ЧАСТЬ II

ЛАБОРАТОРНАЯ РАБОТА №4

ИЗУЧЕНИЕ ОСНОВ ПРОГРАММИРОВАНИЯ ЦИФРОВЫХ СИГНАЛЬНЫХ ПРОЦЕССОРОВ (DSP) НА ПРИМЕРЕ ADSP2181".

А.М. АНОХИНА

1. Перечень разделов справочной литературы, необходимых для проведения работы

1.0. Программная модель и система команд процессоров семейства AD21XX.
1.1. ADSP2181 Data Sheet.
Структура процессора ADSP2181.
1.2. ADSP2100 Family User's Manual.
Система команд.
Система прерываний.
Последовательный порт.
Автобуферизация.
1.3. AD1847 Data Sheet.
Структура ЦАП/АЦП AD1847.
Управление ЦАП/АЦП AD1847.
1.4. EZKIT Lite Reference Manual.
Структура лабораторного устройства EZKIT Lite.
1.5. ADSP2100 Family. Assembler Tools & Simulator Manual.
I/O Operations, SERIAL PORTS.

2. Задание.

2.1. Разобраться в принципах работы программы MEANDR.DSP, предназначенной для генерации прямоугольных колебаний.
Загрузить программу в симулятор и изучить использование регистров процессора и их изменение в процессе работы программы.

2.2. Модифицировать программу MEANDR.DSP так, чтобы на выходе ЦАП
генерировались:

Выход ЦАП1 - прямоугольные импульсы, передний фронт которых совпадает с началом возрастающей части треугольных колебаний на ЦАП2, а задний фронт - с началом спадающей части треугольных колебаний.

Размах : от 0 до 0.5 диапазона
(Исходя из того, что полный диапазон ЦАП эквивалентен -1..1)
Выход ЦАП2 - треугольные колебания (наклонный возрастающий участок, затем наклонный спадающий участок и т.д.).
Размах (с точностью не хуже 1%): 25%..75% диапазона
Длительность возрастающей части (с точностью 1%): 5 мс
Длительность спадающей части (с точностью 1%): 13 мс

2.3. Проверить работоспособность программы на симуляторе.
Откомпилировать и скомпоновать программу.
Отладить при помощи симулятора.

2.4. Загрузить программу в лабораторное устройство EZKIT-Lite и изучить получаемые на выходе устройства сигналы при помощи осциллографа.

2.5. Придумать и реализовать алгоритм для генерации треугольных колебаний с заданными параметрами с точностью 0.01%. Используйте представление чисел "fractional" (дробное) как для подсчета текущей "фазы", так и для задания размаха колебаний. Повторите пп 2.3, 2.4.

3. Методические указания.

3.0. Местонахождение файлов.

Файлы, необходимые для выполнения лабораторной работы, находятся на диске H:, в подкаталоге LAB2. Если на диске H: нет подкаталога LAB2, необходимо скопировать одноименный каталог с диска J:. В дальнейшем по умолчанию будет подразумеваться, что текущим каталогом является H:\LAB2\.

3.1. Запуск симулятора осуществляется командой

sim2181 -a ezkit_lt -e файл-программы

Используйте командный файл для запуска симулятора.

Для освоения симулятора используйте встроенную HELP-систему.

Для изучения работы программы используйте окна:

- Program Memory
- Data Memory
- Computational Registers
- SPORT Registers

3.2. Редактирование текста программы можно производить встроенным редактором оболочки Norton Commander, либо редактором текстов NE.
Компиляция программы осуществляется командой:
asm21 имя-файла -2181
Компоновка:
ld21 имя-файла -a ezkit_lt -e имя-файла -x -g

Используйте командный файл для компиляции и компоновки программы.

3.3. Загрузка скомпонованной программы в EZKIT Lite производится
при помощи программы-монитора, которая запускается под управлением системы Windows.
Запуск Windows осуществляется командой win.

4. Рекомендации для выполнения задания:
Для ознакомления с принципом действия программы изучите ее работу в симуляторе.
Модификацию программы рекомендуется производить в той части, которая обеспечивает генерацию меандра, а также в части, инициализирующей переменные меандра.
Для вычисления мгновенного значения текущего наклонного участка можно воспользоваться операцией умножения, используя в качестве параметров начальное значение напряжения наклонного участка, номер отсчета относительно начала текущего наклонного участка и требуемое изменение напряжения за время одного отсчета.

5. Вопросы.
5.1. Почему на осциллографе не видна ступенчатость треугольного колебания? Почему при генерации меандра видны ярко выраженные колебания на вершинах импульсов до и после фронтов?

6. Дополнительное задание
Модифицируйте программу с целью генерации зацикленной последовательности из нескольких участков различного наклона и различной длительности.

```
{*************************************************************
*********
 *
 *  This sample program is organized into the following
sections:
 *
 *  Assemble time constants
 *  Interrupt vector table
 *  ADSP 2181 intialization
 *  ADSP 1847 Codec intialization
 *  Interrupt service routines

*************************************************************
*********



.module/RAM/ABS=0 loopback;



{*************************************************************
*********
 *
 *  Assemble time constants
 *

*************************************************************
*********

.const  IDMA=                   0x3fe0;
.const  BDMA_BIAD=              0x3fe1;
.const  BDMA_BEAD=              0x3fe2;
.const  BDMA_BDMA_Ctrl=         0x3fe3;
.const  BDMA_BWCOUNT=           0x3fe4;
.const  PFDATA=                 0x3fe5;
.const  PFTYPE=                 0x3fe6;

.const  SPORT1_Autobuf=         0x3fef;
.const  SPORT1_RFSDIV=          0x3ff0;
.const  SPORT1_SCLKDIV=         0x3ff1;
.const  SPORT1_Control_Reg=     0x3ff2;
.const  SPORT0_Autobuf=         0x3ff3;
.const  SPORT0_RFSDIV=          0x3ff4;
.const  SPORT0_SCLKDIV=         0x3ff5;
.const  SPORT0_Control_Reg=     0x3ff6;
.const  SPORT0_TX_Channels0=    0x3ff7;
.const  SPORT0_TX_Channels1=    0x3ff8;
```

```
.const  SPORT0_RX_Channels0=    0x3ff9;
.const  SPORT0_RX_Channels1=    0x3ffa;
.const  TSCALE=                 0x3ffb;
.const  TCOUNT=                 0x3ffc;
.const  TPERIOD=                0x3ffd;
.const  DM_Wait_Reg=            0x3ffe;
.const  System_Control_Reg=     0x3fff;


.var/dm/ram/circ    rx_buf[3];      /* Status + L data +
R data */
.var/dm/ram/circ    tx_buf[3];      /* Cmd + L data + R
data    */
.var/dm/ram/circ    init_cmds[13];
.var/dm             stat_flag;     { Status: 1 - init, 0
- wrk }

.var time;
.var time_max;

.var/circ meandr_values[4];

.init time: 0;
.init meandr_values: -16384, 5,16384,  5;

.init tx_buf:0xc000, 0x0000, 0x0000; /* Initially set MCE
*/

.init init_cmds:
     0xc002,     {
                Left input control reg
                b7-6: 0=left line 1
                      1=left aux 1
                      2=left line 2
                      3=left line 1 post-mixed loopback
                         b5-4: res
                         b3-0: left input gain x 1.5 dB
                      }
        0xc102,     {
                Right input control reg
                         b7-6: 0=right line 1
                               1=right aux 1
                      2=right line 2
                               3=right line 1 post-mixed
loopback
                b5-4: res
                         b3-0: right input gain x 1.5 dB
                      }
     0xc288,     {
                         left aux 1 control reg
                         b7  : 1=left aux 1 mute
```

```
                        b6-5: res
                        b4-0: gain/atten x 1.5, 08= 0dB,
00= 12dB
                }
        0xc388,     {
                        right aux 1 control reg
                b7  : 1=right aux 1 mute
                        b6-5: res
                b4-0: gain/atten x 1.5, 08= 0dB, 00= 12dB
                     }
        0xc488,     {
                left aux 2 control reg
                        b7  : 1=left aux 2 mute
                        b6-5: res
                        b4-0: gain/atten x 1.5, 08= 0dB,
00= 12dB
                     }
     0xc588,     {
                        right aux 2 control reg
                        b7  : 1=right aux 2 mute
                b6-5: res
                        b4-0: gain/atten x 1.5, 08= 0dB,
00= 12dB
                }
        0xc680,     {
                        left DAC control reg
                b7  : 1=left DAC mute
                        b6  : res
                        b5-0: attenuation x 1.5 dB
                     }
        0xc780,     {
                right DAC control reg
                        b7  : 1=right DAC mute
                        b6  : res
                b5-0: attenuation x 1.5 dB
                     }
     0xc85c,     {
                        data format register
                        b7  : res
                b5-6: 0=8-bit unsigned linear PCM
                              1=8-bit u-law companded
                              2=16-bit signed linear PCM
                              3=8-bit A-law companded
                        b4  : 0=mono, 1=stereo
                b0-3: 0=  8.
                      1=  5.5125
                      2= 16.
                      3= 11.025
                      4= 27.42857
                      5= 18.9
                      6= 32.
```

```
                    7= 22.05
                    8=   .
                    9= 37.8
                    a=   .
                    b= 44.1
                    c= 48.
                    d= 33.075
                    e=  9.6
                    f=  6.615
                  (b0) : 0=XTAL1 24.576 MHz; 1=XTAL2
16.9344 MHz
                     }
     0xc909,     {
                        interface configuration reg
                        b7-4: res
               b3  : 1=autocalibrate
                        b2-1: res
                        b0  : 1=playback enabled
                     }
    0xca00,     {
               pin control reg
                        b7  : logic state of pin XCTL1
                        b6  : logic state of pin XCTL0
               b5  : master - 1=tri-state CLKOUT
                              slave  - x=tri-state CLKOUT
               b4-0: res
                     }
      0xcc40,     {
               miscellaneous information reg
                b7  : 1=16 slots per frame, 0=32 slots
per frame
                b6  : 1=2-wire system, 0=1-wire system
                b5-0: res
                     }
     0xcd00;     {
                        digital mix control reg
                        b7-2: attenuation x 1.5 dB
               b1  : res
                        b0  : 1=digital mix enabled
             }
*************************************************************
********
 *
 *  Interrupt vector table
 *
*************************************************************
*********
        jump start;  rti; rti; rti;     {00: reset }
        rti;         rti; rti; rti;     {04: IRQ2 }
        rti;         rti; rti; rti;     {08: IRQL1 }
        rti;         rti; rti; rti;     {0c: IRQL0 }
```

```
        jump next_tx;                   {10: SPORT0 tx }
                        rti; rti; rti;
        jump input_samples;             {14: SPORT1 rx }
                        rti; rti; rti;
        jump irqe;   rti; rti; rti;     {18: IRQE }
        rti;         rti; rti; rti;     {1c: BDMA }
        rti;         rti; rti; rti;     {20: SPORT1 tx or
IRQ1 }
        rti;         rti; rti; rti;     {24: SPORT1 rx or
IRQ0 }
        rti;         rti; rti; rti;     {28: timer }
        rti;         rti; rti; rti;     {2c: power down }
************************************************************
********
 *
 *  ADSP 2181 intialization
 *
************************************************************
*********
start:
        i0 = ^rx_buf;
        l0 = %rx_buf;
        i1 = ^tx_buf;
        l1 = %tx_buf;
        i3 = ^init_cmds;
        l3 = %init_cmds;
        i4 = ^meandr_values;
     l4 = %meandr_values;
     m1 = 1;
     m5 = 1;
{ DAC variables init }
     ar = dm (i4,m5);
     dm (tx_buf+1) = ar; { DAC1 value }
     dm (tx_buf+2) = ar;     { DAC2 value }
     ar = dm (i4,m5);
     dm (time_max) = ar;
{====== S E R I A L   P O R T   #0   S T U F F
============}
ax0 = b#0000001010000111;   dm (SPORT0_Autobuf) = ax0;
    {   ||||!-/!/|-/|/|+- receive autobuffering 0=off,
1=on
        ||||!  ! |  | +-- transmit autobuffering 0=off,
1=on
        ||||!  ! |  +---- | receive m?
        ||||!  ! |        | m1
        ||||!  ! +------- ! receive i?
        ||||!  !          ! i0
        ||||!  !          !
        ||||!  +========= | transmit m?
        ||||!             | m1
        ||||+------------ ! transmit i?
```

```
        |||!              ! i1
        |||!              !
        |||+============= |BIASRND MAC biased rounding
control bit
        ||+-------------- 0
        |+--------------- | CLKODIS CLKOUT disable
control bit
        +---------------- 0           }



 ax0 = 0;    dm (SPORT0_RFSDIV) = ax0;
         {   RFSDIV = SCLK Hz/RFS Hz - 1 }
 ax0 = 0;    dm (SPORT0_SCLKDIV) = ax0;
            {   SCLK = CLKOUT / (2  (SCLKDIV + 1) }

 ax0 = b#1000011000001111;   dm (SPORT0_Control_Reg) =
ax0;
     {   multichannel
         ||+--/|!||+/+---/ | number of bit per word - 1
         |||   |!!!!       | = 15
         |||   |!!!!       |
         |||   |!!!!       |
         |||   |!||+====== ! 0=right just, 0-fill;
1=right just,       signed
         |||   |!||        ! 2=compand u-law; 3=compand
A-law
         |||   |!|+------- receive framing logic 0=pos,
1=neg
         |||   |!+-------- transmit data valid logic
0=pos, 1=neg
         |||   |+========= RFS 0=ext, 1=int
         |||   +---------- multichannel length 0=24, 1=32
words
         ||+-------------- | frame sync to occur this
number of clock
         ||                | cycle before first bit
         ||                |
         ||                |
         |+---------------- ISCLK 0=ext, 1=int
         +----------------- multichannel 0=disable,
1=enable}



          {   non-multichannel
         |||!|||!!!!+---/ | number of bit per word - 1
         |||!|||!!!!      | = 15
         |||!|||!!!!      |
         |||!|||!!!!      |
```

```
         |||!|||!|||+===== !0=righ just,0-fill;1=right
just,signed
         |||!|||!||+------ ! 2=compand u-law; 3=compand
A-law
         |||!|||!|+------- receive framing logic 0=pos,
1=neg
         |||!|||!+-------- transmit framing logic 0=pos,
1=neg
         |||!|||+========= RFS 0=ext, 1=int
         |||!||+---------- TFS 0=ext, 1=int
         |||!|+----------- TFS width 0=FS before
data,1=FS in sync
         |||!+------------ TFS 0=no, 1=required
         |||+============= RFS width 0=FS before
data,1=FS in sync
         ||+-------------- RFS 0=no, 1=required
         |+--------------- ISCLK 0=ext, 1=int
         +---------------- multichannel 0=disable,
1=enable      }

     ax0 = b#0000000000000111;   dm (SPORT0_TX_Channels0)
= ax0              {            ^15          00^
transmit word enables: channel # == bit # }
     ax0 = b#0000000000000111;   dm (SPORT0_TX_Channels1)
= ax0;
         {   ^31          16^   transmit word enables:
channel # == bit # }
        ax0 = b#0000000000000111; dm
(SPORT0_RX_Channels0) = ax0;
         {      ^15          00^   receive word enables:
channel # == bit # }
        ax0 = b#0000000000000111; dm
(SPORT0_RX_Channels1) = ax0;
           {   ^31          16^   receive word enables:
channel # == bit # }

{=== S Y S T E M   A N D   M E M O R Y   S T U F F
======}
        ax0 = b#0000111111111111;   dm (DM_Wait_Reg) =
ax0;
            {   |+-/+-/+-/+-/+-/- ! IOWAIT0
                ||  |  !  |       !
                ||  |  !  |       !
                ||  |  !  +------ | IOWAIT1
                ||  |  !           |
                ||  |  !           |
                ||  |  +--------- ! IOWAIT2
                ||  |              !
                ||  |              !
                ||  +------------ | IOWAIT3
                ||                 |
```

```
                ||                  |
                |+================ ! DWAIT
                |                   !
                |                   !
                +------------------ 0          }


        ax0 = b#0001000000000000;   dm
(System_Control_Reg) = ax0;
             {   +-/!||+-----/+-/- | program memory wait
states
                 |  !|||           | 0
                 |  !|||           |
                 |  !||+---------- 0
                 |  !||            0
                 |  !||            0
                 |  !||            0
                 |  !||            0
                 |  !||            0
                 |  !||            0
                 |  !|+----------- SPORT1 1=serial port,
0=FI, FO,                    IRQ0, IRQ1,..
                 |  !+------------ SPORT1 1=enabled,
0=disabled
                 |  +============= SPORT0 1=enabled,
0=disabled
                 +------------------ 0
                                     0
                                     0            }

        ifc = b#00000011111111;      { clear pending
interrupt }
        nop;
       icntl = b#00000;
           {     ||||+- | IRQ0: 0=level, 1=edge
                 |||+-- | IRQ1: 0=level, 1=edge
                 ||+--- | IRQ2: 0=level, 1=edge
                 |+---- 0
                 |----- | IRQ nesting: 0=disabled,
1=enabled
            }
        mstat = b#1000000;
            {     ||||||+- |Data register bank select
                  |||||+-- |FFT bit reverse mode (DAG1)
                  ||||+--- |ALU overflow latch mode,
1=sticky
                  |||+---- |AR saturation mode,
1=saturate, 0=wrap
                  ||+----- | MAC result, 0=fractional,
1=integer
                  |+------ | timer enable
```

```
                  +------- | GO MODE            }

*************************************************************
********
 *
 *  ADSP 1847 Codec intialization
 *
*************************************************************
*********

        {   clear flag }
        ax0 = 1;
        dm(stat_flag) = ax0;




{   enable transmit interrupt }
        imask = b#0001000000;
            {     |||||||||+ | timer
                  ||||||||+- | SPORT1 rec or IRQ0
                  |||||||+-- | SPORT1 trx or IRQ1
                  ||||||+--- | BDMA
                  |||||+---- | IRQE
                  ||||+----- | SPORT0 rec
                  |||+------ | SPORT0 trx
                  ||+------- | IRQL0
                  |+-------- | IRQL1
                  +--------- | IRQ2}
        ax0 = dm (i1, m1);          { start interrupt }
        tx0 = ax0;
check_init:
        ax0 = dm (stat_flag);       { wait for entire
init }
        af = pass ax0;              { buffer to be sent
to }
        if ne jump check_init;      { the codec
}
        ay0 = 2;
check_aci1:
        ax0 = dm (rx_buf);    { once initialized, wait
for codec }
        ar = ax0 and ay0;     { to come out of
autocalibration }
        if eq jump check_aci1;      { wait for bit set }
check_aci2:
        ax0 = dm (rx_buf);          { wait for bit clear
}
        ar = ax0 and ay0;
        if ne jump check_aci2;
```

```
        idle;
        ay0 = 0xbf3f;                { unmute left DAC }
        ax0 = dm (init_cmds + 6);
        ar = ax0 AND ay0;
        dm (tx_buf) = ar;
        idle;
        ax0 = dm (init_cmds + 7);    { unmute right DAC }
        ar = ax0 AND ay0;
        dm (tx_buf) = ar;
        idle;
     ifc = b#00000011111111;      { clear any pending
interrupt }
        nop;
        imask = b#0001010000;     { enable tx0 & irqe
interrupt }
            {     |||||||||+ | timer
                  ||||||||+- | SPORT1 rec or IRQ0
                  |||||||+-- | SPORT1 trx or IRQ1
                  ||||||+--- | BDMA
                  |||||+---- | IRQE
                  ||||+----- | SPORT0 rec
                  |||+------ | SPORT0 trx
                  ||+------- | IRQL0
                  |+-------- | IRQL1
                  +--------- | IRQ2}
{  wait for interrupt and loop forever }
talkthru:       idle;
     jump talkthru;
{*********************************************************
********
 *  Interrupt service routines
*********************************************************
********
   receive interrupt is not used}
input_samples:
     rti;
{-------------------------------------------------------
--------
 transmit interrupt used for Codec initialization, and
Data generating
 -------------------------------------------------------
--------}
next_tx:
     ena sec_reg;
     ar = dm (stat_flag);
     ar = pass ar;
     if eq jump next_data;
{ init Codec }
     ax0 = dm (i3, m1);           { fetch next control
word and }
```

```
     dm (tx_buf) = ax0;          { place in transmit slot
0    }
     ax0 = i3;
     ay0 = ^init_cmds;
     ar = ax0 - ay0;
     if gt rti;        { rti if more control words still
waiting }
     ax0 = 0xaf00;               { else set done flag and
}
     dm (tx_buf) = ax0;     { remove MCE if done
initialization }
     ax0 = 0;
     dm (stat_flag) = ax0;       { reset status flag }
     rti;
next_data:
     ay0 = dm (time);
     ay1 = dm (time_max);
     ar = ay0+1;
     af = ay1-ar;
     if gt jump calc;
{ change meandr variables }
     ar = dm (i4,m5);
     dm (tx_buf+1) = ar;
     dm (tx_buf+2) = ar;
     ar = dm (i4,m5);
     dm (time_max) = ar;
     ar = 0;             { reset time }
calc:
     dm (time) = ar;
     rti;

{  INTERRUPT button service - indicator FL1 flashing}
irqe:   toggle fl1;
     rti;

.endmod;
```