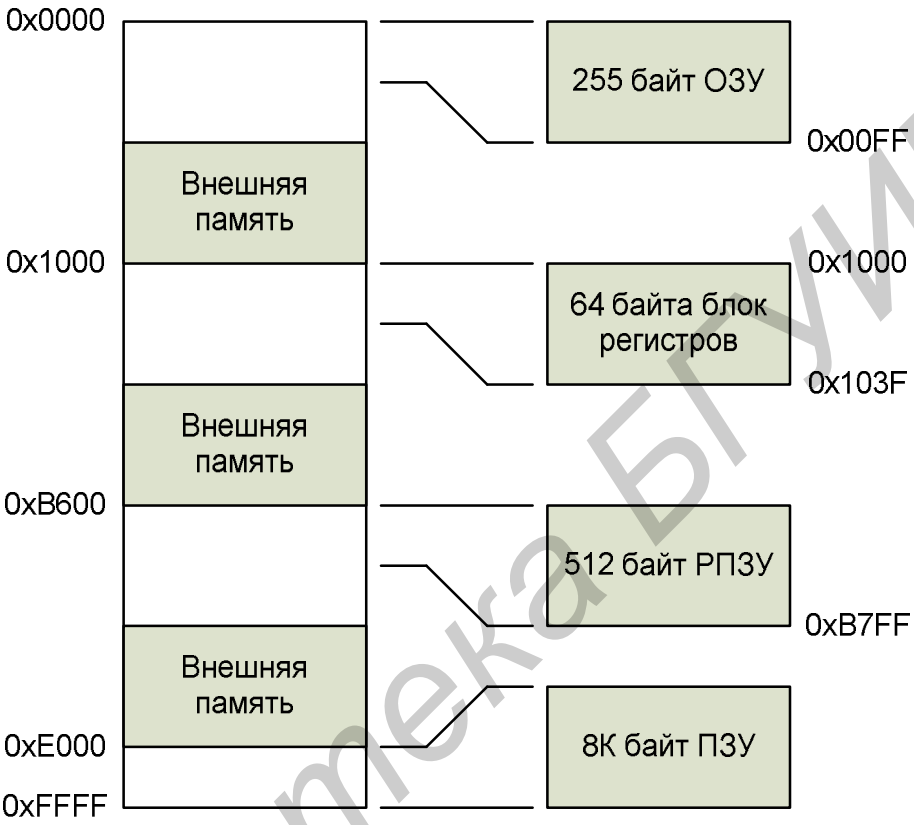


Рис. 1.4. Карта распределения адресного пространства.

Оперативная память (ОЗУ) при начальной установке микроконтроллера (RESET) занимает адресное пространство $0000-$00FF. Для обращения к внутренним регистрам микроконтроллера выделено 96 адресов, которые при начальной установке (процедура RESET) располагаются в позициях $1000-$105F (начало страницы 1). В число этих регистров, кроме служебных регистров процессора, входят регистры (рис. 1.1): параллельных портов PORTA, PORTB, PORTC, PORTCL, DDRC, PORTD, DDRD, PORTE, последовательных портов SPDR, SPCR, SPSR, SCDR, SCCR1, SCCR2, SCSR, BRR, таймера TCNT, TIC1-3, TOC1-5, TMSK1, TMSK2, TFLG1, TFLG2, TCTL1, TCTL2, OCIM, OCID, CFORC (16-разрядные регистры TCNT, TIC, TOC занимают по две позиции адресного пространства), счетчика импульсов PACTL, PACNT, блока контроля функционирования COPRST, аналого-

цифрового преобразователя ADR1-4, ADCTL. Последующая загрузка в регистр INIT ($103D)

7 4	3 0
RAM3-0	REG3-0

младших битов REG3-0 позволяет перемещать этот регистровый блок в начало любой страницы адресуемой памяти. При обращении к данным адресам производится выборка соответствующего регистра, а не размещенных на этой странице ячеек ОЗУ, ПЗУ или внешней памяти.

1.4. Способы адресации и система команд

Процессор микроконтроллера выполняет набор операций над 8- или 16-разрядными операндами, размещенными в регистрах и памяти. Команды имеют длину от 1 до 4 байтов: первый байт содержит код операции, перед ним может идти префиксный байт, указывающий на обращение к одному из индексных регистров X или Y (имеет значения $18, $1A или $CD), второй и третий байты обеспечивают адресацию операнда. Два варианта команд условных ветвлений по значению бита BCLR, BSET с индексной адресацией, использующие регистр X, содержат пять байтов.

Для выборки операндов используются те же способы адресации, что и в микроконтроллерах семейств M68HC05, M68HC08:

- регистровая (операнд располагается в регистрах A, B или X, Y);

косвенно-регистровая (адресом операнда служит содержимое регистра X или Y);

- индексная (адрес операнда образуется сложением содержимого регистра X или Y и заданного во втором байте команды 8-разрядного смещения, которое является числом без знака);
- прямая (8- или 16-разрядный адрес операнда задается во втором и третьем байтах команды);
- непосредственная (8- или 16-разрядный операнд содержится во втором и третьем байтах команды);
- относительная (используется только в командах ветвления, адрес команды образуется сложением текущего содержимого PC и заданного во втором байте команды 8-разрядного смещения, которое является числом со знаком).

В большинстве команд, использующих прямую адресацию, может задаваться как 8-, так и 16-разрядный адрес. Однако в командах битовых операций BCLR, BSET, BRCLR, BRSET используются только 8-разрядные адреса (короткая адресация в диапазоне $0000-$FFFF).

Система команд представлена следующими группами:

1) команды пересылки данных, связанные с аккумуляторами (LDAB (load accum. B), LDD, STAB, TAB (transfer A to B), CLRA (Clear A), PSHA (push A to stack), PULA,...),

2) команды пересылки для стека и индексных регистров (PSHX, TSX (transfer SP to X),...),

3) команды переходов (JMP, JSR (jump to subroutine), RTS (return from subroutine), переходы по условиям и состояниям битов,...),

4) арифметические команды (ADD, SUB, INC, DEC, MUL, DIV, CMP,...),

5) логические команды (AND, OR, EOR, COM,...)

6) команды работы с битами (установка, сброс, проверка, сравнение, сдвиги,...),

7) специальные команды (STOP, WAI (wait for interrupt), SWI (software interrupt),...).

При выполнении команд пересылки, арифметических и логических операций, сдвигов, битовых операций происходит изменение значений определенных признаков в регистре условий CCR. Регистр CCR содержит значения признаков переноса C, переполнения V, нулевого результата Z, знака N, запрещения прерывания I, переноса с заемом H, признак X запрещение обслуживание внешнего запроса прерывания, поступающего на вход XIRQ#, и S запрещение перевода микроконтроллера в режим останова по команде STOP.

Формат содержимого регистра условий CCR

7	6	5	4	3	2	1	0
S	X	H	I	N	Z	V	C

Изменение всех признаков происходит при загрузке регистра CCR командами TAP, RTI. Изменение признака I осуществляется командами CLI, SEI; кроме того, значение I = 1 устанавливается при выполнении команды прерывания SWI.

1.5. Встроенный монитор

После включения питания и нажатия кнопки S1 (RESET) см. рис. 1.2 на экране терминала отображается следующее сообщение:

```
BUFFALO X.X (ext)
>
```

где X.X версия программы монитора, (ext) — если программа монитора расположена во внешней по отношению к ОЭВМ ППЗУ и (int) — если монитор содержится в пределах ЭППЗУ внутри микроконтроллера.

Выдача приглашения на экран компьютера свидетельствует о том, что отладочный модуль готов к работе и ожидает ввода команды с терминала пользователя. Командная строка должна иметь следующий формат:

```
><команда> [<параметры>]<CR>
```

здесь:

<команда> - мнемоническое обозначение команды (одна буква для большинства команд),

<параметры> - выражение или адрес.

1.6. Команды монитора

Опишем основные команды монитора, которые понадобятся для выполнения практической части данной лабораторной работы.
Ассемблер/дизассемблер.

```
ASM [<адрес>]
```

где <адрес> представляет собой адрес начала ассемблирования/дизассемблирования программы. Если адрес не указан, то по умолчанию ссылается на внутреннее ОЗУ.

Команда ASM позволяет пользователю вводить в память ОМ программу в виде мнемонических команд. После набора каждой команды и ввода ее нажатием клавиши (RETURN) происходит преобразование мнемоники в машинный код с записью его в память по указанному в команде ASM адресу. После ассемблирования команды указатель переводится на следующую свободную ячейку памяти с приглашением к вводу следующей команды. Таким образом, строка за строкой, вводится вся программа. Платой за простоту работы с таким ассемблером является ряд существенных ограничений. Поскольку ассемблер транслирует одновременно только одну команду, он, очевидно, не может вычислить адрес, скажем, десятой команды, следующей за вводимой. Как следствие, использование меток невозможно. Для вычисления адреса перехода, например, в команде вызова подпрограммы приходится вводить в команде вызова любой "фиктивный" адрес, вводить всю программу, после чего, когда адрес перехода становится известен, возвращаться к редактированию команды вызова и устанавливать в ней правильный адрес.
Правила синтаксиса для ассемблера следующие:

Все вводимые числа воспринимаются как шестнадцатеричные, поэтому никакие основные указатели (например, $ = шестнадцатеричный, % = двоичный, и т.д.) не разрешаются.

Операнды должны быть отделены одним или большее количество пробелов или символов табуляции.

Любые символы после правильно введенной мнемоники и связанной с ней операндов, воспринимаются как комментарии и игнорируются.

Способы адресации обозначены следующим образом:

- Непосредственная адресация записывается, предшествуя адрес знакам # ;
- Индексная адресация обозначается запятой. Перед запятой должно присутствовать однобайтное относительное смещение (даже если оно нулевое), и после запятой должен следовать X или Y, в зависимости от используемого индексного регистра (например, LDAA 00, X).

Прямя и расширенная адресация определяется длиной адресного операнда (1 или 2 цифры определяют прямую, 3 или 4 цифры определяют расширенную).

Относительные смещения для команд перехода, вычисляются ассемблером. Поэтому после любой команды перехода должен следовать адрес перехода, а не относительное смещение к этому адресу.

Несмотря на описанные ограничения, команда ASM позволяет достаточно просто вводить небольшие программы (в частности, все программы настоящего лабораторного практикума могут быть введены таким образом).

Работа ассемблера завершается вводом символа комбинацией клавиш (CTRL) A (см. пример 1.1).

Пример 1.1.

```
>ASM C000<CR>
C000 STX $FFFF
>LDAA #55<CR>    Непосредственная адресация необходим
86 55    символ # перед операндом.
C002 STX $FFFF
>STAA C0<CR> Режим прямой адресации.
97 C0
```

Установка точки останова (BR - Breakpoint).

```
BR [-][<адрес>]…
```

где [-] – удаляет (очищает) все контрольные точки.

[< адрес >] ... – удаляемый адрес или несколько адресов из таблицы точек останова.

Команда BR устанавливает на указанный адрес точку останова. Во время выполнения программы пользователя остановка происходит непосредственно перед выполнением команды, адрес которой является точкой останова.

Нельзя помещать точку останова на команду программного прерывания, так как эту команду монитор использует для обработки точек останова. Можно установить максимум 4 точки останова. После установки точки останова на экран выводятся адреса всех установленных точек останова. В одной командной строке может быть введено несколько точек останова (см. пример 1.2).

Пример 1.2.

```
>BR C003<CR>                    Установить точку останова
                                по адресу
C003 0000 0000 0000             C003.
>
>BR C003 C005 C007 C009<CR>     Устанавливает четыре
                                контрольных
C003 C005 C007 C009             точки.
>
```

Запуск программы на выполнение (G - Go to).

```
G [<адрес>]
```

где [<адрес>] – начальный адрес, с которого начнется выполнение программы.

Команда G позволяет начать выполнение программы в режиме реального времени с заданного пользователем адреса. Когда в команде G адрес не указывается, программа выполняется с текущего адреса програмного счетчика. Выполнение программы продолжается до тех пор, пока не встретится точка останова или до нажатия кнопок S1 (RESET) на плате OM (см. пример 1.3).

Пример 1.3.

```
>G C000 <CR>
```

Загрузка (LOAD).

```
LOAD <команда загрузки с хоста> <T>
```

Здесь:

< команда загрузки с хоста > – загрузить S-records в EVB через порт хоста,

<T> – загрузить S-records в EVB через порт терминала.

Команда LOAD перемещает (загружает) объектные данные в формате S-records с центрального компьютера в EVB. Так как EVB монитор обрабатывает только корректные данные S-records, то он может прервать операцию загрузки. Если начальный адрес S-records указывает на неправильный блок памяти, то на терминал CRT выводится сообщение о неправильном адресе “error addr xxxx” (xxxx – неправильный адрес) (см. пример 1.4).

Пример 1.4.

```
>LOAD H
```

Отображение памяти (MD - Memory Display).

```
MD [<адрес1> [<адрес2>]]
```

здесь:

<адрес1> – начальный адрес;

[<адрес2>] – конечный адрес.

Команда MD используется для отображения содержимого памяти начиная с <адреса1> до <адреса2>, причем количество отображаемых байт кратно 16 (т.е. команда MD $0100 $0102, например, приведет к отображению ячеек $0100 - $010F).

1.7. Подготовка, загрузка и отладка программ

Приступим к выполнению практической части данной лабораторной работы. Во всех последующих лабораторных работах необходимо будет выполнить стандартную последовательность действий:

- вести программу в память отладочного модуля;
- ввести исходные данные перед выполнением программы;
- запустить программу на выполнение;
- проанализировать результаты работы программы.

Выполнение указанных операций можно выполнить с использованием команд отладчика BUFFALO. Однако менее трудоемко и более наглядно эти действия можно выполнить с помощью кросс-ассемблера HCW. Процедура работы с кросс-ассемблером следующая. В окне редактора вводится исходный текст программы. Если возникают ошибки в процессе создания программы, они сразу же отображаются на экране. После завершения редактирования необходимо открыть окно терминала нажатием кнопки Show terminal на панели

инструментов. Далее нажатием последовательности кнопок Select port, Configure port и Apply port settings необходимо настроить терминал на работу с отладочным модулем. После получения в окне терминала приглашения монитора BUFFALO нажатием кнопки Save code to device производиться загрузка программы в память ОМ. Если все предыдущие действия выполнены правильно и загрузка прошла успешно, то необходимо запустить программы на выполнение.

1.8. Пример построения программ

Рассмотрим пример построения программы для микроконтроллера, которая реализует математическое выражение (a+b)/(a-b).

Листинг программы:

```
org $c000
a equ 4
b equ 2
d equ 2
e equ 2

 ldx #str
 jsr $ffc7 .OUTSTR
 ldaa #a
 ldab #b
 mul #c
 idiv #e
 negb
 aba* a-b->a
 tab
 clra
 xgdx    * x<->d
 ldaa #a
 ldab #b
 aba* a+b->a
 tab* a->b
 clra    * 0->a
 idiv    * d/x->x, d%x->d
 xgdx    * x<->d
 Pshb
 pula
 psha
 jsr $ffb2 .OUTLHL
 pula
 jsr $ffb5 .OUTRHL
 xgdx
 ldx #str2
 jsr $ffc7 .OUTSTR
 pshb
 pula
 psha
 jsr $ffb2 .OUTLHL
 pula
 jsr $ffb5 .OUTRHL

 rts

str fcc "(a+b)/(a-b)= "
 db 4
str2 fcc "(a+b)%(a-b)= "
 db 4
```

1.9. Задание на лабораторную работу

1. Изучить структуру аппаратного комплекса программирования и отладки на базе микроконтроллера MC68HC11.

2. Изучить структуру и режимы функционирования микроконтроллера MC68HC11.

3. Написать программу на языке ассемблер для микроконтроллера MC68HC11, которая реализует математическое выражение.

Перечень математических выражений, рекомендуемых для реализации на микроконтроллере MC68HC11:

1. $Y = (a^2 + b^2) / (a – b)$

2. $Y = (a^3 + b) / (b – 2) – (a + c)$

3. $Y = (a + c) / (b^2 – a) – (a + c)$

4. $Y = ((a – b) * c) / (a + c) + (b – c)$

5. $Y = ((a + c) * (c – 2)) / 4$

6. $Y = (a + b – a * c) / (a + c^2)$

7. $Y = (a – b) * (b – 2) – (a + c) / 24$

8. $Y = (a + b) / (2 – b) – (b – a)$

9. $Y = (a^3 + b^3) / (b^3 – 3) + (c – b * a)$

10. $Y = (a^3 + b) / (b – 2) – (a + c)$

11. $Y = (a^2 – b^3) / (a – c) * (c – a)$

12. $Y = (a + b) / (b^2 – c) – (a + c)$

Примечание: переменные a,b и c незнаковые числа в интервале от $0-$FF.

2. ЛАБОРАТОРНАЯ РАБОТА «ПРОГРАММИРОВАНИЕ/СТИРАНИЕ РЕПРОГРАММИРУЕМОГО ПЗУ МИКРОКОНТРОЛЛЕРА MC68HC11»

2.1. Введение

Микроконтроллер содержит репрограммируемое ПЗУ емкостью 512 байт с электрической записью и стиранием. Программирование такого РПЗУ пользователем не требует специального программатора и подключения источника повышенного напряжения. РПЗУ содержит 32 строки по 16 байтов и располагается в адресном пространстве $B600-$B7FF. Запись и стирание РПЗУ осуществляются программно с помощью блока программирования (БПР). При этом возможно стирание всего объема РПЗУ, стирание строки или стирание одного байта; запись выполняется только побайтно. Входящий в БПР регистр PPROG (адрес $103B)

7	6	5	4	3	2	1	0
ODD	EVEN	0	BYTE	ROW	ERASE	EELAT	EEPGM

содержит биты, определяющие режим работы РПЗУ:

ODD, EVEN — используются при тестировании РПЗУ на заводе-изготовителе (контрольная запись-считывание), при этом к выводу IRQ# микросхемы подключается повышенное напряжение+ 20 В;

BYTE, ROW — устанавливают режим стирания содержимого РПЗУ: при BYTE = 1 побайтное стирание, при BYTE = O, ROW = 1 стирание строки, при BYTE = 0, ROW = 0 стирание всего содержимого РПЗУ;

ERASE, EELAT — задают режим стирания при ERASE = 1, режим программирования при ERASE = 0, EELAT = 1, режим считывания при ERASE = 0, EELAT= 0;

EEPGM — подключает при EEPGM = 1 напряжение программирования-стирания к РПЗУ; этот бит может быть записан только после установки EELAT = 1.

Длительность такта машинного времени Tc = 1/Ft определяется генератором тактовых импульсов (ГТИ), частота следования которых Ft задается кварцевым резонатором, подключаемым к выводам EXTAL, XTAL, или внешними импульсами, подаваемыми на вход XTAL (см. рис. 1.2). При этом частота Ft в 4 раза меньше частоты внешнего резонатора или генератора Fg = 4Ft = 4/Tc. Импульсы с частотой Ft поступают на выход E

микроконтроллера для синхронизации работы других устройств системы. Максимальное значение Ft для данной модели составляет 4 МГц при номинальном напряжении питания Vп = 5 В. Ток, потребляемый от источника питания в рабочем режиме при Vп = 5 В, Ft = 4 МГц не превышает 50 мА .

2.2. Алгоритм стирания РПЗУ

Стирание РПЗУ выполняется следующим образом.

1. В регистр PPROG с помощью команды STA загружается содержимое, в котором биты ERASE = EELAT = 1, EEPGM = 0, а значение битов BYTE, ROW соответствует выбранному способу стирания.

2. С помощью команды STA выполняется запись в РПЗУ любого числа; при стирании байта запись производится по адресу данного байта, при стирании строки — по адресу любого байта данной строки, при стирании всего РПЗУ — по любому адресу РПЗУ.

3. Включается внутреннее напряжение стирания-программирования с помощью команды STA, выполняющей запись в регистр PPROG содержимого, в котором биты ERASE = EELAT = EEPGM = 1, а значения битов BYTE, ROW соответствуют выбранному способу стирания.

4. С помощью команды JSR вызывается подпрограмма, обеспечивающая задержку на время, необходимое для стирания (не менее 10 мс).

5. Устанавливается нулевое значение содержимого PPROG с помощью команды CLR (окончание процесса стирания).

Если выполняется стирание нескольких байтов или строк, то п.1 данной процедуры достаточно выполнить один раз. При этом в п.5 после стирания очередного байта или строки следует устанавливать только значение EEPGM = 0 (выключение напряжения программирования-стирания), после чего повторять п.2 (запись по новому адресу) и п.п.3-5. После стирания последнего байта (строки) необходимо перевести РПЗУ в режим считывания с помощью команды CLR, устанавливающей в 0 его содержимое.

2.3. Пример реализации алгоритма стирания

Листинг программы стирания ячеек РПЗУ

```
ORG $C000

REGBAS   EQU $1000 * Стартовый адрес блока регистров
TCNT     EQU $0E *Счетчик (16-бит)
TOC1     EQU $16 *OC1 регистр (16-бит)
```

```
TFLG1    EQU $23

     LDX #$b60F   *Адрес стираемого байта
     LDAB #%00010110  *Конфигурация регистра PPROG
     STAB $103B   *Программирования регистра PPROG
     STAB 0,X     *Запись произвольного байта по адресу
стираемого байта
     LDAB #$17
     STAB $103B   *Подача напряжения программирования
     JSR DLY10    *Вызов процедуры задержки на 10 мс
     CLR $103B    * Выключение высокого напряжения
программирования
     RTS

INITDELAY    LDX #REGBAS Указатель на блок регистров
             LDAA #$80
             STAA TFLG1,X
             RTS

DLY10    JSR INITDELAY    *Вызов процедуры инициализации
         LDD TCNT,X
         ADDD #20000
         STD TOC1,X
LP1      BRCLR TFLG1,X $80 LP1
         RTS
```

2.4. Программирование РПЗУ

Программирование РПЗУ осуществляется побайтно с помощью следующей процедуры.

1. В регистр PPROG с помощью команды STA загружается содержимое, в котором биты ERASE = EEPGM = O, EELAT = 1, биты BYTE, ROW имеют любые значения.

2. С помощью команды STA по адресу программируемого бита производится запись его значения, предварительно загруженного в аккумулятор.

3. Включается внутреннее напряжение программирования-стирания с помощью команды STA, устанавливающей в регистре PPROG бит EEPGM = 1 при сохранении значений ERASE = 0, EELAT = 1.

4. С помощью команды JSR вызывается подпрограмма задержки на время не менее 10 мс.

5. Устанавливается режим считывания РПЗУ (окончание программирования) с помощью команды CLR, устанавливающей нулевое содержимое регистра PPROG.

При программировании ряда байтов п. 1 процедуры может выполняться один раз, а затем для записи каждого байта повторяются п.п.2-5, где в п.5 устанавливается в 0 только бит EEPGM. После программирования последнего байта РПЗУ переводится в режим считывания установкой нулевого содержимого регистра PPROG.

2.5. Пример реализации алгоритма программирования

Листинг программы программирования ячейки РПЗУ

```
ORG $C000

REGBAS   EQU $1000 * Стартовый адрес блока регистров
TCNT     EQU $0E *Счетчик (16-бит)
TOC1     EQU $16 *OC1 регистр (16-бит)
TFLG1    EQU $23

     LDX #$b60F   *Адрес программируемого байта
     LDAB #%00010110  *Конфигурация регистра PPROG
     STAB $103B   *Программирования регистра PPROG
     LDAA #10 Записываемые данные
     STAA 0,X     *Запись байта по адресу программируемого
байта
     LDAB #$03
     STAB $103B   *Подача напряжения программирования
     JSR DLY10    *Вызов процедуры задержки на 10 мс
     CLR $103B    * Выключение высокого напряжения
программирования
     RTS

INITDELAY    LDX #REGBAS Указатель на блок регистров
             LDAA #$80
             STAA TFLG1,X
             RTS
```

```
DLY10    JSR INITDELAY    *Вызов процедуры инициализации
         LDD TCNT,X
         ADDD #20000
         STD TOC1,X
LP1      BRCLR TFLG1,X $80 LP1
         RTS
```

2.6. Задание на лабораторную работу

1. Изучить алгоритмы стирания и программирования внутреннего репрограммируемого ПЗУ микроконтроллера MC68HC11.

2. Запрограммировать алгоритмы стирания и программирования внутреннего репрограммируемого ПЗУ в различных вариантах (стирание строки – программирование строки байт; стирание всей памяти – программирование строки байт; стирание произвольного байта и его запись).

3. ЛАБОРАТОРНАЯ РАБОТА «ЗНАКОМСТВО СО СТАРТОВЫМ НАБОРОМ TMS320VC5510 DSK»

3.1. Состав, основные характеристики и возможности стартового набора TMS320VC5510 DSK.

Уровень развития инфраструктуры средств, обеспечивающих разработку цифровых систем, является одним из основных критериев выбора элементной базы при реализации современных специализированных микропроцессорных систем. Понимание такой ситуации заставляет ведущих фирм-изготовителей цифровых процессоров (ЦП) и их партнеров активно строить отладочные аппаратные комплексы и программные средства проектирования и отладки цифровых систем, которые создают условия для максимально быстрого, эффективного и комфортного решения прикладных задач. Структура такого комплекса для процессоров TMS320 компании Texas Instruments (www.ti.com), ориентированна по своим характеристикам на решение широкого круга технических задач в различных областях человеческой деятельности, представлена на рис. 3.1.

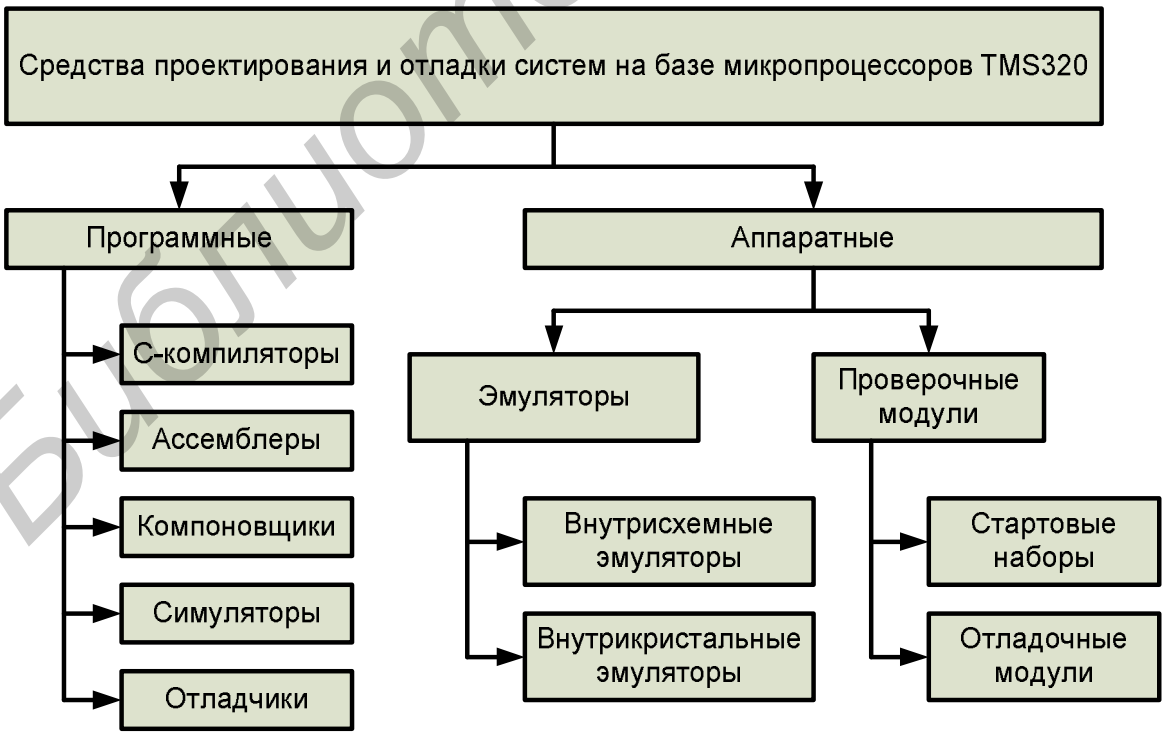


Рис. 3.1. Комплекс аппаратных и программных средств проектирования и отладки систем на базе ЦП компании Texas Instruments

Средства проектирования и отладки систем цифровой обработки сигналов на базе процессоров компании Texas Instruments

Стартовый набор TMS320VC5510 DSK (рис. 3.2) поставляется в следующей комплектации:

- модуль TMS320VC5510 DSK;
- переходник ≈220 В / =5 В;
- питающий кабель;
- компакт-диск с версией v.2.2 интегрированной среды разработки Code Composer Studio (CCS) для TMS320VC5510 DSK;
- кабель USB;
- техническое описание.

Рис. 3.2. Стартовый набор TMS320VC5510 DSK

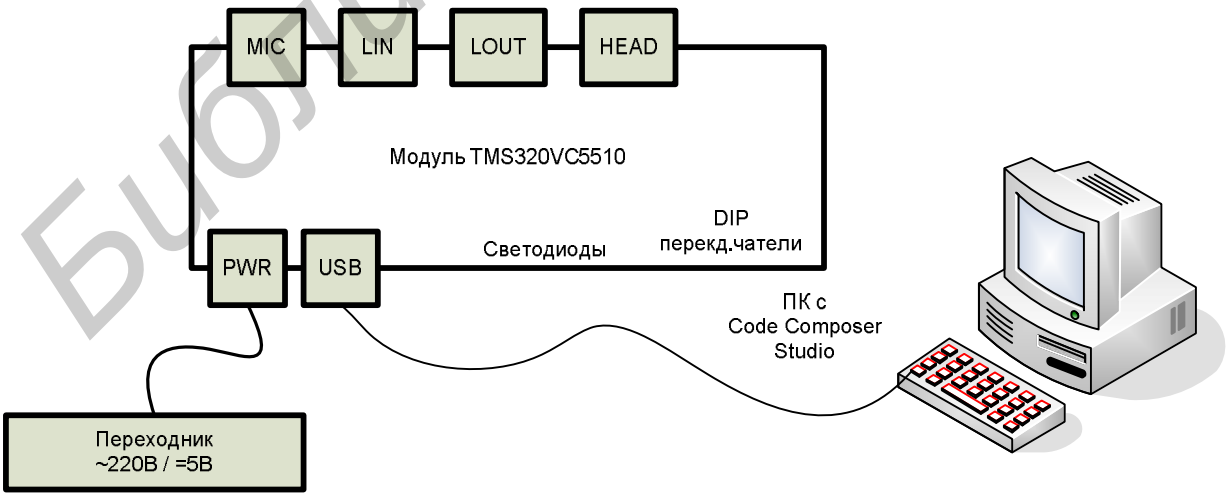


Рис. 3.3. Схема соединения модуля TMS320VC5510 DSK с персональным компьютером

Модуль TMS320VC5510 DSK (в дальнейшем − DSK) конструктивно представляет собой многослойную печатную плату размером 210×115 мм с размещенными на ней компонентами. Соединение DSK с персональным компьютером (ПК) выполняется кабелем USB через порт USB (J201), электропитание подается на разъем J6 через переходник ≈220 В / =5 В, обеспечивающий силу тока до 3 А (рис. 3.3). При отсутствии расширительных плат DSK потребляет 0,5 − 0,75 А.

Программно TMS320VC5510 DSK поддерживается сокращенной версией интегрированной среды разработки (IDE − Integrated Development Environment) Code Composer Studio v.2, функционирующей под операционными системами Windows 98SE / ME / 2000 (SP или выше) или XP. Рекомендуемая конфигурация ПК определяется тактовой частотой 500 МГц, наличием 600 Мбайт свободного пространства на жестком диске, 128 Мбайт оперативной памяти и видеоадаптера SVGA с разрешением 1024×768 пикселей.
Обобщенная структурная схема DSK представлена на рис. 3.4.

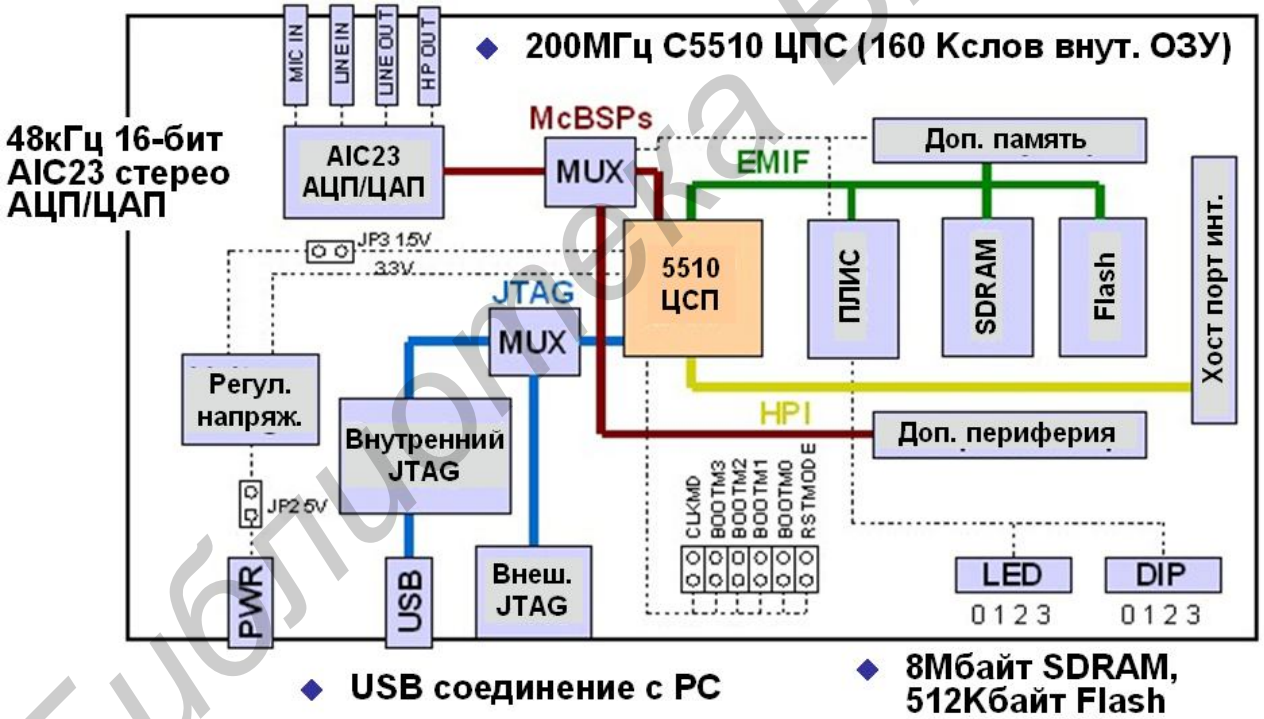


Рис. 3.4. Обобщенная структурная схема модуля TMS320VC5510 DSK

Основным элементом DSK является процессор платформы C5000 TMS320VC5510 − 200, наиболее популярной областью применения, которого выступает цифровая обработка речевых сигналов для систем мобильной связи. TMS320VC5510 − 200 представляет собой ЦП, использующий представление чисел в формате 16-битного слова с фиксированной точкой. Процессор характеризуется оптимальным для ресурсоемких мобильных приложений

соотношением высокой производительности, малого энергопотребления, малых габаритов и относительно низкой стоимости. Тактовая частота процессора составляет 200 МГц, пиковая производительность оценивается в 400 MIPS. Объем ПЗУ равен 16К слов (16К х 16 бит или 32К байт), ОЗУ − 320К байт. Питание ядра осуществляется напряжением =1,6 В, периферии − =3,3 В.

Система эмуляции DSK предусматривает два различных эмуляционных режима. Для взаимодействия с внешним эмулятором типа XDS − 510 на плате модуля установлен порт «Внешний JTAG», обеспечивающий сигнальный последовательный интерфейс JTAG (стандарт IEEE 1149.1). Возможность работы без внешнего эмулятора обеспечена USB JTAG контроллером «Внутренний JTAG», реализующим эмуляционную логику DSK. Коммутация обоих эмуляционных каналов с эмуляционным портом процессора выполнена посредством мультиплексора «JTAG Mux».

Система памяти DSK включает флэш-ПЗУ «Flash» емкостью 256 К слов и асинхронное статическое ОЗУ емкостью 64 К слов. Интерфейс расширения памяти (80-контактный порт «Дополнительная память») делает возможным использование внешних элементов памяти.

Основными элементами системы аналого-цифрового и цифро-аналогового преобразования DSK являются аудио интерфейс, кодек «AIC23 АЦП/ЦАП» с выходным стереофоническим аудио усилителем. Аудио интерфейс содержит линейные вход «Line In» и выход «Line Out», микрофонный вход «Mic In» и выход на динамик или головные телефоны «HP Out». При этом микрофонный вход используется в монофоническом режиме и имеет потенциометр регулировки уровня сигнала R33, остальные тракты используются в стереофоническом режиме. Частота дискретизации кодека выбирается пользователем из возможных значений 48, 24, 12, 8 и 6 КГц. К аналоговому выходу кодека подключен фильтр с частотой среза 30 КГц.

Интерфейс расширения периферии DSK представлен 80-контактным портом «Дополнительная периферия», соединенным с синхронными многоканальными буферизированными последовательными портами McBSP0 и McBSP1 процессора посредством буферного устройства. Хост-интерфейс DSK представлен 80-контактным портом «Хост порт интерфейс».

Основными элементами системы синхронизации являются генератор частоты 24 МГц, исходной для формирования сетки тактовых частот для ЦП и генератор формирования частоты 12,288 МГц, используемой для получения различных градаций частот дискретизации кодека.

Логическое устройство DSK «ПЛИС» реализовано на многократно программируемой логической интегральной схеме (ПЛИС) типа CPLD (Complex Programmable Logic Device), загрузка программы в которую осуществляется в процессе изготовления DSK. Логическое устройство

содержит восемь 8-битных регистров, назначение которых определяет основные функции устройства:

- контроль состояния и управление пользовательскими светодиодами и хранение информации о состоянии переключателей;
- идентификация, контроль и управление состоянием дочерней платы, подключаемой к DSK;
- формирование 16-битных команд управления кодеком;
- хранение 4-битового кода версии CPLD, записываемого в период компиляции от источника VHDL, и 3-битового кода версии модуля DSK, записываемого на этапе сборки стартового набора;
- осуществление программируемого взаимодействия интерфейса внешней памяти ЦП с ОЗУ, ПЗУ и интерфейсом расширения памяти DSK;
- коммутация синхронного многоканального буферизированного последовательного порта McBSP2 процессора со стереофоническим кодеком или хост-интерфейсом DSK через мультиплексор, определение варианта системы адресации дочерней платы, определение готовности кодека к работе;
- выбор рабочей частоты дискретизации (48, 24, 12, 8, 6 МГц) для кодека.

Система электропитания DSK содержит регулятор напряжения, понижающий поступающее напряжение = 5 В до значений =1,6 В и =3,3 В, требуемых для питания отдельных элементов DSK. Защитный элемент системы формирует команду перезагрузки DSK в случае выхода любой из градаций питающих напряжений из диапазона допустимых значений.

Имеющиеся в модуле переключатели и перемычки предоставляют возможность изменения режима функционирования стартового набора. Реализуемые функции: принудительная перезагрузка DSK, выбор режима функционирования ЦП ("микропроцессор − микроконтроллер") и режима формирования тактовой частоты ЦП с использованием системы фазовой автоподстройки частоты (PLL − Phase-Locked Loop), имеющейся на кристалле процессора.

Система индикации состоит из восьми светодиодов, отображающих состояние и режимы функционирования DSK. Пользовательские светодиоды используются тестом включения питания (POST − Power On Self Test) DSK и через адресное пространство доступны для пользовательских программ. Системные светодиоды отображают наличие питания на модуле, используемый режим эмуляции стартового набора, активацию режима перезагрузки и использование USB порта.

3.2. Особенности структуры внутреннего ядра микропроцессора TMS320VC5510

В основе микропроцессора лежит ядро цифрового сигнального процесса, структура которого функционально разделена на блоки: MU (Memory interface Unit) – блок интерфейса памяти, IU (Instruction buffer Unit) – блок буфера инструкций, PU (Program flow Unit) – блок работы с адресами команд, AU (Address data flow Unit) – блок манипуляции с адресами данных; DU (Data computation Unit) – блок обработки данных (рис. 3.5). Каждый из блоков участвует в процессе исполнения машинной команды (загрузка, декодирование, формировании адресов операндов, выборка операндов, выполнение, сохранение).

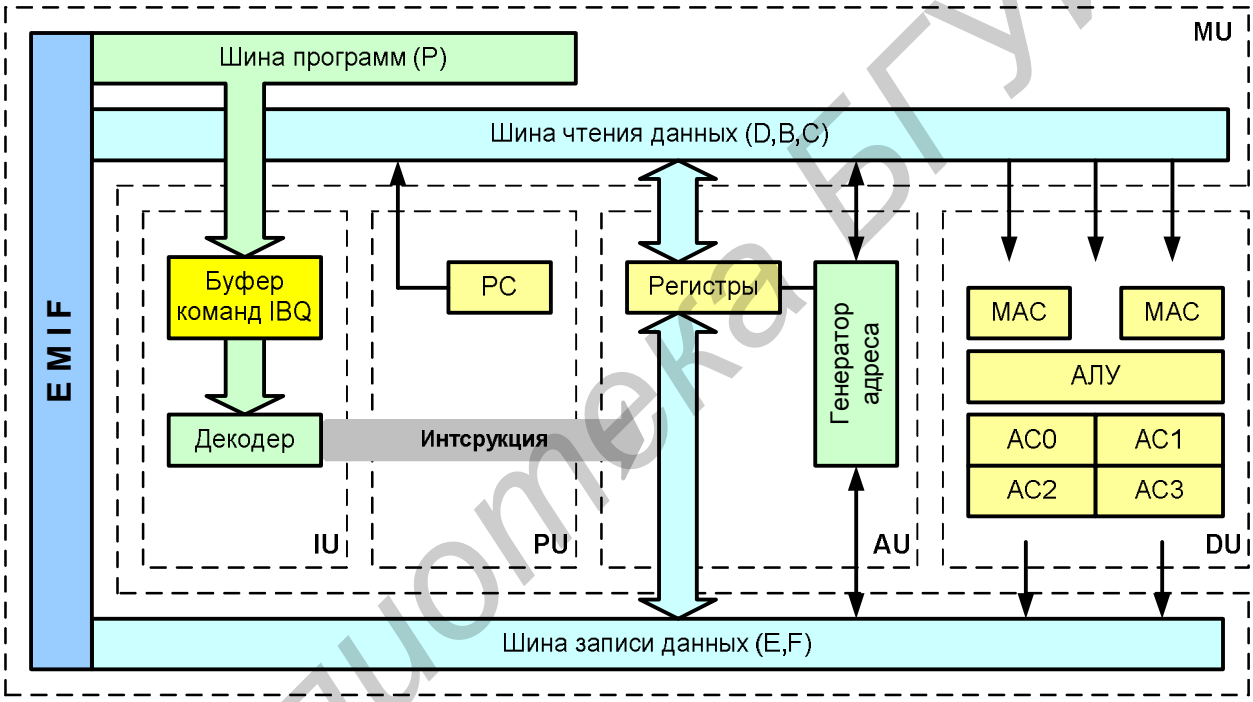


Рис. 3.5. Упрощенная структура ядра ЦП

Функционирования ядра микропроцессора упрощенно описывается следующей последовательностью. Блок PU генерирует адрес в пространстве программ, на который указывает программный счетчик (PC – Program Counter), затем выставляет его на шину P (Program Bus – Шина программ), при этом отслеживая случаи аппаратного зацикливания, условные переходы. Программный код по шине P порциями по 32 бита (4 байта) переписывается из внутренней или внешней памяти в буфер инструкций IBQ (Instruction Buffer Quire) модуля IU, размер которого равен 64x8 слов. Параллельно выполняется декодирование текущей инструкции из IBQ. Далее, в блоке AU формируется адрес для выборки операндов из пространств памяти данных и ввода/вывода

при использовании полнофункциональной логики и 16-ти разрядного АЛУ, где выполняются арифметические, логических, сдвиговых операции, и выставляется на шины D,B,C (Data read Busses – шины чтения данных). Данные с шин попадают в вычислительный блок DU, который включает 40-битное многорегистровое циклическое сдвиговое устройство, для обеспечения сдвига в диапазоне от –32 до 31, 40-битное АЛУ, пара блоков умножителей с накоплением (MAC), осуществляющих 17- разрядное умножение и 40 - разрядное суммирование или вычитание в одном цикле. Результат большинства операций остается в одном из 4-х 14 битных аккумуляторе, который затем можно сохранить через шину F или E (Data Write Busses – Шины записи данных). Связь ядра ЦП с пространствами памяти программ, данных и пространством I/O (ввода/вывода) обеспечивает MU, используя 12 независимых шин (три шины чтения данных, две шины записи данных, пять шин адреса данных, одна шина чтения программ, одна шина адреса программ) и расширенный интерфейс к памяти (EMIF – External memory Interface).

Аппаратный конвейер позволяет ускорить процесс выполнения операций в микропроцессоре. В ЦП C55xx реализованы два конвейера:

- основной 7-ми уровневый командный конвейер, который использует блоков IU, AU, DU, шины чтения данных D,B,C и шины записи данных F и E (фазы выполнения от декодирования до сохранения результат);
- дополнительный 3-х уровневый конвейер для загрузки инструкций в IBQ – блок PU, шина программ P (фазы выполнения от генерации адреса программы до загрузки в IBQ). Немаловажным преимуществом является наличие аппаратной защиты конвейера от сбоев, что значительно упрощает процесс программирования, но заставляет обращать внимание на последовательность команд при достижении максимальной эффективности.

3.3. Особенности использования MAC и других операций

Большинство алгоритмов ЦОС, с точки зрения реализации, сводится к выполнению функции MAC. Достоинством данных процессоров является возможность выполнения операции умножения с накоплением за один такт. Например, алгоритм цифровой фильтрации входного сигнала $x(n)$ фильтром с конечной импульсной характеристикой $a(i)$, математически выражается как

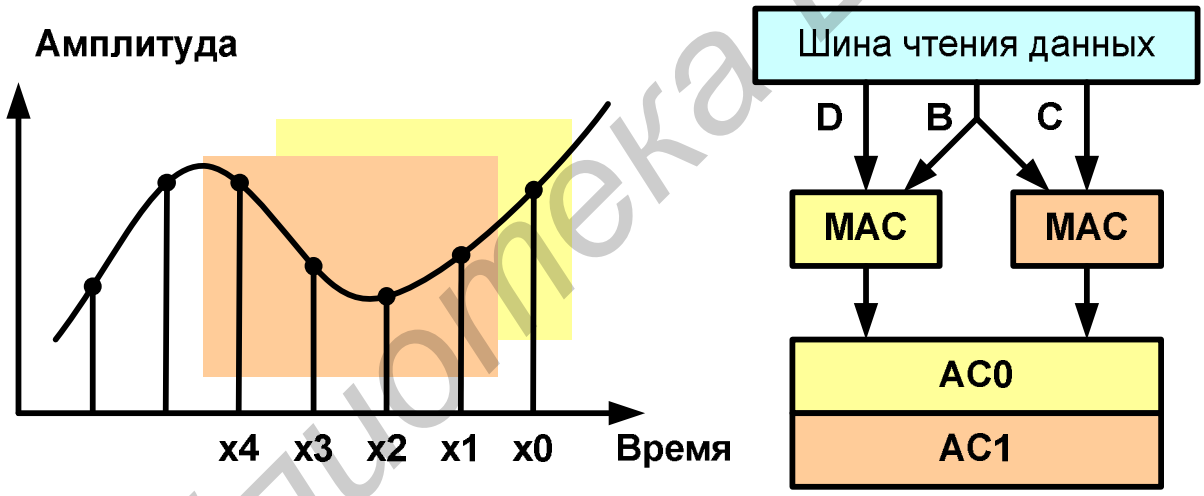
$$y(j)=\sum_{i=0}^{N} a(i)x(j-i).$$

Отсчеты входного сигнала (x(0), x(1), x(2), x(3)) и коэффициенты фильтра (a(0), a(1), a(2), a(3)) перемножаются по парно и складываются для формирования результата y(0) = a(0)*x(0) + a(1)*x(1) + a(2)*x(2) + + a(3)*x(3). На следующем шаге для получения y(1) = a(0)*x(1) + a(0)*x(2) +

+ a(0)*x(3) + a(0)*x(4), необходимо выполнитесь те же действия с отсчетами (x1,x2,x3,x4) и т.д. Однако, учет особенности архитектуры микропроцессора, наличие двух модулей MAC, позволяет ускорить в два раза вычислительный процесс. Каждый раз отсчеты входного сигнала сворачиваются с одними и теми же коэффициентами, следовательно, появляется возможность формирования сразу двух результатов y(0) и y(1) параллельно используя два модуля MAC, алгебраически это может быть записано как

```
MAC *AR2+,*CDP+,AC0 :: MAC *AR3+,*CDP+,AC1
```

где *AR2 и *AR3 – указатель на первый и второй отчет массива входной последовательности соответственно, *CDP указатель на элемент массива коэффициентов, который в данном случае будет изменяться только на 1. Два двоеточия между операциями MAC означают параллельное выполнение операций MAC, тем самым, говоря о том, что процессор применит сдвоенную операцию MAC (Dual-MAC).



а б

Рис. 3.6. Реализация MAC операции: отсчеты входной последовательности (а); структура, реализующая сдвоенную операцию MAC, которая входит в состав модуля D (б)

Входные отсчеты указываемые *AR2 и *AR3 поступают в блоки MAC по внутренним шинам D и C, а коэффициенты фильтра по шине B, результат вычисления записывается в аккумуляторы AC0 и AC1 соответственно, как показано на рис. 3.6.

Типичная реализация алгоритма фильтрации входной последовательности на языке C и её ассемблерный код, приведены в примере 3.1.

Пример 3.1

Реализация алгоритма цифровой фильтрации на языке С и ассемблер.

```
U32 tmp1, tmp2;
for (I = 0; I < blksize;
i+=2) {
  temp1 = temp2 = 0;

  for (j=0;j<taps;j++) {
    tmp1 += ((U32)dat[i+j] *
coef[j]);
    tmp2 += ((U32)dat[i+j+1]
* coef[j]);
  }
  res[i] = (tmp1 >> 16);
  res[i+1] = (tmp2 >> 16);


}
```

```
RPTBlocal done
    MPY *AR2+,*CDP+,AC0
 :: MPY *AR3+,*CDP+,AC1
 || RPT #13    ; RPT
параллельно с MPYs
    MAC *AR2+,*CDP+,AC0
 :: MAC *AR3+,*CDP+,AC1
    MAC *(AR2-T0),*CDP+,AC0
 :: MAC *(AR3-T0),*CDP+,AC1
done:
    MOV
pair(hi(AC0)),dbl(*AR4+)
```

Преимущество данной схемы так же проявляется при обработке двух входных сигналов одним фильтром, например, обработка входного стерео сигнала входным фильтром или при одновременной обработке одного входного сигнала двумя различными фильтрами, например, при реализации базовой декомпозиции вэйвлет-преобразования или любым другим банком цифровых фильтров.

Команды повторения RPT или RPTB позволит решить задачу организации внутренних и вложенных циклов, однако возникновение необходимости дополнительно загрузки программного кода потребует дополнительных циклов доступа к внутренней памяти программ. В случаях, когда блок повторения или внутренний цикл может быть полностью размещен в IBQ, рекомендуется применять команду RPTBlocal, как это показано на примере 3.1.

3.4. Подключение внешней памяти через EMIF

Доступ к внешнему адресному пространству осуществляется через расширенный интерфейс к памяти (EMIF – External Memory Interface), здесь всё адресное пространство разделено на 4 подпространства по 4 Мбайта, указываемых внешними выводами CEx (рис. 3.7).

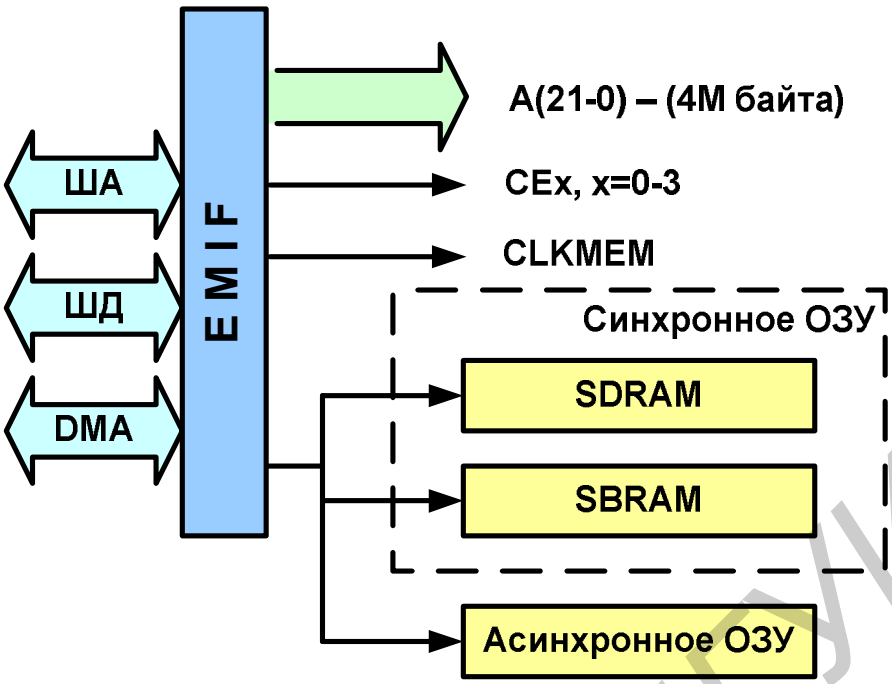


Рис. 3.7. Сопряжение внутренних шин ЦП с внешней памятью через EMIF

Каждое из 4-х адресных подпространств обеспечивает подключение внешней синхронной памяти (синхронное динамическое ОЗУ (SDRAM), синхронное пакетное статическое ОЗУ (SBSRAM)), или асинхронной памяти / устройств ввода вывода в зависимости от значения бит в поле MTYPE регистра CEx SCR. При подключении синхронного ОЗУ EMIF ЦП поддерживает режим 32-битного чтения и записи на разных тактовых частотах кратных частоте ЦП (1, 1/2, 1/4, 1/8, 1/16).

Внешние устройства ввода/вывода подключаемые к микропроцессору имеют свое изолированное пространство (64К) по отношению к пространству памяти программ и данных, что позволяет подключить значительное количество внешних устройств.

Карта памяти программ/данных показана на рис. 3.8. Для обращения к байтам пространства памяти программ ЦП использует 24 битный адрес. При получении доступа к пространству памяти данных ЦП используется только 23 бита адреса (младший бит равен 0). Вся память данных ЦП адресуется страницами по 64К (128 станиц). В начале 0 страницы пространства адресов зарезервирована область под регистры, отображаемые на память (MMR), 128-я станица выделяема под внутреннее ПЗУ, которое разбито на 3 блока 32,16 и 16К байт, причем последние 16К могут быть сконфигурированы как секретное ПЗУ (Secure ROM).

Подключение асинхронной памяти или устройств ввода/вывода через EMIF ЦП предоставляет полную гибкость в программировании временных диаграмм циклов чтения/записи и формата данных (8/16/32 бита). Регистры

управления пространством памяти CEx CSR1 управляет циклами асинхронного чтения (пример временной диаграммы показан на рис. 3.9), а регистра CEx CSR2 – циклами записи.

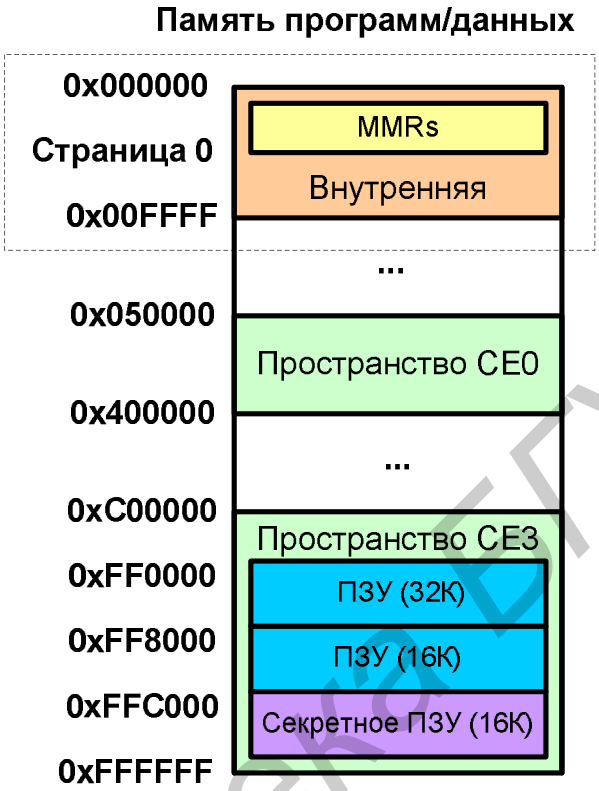


Рис. 3.8. Карта памяти ЦП C5510

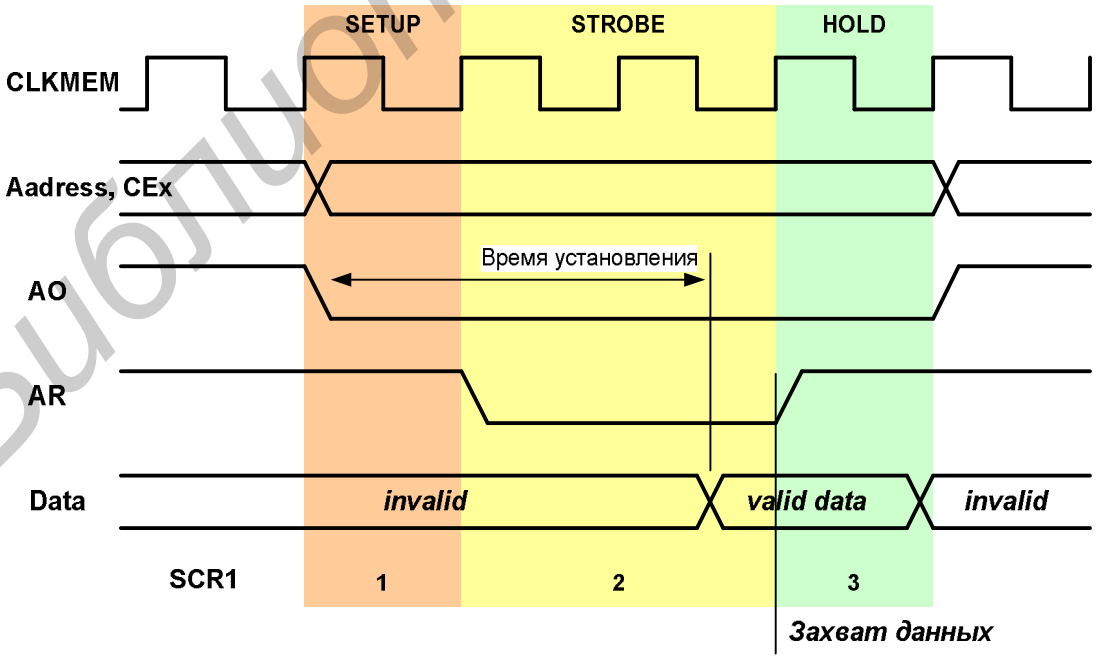


Рис. 3.9. Временная диаграмма цикла чтения из асинхронного ОЗУ/портов ввода/вывода

Доступ к внутренней памяти, как со стороны адресного пространства программ, так и со стороны адресного пространства данных возможен к одним и тем же блокам памяти, отличие состоит в способе чтения данных (пространство программ - доступ байтный, порциями по 4 байта, пространство данных – доступ словный 16 или 32).

3.5. Введение в CCS, 5510DSK, DSK_APP1

Целью данной лабораторной работы является ознакомление с 5510DSK, Code Composer Studio (SSC) и приложениям для аудиообработки dsk_app1.

Структура приложения, которое реализуется программным обеспечением dsk_app1, показана на рис. 3.10.

Входной сигнал снимается с аналого-цифрового преобразователя (АЦП) и через последовательный порт McBSP под управлением канала контроллера прямого доступа (DMA) записывается в один из входных массивов (PING или PONG). В зависимости от положения переключателей DIP_1 ядро ЦП выполняет сложение входной последовательности со сгенерированным сигналом синусоиды или нет. В зависимости от положения переключателя DIP_2 входная последовательность просто копируется или фильтруется полосовым фильтром. Результирующая последовательность записывается в выходной буфер PING или PONG для передачи каналом DMA через McBSP в цифро-аналоговый преобразователь для воспроизведения.

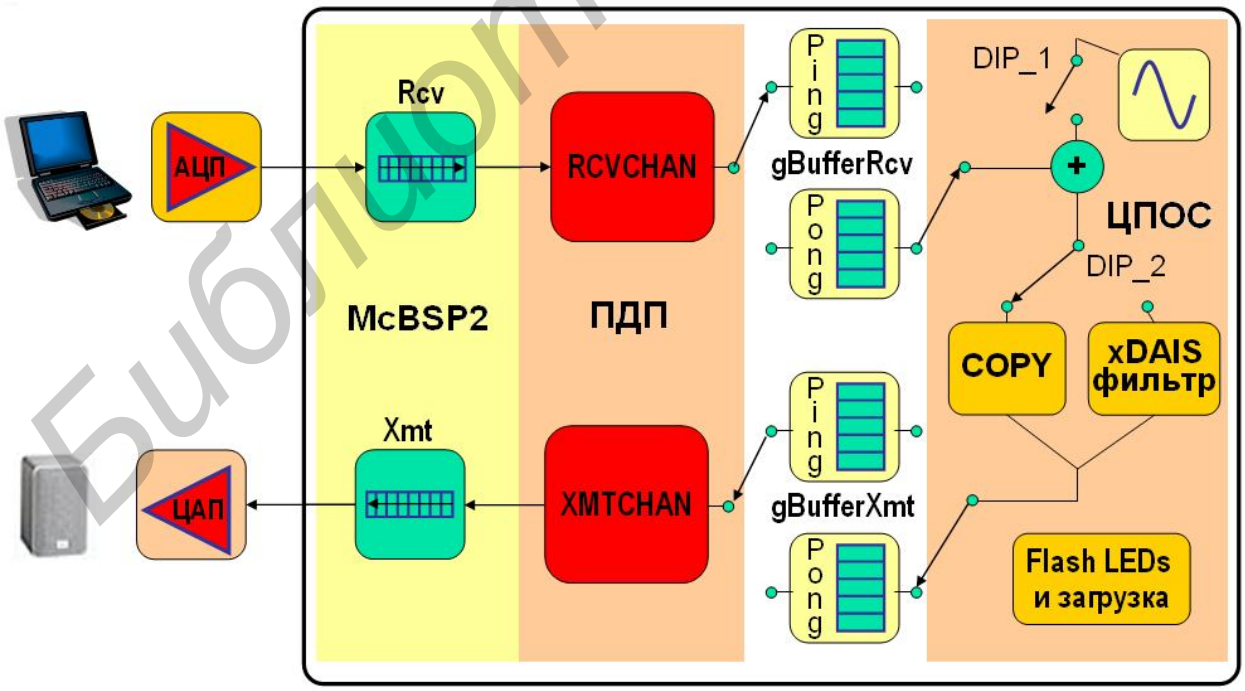


Рис. 3.10. Структура системы реализуемой на стартовом наборе, приложением dsk_app1

3.6. Настройка и тестирование аппаратного модуля DSK

Последовательность настройки и тестирования аппаратного модуля DSK, перед запуском приложения, выполняется согласно следующим шагам.

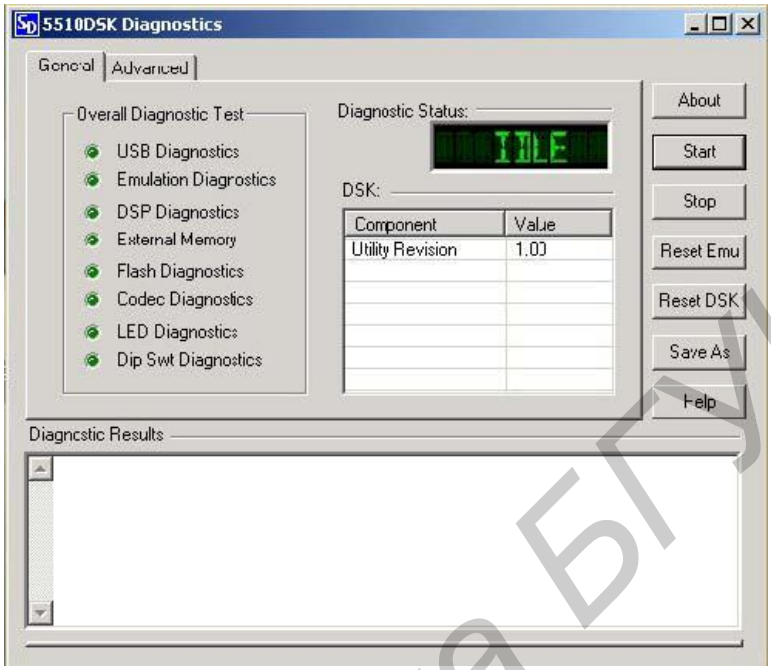


Рис. 3.11. Диалоговое окно утилиты диагностики стартового модуля DSK

1. Присоедините кабели: USB порт (от ПЭВМ к DSK), динамики (наушники) к разъему DSK для наушников.

2. Присоедините кабель питания и наблюдайте POST (самотестирование после включения). В течении POST, 9 различных тестов выполняются последовательно. Когда светодиоды прекратят моргать и останутся в положении ON (включено) POST завершил свое выполнение.

3. Запуск утилиты диагностики DSK 5510. 5510 DSK Diagnostic применяется для тестирования и диагностики составных частей стартового набора, внешний вид утилиты показан на рис. 3.11.

3.7. Знакомство с аудиоприложением для DSK (dsk_app1.c)

Последовательность загрузки и запуска проекта, а так же как осуществляется работа с DIP переключатели в приложении dsk_app1:

1. Запустите Code Composer Studio (CCS).

2. Откройте аудио проект. Выберите:

Project → Open

и найдите файл dsk_app1.pjt, который располагается в

<диск>:\<CCS директорий ”ti”>\examples\dsk5510\bsl\dsk_app\dsk_app1.pjt

3. Установите опцию «Load Program After Build» (Загрузка программы после построения), для этого выберите: Option → Customize, далее закладку «Program Load Option» (Настройка загрузки программы) и отметьте пункт «Load Program After Build» (Загрузка программы после построения).

4. Подключите свободный конец аудио кабеля от ПЭВМ к линейному входу DSK и запустите на проигрывание музыкальный файл.

5. Скомпилируйте проект dsk_app1. Для этого выберите:

Project → Rebuild All

или нажав кнопку “Rebuild All” (скомпилировать Всё). Появится информационное окно, которое будет отражать отдельные шаги выполнения: этапов компиляции, ассемблирования, компоновки и другие.

6. Запустите приложение, выбрав Debug → Run или нажмите кнопку (Run) запуска на левой вертикальной панели инструментов. Когда все DSK переключатели подняты, приложение запущено в режиме пропуска аудио сигнала через систему и в наушниках или спикерах должна быть слышна музыка. При этом светодиодные индикаторы LED #2 и #3 сигнализируют о использовании в обработке буферов PING и PONG соответственно.

7. Откройте CPU Load Graph (График загрузки процессора), выбрав DSP/BIOS → CPU Load Graph. График загрузки процессора будет показывать приблизительно 4,5 % (если все переключатели подняты, т.е. не включены).

8. Включите фильтр верхних частот, нажатием на DIP переключатель #3 (см. табл. 3.1) и обратите на изменения в музыке. Приложение считывает состояние DIP переключателей и запускает КИХ-фильтр, при этом загрузка ЦП увеличится, что отмечается на графике загрузки, который показывает 9%.

Таблица 3.1

Функции DIP переключателей в данном приложении

Переключатель	Предназначение
0	Мигание светодиода LED #1 (периодическая функция, которая переключается раз в 500 мс)
1	Добавляет холостую загрузку (около 20 % загрузки процессора)
2	Не используется
3	Добавляет высокочастотный фильтр (dsk_app1 использует ассемблерную реализацию фильтра, а dsk_app2 использует реализацию фильтра на языке Си)

9. Нажмите DIP-переключатель #0 (см. табл. 3.1), светодиод LED #0 начнет моргать. Данный переключатель включает периодическую функцию, которая переключает светодиод (из положения включено в положение выключено, и обратно) каждые 500 мс. Общий эффект заключается в том, что светодиод моргает каждую секунду. График загрузки значительно не изменится.

10. Нажмите DIP переключатель #1 (см. табл. 3.1). Данное действие добавляет пустые команды для увеличения загрузки ЦП, что может помочь при моделировании функций в системе, которая ещё не написана. Значение графика загрузки увеличится до 30 %, т.е. дополнительно 20 % загрузки процессора.

3.8. Устранение ошибок/проблем в CCS

Во время отладки в CCS, могут возникать ошибки. Во многих случаях, ошибки легко устраняются без закрытия CCS и отключения питания от платы. Приложение аудиообработки требует инициализации GEL во время запуска CCS (что не выполняется автоматически). Если был выполнен сброс процессора, то также были стерты все настройки внешнего расширенного интерфейса (EMIF) и других интерфейсов, которые были заданы соответствующим образом через конфигурационную базу данных (файл dsk_app1.cdb). Несколько способов по устранению ошибок:

1. Обратите внимание на состояние ЦП «CPU Running» (Процессор работает) или «CPU Halted»(Процессор остановлен). Попытки выполнить что-нибудь в процессе работы ЦП может приводить к ошибкам. Таким образом, остановка ЦП может решить проблему.

2. После остановки процессора, попытайтесь его вновь запустить. Если это не работает, остановите процессор и попытайтесь выполнить сброс процессора в режиме отладки Debug → Reset CPU. Запустите процессор снова. Перезапуск приводит к переходу на символ входа (_c_int00).

3. Остановите процессор, затем выберите GEL → C5510_DSK_Configuration → C5510_DSK_Init (подождите несколько секунд), перегрузите программу и запустите её снова. Программа инициализации DSK выполняет полный сброс аппаратной части DSK и памяти.

4. Если обнаружилась ошибка и CCS предлагает «Устранить» ее - нажмите Да, только выполняйте это не в среде работающей в режиме реального времени (RTA) (таких как график загрузки процессора). Обычно лучше использовать метод полного восстановления.

5. Если нет результата от проделанной работы, время использовать более эффективный метод. Закройте CCS, отключите питание от DSK, подсоедините

питание назад, повторно вызовите CCS, выполните инициализацию DSK (DSK init), перезагрузите вашу программу, запустите снова.

3.9. Составные части аудиоприложения

Конфигурационная база данных (.cdb файл) используется абсолютно во всем - при управлении памятью, устройствами фазовой подстройки частоты (PLL), распределением потоков, настройки холостого хода (режимы снижения энергопотребления), и периодическими функциями и т.д. Для доступа к .cdb файлу, выберите dsk_app1.pjt в «Окне проекта» (Project Window), в папке DSP/BIOS Config (Конфигурация DSP/BIOS) располагается dsk_app1.cdb.

Менеджер памяти – это инструмент, где определяется карта памяти системы и где располагаются различные части приложения (исходный код, данные, константы). Менеджер памяти размещается доступен через конфигурационную базу данных, секция Mem – Memory Section Manager (Менеджер секций памяти). Свойства данного менеджера отображают размещение каждой секции в пространстве памяти, например, все глобальные переменные, размещаются в секции .bss, которая расположена в области памяти называемой DARAM (ОЗУ с двойным доступом). Менеджер памяти заботится обо всех сгенерированных компилятором секциях, таких как .bss. Однако, можно специально определить секцию для важного массива и расположить его где-то произвольно на карте памяти. Это поддерживается через директиву #pragma в исходном коде, чтобы назначить секцию, определенную пользователем для массива коэффициентов так, чтобы её можно было бы разместить в отдельном блоке SARAM (ОЗУ с одиночным входом) – отдельно от DARAM. Единственный способ сделать это – использовать директиву #pragma (в файле highpass.h) и специальный пользовательский командный файл компоновки. Дополнительно в пользовательском файле userlink_app1.cmd указывается, что секция коэффициентов размещается в области памяти SARAM_A.

Основной файл dsk_app1.c, расположенный в папке Source (Источники) Project Window (Окне проекта.) содержит программу инициализации необходимого оборудования, декларирует буферы, процедуры, реализующие основные действия данного приложения. Так как кодек посылает стерео данные, необходимо выполнять сортировку каналов (т.е. разделение потока данных через контроллер ПДП) на L (левый) и R (правый). Rcv буферы предназначены для ввода музыки. Xmt для вывода обработанной музыки. Дополнительный буфер задержки требуется для функций библиотеки DSPLIB, которые используются для фильтрации музыки.

Процедура processBuffer() описывает действия связанные с фильтрацией и копированием. Обратите внимание, что части программы для ветви пинг и понг выглядят одинаково, однако DIP переключатель #3 определяет будут ли просто копироваться данные из Rcv в Xmt буферы или будет выполняться фильтрация данных.

Процедура dmaHwi() обработки прерывания (interrupt service Routine (ISR)), которая выполняется, когда контроллер ПДП заполнит входной буфер Rcv данными. В данной точке приложение или просто переносит данные с входа на выход, или фильтрует входные данные, что зависит от положения переключателя DIP #3. Для информирования процедуры processBuffer() используются программные прерывания (SWI) или программный интерфейс приложения (API) для того, чтобы сказать какой из двух процессов необходимо выполнять: пинг или понг. dmaHwi() обработчик прерывания ISR – processBuffer() процедура программного прерывания (SWI), которая прерывается dmaHwi().

Основная процедура main(), где выполняется вся инициализация и запуск каналов контроллера ПДП. После выполнения main программа попадает в функция холостого хода операционной системы DSP/BIOS. Приложения, не применяющие DSP/BIOS обычно используют бесконечный цикл while() в основном цикле. Однако, в приложениях использующие BIOS, функция BIOS_Init запускается перед передачей управления в основную процедуру main(). BIOS_Init вызывает процедуру main() и когда main() завершается (или выпадает), то управление возвращается обратно BIOS (когда это попадает напрямую в цикл холостого хода BIOS IDL).

Dsk_app1 использует специализированную библиотеку DSPLIB для фильтрации входящей музыки. Библиотека DSPLIB включат в себя 50 ассемблерных функций, вызываемых из Си, которые нужно использовать для выполнения различных функций цифровой обработки сигналов. В данном примере применяется fir2 функция, которая выполняет КИХ фильтрацию, используя операцию сдвоенного MAC.

3.10. Профилирование приложения в реальном времени

Для оценки количества требуемых циклов и скорости работы функции фильтра, используют профилирование исходного кода, используя встроенный профайлер. Однако этот метод профилирования не позволяет проводить обработку в реальном времени, и поэтому аудообработка не сможет нормально работать. Верный способ профилирования процедуры – использование статистики, которое доступно через пункты меню DSP/BIOS → Statistics View.

Statistic View (Окно статистики) предоставляет информацию о временных характеристиках большинства потоков в системе.

Пример, описывающий последовательность выполнения операции профилирования функции fir2, выглядит следующим образом.

Определите место процедуры processBuffer() в исходном файле dsk_app1.c. Найдите первый вызов процедуры fir2(), которая выглядит следующим образом

```
fir2(gBufferRcvPingL, COEFFS, …);
```

Добавьте следующую строку перед первым вызовом функции fir2()

```
STS_set(&fir_time, CLK_gethtime());
```

Данная строчка кода будет говорить о том, что начать отсчет следует здесь. &fir_time - есть объект статистики. Сразу же за первым вызовом функции fir2(), добавьте следующее выражение.

```
STS_delta(&fir_time, CLK_gethtime());
```

В результате получим два объявления перед и после первой функции fir2() как показано ниже:

```
STS_set(&fir_time, CLK_gethtime());
fir2(gBufferRcvPingL, COEFFS, gBufferXmtPingL,…);
STS_delta(&fir_time, CLK_gethtime());
```

Статистическое слежение будет выполняться запуском часов с нуля на первом определении и затем рассчитывать задержку до второго объявления. Аттестация будет отображаться через fir_time в окне статистики.

Для определения объекта статистики, необходимо выполнить изменения в конфигурационной базе данных. Откройте STS – Менеджер объектов статистики в пункте Instrumentation (Инструментарий) и объявите объект статистики fir_time, при этом необходимо установить единицы измерения High Resolusion Time Based (Базироваться на высоком разрешении времени).
Завершив изменения и выполнив компиляцию проекта, оценивает значение статистики, которое должно быть равным приблизительно 111К циклов, это среднее число циклов.

3.11. Задание к лабораторной работе

1. Ознакомиться с работой приложения dsk_app1/

2. Понять предназначение DIP переключателей и их влияние на работу приложения.

3. Выполнить профилирование в реальном времени, отдельных составляющих приложения dsk_app1, по аналогии с предложенным методом.

4. Исследовать возможность интегрированной среды CCS, отобразив:

- Графическое отображение памяти (View → Graph → Time/Frequency)

- Оживление графического представления (используйте кнопку оживления и отобразите графически входящую музыку)

- Смешанное отображение исходного текста и ассемблера. (View → Mixed Source/ASM)

- Содержимое регистров центрального процессора и периферии. (View → CPU Registers)

- GEL файлы (GEL опции меню, создание вашего собственных GEL файлов)

- Работа с рабочим пространством (создание, сохранение, восстановление) (File → Workspace)

- Мастер кода (Option → Customize)

4. ЛАБОРАТОРНАЯ РАБОТА «СПОСОБЫ ЭНЕРГОСБЕРЕЖЕНИЯ»

4.1. Введение

Цель данной лабораторной работы заключается в исследовании энергосберегающих методов для C5510, а именно: снижение частоты в системе автоподстройки частоты, перевод в режим холостого хода не используемых ресурсов, уменьшение напряжения питания с 1.6 до 1.1 вольта.

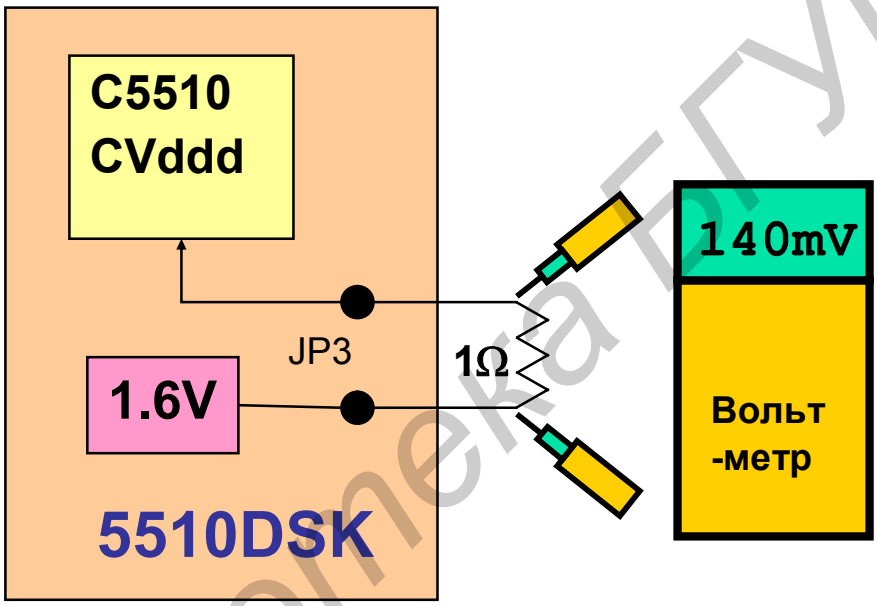


Рис. 4.1. Структура схема подключения лабораторного оборудования

Для выполнения лабораторной работы необходимо иметь дополнительное оборудование: цифровой вольтметр, сопротивление 1 – Ом с контактным приспособлением, пара щупов. Последовательное включение 1 Ом резистора вместо перемычки J3, позволит выполнить измерение напряжения (мВ), что будет эквивалентно измерению напряжения (мА) согласно закону Ома. (рис. 4.1).

На рис. 4.2 показана перемычка JP3, к которой необходимо выполнить подключение сопротивления и щупов цифрового вольтметра. Схема прототип включения лабораторного оборудования для измерения потребляемого тока показана на рис. 4.3.

Подключите к DSK аудио кабели, подайте напряжение питания и запустите воспроизведение музыки на ПЭВМ, при этом DIP переключатель #3

должен быть нажат. Выполните запуск CCS, загрузите dsk_app2.pjt, расположенный
<директорий>:\<папка CCS>\examples\dsk5510\bsl\dsk_app\dsk_app2.pjt.

Следуя описанным ниже алгоритмам, определите потребляемую мощность, при применении различных способов энергосбережения.

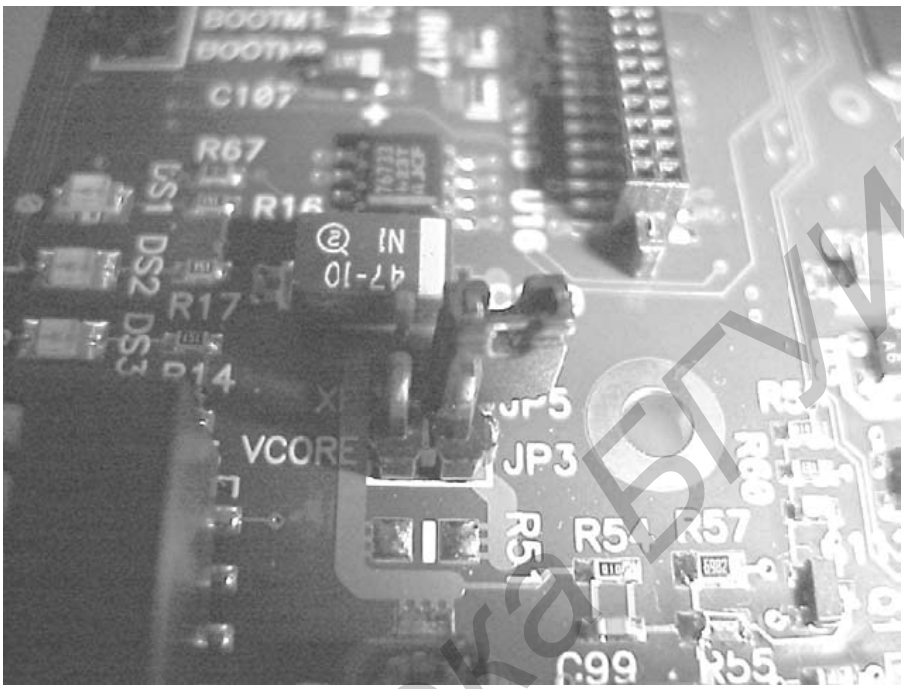


Рис. 4.2. Вид перемычки JP3 на плате DSK 5510

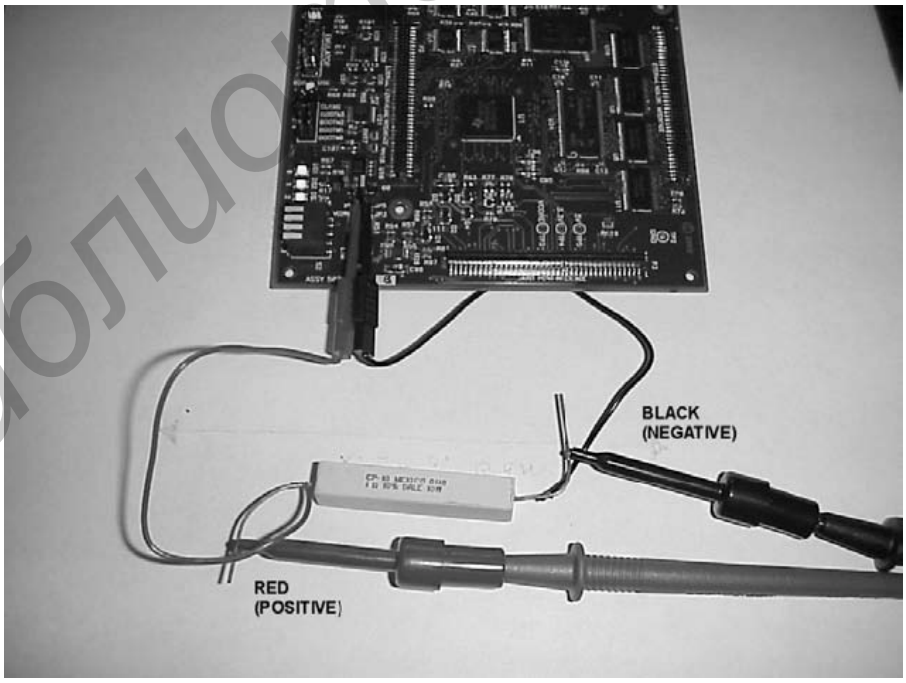


Рис. 4.3. Схема прототип включения лабораторного оборудования для измерения, потребляемого тока

4.2. Номинальная потребляемая мощности @ 200 МГц, 1,6 В

Текущее значение тактовой частоты системы фазовой подстройки частоты описывается в конфигурационной базе данных dsk_app2.cdb. Откройте .cdb файл. Нажмите на знак + рядом с System (Система), правой кнопкой мыши нажмите на Global Settings (Глобальные настройки) и выберите Properties (Свойства). Обратите внимание на текущую тактовую частоту в 200 МГц и значение регистра CLKMD (0x2cd0). От этого зависит значение умножителя, равное 25 и значение делителя, равное 3 (2+1). При входной частоте 24 МГц, в результате умножения и деления получается частота процессора 200 МГц.

Закройте окно Global Setting Properties (свойства глобальных настроек), выберите в настройках конфигурации сборки «Release» и выполните сборку, загрузку и запуск проекта. Конфигурация сборки проекта «Release» использует 2-й уровень оптимизации без отображения отладочной информации. Музыка должна играть через наушники или колонки, а светодиодные индикаторы – мигать, а DIP переключатель #3 должен быть нажат (включен фильтр верхних частот)

График загрузки процессора будет показывать значение, равное 9%, а значение тока будет 145 мА.

4.3. Потребляемая мощность @ 16 МГц, 1,6 В

Изменение тактовой частоты работы ЦП влечет к значительному уменьшению потребления электроэнергии, однако применима не во всех система, так как зависит от особенностей реализуемого приложения.

Снижение тактовой частоты системы фазовой автоподстройки частоты до 16 МГц приведет к увеличению загрузки процессора до 75 %, оставшиеся 25 % будут обеспечивать запас производительности для будущих функций. Основная идея, заключается в сокращении потребляемой мощности и это выполняется следующим образом.

Откройте .cdb-файл, измените скорость работы процессора на 16 и значение регистра CLKMD на 0x2150. Необходимо изменить «оба» значения, иначе временные значения любого периода будут не верными (тактовая частота BIOS базируется на значении установленном в поле ввода скорости процессора (DSP Speed) и должна соответствовать частоте вычисленной при настройке регистров CLKMD). 0x2150 программирует систему фазовой автоподстройки частоты с множителем 2 и делителем 3(2+1).

Соберите проект в конфигурации «Release», загрузите и запустите его. Значения потребления станет равным 15мА.

4.4. Потребляемая мощность @ 16 МГц, 1,6 В + отключение ЦП

Следующий способ сокращения энергопотребления заключается в отключении ЦП во время режима холостого хода (CPU IDLE).

Для этого, необходимо добавьте процедуру powerDown(). Откройте dsk_app2.c для редактирования. Добавьте следующую процедуру в конец файла:

```
void powerDown(void)
{
PWR_powerDown(PWR_WAKEUP_MI);
}
```

Функция PWR_powerDown() вызывает стандартную процедуру из библиотек поддержки чипа (CSL – Chip Support Library) powerDown и выполняет инструкцию IDLE с учетом настроек переданных в функцию.

Выбор IDLE в настройках понижения питания в CSL выполняется следующим образом. Откройте .cdb-файл и нажмите знак плюс рядом с Chip Support Library (Библиотеки поддержки чипа). Нажмите знак плюс рядом с Power (Энергия), нажмите правой кнопкой мыши на Power Configuration Manager (Менеджер конфигурации энергии) и выберите Properties (Свойства). Поставьте птичку напротив Enable Pre-initialization (разрешение предварительной инициализации). Нажмите на вкладку Power Save During IDLE Instruction (Выполнять энергосберегающие функции при выполнении инструкции IDLE). Проверьте отсутствие птички CPU Disable (блокировка процессора). Нажмите Ok, применив настройки.

Примечание: график загрузки процессора должен быть закрыт и опция RTDX – Real-Time Data Transmit (передача данных в реальном времени), для этого откройте .cdb-файл, выберите System и в свойствах глобальных настроек отмените следующие две строки

```
Enable Real Time Analysis
```
(Включение анализа в реальном времени)
```
Enable All Trace Event Classes
```
(Включение всех классов трассировки событий)

Создайте поток холостого хода для функции powerDown, нажав знак плюс, следующий после Scheduling (планирование). Нажмите знак плюс рядом с IDL-IDL Function Manager (Менеджер функций холостого хода). Нажмите правой кнопкой мыши на IDL и выберите Insert IDL (вставить IDL).

Переименуйте IDL0 в IDL_powerDown. Нажмите правой кнопкой мыши на IDL_powerDown и выберите Properties (свойства). Измените функцию на _power Down. Отмените Include in CPU load calibration (Включить в калибровку загрузки процессора). Если данный пункт включен, то BIOS будет запускать данную функцию перед основной программой main(). Подключите файл заголовка <csl_pwr.h> в файле dsk_app2.c.

Соберите проект в конфигурации «Release», загрузите и запустите. Цифровой вольтметр будет показывать 14 мА, что составляет примерно 10%-ный выигрыш и позволит при длительном использовании системы ещё больше продлить срок службы батарейки.

4.5. Потребляемая мощность @ 16 МГц, 1,1 В + отключение ЦП

Данный способ снижения энергопотребления заключается в уменьшении напряжения питания ядра ЦП до 1.1В, для этого в структуре модуля DSK предусмотрена схема управления выводом питания микросхемы стабилизации напряжения питания ядра ЦП через выход GPIO0 (внешний выход общего назначения 0) ЦП. Для активизации данной опции необходимо выполнить дополнительное конфигурирование в CSL. Нажмите знак плюс рядом с GPIO и нажатием правой кнопки мыши на конфигурации выберите Properties (Свойства). Включите опцию Enable Pre-Initialization (включить предварительную инициализацию). Нажмите на вкладку Non-Power Down I/O Pins (контакты ввода/вывода, остающиеся под напряжением) и выберите Output Low (Низкий выход) для IO0. Если на GPIO0 высокий уровень (или сконфигурирован как вход), то выбрано 1,6 В. Если GPIO0 низкий (и сконфигурирован как выход) 1,1 В выбрано. Примените и сохраните изменения в конфигурационном .cdb файле. Соберите проект, используя конфигурацию «Release». Новое значение составит 9mA.

Данный тип измерения тока применим только для ядра микропроцессора – он не может быть применен для других устройств системы, которые тоже требуют электропитание: EMIF (Расширенный интерфейс к памяти), кодеки, регуляторы напряжения, входные генераторы, некоторые большие модули памяти и т.п. Однако, программа dsk_app2.out использует только внутреннюю память, поэтому только кодек и генератор потребляют дополнительную мощность (что минимально). Запущенный на частоте 200 МГц и напряжении питания 1.6В ЦП потребляет 232 мВт. После применения оптимизации («Release» конфигурация), которая позволила сократить частоту процессора до 16 МГц и применить инструкцию IDLE в режиме холостого посредством BIOS IDL позволило сократить потребляемой энергии с 232 мВт до 14 мВт.

4.6. Задание к лабораторной работе

1. Исследователь эффективность методов энергосбережения.

2. Выполнить реализацию методов энергопотребления, каждого в отдельности.

3. Определить насколько уменьшится энергопотребление ядра ЦП при применении, методов описанных выше, в отдельности и совместно. Составить сводную таблицу.

4. Сделать заключение какой из выше перечисленных методов энергосбережения, дает больший эффект.

5. Заполнить приведенную таблицу 4.1.

Таблица 4.1

ЦП [МГц]	ЦП [В]	CLKMD	Загрузка ЦП	Ток [мВ]
200	1,6	0x2cd0		
16	1,6			
16+IDLE	1,6			
16+IDLE	1,1			

СОДЕРЖАНИЕ

1. ЛАБОРАТОРНАЯ РАБОТА «ОСНОВЫ ПРОГРАММИРОВАНИЯ 8-РАЗРЯДНЫХ МИКРОПРОЦЕССОРОВ СЕМЕЙСТВА MC68HC11» 3

1.1. Отладочный модуль M68HC11EVB 4

1.2. Структура и режимы функционирования микроконтроллера.....5

1.3. Память микроконтроллера 9

1.4. Способы адресации и система команд.....11

1.5. Встроенный монитор 12

1.6. Команды монитора 13

1.7. Подготовка, загрузка и отладка программ 16

1.8. Пример построения программ.....17

1.9. Задание на лабораторную работу.....18

2. ЛАБОРАТОРНАЯ РАБОТА «ПРОГРАММИРОВАНИЕ/СТИРАНИЕ РЕПРОГРАММИРУЕМОГО ПЗУ МИКРОКОНТРОЛЛЕРА MC68HC11» 19

2.1. Введение.....19

2.2. Алгоритм стирания РПЗУ 20

2.3. Пример реализации алгоритма стирания.....20

2.4. Программирование РПЗУ.....21

2.5. Пример реализации алгоритма программирования 22

2.6. Задание на лабораторную работу.....23

3. ЛАБОРАТОРНАЯ РАБОТА «ЗНАКОМСТВО СО СТАРТОВЫМ НАБОРОМ TMS320VC5510 DSK» 24

3.1. Состав, основные характеристики и возможности стартового набора TMS320VC5510 DSK. 24

3.2. Особенности структуры внутреннего ядра микропроцессора TMS320VC5510 29

3.3. Особенности использования MAC и других операций 30
3.4. Подключение внешней памяти через EMIF 32
3.5. Введение в CCS, 5510DSK, DSK_APP1 35
3.6. Настройка и тестирование аппаратного модуля DSK 36
3.7. Знакомство с аудиоприложением для DSK (dsk_app1.c).....36
3.8. Устранение ошибок/проблем в CCS.....38
3.9. Составные части аудиоприложения. 39
3.10. Профилирование приложения в реальном времени. 40
3.11. Задание к лабораторной работе.....42
4. ЛАБОРАТОРНАЯ РАБОТА
«СПОСОБЫ ЭНЕРГОСБЕРЕЖЕНИЯ».....43
4.1. Введение.....43
4.2. Номинальная потребляемая мощности @ 200 МГц, 1,6 В.....45
4.3. Потребляемая мощность @ 16 МГц, 1,6 В.....45
4.4. Потребляемая мощность @ 16 МГц, 1,6 В + отключение ЦП 46
4.5. Потребляемая мощность @ 16 МГц, 1,1 В + отключение ЦП 47
4.6. Задание к лабораторной работе 48

Св. план 2006, поз.65

Учебное издание

Петровский Алексей Александрович

МИКРОПРОЦЕССОРНЫЕ СРЕДСТВА И СИСТЕМЫ

Лабораторный практикум
для студентов специальности I-40 02 01
«Электронные вычислительные машины, системы и сети»
всех форм обучения

Ответственный за выпуск А.А. Петровский

Подписано в печать 20.07.2006.	Формат 60x84 1/16.	Бумага офсетная.
Гарнитура «Таймс».	Печать ризографическая.	Усл. печ. л.3,14.
Уч.-изд. л. 2,8.	Тираж 150 экз.	Заказ 80.

Издатель и полиграфическое исполнение: Учреждение образования
«Белорусский государственный университет информатики и радиоэлектроники»
ЛИ №02330/0056964 от 01.04.2004. ЛП №02330/0131666 от 30.04.2004.
220013, Минск, П. Бровки, 6