Лабораторная работа № 3 «Ввод-вывод с DSK»

Цель работы: изучение основных принципов ввода и вывода сигнала с отладочной платы TMS320C6713 DSK; программно управлять АЦП и ЦАП, входящими в состав DSK; получение начальных навыков обработки аудиосигнала на ЦСП.

3.1. Ввод и вывод сигнала с отладочной платы

Типичные ЦОС-приложения должны включать базовую систему, показанную на рис.3.1, состоящую из АЦП, ЦСП и ЦАП. На входе часто используется сглаживающий фильтр для подавления частоты выше частоты Найквиста, определяемой как половина частоты выборки Fs. При отсутствии антиалиасингового фильтра возможен эффект наложения (aliasing), когда сигнал с частотой выше чем половина Fs маскируется сигналом с более низкой частотой.

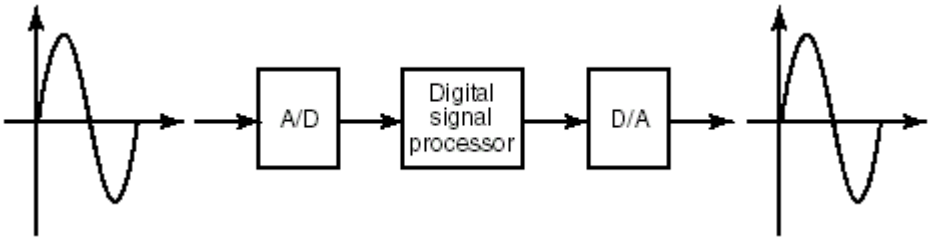

Рис. 3.1. Система цифровой обработки сигналов

Ядром системы является ЦСП TMS320C6713 [4]. Процессор выполняет программу, задаваемую разработчиком, реализуя требуемый алгоритм обработки сигнала. Функции АЦП/ЦАП выполняет микросхема кодека AIC23, обеспечивая ввод/вывод цифрового сигнала в процессор/из процессора. Кодек поддерживает частоты дискретизации в диапазоне 8 – 96 кГц. Аналоговые сигналы поступают на плату/выводятся с платы посредством стандартных аудиоразъемов линейного входа/выхода (Line in/Line out), микрофонного входа (Mic in) и выхода наушников (HP out).

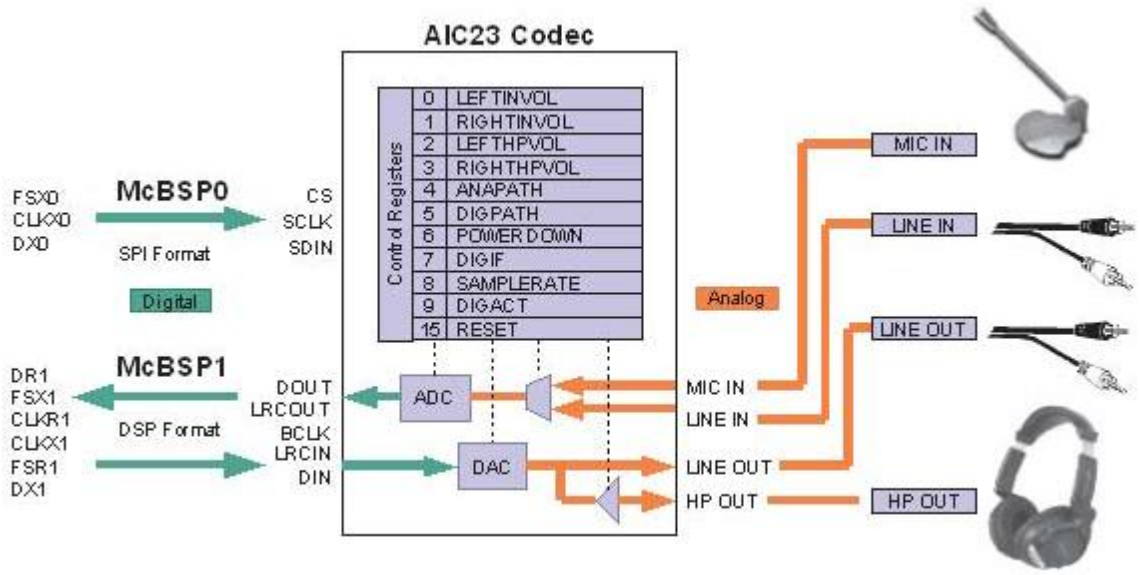

Рис. 3.2. Входы/выходы отладочной платы

3.2. Опрос кодека в цикле

Для включения DSK необходимы: USB-кабель для подключения DSK к ПК и кабель питания для подачи напряжения питания +5В через адаптер от розетки 220 В. Сначала подключается USB-кабель, затем подается питание на плату.

После включения платы необходимо открыть среду CCS. Запустите SetupCCS.exe и выберите конфигурацию 6713 DSK.

В качестве источника аудиосигнала используется ПК, а в качестве устройства воспроизведения – наушники. Таким образом, необходимо подключить разъем аудиовыхода ПК к линейному входу платы DSK (обозначение LINE IN на плате) с помощью аудиокабеля, а разъем выхода наушников платы DSK (обозначение HEADPHONE OUT на плате) подключить к наушникам.

Создайте новый проект, добавьте в него библиотеки:

rts6700.lib C:\\CCStudio_v3.1\C6000\cgtools\lib

csl6713.lib C:\\CCStudio_v3.1\C6000\csl\lib

dsk6713bsl.lib C:\CCStudio_v3.1\C6000\dsk6713\lib

rts6713.lib – библиотека поддержки реального времени. Библиотека описывает ресурсы архитектуры C671x и необходима Си-компилятору, чтобы переводить программы с языка Си на ассемблерные коды процессора C671x.

csl6713.lib – библиотека поддержки кристалла. Библиотека описывает ресурсы процессора. Эти описания затем используются в файле c6713dskinit.c для

конфигурации процессора. Например, здесь описана функция MCBSP_start(). В программе остается лишь вызывать эту функцию с нужными параметрами, и в результате автоматически начнет работать последовательный порт.

dsk6713bsl.lib – библиотека поддержки платы DSK. Библиотека описывает ресурсы платы. Эти описания затем используются в файле c6713dskinit.c для конфигурации платы. Например, здесь описана функция DSK6713_init(), автоматически настраивающая часть ресурсов платы. Тексты библиотечных функций скрыты от разработчика.

Командный файл c6713dsk.cmd:

```
        -c
        -heap  0x1000
        -stack 0x1000
        -u __vectors
        -u _auto_init

        _HWI_Cache_Control = 0;
        _RTDX_interrupt_mask = ~0x000001808;

        MEMORY
        {
         VECS:    o=00000000h  l=00000200h      /*    interrupt
vectors */
         PMEM:    o=00000200h  l=0000FE00h  /*   Internal   RAM
(L2) mem */
         BMEM:    o=80000000h  l=01000000h  /* CE0,  SDRAM,  16
MBytes   */

        }

        SECTIONS
        {
            .intvecs  >    0h
            .text     >    BMEM
            .rtdx_text     >    BMEM
            .far      >    BMEM
            .stack         >    BMEM
            .bss      >    BMEM
            .cinit         >    BMEM
            .pinit         >    PMEM
            .cio      >    BMEM
            .const         >    BMEM
            .data     >    BMEM
```

```
            .rtdx_data     >    BMEM
            .switch        >    BMEM
            .sysmem        >    BMEM
        }
```

Файл исходного кода lab5.c:

```
        #define CHIP_6713 1
        #include <dsk6713.h>
        #include <dsk6713_aic23.h>

        /* Codec configuration settings */
        DSK6713_AIC23_Config config = {
            0x0017, // 0 DSK6713_AIC23_LEFTINVOL   Left line
input channel volume
            0x0017, // 1 DSK6713_AIC23_RIGHTINVOL Right line
input channel volume
            0x00e9,  //  2  DSK6713_AIC23_LEFTHPVOL    Left
channel headphone volume
            0x00e9,  //  3  DSK6713_AIC23_RIGHTHPVOL  Right
channel headphone volume
            0x0011, //  4  DSK6713_AIC23_ANAPATH       Analog
audio path control
            0x0000, //  5  DSK6713_AIC23_DIGPATH      Digital
audio path control
            0x0000, // 6 DSK6713_AIC23_POWERDOWN  Power down
control
            0x0043, //  7  DSK6713_AIC23_DIGIF        Digital
audio interface format
            0x0081, // 8 DSK6713_AIC23_SAMPLERATE Sample rate
control
            0x0001   //  9  DSK6713_AIC23_DIGACT       Digital
interface activation
        };

        void main()
    {
         DSK6713_AIC23_CodecHandle hCodec;
         Uint32 input=0, output;
         /*  Initialize  the  board  support  library,  must  be
called first */
         DSK6713_init();

         /* Start the codec */
         hCodec = DSK6713_AIC23_openCodec(0, &config);
         while(1)
         {
           while (!DSK6713_AIC23_read(hCodec, &input));
```

```
            output = input;
            while (!DSK6713_AIC23_write(hCodec, output));
        }
            /* Close the codec */
            DSK6713_AIC23_closeCodec(hCodec);
    }
```

В данном примере используется непрерывный опрос кодека для определения готовности данных.

Функция DSK6713_AIC_read() осуществляет чтение данных с линии DRR многоканального буферизированного последовательного порта McBSP0.

Если в регистре готовности приемника RRDY бит 1 установлен в 1, то тестируется третий бит (0x2) регистра управления последовательным портом (SPCR) для определения готовности передачи информации с последовательного порта.

Функция DSK6713_AIC_write() осуществляет запись выходных данных в ЦАП через регистр передатчика DXR McBSP0. Вначале осуществляется операция AND содержимого регистра SPCR и константы 0x20000, которая проверяет готовность передачи данных (XRDY – 17-й бит регистра). Далее выполнение программы переходит в состояние ожидания внутри бесконечного цикла while(1) до момента, когда данные будут готовы для передачи. Тогда выполнение прерывается для очередного ввода и вывода цифровых выборок.

Укажите в настройках проекта (Project→Build Options→Compiler→Preprocessor→Include Search Path) путь к подключаемым файлам (C:\CCStudio_v3.1\C6000\dsk6713\include).

Если проект скомпилировался без ошибок, загрузите программу в ЦСП.

Включите воспроизведение музыкального фрагмента на ПК.

Запустите программу на выполнение.

Увеличьте громкость воспроизведения за счет цифровой обработки сигнала. Для этого достаточно перед выводом очередного отсчета умножить его на коэффициент усиления.

Изменять значение коэффициента усиления можно в процессе выполнения программы. Для этого добавьте переменную amplitude, которая будет

использоваться в качестве коэффициента усиления. Создайте gel-файл Amplitude.gel:

```
    /*Amplitude.gel  Create  slider  and  vary  amplitude  of
sinewave*/

    menuitem "Sine Amplitude"

    slider Amplitude(1,35,5,1,amplitudeparameter)
    /*incr by 5,up to 35*/
    {
       amplitude = amplitudeparameter;          /*vary amplit
of sine*/
    }
```

Загрузите gel-файл через меню File→Load GEL. Постройте проект (Rebuild All) и запустите его на выполнение. Вам станет доступна опция меню GEL→Sine Amplitude→Amplitude, с помощью которой можно изменять значение переменной amplitude, влияющей на громкость сигнала, в процессе выполнения программы.

3.3. Вывод сигнала с помощью прерывания

Измените файл исходного кода следующим образом:

```
#define CHIP_6713 1
#include <dsk6713.h>
#include <dsk6713_aic23.h>

/* Codec configuration settings */
DSK6713_AIC23_Config config = {
    0x0017, //  0  DSK6713_AIC23_LEFTINVOL   Left  line  input
channel volume
    0x0017, //  1  DSK6713_AIC23_RIGHTINVOL  Right  line  input
channel volume
    0x00d8,  //  2  DSK6713_AIC23_LEFTHPVOL    Left  channel
headphone volume
    0x00d8,  //  3  DSK6713_AIC23_RIGHTHPVOL  Right  channel
headphone volume
    0x0011, // 4 DSK6713_AIC23_ANAPATH     Analog audio path
control
    0x0000, // 5 DSK6713_AIC23_DIGPATH    Digital audio path
control
    0x0000, // 6 DSK6713_AIC23_POWERDOWN  Power down control
    0x0043,  //  7  DSK6713_AIC23_DIGIF         Digital  audio
interface format
    0x0001,   //   8   DSK6713_AIC23_SAMPLERATE   Sample   rate
control
```

```
    0x0001  // 9 DSK6713_AIC23_DIGACT      Digital interface
activation
};

interrupt void c_int11();
void initIRQ(int IRQ_id);

void main()
{
    DSK6713_AIC23_CodecHandle hCodec;

     /* Initialize the board support library, must be called
first */
    DSK6713_init();

    /* Start the codec */
    hCodec = DSK6713_AIC23_openCodec(0, &config);
    DSK6713_AIC23_setFreq(hCodec,1);
    // Configure buffered serial ports for 32 bit operation
(L+R in one read/write)
    MCBSP_FSETS(SPCR1, RINTM, FRM);
    MCBSP_FSETS(SPCR1, XINTM, FRM);
    MCBSP_FSETS(RCR1, RWDLEN1, 32BIT);
    MCBSP_FSETS(XCR1, XWDLEN1, 32BIT);

    initIRQ(11);
    while(1);
    /* Close the codec */
    DSK6713_AIC23_closeCodec(hCodec);
}

interrupt void c_int11()
{
    Uint32 temp;
    temp = MCBSP_read(DSK6713_AIC23_DATAHANDLE); // read L+R
channels
    MCBSP_write(DSK6713_AIC23_DATAHANDLE,temp); // write L+R
channels
}

void initIRQ(int IRQ_id)
{
    // Globally disables interrupts by clearing the GIE bit
of the CSR register.
    IRQ_globalDisable();
    // map the event ID associated with the handling of
McBSP1
```

```
    // with the physical interrupt IRQ_id
    IRQ_map(IRQ_EVT_RINT1, IRQ_id);
    // resets the event ID by disabling then clearing it.
    IRQ_reset(IRQ_EVT_RINT1);
    // Globally enables interrupt. This function globally
enables
    // interrupts by setting the GIE bit of the CSR register
to 1.
    IRQ_globalEnable();
    // Enables the NMI interrupt event
    IRQ_nmiEnable();
    // enable the specified event
    IRQ_enable(IRQ_EVT_RINT1);
}
```

Этот пример может использоваться как основная программа для построения других проектов, использующих встроенный кодек. Например, чтобы создать цифровой фильтр, необходимо вставить соответствующий алгоритм между функциями "ввод" и "вывод".

После инициализации DSK программа входит в бесконечный цикл в ожидании прерывания. Во время прерывания выполнение продолжается в программе обработки прерывания (ISR) c_int11. Прерывания вызываются с интервалом Ts = 1/Fs = 1 / (48 кГц) = 0.0208 мс, во время которого входное значение выборки читается из кодека АЦП, а затем посылается как выходное на кодек ЦАП.

После обработки прерывания выполнение возвращается к оператору while(1), ожидающему последующего прерывания (обратите внимание, что вместо ожидания в пределах бесконечного цикла while(1), можно обрабатывать произвольный код).

Добавьте к проекту файл vectors.asm:

```
*-----------------------------------------------------------
------------------
* Global symbols defined here and exported out of this file
*-----------------------------------------------------------
------------------
    .global _vectors ;global symbols
    .global _c_int00
    .global _vector1
    .global _vector2
    .global _vector3
    .global _vector4
    .global _vector5
    .global _vector6
```

```
    .global _vector7
    .global _vector8
    .global _vector9
    .global _vector10
    .global _c_int11 ; Hookup the c_int11 ISR in main()
    .global _vector12
    .global _vector13
    .global _vector14
    .global _vector15
*-------------------------------------------------------
------------------
* Global symbols referenced in this file but defined
somewhere else.
* Remember that your interrupt service routines need to be
referenced here.
*-------------------------------------------------------
------------------
    .ref _c_int00 ;entry address
*-------------------------------------------------------
------------------
* This is a macro that instantiates one entry in the
interrupt service table.
*-------------------------------------------------------
------------------
RST_VEC_ENTRY .macro addr
    NOP
    MVKL addr,B0
    MVKH addr,B0
    B B0
    NOP
    NOP 2
    NOP
    NOP
    .endm

VEC_ENTRY .macro addr
    STW B0,*--B15
    MVKL addr,B0
    MVKH addr,B0
    B B0
    LDW *B15++,B0
    NOP 2
    NOP
    NOP
    .endm
*-------------------------------------------------------
------------------
```

```
* This is a dummy interrupt service routine used to
initialize the IST.
*-------------------------------------------------------
------------------
_vec_dummy:
    B B3
    NOP 5
*-------------------------------------------------------
------------------
* This is the actual interrupt service table (IST). It is
properly aligned and
* is located in the subsection .text:vecs. This means if you
don't explicitly
* specify this section in your linker command file, it will
default and link
* into the .text section. Remember to set the ISTP register
to point to this
* table.
*-------------------------------------------------------
------------------
    .sect ".vectors" ;aligned IST section
    .align 1024
_vectors:
_vector0: RST_VEC_ENTRY _c_int00 ;RESET
_vector1: VEC_ENTRY _vec_dummy ;NMI
_vector2: VEC_ENTRY _vec_dummy ;RSVD
_vector3: VEC_ENTRY _vec_dummy
_vector4: VEC_ENTRY _vec_dummy
_vector5: VEC_ENTRY _vec_dummy
_vector6: VEC_ENTRY _vec_dummy
_vector7: VEC_ENTRY _vec_dummy
_vector8: VEC_ENTRY _vec_dummy
_vector9: VEC_ENTRY _vec_dummy
_vector10: VEC_ENTRY _vec_dummy
_vector11:  VEC_ENTRY  _c_int11  ;  Hookup  the  c_int11  ISR  in
main()
_vector12: VEC_ENTRY _vec_dummy
_vector13: VEC_ENTRY _vec_dummy
_vector14: VEC_ENTRY _vec_dummy
_vector15: VEC_ENTRY _vec_dummy
```

В настройках проекта измените модель памяти (Project→Build Options→Compiler→Advanced→Memory Models→Far (--mem_model: data=far).

Выполните построение проекта, проверьте работу программы.

3.4. Генерация сигнала с использованием прерывания

Добавьте в проект таблицу sinetable, которая содержит выборки синусоиды:

```
#define SINE_TABLE_SIZE   48
Int16 sinetable[SINE_TABLE_SIZE] = {
0x0000,  0x10b4,  0x2120,  0x30fb,  0x3fff,  0x4dea,  0x5a81,
0x658b,
0x6ed8,  0x763f,  0x7ba1,  0x7ee5,  0x7ffd,  0x7ee5,  0x7ba1,
0x76ef,
0x6ed8,  0x658b,  0x5a81,  0x4dea,  0x3fff,  0x30fb,  0x2120,
0x10b4,
0x0000,  0xef4c,  0xdee0,  0xcf06,  0xc002,  0xb216,  0xa57f,
0x9a75,
0x9128,  0x89c1,  0x845f,  0x811b,  0x8002,  0x811b,  0x845f,
0x89c1,
0x9128,  0x9a76,  0xa57f,  0xb216,  0xc002,  0xcf06,  0xdee0,
0xef4c
};
```

Измените исходный текст проекта таким образом, чтобы на ЦАП поступали выборки синусоиды из таблицы.

Выполните построение проекта, проверьте работу программы.

Определите частоту сгенерированной синусоиды.

Изменить частоту дискретизации кодека можно с помощью функции:

```
DSK6713_AIC23_setFreq(hCodec,f);
```

Значение параметра f определяет выбранную частоту дискретизации (табл. 3.1). По умолчанию в кодеке используется частота дискретизации 48 кГц.

Таблица 3.1

f	Fs
1	8 кГц
2	16 кГц
3	24 кГц
4	32 кГц
5	44 кГц
6	48 кГц
7	96 кГц

Изменяя частоту дискретизации, оцените каким образом, влияет значение частоты дискретизации кодека на воспроизводимый сигнал.

3.5. Контрольные вопросы

1) Какие существуют средства ввода-вывода аналоговых сигналов на плату TMS320C6713 DSK?

2) Какие необходимы библиотеки для работы с кодеком?

3) Поясните принцип ввода сигнала в первом примере (пункт 3.2).

4) Поясните принцип ввода сигнала во втором примере (пункт 3.3).

5) Каким образом выбирается частота дискретизации кодека?