Введение & Установка

1

1.1. Введение

Средства разработки для семейства процессоров ADSP-2100 включают в себя комплект программных инструментов проектирования. Программное обеспечение состоит из инструментов программирования на языках ассемблер и С, а также программ моделирования работы процессоров, позволяющие разрабатывать и отлаживать системы DSP. Программное обеспечение разработчика может быть запущено на IBM (или IBM- совместимых) PC и рабочих станциях SUN4.

Средства разработки состоят из нескольких программ: системного конфигуратора (*system builder*), ассемблера (*assembler*), редактора связей или компоновщика (*linker*), программы разделителя для программатора ППЗУ (*PROM splitter*), программы моделирования или симулятора (*simulator*) и С компилятора (*compiler*). Это руководство описывает первые пять программ.

Для получения информации С компиляторе для семейства процессоров ADSP-2100, обратитесь к *ADSP-2100 Family C Tools Manual & ADSP-2100 Family C Runtime Library Manual,* соответственно.

Для получения информации по архитектуре и системному интерфейсу каждого процессора, обратитесь к *ADSP-2100 Family User's Manual.*

Семейство ADSP-2100 включает в себя следующие процессоры:

Процессор	*Описание*
ADSP-2100/ADSP-2100A	микропроцессор DSP
ADSP-2101	микрокомпьютер DSP
ADSP-2105	микрокомпьютер DSP
ADSP-2115	микрокомпьютер DSP
ADSP-2111	микрокомпьютер DSP с Хост портом
ADSP-21msp50/55/56	микрокомпьютер DSP смешанного сигнала
ADSP-2171	микрокомпьютер DSP с Хост портом

Это руководство содержит полную информацию по разработке программ для всех перечисленных процессоров.

Версии процессоров с ПЗУ, программируемым фотошаблонами, такие как ADSP-2102 и ADSP-2106, не перечислены в списке, однако программы для этих изделий создаются тем же способом, как и для стандартных компонентов.

Другие процессоры, которые пополнят семейство ADSP-2100 в будущем, будут полностью программно-совместимыми и позволят использовать средства разработки ADSP-21xx и данное руководство.

В этом руководстве, термин «ADSP-21xx» используют в основном для ссылки на один или все процессоры семейства ADSP-2100. Термин «ADSP-2100» использован для обозначения ADSP-2100 и ADSP-2100A.

Заметьте, что любые возможности программного обеспечения или текстовые ссылки относящиеся к памяти начальной загрузки, внутренней памяти или памяти на кристалле, могут быть отнесены ко всем процессорам ADSP-21xx, исключая ADSP-2100. Этот процессор не включает память на кристалле и не использует память начальной загрузки.

Каждая версия программного обеспечения поставляется с сопровождающими замечаниями. Эти замечания описывают текущую версию и дают информацию по любой замене программного обеспечения. ***Пожалуйста, позаботьтесь вернуть регистрационную карту с вашими координатами!*** *Это позволит нам поддерживать вас информацией о последующих версиях программного обеспечения.*

1.2. Содержание

Руководство содержит следующие главы, описывающие программное обеспечение:

- Глава 2: Системный конфигуратор

Системный конфигуратор это программный инструмент для описания особенностей оборудования. Вы создаете исходный файл состава системы, который содержит соотношение памяти ОЗУ и ПЗУ, размещение памяти программ и данных, размещение портов ввода/вывода целевого оборудования. Чтобы облегчить эту задачу используют высокоуровневые конструкции.

- Глава 3: Ассемблер

Ассемблер переводит ваши программы на языке ассемблер. Он поддерживает синтаксис инструкций семейства ADSP-2100 и поддерживает возможность гибких макропроцессов. Препроцессор поддерживает директивы С препроцессора в исходном коде. Исходный код разделяют в определенные установками модули (файлы). Прилагается полный набор диагностики.

- Глава 4: Редактор связей

Редактор связей осуществляет процесс редактирования отдельно -ассемблированных модулей и генерирует исполняемый файл. Он может выполнить связывание модулей в библиотеку исполняемых подпрограмм. Он размещает скомпонованную программу и данные в аппарате целевой системы, как указано в выходном файле построителя системы и может создать несколько загружаемых страниц исполняемых файлов для процессоров с загружаемой памятью.

- Глава 5: Разделитель программ для записи в ППЗУ

Разделитель программ для записи в ППЗУ использует выходные данные редактора связей и генерирует файл для записи в ППЗУ в форматах различных промышленных стандартов. Форматы этих файлов записаны в Приложении B.

Следующая информация не включена в перевод, и может быть найдена в оригинале руководства *ADSP-2100 Family Assembler Tools & Simulator Manual*.

- Главы 6 – 13: Программа моделирования (симулятор)
- Приложение A, Коды инструкций.
- Приложение B, Форматы файлов
- Приложение C, Сообщения об ошибках
- Приложение D, Таблицы векторов прерываний
- Приложение E, Сообщения об ошибках симулятора.
- Приложение F, Форматы файлов данных симулятора.

1.3. Процесс разработки системы

Рис.1.1 показывает этапы процесса разработки системы ADSP-21xx. Процесс разработки начинается с задачи определения аппаратного обеспечения системы. Определить аппаратное окружение можно с использованием программного инструмента –системного конфигуратора. Необходимо написать файл описания системы, который будет содержать входные данные для системного конфигуратора; этот файл специфицирует конкретную систему. Системный конфигуратор считывает данные и генерирует файл описания архитектуры системы, который предоставляет информацию о системе редактору связей, программе моделирования и эмулятору (в случае его использования).

Вы начинаете написание программы с создания исходных модулей на языке ассемблер. Ассемблерный программный модуль это термин языка ассемблер включающий главную программу, подпрограмму или объявление переменных. Каждый программный модуль обрабатывается ассемблером отдельно.

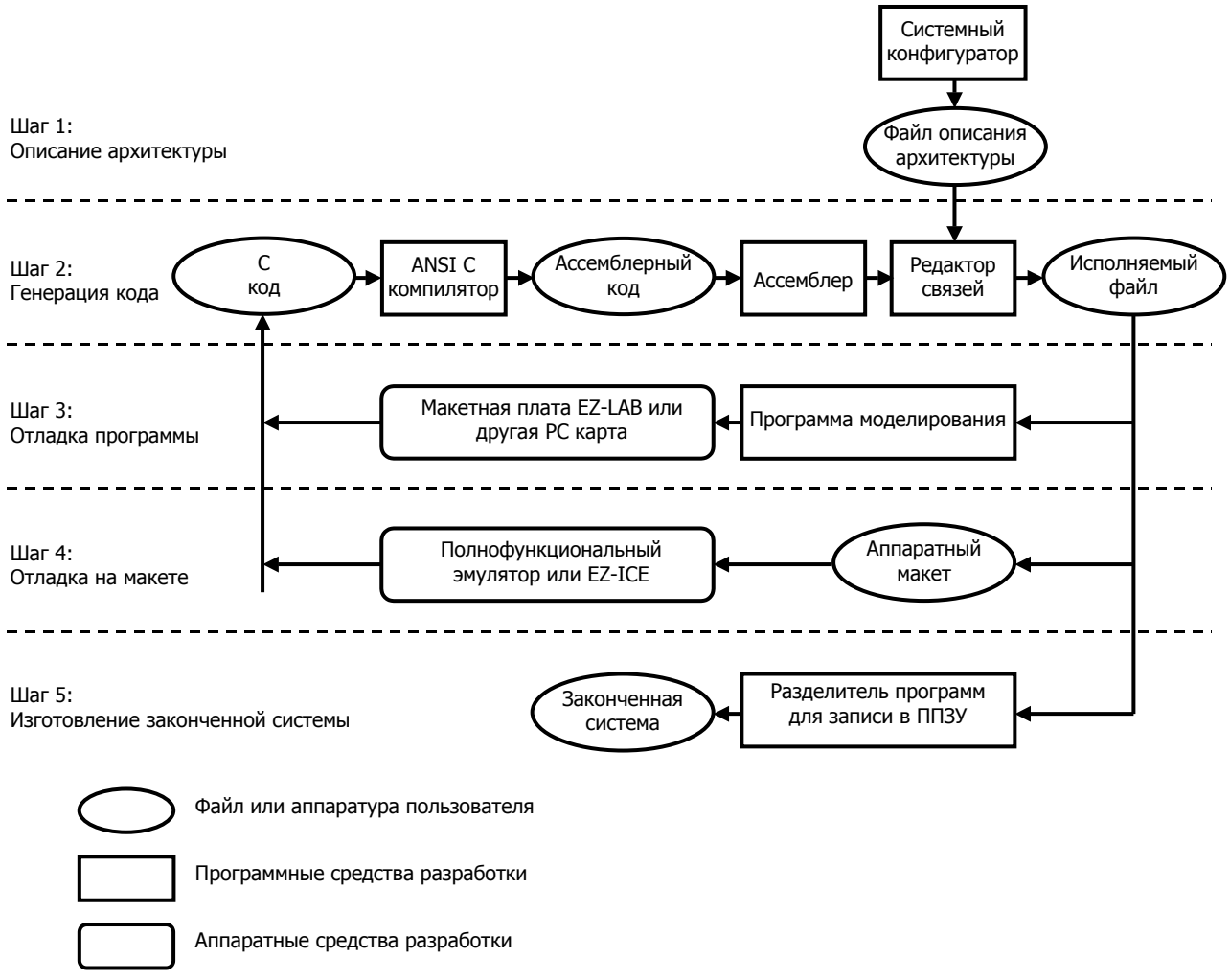

Рис.1.1. Процесс разработки системы ADSP-21xx.

Редактор связей читает информацию о конкретной системе из файла описания архитектуры, чтобы определить размещение фрагментов программы и данных. В ассемблерных модулях вы можете определять каждый фрагмент программы/данных, как полностью перемещаемый, перемещаемый в пределах определенного сегмента памяти, или неперемещаемый (размещаемый по абсолютному адресу). Неперемещаемые модули программ или данных размещают по указанным адресам памяти. Перемещаемые объекты размещаются в памяти редактором связей.

Пользуясь файлом описания архитектуры и ассемблированными программными модулями, редактор связей определяет размещение перемещаемых модулей программ и данных и присваивает всем модулям в памяти адреса с правильными атрибутами (CODE или DATA, RAM или ROM). Редактор связей генерирует исполняемый файл, содержащий копию содержимого памяти, который может быть использован для проверки программой моделирования или эмулятором.

Симулятор предусматривает окна, которые показывают различные части аппаратных средств. Для отображения аппаратных средств, симулятор конфигурирует собственную память в соответствии с файлом описания архитектуры и моделирует порты ввода/вывода отображенные в памяти. Моделирование позволяет провести отладку системы и проанализировать исполнение программы перед загрузкой в аппаратный макет.

После полного моделирования вашей системы и проверки программного обеспечения, к аппаратному макету подключают эмулятор, чтобы проверить схему, синхронизацию и выполнение программы в реальном режиме времени. Эмулятор имеет оверлейную память, которая может быть использована вместо компонент памяти конкретной системы.

Разделитель программ для записи в ППЗУ - это программный инструмент, переводящий выходные программы редактора связей (копии содержимого памяти) в формат файла промышленного стандарта для программатора ППЗУ. После того, как вы запрограммировали микросхему ППЗУ и подключили ее к процессору ADSP-21xx, размещенному на целевой плате, ваша система готова к запуску.

1.4. Константы

Константы включают в себя числовые константы и символы определенные как константы. Символические константы могут использоваться вместо числовых. Символ должен быть объявлен как константа для системного конфигуратора или ассемблера с использованием директивы `.CONST`. Объявление константы для системного конфигуратора не переносится в ассемблер, таким образом вы должны написать новые объявления для констант в вашей исходной программе на языке ассемблер.

1.5. Основания счисления

Основания счисления, которые могут быть использованы в исходной программе включают шестнадцатеричную, восьмеричную, двоичную и десятичная. Их обозначают следующим образом.

Для шестнадцатеричных чисел, префикс `0x` (ноль и x) или `H#`:

```
0x24FF      H#CF8A
```

Для восьмеричных , префикс `0` (ноль):

```
0777
```

Двоичные числа обозначаются с префиксом `B#`:

```
B#01110100
```

Десятичные числа используются по умолчанию (префикс отсутствует):

```
1024        +1024       -55
```

1.6. Набор символов

Программное обеспечение средств разработки ADSP-21xx распознает следующие символы:

- Буквы верхнего регистра от «A» до «Z»
- Буквы нижнего регистра от «a» до «z»
- Цифры от «0» до «9»
- ASCII графические символы - т.е. печатные символы отличные от букв и цифр (пунктуации и т.д.)
- ASCII не графические: пробел, табуляция, возврат каретки, перевод строки, перевод страницы (символ или символы «новая строка» интерпретируются в соответствии с установками, при которых символ был получен.)

1.7. Символы

Символ это строка из знаков, используемая одним из двух способов. Символы, которые вы определяете во входном файле системного конфигуратора или программе на языке ассемблер используются для представления, например, сегмента памяти, адреса или значения данных. Другие символы резервируют ключевые слова распознаваемые системным конфигуратором или ассемблером. Зарезервированные символы приведены в главах 2 и 3.

Символ, определенный на языке ассемблер может быть именем модуля, переменной, буфером данных, программной меткой, портом ввода/вывода, макросом или константой.

Символы начинаются со знака из следующих наборов:

- прописные буквы от «A» до «Z»
- строчные буквы от «a» до «z»
- знак подчеркивания «_»

с последующими знаками из наборов:

- прописные буквы от «A» до «Z»
- строчные буквы от «a» до «z»
- знак подчеркивания «_»
- цифры от «0» до «9»

Другими словами, ваши символы не должны начинаться с цифры. Символ может быть длиной до 32 знаков. Здесь приведено несколько примеров типичных символов и что они могут обозначать:

`main_prog`	программный модуль на языке ассемблер
`xoperand`	переменная данных
`input_array`	буфер данных
`subroutine1`	программная метка
`AtoD_INPUT`	порт, отображенный в памяти

1.7.1. Чувствительность к регистру

Программное обеспечение средств разработки на языке ассемблер ADSP-21xx может быть установлено как чувствительным к регистру, с различием между буквами верхнего и нижнего регистров, так и нечувствительным к регистру, принимая буквы верхнего и нижнего регистров за одни и те же знаки.

По умолчанию чувствительность к регистру отключена, и вы можете вводить текст в любых комбинациях верхнего и нижнего регистров. Язык C, тем не менее различает регистр символов, и если вы используете компилятор C семейства ADSP-2100, то системный конфигуратор и ассемблер должны быть также установлены чувствительными к регистру. Это достигается включением в командную строку ключа (`-c`). Для уточнения деталей смотрите главы 2 и 3.

1.8. Выражения ассемблера

Ассемблер семейства ADSP-2100 может преобразовывать отдельные выражения в исходном коде. Выражение может быть использовано там, где вычисляется числовая константа .

Разрешены два вида выражений:

- арифметические или логические операции с двумя или более целыми константами

  примеры: `29+129` `(128-48)*3` `0x55&0x0F`

- символ + или - целая константа

  примеры: `data–8` `data_buffer+15` `startup+2`

Символы могут быть переменными, буферами данных или программными метками. Все эти символы представляют собой значение адреса, которое определяется редактором связей. Сложение или вычитание константы обозначает смещение от адреса.

Простые арифметические или логические выражения могут быть использованы для объявления символических констант с помощью директивы `.CONST`. Эти выражения могут использовать следующие наборы операторов, распознаваемых языком программирования C (записаны в порядке предпочтения):

`( )`	левая, правая скобки
`~ -`	поразрядное дополнение, одиночный минус
`* / %`	умножение, деление, модуль
`+ -`	сложение, вычитание
`<< >>`	битовые сдвиги
`&`	битовое “AND”
`|`	битовое “OR”
`^`	битовое “XOR”

Выражения также могут быть использованы при вводе команды в одной из программ моделирования семейства ADSP-2100. Симуляторы распознают в дополнительный набор выражений и операторов, которые описаны в разделе «Выражения» главы 6.

Самое важное различие между ассемблерными выражениями и выражениями симулятора состоит в том, что содержимое памяти (например, переменные) и содержимое регистров процессора могут быть использованы как операнды *только в симуляторе.* Ассемблер не может преобразовывать значения в памяти и значения регистров во время ассемблирования, однако, симулятор имеет доступ к мгновенным значениям в памяти и регистров во время моделирования.

1.9. Принятые соглашения

Этот раздел приводит список принятых в этом руководстве соглашений.

- Ключевые слова (символы зарезервированные системой) представлены в тексте знаками верхнего регистра, хотя реально они могут быть введены знаками любого регистра. Обе формы ключевых слов зарезервированы.

- Символы в нижнем регистре выделенные курсивом, например, *`jumplabel`*, обычно представляют символы определяемые пользователем. такие как программная метка, переменная или имя файла.

- Квадратные скобки, [ ] , заключают необязательные величины, размер сегмента памяти (в директиве `.SEG` системного конфигуратора) или длину буфера данных (в директиве ассемблера `.VAR`)

- Многоточие, . . . , показывает , что предшествующие элементы могут быть повторены.

- Термин **ADSP-21xx** используется для ссылки на один или все процессоры семейства ADSP-2100.

- Термин **ADSP-2100** используется для обозначения как процессора ADSP-2100, так и процессора ADSP-2100A.

- Сокращения **DM, PM** и **BM** используются вместо памяти данных, памяти программ и памяти начальной загрузки, соответственно.

- С того момента, когда директива ассемблера `.VAR` была использована для объявления и однословной переменной данных, и многословных буферов данных, термин **буфер данных (*data buffer*)** обозначает как переменную, так и буфер.

1.10. Установка и замечания по версии

Детали процесса установки программного обеспечения могут отличаться от версии к версии; замечания по версии сопровождают каждый новый релиз должны и содержать эти детали. Вы должны всегда проверять выполнение указаний для используемой версии перед продолжением установки.

Инструкции по установке для различных базисных платформ приведены отдельно для каждой версии.

1.10.1. Файлы и переменные окружения

Программа установки создает на вашем жестком диске различные поддиректории и файлы. Файлы могут размещаться в каталоге, принятом по умолчанию или в другом каталоге, заданном пользователем. Вы должны найти установленными следующие исполняемые программные файлы:

Имя файла	*Описание*
BLD21.EXE	системный конфигуратор
ASM21.EXE	ассемблер С препроцессор
ASMPP.EXE	ассемблер препроцессор
ASM2.EXE	ассемблер
LD21.EXE	редактор связей (компоновщик)
LIB21.EXE	библиотекарь (программа для работы с библиотеками программ)
SPL21.EXE	разделитель программ для программатора ППЗУ
HSPL21.EXE	разделитель программ для загрузки через порт HIP (для использования с ADSP-2111 и ADSP-21msp50)

Новые версии программного обеспечения могут включать другие файлы. Смотрите информацию по изменениям в замечаниях.

Создаются следующие переменные окружения, которым во время установки присваиваются значения по умолчанию:

Переменные окружения	*Описание*
ADI_DSP	путь к директории содержащей инсталлированные файлы
ADII	путь(и) к директориям INCLUDE, используемых ассемблером

Это полный набор переменных окружения для программного обеспечения средств разработки семейства ADSP-21xx, включая программу моделирования (симулятор) и С компилятор.

Переменные окружения создаются когда вы устанавливаете любую часть программного обеспечения. Переменные окружения требуются для правильной работы программного обеспечения.

Когда программное обеспечение успешно установлено, вы подготовлены к тому, чтобы писать программу и использовать программные инструменты.

1.10.2. Пример архитектуры и исходные программные файлы

Несколько примеров программирования систем включено в программное обеспечение. Эти файлы расположены в директории названной \EXAMPLES, которая создается в корне каталога, выбранного при установке. Примеры включены для того, чтобы помочь вам научиться как писать программы для ADSP-21xx и использовать программное обеспечение.

Файлы названы в соответствии с номером примера. Каждый пример включает файл системной архитектуры .SYS, один или более файлов ассемблерных модулей .DSP, и командный файл .BAT. Запуск командного файла будет вызывать системный конфигуратор, ассемблер и редактор связей в порядке создания исполняемого .EXE файла, который может быть загружен и запущен на симуляторе ADSP-21xx.