РЕАЛИЗАЦИЯ АЛГОРИТМОВ СБОРА И ОБРАБОТКИ ДАННЫХ В АВТОМАТИЗИРОВАННЫХ СИСТЕМАХ НА ОСНОВЕ МНОГОФУНКЦИОНАЛЬНОЙ ПЛАТЫ ВВОДА/ВЫВОДА L-783

Цель работы: изучение алгоритмов сбора и обработки информации на примере автоматизированной системы измерений координат смещений торцов лопаток компрессора на основе многофункциональной платы ввода/вывода L-783 с сигнальным процессором ADSP 2184/2186.

1 Реализация алгоритмов управления сбором информации в системе измерений координат смещений торцов лопаток компрессора

Система предназначена для измерения координат смещений торцов лопаток компрессора на нестационарных режимах работы газотурбинного двигателя [1-3]. В состав системы входят комплект датчиков, многофункциональная плата ввода/вывода L-783 (ЗАО «Л-КАРД») и персональный компьютер. Комплект датчиков включает в себя датчик частоты вращения ротора (ДЧВ) с формирователем, датчики температуры и кластер из трех одновитковых вихретоковых датчиков (ОВТД) с индивидуальными преобразователями сигналов. В состав

преобразователя сигналов ОВТД входит генератор тактовых импульсов (ГТИ), формирующий управляющие сигналы импульсов питания каждого датчика и импульсов внешней синхронизации АЦП модуля L-783. Выходные сигналы этих преобразователей подаются на аналоговые входы модуля L-783. На цифровой вход L-783 подаются сформированные импульсные сигналы ДЧВ, их период с помощью таймера преобразуется в код, характеризующий скорость вращения ротора.

На рисунке 1 показана временная диаграмма преобразований сигналов датчиков и регистрации кодов измерения в буферной памяти L-783, а схема алгоритма управления, реализующего эту временную диаграмму, представлена на рисунке 2.

Пусть число датчиков в кластере ОВТД равно $n_{чэ}$, преобразование периода вращения ротора τ_p в цифровой код C_τ производится на каждом обороте ротора, а преобразование сигналов ОВТД в код (C_1, C_2, …, $C_{n_{чэ}}$) выполняется с постоянной максимально возможной частотой f_0 (минимальным периодом τ_0)[1]. Цифровой код C_τ, полученный на первом обороте ротора, используется для вычисления периода вращения ротора τ_p и моментов времени опроса ОВТД $t_{ц1}$, $t_{ц2}$, …, $t_{цn_л}$, причем эти вычислительные операции осуществляются в течение

[1] Исходя из условия получения нескольких отсчетов на лопатку на скоростях вращения ротора до 12000 об/мин максимальное значение частоты опроса f_0=1 МГц.

следующего (второго) оборота ротора. В предположении, что скорость вращения ротора сохраняется неизменной на отрезке времени $\tau_{p1}+\tau_{p2}+\tau_{p3}$ в течение третьего оборота производится выбор кодов, соответствующих сигналам ОВТД по каждой лопатке в моменты $t_{ц1}$, $t_{ц2}$, …, $t_{цn_л}$ (от начала третьего периода) и только эти значения кодов измерительных каналов ОВТД фиксируются в буферной памяти.

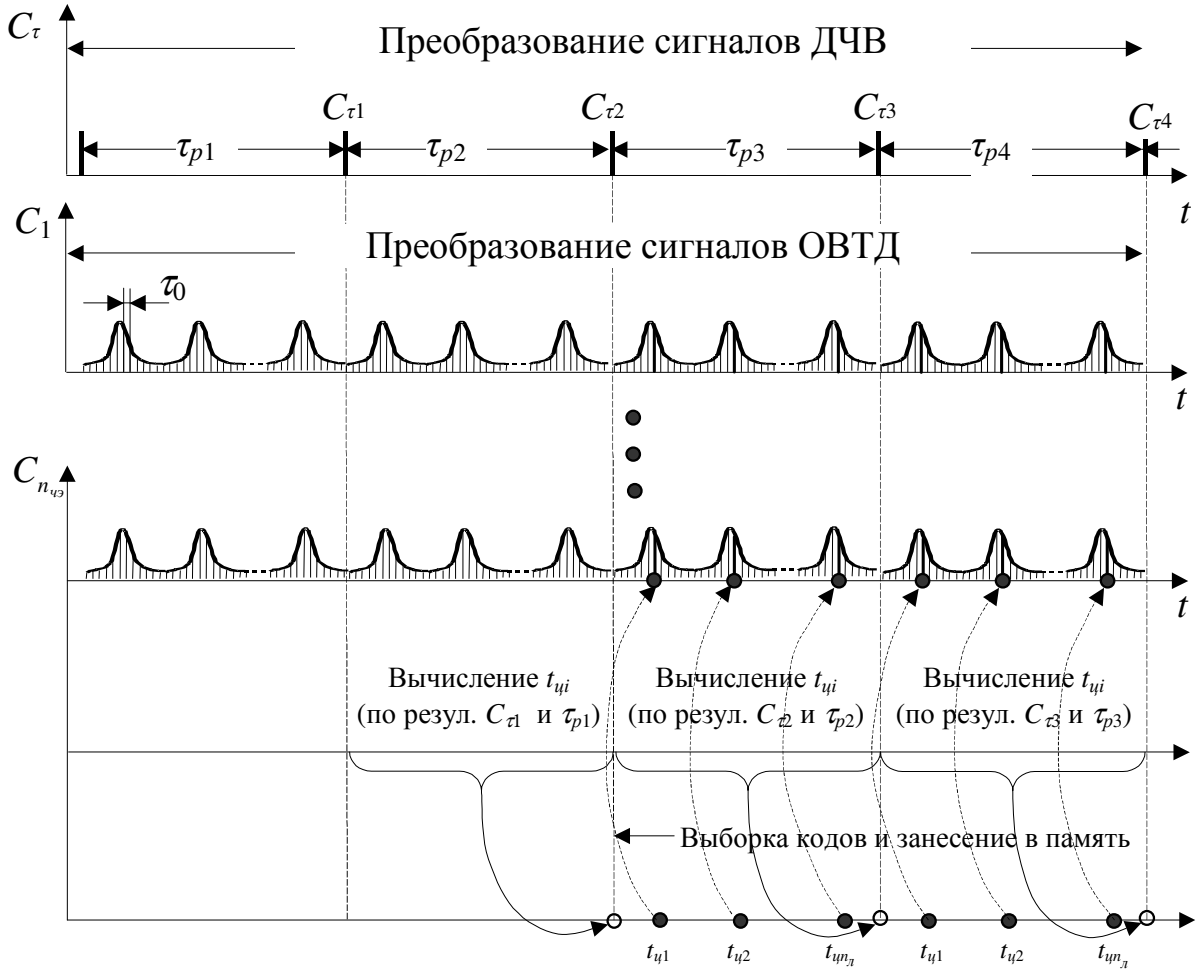

Рисунок 1 – Временная диаграмма преобразований сигналов и выборки кодов

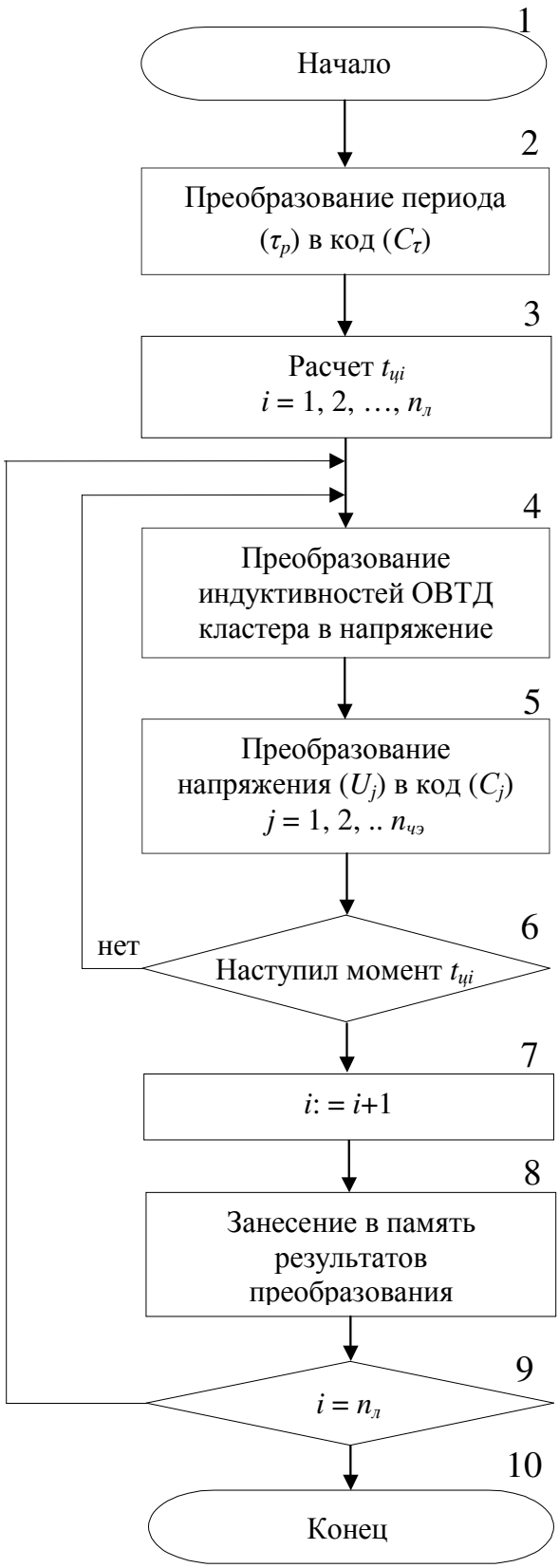

Рисунок 2 –Схема алгоритма управления преобразованиями сигналов и выборки кодов

Каждый последующий период данные, занесенные в буферную память, обновляются. При этом задержка между преобразованием скорости вращения в код и выборкой кодов ОВТД составляет один оборот ротора. Вместе с тем, представляется очевидным тот факт, что вычисление τ_p и $t_{ц1}, t_{ц2}, \ldots, t_{цn_л}$ не требует времени, равного целому периоду (если вычисления заканчиваются до момента $t_{ц1}$, то выборку кодов ОВТД можно произвести уже на следующем периоде, т.е. практически без задержки).

Схема алгоритма управления (рисунок 2) отражает только последовательность операций, которые происходят на протяжении 1-го, 2-го и 3-го периодов вращения ротора. Вначале осуществляется преобразование периода в код и расчет моментов опроса $t_{ц1}, t_{ц2}, \ldots, t_{цn_л}$ (блоки 2, 3). Преобразование индуктивностей ОВТД кластера и преобразование напряжений в код осуществляются также каждый период τ_0 (блоки 4-5), а занесение в память результатов преобразования только в момент $t_{цi}$ (блоки 6-9). Указанные операции повторяются для всех $n_л$ лопаток.

2 Структура многофункциональной платы ввода/вывода информации L-783

Для создания систем измерения на базе ПЭВМ различного спектра технических приложений широко используются модули

ЗАО «Л-КАРД» для ввода/вывода аналоговых и цифровых сигналов (www.lcard.ru). В данной работе используется многофункциональная плата ввода/вывода L-783 [4]. Эта плата содержит цифровой сигнальный процессор (DSP) с возможностью его программирования под прикладные задачи, что позволяет осуществлять распределенную обработку информации, получаемой с объекта.

Структурная схема L-783 представлена на рисунке 3. Интерфейсная микросхема (PLX PCI9050-1) обеспечивает скоростной обмен данными с ПЭВМ по шине PCI (до 10 Мбайт/сек). Цифровой сигнальный процессор (ADSP-2184 / ADSP-2186) управляет всей периферией на плате, а именно: коммутатором, программируемым усилителем, АЦП (LTC1412), ЦАП (AD7249) и цифровыми линиями. DSP имеет внутреннюю память программ и данных и обладает своим собственным контроллером прямого доступа к памяти (ПДП) для доступа к любой ячейке памяти. Такая организация позволяет обращаться к памяти процессора, не прерывая его работы, что удобно при построении программ, работающих в режиме реального времени. Протокол работы с каналом ПДП предусматривает также возможность загрузки управляющей программы (*LBIOS*), которая реализует требуемые алгоритмы управления и обработки измерительной информации. АЦП в составе L-783 производит преобразование аналоговых сигналов в 12-ти разрядный цифровой код в одном из четырех диапазонов (±5В, ±2.5В,

±1.25В, ±0.625В) с программируемой частотой дискретизации (до 3 МГц по одному каналу). Имеется линия для синхронизации ввода данных. Синхронизация ввода данных возможна как аналоговым сигналом (по уровню), так и цифровым (по фронту). В программируемом усилителе предусмотрено задание коэффициентов усиления. Коммутатор входных аналоговых сигналов обеспечивает подключение и опрос 16 дифференциальных каналов или 32 каналов с общей землей. Плата имеет также 16 цифровых входов, работа с которыми осуществляется через регистр в адресном пространстве ввода/вывода DSP. Наличие встроенного в DSP таймера и цифровых входов позволяет осуществлять ввод и преобразование периодических импульсных сигналов

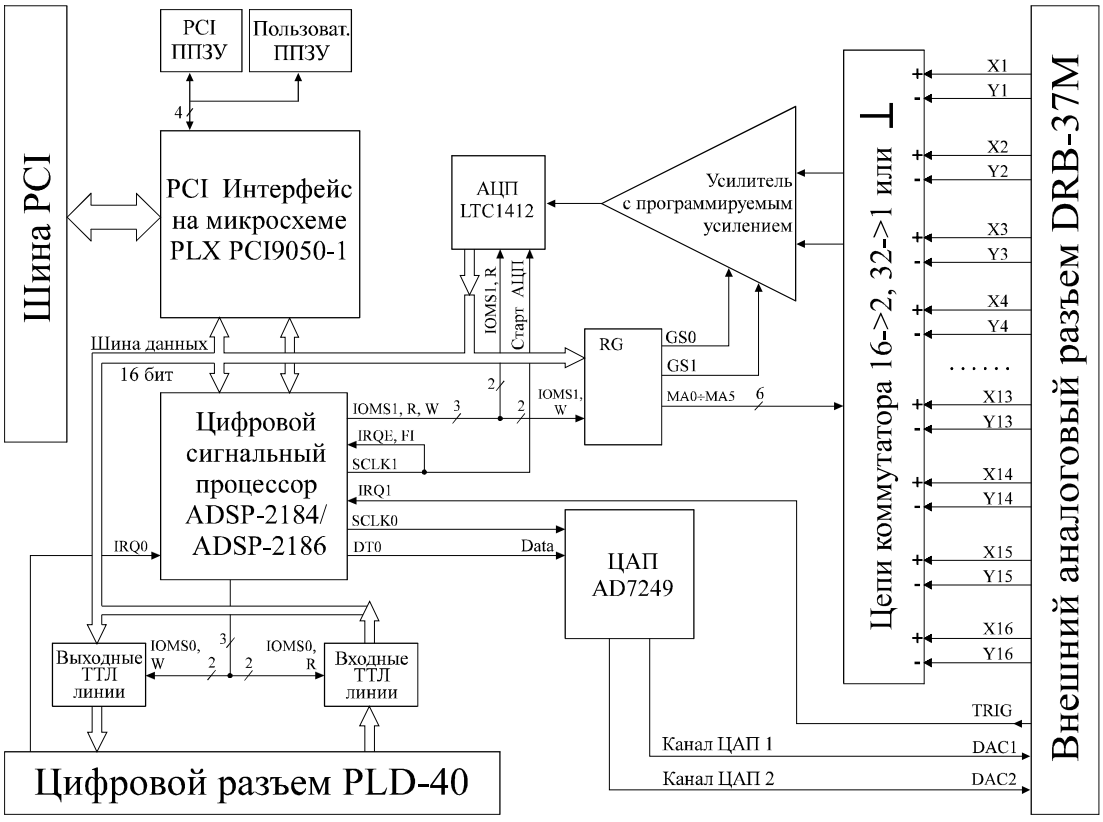

Рисунок 3 – Структурная схема L-783

Хотя на этой плате установлен DSP, как правило, большинству разработчиков конкретных систем измерения не приходится его программировать, поскольку в комплект поставки входит законченная управляющая программа для DSP (*LBIOS*), позволяющая осуществлять ввод-вывод с аналоговых каналов в самых различных режимах. В *LBIOS* реализованы наиболее часто используемые алгоритмы ввода-вывода. DSP может обеспечить ввод аналоговой информации и анализ ее в независимом от компьютера режиме с последующим сообщением о результатах анализа. В тоже время L-783 представляет собой законченную микропроцессорную систему, позволяющую продвинутому пользователю реализовать свои собственные специализированные алгоритмы обработки сигналов на уровне программирования установленного на плате сигнального процессора ADSP-2184/ADSP-2186 фирмы Analog Devices, Inc. (www.analog.com).

Для работы в среде Windows /98/NT/XP/7 на ЗАО «Л-Кард» разработано штатное программное обеспечение в виде драйвера платы и набора подпрограмм (API-функции), максимально упрощающих процесс сбора данных с плат в реальном масштабе времени.

В состав штатного ПО входит также программа *LGraph2*, которая представляет собой законченный программный комплекс, реализующий в сочетании с измерительными модулями ввода/вывода функции многоканального аналого-

цифрового самописца, осциллографа и анализатора сигналов. *LGraph2* позволяет регистрировать данные в файлы с последующим экспортом данных в *Matlab*, *Origin*, *Microsoft Excel*. Во время регистрации данных программа *LGraph2* позволяет осуществлять одновременную визуализацию вводимых данных в виде графиков.

3 Ввод аналоговых сигналов в *LBIOS* L-783

Исполняемый код *LBIOS* написан с учетом возможности взаимодействия драйвера с программой в ПЭВМ по так называемому «принципу команд». Это означает следующее: в выделенной ячейке данных памяти DSP (*L_COMMAND*) программой ПЭВМ указывается номер выполняемой команды; в DSP инициируется прерывание ***IRQ2***; обработчик этого прерывания в составе *LBIOS* выполняет все необходимые для данной команды действия; по окончании выполнения команды в ячейку *L_COMMAND* заносится число *0x0*.

Для реализации непрерывных во времени процессов сбора аналоговой информации и управления в *LBIOS* организованно два кольцевых буфера: для приема данных с АЦП и для выдачи данных на ЦАП. Данные с АЦП собираются в кольцевой буфер, организованный в памяти DSP. При заполнении части буфера генерируется прерывание в ПЭВМ. Виртуальный драйвер L-783 в составе API по этим прерываниям вычитывает данные и помещает их в большой кольцевой буфер, реализованный уже в ОЗУ компьютера. Большой кольцевой буфер драйвера доступен

пользовательскому приложению - имеется указатель на начало этого буфера. Кроме того, пользователю доступен счетчик заполнения буфера (тоже посредством указателя). Пользовательское приложение на персональном компьютере может либо забирать данные из буфера для сохранения непрерывного потока данных; либо обрабатывать данные на месте - тогда старые данные будут затираться новыми.

3.1 Используемые термины и форматы данных

Параметры штатного ПО платы L-783 в терминологии предметной области приведены в таблице 1. Для управления работой АЦП при сборе данных с входных аналоговых каскадов определяется такой параметр, как восьмиразрядный логический номер канала АЦП (*Channel*). С помощью логического номера канала АЦП задаются различные режимы функционирования платы посредством следующих битовых полей: физический номер аналогового канала; управление включением режима калибровки нуля, при этом вход каскада с программируемым коэффициентом усиления (PGA) просто заземляется; тип подключения входных каскадов - 16 дифференциальных входных аналоговых каналов или 32 входных канала с общей землёй; коэффициент усиления, причём для каждого канала можно установить свой индивидуальный коэффициент усиления. Таким образом, логический номер канала является управляющим словом для АЦП. Формат логического номера канала показан в

таблице 2, задание коэффициента усиления с помощью битов ***GS1*** и ***GS0*** – в таблице 3.

Например, логический номер канала равный 0x2 означает дифференциальный режим работы 3$^{\text{его}}$ канала (входа) с единичным усилением (диапазоном входного напряжения 5 В), 0x82 – с усилением равным 4 (диапазоном входного напряжения ±1.25 В), 0x22 – 3$^{\text{ий}}$ вход с единичным усилением в режиме работы с общей землей.

Таблица 1 – Термины

Название	Назначение
ADC_Rate	Частота работы АЦП в кГц
Inter_Kadr_Delay	Межкадровая задержка в мс
DAC_Rate	Частота работы ЦАП в кГц
Buffer	Указатель на целочисленный массив для ввода данных АЦП
N_Points	Число отсчетов в этом массиве
Channel	Логический номер канала
Control_Table	Управляющая таблица, содержащая целочисленный массив с логическими номерами каналов для циклического ввода данных с АЦП
Control_Table_Length	Длина управляющей таблицы

Таблица 2 – Формат логического номера канала АЦП

Номер бита	Обозначение	Функциональное назначение
0	***MA0***	0$^{\text{ой}}$ бит номера канала
1	***MA1***	1$^{\text{ый}}$ бит номера канала
2	***MA2***	2$^{\text{ой}}$ бит номера канала
3	***MA3***	3$^{\text{ий}}$ бит номера канала
4	***MA4***	Калибровка нуля/4$^{\text{ый}}$ бит номера канала
5	***MA5***	16 диф./32 общ.
6	***GS0***	0$^{\text{ой}}$ бит коэффициента усиления
7	***GS1***	1$^{\text{ый}}$ бит коэффициента усиления

Таблица 3 – Коэффициент усиления (биты ***GS1*** и ***GS0***)

Бит ***GS1***	Бит ***GS0***	Усиление	Диапазон, В
0	0	1	±5.0
0	1	2	±2.5
1	0	4	±1.25
1	1	8	±0.625

Массив логических номеров каналов АЦП (*Control_Table*) задает циклическую последовательность работы АЦП при вводе данных. Кадр отсчетов представляет собой последовательность отсчетов с логических каналов от *Control_Table[0]* до *Control_Table[Control_Table_Length-1]*. Временные параметры кадра для *Control_Table_Length=5* приведены на рисунке 4.

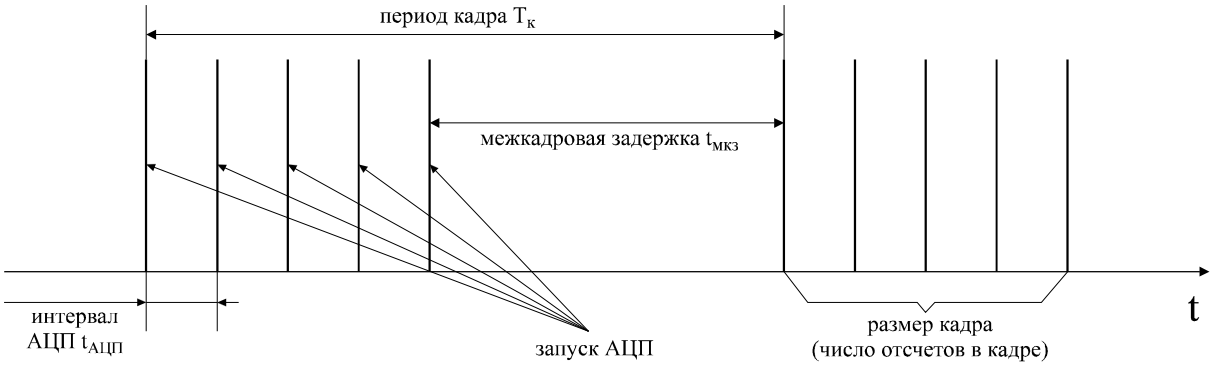

Рисунок 4 – Кадр отсчетов

Период кадра $T_к$ – это временной интервал между соседними кадрами, межкадровая задержка $t_{мкз}$=*Inter_Kadr_Delay* - временной интервал между последним отсчетом текущего кадра и первым отсчетом следующего, интервал $t_{АЦП}$ – интервал запуска АЦП или межканальная задержка. Тогда частота работы АЦП вычисляется как *ADC_Rate*=$1/t_{АЦП}$, при этом величина $t_{мкз}$

не может принимать значения меньшие, чем $t_{АЦП}$. Если размер кадра, т.е. число отсчетов АЦП в кадре, равен *Control_Table_Length*, то все эти временные интервалы можно связать следующей формулой:

$$T_к = (Control\_Table\_Length-1) * t_{АЦП} + t_{мкз},$$

или

$$T_к = (Control\_Table\_Length-1) / ADC\_Rate + Inter\_Kadr\_Delay.$$

Временные параметры *ADC_Rate* и *Inter_Kadr_Delay* используются в API штатного ПО для задания необходимого режима работы АЦП и *LBIOS*.

3.2 Алгоритмы управления сбором аналоговой информации

Ввод аналоговой информации осуществляется с помощью 12-ти разрядного АЦП LTC1412 [5], коммутатора измерительных каналов и программируемого усилителя. Для запуска преобразования в АЦП используются внутренне генерируемые тактовые синхроимпульсы ***SCLK1*** последовательного порта ***SPORT1***, при этом сам последовательный порт работает в режиме альтернативной конфигурации [6]. С помощью регистра управления работой ***SPORT1*** и делителя частоты синхроимпульсов ***SCLKDIV***, адреса которых в памяти данных *DM(0x3FF2)* и *DM(0x3FF1)* соответственно, можно разрешить или запретить генерацию этих импульсов, а также задавать частоту их следования, определяя, таким образом, частоту работы

АЦП. Частота работы АЦП определяется по следующей формуле:

$ADC\_Rate = f_q/(\boldsymbol{SCLKDIV}+1)$ кГц,

где f_q=20000.0 кГц – частота установленного на плате кварца, ***SCLKDIV*** – содержимое регистра делителя частоты синхроимпульсов ***SPORT1***.

Для обмена данными с АЦП используется регистр в пространстве ввода-вывода DSP *IO(1)*. Задний фронт синхроимпульса ***SCLK1*** заведен одновременно на линию запроса прерывания ***IRQE*** и входной флаг ***Flag In***.

Коммутация измерительного канала и задание коэффициента усиления сигнала происходит с использованием двух управляющих регистров - буферного регистра адреса/усиления канала и выходного регистра адреса/усиления канала. В обработчике прерывания ***IRQE*** первым делом надо записать в буферный регистр адреса/усиления следующее значение логического канала (см. таблицу 2), а потом уже считывать данные с АЦП.

Запись данных в буферный и выходной регистры производится через первую ячейку пространства ввода-вывода *IO(1)* в соответствии со значением флага ***FL0***:

– если ***FL0***=0 во время записи в *IO(1)*, то запись осуществляется в буферный регистр;

– если ***FL0***=1 во время записи в *IO(1)*, то запись осуществляется в выходной регистр.

Временные диаграммы работы АЦП приведены на рисунке 5. Данные из буферного регистра переписываются в выходной регистр по спадающему фронту сигнала ***BUSY*** АЦП. При этом данные, записанные в буферный регистр при обработке прерывания от выборки *N*, будут действовать для выборки *N*+2.

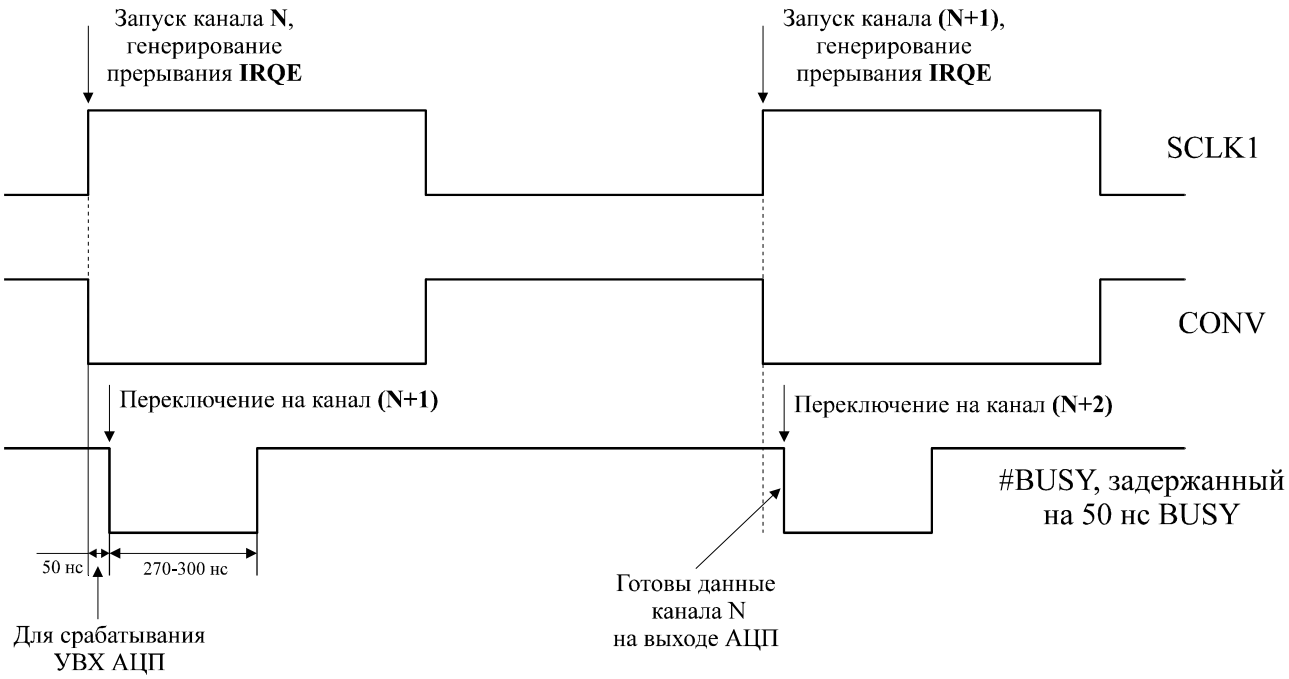

Рисунок 5 – Временные диаграммы работы АЦП

Листинг программы инициализации АЦП приведен на рисунке 6, а листинг подпрограммы обработки прерывания ***IRQE*** – на рисунке 7. Предполагается, что в штатном *LBIOS* управляющая таблица логических номеров каналов *Control_Table* реализована с помощью кольцевого буфера *(I2, M2, L2),* а буфер измеренных данных АЦП *Buffer* – с помощью кольцевого буфера *(I3, M3, L3).* В начале инициализации осуществляется выбор использования последовательного порта ***SPORT1*** в режиме альтернативной конфигурации (в режиме использования битовых

входов/выходов ***Flag In***, ***Flag Out, IRQ0*** и ***IRQ1***). Внутренне генерируемые синхроимпульсы ***SCLK1*** используются в качестве сигналов запуска АЦП, синхроимпульсы кадровой синхронизации не используются. На время инициализации АЦП остановлен установкой источника синхроимпульсов ***SCLK1*** в режим внешнего (так как внешний источник отсутствует, то циклический запуск АЦП приостанавливается).

```
{ Конфигурирование режимов SPORT в системном регистре 0x3FFF:  }
{ SPORT0 и SPORT1 - disable, SPORT1 – альтернативная конфигурация }
        AR=0x0400;
        DM(Sys_Ctrl_Reg)=AR;            { 0x3FFF – System Control Register }
{ Задание частоты запуска АЦП ADC_Rate=20000.0/(99+1)=200.0 кГц }
        AR = 99;                        { частота запуска АЦП (SCLK1) }
        DM(Sport1_Sclkdiv) = AR;        { 0x3FF1 - Serial Clock Divide Modulus}
{ Инициализация регистра кадровой синхронизации }
        AR = 0xF;                       { может быть любое число }
        DM(Sport1_Rfsdiv)=AR;       { 0x3FF0-Receive Frame Sync Divide Modulus }
{ Остановка АЦП - задание SCLK1 от внешнего генератора }
{ Управляющее слово для SPORT1: тактовый сигнал SCLK1 – внешний }
{ кадровая синхронизация – не используется }
        AR = 0x3C1F;                    { 0111 1100 0001 1111}
        DM(Sport1_Ctrl_Reg) = AR;       { 0x3FF2 – SPORT1 Control Register}
{ непосредственная подготовка к опросу }
{ очистка предыдущих запросов на прерывания в регистре сброса
прерываний}
        IFC = 0xFF; NOP;
{ разрешим прерывание IRQE в регистре разрешения прерываний}
        IMASK=0x10; NOP;
{ зададим усиление и номер канала для текущего и следующего отсчетов, }
{предполагая наличия кольцевого буфера с управляющей таблицей (I2,M2,L2)}
        AR=DM(I2, M2);
        SET FL0;
        IO(1)=AR;       { для текущего – в выходной регистр }
        RESET FL0;
        AR=DM(I2, M2);
        IO(1)=AR;       { для следующего – в буферный регистр }
{ запуск АЦП, т.е. сделаем SCLK1 внутренними }
        AR = 0x7C1F;                    { 0111 1100 0001 1111}
        DM(Sport1_Ctrl_Reg) = AR;       { 0x3FF2 – SPORT1 Control Register}
```

Рисунок 6 – Листинг программы инициализации АЦП

В режиме инициализации также задается номер измерительного канала и коэффициент усиления для текущего и следующего отсчетов – для корректной циклической работы в подпрограмме обработки прерывания ***IRQE***. Сама же циклическая работа стартует после того, как источник синхроимпульсов ***SCLK1*** устанавливается внутренним.

В подпрограмме обработки прерывания ***IRQE*** в буферном регистре устанавливается логический номер канала для следующего отсчета (прерывания), считывается результат преобразования и помещается в кольцевой буфер *Buffer*.

```
{ Обработчик прерывания IRQE в режиме высокоскоростного опроса }
IrqE_Handler:
    AF=AF+1, MR1=DM(I2,M2);
    IO(1)=MR1;   { усиление и номер канала для следующего отсчета }
    MR1=IO(1);
    DM(I3, M3)=MR1; { результат преобразования – в кольцевой буфер }
{  отведенные 4 байта закончились, поэтому RTI используется из
{ следующего по таблице векторов прерываний  обработчика }
BDMA_Handler:
    RTI; NOP; NOP; NOP;
```

Рисунок 7 – Листинг подпрограммы обработки прерывания ***IRQE***

Такой примитивный обработчик прерывания содержит лишь четыре ассемблерные команды и укладывается в таблицу векторов прерываний DSP[2]

[2] Здесь используется «программный трюк»: для возврата из прерывания используется команда *RTI*, принадлежащая следующему по таблице векторов обработчику прерываний.

4 Модификация штатной версии *LBIOS*

Реализация алгоритма управления сбором измерительной информации в системе измерений координат смещений торцов лопаток компрессора потребовала модификации *LBIOS* платы.

Основная идея такой модификации состоит в том, что подпрограмма обработки прерывания ***IRQE*** (рисунок 7) фактически является подпрограммой опроса АЦП, причем цикл опроса задействован только тогда, когда источник синхроимпульсов *SCLK1* устанавливается внутренним. Поэтому эту особенность можно использовать для пропуска моментов времени опроса, отличных от меток $t_{цi}$ (см. временную диаграмму на рисунке 1).

Запуск АЦП для первого канала кадра осуществляется одновременно с приходом импульса синхронизации от ГТИ на соответствующий цифровой вход платы, если порядковый номер этого импульса равен текущей метки времени опроса $t_{цi}$. Преобразование по следующему каналу из списка *Control_Table* осуществляется через фиксированный интервал времени, равный запрограммированному периоду опроса АЦП (1/*ADC_Rate*). Преобразования повторяются для всех каналов ОВТД в течение периода кадра $T_к$, который равен периоду следования импульса синхронизации.

Подпрограмма внешней синхронизации запускается с приходом каждого импульса синхронизации. Она выполняет счет

этих импульсов и, если текущее значение счетчика совпадает с очередным значением метки времени, разрешает работу подпрограммы опроса АЦП.

Для внешней синхронизации с ГТИ используется вход ***TRIG*** на внешнем разъеме аналоговых входов (Приложение А), который подключен к линии ***IRQ1*** процессора. В штатной версии *LBIOS* это прерывание сконфигурировано для работы по фронту и используется для внешней синхронизации ввода данных с АЦП, т.е. обработка прерывания осуществляется на каждый импульс ГТИ. Листинг подпрограммы обработки прерывания ***IRQ1*** показан на рисунке 8.

Предполагается, что массив меток времени опроса расположен в буфере *DM(I7, M7)*, при этом *M7=0*, а сдвиг указателя *I7* при необходимости осуществляется с использованием команды *MODIFY*.

```
{ Обработчик прерывания Irq1  }
Irq1Handler:
ENA SEC_REG;            { используется второй банк регистров }
AY0=DM(Count);
AR=AY0+1;             { подсчет импульсов ГТИ в ячейке Count и регистре AR }
DM(Count)=AR;
AY0=DM(I7,M7);          { считываем из буфера tцi без инкремента, M7=0 }
AR=AR-AY0;
IF NE RTI;              { t!= tцi – пропускаем импульс без обработки }
MODIFY(I7, M4);         { передвигаемся на следующую tцi, M4=1  }
DIS SEC_REG;            { возврат к первому банку регистров }
{ штатный код}
DM(Sport1_Ctrl_Reg)=MY0; {  разрешение SCLK1 }
RTI;
```

Рисунок 8 – Листинг подпрограммы обработки прерывания ***IRQ1***

Модернизированная таким образом подпрограмма обработки прерывания ***IRQ1*** игнорирует «лишние» импульсы ГТИ, а стандартные функции опроса АЦП и регистрации измеренных данных в кольцевом буфере разрешаются только для заданных меток времени $t_{ц1}$, $t_{ц2}$, …, $t_{цn_л}$.

Сброс счетчика импульсов (переменная *Count*) осуществляет подпрограмма измерения периода вращения, которая инициируется сигналом прерывания ***IRQ0*** от импульса ДЧВ и измеряет временной интервал между прерываниями с помощью встроенного таймера. В качестве эталонной частоты, заполняющей измеряемый период, используется частота внутреннего генератора DSP, поделенная на некоторый коэффициент (его значение указывается в регистре таймера *Tscale*). Измеренный код периода вращения фиксируется в выделенной ячейке памяти (*Nd*), из которой считывается при отработке соответствующей команды от драйвера ПЭВМ для расчета меток времени опроса каналов. Листинг подпрограммы обработки прерывания ***IRQ0*** показан на рисунке 9.

В подпрограмме обработки прерывания ***IRQ0*** также используется теневой набор регистров. После регистрации измеренного кода осуществляется повторный запуск таймера для измерения следующего периода вращения.

Вычисление и передачу в память DSP значений меток времени осуществляет подпрограмма вычислений меток времени, функционирующая в ПЭВМ.

```
{ Обработчик прерывания Irq0 }
Irq0Handler:
ENA SEC_REG;            { используется второй банк регистров }
AY0=DM(Tcount_reg); {считывание значения таймера в дополнительном коде}
DIS TIMER;              { запрет таймерного счета на время перезапуска }
AX0=0xFFFF;             { вычисление прямого кода }
AR=AX0-AY0;
DM(Nd)=AR;          { Nd – код пропорциональный периоду вращения ротора }

{ повторный запуск таймера }
DM(Tperiod_reg)=AX0;
DM(Tcount_reg)=AX0;
ENA TIMER;              { разрешение таймерного счета }

{ обнуление счетчика импульсов ГТИ Count }
MX0=0;
DM(Count)=MX0;
I7=^Tz;     { повторная инициализация указателя массива меток времени tqi }
DIS SEC_REG;            { возврат к первому банку регистров }
RTI;
```

Рисунок 9 – Листинг подпрограммы обработки прерывания ***IRQ0***

Таким образом, в кольцевом FIFO буфере, реализованном во внутренней памяти DSP, регистрируются по 3 кода каналов ОВТД на каждую лопатку. При заполнении буфера генерируется прерывание в ПЭВМ. Драйвер платы по этим прерываниям считывает данные и помещает их в большой кольцевой FIFO буфер, реализованный в оперативной памяти компьютера.

5 Лабораторный стенд для отладки и проверки работоспособности

Для проверки и отладки разработанных алгоритмов в лабораторной работе используется специальный стенд, имитирующий выходные сигналы ДЧВ и ОВТД в режиме реального времени. В его состав входят 3 банка ППЗУ, 3 высокоскоростных одноканальных ЦАП и устройство

управления. Устройство управления содержит генератор тактовых импульсов, счетчик адреса памяти и формирователи запуска ЦАП и внешнего синхроимпульса для системы измерения. С помощью программатора во все ППЗУ внешнего блока производится запись кодов, соответствующих имитируемым сигналам на выходе преобразователей кластера $ОВТД_1 \div ОВТД_3$ в виде напряжений ($U_1 \div U_3$), а в одно из ППЗУ – запись данных о скорости вращения ротора и соответствующем сигнале на выходе ДЧВ ($f_{ДЧВ}$) в виде дискретного сигнала. Имитация выходных сигналов датчика частоты вращения (ДЧВ) с формирователем и кластера ОВТД с преобразователями осуществляется следующим образом: в начале каждого тактового импульса устройства управления инкрементируется счетчик адреса, и выдаваемые коды из каждого банка ППЗУ поступают параллельно на входные шины ЦАП, жестко закрепленными за этими ППЗУ. С некоторым временным сдвигом (для завершения цифро-аналогового преобразования) формируется синхроимпульс для системы измерения.

6 Порядок выполнения работы

1. Изучите, как реализуется работа АЦП в штатной версии *LBIOS* (файлы *l783.dsp*, *const.h*, *var.h* находятся в рабочем каталоге лабораторной работы */L783LBIOS*). Попробуйте структурировать исходный текст в файле *l783.dsp* на блоки инициализации, подпрограммы обработки прерываний,

подпрограммы, реализующие опрос АЦП. Обратите внимание на то, какие при этом используются регистры данных и периферийных устройств DSP, по какому интерфейсу подключается АЦП, какие входы/выходы используются для запуска преобразования, для обнаружения завершения преобразования.

2. Выполните трансляцию штатного *LBIOS* с помощью командного файла *l783.bat*. Поместите полученный исполняемый файл *l783.bio* в корневой каталог программы ***LGraph2***.

3. Под руководством преподавателя подключите аналоговый разъем имитатора сигналов датчиков к аналоговому разъему платы L-783, цифровые линии – к цифровому разъему платы L-783. Включите источник питания имитатора.

4. Запустите программу ***LGraph2***. Укажите режимы работы для используемых первых трех измерительных каналов (максимальная частота опроса 3 МГц, покадровая синхронизация, коэффициент усиления – 4). Посмотрите измеряемые сигналы на экране «виртуального осциллографа». Найдите диапазоны измеряемых напряжений по всем трем каналам и временные интервалы, соответствующие огибающим лопатки и межлопаточному промежутку. Оцените период вращения ротора при условии, что на имитируемой ступени находятся 100 лопаток.

5. Запустите программу ***LGraph2*** в режиме «регистратора». Задайте имя файла данных, в котором будет сохраняться

измеряемая информация и время сбора (1-10 секунд). По окончании сбора также посмотрите графики собранной измерительной информации. Проверьте соответствие найденных в п.4 диапазонов и временных интервалов и зарегистрированных данных. Оцените, сколько периодов вращения ротора удалось зарегистрировать.

6. Модифицируйте *LBIOS* в соответствии с индивидуальным заданием. Повторите пункты 2-5 и проверьте работоспособность своей программы.

Лабораторная работа считается выполненной, если разработанная программа отлажена и функционирует в соответствии с индивидуальным заданием.

7 Контрольные вопросы

1. Как кольцевые буферы помогают организовать непрерывный сбор аналоговой информации в штатном ПО платы L-783?

2. Какие дискретные входы DSP используются для обнаружения завершения преобразования в АЦП?

3. Перечислите основные особенности организации прерываний в семействе ADSP 21xx. Какие вектора прерываний ADSP 2184/2186 используются в данной работе?

4. Для каких целей используется в семействе ADSP 21xx теневой набор регистров?

5. Какие действия инициализации необходимо выполнить для использования таймера в качестве преобразователя «Время-код»?

8 Индивидуальные задания

1. Реализовать опрос трех каналов ОВТД для массива интервалов меток времени $[t_{ц1_{low}}, t_{ц1_{high}}]$, $[t_{ц2_{low}}, t_{ц2_{high}}]$, …, $[t_{цn_{л_{low}}}, t_{цn_{л_{high}}}]$ с использованием модификации обработчика прерывания ***IRQ1***. Выборку и запись данных для каждого синхроимпульса ГТИ необходимо осуществлять только в интервале от $t_{цi_{low}}$ до $t_{цi_{high}}$, $i = 1..n_л$. Вне интервала синхроимпульсы ГТИ игнорируются. Массив $[t_{ц1_{low}}, t_{ц1_{high}}]$, $[t_{ц2_{low}}, t_{ц2_{high}}]$, …, $[t_{цn_{л_{low}}}, t_{цn_{л_{high}}}]$ задан в текстовом файле *tz.dat*.

2. Реализовать измерение периода вращения ротора с помощью встроенного таймера DSP и внешнего прерывания ***IRQ0.*** Выполнить инициализацию таймера для следующих исходных данных: диапазон измерений 5 мс - 50 мс, предельно допустимая погрешность измерения 1 мкс. Для проверки работоспособности программы с помощью ***LGraph2*** можно записывать измеренные коды периода в кольцевой FIFO буфер АЦП.

3*. Реализовать опрос трех каналов ОВТД по трем массивам интервалов меток времени с использованием модификации обработчика прерывания ***IRQ1***. Для каждого *j*-го канала ОВТД задан свой массив интервалов меток времени $[t_{цj1_{low}}, t_{цj1_{high}}]$,

$[t_{цj2_{low}}, t_{цj2_{high}}]$, …, $[t_{цjn_{л_{low}}}, t_{цjn_{л_{high}}}]$. Выборку и запись данных для каждого синхроимпульса ГТИ необходимо осуществлять только в интервале от $t_{цji_{low}}$ до $t_{цji_{high}}$, $j = 1..3$, $i = 1..n_л$. Вне интервала синхроимпульсы ГТИ игнорируются. Массивы меток времени $[t_{цj1_{low}}, t_{цj1_{high}}]$, $[t_{цj2_{low}}, t_{цj2_{high}}]$, …, $[t_{цjn_{л_{low}}}, t_{цjn_{л_{high}}}]$, $j = 1..3$ заданы в текстовых файлах *tz1.dat*, *tz2.dat*, *tz3.dat*, соответственно.

4*. Реализовать измерение периода вращения ротора с помощью встроенного таймера DSP и внешнего прерывания ***IRQ0.*** Выполнить инициализацию таймера для следующих исходных данных: диапазон измерений 5 мс - 200 мс, предельно допустимая погрешность измерения 1 мкс. Для увеличения разрядности таймера (диапазона измерения) можно использовать прерывание по переполнению таймера и дополнительную переменную – счетчик переполнений. Для проверки работоспособности программы с помощью ***LGraph2*** можно записывать измеренные коды периода в кольцевой FIFO буфер АЦП.

9 Литература

1. Беленький Л.Б., Скобелев О.П. Методические указания к выполнению лабораторной работы «Одновитковый вихретоковый датчик и устройство нормализации его сигнала». Самара, ГОУ ВПО «ПГАТИ», 2006.

2. Беленький Л.Б., Боровик С.Ю., Камаева О.И., Скобелев О.П., Тулупова В.В. Методические указания к выполнению лабораторной работы «Технические средства системы сбора измерительной информации для экспериментальных исследований газотурбинного двигателя». Самара, ГОУ ВПО «ПГАТИ», 2006.

3. Кластерные методы и средства измерения деформаций статора и координат смещений торцов лопаток и лопастей в газотурбинных двигателях/ Под общ. ред. Скобелева О.П. – М.: Машиностроение, 2011. – 298 с.

4. Платы L-761, L-780 и L-783. Техническое описание и инструкция по эксплуатации. ЗАО “Л-КАРД”, 2002. -104 с.

5. АЦП LTC1412 http://www.linear.com/product/LTC1412

6. Марков С. Цифровые сигнальные процессоры. Книга 1. М.: фирма МИКРОАРТ, 1996. – 144 с.

Приложение А. Описание разъема аналоговых входов платы L-783

На внешний разъем DRB-37M платы выведены следующие линии аналогового ввода/вывода (***X*** означают не инвертирующие входы платы, ***Y*** – инвертирующие):

N линии	**Назначение**	**N линии**	**Назначение**
1	вход X16	20	вход Y16
2	вход X15	21	вход Y15
3	вход X14	22	вход Y14
4	вход X13	23	вход Y13
5	вход X12	24	вход Y12
6	вход X11	25	вход Y11
7	вход X10	26	вход Y10
8	вход X9	27	вход Y9
9	вход X8	28	вход Y8
10	вход X7	29	вход Y7
11	вход X6	30	вход Y6
12	вход X5	31	вход Y5
13	вход X4	32	вход Y4
14	вход X3	33	вход Y3
15	вход X2	34	вход Y2
16	вход X1	35	вход Y1
17	AGND – аналоговая земля	36	GND32 - общий повод для 32х канального режима
18	DAC1 – выход ЦАП 1	37	TRIG – вход внешней ТТЛ синхронизации сигнала.
19	DAC2 – выход ЦАП 2		------

Приложение Б. Описание разъема цифровых входов платы L-783

Кабель AC-032 используется как переходник для перевода цифровых линий с разъема PLD-40, находящегося на плате, на заднюю панель Вашего компьютера в виде разъема DI-37M. Назначение выводов этого кабеля следующее:

N линии	Назначение	N линии	Назначение
1	GND	20	GND
2	VCC+5B	21	VCC+5B
3	OUT15	22	OUT16
4	OUT13	23	OUT14
5	OUT11	24	OUT12
6	OUT9	25	OUT10
7	OUT7	26	OUT8
8	OUT5	27	OUT6
9	OUT3	28	OUT4
10	OUT1	29	OUT2
11	IN15	30	IN16
12	IN13	31	IN14
13	IN11	32	IN12
14	IN9	33	IN10
15	IN7	34	IN8
16	IN5	35	IN6
17	IN3	36	IN4
18	IN1	37	IN2
19	INT – внешнее ТТЛ совместимое прерывание для DSP		------

Внимание!!! Подключение к данному разъему можно производить только при выключенном питании Вашего компьютера. Необходимо внимательно следить за тем, чтобы в процессе эксплуатации не было случайных замыканий между контактами разъема GND и VCC+5B, иначе плата может выйти из строя!!!

Приложение В. Программная модель и система команд ADSP21xx[3]

1 Обзор

С точки зрения программиста, процессор состоит из трех вычислительных устройств, двух генераторов адресов данных и генератора адресов инструкций, а также периферии на кристалле и памяти. Почти все операции с использованием этих архитектурных компонент оперируют с одним или большим количеством регистров - для сохранения данных, для слежения за такими значениями как указатели, или, например, для указания режимов работы.

Внутренние регистры содержат данные, адреса, управляющую или статусную информацию. Например, *AX0* содержит операнд АЛУ (данные); *I4* содержит указатель (адрес) генератора адресов данных; *ASTAT* содержит флаги статуса арифметических операций; поля, содержащиеся в регистре *DWAIT*, управляют количеством циклов ожидания, используемых при доступе к различным зонам памяти данных.

Есть два типа доступа к регистрам. Некоторые регистры, такие, как *MX0* и *IMASK*, могут быть непосредственно записаны или считаны ассемблерной инструкцией, например:

MX0=1234;

IMASK=0xF;

[3] Текст приложения носит справочный характер и соответствует работе [6].

Регистры, доступ к которым производится по адресам памяти, читаются и пишутся чтением или записью соответствующих адресов памяти данных. Например, следующий код очистит регистр *DWAIT*, который находится по адресу памяти данных *0x3FFE*:

```
AX0=0;
DM(0x3FFE)=AX0;
```

В этом примере *AX0* используется для задания константы 0, так как в системе команд нет инструкции для записи константы по непосредственному адресу.

Регистры, доступные в процессорах семейства, показаны на рисунке В-1. Не все из этих регистров доступны на конкретной модели процессора. Регистры сгруппированы по выполняемой ими функции: генераторы адресов данных (DAGs), генератор адресов инструкций (программный автомат), вычислительные устройства (ALU, MAC и SHIFTER), устройство обмена между шинами, интерфейс с памятью, последовательные порты, таймер, порт прямого доступа к памяти и аналоговый интерфейс.

1.1 Генераторы адресов данных

DAG1 и DAG2 имеют по 12 14-разрядных регистров: 4 индексных (*I*) регистра для хранения указателей, 4 регистра-модификатора (*M*) для модификации указателей и 4 регистра длины (*L*) для реализации кольцевых буферов. DAG1 адресует только память данных и имеет возможность битового обращения (реверс бит) генерируемого адреса. DAG2 адресует как память

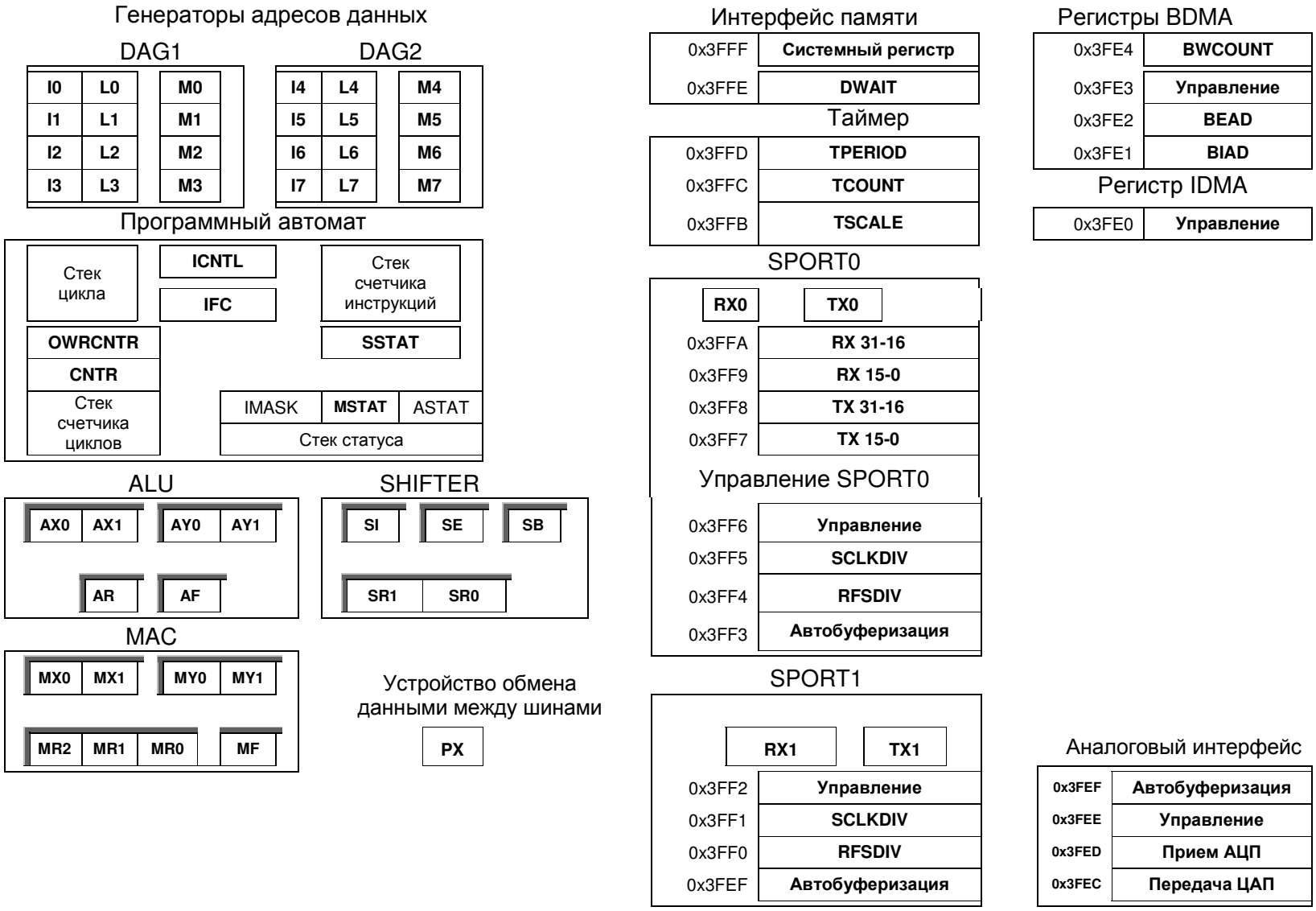

Рисунок В-1 – Регистры процессоров семейства ADSP-21xx

программ, так и память данных и может генерировать адреса для безусловных переходов (переходы и вызовы процедур), как и адреса для доступа к данным.

Например, инструкция:

AX0=DM(I0,M0);

производит косвенное чтение памяти данных (DM – Data Memory) по адресу, указываемому *I0*. После завершения чтения указатель *I0* модифицируется на значение, хранимое в *M0*.

Инструкция

PM(I4,M5)=MR1;

производит косвенную запись памяти инструкций (PM — Program Memory) по адресу, указываемому *I4* с пост-модификацией регистра *I4* регистром *M5*.

Инструкция

JUMP (I4);

показывает пример косвенного перехода.

1.2 Генератор адресов инструкций

Регистры, относящиеся к генератору адресов инструкций, управляют подпрограммами, циклами и прерываниями. Они также содержат его статус и выбирают режим операции.

- **Прерывания.** Регистр *ICNTL* управляет вложением прерываний и их чувствительностью ко внешним прерываниям; регистр *IFC* позволяет программно очищать и генерировать прерывания; регистр *IMASK* маскирует (запрещает) отдельные прерывания.

• **Счетчики циклов.** Регистр *CNTR* содержит значение счетчика текущего цикла. Стек счетчика циклов позволяет вложение циклов до 4 уровней с использованием аппаратного счетчика циклов. Запись в этот регистр сохраняет его текущее содержимое в стеке счетчика циклов перед записью нового значения. Например, инструкция

CNTR=10;

сохраняет текущее содержимое *CNTR* в стеке счетчика циклов и затем загружает в него значение 10.

Инструкция *OWRCNTR* позволяет изменить значение счетчика текущего цикла без его сохранения в стеке счетчика циклов.

• **Биты статуса и режима.** Регистр статуса стека (*SSTAT*) содержит флаги пустоты и полноты стеков. Регистр арифметического статуса (*ASTAT*) содержит статусные флаги вычислительных устройств. Регистр режима и статуса (*MSTAT*) содержит различные управляющие флаги. Он содержит 7 бит, которые управляют выбором дополнительного набора регистров, режимом реверса DAG1, режимом переполнения и насыщения АЛУ, расположением результата MAC, разрешением таймера и GO-режимом.

Используйте инструкции управления режимом (*ENA*, *DIS*) для удобного разрешения/запрещения режимов процессора.

• **Стеки.** Генератор адресов инструкций содержит четыре стека, которые позволяют проводить вложение циклов, подпрограмм и прерываний.

Стек счетчика инструкций имеет ширину 14 бит и глубиной в 16 позиций. Он содержит адреса возврата из подпрограмм и программ обработки прерываний, и адреса вершины цикла для циклов. Работа с этим стеком происходит автоматически при вызове подпрограмм и прерываний. Вдобавок, существуют инструкции (*PUSH*, *POP*) для ручного помещения и извлечения данных из стека счетчика инструкций.

Стек циклов имеет ширину 18 бит, 14 бит для хранения адреса конца цикла и 4 бита для хранения условия завершения цикла. Этот стек содержит 4 элемента. Значение в нем автоматически сохраняется при исполнении инструкции *DO UNTIL*. Это значение автоматически извлекается из него во время выхода из цикла, если цикл был вложенным. Значение из стека циклов может быть вручную извлечено инструкцией *POP LOOP*.

Стек статуса, значение в котором автоматически сохраняется при входе в процедуру обработки прерывания, вмещает содержимое регистров маскирования прерываний (*IMASK*), режима и статуса (*MSTAT*), и арифметического статуса (*ASTAT*). Глубина и ширина этого стека зависит от модели процессора, ибо различные модели поддерживают различное количество прерываний. Содержимое этого стека автоматически восстанавливается при исполнении инструкции *RTI* (возврат из прерывания). В/из стека статуса можно сохранять/восстанавливать значения вручную с помощью инструкций *PUSH STS* и *POP STS*.

Стек счетчика циклов имеет ширину 14 бит и содержит значения счетчиков (*CNTR*) для вложенных циклов,

использующих аппаратный счетчик. Текущее значение в него автоматически помещается при записи нового значения в регистр *CNTR*. Значение из стека счетчика циклов может быть вручную извлечено инструкцией *POP CNTR*.

1.3 Вычислительные устройства

Регистры вычислительных устройств содержат данные.

ALU и MAC требуют двух операндов для большинства операций. Регистры *AX0*, *AX1*, *MX0*, *MX1* содержат операнды X, а регистры *AY0*, *AY1*, *MY0*, *MY1* содержат операнды Y.

Регистры *AR* и *AF* содержат результаты ALU; *AF* может служить Y-операндом для следующей операции ALU, а *AR* может служить X-операндом для любого вычислительного устройства. Также, регистры *MR0*, *MR1*, *MR2* и *MF* содержат результаты MAC-а и могут использоваться как операнды для дальнейших вычислений. 16-битные регистры *MR0* и *MR1* вместе с 8-битным *MR2* могут содержать 40-битный результат MAC.

SHIFTER может получать исходные данные из ALU или MAC, из своих собственных выходных регистров или из специального входного регистра сдвигового массива (*SI*). Он хранит 32-битный результат в регистрах *SR0* и *SR1*. Регистр *SB* содержит экспоненту для блочных операций с плавающей точкой. Регистр *SE* содержит значение сдвига для операций нормализации и денормализации.

Регистры вычислительных устройств имеют своих "дублеров", как показано на рисунке B-1 в виде второго набора регистра, нарисованного "тенью". Выбор одного из этих наборов регистров

контролируется битом в регистре режима и статуса (*MSTAT*); этот бит устанавливается и очищается следующими инструкциями:

ENA SEC_REG; {выбрать вторичный набор регистров}

DIS SEC_REC; {выбрать первичный набор регистров}

1.4 Набор команд

Набор инструкций обеспечивает полный контроль над тремя вычислительными устройствами процессора: ALU, MAC и SHIFTER. Арифметические инструкции работают с 16-битными операндами непосредственно; также приняты необходимые меры для работы с числами повышенной точности.

Высокоуровневый синтаксис ассемблера процессоров семейства' одновременно хорошо читаем и эффективен. В отличие от многих ассемблеров, ассемблер ADSP 21xx использует алгебраическую нотацию для записи арифметических операций и пересылок данных, что приводит к высокой читаемости исходного кода. В то же время не происходит потери производительности: каждый оператор ассемблерной программы транслируется в одну 24-битную инструкцию, которая исполняется за один цикл. В наборе инструкций нет инструкций, выполняющихся дольше одного цикла (если доступ к памяти не требует циклов ожидания).

Во всех процессорах семейства инструкция *IDLE* введена для помещения процессора в состояние ожидания прерывания. Эта инструкция переводит процессор в состояние пониженного энергопотребления во время ожидания прерывания.

Поддерживаются два режима адресации для доступа к памяти. Прямая адресация использует непосредственное выражение в

качестве адреса памяти; косвенная адресация использует индексные (*I)* регистры генераторов адресов данных.

24-битное слово инструкции позволяет распараллелить операции. Набор инструкций позволяет исполнить за один цикл любую из следующих комбинаций инструкций:

• любую операцию ALU, MAC или SHIFTER, условную или безусловную;

• любую пересылку регистр-регистр;

• любое чтение или запись памяти данных;

• вычисление с одновременной пересылкой регистр-регистр;

• вычисление с одновременной записью или чтением памяти;

• вычисление с одновременным чтением 2 операндов из памяти.

Набор инструкций позволяет программировать с максимальной гибкостью. Он обеспечивает пересылку из любого регистра в любой, и пересылку из большинства регистров в память. В. дополнение почти любая инструкция ALU, MAC и SHIFTER может быть скомбинирована с любой пересылкой регистр-регистр или с пересылкой регистр-память.

Набор инструкций процессоров семейства ADSP-21xx разделен на следующие категории:

• Вычислительные: ALU, MAC и SHIFTER.

• Пересылки данных.

• Управление программой.

• Многофункциональные инструкции

• Различные инструкции.

2 Многофункциональные инструкции

Многофункциональные инструкции используют преимущества внутреннего параллелизма процессоров семейства, обеспечивая комбинацию пересылок данных, чтения/записи памяти и вычислений, все за один цикл.

2.1 Операции ALU/MAC с одновременным чтением памяти данных и памяти инструкций

Одной из наиболее часто используемых операций в цифровой обработке сигналов является сумма произведении, которая обычно выполняется так:

• Загрузить.2 операнда (коэффициент и значение в точке).

• Умножить операнды и сложить результат с предыдущими произведениями.

Процессоры семейства ADSP-21xx могут исполнять обе загрузки данных и умножение с накоплением за один цикл. Цикл умножений с накоплением может быть выписан на ассемблере процессоров семейства всего в две строчки. Так как память инструкций на чипе является достаточно быстрой для загрузки операнда и следующей команды, циклы такого типа могут исполняться с максимальной скоростью — за один процессорный цикл.

*MR=MR+MX0*MY0(SS), MX0=DM(I0,M0), MY0=PM(I4,M5);*

Первая часть этой инструкции (до первой запятой) суммирует предыдущее содержимое регистра *MR* с произведением регистров *MX0* и *MY0*, причем оба операнда считаются знаковыми (*SS*).

Вторая и третья части этой многофункциональной инструкции загружает два новых операнда. Один из них считывается из памяти данных (DM), указатель на данные находится в индексном регистре *I0*, после загрузки происходит пост-модификация указателя значением, содержащемся в регистре *M0*. Другой считывается из памяти инструкций (PM), указатель на данные находится в индексном регистре *I4*, после загрузки происходит пост-модификация указателя значением, содержащемся в регистре *M5*. Косвенная адресация памяти использует синтаксис, похожий на адресацию массива, где регистры генератора адресов данных предоставляют индекс и значение, на которое его необходимо изменить. Любой индексный регистр может быть скомбинирован с любым регистром модификации в пределах одного генератора адресов данных.

Регистры читаются в начале процессорного цикла и пишутся в конце. Операнды, находящиеся в регистрах *MX0* и *MY0* в начале цикла, умножаются, и результат добавляется к содержимому регистра *MR*. Новые операнды, загружаемые в конце того же процессорного цикла, перезаписывают старое содержимое регистров *MX0* и *MY0* после того, как умножение уже произошло, и доступны для вычислений в следующем цикле. Безусловно, можно загружать любые данные вместе с вычислениями.

Вычислительной частью такой инструкции может быть любая безусловная инструкция ALU или любая инструкция MAC, кроме насыщения. Также есть другие ограничения: следующий X-операнд должен загружаться в *MX0* из памяти данных и

следующий Y-операнд должен загружаться в *MY0* из памяти инструкций (внешняя и внутренняя память не отличаются с точки зрения набора инструкций). Результат вычислений должен идти в регистр результата (*MR* или *AR*), а не в регистр обратной связи (*MF* или *AF*).

2.2 Чтение памяти данных и памяти инструкций

Эта вариация многофункциональной инструкции есть специальный случай описанной только что инструкции, где вычислительная часть опущена. Происходит только двойная загрузка операндов, как показано ниже:

AR0=DM(I2,M0), AY0=PM(I4,M6);

Как и для предыдущей инструкции, X-операнд должен загружаться из памяти данных и Y-операнд должен загружаться из памяти инструкций.

2.3 Вычисление с чтением из памяти

Если производится одно чтение из памяти вместо двух, как было в двух предыдущих командах, можно исполнять более широкий класс вычислений. Возможны все операции ALU, кроме деления, все операции MAC, и все операции SHIFTER, кроме немедленного сдвига. Вычисление должно быть безусловным. Пример данного типа многофункциональной инструкции:

AR=AX0+AY0, AX0=DM(I0,M3);

Здесь производится сложение в ALU, и в то же время происходит загрузка операнда из памяти данных. Ограничения здесь такие же, как и для предыдущих инструкций. Значение *AX0*, используемое в качестве операнда сложения, является значением в

начале цикла. Операция чтения данных загружает новое значение в этот регистр в конце цикла. По этой самой причине, регистр-результат (*AR* в данном примере) не может быть приемником для чтения из памяти.

2.4 Вычисление с записью в память

Вычисление с записью в память аналогично по структуре вычислению с чтением из памяти, однако порядок инструкций меняется на противоположный. Сначала записывается операция записи в память, затем - вычисления, как показано ниже:

DM(I0,M0)=AR, AR=AX0+AY0;

Опять значение исходного регистра для записи в память (*AR* в данном примере) является значением в начале цикла. Вычисление приводит к записи нового значения в этот регистр в конце цикла. Перестановка инструкций в данной записи недопустима и приведет к генерации ассемблером предупреждения, ибо инструкция с измененным порядком частей требовала бы записи результата вычислений в память, так как нужно записывать старое значение регистра. Нет ограничений на использование одного и того же регистра (то есть можно использовать различные регистры в различных частях инструкции), хотя это и является наиболее органичным путем распараллеливания вычислений. Ограничения на вычислительные операции такие же, как и для вычисления с чтением из памяти. Возможны все операции ALU, кроме деления, все операции MAC, и все операции SHIFTER, кроме немедленного сдвига. Вычисление должно быть безусловным.

2.5 Вычисление с пересылкой регистр-регистр

Последний тип многофункциональных инструкций производит вычисление с пересылкой регистр-регистр. Почти все ограничения, накладываемые на предыдущие типы многофункциональных инструкций, применимы к данному типу.

AR=AX0+AY0, AX0=MR2;

Здесь происходит сложение и загрузка нового операнда в *AX0* из *MR2*.

Как и раньше, значение *AX0*, используемое в качестве операнда сложения, является значением в начале цикла. Пересылка регистр-регистр может быть использовать любой регистр ALU, MAC и SHIFTER (кроме регистров обратной связи *AF* и *MF* и регистра *SB*).

В этом примере, пересылка регистр-регистр загружает регистр *AX0* новым значением в конце цикла. Возможны все операции ALU, кроме деления, все операции MAC, и все операции SHIFTER, кроме немедленного сдвига. Вычисление должно быть безусловным.

3 Инструкции ALU, MAC и SHIFTER

Эта группа инструкций производит вычисления. Все эти инструкции могут исполняться условно, за исключением деления (ALU) и немедленного сдвига (SHIFTER).

3.1 Инструкции ALU

Вот пример инструкции ALU, сложение с переносом:

IF AC AR=AX0+AY0+C;

Условное выражение *IF AC* (которое может отсутствовать) проверяет флаг переноса ALU (*AC*); если предыдущая инструкция сгенерировала флаг переноса, выполняется данная инструкция, в противном случае происходит исполнение инструкции *NOP* и исполнение переходит к следующей инструкции. Алгебраическое выражение *AR=AX0+AY0+C* означает, что содержимое регистра *AR* будет содержать сумму *AX0* и *AY0* плюс значение флага переноса.

3.2 Инструкции MAC

Вот пример одной из инструкций MAC, умножение с суммированием:

*IF NOT MV MR=MR+MX0*MY0(UU);*

Условное выражение *IF NOT MV* проверяет флаг переполнения MAC. Если флаг сброшен, исполняется инструкция *NOP*. Выражение *MR=MR+MX0*MY0* - это операция умножения с суммированием: регистр результата (*MR*) получает в результате сумму своего предыдущего значения с произведением входных X и Y регистров. Модификатор в скобках (*UU*) сообщает процессору, что операнды беззнаковые. Такой модификатор может быть выбран из пяти доступных: (*SS*) означает что оба операнда знаковые; (*US*) означает, что первый операнд - беззнаковый, а второй— знаковый; (*SU*) означает, что первый операнд - знаковый, а второй - беззнаковый; (*RND*) означает округление результата (знаковых операндов).

3.3 Инструкции SHIFTER

Вот пример одной из инструкций - нормализация:

IF NOT CE SR=SR OR NORM SI (HI);

Условное выражение *IF NOT CE* проверяет условие "если счетчик числа повторений ненулевой". Если условие не выполняется, исполняется инструкция *NOP*. Приемником для всех операций SHIFTER является регистр результата *SR* (приемником для операции определения экспоненты является регистр *SE* или *SB*, как показано ниже). В этой примере, *SI* (входной регистр) является операндом. Длина и направление сдвига контролируется знаковым числом в регистре *SE* для всех операций сдвига, кроме немедленного сдвига. Положительные значения приводят к сдвигу влево; отрицательные - к сдвигу вправо. Модификатор "*SR OR*" (который может быть опущен) производит операцию “логическое или” результата с текущим содержимым регистра *SR*; это позволяет получить 32-битное значение в *SR* из 16-битных кусочков, "*NORM*" - это оператор, а (*HI*) -модификатор, который определяет, происходит ли сдвиг относительно верхней (*HI*) или нижней (*LO*) 16-битной части *SR*. Если модификатор "*SR OR*" опускается, результат сразу записывается в регистр *SR*.

4 Пересылка данных: чтение и запись

Инструкции пересылки данных производят пересылку данных в/из внешней памяти и регистров. Регистры разделены на 2 группы, первая из которых называется *reg* и содержит почти все регистры и *dreg*, регистры данных, которая является подмножеством первой. Лишь регистр *PC* (указатель инструкции) и регистры обратной связи ALU и MAC (*AF* и *MF*) недоступны для инструкций пересылки данных.

Пример инструкции:

AX0=DM(I0,M0);

5 Инструкции контроля исполнения программы

Управление исполнением программы для процессоров семейства простое, но мощное. Вот пример инструкции управления программой:

IF EQ JUMP my_label;

JUMP - это инструкция перехода. *my_label* - любой идентификатор, который вы хотите использовать как адрес перехода. Вместо метки, можно использовать индексный регистр DAG2. По умолчанию, область видимости для метки - модуль, в котором она объявлена. Директива ассемблера *.ENTRY* делает метку доступной как точку входа для процедур, внешних по отношению к модулю. А директива *.EXTERNAL* делает возможным использование метки, объявленной в другом модуле.

Если используется условие проверки счетчика (*CE*, *NOT CE*), счетчик должен быть ранее инициализирован исполнением инструкции *CNTR*. Инструкции *JUMP* и *CALL* допускают использование дополнительных условных выражений, "*FLAG_IN*" и "*NOT FLAG_IN*" для записи условных переходов в зависимости от состояния линии *FI*, но только с непосредственной адресацией, использовать DAG2 как источник адреса нельзя.

Инструкции *RTS* (возврат из подпрограммы) и *RTI* (возврат из процедуры обработки прерывания) обеспечивают условный возврат из подпрограммы или из процедуры обработки прерывания, соответственно.

Инструкция *IDLE* обеспечивает путь для ожидания прерываний. Эта инструкция заставляет процессор ожидать при пониженном потреблении энергии сигнала прерывания. После обслуживания прерывания, управление переходит на инструкцию, следующую за *IDLE*. При исполнении этой инструкции процессор использует меньше энергии, чем при исполнении циклов, созданных с помощью *JUMP*.

6 Другие инструкции

Есть несколько инструкций, не попавших ни в одну из приведенных групп. Это инструкция *NOP*, не выполняющая никакой операции. Инструкции *PUSH/POP* позволяют вручную контролировать стеки статуса, счетчика, счетчика команд и циклов; обслуживание прерываний автоматически работает с некоторыми из этих стеков.

Инструкция управления режимом используется для разрешения и запрещения некоторых режимов работы процессора.

За единственной командой *ENA* (ENAble - разрешить) или *DIS* (DISable - запретить) может следовать любое количество идентификаторов режимов, разделенное запятыми. Команда *ENA* или *DIS* может также повторяться. Все семь режимов, которые могут быть установлены или сброшены в одной инструкции, находятся в регистре *MSTAT*.

Инструкция *MODIFY* модифицирует адрес, находящийся в *I*-регистре, значением, хранящемся в *M*-регистре, без реального доступа к памяти. Как обычно, *I* и *M* регистры должны быть из

одного DAG; любой из *I0-I3* может быть использован с любым из *M0-M3*, то же для *I4-I7* и *M4-M7*.

Выходы *FO* (Flag Out), *FL0*, *FL1*, *FL2* могут быть установлены, сброшены или переключены инструкциями *SET*, *RESET* и *TOGGLE*. Эта инструкция обеспечивает внешние коммуникации процессора.

7 Структуры данных и переменные

Программное обеспечение поддерживает описание и использование одномерных массивов. Массив может содержать одно значение (переменную) или несколько значений (массив). Вдобавок, массив может использоваться как кольцевой буфер.

7.1 Массивы

Термины массив, буфер данных и переменная являются взаимозаменяемыми. Массивы объявляются ассемблерными директивами и могут быть косвенно адресованы или доступны непосредственно по имени, могут инициализироваться непосредственными в директиве или из внешних файлов данных, могут быть линейными или с кольцевой адресацией.

Массив объявляется директивой, подобной следующей:

.VAR/DM my_array[128];

Эта директива объявляет массив *my_array* из 128 16-битных элементов, расположенный в памяти данных (DM). Специальные операторы ^ и % обозначают адрес и длину массива, соответственно. Эти операторы могут использоваться как показано ниже:

I0=^my_array; {указатель на адрес массива}

MX0=DM(I0,M0); {загрузка MX0 из массива}

Эти инструкции загружают в регистр *MX0* значение первого элемента массива *my_array*. Используя возможность автоматической модификации адресов можно исполнять вторую из этих инструкций в цикле и таким образом поочередно адресовать все элементы массива.

Если же вам необходимо адресовать только первую ячейку массива можно непосредственно использовать имя массива как метку, например в следующих обстоятельствах:

MX0=DM(my_array);

Линкер подставляет реальный адрес этой метки. Также возможно инициализировать целый массив/буфер из файла данных с использованием директивы *.INIT*:

.INIT my_array: <filename.dat>;

Эта директива ассемблера считывает значение из файла *filename.dat* в массив во время сборки программы.

Массив или буфер данных с длиной один является простой переменной размером в слово, и объявляется так:

.VAR/DM my_param;

7.2 Кольцевые массивы

Достаточно распространенной задачей цифровой обработки сигналов является организация кольцевого буфера. Это непосредственно достигается использованием генераторов адресов данных (DAG) с использованием *L*-регистров. Сначала вы должны объявить буфер кольцевым:

.VAR/DM/CIRC my_array[128];

Это позволяет компоновщику поместить этот массив по допустимому адресу. Следующим шагом вы должны инициализировать *L*-регистр, обычно с использованием ассемблерного оператора % (или с использованием константы), а также *I*-регистр и *M*-регистр:

```
L0= %my_array;    {длина кольцевого буфера}
I0=^ my_array;    {указатель на адрес буфера}
M0=1;             (значение инкремента = 1}
```

Далее, инструкция:

```
MX0=DM(I0,M0);  {загрузка MX0 из буфера}
```

выполняемая в цикле, выбирает поочередно все элементы массива *my_array*, причем производится автоматический сдвиг к началу массива при достижении последнего элемента. Для нециклических массивов любой длины *L*-регистр (соответствующий *I*-регистру) необходимо устанавливать в *0*.

8 Косвенная линейная (не кольцевая) адресация

Косвенная адресация производится загрузкой адреса в индексный (*I*) регистр и указанием одного из доступных регистров-модификаторов (*M*).

Соответствующие им *L*-регистры используются для циклического перехода указателя для кольцевых буферов данных. Работа с буфером как с кольцевым происходит, только если *L*-регистр ненулевой. Напротив, для линейной адресации *L*-регистр, соответствующий *I*-регистру, должен быть установлен в нуль. Не думайте, что *L*-регистры автоматически инициализируются или могут быть проигнорированы; после сброса процессора регистры

I, *L* и *M* содержат случайные значения. Ваша программа должна инициализировать все *L*-регистры, соответствующие используемым *I*-регистрам.

9 Время исполнения инструкций

Все инструкции исполняются за один цикл, за исключением следующих случаев:

• Одновременный доступ к внешней памяти более одного раза за инструкцию.

• Циклы ожидания.

• Автобуферизация SPORT.

10 Система команд

10.1 Синтаксис описания инструкций

Все инструкции заканчиваются точкой с запятой. Запятая отделяет отдельные части в многофункциональной инструкции.

Квадратные скобки *[ ]*	Все, что в них находится, является необязательной частью инструкции.
Вертикальная линия /	Список операндов, разделяющихся вертикальной линией означает, что должен быть выбран один из операндов.

При описании регистров статуса используется следующие обозначения:

*	Звездочка означает, что данный бит изменяется после выполнения инструкции
-	Черточка означает, что данный бит не затрагивается инструкцией

0 или *1*	Означает, что этот бит всегда очищается или устанавливается инструкцией
cond	одно из условий *EQ*, *NE*, *GT*, *GE*, *LT*, *LE*, *NEG*, *POS*, *AV*, *NOT AV*, *AC*, *NOT AC*, *MV*, *NOT MV*, *NOT CE*
term	одно из условий *EQ*, *NE*, *GT*, *GE*, *LT*, *LE*, *NEG*, *POS*, *AV*, *NOT AV*, *AC*, *NOT AC*, *MV*, *NOT MV*, *CE*, *FOREVER*

Синтаксис	**Условие**	**Верно тогда, когда**
EQ	равен нулю	*AZ=1*
NE	не равен нулю	*AZ=0*
LT	меньше нуля	*AN XOR AV=1*
GE	больше либо равно нулю	*AN XOR AV=0*
LE	меньше либо равен нулю	*(AN XOR AV) OR AZ=1*
GT	больше нуля	*(AN XOR AV) OR AZ=0*
AC	перенос АЛУ	*AC=1*
NOT AC	нет переноса в АЛУ	*AC=0*
AV	переполнение АЛУ	*AV=1*
NOT AV	нет переполнения в АЛУ	*AV=0*
MV	переполнение умножителя	*MV=1*
NOT MV	нет переполнения умножит.	*MV=0*
NEG	X отрицателен	*AS=1*
POS	X положителен	*AS=0*
CE	число повторений истекло	
FOREVER	"вечный цикл"	

<data>=<const> | %<symbol> | ^<symbol>.

<addr>	Кодирует непосредственное значение адреса, которое должно быть помещено в инструкцию. Адрес может быть либо непосредственным значением (константой), либо программной меткой.
Конструкция *[,...]*	Означает сколько угодное повторение предыдущего параметра.
<ALU>,*<MAC>*, *<SHIFT>*	Операции ALU, MAC, SHIFTER соответственно
<dregs>	Один из регистров *AX0*, *AX1*, *AY0*, *AY1*, *AR*, *MX0*, *MX1*, *MY0*, *MY1*, *MR0*, *MR1*, *MR2*, *SI*, *SE*, *SR0*, *SR1*.
<regs>	Один из регистров *AX0*, *AX1*, *AY0*, *AY1*, *AR*, *MX0*, *MX1*, *MY0*, *MY1*, *MR0*, *MR1*, *MR2*, *SI*, *SE*, *SR0*, *SR1*, *I0-I7*, *M0-M7*, *L0-L7*, *SB*, *PX*, *ASTAT*, *MSTAT*, *SSTAT* (только для чтения), *IMASK*, *ICNTL*, *CNTR*, *OWRCNTR* (только для записи), *RX0*, *RX1*, *TX0*, *TX1*, *IFC* (только для записи).
<exp>	константа от -127 до 127

ALU Operation

xop - один из регистров *AX0*, *AX1*, *AR*, *MR2*, *MR1*, *MR0*, *SR1*, *SR0*.

yop - один из регистров *AY0*, *AY1*, *AF*.

MAC Operation

xop - один из регистров *MX0*, *MX1*, *AR*, *MR2*, *MR1*, *MR0*, *SR1*, *SR0*.

yop - один из регистров *MY0*, *MY1*, *MF*.

SHIFT Operation

xop - один из регистров *AR*, *MR2*, *MR1*, *MR0*, *SI*, *SR1*, *SR0*.

10.2 Команды

Команда	Операция	ASTAT: SS MV AO AS AC AV AN AZ
Операции ALU		
Сложение | с переносом	*[IF cond] AR|AF=xop (+yop)|(+C)|(+yop+C)*	- - - - * * * *
Вычитание| с заемом	*[IF cond] AR|AF=xop (-yop)|(+C-1)|(-yop+C-1)*	- - - - * * * *
	[IF cond] AR|AF=-xop (+yop)|{+C-1)|(+yop+C-1)	- - - - * * * *
Отрицание	*[IF cond] AR|AF=-xop|- yop*	- - - - * * * *
Инкремент	*[IF cond] AR|AF =yop+1*	- - - - * * * *
Декремент	*[IF cond] AR|AF =yop-1*	- - - - * * * *
Абсолютное значение	*[IF cond] AR|AF = ABS xop*	- - - * 0 * * *
"И" | "ИЛИ" | "Исключительно ИЛИ"	*[IF cond] AR|AF = xop AND|OR|XOR yop*	- - - - 0 0 * *
Пропускает вход на выход	*[IF cond] AR|AF = PASS xop|yop|-1| 0| 1*	- - - - 0 0 * *
"НЕ"	*[IF cond] AR|AF = NOT xop|yop|0*	- - - - 0 0 * *
Деление	*DIVS yop, xop* для данной команды *yop!=AY0*	- - * - - - - -
	DIVQ xop	- - * - - - - -
Операции MAC		
Умножение	*[IF cond] MR|MF= xop*yop(SS|SU|US|UU|RND)*	- * - - - - - -
Умножение со сложением	*[IF cond] MR|MF=MR+xop*yop(SS|SU|US|UU|RND)*	- * - - - - - -
Умножение с вычитанием	*[IF cond] MR|MF=MR-xop*yop(SS|SU|US|UU|RND)*	- * - - - - - -
Очистка	*[IF cond] MR|MF= 0*	- 0 - - - - - -
Передача *MR*	*[IF cond] MR|MF= MR[(RND)]*	- * - - - - - -
Условное насыщение *MR*	*IF MV SAT MR*	- - - - - - - -
Операции SHIFT		
Арифметический сдвиг	*[IF cond] SR = [SR OR] ASHIFT xop (HI|LO)*	- - - - - - - -
Логический сдвиг	*[IF cond] SR = [SR OR] LSHIFT xop (HI|LO)*	- - - - - - - -
Нормализация	*[IF cond] SR = [SR OR] NORM xop (HI|LO)*	- - - - - - - -
Выделение экспоненты	*[IF cond] SE = EXP xop (HI|LO|HIX)*	* - - - - - - -
Экспонента блока	*[IF cond] SB = EXPADJ xop*	- - - - - - - -
Немедленный арифметический сдвиг	*SR = [SR OR] ASHIFT xop BY <exp> (HI|LO)*	- - - - - - - -
Немедленный логический сдвиг	*SR = [SR OR] LSHIFT xop BY <exp> (HI|LO)*	- - - - - - - -

	Операция	ASTAT:
		SS MV AO AS AC AV AN AZ
Пересылка данных		
Пересылка регистра	*reg=reg*	- - - - - - - -
Загрузка регистра	*reg=<data>*	- - - - - - - -
Чтение памяти данных (непосредственная адресация)	*reg=DM(<addr>)*	- - - - - - - -
Чтение памяти данных (косвенная адресация)	*dreg=DM(Ik;Mn); n,k = 0-3 или n,k = 4-7*	- - - - - - - -
Чтение памяти программ (косвенная адресация)	*dreg=PM(Ik,Mn); n,k= 4-7*	- - - - - - - -
Запись памяти данных (непосредственная адресация)	*DM(<addr>)=reg*	- - - - - - - -
Запись памяти данных (косвенная адресация)	*DM(Ik,Mn)=dreg|<data>; n,k=0-3 или n,k = 4-7*	- - - - - - - -
Запись памяти программ (косвенная адресация)	*PM(Ik,Mn)=dreg; n,k = 4-7*	- - - - - - - -
Инструкции управления		
Переход	*[IF cond] JUMP (In) | <addr>; n=4-7*	- - - - - - - -
Вызов подпрограммы	*[IF cond] CALL (In) | <addr>; n=4-7*	- - - - - - - -
Переход или вызов по *FLAG_IN* выводу	*IF [NOT] FLAG_IN JUMP|CALL <addr>*	- - - - - - - -
Модификация вывода *FLAG_OUT*	*[IF cond] SET|RESET|TOOGLE FLAG_OUT [,FL0[,FL1[,FL2]]]; [,...]*	- - - - - - - -
Возврат из подпрограммы	*[IF cond] RTS*	- - - - - - - -
Возврат из прерывания	*[IF cond] RTI*	- - - - - - - -
Цикл *DO UNTIL*	*DO <addr> [UNTIL term]*	SSTAT: * 0 - - - - * 0
Стековые операции	*[[PUSH|POP] STS] [,POP CNTR] [,POP PC] [,POP LOOP]*	SSTAT: - * * * - * - *
	sreg=TOPSTACK; TOPSTACK=<sreg>, где sreg=dreg + DAGs	- - - - - - - -
Останов процессора	*IDLE[n], n=16,32,64,128*	- - - - - - - -

Команда	Операция	ASTAT:
		SS MV AO AS AC AV AN AZ
Установка режимов	*ENA|DIS (BIT_REV|AV_LATCH|AR_SAT|SEC_REG|G_MODE| M_MODE|TIMER[,…])*	- - - - - - - -
Модификация регистра адреса	*MODIFY (In,Mk); n,k=0-3; или n,k=4-7*	- - - - - - - -
Пустая операция	*NOP*	- - - - - - - -

Многофункциональные инструкции		
Вычисление с чтением из памяти	*<ALU>|<MAC>|<SHIFT>, dreg=DM(In,Mk)|PM(Ip,Mr); n,k=0-3 или n,k,p,r=4-7*	<ALU> - - - * * * * * <MAC> - * - - - - - - <SHIFT> * - - - - - - -
Вычисление с пересылкой регистра	*<ALU>|<MAC>|<SHIFT>, dreg=dreg*	<ALU> - - - * * * * * <MAC> - * - - - - - - <SHIFT> * - - - - - - -
Вычисление с записью памяти	*DM(In,Mk)|PM(Ip,Mr)=dreg, <ALU>|<MAC>|<SHIFT>; n,k=0-3 или n,k,p,r=4-7*	<ALU> - - - * * * * * <MAC> - * - - - - - - <SHIFT> * - - - - - - -
Чтение памяти данных и программ	*AX0|AX1|MX0|MX1=DM(In,Mk), AY0|AY1|MY0|MY1=PM(Ip,Mr); n,k=0-3 или n,k,p,r=4-7*	- - - - - - - -
Вычисление ALU| MAC с чтением памяти данных и программ	*<ALU>|<MAC> AX0|AX1|MX0|MX1=DM(In,Mk), AY0|AY1|MY0|MY1=PM(Ip,Mr); n,k=0-3 или n,k,p,r=4-7*	<ALU> - - - * * * * * <MAC> - * - - - - - -