Системный конфигуратор 2

2.1. Введение

Системный конфигуратор это программный инструмент для описания вашего оборудования. Каждая система ADSP-21xx может иметь уникальную аппаратную конфигурацию и использовать различную структуру памяти. Системный конфигуратор вводит определения вашей аппаратной конфигурации, включая память и порты ввода/вывода, в виде файла, читаемом редактором связей и программой моделирования (симулятором). Редактору связей эта информация требуется для распределения хранения программ и данных в доступном пространстве памяти. Симулятор должен моделировать вашу системную архитектуру и процессор ADSP-21xx, базирующийся на ней.

Системный конфигуратор может быть использован для предварительной установки распределения памяти для эмулятора ADSP-21xx. Подробнее данный вопрос рассмотрен в разделе «Распределение сегментов для эмулятора».

Рис.2.1 показывает конфигурацию памяти, позволяющую адресовать максимальное число слов в каждом пространстве памяти для каждого процессора. Системный конфигуратор позволяет создавать описания системной архитектуры только в пределах этих ограничений.

	ADSP-2100 (вся память внешняя)	ADSP-2105 ADSP-2115	ADSP-2101 ADSP-2111
Память данных 16-бит данные	16 К максимум	14.5 К максимум (0.5 К внутренней, 14 К внешней)	15 К максимум (1 К внутренней, 14 К внешней)
Память программ 24-бит программы, или 16/24-бит данные	16 К смешанной памяти программ и данных, или 32 К (16 К программ и 16 К данных)	15 К максимум (1 К внутренней, 14 К внешней)	16 К максимум (2 К внутренней, 14 К внешней)
Память начальной загрузки 24-бит программы, 16/24-бит данные, дополненные до 32-битных слов	----------	8 К максимум (32 Кб, организованных в 32-битные слова)	16 К максимум (64 Кб, организованных в 32-битные слова)

Рис.2.1. Конфигурации памяти семейства ADSP-2100 .

Каждое пространство памяти адресуется раздельно. Адреса в памяти программ отличаются от адресов в памяти данных. Память начальной загрузки адресуется особым образом и используется процессором во время загрузки (**Примечание**: в ADSP-2100 память начальной загрузки отсутствует).

Вы должны указать для системного конфигуратора *исходный файл системной конфигурации* (*system specification source file*), обычно с расширением `.SYS`; этот файл описывает ваше целевое оборудование. В этой главе приведены директивы системного конфигуратора, которые используются для написания файла.

Системный конфигуратор обрабатывает входной файл и создает *файл описания архитектуры* (*architecture description file*) с расширением `.ACH`. Файл описания архитектуры расшифровывается редактором связей для определения размещения программ и фрагментов данных в памяти. Этот файл используется и симулятором для моделирования конфигурации памяти системы и для установки размещения памяти целевой системы эмулятором. Системный конфигуратор выводит сообщения об ошибках, если они обнаружены, если они не обнаружены, показывает на экране полную архитектуру. Если необходимо обратится к полной архитектуре с целью отладки или документирования, вам потребуется перенаправить этот вывод в файл, используя действующую систему команд операционной системы.

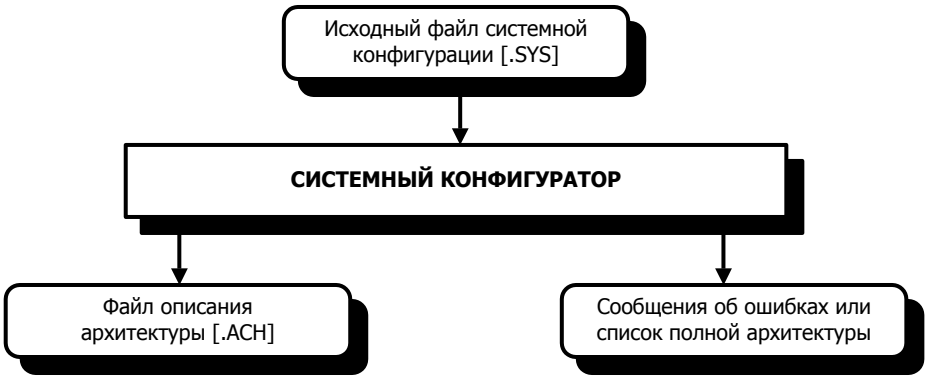

Рис.2.2. Вход/выход системного конфигуратора.

2.1.1. Регистры управления отображенные в памяти

Все процессоры ADSP-21xx, кроме ADSP-2100, имеют набор *регистров управления отображенных в памяти* (*memory-mapped control registers*), которые конфигурируют различные режимы работы процессора. Эти регистры расположены в скрытой части внутренней памяти. Это пространство памяти находится в верхней части 1K внутренней памяти данных по адресам 0x3C00-0x3FF. Вы не можете объявлять сегмент памяти в этом пространстве, и управляющие регистры не могут быть определены во входном файле системного конфигуратора. Вместо этого, каждый регистр должен быть инициализирован в вашей программе на языке ассемблер записью слов данных по присвоенному адресу. (Запись и чтение в регистры разрешены, даже если этот сегмент памяти не может быть объявлен директивой системного конфигуратора `.SEG`). Адрес и формат каждого регистра управления приведены в приложении E этого руководства и в описании регистров управления/состояния *ADSP-2100 Family User's Manual.*

2.2. Запуск системного конфигуратора

Для запуска системного конфигуратора наберите:

```
BLD21 filename[.ext] [-c]
```

где `filename` это ваш исходный файл системной конфигурации. Имя файла может иметь расширение, если оно отсутствует, системный конфигуратор добавляет по умолчанию расширение `.SYS`. Системный конфигуратор создает выходной файл описания архитектуры `filename.ACH`, использующий имя вашего входного файла .

Существует единственный дополнительный ключ включаемый в командную строку вызова системного конфигуратора. Ключ `-c` делает системный конфигуратор зависимым к регистру, различая использование верхних или нижних регистров (прописных или строчных) символов. Этот ключ предназначен для совместимости с компилятором семейства ADSP-2100.

Если ключ `-c` не используется, то выход системного конфигуратора составляют символы верхнего регистра. Вы должны использовать это ключ для сохранения введенных в нижнем регистре символов (строчных букв). При использовании этого ключа ассемблер различает регистр, как это требуется при ассемблировании скомпилированного C кода. Если вы обратитесь (в ассемблерном коде) к сегменту памяти в нижнем регистре написания, и ассемблер запущен в режиме зависимости к регистру, то имя сегмента не будет распознано до тех пор, пока системный конфигуратор не будет запущен с ключом `–c`. Если вы забыли синтаксис командной строки системного конфигуратора, наберите:

```
BLD21 -help
```

Этот ключ позволяет просмотреть список возможных команд. Ключ `–help` работает во всех программах средств разработки.

2.3. Использование символов и зарезервированные слова

В файле описания конфигурации вы назначаете символические имена целевой системе, сегментам памяти и портам ввода/вывода отображенным в памяти. Вы можете использовать имена сегментов памяти в ваших программах, чтобы назначать фрагменты программ и данных этим сегментам. Это назначение проходит от ассемблера к редактору связей и определяет точное размещение вашей программы в памяти. Все символические имена должны быть уникальны. Символическое имя это строка из букв, цифр и подчеркнутых знакомест, или подчеркнутое знакоместо перед первым символом. Небольшая группа символов резервируется для использования в качестве ключевых слов системного конфигуратора - вы не можете использовать эти символы в ваших файлах описания конфигурации. Таблица 2.1 приводит ключевые слова системного конфигуратора.

ABC	CODE	PM	DM
ADSP2100	DATA	PORT	ROM
ADSP2101	CONST	SEG	RAM
ADSP2105	ENDSYS	SYSTEM	BOOT
ADSP2111	ADSP2150	MMAP0	MMAP1

Таблица 2.1. Ключевые слова системного конфигуратора.

Ключевые слова ассемблера представлены в главе 3, они также зарезервированы для использования в программах средств разработки. Пожалуйста проверьте этот список перед тем, как вы выберете имена для ваших системных компонент. При использовании зарезервированных слов редактор связей выдаст сообщение об ошибке.

2.4. Файл системной конфигурации

Исходный файл системной конфигурации определяет соотношение памяти программ и данных в вашей системе. Для процессоров с памятью начальной загрузки файл объявляет также каждую страницу памяти начальной загрузки, которая будет использоваться. Комментарии заключаются в скобки {} и могут быть расположены в любом месте файла. Вложенные комментарии не разрешены.

Собственный файл можно создать с помощью любого простого текстового редактора. Не используйте текстовые процессоры, например Microsoft Word, которые вставляют специальные управляющие коды.

2.4.1. Пример файла системной конфигурации ADSP-2100

На Рис.2.3 приведен пример исходного файла системной конфигурации для ADSP-2100. (Термин «ADSP-2100» используется для определения ADSP-2100 и ADSP-2100A).

```
.SYSTEM fir_system;                          {системное имя}
.ADSP2100;                                   {определение процессора}
.SEG/PM/ROM/ABS=0/CODE        prog_mem [4096];  {программа}
.SEG/PM/RAM/ABS=4096/DATA     coeff_table [15]; {таблица коэффициентов}
.SEG/DM/RAM/ABS=0/DATA        delay_line [15];  {данные}
.PORT/DM/ABS=16382            ad_sample;        {порт отраженный в памяти}
.PORT/DM/ABS=16383            da_data;          {порт отраженный в памяти}
.ENDSYS;
```

Рис.2.3. Файл системной конфигурации ADSP-2100.

Первая директива в файле .SYSTEM. Она назначает имя fir_system описанию архитектуры и обозначает начало файла. Команда .ADSP2100 идентифицирует тип процессора, в приведенном примере это процессор ADSP-2100. Данная команда обязательна.

Директивы .SEG объявляют сегменты системной памяти и их характеристики. Сегменты памяти могут быть объявлены в любом порядке. В этом примере объявлены три сегмента. Первый, prog_mem - это сегмент размером 4 Кслова, который будет хранить код программы. Сегмент coeff_table - это блок размером 15 слов памяти программ объявлен для хранения данных; данные устанавливают коэффициенты КИХ-фильтра. Третий сегмент, delay_line, требуется для сохранения промежуточных данных алгоритма фильтрации; этот сегмент находится в памяти данных ADSP-2100.

Две директивы .PORT объявляют порты ввода/вывода отраженные в памяти. Имена портов ad_sample и da_data предполагает внешнюю связь с АЦП и ЦАП. Порты расположены в памяти данных с абсолютными адресами, заданными параметром ABS. Имена портов становятся символами, которые могут использоваться в программе для чтения и записи данных. Последнее объявление в файле системной конфигурации это директива .ENDSYS. Системный конфигуратор завершает свою работу при достижении директивы .ENDSYS.

2.4.2. Пример файла системной конфигурации ADSP-2101

На Рис.2.4 приведен пример исходного файла системной конфигурации для ADSP-2101. Первая директива в файле это директива .SYSTEM. Она присваивает имя fir_system описанию архитектуры и обозначает начало файла. Команда .ADSP2101 идентифицирует тип процессора, здесь подразумевается процессор ADSP-2101. Данная команда обязательна.

Директива .MMAP0 определяет в этой системе состояние вывода MMAP на ADSP-2101. Определение MMAP как 0 означает, что программа в памяти начальной загрузки должна быть загружена во внутреннюю память программ начиная с адреса 0x0000.

Директива .SEG объявляет сегменты системной памяти и их характеристики. Сегменты памяти могут быть объявлены и другим способом. В этом примере сегменты объявлены в полной конфигурации внешней и внутренней памяти программ и данных ADSP-2101.

```
.SYSTEM fir_system;                                  {системное имя}
.ADSP2101;                                           {определение процессора}
.MMAP0;                          {загрузка программы из памяти начальной загрузки}
.SEG/PM/ROM/BOOT=0 boot_mem [2048];  {страница блока памяти начальной загрузки}
.SEG/PM/RAM/ABS=0/CODE/DATA    int_pm [2048];    {внутренняя память программ}
.SEG/PM/RAM/ABS=2048/CODE/DATA ext_pm [14336];   {внешняя память программ}
.SEG/PM/RAM/ABS=0/DATA         ext_dm [14336];   {внешняя память данных}
.SEG/PM/RAM/ABS=0/DATA         int_dm [1024];    {внутреняя память данных}
.ENDSYS;
```

Рис.2.4. Файл системной конфигурации ADSP-2101.

(**Примечание**: обращение к памяти программ и данных, не включает память начальной загрузки, которую следует рассматривать как уникальную область памяти в системной архитектуре). Сегмент boot_mem это сегмент из 2 Кслов одной страницы памяти. Если система основана на ADSP-2105, размер сегмента должен быть 1К (или меньше) .

Объявление int_pm идентифицирует внутреннюю (расположенную на чипе) память программ размером 2 Кслова, начинающейся с адреса 0x0000. В ADSP-2101 (также как и в ADSP-2105, ADSP-2111, ADSP-21msp50) эта память может хранить как программу, так и данные и должна быть объявлена.

Следующая строка объявляет ext_pm, как сегмент размером 14 Кслов внешней памяти программ, начинающейся с адреса 2048, которая может содержать программу и данные. Следующая строка объявляет ext_dm как сегмент размером 14 Кслов внешней памяти данных, начинающейся с адреса 0. Предпоследняя строка объявляет int_dm как сегмент размером 1 Кслов внутренней памяти данных, начинающейся с адреса 14336. Верхние 1К памяти данных резервируется под регистры управления, отраженные в памяти и не могут быть объявлены как сегмент.

Последняя строка в файле системной конфигурации включает директивы .ENDSYS. Системный конфигуратор завершает свою работу при достижении директивы .ENDSYS.

2.5. Директивы системного конфигуратора

Этот раздел описывает директивы системного конфигуратора и их синтаксис. Формат некоторых директив включает аргументы и параметры. Параметры следуют сразу за директивой и отделяются косой чертой, /, слешем; аргументы следуют за параметрами. Основная форма директивы выглядит следующим образом:

```
.DIRECTIVE /параметр/параметр/..аргумент;
```

2.5.1. Название системы (.SYSTEM)

Директива .SYSTEM должна быть первой командой в исходном файле системной конфигурации. Вы присваиваете имя вашей системе ADSP-21xx, указывая его в аргументе этой директивы. Системное имя будет показано в программе моделирования. Директива .SYSTEM имеет формат:

```
.SYSTEM имя_системы;
```

Директива .ENDSYS должна быть последней командой в файле. Системный конфигуратор останавливает свою работу на директиве .ENDSYS. Директива .ENDSYS имеет форму:

```
.ENDSYS;
```

2.5.2. Определение процессора (.ADSP21XX)

Эта директива определяет какой процессор семейства ADSP2100 используется в вашей системе. Эта информация передается редактору связей и симулятору через выходной файл с расширением .ACH.

После этого редактор связей в состоянии определить место расположения программы и данных в соответствии с организацией памяти для каждого процессора. Эта директива принимает одну из следующих форм:

```
.ADSP2100;       для ADSP-2100 и ADSP-2100A
.ADSP2101;
.ADSP2105;
.ADSP2111;
.ADSP2150;       для ADSP-21msp50 и ADSP-21msp55
.ADSP2151;       для ADSP-21msp51 и ADSP-21msp56

.ADSP2101MV;     для вариантной системы ADSP-2101 (ADSP-2115)
.ADSP2101P;      для системы со страничной памятью ADSP-2101
```

Когда симулятор ADSP-21msp50 загружается с файлом архитектуры ADSP-21msp51, он автоматически конфигурирует себя для распределения памяти ADSP-21msp51. Если используется файл архитектуры ADSP-21msp51 и разряд ROMENABLE установлен в 1, программа моделирования конфигурирует память программ PM[0x800] - PM[0x1000] как ПЗУ. Разряд ROMENABLE отображается в окне регистров управления.

2.5.3. Вывод MMAP (.MMAP)

Эта директива используется только для тех процессоров семейства ADSP-2100, которые обладают встроенной памятью, памятью начальной загрузки и выводом MMAP (т.е. все за исключением ADSP-2100). Директива определяет логическое состояние вывода MMAP процессора в вашей целевой системе. Эта директива принимает одну из двух форм:

```
.MMAP0                вывод MMAP поддерживает низкий уровень
.MMAP1                вывод MMAP поддерживает высокий уровень
```

Если используется .MMAP0, загрузка программы из памяти начальной загрузки происходит после перезагрузки и начинается с адреса 0x000 внутренней памяти программ. Если используется .MMAP1, загрузка программы из памяти начальной загрузки недоступна и внутренняя память программ отображается на верхние адреса пространства памяти программ. Когда эта директива пропускается, программа моделирования принимает по умолчанию значение MMAP=1.

2.5.4. Объявление сегмента памяти (.SEG)

Директива .SEG определяет специальную секцию системной памяти и описывает ее атрибуты. Не существует разделения памяти по умолчанию – вы должны определить системную память с помощью директив .SEG самостоятельно. Эта информация передается редактору связей, симулятору и эмулятору через выходной файл с расширением .ACH. Директива .SEG имеет форму:

```
.SEG/параметр/параметр .имя_сегмента[размер];
```

Сегменту присваивают символическое имя имя_сегмента. Присвоенное имя позволит вам точно разместить фрагменты программы и данных в памяти. На языке ассемблер это достигается параметром SEG. Вы должны указать длину сегмента внутри скобок. Эта величина интерпретируется как количество слов (16-разрядных данных или 24-разрядных инструкций) в сегменте. Размер сегмента памяти данных в битах составляет 2 x числа слов, размер сегмента памяти программ в битах составляет 3 x числа слов. Размер сегмента памяти начальной загрузки в байтах составляет 4 x числа слов, из-за расширения памяти начальной загрузки дополнительным байтом для достижения выравнивая слова.

Для директивы .SEG необходимо указать два параметра:

```
PM или DM или BOOT=0-7          пространство памяти
RAM или ROM                     тип памяти
```

Четыре параметра являются необязательными:

```
ABS=address                     абсолютный стартовый адрес
DATA или CODE или DATA/CODE     что хранится в сегменте
EMULATOR или TARGET             распределение памяти для эмулятора
INTERNAL                        расположение в памяти процессора
                                ADSP-2101 с вариантной памятью
```

Параметры PM/DM/BOOT показывают какой сегмент пространства памяти занят памятью программ, памятью данных, памятью начальной загрузки.

Оставшиеся опции определяют тип памяти, начальный адрес сегмента, тип содержимого (данные и/или программы) и распределение памяти эмулятора. Если вы используете один из эмуляторов ADSP-21xx, обратитесь к разделу «Распределение сегментов для эмулятора».

Каждое пространство памяти адресуется раздельно: адрес 0x10 в памяти программ отличается от адреса 0x10 в памяти данных. Сегмент памяти программ PM может хранить только CODE (программы), только DATA (данные) или одновременно программы и данные. Если вы не укажете ни одну из опций, по умолчанию принимается хранение программ. Для сегмента PM содержащего программу и данные должны быть указаны оба параметра. Процессоры ADSP-21xx, требующие передачи данных из или в память программ должны быть объявлены с сегментами, которые имеют параметр данных. Если ваша система требует, чтобы исполняемая программа была записана или прочитана процессором, сегменты, чтобы быть доступными, должны быть объявлены с обоими параметрами CODE (программа) и DATA (данные) опциями.

Сегменты памяти данных DM должны быть только DATA; это значение принимается по умолчанию, если параметр DATA пропущен. Если сегменту DM присвоен параметр CODE выдается сообщение об ошибке. Сегменты памяти начальной загрузки принимают по умолчанию параметры CODE и DATA, с тех пор как в большинстве систем используется для хранения и данных, и программ, поэтому эти параметры могут пропущены. Параметр памяти начальной загрузки должен определять только номер страницы, например, BOOT=0. Необходимо разделить объявление сегмента для каждой страницы памяти начальной загрузки вашей системы.

Система может иметь до 8 страниц памяти начальной загрузки с номерами от 0 до 7. Каждая страница памяти начальной загрузки для ADSP-2101, ADSP-2111, ADSP21ms50 может сохранять до 2К программ и данных. Каждая страница памяти начальной загрузки для ADSP-2105, ADSP-2115 может сохранять до 1К слов. Параметр ABS для страниц памяти начальной загрузки не используется – системный конфигуратор самостоятельно присваивает соответствующие адреса.

2.5.4.1. Пример директивы .SEG

Пример

```
.SEG/PM/RAM/ABS=0/CODE/DATA  restart [2048];
```

объявляет сегмент памяти программ ОЗУ, называемый restart, который расположен по адресу 0. Сегмент может содержать до 2048 слов программ и данных.

Пример

```
SEG/ROM/BOOT/=0  boot_mem[1536];
```

объявляет сегмент памяти начальной загрузки boot_mem, который расположен на странице номер 0 памяти начальной загрузки (автоматически соответствует адресу 0 в памяти начальной загрузки). Длина сегмента составляет 1536 слов. Сегменты памяти начальной загрузки должны всегда быть объявлены как ROM (ПЗУ).

2.5.4.2. Распределение сегментов для эмулятора

Системный конфигуратор позволяет вам заранее установить распределение памяти для эмулятора ADSP-21xx через файл .ACH. Для этой цели используются два дополнительных параметра директивы .SEG:

```
      /EMULATOR        размещает сегмент для оверлейной памяти эмулятора
или
      /TARGET          размещает сегмент памяти целевой платы
```

Эти установки настраивают распределение памяти эмулятора, когда программа эмулятора будет запущена. Сегменты должны быть разделены на блоки по 1К или больше. После запуска эмулятора вы сможете изменить распределение памяти командами эмулятора.

Если вы используете эмулятор в автономном режиме (без целевой платы), вы должны распределить *все* сегменты памяти для эмулятора. Можно распределить все сегменты для эмулятора в течение начальной стадии интеграции аппаратной/аппаратно-программной части системы. Для разрешения вопросов обратитесь к *ADSP-21xx Emulator Manual*.

2.5.5. Порты ввода/вывода отображенные в памяти (.PORT)

Директива .PORT объявляет порт ввода/вывода отображенный в памяти. Вы должны присвоить уникальное имя и адрес каждому порту в вашей системе. Порты могут быть определены в памяти данных или памяти программ. Для процессоров с внутренней памятью, портам могут быть присвоены адреса только во внешней памяти.

Директива .PORT принимает одну из двух форм:

```
      .PORT/DM/ABS=address  port_name;
или
      .PORT/PM/ABS=address  port_name;
```

Опция DM показывает, что порт отображен в памяти данных; PM показывает, что порт отображен в памяти программ. Если параметр не задан, по умолчанию принимается DM.

Порт располагается по абсолютному адресу, который вы определите (параметром ABS), и присваивается символическому имени port_name. Этот символ могут использовать инструкции ассемблера для доступа к порту, например:

```
      .PORT/DM/ABS=0x0400   ad_sample;
```

Объявляет имя порта ad_sample, который отображен в памяти данных по адресу 0x4000 (шестнадцатеричный) или 1024 (десятичный). Ассемблерная программа обращается к этому символу, чтобы представить его редактору связей базирующемуся на содержании файла описания системной конфигурации .ACH. Отображение порта в памяти данных позволяет 16-разрядное чтение/запись, в то время как отображение порта в памяти программ допускает 16- или 24-разрядную передачу. (Обратитесь к *ADSP-2100 Family User's Manual* для описания 24-разрядной передачи данных).

2.5.6. Константы системного конфигуратора (.CONST)

Директива .CONST определяет константы системного конфигуратора. Однажды объявив символическую константу, вы можете использовать её вместо реального числа. Это символическое определение распознается только системным конфигуратором, поэтому не переходит на ассемблер и программу моделирования. Директива .CONST имеет форму:

```
.CONST constant_name=константа или выражение, ...;
```

В выражениях могут использоваться только арифметические или логические операции из двух или более целочисленных констант; использование символов не разрешено.

Одна директива .CONST может содержать одно или более объявлений констант на одной строке, разделенных запятыми. Список объявлений не может быть продолжен на следующей строке.

Чтобы приравнять символ *taps* к 15, для примера, вы должны написать следующую директиву .CONST:

```
.CONST taps=15;
```

2.6. Рассмотрение особенностей процессоров

Благодаря некоторым уникальным характеристикам ADSP-2100, ADSP-2105 и ADSP-2115, следующие руководящие указания должны быть приняты во внимание при написании файла системной конфигурации для этих процессоров.

2.6.1. Системы ADSP-2100

Для ADSP-2100 возможны две различные конфигурации памяти программ:

- 16К слов смешанных программ и данных

или

- 32К слов - 16К программ, 16К данных.

Расширенная конфигурация 32К требует использования вывода процессора PMDA как дополнительной (высшего порядка) адресной линии для памяти программ. Если эта конфигурация работает, нижние 16К должны быть только программными. Сегменты PM, объявленные в этой области, должны иметь параметр только CODE. Верхние 16К пространства памяти должны быть только данными. Сегменты PM, объявленные в этой области, должны иметь параметр только DATA.

Если ваша система ADSP-2100 включает 32К расширенного пространства памяти программ, системный конфигуратор будет выдавать сообщение об ошибке при попытке объявить сегмент PM с обоими параметрами CODE и DATA.

2.6.2. Системы ADSP-2105 и ADSP-2115

Поскольку ADSP-2105 и ADSP-2115 имеют половину внутренней памяти ADSP-2101, пространство внутренней памяти, которое может быть объявлено, ограничено системным конфигуратором. Вы не можете объявить сегменты памяти в любых порциях в диапазоне следующих адресов:

Внутренние адреса памяти данных	14848 – 15359	(0x3A00 - 0x3BFF).
Внутренние адреса PM (MMAP=0)	1024 – 2047	(0x3C00 - 0x07FF).
Внутренние адреса PM (MMAP=1)	15360 – 16383	(0x3C00 - 0x3FFF).

Эти диапазоны соответствуют верхним 1K внутренней памяти `PM` ADSP-2101 и верхним 1/2K внутренней памяти `DM` ADSP-2101, которые отсутствуют в ADSP-2105/ADSP-2115. Поскольку редактор связей выделяет информацию из файла `.ACH`, выработанного системным конфигуратором, он не размещает программы или данные в этих диапазонах памяти.

Симулятор ADSP-2101 используется для моделирования систем ADSP-2105 и ADSP-2115. При запуске этого симулятора с архитектурным файлом ADSP-2105 или ADSP-2115 (процессор с вариантной памятью `.ADSP2101MV`), симулятор конфигурируется на различное количество внутренней памяти. Часть памяти на чипе ADSP-2101, которая не существует в этих процессорах будет показана как несуществующая.

2.6.2.1. Создание 1K страниц начальной загрузки

Так как ADSP-2105 и ADSP-2115 имеют страницы начальной загрузки объемом 1K, ваше объявление страницы должно специфицировать сегмент размером 1024 слов (или меньше). Например::

```
.SEG/BOOT=0/ROM  page_0[1024];
```

При последующем использовании программы разделителя для подготовки исходного файла для программатора, необходимо включить ключ `-bs` для создания страниц размером 1K (смотрите главу 5, Разделитель программ для записи в ППЗУ).

2.6.2.2. Замена ADSP-2105/ADSP-2115 на ADSP-2101

ADSP-2105 и ADSP-2115 являются совместимыми по выводам с ADSP-2101, позволяя делать прямую замену компоненты в вашей системе. Существует несколько деталей, касающихся такой замены, которые должны быть приняты во внимание при проектировании.

Чтобы произвести замену без модификации оборудования, необходимо изначально разработать аппаратную и аппаратно-программную часть для использования загрузочных страниц размером 2K (которые хранят только 1024 слов). Для этого нужно объявить сегменты страниц начальной загрузки размером 2048 слов:

```
.SEG/BOOT=0/ROM  page_0[2048];
```

Поскольку ваша программа для ADSP-2105/ADSP-2115 должна быть загружена в 1K страницы, система не будет использовать 1K памяти между каждыми страницами программ/данных. Процессор будет правильно загружать каждую 1K страницу программы, игнорируя неиспользуемые пустые сегменты между ними.

Другими словами, загружаемые страницы размером 2K будут содержать программы/данные в нижней половине каждой страницы.(адреса слов 0-1023), а верхняя половина будет пустая. Когда вы замените систему на ADSP-2101, вы сможете использовать разделитель программ для записи в ППЗУ для создания страниц начальной загрузки размером 2K.

2.7. Процессоры ADSP-2101 с вариантной памятью

Процессоры с вариантной памятью, такие как ADSP-2115, это производные ADSP-2101, которые содержат различные конфигурации памяти, расположенной на кристалле. Чтобы поддержать моделирование процессоров с вариантной памятью, программное обеспечение средств разработки позволяет легко конфигурировать системный архитектурный файл и симулятор для процессора, который вы используете.

Системы, основанные на процессорах с вариантной памятью, могут быть определены с любым количеством памяти программ (PM) 0 – 16K слов и с любым количеством памяти данных (DM) 0 - 15K слов. Часть PM и DM пространства памяти могут быть свободно определены как ROM и/или INTERNAL (т.е. расположенная на кристалле). Эти определения вводятся в файле системной конфигурации .SYS. После обработки вашего .SYS файла системным конфигуратором, создается файл .ACH, который используется редактором связей и симулятором ADSP-2101.

Необходимо предпринять следующие действия для моделирования процессора ADSP-2101 с вариантной памятью:

1. Используйте следующую директиву системного конфигуратора в вашем .SYS файле системного конфигуратора:

   ```
   .ADSP2101MV
   ```

2. Для любых ваших сегментов памяти, расположенных в вариантной памяти на кристалле, используйте параметр INTERNAL в соответствующей директиве .SEG:

3. Для любых из ваших сегментов памяти ПЗУ, используйте параметр /ROM в соответствующей директиве .SEG.

4. Ассемблируйте, связывайте и моделируйте вашу программу обычным способом. Редактор связей и симулятор ADSP-2101 прочитают архитектурный файл .ACH, созданный системным конфигуратором, определяя внутренние и/или ПЗУ сегменты памяти соответственно.

2.8. Проектирование систем со страничной памятью

Программное обеспечение средств разработки позволяет проектировать ADSP-2101 системы, которые адресуются большим пространством внешней памяти, применяя схему со страничной памятью данных. Такое расширение можно применить только к памяти данных.

Рис.2.5 показывает пример систем такого типа, где память данных ADSP-2101 расширена тремя дополнительными страницами. Страница 0 - это стандартное пространство 16К памяти данных ADSP-2101. В системе со страничной памятью, страница 0 разделена на *пространство данных*, *пространство ввода/вывода* и память данных на чипе (адреса 0x3800-3FFF).

Величина `PAGESIZE` (размер страницы), которую вы должны обозначить в системном конфигураторе, определяет границу между *пространством данных* и *пространством ввода/вывода*. Величина `DMIOEND`, которую также необходимо определить и которая не может быть больше чем 0x37FF, это последний адрес *пространства ввода/вывода*. *Пространство ввода/вывода* содержит порты отраженные в памяти и специальную ячейку расположенную в памяти, называемую *регистр страницы памяти данных*.

Модули программ, буферы данных и переменные, которые вы хранятся в страничной памяти должны быть заключены внутри собственной страницы и не пересекать ее границы.

2.8.1. Функции системного конфигуратора для страничной памяти

Для создания файла архитектуры для системы со страничной памятью, необходимо использовать директиву `.ADSP2101P`. Например, чтобы организовать 4К страницы, нужно использовать следующие параметры в вашем `.SYS` файле :

```
.ADSP2101P/PAGESIZE=4096;          {страничная система памяти}
                                   {размер страницы 4К}
```

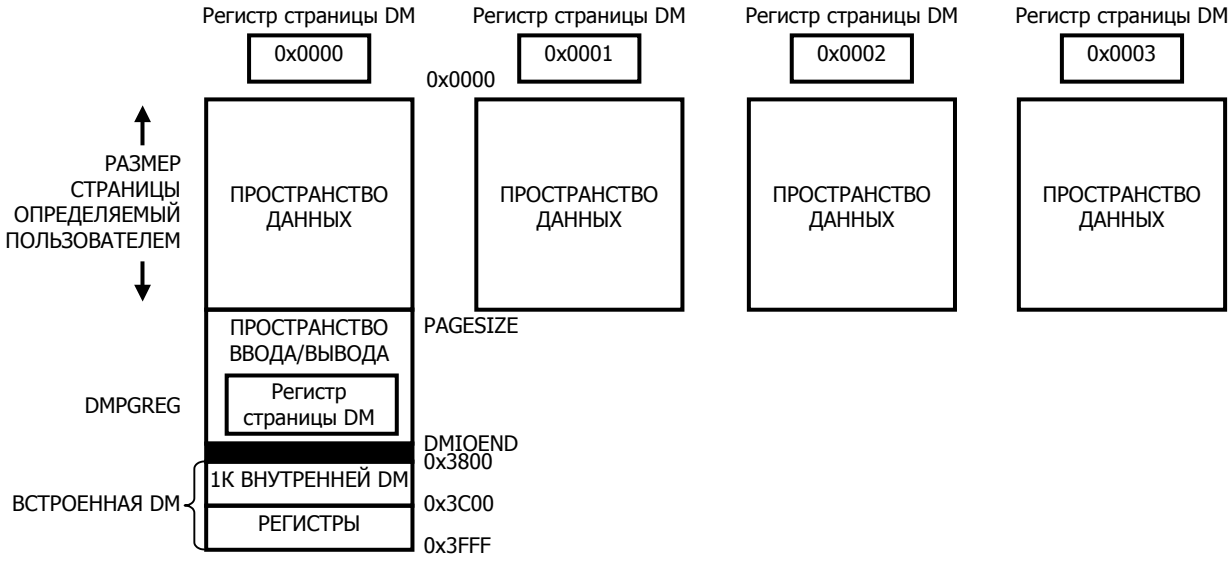

Рис.2.5. Страничная система памяти данных ADSP-2101.

Параметр PAGESIZE определяет количество слов пространства данных в каждой странице памяти. По умолчанию размер страницы равен 8192 словам. Вы должны также использовать системный конфигуратор, чтобы определить для вашей системы регистр страницы памяти данных. Этот регистр используется для адресации на различные страницы. В примере, показанном на Рис.2.5, значение регистра страницы памяти данных изменяется от 0 до 3.

Регистр страницы памяти данных определяется директивой .PORT и должен быть назван DMPGREG. Чтобы определить регистр страницы памяти данных с адресом 8192 (0x2000), необходимо произвести следующую запись:

```
.PORT/DM/ABS=0x2000 DMPGREG;      {регистр страницы памяти данных}
```

Регистр страницы памяти данных реализуется как регистр отраженный в памяти вашего системы. Значение регистра используется как старшие разряды адреса всей памяти данных; эти разряды производят выбор между различными страницами.

2.8.2. Использование имен сегментов для страниц

Специальный синтаксис параметра /ABS (директивы .SEG системного конфигуратора) позволяет вам определить и назвать сегмент памяти, который размещен на отдельной странице памяти данных. Формат этой директивы:

```
.SEG/параметры/ABS=pq#:addr  имя_сегмента[длина_сегмента];
```

Этот синтаксис определяет номер страницы и стартовый адрес сегмента (в десятичном формате от 0 до 16,384). Параметры DM, RAM/ROM и DATA должны быть также указаны. Например, чтобы определить имена сегментов для страниц памяти данных, показанных на Рис.2.5 должны быть использованы следующие директивы:

```
.SEG/DM/RAM/DATA/ABS=0:0  page0[4096];
.SEG/DM/RAM/DATA/ABS=1:0  page1[4096];
.SEG/DM/RAM/DATA/ABS=2:0  page2[4096];
.SEG/DM/RAM/DATA/ABS=3:0  page3[4096];
```

Ваш исходный C код и/или исходные ассемблерный модули могут быть размещены в одном из этих определенных на страницах сегментов, используя параметр ассемблера .SEG или ключ C компилятора -DMSEG.

2.8.3. Функции ассемблера для страничной памяти

Ассемблер распознает особую директиву, которая указывает на системы со страничной памятью. Эта директива называется .PAGE, и она должна быть применена во всех исходных модулях на языке ассемблер, которые составляют части системы со страничной памятью:

```
.PAGE;
```

Новый оператор ассемблера PAGE *buffer_name* используется для получения номера страницы (старших разрядов адреса) буфера данных или переменной:

```
AX0=PAGE array0;          (получить номер страницы array0)
```

Эта инструкция определяет номер страницы буфера array0 и загружает его в регистр AX0. Заметьте, что оператор PAGE работает подобно оператору получения указателя адреса (^) и оператору определения размера (%).

2.8.4. Функции С компилятора для страничной памяти

С компилятор распознает несколько параметров в командой строке, которые указывают на поддержку системы со страничной памятью. При компиляции программы для системы со страничной памятью используют ключ -FARDATA:

```
CC21 source.c -FARDATA
```

Ключ -FARDATA сообщает компилятору, что переменные и множества в файле source.c будут использоваться в системе со страничной памятью, и поэтому для этих переменных/множеств должны быть создана постранично адресуемая информация (т.е. использующая старшие разряды адреса). Для переменных и множеств, размещенных в DM, компилятор генерирует коды, которые включают номер страницы, содержащейся в регистре страницы памяти данных DMPGREG, предварительно определенном в системном конфигураторе. Если вы используете ключ -FARDATA для того чтобы сохранить данные в страничной памяти, вы должны определить имена сегментов, в которых ваши данные будут размещены. Ключ -FARDATA указывает компилятору разместить все данные из файла source.c в сегменте DM по умолчанию, называемом DDEFAULT. Необходимо определить DDEFAULT в системном конфигураторе перед компиляцией и редактированием связей. Для системы, показанной на Рис.2.5, страница 0 может быть определена как сегмент DDEFAULT следующей командой системного конфигуратора :

```
.SEG/DM/RAM/DATA/ABS=0:0  DDEFAULT[4096];
```

Ключ компилятора -DMSEG используется для переопределения размещения данных. Например, чтобы разместить данные DM из src.c в сегмент памяти, названный table3 (вместо DDEFAULT), необходимо указать в командной строке следующие параметры:

```
cc21 src.c -fardata -dmseg table3
```

Ключ -FARDATA оказывает несколько другое воздействие на компилируемый код и выполнение программы:

1. Стек исполняемой программы должен быть размещен во внутренней памяти ADSP-2101. По умолчанию стек располагается в памяти данных, если не используется ключ компилятора -pmstack.

2. Компилятор назначает страницу памяти данных каждой откомпилированной функции. При выполнении, она получает доступ только к тем данным, которые расположены на этой странице.

3. Часть функций С библиотеки рабочих программ используют небольшую часть памяти - эта память будет размещена во внутренней памяти ADSP-2101.

2.8.5. Использование адресов страниц в программе моделирования

При работе с симулятором ADSP-2101, когда моделируется система со страничной памятью, вы можете специфицировать адреса памяти информацией о странице. Синтаксис для адресации страничной памяти включает номер страницы и адрес (в десятичном формате от 0 до 16.383):

синтаксис	*пример*
`DM(pg#:addr)`	`dm(0:8193)`