Notes on using Analog Devices' DSP, audio, & video components from the Computer Products Division
Phone: (800) ANALOG-D or (781) 461-3881, FAX: (781) 461-3010, EMAIL: dsp.support@analog.com

AD1843/ADSP-2181 Loopback Example

Last Modified: 09/25/96

The following example can be used as a reference for powering up and programming the AD1843 using an ADSP-2100 Family DSP. The user can test the talkthrough example by connecting a microphone to the AD1843's mic input and stereo speakers to the Auxiliary 2 output of the AD1843. In this example, the AD1843 samples incoming audio data and sends the digital information out of it's serial interface to the DSP. The DSP simply captures the data from its receive side of it's SPORT0 interface and transmits the data back out of SPORT0 to the AD1843. The codec then converts the digital stream back to audio data on the AUX2 output pin.

This example uses the Momentum Data Systems Hawk-81 board as the test platform. However, it can be slightly modified to be well suited for other 2100 Family DSP systems.

```
{*****************************************************************************
 *
 *  File:     AUX2OUT.DSP
 *  Version:  2.00
 *  Date:     1-24-96
 *  Author:   ***
 *  Company:  Momentum Data Systems
 *
 *
 *  This sample program is organized into the following sections:
 *
 *  Assemble time constants
 *  Interrupt vector table
 *  ADSP 2181 intialization
 *  ADSP 1843 Codec intialization
 *  Interrupt service routines
 ****************************************************************************}


{   search for 'CHANGING INPUT SOURCE' to change input source }


.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[6];      /* Status + L data + R data */
.var/dm/ram/circ                tx_buf[6];      /* Cmd + L data + R data    */
.var/dm/ram/circ                init_cmds[58];
.var/dm/ram/abs=0x3fdf          currstat;
.var/dm/ram                     adcl, adcr;


.init tx_buf: 0x0080, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000; /* Initially set MCE */

.init init_cmds:
        28,
        0x4800, { 28:   codec config - fundamental settings: R/W
                        b15   : convertor, 0 = normal, 1 = initiate power down
                        b14   : 1 = autocalibration enabled
                        b13   : 0 = clock generator 3 powered down
                        b12   : 0 = clock generator 2 powered down
                        b11   : 0 = clock generator 1 powered down
                        b10   : 0 = CLKOUT pin powered down
                        b09-08: external control: XCTL1 and XCTL0
                        b07   : 0 = CONV3 pin powered down
                        b06   : 0 =  BIT3 pin powered down
                        b05   : 0 = CONV2 pin powered down
                        b04   : 0 =  BIT2 pin powered down
                        b03   : 0 = CONV1 pin powered down
                        b02   : 0 =  BIT1 pin powered down
                        b01   : 0 = right channel single-ended, 1 = differential
                        b00   : 0 = left channel single-ended, 1 = differential
                    }
        27,
        0x13d3, { 27:   codec config - power down: R/W
                        b15   : 0, see manual
                        b12   : DAC 2 to DAC 1 mix, 0=powered down, 1=enabled
                        b09   : DAC 2, 0 = powered down, 1 = enabled
                        b08   : DAC 1, 0 = powered down, 1 = enabled
                        b07   : all conv analog, 0 = powered down, 1 = enabled
                        b06   : headphone, 0 = powered down, 1 = enabled
                        b04   : analog in to out, 0 = powered down, 1 = enabled
                        b01   : ADC right, 0 = powered down, 1 = enabled
                        b00   : ADC left, 0 = powered down, 1 = enabled
                    }


        26,
        0x0505, { 26:   codec config - serial interface: R/W
                        b15   : 1 = DAC 2 is mono
                        b14   : 1 = DAC 1 is mono
                        b11-10: DAC 2 data format select
                                0 = 8-bit linear
                                1 = 16-bit linear
                                2 = 8-bit u-Law
                                3 = 8-bit A-Law
                        b09-08: DAC 1 data format select
                                0 = 8-bit linear
                                1 = 16-bit linear
                                2 = 8-bit u-Law
                                3 = 8-bit A-Law
                        b07   : SCLK frequency, 0=12.288 MHz, 1=16.384 MHz
                        b06   : frame size, 0=32 slots, 1=16 slots
                        b05   : frame size change timing, 0=complete frame,1=now
                        b03-02: ADC right channel data format select
                                0 = 8-bit linear
                                1 = 16-bit linear
                                2 = 8-bit u-Law
                                3 = 8-bit A-Law
                        b01-00: ADC left channel data format select
                                0 = 8-bit linear
                                1 = 16-bit linear
                                2 = 8-bit u-Law
                                3 = 8-bit A-Law
                    }
         0,
        0x0000, { 00:   Codec status and revision id: R
                        b15   : 1 if internal clock not yet stabilized.
```

```
                    b14   : 1 if pwoered down.
                    b03-00: revision id
                }
 1,
0x0000, { 01:   channel status flag: R, W to clear
                    b09   : DAC 2 underrun
                    b08   : DAC 1 underrun
                    b03-02: ADC right overrange:
                            0 = > -1 dB under
                            1 = >  0 dB under
                            2 = >  0 dB over
                            3 = >  1 dB over
                    b01-00: ADC left overrange:
                            0 = > -1 dB under
                            1 = >  0 dB under
                            2 = >  0 dB over
                            3 = >  1 dB over
                }
 2,
{ CHANGING INPUT SOURCE }
{ 0x4040 Aux 1 input, jack closest to gold fingers }
{ 0x3f3f Mic input, the jack next to Aux 1; dynamic mic required }
0x4040, { 02:   input control - ADC source and gain/attn: R/W
                    b15-13: left ADC source
                            0 = line
                            1 = mic
                            2 = aux 1
                            3 = aux 2
                            4 = aux 3
                            5 = mono
                            6 = DAC 1
                            7 = DAC 2
                    b12   : mic gain, 0 = 0 dB, 1 = 20 dB
                    b11-08: ADC gain, 0 = 0 dB, 15 = 22.5 dB
                    b07-05: right ADC source
                            0 = line
                            1 = mic
                            2 = aux 1
                            3 = aux 2
                            4 = aux 3
                            5 = mono
                            6 = DAC 1
                            7 = DAC 2
                    b04   : mic gain, 0 = 0 dB, 1 = 20 dB
                    b03-00: ADC gain, 0 = 0 dB, 15 = 22.5 dB
                }
 3,
0x8888, { 03:   mix control - DAC 2 to mixer: R/W
                    b15   : left DAC 2 mix mute, 1 = muted
                    b12-08: left DAC 2 mix gain/attn, 0=12, 8=0, 31=-34.5
                    b07   : right DAC 2 mix mute, 1 = muted
                    b04-00: right DAC 2 mix gain/attn, 0=12, 8=0, 31=-34.5
                }
 4,
0x8888, { 04:   mix control - Aux 1 to mixer: R/W
                    b15   : left aux 1 mix mute, 1 = muted
                    b12-08: left aux 1 mix gain/attn, 0=12, 8=0, 31=-34.5
                    b07   : right aux 1 mix mute, 1 = muted
                    b04-00: right aux 1 mix gain/attn, 0=12, 8=0, 31=-34.5
                }
 5,
0x8888, { 05:   mix control - Aux 2 to mixer: R/W
                    b15   : left aux 2 mix mute, 1 = muted
                    b12-08: left aux 2 mix gain/attn, 0=12, 8=0, 31=-34.5
                    b07   : right aux 2 mix mute, 1 = muted
                    b04-00: right aux 2 mix gain/attn, 0=12, 8=0, 31=-34.5
                }
 6,
0x8888, { 06:   mix control - Aux 3 to mixer: R/W
                    b15   : left aux 3 mix mute, 1 = muted
                    b12-08: left aux 3 mix gain/attn, 0=12, 8=0, 31=-34.5
                    b07   : right aux 3 mix mute, 1 = muted
                    b04-00: right aux 3 mix gain/attn, 0=12, 8=0, 31=-34.5
                }
 7,
0x8888, { 07:   mix control - mic to mixer: R/W
                    b15   : left mic mix mute, 1 = muted
                    b12-08: left mic mix gain/attn, 0=12, 8=0, 31=-34.5
                    b07   : right mic mix mute, 1 = muted
                    b04-00: right mic mix gain/attn, 0=12, 8=0, 31=-34.5
```


Notes on using Analog Devices' DSP, audio, & video components from the Computer Products Division
Phone: (800) ANALOG-D or (781) 461-3881, FAX: (781) 461-3010, EMAIL: dsp.support@analog.com

```
            }
 8,
0x8818, { 08:   mix/misc control - mono to mixer & misc: R/W
                b15   : mono mix mute, 1 = muted
                b12-08: mono mix gain/attn, 0=12, 8=0, 31=-34.5
                b07   : 1 = all mix to DAC 1 muted
                b06   : 1 = mono output muted
                b05   : 1 = headphone left and right muted
                b04   : headphone output swing, 0 = 2 Vpp, 1 = 4 Vpp
                b03   : 1 = sum left and right to DAC 1 muted
                b01   : 0 = DAC 2 gain change on 0 cross, 1=immediately
                b00   : 0 = DAC 1 gain change on 0 cross, 1=immediately
            }
 9,
0x0808, { 09:   output control - DAC 1 analog gain/attn: R/W
                b15   : 1 = left DAC 1 muted
                b13-08: left DAC 1 gain/attn, 0=+12, 8=0, 63=-82.5
                b07   : 1 = right DAC 1 muted
                b05-00: right DAC 1 gain/attn, 0=+12, 8=0, 63=-82.5
            }
10,
0x0808, { 10:   output control - DAC 2 analog gain/attn: R/W
                b15   : 1 = left DAC 2 muted
                b13-08: left DAC 2 gain/attn, 0=+12, 8=0, 63=-82.5
                b07   : 1 = right DAC 2 muted
                b05-00: right DAC 2 gain/attn, 0=+12, 8=0, 63=-82.5
            }
11,
0x0000, { 11:   output control - DAC 1 digital attn: R/W
                b15   : 1 = left DAC 1 muted
                b13-08: left DAC 1 digital attn, 0=0, 63=-94.5
                b07   : 1 = right DAC 1 muted
                b05-00: right DAC 1 digital attn, 0=0, 63=-94.5
            }
12,
0x0000, { 12:   output control - DAC 2 digital attn: R/W
                b15   : 1 = left DAC 2 muted
                b13-08: left DAC 2 digital attn, 0=0, 63=-94.5
                b07   : 1 = right DAC 2 muted
                b05-00: right DAC 2 digital attn, 0=0, 63=-94.5
            }
13,
0x8888, { 13:   digital mix control - ADC to DAC 1: R/W
                b15   : 1 = left ADC out to left DAC 1 in muted
                b13-08: attn for above, 0=0, 63=-94.5 dB
                b07   : 1 = right ADC out to left DAC 1 in muted
                b13-08: attn for above, 0=0, 63=-94.5 dB
            }
14,
0x8888, { 14:   digital mix control - ADC to DAC 2: R/W
                b15   : 1 = left ADC out to left DAC 2 in muted
                b13-08: attn for above, 0=0, 63=-94.5 dB
                b07   : 1 = right ADC out to left DAC 2 in muted
                b13-08: attn for above, 0=0, 63=-94.5 dB
            }
15,
0x0505, { 15:   codec config - channel sample rate source select: R/W
                b11-10: DAC 2 sample rate
                        0 = 48 kHz
                        1 = clock generator 1
                        2 = clock generator 2
                        3 = clock generator 3
                b09-08: DAC 1 sample rate
                        0 = 48 kHz
                        1 = clock generator 1
                        2 = clock generator 2
                        3 = clock generator 3
                b03-02: ADC right sample rate
                        0 = 48 kHz
                        1 = clock generator 1
                        2 = clock generator 2
                        3 = clock generator 3
                b01-00: ADC left sample rate
                        0 = 48 kHz
                        1 = clock generator 1
                        2 = clock generator 2
                        3 = clock generator 3
            }
16,
0x0000, { 16:   clock generator 1 control - mode: R/W
```

```
                        b15   : 0 = referenced to XTALI
                                    CRA17 and C1X8/7 define sample rate
                                    CRA18 shifts phase
                                    C1M7:0 and C1P200 define bit rate
                                1 = referenced to SYNC1
                                    C1VID and C1M7:0 define sample rate
                                    phase is lock to SYNC1
                                    C1M7:0 and C1P200 define bit rate unless
                                        Video Lock Mode (no bit clock)
                        b14   : C1VID, not used if using XTALI; see manual
                        b13   : C1PLLG, not used if using XTALI; see manual
                        b12   : C1P200, 1 = bit clk == C1M7:0 + 200, see manual
                        b11   : C1X8/7, 1 = conv clk == 8/7 * CRA17, see manual
                        b10   : C1C128, CONV1 rate, 1 = x 128, 0 = x 1, manual
                        b07-04: C1M7:0, bit clock rate, see manual
                    }
        17,
        8000,   { 17:   clock generator 1 control - sample rate: R/W
                        b15-00: when not ref SYNC1, sample rate in Hz, == 48 kHz
                    }
        18,
        0x0000, { 18:   clock generator 1 control - phase shift: R/W
                        b08   : 1 = phase retarded, 0 = advanced
                        b07-00: phase shift magnitude, see manual
                    }
        19,
        0x0000, { 19:   clock generator 2 control - mode: R/W
                        see clock generator 1
                    }
        20,
        0xbb80, { 20:   clock generator 2 control - sample rate: R/W
                        see clock generator 1
                    }
        21,
        0x0000, { 21:   clock generator 2 control - phase shift: R/W
                        see clock generator 1
                    }
        22,
        0x0000, { 22:   clock generator 3 control - mode: R/W
                        see clock generator 1
                    }
        23,
        0xbb80, { 23:   clock generator 3 control - sample rate: R/W
                        see clock generator 1
                    }
        24,
        0x0000, { 24:   clock generator 3 control - phase shift: R/W
                        see clock generator 1
                    }
        25,
        0x4000; { 25:   codec config - digital filter and mode select: R/W
                        b15   : 1 = digital resampler mode, see manual
                        b14   : 1 = DAC digital mix enabled
                        b09   : 0, see manual
                        b08   : 0, see manual
                        b07-06: digital ADC right channel source select
                                0 = ADC result
                                1 = res
                                2 = digital DAC 1 (digital resampler)
                                3 = digital DAC 2 (digital resampler)
                        b05-04: digital ADC left channel source select
                                0 = ADC result
                                1 = res
                                2 = digital DAC 1 (digital resampler)
                                3 = digital DAC 2 (digital resampler)
                        b01   : 0, see manual
                        b00   : 0, see manual
                    }

{*****************************************************************************
 *
 *  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 }
        rti;         rti; rti; rti;     {10: SPORT0 tx }
        rti;         rti; rti; rti;     {14: SPORT1 rx }
```


Notes on using Analog Devices' DSP, audio, & video components from the Computer Products Division
Phone: (800) ANALOG-D or (781) 461-3881, FAX: (781) 461-3010, EMAIL: dsp.support@analog.com

```
        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 }


{*****************************************************************************
 *
 *  ADSP 2181 intialization
 *
 ****************************************************************************}
start:
{=================== assert /RESET and /PWRDWON on codec =====================}
        io (0x001c) = ax0;
            {   Ths least significant 8 bits address of the last I/O space }
            {   access is latched in accordance to the MAFE spacification. }
            {   MA? mapping for 1843 MAFE : }
            {       a0: RESET/              }
            {       a1: PWRDWN/             }
            {       a2: BM                  }
            {       a3: CS                  }
            {       a4: PDMNFT              }

{-----------------------------------------------------------------------------
    1843 and SPORT auto-buffering interrupt timing:

        these slots are loaded into SPORT buffer
         1        2        3        4        5   X    0
         V        V        V        V        V        V
    tx  (slot 0) (slot 1) (slot 2) (slot 3) (slot 4) (slot 5)
    rx  (slot 0) (slot 1) (slot 2) (slot 3) (slot 4) (slot 5)
                v        v        v        v        v        v
                0        1        2        3        4        5
                these slots are taken out from SPORT buffer


    V : this is the instance where the next word to be transmitted is loaded
        into SPORT buffer and the auto-buffering register is incremented.
    v : this is the instance where the already received word is taken out
        from the SPORT buffer and the auto-buffering register is incremented.

    X : this is where we should write slot 0-5 data so that slot 0 datum will
        be picked up at 0/V.  Data for slot 1-5 may be written any time before
        next frame begins but datum for slot 0 *MUST* be written before the
        transmission of slot 5 datum, which is approximately within 30
        instructions including interrupt latency.  This is so that all data
        will go out in the same slot.

        When using single transmit interrupt to also drive receive routine
        slot 4 and slot 5 data will not be available within approximately
        the first 70 instructions.  This is acceptable for 1843 because
        receive slot 4 and slot 5 are not used.


        This also means that at starting time receive address pointer should
        point to slot 0 location but transmit address pointer should point
        to slot 1 location.

 ----------------------------------------------------------------------------}


        i0 = ^rx_buf;
        l0 = %rx_buf;
        i1 = ^tx_buf + 1;
        l1 = %tx_buf;

        i4 = ^init_cmds;
        m4 = 1;
        l4 = 0;


        m1 = 1;


        ena sec_reg;                { use shadow register bank }
        ax1 = 0x0000;
        ay1 = 0x001c;
        dis sec_reg;                { use shadow register bank }
```

```
{================== 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=right 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#0000000000111111;   dm (SPORT0_TX_Channels0) = ax0;
            {   ^15          00^   transmit word enables: channel # == bit # }
        ax0 = b#0000000000111111;   dm (SPORT0_TX_Channels1) = ax0;
            {   ^31          16^   transmit word enables: channel # == bit # }
        ax0 = b#0000000000111111;   dm (SPORT0_RX_Channels0) = ax0;
            {   ^15          00^   receive word enables: channel # == bit # }
        ax0 = b#0000000000111111;   dm (SPORT0_RX_Channels1) = ax0;
            {   ^31          16^   receive word enables: channel # == bit # }


{================== S E R I A L   P O R T   #1   S T U F F ==================}
        ax0=0;      dm(SPORT1_Autobuf)=ax0;     { autobuffering disabled }
        ax0=0;      dm(SPORT1_RFSDIV)=ax0;      { RFSDIV not used }
        ax0=0;      dm(SPORT1_SCLKDIV)=ax0;     { SCLKDIV not used }
        ax0=0;      dm(SPORT1_Control_Reg)=ax0; { ctrl reg functions disabled }
```

```
{================ T I M E R   S T U F F ==================}
        ax0=0;      dm(TSCALE)=ax0;             { timer not being used }
        ax0=0;      dm(TCOUNT)=ax0;
        ax0=0;      dm(TPERIOD)=ax0;

{============== 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, 1=FI, FO, IRQ0, IRQ1,..
                |  !+------------ SPORT1 1=enabled, 0=disabled
                |  +============= SPORT0 1=enabled, 0=disabled
                +---------------- 0
                                  0
                                  0
            }


        {-----------------------------------------------------------------}
        {   Reset default value for (PFTYPE) is acceptable.               }
        {   For Hawk 81 PM and DM must be enabled to use external         }
        {   overlay memory; BM must be disabled to use Byte Wide memory.  }
        {-----------------------------------------------------------------}


        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
            }
```


Notes on using Analog Devices' DSP, audio, & video components from the Computer Products Division
Phone: (800) ANALOG-D or (781) 461-3881, FAX: (781) 461-3010, EMAIL: dsp.support@analog.com

```
{*****************************************************************************
 *
 *  ADSP 1843 Codec intialization
 *
 *****************************************************************************}



{================== de-sert /RESET and /PWRDOWN on codec ====================}
        {   enable transmit interrupt }
        imask = b#0001000000;
            {     |||||||||+ | timer
                  ||||||||+- | SPORT1 rec or IRQ0
                  |||||||+-- | SPORT1 trx or IRQ1
                  ||||||+--- | BDMA
                  |||||+---- | IRQE
                  ||||+----- | SPORT0 rec
                  |||+------ | SPORT0 trx
                  ||+------- | IRQL0
                  |+-------- | IRQL1
                  +--------- | IRQ2
            }

        io (0x001f) = ax0;
            {   MA? mapping for 1843 MAFE : }
            {       a0: RESET/              }
            {       a1: PWRDWN/             }
            {       a2: BM                  }
            {       a3: CS                  }
            {       a4: PDMNFT              }


        {   by de-asserting /RESET and /PWRDOWN will casue receive to start
            automatically. }



        ay0 = 0x8000;
{-- waiting for INIT to go 0 ------------------------------------------------}
wt_init1:
        idle;
        ax0 = dm (rx_buf + 1);
        none = ax0 and ay0;
        if ne jump wt_init1;


{-- power up conversion resources -------------------------------------------}
        ax0 = dm (i4, m4);      dm (tx_buf + 0) = ax0;
        ax0 = dm (i4, m4);      dm (tx_buf + 1) = ax0;
        idle;


{-- wait for resources standby ready ----------------------------------------}
        ax0 =        0x0080 + 00;   dm (tx_buf + 0) = ax0;
        ay0 = 0x4000;
{-- waiting for PDNO to go 0 ------------------------------------------------}
wt_init2:
        idle;
        ax0 = dm (rx_buf + 1);
        none = ax0 and ay0;
        if ne jump wt_init2;


        cntr = 26;
        do initl until ce;
        ax0 = dm (i4, m4);      dm (tx_buf + 0) = ax0;
        ax0 = dm (i4, m4);      dm (tx_buf + 1) = ax0;
        idle;
initl:  nop;


        {   install ISR }
        i4 = ^isrVec;
        ar = pm (i4, m4);
        i4 = 0x10;
        pm (i4, m4) = ar;
```

```
        {   set 0x0100 for phone off hook, clear for on hook. }
        ax0 = 0x4900;           dm (tx_buf + 1) = ax0;

{-------------------------------------------------------------------------------
 -
 -  wait for interrupt and loop forever
 -
 -------------------------------------------------------------------------------}


talkthru:
        idle;
        jump talkthru;


isrVec:
        jump output_samples;            {10: SPORT0 tx }


{*******************************************************************************
 *
 *  Interrupt service routines
 *
 *******************************************************************************}

{-------------------------------------------------------------------------------
 -
 -  receive interrupt used for loopback
 -
 -  960123 modified with input from Al Clark
 -------------------------------------------------------------------------------}
output_samples:
        ena sec_reg;                { use shadow register bank }

        {   assume nothing for DAC }
        dm (tx_buf + 0) = ay1;      { ay1 holds default command }

        {   anything from ADC for us? }
        ax0 = dm (rx_buf + 0);      { get status word from codec }

tstLt:
        ar = tstbit 8 of ax0;       { Check for Left A/D available }
        if eq jump tstRt;

        ar = setbit 8 of ax1;
        ax1 = ar;
        ar = dm (rx_buf + 2);       { Fetch Data }
        dm (adcl) = ar;

tstRt:
        ar = tstbit 9 of ax0;       { Check for Right A/D available }
        if eq jump tst_DAC;

        ar = setbit 9 of ax1;
        ax1 = ar;
        ar = dm (rx_buf + 3);
        dm (adcr) = ar;

tst_DAC:{ loopback }                { Transmit data by syncing to Right Channel }

        ay0 = 0x0300;
        ar = ax1 xor ay0;           { are both right & left available }
        if ne rti;

        ax1 = 0;                    { Clear one shot }
        ax0 = 0x0300;
        ar = ax0 or ay1;
        dm (tx_buf + 0) = ar;

        ar = dm (adcl);
        dm (tx_buf + 2) = ar;
        dm (tx_buf + 4) = ar;

        ar = dm (adcr);
        dm (tx_buf + 3) = ar;
        dm (tx_buf + 5) = ar;

        rti;
```

```
.endmod;
```