Редактор связей

4

4.1. Введение

Редактор связей (компоновщик) ADSP-21xx генерирует исполняемую программу путем связывания отдельно ассемблированных модулей. Основным выходом редактора связей является файл отображения памяти с расширением .EXE. Этот файл загружают в программу моделирования для отладки. После того, как программа полностью отлажена, применяют программу разбиения памяти для программатора ППЗУ.

Как было описано в предыдущей главе, ассемблер обрабатывает каждого модуль исходного кода и создает объектный файл (.OBJ), файл кода (.CDE) и файл инициализации (.INT). Объектный файл содержит информацию по размещению в памяти и символьную информацию, в то время как файл кода содержит коды операций ADSP-21xx с помеченными неразрешенными символами. Файл инициализации содержит информацию, относящуюся к переменным данным и буферам. Инициализация данных должна обеспечиваться файлом данных, указанным с помощью директивы ассемблера .INIT. Редактор связей читает данные из этого файла и объединяет их в файл .EXE. Изменения в инициализируемых данных потребует повторного вызова редактора связей. На следующей странице Рис.4.1. показывает входные и выходные файлы для редактора связей.

Редактор связей сканирует каждый ассемблированный модуль и разрешает обращения между модулями к глобальным и внешним символам. Он назначает (раздает) адреса перемещаемых программ и фрагментов данных. Редактор связей читает также файл описания архитектуры .ACH для создания карты системной памяти и размещения в ней программы и данных. Идентификация архитектурного файла для редактора связей происходит включением ключа -a в строке вызова программы.

Редактор связей может создавать три различных файла. Файл отображения памяти (*memory image file*) .EXE создается всегда – это исполняемая программа, которая содержит реальные коды операций и данные, которые должны быть расположены в памяти.

Вспомогательный файл распределения (*memory listing file*) .MAP суммирует информацию, относящуюся к созданной программе.

Вспомогательный файл таблицы символов (*symbol table file*) .SYM перечисляет все символы, встреченные редактором связей и их абсолютные адреса. Этот файл также используется программами моделирования (симуляторами) ADSP-21xx и эмуляторами.

©АВТЭКС Санкт-Петербург. http://www.autex.spb.ru, E-mail: info@autex.spb.ru

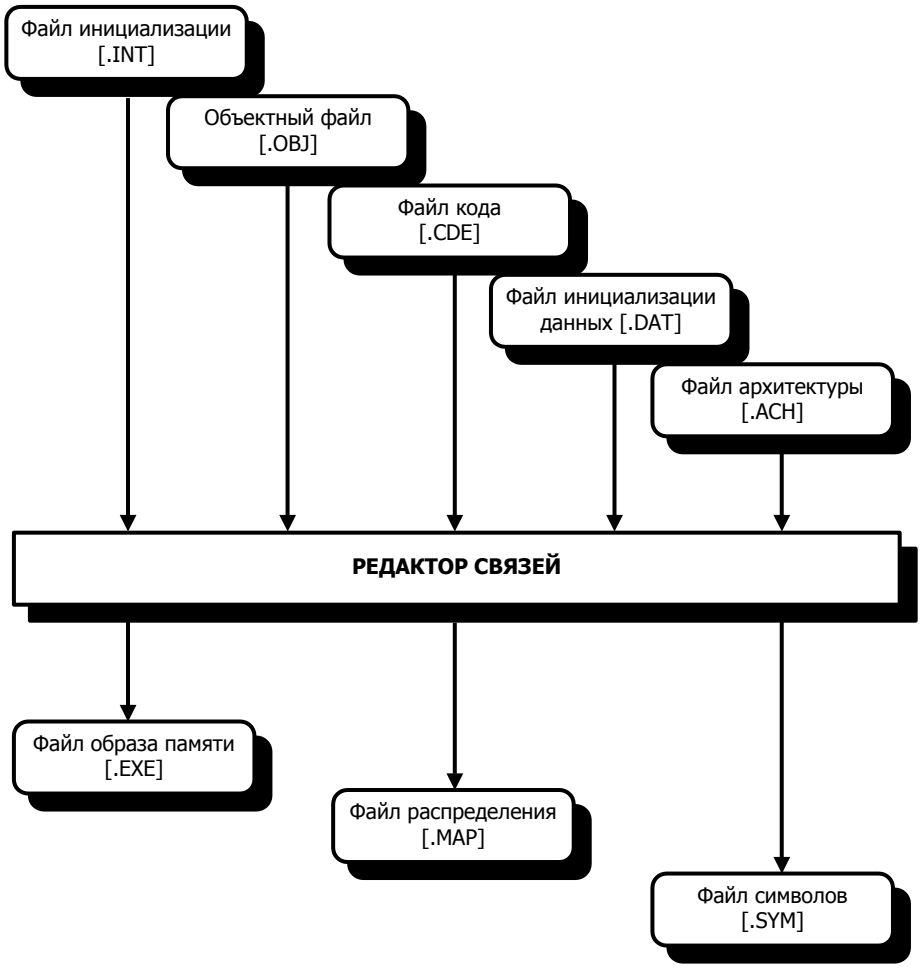

Рис.4.1. Вход/выход редактор связей.

Инициализирующие файлы данных (`.DAT`) не обозначены запуске редактора связей явно, поскольку они перечислены (директивой `.INIT`) в файле исходного кода. Файлы данных объединяют с помощью редактора связей. Если в файлах с данными сделаны изменения, необходимо просто запустить редактор связей заново. (**Примечание**: с тех пор как директива ассемблера `.VAR` используется для объявления как однословных переменных, так и многословных буферов данных, термин «буфер данных» включает в себя как переменные, так и буфера.)

©АВТЭКС Санкт-Петербург. http://www.autex.spb.ru, E-mail: info@autex.spb.ru

4.2. Запуск редактора связей

Редактор связей вызывают перечисляя файлы, которые должны быть скомпонованы:

```
LD21  файл1 [файл2...] [-ключ ... ]
```

Например:

```
LD21  mine subl subl2  -a archfile
```

Каждый входной файл должен содержать только один модуль. Если файлы отсутствуют в текущем каталоге вашей операционной системы, с каждым именем файла должен быть путь к нему. Имена файлов должны идентифицировать ассемблерные выходные файлы без расширений (т.е. `.CDE`, `.OBJ`, `.INT`). Выходные файлы редактора связей получают по умолчанию имя 210X. Вы можете переименовать эти файлы, используя ключ `-e` (смотрите раздел «Параметры ключей редактора связей»).

Редактор связей может быть также вызван с ключом `-i`, который называет отдельный файл, содержащий список файлов для компоновки:

```
LD21  -i файл_все  [-ключ ...]
```

В этом случае редактор связей читает дополнительный файл *файл_все*, который является простым текстовым файлом, содержащим в каждой строке одно имя файла.

Другие ключи управляют различными операциями редактора связей. Они могут быть введены как в верхнем, так и в нижнем регистре. Несколько ключей должны быть отделены друг от друга хотя бы одним пробелом.

Редактор связей размещает память для модулей в соответствии с порядком, в котором они были перечислены. Для модулей, которые содержат объявления циклических буферов данных, изменения в порядке входных файлов могут определить, сможет ли программа быть успешно скомпонована в доступном пространстве памяти. Поэтому:

Модули, содержащие циклические буферы различных размеров должны быть перечислены для редактора связей в порядке понижения размеров буферов.

Это позволит редактору связей размещать вначале буфера большего размера, которые имеют большие ограничения на допустимые базовые адреса.

Если вы забыли синтаксис для вызова редактора связей, наберите в командной строке:

```
LD21  -help
```

Эта команда покажет синтаксис вызова программы и покажет список доступных ключей.

Если вы ассемблировали исходную программу, сгенерированную С компилятором, редактор связей должен быть вызван с ключом `-c`. Для получения детальной информации обратитесь к *ADSP-2100 Family C Tools Manual* и *ADSP-2100 Family C Runtime Library Manual*.

©АВТЭКС Санкт-Петербург. http://www.autex.spb.ru, E-mail: info@autex.spb.ru

4.2.1. Размещение модулей на загрузочных страницах

Редактор связей предоставляет альтернативу ассемблерным директивам .SEG/BOOT для размещения модулей на страницах памяти начальной загрузки. Для этого в командном файле применяется специальный параметр. Чтобы заставить редактор связей разместить копию модуля на одной или более загрузочных страницах, необходимо перечислить имена файлов модулей следующим образом:

имя_файла B *hh*

Параметр *hh* представляет собой двухсимвольный шестнадцатеричный номер страницы памяти начальной загрузки. В этом 8-разрядном селекторе для страниц 0-7, любой разряд, установленный в «1» соответствует размещению модуля на этой странице:

$hh_{7\text{-}0}$ =

pg7	**pg6**	**pg5**	**pg4**	**pg3**	**pg2**	**pg1**	**pg0**

В следующем примере редактор связей разместит копии модуля subprog на загрузочных страницах с 0 по 4. Компоновщик вызывается с ключом -i, называя командный списочный файл. Заметьте, что селектор *hh* не требует шестнадцатеричного префикса «0x».

```
subprog  B  1F
```

4.2.2. Ключи редактора связей

Ключи редактора связей перечислены в таблице 4.1; часть из них требуют аргументы.

Ключ	*Воздействие*
-a *имя_файла*	указание файла описания архитектуры *имя_файла* .ACH
-c	создание стека для компилированных C программ (DM)
-dir *каталоги*;	указание директорий для поиска файлов библиотек
-dryrun	быстрый запуск для теста на ошибки (не создает .EXE файл)
-e *имя_файла*	присваивает выходным файлам имя *имя_файла*
-g	создает файл таблицы символов .SYM
-i *файл_все*	указание командного списочного файла
-lib	компоновка с C библиотекой рабочих программ (используется только с «-c»)
-p	размещает копию библиотечной подпрограммы на загрузочных страницах
-pmstack	перемещение C стека в память программ (PM) (используется только с «-c»)
-rom	используется ПЗУ версия C библиотеки программ (используется только с «-c»)
-s *размер*	создает динамическую память («кучу») C (используется только с «-c»)
-user *имя_файла*	поиск библиотечного файла, созданного утилитой построителя библиотеки LIB21
-x	создает файл распределения памяти .MAP

Таблица 4.1. Ключи редактора связей.

4.2.2.1. Указание файла описания архитектуры (-a)

Вы должны указать файл описания архитектуры .ACH с помощью ключа –a.. Имя файла может быть дано без расширения .ACH:

```
LD21  файл1  файл2  -a имя_файла
```

Редактору связей требуется информация об архитектуре системы, чтобы разместить программу и данные вашей программы в доступной системной памяти.

4.2.2.2. Создание стека исполняемой С программы (-c)

Ключ -c должен использоваться при компоновке программных модулей, сгенерированных компилятором семейства ADSP-2100. Эти программы требуют наличия стека исполняемой программы. Задание ключа -c приводит к следующему. Во-первых, редактор связей создает метку:

```
____top_of_ram          (метка с четырьмя символами подчеркивания)
```

которая присваивается самым верхним доступным адресам памяти данных (или память программ, если был задан ключ -pmstack). Во-вторых, редактор связей находит и подсоединяет исполняемый заголовок С программы, который представляет собой файл на языке ассемблере, требуемый для откомпилированных С программ. Метка ____top_of_ram используется исполняемой программой для расположения и инициализации стека в памяти процессора. Исполняемый заголовок программы устанавливает регистры, используемые для управления стеком и вызывает главную подпрограмму С программы. Стек не имеет ограничений по размерам; это позволяет увеличивать его (в направлении младших адресов).

Для каждого из процессоров ADSP-21XX используются различные заголовки исполняемой программы. Эти файлы поставляются с программным обеспечением компилятора С и имеют уникальное имя файла, указывающее с каким из процессоров его следует использовать. Необходимо выбрать соответствующие файлы для вашего системного процессора и переименовать их перед компоновкой в:

```
run_hdr.OBJ
run_hdr.CDE
run_hdr.INT
```

Редактор связей ищет файлы именно с этими именами. Вы можете также выбрать исходный ассемблерный файл .DSP для модификации. В этом случае вы должны отредактировать и заново ассемблировать модуль заголовка исполняемой программы перед компоновкой. Для получения детальной информации обратитесь к *ADSP-2100 Family C Tools Manual* и *ADSP-2100 Family C Runtime*.

Переменная окружения ADIRTH используется для поиска заголовка исполняемой программы. Редактор связей ищет этот файл в соответствии с назначенным путем. Вы должны установить переменную окружения, указывающую на каталог, в котором сохранен файл.

Например:

```
SET  ADIRTH=C:\ADI_DSP\LIB\
```

Обязательно должен присутствовать завершающий символ косой черты (слеш); не включайте дополнительные пробелы. Если редактор связей вызван с ключом `-c`, но не может найти файл заголовка исполняемой программы, это вызовет сообщение об ошибке.

4.2.2.3. Пути поиска для библиотечных подпрограмм (-dir)

Редактор связей позволяет вам использовать часто применяемые файлы с подпрограммами как библиотеки. Когда программа компонуется с вызовом одной из библиотечных подпрограмм, редактор связей автоматически разыскивает и присоединяет соответствующий файл. Необходимо указать редактору связей, где найти ваши библиотечные файлы, используя ключ `-dir` или переменную окружения `ADIL`. Редактор связей сначала осуществляет поиск по каталогам, указанным в ADIL, и затем смотрит аргументы ключа `-dir` (если необходимо). Например, чтобы установить переменную среды `ADIL` на каталог C:\DSP\LIBRARY\ вы должны ввести:

```
SET  ADIL=C:\DSP\LIBRARY\
```

Если перечисляются несколько путей DOS, они должны быть разделены точками с запятой. Обязательно должен присутствовать завершающий символ косой черты (слеш); не включайте дополнительные пробелы.

В операционной системе Unix на рабочих станциях Sun, аналогичная команда должна выглядеть следующим образом:

```
setenv  ADIL "/dsp/library/INCLUDE/"
```

4.2.2.4. Указание имен выходных файлов (-e)

Ключ `-e` позволяет указать имена выходных файлов редактора связей. Если этот ключ не используется, по умолчанию присваиваются следующие имена:

`210X.EXE, 210X.SYM и 210X.MAP.`

Пример использования ключа –e:

`LD21` *`файл1 файл2`* `-e` *`имя_файла`*

4.2.2.5. Компоновка С библиотеки рабочих программ ADSP-21XX (-lib)

Этот ключ должен использоваться, если компонуемая программа была сгенерирована С компилятором семейства ADSP-2100 и подключает некоторые функции С библиотеки рабочих программ ADSP-21XX. Эта библиотека является набором подпрограмм ANSI стандарта для цифровой обработки сигнала.

Ключ `-lib` принуждает редактор связей искать С библиотеку и компоновать любые вызванные функции. Ключ `-lib` используется только вместе с ключом `-c`.

4.2.2.6. Копирование библиотечных подпрограмм в загрузочные страницы (-p)

Ключ `-p` указывает редактору связей, что необходимо разместить копию библиотечной подпрограммы в каждой загрузочной странице, откуда она вызывается. Этот ключ должен быть применен даже в том случае, когда подпрограмма (или подпрограммы) вызывается на одной странице.

Обычно этот ключ применяется для указания часто используемых подпрограмм в файлах текущего каталога совместно с ключом `-dir`:

```
LD21  файл1 файл2  -p  -dir .
```

Теперь редактор связей найдет все вызовы этих подпрограмм в ваших загрузочных страницах и присвоит им соответствующий код. (**Примечание**: редактор связей не будет автоматически искать файлы в текущей директории. Если вы сохранили там ваши библиотечные файлы, необходимо использовать ключ `-dir` или переменную окружения `ADIL`, чтобы указать это). Ключ `-p` может применяться только для систем с памятью начальной загрузки (все процессоры ADSP-21XX, за исключением ADSP-2100).

4.2.2.7. С стек исполняемой программы в памяти программ (-pmstack)

Ключ `-c` указывает редактору связей реализовать стек исполняемой программы в памяти данных для программ сгенерированных С компилятором ADSP-21XX. Ключ `-pmstack` передвигает стек в память программ.

Если программа была откомпилирована с ключом `-pmstack`, необходимо компоновать ее также с ключом `-pmstack`. Ключ используется только вместе с ключом `-c`.

4.2.2.8. ПЗУ версия С библиотеки рабочих программ ADSP-21XX (-rom)

Функции библиотеки рабочих программ ADSP-21XX, могут быть размещены редактором связей в ПЗУ или ОЗУ (по умолчанию). Если программа была откомпилирована с ключом `-crom`, необходимо компоновать ее с ключом `-rom`. Ключ используется только вместе с ключом `-c`.

4.2.2.9. Создание динамической памяти С (-s)

Ключ -s указывает редактору связей на необходимость резервирования области памяти, выделяемой программе для динамически размещаемых структур данных, «кучи», для программ, сгенерированных С компилятором ADSP-21XX. Ключ `-s` может применяться только вместе с ключом `-c`, который вызывает создание стека исполняемой программы.

```
LD21  файл1 файл2  -c  -s размер_кучи
```

Аргумент *размер_кучи* должен быть целым числом, преобразуемым редактором связей в единицы слов памяти. Необходимо указывать ключ `-s`, для использования таких С функций, как `malloc`, `calloc` или `free`.

Обычно вершина стека размещена по самому старшему доступному адресу памяти данных (или памяти программ, если задан ключ `-pmstack`) и перемещается в направлении нижних адресов. Когда задан ключ `-s`, «куча» размещается в самых старших доступных адресах памяти. Стек сдвигается вниз, и начинается сразу после окончания блока «кучи».

Если используется ключа `-s`, редактор связей создает метку:

```
____top_of_stack      (метка с четырьмя символами подчеркивания)
```

адрес которой равен

```
____top_of_stack = ____top_of_ram - heap_size
```

4.2.2.10. Поиск библиотечного файла быстрого доступа (-user)

Ключ `-user` указывает библиотечный файл, который получен с помощью утилиты построения библиотек LIB21. При использовании этого ключа редактор связей будет искать только обозначенный файл для компоновки библиотечных подпрограмм. Смотрите раздел «Построение единой библиотеки для быстрого доступа».

4.3. Как работает редактор связей

Данные раздел дает представление о том, как работает редактор связей, когда он обрабатывает программные модули. При лучшем понимании принципов работы редактора связей, вы сможете структурировать свои программы для более эффективного использования памяти.

Работа редактора связей состоит в комбинировании ассемблированных модулей и инициализации данных в исполняемой программе, называемой файлом отображения памяти (`.EXE`). Две основные задачи: размещение памяти и разрешение символов.

4.3.1. Распределение памяти

Редактор связей читает каждый модуль кода и объявления переменных данных/буферов для определения типа памяти, в которой они должны быть расположены – ОЗУ или ПЗУ, память программ или память данных, имя сегмента и т.д.

Редактор связей читает также содержимое файла описания архитектуры `.ACH`, чтобы определить какие пространства памяти доступны и какие у них характеристики.

На основе этой информации происходит размещение каждого модуля, буфера и переменной в соответствующем типе памяти. Если объект имеет абсолютный адрес, указанный параметром `ABS`, его называют неперемещаемым. Если абсолютный адрес не присвоен, объект считается перемещаемым. Редактор связей присваивает адрес каждому перемещаемому объекту.

Если определитель сегмента (SEG) объединяется с параметром ABS, объявленный объект также считается неперемещаемым. Если определитель SEG используется без абсолютной адресации, тогда объявленный объект считается перемещаемым в пределах названного сегмента.

Редактор связей размещает объекты в памяти в следующей последовательности:

1. **неперемещаемые объекты** – буферы данных и модули с параметром ABS.

2. **перемещаемые внутри сегмента циклические буферы** – буферы данных с параметрами CIRC и SEG.

3. **перемещаемые внутри сегмента нециклические буферы и модули** – модули и буферы данных с параметрами SEG

4. **перемещаемые циклические буферы** – буферы данных с параметрами CIRC.

5. **перемещаемые модули и нециклические буферы** – все оставшиеся модули и нециклические буферы.

Для процессоров ADSP-21XX с памятью начальной загрузки на кристалле, редактор связей разместит как можно больше загрузочных кода и данных во внутренней памяти, перед тем, как использовать внешнюю.

Циклические буферы могут быть размещены в некоторых пределах памяти, принимая во внимание характеристики аппаратной адресации циклических буферов процессоров ADSP-21XX. Обычно циклический буфер должен начинаться с базового адреса, который кратен 2^n, где n - число разрядов, требуемых для представления длины буфера в двоичном выражении.

Редактор связей размещает циклические буферы в памяти, в соответствии с этим ограничением. Если длина циклического буфера равна, например, 13 словам, редактор связей размещает его с базового адреса, который кратен 16.

4.3.1.1. Размещение памяти начальной загрузки

Система может иметь до 8 страниц памяти начальной загрузки. Одна страница может хранить до 2048, 24 разрядных слов памяти программ для ADSP-2101, ADSP-2111 и ADSP-21msp50. Страницы начальной загрузки ADSP-2105 и ADSP-2115 размещают до 1024 слов.

Следует помнить важную вещь, касающуюся разницы между тем, что происходит при компоновке и тем, что происходит во время выполнения программы. Любой модуль, объявленный с параметром BOOT, редактор связей размещает в пространстве памяти начальной загрузки. Это размещение, тем не менее, касается только хранения программ перед исполнением (когда страница загружается и выполняется).

Редактор связей должен также разместить адресное пространство для данных и программ загружаемого модуля в памяти программ или данных, где они располагаются во время выполнения. Таким образом, редактор связей назначает области в памяти начальной загрузки и памяти программа/данных для всех загружаемых модулей, обеспечивая логический интерфейс между загрузочной памятью и памятью во время выполнения.

Если вы захотите, чтобы в памяти процессора (внешней или внутренней) во время выполнения отдельной загрузочной страницы существовала незагружаемая подпрограмма или буфер данных, вы должны использовать параметр BOOT, чтобы связать его с этой страницей. При совместном использовании с параметром STATIC, редактор связей зарезервирует пространство для объекта во время выполнения страницы.

Выходной файл карты распределения памяти программы (.MAP) показывает как расположение вашей программы в памяти начальной загрузки, так и соответствующее отображение кода в памяти во время выполнения (после выполнения начальной загрузки).

Этот файл помогает понять перенос из памяти начальной загрузки в рабочую память. Вы не можете указать область, где модуль будет размещен в памяти начальной загрузки - редактор связей самостоятельно реализует эту функцию, составляя эффективно упакованные загрузочные страницы.

4.3.2. Разрешение символов

Чтобы разрешить программные символы, редактор связей должен поставить в соответствие каждому символу определенный адрес в пространстве памяти. Имена программных меток, переменных/буферов являются символами, которые определены в исходном коде. Ассемблер просто пропускает их редактору связей, задача которого определить адрес каждого из символов, после размещения в памяти всех модулей.

Обращение к символу может происходить только внутри модуля, где он определен, пока он не определен директивами ENTRY или GLOBAL. Эти директивы ассемблера расширяют диапазон обращения к символу. Другие модули должны объявить этот символ директивой EXTERNAL перед обращением к нему.

Для каждой ссылки EXTERNAL, редактор связей ищет в других модулях определения ENTRY или GLOBAL. При нахождении нескольких совпадений выводится сообщение об ошибке. Если поиск не выявил определений символа, редактор связей включает поиск библиотечного файла в соответствии с последовательностью, описанной в разделе «Последовательность поиска библиотеки». Этот поиск включает просмотр переменной среды окружения ADIL и аргументов ключей -user, -dir.

Если подключаемый символ не найден с помощью поиска в библиотеке, редактор связей выводит сообщение об ошибке.

После того как распределение памяти завершено, и все внешние ссылки разрешены, редактор связей присваивает каждому символу значение адреса.

Редактор связей вырабатывает файл таблицы символов .SYM (если задан ключ -g), который содержит список всех встреченных программных символов и их адресов. Этот файл показывает какие символы могут быть доступны каждому модулю.

4.4. Использование библиотечных файлов ваших подпрограмм

Редактор связей ADSP-21XX позволяет вам использовать библиотечные файлы ваших часто используемых программ и подпрограмм. Библиотечные подпрограммы делаются доступными для любых компонуемых программ. Простой ссылкой на функцию вы заставляете редактор связей искать и компоновать соответствующий библиотечный файл.

Чтобы приготовить свою библиотеку, вы должны сделать следующее:

Что делаете вы:

1. Пишете библиотечные подпрограммы, располагая в начале каждой из них метку. Объявляете эти метки как глобальные через директиву `ENTRY`.

2. Объявляете начальную метку библиотечной подпрограммы как внешнюю в начале каждого модуля, использующего эту функцию с помощью директивы ассемблера `EXTERNAL`.

3. Ассемблируйте библиотечные подпрограммы в одном или более модулях (один модуль на файл).

4. Указываете редактору связей, где можно найти ваши библиотечные файлы (задавая путь у каталогу в переменной среду окружения `ADIL` или ключом `-dir`).

Теперь, когда вы компонуете программу, которая использует ваши библиотечные подпрограммы, редактор связей делает следующее:

Что делает редактор связей:

1. Располагает программу в памяти, собирает все символы в программе и пытается определить адрес обращения к каждой метке (известное как «разрешение символов», описанное ранее).

2. Поскольку метки библиотечных подпрограмм являются неразрешенными (ненайденными) символами, редактор связей автоматически открывает и ищет их в каждом объектном файле (`.OBJ`), находящимся в каталогах, обозначенных в переменной `ADIL` или ключе `-dir`. Проверяются все метки, определенные через директиву `ENTRY`.

3. Любой файл, содержащий метку, на которую ссылается программа, включается процесс компоновки.

Для систем ADSP-21XX с несколькими страницами начальной загрузки, если применяется какая-либо библиотечная подпрограмма, должен быть указан ключ `-p`. Этот ключ вынуждает редактор связей разместить копии подпрограмм на всех страницах начальной загрузки, где это требуется.

©АВТЭКС Санкт-Петербург. http://www.autex.spb.ru, E-mail: info@autex.spb.ru

4.4.1. Построение единой библиотеки для быстрого доступа

Программное обеспечение разработчика ADSP-21XX включает построитель библиотеки – утилиту **LIB21**, которая позволяет вам записать несколько библиотечных подпрограмм и упаковать их в один файл для быстрого доступа. После создания библиотечного файла необходимо вызвать редактор связей с ключом `-user` *имя_библиотеки*, который позволяет находить, выделять и компоновать необходимые подпрограммы из файла *имя_библиотеки*.

Использование утилиты LIB21 и ключа `-user` подразумевает то же самое, что и применение ключа `-dir` или переменной среды окружения `ADIL`.

Преимущество утилиты LIB21 состоит в том, что редактор связей работает быстрее. Утилита LIB21 вызывается одним из двух способов:

```
LIB21  имя_библиотеки  файл1 [файл2...]  [-V версия]
```

или

```
LIB21  имя_библиотеки  -i списочный_файл  [ -V версия]
```

Выходной файл *имя_библиотеки.A* объединяет в себе отдельно ассемблированные модули, перечисленные в командной строке (*файл1*, *файл2* и т. д.). Эти модули должны быть указаны без расширений. Ключ `-i` имеет такое же действие, как и при вызове редактора связей. Файл *списочный_файл* содержит список с одним именем файла на строке. Ключ `-V` позволяет вставить номер версии для подпрограмм в библиотечном модуле; аргумент *версия* является символьной строкой. Строка версии не влияет на исполнение программы.

Ниже приведен пример создания библиотечного файла быстрого доступа:

```
LIB21  filler  taps  coeffs  start_input  -v V1.0
```

Утилита LIB21 создаст файл `FILTER.A`, объединяющий три входных модуля. Символьная строка версии «`V1.0`» вставляется в файл. Редактор связей можно вызывать строкой:

```
LD21  main  sum  graph  -user filter
```

Что вызовет компоновку модулей `main`, `sum` и `graph`. Полагается, что из файла `FILTER.A` будет использована одна или более библиотечных подпрограмм, и редактор связей выделит требуемые функции и включит их в процесс компоновки.

4.4.2. Последовательность поиска библиотеки

Существует несколько путей использования библиотечных подпрограмм в сочетании с редактором связей: ключ `-dir`, переменная среды окружения `ADIL`, утилита LIB21 и ключ `-user`, ключ `-lib`. При использовании различных комбинации, редактор связей разбирает их в следующей последовательности:

1. Если задан ключ `–user`, ищется и открывается файл библиотеки быстрого доступа. Последовательность поиска каталога с файлом:

    a) текущая директория, затем
    b) директории, перечисленные в переменной `ADIL` (если необходимо)

2. Если задан ключ `-lib`, ищется С библиотека рабочих программ ADSP-21XX. Просматриваются каталоги, перечисленные в переменной `ADIL`.

3. Для неразрешенных ссылок просматриваются директории, перечисленные в переменной `ADIL`.

4. Если задан ключ `-dir`, просматриваются директории, перечисленные в аргументах.

4.5. Системы с несколькими страницами начальной загрузки

Реализация систем с несколькими страницами начальной загрузки может быть простой или сложной, зависящей от того, сколько программ и данных требуется разместить. Если каждая страница полностью независима и содержит всю необходимую информацию, распределение памяти редактором связей проходит относительно просто.

Однако множество систем должны распределять программы или структуры данных между загрузочными страницами. Стандартная операция компоновки не позволяет сделать это, поскольку редактор связей стирает ранее существующую карту разделения памяти для каждой страницы. Для предотвращения этого, вы должны использовать один из следующих технических приемов:

- повторение параметра `BOOT` в модуле (содержащем программы и/или структуры данных), с указанием каждой страницы, которую он требует.

- задание ключа `-p` редактора связей, чтобы копировать библиотеку рабочих подпрограмм на каждую страницу начальной загрузки.

- использование определителя `STATIC`, чтобы оставить модуль или буфер/переменную в памяти, во время загрузки новых страниц.

Для достижения различных результатов эти технические приемы могут быть использованы отдельно или в некоторой комбинации. Первые два приема имеют одинаковый конечный результат, в то время как определитель `STATIC` обладает другим механизмом действия. Применение этих определителей предотвращает перезапись модуля (или буфера) при загрузке страницы, как страницы 0 при перезапуске (если MMAP=0), так и страниц 1-7 под управлением программного обеспечения. Это относится к модулям/буферам как для внутренней, так и для внешней памяти процессора.

Когда редактор связей определяет память для размещения программы, он рассматривает девять независимых частей памяти: незагружаемые память программ и данных, и загружаемые страницы 0-7. Незагружаемую память определяют как начальное состояние памяти программ и данных, перед загрузкой любой страницы и выполнением кода.

До применения определителя `STATIC` эти девять частей рассматривают совершенно независимо друг от друга. В отсутствии любых объявлений `STATIC`, редактор связей полагает, что каждая из девяти частей начинается с чистого листа, и каждая загружаемая страница имеет собственное распределение памяти, доступное при загрузке.

©АВТЭКС Санкт-Петербург. http://www.autex.spb.ru, E-mail: info@autex.spb.ru

4.5.1. Перезагрузка под управлением программы

Загрузка новой страницы во время выполнения программы описана в главе «Интерфейс памяти» руководства пользователя семейства ADSP-2100. Выполнение перехода от одной страницы к другой происходит перезагрузкой под управлением программного обеспечения. Перезагрузка управляется регистром System Control Register, отображенным во внутренней памяти данных по адресу 0×3FFF. Этот регистр содержит разряд BFORCE (управляющий загрузкой) и поле BPAGE (выбора загружаемой страницы).

Программы, реализованные в нескольких страницах начальной загрузки, выполняются следующим образом:

1. Страница 0 загружается и выполняется после перезапуска системы.

2. Когда должна быть загружена новая страница, устанавливают разряды поля BPAGE.

3. Для перехода на новую страницу устанавливают бит BFORCE. Внутренняя память программ загружается из новой страницы; внутренняя память данных не меняется.

4.5.2. Пример совместного использования статического буфера

Этот раздел приводит пример совместного использования буфера данных на нескольких загрузочных страницах. Этот же метод может быть использован для разделения подпрограмм, разрешая вызывать их кодом, расположенным на разных страницах. Этот пример использует страницы начальной загрузки размером 2K, которые могут быть реализованы на процессорах ADSP-2101, ADSP-2111 и ADSP-21msp50. В этом примере для защиты от перезаписи буфера используется определитель STATIC. Страницы 0, 1 и 2 совместно используют этот буфер; код загруженный из этих страниц получает доступ к данным. Буфер объявлен в исходном коде на загрузочной странице 0:

```
.VAR/PM/RAM/STATIC  dat1[64];
.GLOBAL  dat1;
.INIT  dat1: <frames.dat>;
```

Буфер с именем dat1 и длиной в 64 слова определен как глобальный, чтобы сделать его видимым снаружи модуля. Другие модули должны определить dat1, как внешний, чтобы иметь к нему доступ.

Редактор связей считывает значения 64 слов данных из файла frames.dat для инициализации ими буфера dat1. Данные включены в порции загружаемой памяти EXE файла. Этот файл переводится разделителем программ для записи в ППЗУ в файл формата программатора ППЗУ.

Рис.4.2. показывает память нулевой, первой и второй страницы начальной загрузки. Заметьте, что редактор связей размещает статический объект dat1 на вершине страницы 0 и резервирует 64 верхних адреса на страницах 1 и 2. Если на этих страницах необходимо будет расположить больше кода, и адресное пространство для буфера dat1 будет перекрыто, редактор связей выведет сообщение об ошибке.

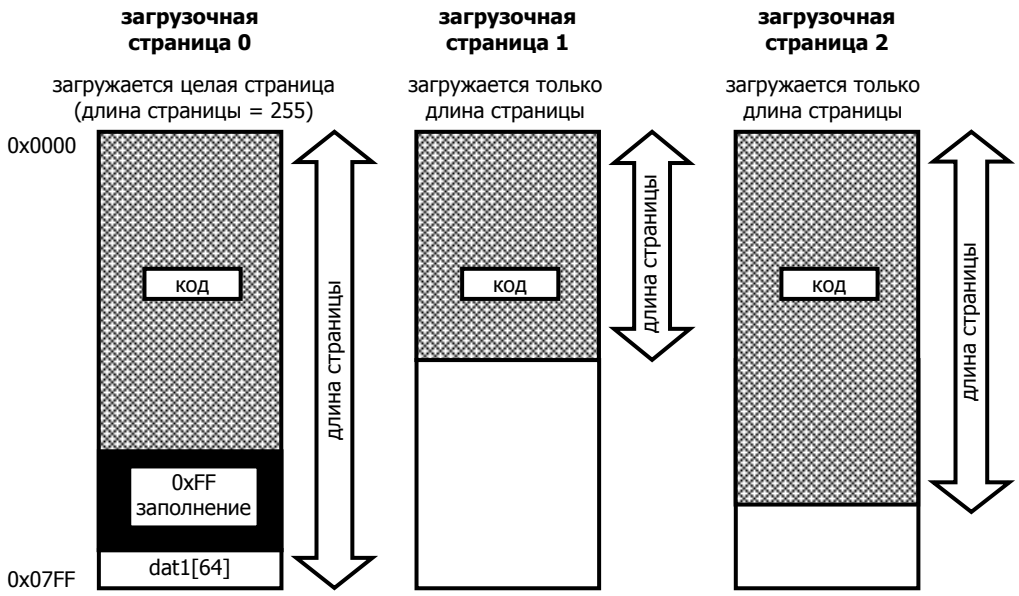

Рис.4.2. Статический буфер на страницах начальной загрузки.

На рисунке видно, что размещение защищает буфер от перезаписи, поскольку страницы 1 и 2 загружаются только длиной страниц. Схема загрузки адресов процессоров ADSP-21XX загружает внутреннюю память программ, начиная с самого старшего используемого адреса (как определено длиной страницы). Поскольку код этих страниц содержит меньше, чем (2048-64) слов, при загрузке он не будет переписывать `dat1`. Начальные данные для `dat1` содержатся в копии страницы 0, поэтому длина загружаемой страницы 255 слов (страницы загружена полностью). Редактор связей размещает заполняющие байты со значением 0×FF в области нулевой страницы между окончанием кода и буфером `dat1`.

(**Примечание**: длина страницы определяется как:
pagelength = (число 24-разрядных PM слов / 8) – 1

Еще одно применение определителя `STATIC`, это создание в памяти «разделенной» (совместно используемой) подпрограммы или буфера с абсолютным адресом. Адрес должен быть выбран таким, чтобы он был больше, чем длина самой большой загружаемой страницы. В этом случае вы делаете работу за редактора связей - размещаете объект в памяти, гарантируя, что он не перекроется в течение загрузки новых страниц. Однако бывает легче, при разработке сложных систем, позволить редактору связей сделать эту работу самостоятельно.

4.5.3. Пример использования статических и динамических сегментов

Этот раздел описывает пример системы с многостраничной организацией памяти начальной загрузки. Будет использоваться процессор ADSP-2101, для которого создается файл описания архитектуры, определяющий статические и динамические сегменты памяти. Статический сегмент будет содержать код, используемый всеми загружаемыми страницами, в то время как динамический сегмент будет содержать код, определенный для каждой страницы.

Система будет использовать страницы 0, 1 и 2. Страница 0 загружается при перезапуске системы. При последующей загрузке страниц 1 и 2, каждая из них будет частично перезаписывать внутреннюю память программ. Перезаписываемая часть будет «динамическим» сегментом, а защищенная часть будет «статическим» сегментом. Ниже приведен файл описания архитектуры для системного конфигуратора:

```
.SYSTEM overlay_test;
.ADSP2101;
.MMAPO;

.SEG/PM/RAM/CODE/ABS=0          dynamic[0×400];{нижние 1K внутренней PM}
.SEG/PM/RAM/CODE/DATA/ABS=0×400 fixed[0×400];  {верхние 1K внутренней PM}
.SEG/DM/RAM/DATA/ABS=0×3800     int_dm[0×400]; {внутренняя DM}

.SEG/BOOT=0/ROM  boot 0[0×800];  {загрузочная страница 0}
.SEG/BOOT=1/ROM  boot 1[0×800];  {загрузочная страница 1}
.SEG/BOOT=2/ROM  boot 2[0×800];  {загрузочная страница 2}

.ENDSYS;
```

Сегменты `dynamic` и `fixed` расположены во внутренней памяти программ. Общий код для всех загружаемых страниц помещен в сегмент `fixed`; все коды, не являющиеся общими размещены в сегменте `dynamic`. Размер сегмента выбран здесь для примера – для реального приложения размер сегмента должен определяться размером кода. Следующие операторы объявляют три постранично определенных модуля:

```
.MODULE/RAM/SEG=dynamic/BOOT=0  mod01;
.MODULE/RAM/SEG=dynamic/BOOT=0  mod02;
.MODULE/RAM/SEG=dynamic/BOOT=0  mod03;

.MODULE/RAM/SEG=dynamic/BOOT=1  mod11;
.MODULE/RAM/SEG=dynamic/BOOT=1  mod12;
.MODULE/RAM/SEG=dynamic/BOOT=1  mod13;

.MODULE/RAM/SEG=dynamic/BOOT=2  mod21;
.MODULE/RAM/SEG=dynamic/BOOT=2  mod22;
.MODULE/RAM/SEG=dynamic/BOOT=2  mod23;
```

Следующий оператор объявляет единственный модуль, который будет использоваться на всех трех загружаемых страницах:

```
.MODULE/RAM/STATIC/SEG=fixed/BOOT=0  common_mod;
```

Наконец, предположим, что следующее объявление буфера содержится в некотором другом модуле, загружаемом из страницы 0:

```
.VAR/DM/RAM/STATIC/SEG=int_dm  common_buf[100];
```

Буфер данных хранит 100 слов, которые должны быть доступны из всех трех загружаемых страниц. Следовательно, необходимо использовать объявление `STATIC`. Модуль, который объявляет и инициализирует `common_buf`, должен включать программу копирования буфера после загрузки из внутренней памяти программ во внутреннюю память данных, так как редактор связей только резервирует место, но не обеспечивает механизм записи данных. Рис.4.3. показывает копии загрузочных страниц 0, 1 и 2 после компоновки программы.

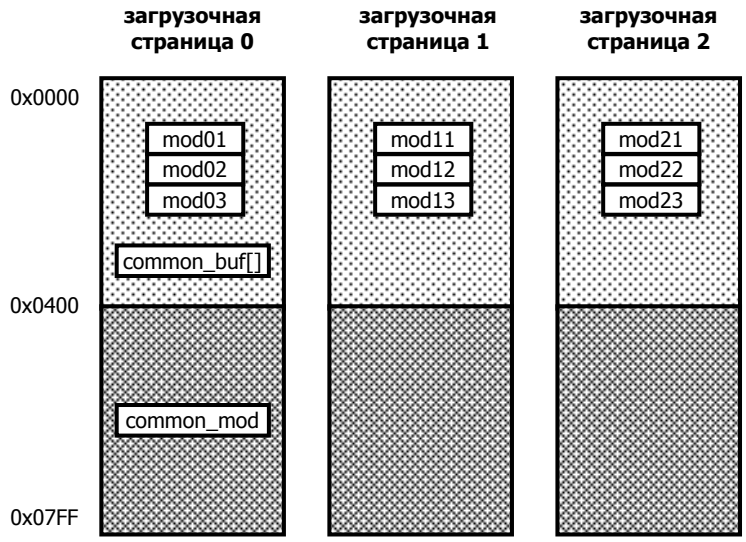

Рис.4.3. Статические и динамические объекты в памяти начальной загрузки.

Рис.4.4 показывает состояние внутренней памяти ADSP-2101 после загрузки страницы 0 и копирования буфера `common_buf` в память данных. Цикл начальной загрузки процессоров ADSP-21xx загружает внутреннюю память программ, начиная самого старшего используемого адреса, декрементируя его до нуля. Младшие адреса загружаются с каждой перезагрузкой. Поскольку страницы 1 и 2 только загружают код в сегмент `dynamic`, сегмент `fixed` не перезаписывается. Это позволяет модулю `common_mod` постоянно оставаться в памяти.

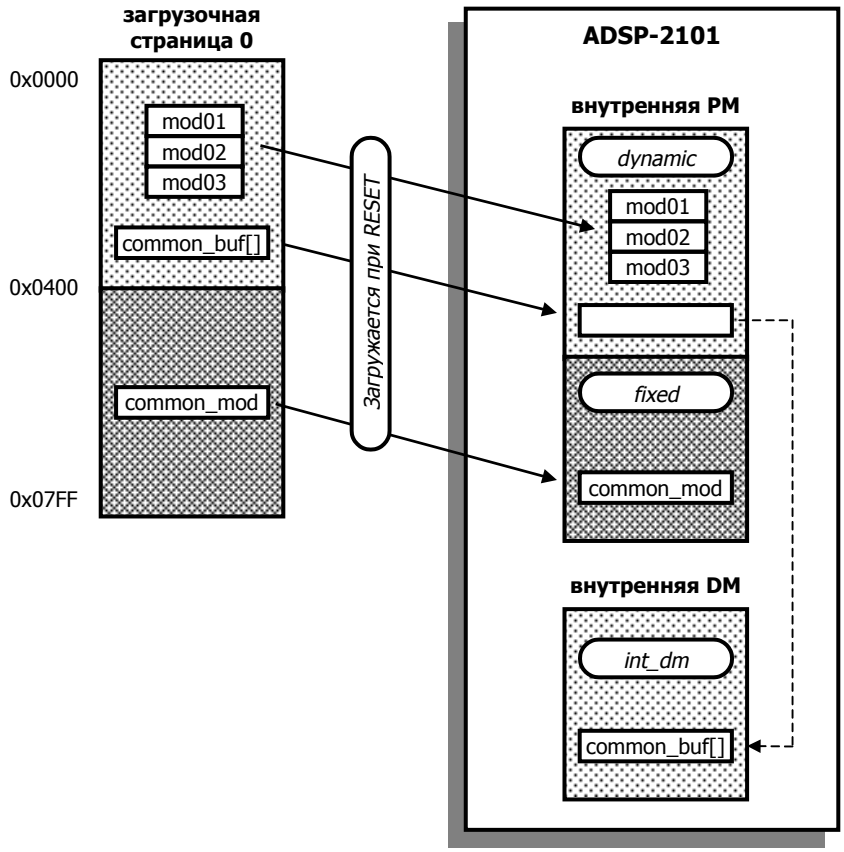

Рис.4.4. Состояние памяти страницы 0 во время выполнения программы.

4.6. Структура файла карты распределения памяти

Файл карты распределения памяти (.MAP) помогает вам интерпретировать результаты компоновки. Редактор связей генерирует этот файл, если был указан ключа -x. Данный файл обеспечивает следующую информацию:

- Символы

Список перекрестных ссылок всех программных символов, отсортированный по модулям. Для каждого символа предоставляется следующая информация:

Тип символа	название модуля, буфера/переменной или программной метки
Адрес	базовый адрес для модулей и буферов
Длина	для модулей и буферов
Область памяти	PM, DM или BM

- Сегменты памяти

Структура сегментов памяти, объявленных системным конфигуратором и описанная в файле архитектуры .ACH. Для каждого сегмента показаны базовый адрес, длина, атрибуты памяти.

- Память начальной загрузки память программ во время выполнения

Карта адресов модулей и структур данных на каждой странице начальной загрузки и соответствующая карта кода, загруженного во внутреннюю память программ. Информация о загрузочных байтовых адресах ППЗУ и требуемом размере ППЗУ.

- Статическая и динамическая память

Структура статической памяти программ, динамической памяти данных и статической памяти данных. Включена информация об адресе, длине и атрибутах памяти.

- Сообщения об ошибках

- Библиотеки

Список найденных библиотечных файлов, включенных в процесс компоновки.

Пример структуры файла карты распределения памяти показан на Рис.4.5.

```
ADSP-21XX Linker Version 3.0              Analog Devices, Inc.
final (final.exe) mapped according to FIR_SYSTEM (sysb2101.ach)

xref for module: MAIN_ROUTINE       boot memory page(s) 0,
   MAIN_ROUTINE               pm 0:0000 [003B]        module(global)
   DATA_BUFFER                dm 0:3800 [000F]        variable(global)
   COEFFICIENT                pm 0:0040 [000F]        variable(global)
   RESTARTER                  pm 0:001C               label
   CLEAR_BUFFER               pm 0:0024               label
   WAIT                       pm 0:0039               label
   FIR_START                     0:004F [0000]        extern(FIR_ROUTINE)

xref for module: FIR_ROUTINE        boot memory page(s) 0,
   FIR_ROUTINE                pm 0:004F [000A]        module(global)
   FIR_START                  pm 0:004F               label
   CONVOLUTION                pm 0:0054               label
   COEFFICIENT                   0:0040 [000F]        extern(MAIN_ROUTINE)
   DATA_BUFFER                   0:3800 [000F]        extern(MAIN_ROUTINE)

21xx memory per FIR_SYSTEM (sysb2101.ach):
   internal 2101 pm ram mapped to 0000 - 0800 (auto booted at reset)
   internal 2101 dm ram mapped to 3800 - 3BFF
   0000 - 07FF [  2048.] external bm rom code BOOT_MEM
   0000 - 07FF [  2048.] internal pm ram data/code INT_PM
   0800 - 3FFF [ 14336.] external pm ram data/code EXT_PM
   3800 - 3BFF [  1024.] internal dm ram data INT_DM
   0000 - 37FF [ 14336.] external dm ram data EXT_DM

boot memory and bootable runtime program memory map:
   boot page 0 (auto boot)

   bm:0000-003A (x8rom:0000-00EB) pm:0000-003A [59.]
   ram module           MAIN_ROUTINE of MAIN_ROUTINE

   bm:0040-004E (x8rom:0100-013B) pm:0040-004E [15.]
   ram circ variable    COEFFICIENT of MAIN_ROUTINE

   bm:004F-0058 (x8rom:013C-0163) pm:004F-0058 [10.]
   ram module           FIR_ROUTINE of FIR_ROUTINE

   8k of boot memory rom space required for this bootable runtime map.
   Most convenient boot memory rom size is 8k bytes (64k bits).

fixed program memory map:
   fixed program memory rom: 0.
   fixed program memory ram: 0.

dynamic data memory map:
   boot page 0
   3800 - 380E [15.]    ram circ variable DATA_BUFFER of MAIN_ROUTINE

fixed data memory map:
   fixed data memory rom: 0.
   fixed data memory ram: 0.

21xxlnk: final, 21xx memory use:
   program memory rom: 0.; program memory ram: 0.;
   data memory rom: 0.; data memory ram: 0.
```

Рис.4.5. Структура файла карты распределения памяти.