ПРОГРАММИРОВАНИЕ 12

12.1 ОБЗОР

С точки зрения программирования, процессоры семейства ADSP-2100 состоят из трех вычислительных устройств, двух генераторов адреса данных, программного автомата. Кроме того, они могут иметь дополнительные устройства и внутреннюю память на кристалле, размеры и вид которой зависят от модификации процессора. Почти все операции, в которых участвуют указанные элементы архитектуры процессора, выполняются с использованием одного или нескольких регистров, необходимых, например, для хранения данных, нахождения адресов значений (указателей) или для определения операционных режимов.

Внутренние регистры используются для хранения данных, адресов, информации, относящейся к управлению или состояниям процессора. Например, в регистре AX0 содержится операнд (данные) АЛУ, в I4 - указатель Генератора адреса данных 2 (адрес), в ASTAT - флаги состояний арифметических операций, а группа разрядов DWAIT задает число состояний ожидания для различных зон памяти данных.

Обращение к регистрам может осуществляться двумя способами. Содержимое назначенных регистров, таких как MX0 и IMASK, может считываться и записываться при помощи непосредственных явных команд языка ассемблер. Например:

```
M X 0 = 1 2 3 4 ;
I M A S K = 0 x F ;
```

Содержимое регистров, отображенных в карте памяти, - регистра управления системой, регистра управления состояниями ожидания, регистров таймера, последовательных портов и др., - становится доступным при считывании и записи соответствующих ячеек в памяти данных. Например, для очистки содержимого регистра управления состояниями ожидания, отображенного в карте памяти данных по адресу 0x3FFE, используется следующий код команды:

```
A X 0 = 0 ;
D M ( 0 x 3 F F E ) = A X 0 ;
```

В данном случае AX0 используется для хранения константы 0, так как нет команды, с помощью которой можно было бы непосредственно записать значение данных в память используя при этом непосредственный адрес.

Все регистры, встречающиеся в процессорах семейства ADSP-2100, приводятся на рис. 12.1. В каждом процессоре семейства не обязательно имеются все из нижеуказанных регистров. Регистры сгруппированы функционально: регистры генераторов адреса данных, регистры программного автомата, вычислительных устройств (АЛУ, умножителя-накопителя, устройства сдвига), обмена данными между шинами, интерфейса памяти, таймера, последовательных портов, порта интерфейса хост-машины и интерфейса прямого доступа к памяти.

12 ПРОГРАММИРОВАНИЕ

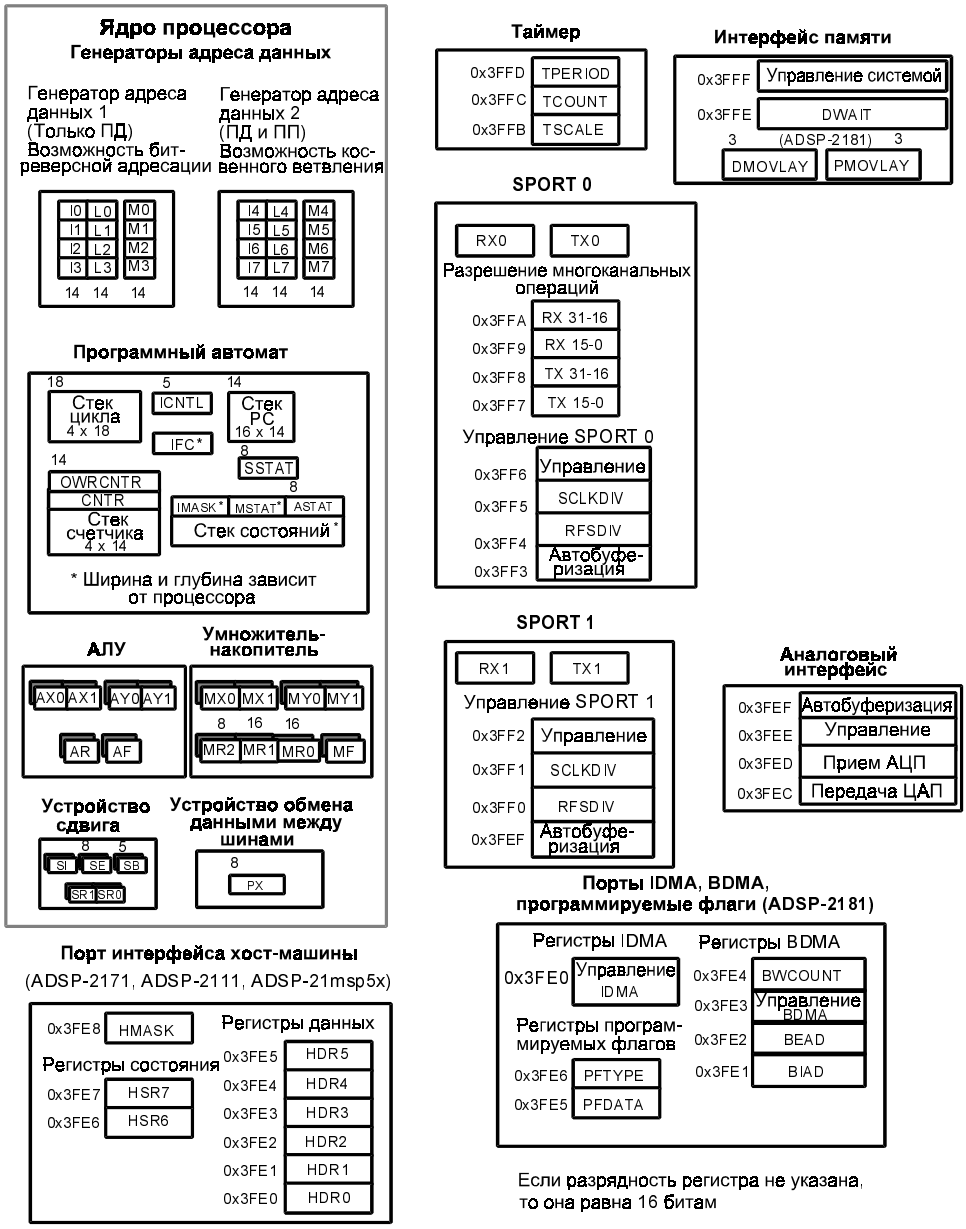

Рис. 12.1 Регистры процессоров семейства ADSP-2100

12.1.1 Регистры генераторов адреса данных

В каждом генераторе адреса данных имеется двенадцать регистров разрядностью 14 бит: четыре индексных (I) регистра, использующихся для хранения указателей адреса, четыре регистра модификации (M) для обновления значений этих указателей и четыре регистра длины (L), использующихся для организации циклических буферов. Генератор адреса данных 1 генерирует только адреса памяти данных и может осуществлять битреверсную адресацию. Генератор адреса данных 2 генерирует адреса как памяти данных, так и памяти программы, а также адреса, которые используются в командах условного ветвления (переходы и вызовы) и для доступа к данным. Например:

```
A X 0 = D M ( I 0 , M 0 ) ;
```

представляет собой команду косвенного считывания из ячейки в памяти данных, адрес которой указан в индексном регистре I0. По завершению считывания содержимое I0 обновляется с использованием для этого модифицирующего значения, содержащегося в M0. Команда:

```
P M ( I 4 , M 5 ) = M R 1 ;
```

представляет собой команду косвенной записи в память программы по адресу, указанному в регистре I4 с последующей модификацией содержимого этого регистра значением из регистра M5. Команда:

```
J U M P  ( I 4 ) ;
```

может служить иллюстрацией команды косвенного перехода.

12.1.1.1 Всегда инициализируйте регистры длины L

В процессорах семейства ADSP-2100 предусмотрено два режима адресации для доступа к памяти данных: косвенная и прямая. Косвенная адресация выполняется за счет загрузки значения адреса в индексный регистр I и одновременного выбора одного из регистров модификации M.

Регистры длины L облегчают адресацию с циклическим возвратом по длине циклического буфера. Циклический буфер организуется только в том случае, когда в регистре L содержится любое значение, кроме нуля. *Для линейной (нециклической) косвенной адресации регистр длины L, соответствующий определенному индексному регистру I, должен содержать значение ноль.* Не стоит надеяться на то, что инициализация регистров L производится автоматически, или что ее можно проигнорировать; после перезапуска процессора в регистрах I, M, L содержатся случайные числа. Следует предусмотреть правильную инициализацию всех регистров L, соответствующих используемым регистрам I, в вашей программе.

12 ПРОГРАММИРОВАНИЕ

12.1.2 Регистры программного автомата

Регистры программного автомата управляют подпрограммами, циклами и прерываниями. Они также содержат указание на состояния и выбранные режимы работы.

12.1.2.1 Прерывания

Регистр ICNTL управляет вложенными прерываниями и срабатыванием внешних прерываний (по фронту или по уровню). Регистр IFC позволяет принудительно устанавливать и сбрасывать прерывания при помощи программных средств. Регистр IMASK маскирует (блокирует) отдельные прерывания. Разрядность регистров IFC и IMASK отличается в разных процессорах, так как различные процессоры семейства ADSP-2100 поддерживают различное количество прерываний.

Процессоры ADSP-2171, ADSP-2181 и ADSP-21msp58/59 поддерживают глобальные команды разрешения (ENA INTS) и запрещения (DIS ENA) прерываний.

После перезапуска процессора все прерывания по умолчанию разрешены. При выполнении команды запрещения прерывания все прерывания маскируются без изменения содержимого регистра IMASK. Блокирование прерываний не влияет на операцию автобуферизации последовательных портов, которые работают в нормальном режиме независимо от того, разрешены или запрещены прерывания. Команда запрещения прерываний маскирует все прерывания пользователя, включая прерывание перехода в режим пониженной мощности.

Команда разрешения прерываний позволяет снова обрабатывать все демаскированные прерывания.

12.1.2.2 Счетчики циклов

Значение счетчика для текущего цикла содержится в регистре CNTR. Использование стека счетчика позволяет вложить до четырех циклов счетчика. При попытке записи значения в регистр CNTR его текущее значение помещается в стек счетчика, а затем только в него записывается новое. Например, по команде:

```
C N T R = 1 0 ;
```

текущее содержимое CNTR помещается в стек счетчика, после чего в этот регистр загружается значение 10.

Особо следует отметить команду OWRCNTR, посредством которой можно перезаписать значение счетчика для текущего цикла без помещения содержимого CNTR в стек счетчика. OWRCNTR не может быть считана (т.е. использована в качестве исходного регистра) и не может записываться в последней команде цикла DO UNTIL.

12.1.2.3 Биты состояния и режима

Регистр состояния стеков (SSTAT) содержит флаги заполнения стеков. В регистре арифметических состояний (ASTAT) хранятся флаги состояний вычислительных устройств. Биты управления различными возможными режимами и опциями хранятся в регистре состояния режима (MSTAT). Регистр MSTAT содержит четыре бита, управляющих выбором теневых регистров вычислительных устройств, бит режима постановки бит в обратном порядке для Генератора адреса данных 1 и биты режимов фиксации переполнения и насыщения для АЛУ. MSTAT имеет еще три дополнительных бита, указывающих на размещение результата умножителя, активизацию таймера и включение режима Go.

Для активизации или блокирования режимов процессора удобно использовать команду управления режимом (ENA, DIS).

12.1.2.4 Стеки

В программном автомате имеется четыре стека, с помощью которых осуществляется вложение циклов, подпрограмм и прерываний.

Стек счетчика программы (PC) имеет разрядность 14 бит и 16 ячеек в глубину. В нем содержатся адреса, по которым осуществляется возврат из подпрограмм и программ обслуживания прерываний, а также верхние адреса циклов. При вызове подпрограмм и обработке прерываний обращение и запись в стек счетчика программ производится автоматически. Кроме того, можно непосредственно поместить или извлечь данные из стека счетчика программ с помощью команд управления счетчиком программ: TOPPCSTACK=reg, reg=TOPPCSTACK..

Стек циклов разрядностью 18 бит имеет 14 бит для адресов конца цикла и 4 бита для хранения условий прекращения цикла. Глубина этого стека - 4 ячейки. Стек циклов автоматически заполняется при выполнении команды DO UNTIL и опустошается при выходе из цикла в случае вложенного цикла. Команда POP LOOP позволяет непосредственно извлечь данные из стека циклов.

В стек состояний, который автоматически заполняется во время обслуживания прерывания процессором, помещается информация о состоянии регистра маскирования прерываний (IMASK), регистров состояния режима (MSTAT) и арифметических состояний (ASTAT). Глубина и разрядность этого стека различна в разных процессорах семейства из-

за того, что разные процессоры обслуживают разное число прерываний. При возвращении в главную программу после обслуживания прерывания (по команде RTI) содержимое стека состояний автоматически извлекается из него. Поместить и извлечь информацию из этого стека можно и непосредственно - при помощи команд PUSH STS и POP STS.

В стеке счетчика разрядностью 14 бит содержатся значения счетчика (CNTR) для вложенных циклов. При каждой записи в регистр CNTR в стек счетчика автоматически помещается текущее значение этого регистра. Команда POP CNTR позволяет непосредственно извлекать данные из стека счетчика.

12.1.3 Регистры вычислительных устройств

Регистры вычислительных устройств используются для хранения данных.

Для большинства операций АЛУ и умножителя-накопителя требуется двое входных данных. Входные значения X содержатся в регистрах AX0, AX1, MX0 и MX1, в то время как входные значения Y содержатся в регистрах AY0, AY1, MY0 и MY1.

Результаты АЛУ помещаются в регистры AR и AF; содержимое регистра AF может снова подаваться на вход Y АЛУ, а содержимое регистра AR может использоваться в качестве входного значения X во всех вычислительных устройствах. Аналогичным образом, результаты умножителя-накопителя помещаются в регистры MR0, MR1, MR2 и MF и могут далее подаваться обратно в качестве входных данных для других вычислений. Двух 16-разрядных регистров, MR0 и MR1, и одного 8-разрядного регистра, MR2, достаточно для хранения результата разрядностью 40 бит, полученного при выполнении операции умножения с накоплением.

Входные значения устройства сдвига могут браться из АЛУ или умножителя-накопителя, а также из его собственных регистров результата или из регистра (SI), предназначенного для хранения входных значений устройства сдвига. Результат устройства сдвига помещается в регистры SR0 и SR1. В регистре SB содержится значение блочного порядка, использующееся при операциях с блочной плавающей точкой. Величина сдвига для операций нормализации и денормализации содержится в регистре SE.

Регистры вычислительных устройств имеют теневые регистры, изображенные на рис. 12.1 позади основных. Теневые регистры полезны, когда требуется выполнить контекстное переключение в течение одного цикла. Выбор набора теневых или основных регистров осуществляется за счет соответствующей установки и сброса бита в регистре MSTAT (состояния режима) при помощи команд:

```
ENA SEC_REG;      {выбор теневых регистров}
DIS SEC_REG;      {выбор основных регистров}
```

12.1.4 Регистр обмена данными между шинами

Регистр PX является 8-разрядным регистром, при помощи которого осуществляется пересылки данных между 16-разрядной шиной ДПД и 24-разрядной шиной ДПП. При передаче данных из памяти программы в любой 16-разрядный регистр и наоборот, в регистр PX помещаются или из него берутся 8 младших бит.

12.1.5 Регистры таймера

Значения периода таймера, счетчика и масштабного коэффициента содержатся в регистрах TPERIOD, TCOUNT и TSCALE, отображенных в карте памяти по адресам 0x3FFD, 0x3FFC и 0x3FFB, соответственно.

12.1.6 Регистры последовательных портов

В обоих последовательных портах, SPORT0 и SPORT1, имеются регистры приема (RX), передачи (TX) и управления. Последние отображены в карте памяти данных по адресам 0x3FEF - 0x3FFA. В SPORT0 также имеются регистры управления многоканальными операциями. Регистр управления каждого последовательного порта содержит биты, управляющие кадровой синхронизацией, компандированием, длиной слова данных, а также, в SPORT0, приемом и передачей данных по нескольким каналам. Значение в регистре SCLKDIV определяет частоту внутренне генерируемых тактовых синхроимпульсов для каждого последовательного порта; значение в регистре RFSDIV определяет частоту внутренне генерируемого сигнала кадровой синхронизации приема. Управление автобуферизацией последовательных портов осуществляется регистрами автобуферизации.

Программирование последовательного порта состоит в записи его регистров управления и, в зависимости от выбранных режимов, регистров SCLKDIV и RFSDIV этого порта. Ниже приводится пример программы для SPORT0, в которой задается 8-битовое компандирование по закону с μ-характеристикой, нормальный режим кадровой синхронизации и работа с внутренними тактовыми синхроимпульсами. В регистре RFSDIV установлено значение 255, что дает 256

```
S I = 0 x B 2 7 ;
D M ( 0 x 3 F F 6 ) = S I ;        {регистр управления SPORT0}
S I = 2 ;
D M ( 0 x 3 F F 5 ) = S I ;        {SCLKDIV = 2}

S I = 2 5 5 ;
D M ( 0 x 3 F F 4 ) = S I ;        {RFSDIV = 255}
```

циклов тактового генератора (SCLK) между подтверждениями RFS. В регистре SCLKDIV содержится значение 2, при котором частота тактовых синхроимпульсов составляет 1/6 частоты SCLKOUT.

12.1.7 Интерфейс памяти и активизация последовательных портов

Биты активизации последовательных портов, а также биты, определяющие конфигурацию SPORT1, находятся в регистре управления системой, отображенном в карте памяти данных по адресу 0x3FFF. Во всех процессорах, кроме ADSP-2181, в нем имеются также группы разрядов, управляющих операцией начальной загрузки, а именно: определяющие выбор страницы, число состояний ожидания и управляющие принудительной начальной загрузкой при помощи программных средств. В этом регистре также имеется группа разрядов PWAIT, определяющая количество состояний ожидания при обращении к внешней памяти программы.

Регистр управления состояниями ожидания, отображенный в карте памяти данных по адресу 0x3FFE, содержит группу разрядов, которая определяет число состояний ожидания для каждого банка памяти данных. В процессоре ADSP-2181 эта группа разрядов также задает число состояний ожидания при обращениях к памяти ввода/вывода. В процессорах с дополнительным ПЗУ на кристалле этот регистр также содержит бит активизации ПЗУ.

В процессоре ADSP-2181 состояния ожидания используются при обращении к внешней оверлейной памяти. Другие отображенные в карте памяти регистры управления портом IDMA и портом прямого доступа к памяти с байтовой организацией используются при выполнении операции начальной загрузки (для выбора страницы памяти с байтовой организацией, задания числа состояний ожидания, принудительной начальной загрузки при помощи программных средств) и обращении к памяти с байтовой организацией по ходу выполнения программы.

12.1.8 Регистры интерфейса хост-машины

Порт интерфейса хост-машины имеется в процессорах ADSP-2171, ADSP-2111, ADSP-21msp58/59. В интерфейсе хост-машины имеется шесть регистров данных, два регистра состояния и регистр маскирования прерываний. Эти регистры отображены в карте памяти данных по адресам 0x3FE7 - 0x3FE0. Регистры состояния содержат флаги состояний для каждого регистра данных. Регистр HMASK позволяет отдельно разрешать или блокировать генерацию прерываний считывания или записи ХИП для каждого регистра данных ХИП. Регистр HMASK отображен в карте памяти данных по адресу 0x3FE8.

12.1.9 Регистры аналогового интерфейса

В аналоговом интерфейсе процессоров ADSP-21msp58/59 имеется четыре регистра, отображенных в карте памяти данных по адресам 0x3FEC - 0x3FEF. Регистр передачи посылает данные на ЦАП для их последующей передачи. В регистр приема поступают данные с АЦП. В регистре управления аналоговым интерфейсом имеются биты, при установке которых выбирается усилитель, коэффициент усиления, аналоговый вход и возможные фильтры.

12.2. ПРИМЕР ПРОГРАММЫ

Ниже приводится фрагмент программы КИХ-фильтра, написанной для процессора ADSP-2111. Каждая часть программы сопровождена комментарием. При условии незначительных модификаций, данная программа может быть выполнена на любом процессоре семейства ADSP-2100. Данная программа КИХ-фильтра в значительной степени демонстрирует концептуальные преимущества архитектуры и набора команд процессоров семейства ADSP-2100.

```
{Программа КИХ-фильтра для ADSP-2111
      - последовательный порт 0 использован для ввода/
вывода
      - тактовые синхроимпульсы генерируются внутренне
      - тактовая частота процессора 12,2888 МГц делится
на последовательные тактовые синхроимпульсы с
частотой 1,536 МГц
      - частота кадровой синхронизации дискретизации 8
кГц}

    .MODULE/RAM/ABS=0   main_routine;   {загрузка програм}
                                        {мы с ППЗУ,}
                                        {MMAP=0}
A   .INCLUDE                <const.h>;
    .VAR/DM/RAM/ABS=0x3800/CIRC   data_buffer[taps];  {внутренний буфер}
B                                                     {данных}
    .VAR/PM/RAM/CIRV          coefficient[taps];
    .GLOBAL                   data_buffer, coefficient;
    .EXTERNAL                 fir_start;
    .INIT                     coefficient:<coeff.dat>;
```

12 ПРОГРАММИРОВАНИЕ

```
{начало программы}
{загрузка адресов векторов прерывания}
 C  JUMP restarter; NOP; NOP; NOP;  {перезапуск преры-}
                                    {вания}
    RTI; NOP; NOP; NOP;             {прерывание IRQ2}
    RTI; NOP; NOP; NOP;             {прерывание записи}
                                    {ХИП}
    RTI; NOP; NOP; NOP;             {прерывание чтения}
                                    {ХИП}
    RTI; NOP;NOP; NOP;              {прерывание пере-}
                                    {дачи SPORT0}
    JUMP fir_start; NOP; NOP;       {прерывание приема}
                                    {SPORT0}
    RTI; NOP; NOP; NOP;             {прерывание записи}
                                    {SPORT1}
    RTI; NOP; NOP; NOP;             {прерывание приема}
                                    {SPORT1}
    RTI; NOP; NOP; NOP;             {прерывание таймера}
{инициализация}
 D  restarter  LO=%data_buffer;     {задание длины цик-}
                                    {лического буфера}
               L4=%coefficient;     {задание длины цик-}
                                    {лического буфера}
               M0=1;                {модифицирующее значение=1}
                                    {для продвижения}
               M4=1;                { по буферу}
               I0=^data_buffer;     {указывает на начало}
                                    {данных}
               i4=^coefficient;     {указывает на начало}
                                    {коэффициентов}
               CNTR=%data_buffer;
               DO clear UNTIL CE;   {очищает буфер данных}
    clear:     DM(I0,M0)=0;
{установка отображенных в карте памяти регистров}
 E             AX0=191;
               DM(0x3FF4)=AX0;     {установка значения делителя}
                                    {для RFS 8 кГц}
               AX0=3;
               DM(0x3FF5)=AX0;     {частота внутренних тактовых}
                                   {синхроимпульсов}
                                   {1,536 МГц}
               AX0=0x69B7;
```

```
DM(0x3FF6)=AX0;   {многоканальный режим}
                  {запрещен}
                  {внутренне генери-}
                  {руемые последо-}
                  {вательные тактовые}
                  {синхроимпульсы}
                  {требуется кадровая}
                  {синхронизация приема}
                  {ширина приема 0}
                  {требуется кадровая }
                  {синхронизация переда-}
                  {чи}
                  {ширина передачи 0}
                  {прерывание кадровой }
                  {синхронизации передачи}
                  {блокировано}
                  {прерывание кадровой }
                  {синхронизации приема}
                  {разрешено}
                  {компандирование по за-}
                  {кону с μ-характеристикой}
                  {слова по 8 бит}
AX0=0x7000;
DM(0x3FFE)=AX0;   {состояния ожидания па-}
                  {мяти данных: 0x3400 - }
                  {0x37FF - 7 состояний}
                  {ожидания, остальные 0}
AX0=0x1000;
DM(0x3FFF)=AX0;   {SPORT0 блокирован}
                  {загрузка со страницы 0}
                  {0 состояний ожидания}
                  {памяти программы}
                  {0 состояний ожидания}
                  {памяти начальной за-}
                  {грузки}
```

```
                ICNTL=0x00;
                 IMASK=0x0018;      {разрешены только пре-}
                                    {рывания SPORT0}
mainloop:       IDLE;              {ожидание прерываний}
.ENDMODE;  JUMP mainloop;
```

Содержимое файла const.h

```
.CONST      taps=15, taps_less_one=14;
```

12.2.1 Подпрограмма установки

Подпрограмма установки и главного цикла выполняет инициализацию регистров процессора, а затем процессор ожидает по команде IDLE в цикле прерываний приема от SPORT0. Таким образом, фильтр управляется прерываниями, по приходу которых управление действиями процессора передается подпрограмме обслуживания прерываний (см. ниже).

Строка ***A*** показывает, что все постоянные объявляются в отдельном файле.

В часть ***B*** рассматриваемой программы вошли директивы ассемблера, которые задают два циклических буфера во внутренней памяти процессора: один - в ОЗУ данных (используется для хранения линии задержки) и один - в ОЗУ программы (используется для хранения коэффициентов фильтра). Эти коэффициенты загружаются из внешнего файла редактором связей. Указанные значения могут быть изменены не прибегая к повторному ассемблированию; требуется лишь повторное установление связей и компоновка.

В части ***C*** показана установка прерываний. Поскольку этот модуль программы расположен по абсолютному адресу ноль (на что указывает описатель-классификатор ABS в директиве .MODULE), первая команда помещается по вектору перезапуска. Первая ячейка отдана под команду вектора перезапуска, по которой осуществляется переход к подпрограмме *restarter*. Неиспользуемые вектора прерываний заполнены командой RTI (возвращение из подпрограммы обслуживания прерывания), за которой следуют пустые циклы. (Так как будет разрешено только одно прерывание, это, скорее, дань условностям программирования, нежели необходимость). Вектор прерывания приема SPORT0 указывает на переход к подпрограмме обслуживания прерывания.

В части ***D*** *restarter*, задаются индексные (I) регистры, регистры модификации (M) и длины (L), используемые в ходе адресации двух циклических буферов. Отличное от нуля значение длины буфера активизирует логику адресации по модулю. Каждый раз, когда происходит прерывание, указатели в регистре I сдви-гаются на одну позицию по длине буфера. В цикле *clear* все значения в буфере памяти данных обнуляются.

В части ***E***, после *clear*, производится установка всех отображенных в карте памяти процессора регистров управления, используемых в данной системе. Информация об инициализации регистров управления дана в Приложении Е.

Последовательный порт SPORT0 установлен на внутреннюю генерацию последовательных тактовых синхроимпульсов с частотой 1,536 МГц, исходя из тактовой частоты работы процессора 12,288 МГц. Требуются оба сигнала кадровой синхронизации: приема (RFS) и передачи (TFS), - однако сигнал RFS генерируется внутренне с частотой 8 кГц, а сигнал TFS подается на процессор с внешнего подключенного к нему устройства.

В итоге, SPORT0 доступен, а его прерывания разрешены. Теперь по команде IDLE процессор входит в состояние ожидания прерываний. По возвращению из подпрограммы обслуживания прерывания, выполнение программы возобновляется с команды следующей за командой IDLE. После однократного выполнения всех команд установки процессор в дальнейшем выполняет команды, указанные в подпрограмме обслуживания прерываний, которая показана ниже.

```
.MODULE/ROM/ fir_routine;          {перемещаемый мо-}
                                   {дуль программы преры-}
                                   {ваний КИХ-фильтра}
.INCLUDE <const.h>;                {включить объявленные}
                                   {постоянные}
.ENTRY fir_start;                  {делает метку видимой}
                                   {за пределами модуля}
.EXTERNAL data_buffer, coefficient; {делает глобальные ди-}
                                   {рективы доступными в}
                                   {модуле}
{программа обслуживания прерывания}
FIR_START;  CNTR=taps_less_one;  {N-1 прохождений по}
                                  {циклу DO UNTIL}
            SI=RX0;              {чтение с SPORT0}
            DM(I0,M0)=SI;        {пересылка данных в }
                                 {буфер}
            MR=0, MY=PM(I4,M4), MX0=DM(I0,M0);
                                 {установка умножителя}
                                 {для работы в цикле}
            DO convolution UNTIL CE;    {CE= счетчик }
                                        {пуст}
convolution: MR=MR+MX0*MY0(SS), MY0=PM(I4,M4), MX0=DM(I0,M0);
                                 {умножение с накопле-}
                                 {нием, выборка следую-}
                                 {щих множителей}
            MR=MR+MX0*MY0(RND);  {N-ный проход с}
                                 {округлением}
```

```
        TX0=MR1;            {запись в последова-}
                            {тельный порт}
        RTI;                {возвращение в главную}
                            {программу}
.ENDMODE;
```

12.2.2 Подпрограмма обслуживания прерывания

Эта подпрограмма пересылает полученные данные в следующую ячейку циклического буфера (перезаписывая старое значение). Все выборки и коэффициенты затем перемножаются, а результаты накапливаются для получения следующего выходного значения. Подпрограмма проверяет, не произошло ли переполнение, и производит операцию насыщения результата до соответствующего полномасштабного значения, а затем записывает результат в регистр передачи последовательного порта SPORT0, после чего происходит возвращение в главную программу.

В первых четырех строках подпрограммы содержится объявление перемещаемого модуля программы (который не находится по абсолютному адресу), в программу включается тот же самый файл постоянных, а точка входа в подпрограмму делается видимой для главной программы при помощи директивы .ENTRY. Аналогичным образом, директива .EXTERNAL делает все метки главной программы видимыми для подпрограммы обслуживания прерывания.

Подпрограмма начинается с загрузки регистра счетчика (CNTR). Из регистра приема данных последовательного порта SPORT0 (RX0) считывается новая выборка данных и записывается в регистр SI; в данном случае выбор этого регистра не играет никакой существенной роли. Затем данные записываются в буфер данных. Благодаря автоматической адресации циклического буфера, новые данные перезаписывают старые, и в буфере всегда содержатся самые свежие N выборок.

Четвертая команда подпрограммы, MR0=0, MY0=PM(I4,M4), MX0= DM(I0,M0), обнуляет регистр результата умножителя (MR) и производит выборку двух первых операндов. Несмотря на то, что при этом осуществляется обращение как к памяти программы, так и памяти данных, указанная команда выполняется в течение всего одного цикла, благодаря особенностям архитектуры процессора.

Метка *convolution* указывает на сам цикл, состоящий из двух команд, одна из которых задает цикл (DO UNTIL), а вторая находится "внутри" него. Команда умножителя-накопителя перемножает и суммирует предыдущий набор операндов, одновременно производя выборку следующих операндов из памяти программы и памяти данных.

Полученное в результате значение передается обратно в регистр передачи TX0 последовательного порта SPORT0, откуда оно пересылается на подключенное к процессору внешнее устройство.