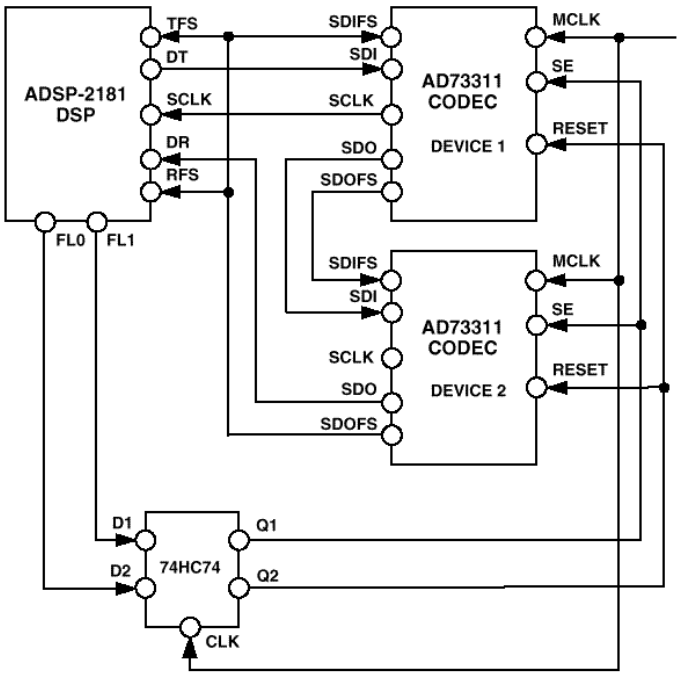
Technical Notes on using Analog Devices' DSP components and development tools
Phone: (800) ANALOG-D, FAX: (781) 461-3010, EMAIL: dsp.support@analog.com, FTP: ftp.analog.com, WEB: www.analog.com/dsp

Interfacing Two AD73311 Codecs to the ADSP-218x

Talkthrough Program for 2 cascaded 73311:
European DSP Apps, E. A.
Tested on SK 2185-Board, 03/98

Hardware Information:
- CODECS are attached to DSP's SPORT 0
- Cascade circuitry: see AD73311 Data Sheet and below
- DSP's internally generated **SCLK1** feeds Master Codec's **MCLK**
- Codecs' reset and SE lines are controlled per Flip-Flop via fl0 and fl1 respectively
- The 2 D-Flip-Flops synchronise Codecs' SE and reset to MCLK
- Programmable flag **PF2** (not on the diagram) of the ADSP-218x is used to clear the 2 Flip-Flop sections (corresponding to Codec reset and SE)



ADSP-218x connected to 2 AD73311 Codecs in cascade

ANALOG DEVICES

```
{*****************************************************************}


.module/RAM/ABS=0 talkthru;
.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                data[2];

                                    {^data -> codec2;
                                    ^data+1 -> codec1}

.var/dm/ram/circ                init_cmds[10];
.var/dm                         stat_flag;


.init init_cmds:

                              {           CR DEV REG   REG    }
                              {           DW ADD ADD   DATA   }
                              {           |+ --- *** ======== }
                              {           |+ --- *** ======== }
        b#1000100100000011,   {reg B,2   b#10 001 001 00000011 }
        b#1000000100000011,   {reg B,1   b#10 000 001 00000011 }
                              { CRB4:6 DMCLK=MCLK; }
                              { CRB2:3 00 -> SCLK = DMCLK/8}


        b#1000101011111001,   {reg C,2   b#10 001 010 11111001 5VEN }
        b#1000001011111001,   {reg C,1   b#10 000 010 11111001 }


        0x8b40,               {reg D,2   b#10 001 011 01000000}
                              {-6dB output gain & 0dB input gain}
```

```
        0x8375,              {reg D,1   b#10 000 011 01110101}
                             {-15 dB outp. gain & 26 dB inp. gain}

        b#1000110000000000,  {reg E,2   b#10 001 100 00000000}
        b#1000010000000000,  {reg E,1   b#10 000 100 00000000}
        b#1000100000010001,  {reg A,2   b#10 001 000 00010001}
        b#1000000000010001;  {reg A,1   b#10 000 000 00010001}
                             { @ reset CRA0=0 -> progr. mode}
                             { CRA1:0 = 01 -> data mode, no mixed-
                  mode}



        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 txcdat; rti; rti; rti;     {10: SPORT0 tx }
        jump input_samples;             {14: SPORT0 rx }
                     rti; rti; rti;
        rti;         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 }


start:

        reset fl1;                  {no SE -> don't enable CODECs'
                                     serial interface yet}

        reset fl0;                  {reset CODECs}



        { Flip-Flops need MCLK first }

        { feed MCLK with SCLK1 = 1/2 CLKOUT }

        ax0 = 0x4000;               { Int SCLK1}
        dm(0x3ff2)=ax0;

        ax0 = 0;
        dm(0x3ff1)=ax0;             { sclk1=0.5 clkout}



        { use SCLK1 out of the FI, FO configuration}
```

```
ax0 = 0x0000;
dm(0x3fff)=ax0;


{low PULSE FOR /CLEAR signal of Flipflops}
{ PROGRAM PF2 AS OUTPUT AND LOW TO CLEAR FF's}

AX0=DM(0X3FE6);
AY0=0X0004;                 {MASK FOR PF2}
AR=AX0 or AY0;              { SET PF2-TYPE TO 1 -> OUTPUT}
DM(0X3FE6)= AR;

AR=DM(0X3FE5);
AR=CLRBIT 2 OF AR;          {MASK FOR PF2 -> 0}
DM(0X3FE5)= AR;

NOP;NOP;NOP;

{ TAKE CLR BACK TO HI}

AR=DM(0X3FE5);
AR=SETBIT 2 OF AR;          {MASK FOR PF2 -> 1}
DM(0X3FE5)= AR;

nop;nop;

i3 = ^init_cmds;
l3 = %init_cmds;
m1 = 1;
i0= ^data; m0=0; l0= %data;
i1= ^data; m1=1; l1= %data;



AX0=b#0000000000000000;         DM(Sport0_Autobuf)=AX0;
AX0=b#0000000000000000;         DM(Sport0_Rfsdiv)=AX0;
AX0=b#0000000000000000;         DM(Sport0_Sclkdiv)=AX0;
AX0=b#0010100000001111;         DM(Sport0_Control_Reg)=AX0;

                            {framed mode, normal}

                            {ext SCLK, ext RFS & TFS, RFS,
                              TFS required, normal framing}
                            {= FS before data, }
                            { TFS=input in Frame Sync Loop-
                              Back Mode}
                            { TFS=output in Nonframe Sync
                              Loop-Back Mode }
```

```
        AX0=b#0001000000000000;
        DM(System_Control_Reg)=AX0;

                                {enable SP0 }

        ifc = b#00000011111111; nop;


        ax0 = 1;
        dm(stat_flag) = ax0;

        imask = b#0001100000;       { enabling tx0 and rx0
                        interr.}


        ax0 = dm (i3, m1);
        tx0 = ax0;                  { sending 1st cmd}
                                    { 1st cmd will also be 1st output
                        from the devices}
                                    { before CODEC comes out of
                        reset, d/s 21}

        set fl0; set fl1;           { relinquish codec reset,
                        fl0}
                                    { enabling SE, fl1}



check_init:
        ax0 = dm (stat_flag);
        af = pass ax0;
        if ne jump check_init;


lop: idle; jump lop;


                                    { receive codecs' Adc DATA
                                    data=codec2
                                    data+1=codec1}
input_samples:
        ax0=rx0;
        {ay0=0xffff;
        ar=ax0 and ay0;}            { means any DSP operation}
        dm(i0,m1)= ax0;

        rti;


txcdat:                             { send cmds to codec 1 and 2}

        ar = dm(stat_flag);
```

```
        ar = pass ar;
        if eq jump tx_dat;          { if init done (all cmds sent to
                        codecs)}

        ax0 = dm (i3, m1);          { next cmd}
        tx0=ax0;
        ax0 = i3;
        ay0 = ^init_cmds;
        ar = ax0 - ay0;             { all cmds sent out ?}
        if gt rti;

                                    { after last cmd: stat_flag :=0}
                                    { after 1st wraparound of
cmds_buffer: i3=^init_cmds}
        ax0 = 0x00;
        dm (stat_flag) = ax0;
        rti;

tx_dat:                             { only after all cmds transmitted
                        to codecs}
                                    { send data to DAC codec1
                                    (^data+1) and codec2(^data)}

          ar=dm(i1,m1);
          tx0=ar;
          rti;

.endmod;
```