TEXAS INSTRUMENTS

TAS3001 EVM

User's Guide

February 2001

Digital Speaker Products

SLAU060

IMPORTANT NOTICE

Texas Instruments and its subsidiaries (TI) reserve the right to make changes to their products or to discontinue any product or service without notice, and advise customers to obtain the latest version of relevant information to verify, before placing orders, that information being relied on is current and complete. All products are sold subject to the terms and conditions of sale supplied at the time of order acknowledgment, including those pertaining to warranty, patent infringement, and limitation of liability.

TI warrants performance of its products to the specifications applicable at the time of sale in accordance with TI's standard warranty. Testing and other quality control techniques are utilized to the extent TI deems necessary to support this warranty. Specific testing of all parameters of each device is not necessarily performed, except those mandated by government requirements.

Customers are responsible for their applications using TI components.

In order to minimize risks associated with the customer's applications, adequate design and operating safeguards must be provided by the customer to minimize inherent or procedural hazards.

TI assumes no liability for applications assistance or customer product design. TI does not warrant or represent that any license, either express or implied, is granted under any patent right, copyright, mask work right, or other intellectual property right of TI covering or relating to any combination, machine, or process in which such products or services might be or are used. TI's publication of information regarding any third party's products or services does not constitute TI's approval, license, warranty or endorsement thereof.

Reproduction of information in TI data books or data sheets is permissible only if reproduction is without alteration and is accompanied by all associated warranties, conditions, limitations and notices. Representation or reproduction of this information with alteration voids all warranties provided for an associated TI product or service, is an unfair and deceptive business practice, and TI is not responsible nor liable for any such use.

Resale of TI's products or services with *statements different from or beyond the parameters* stated by TI for that product or service voids all express and any implied warranties for the associated TI product or service, is an unfair and deceptive business practice, and TI is not responsible nor liable for any such use.

Also see: Standard Terms and Conditions of Sale for Semiconductor Products. www.ti.com/sc/docs/stdterms.htm

Mailing Address:

Texas Instruments
Post Office Box 655303
Dallas, Texas 75265

Preface

Read This First

About This Manual

This user's guide describes the operation of the TAS3001 EVM. The user's guide contains descriptions and schematics for a stereo application.

How to Use This Manual

This document contains the following chapters:

- Chapter 1 Overview
- Chapter 2 System Components
- Chapter 3 Board Operation
- Chapter 4 TAS3001 EVM Block Diagram
- Chapter 5 MSP430 Microcode Example
- Appendix A Schematics

Notational Conventions

This document uses the following conventions.

- Program listings, program examples, and interactive displays are shown in a `special typeface` similar to a typewriter's.

  Here is a sample program listing:

```
RESET   MOV     #Stack,SP       ;initialize stack pointer
        CALL    #Setup          ;call device setup routine
        CALL    #Set_DCO        ;sw calibration of DCO
```

Related Documentation From Texas Instruments

- TLC320AD77 Literature Number SLAS194
- TAS3001 Literature Number SLAS226
- MSP430E112 Literature Number SLAS219
- TPS7233 Literature Number SLVS102F
- TLV2362PWR Literature Number SLOS195B

FCC Warning

This equipment is intended for use in a laboratory test environment only. It generates, uses, and can radiate radio frequency energy and has not been tested for compliance with the limits of computing devices pursuant to subpart J of part 15 of FCC rules, which are designed to provide reasonable protection against radio frequency interference. Operation of this equipment in other environments may cause interference with radio communications, in which case the user at his own expense will be required to take whatever measures may be required to correct this interference.

Contents

Figures

Tables

Chapter 1

Overview

The TAS3001 EVM board demonstrates the operation of the digital equalization and the dramatic improvements that 32-bit digital audio signal processing can make on sound quality.

Topic **Page**

1.1 Description

This user's guide describes the operation of the TAS3001 EVM. The user's guide contains descriptions and schematics for a stereo application. The board described is an example design that can be customized for specific applications.

An analog stereo input signal is provided through a 3.5-mm stereo jack. The left and right channels of the signal are filtered and converted into left and right digital signals via the TLC320AD77's analog-to-digital converter. The digital signals are then equalized by the TAS3001 and converted back to analog via the TLC320AD77's digital-to-analog converter. The analog left and right signals are amplified at the output stage and piped to the line output to drive amplified speakers via a 3.5-mm stereo jack. The output stage is capable of driving internally-amplified speakers or a stereo amplifier's line input.

Additionally, the speaker equalization board contains a two-channel, 24-watt-per-channel audio power amplifier. The output of the power amplifier is differential; therefore, there are no capacitors in its output circuit that could limit its low frequency response. The output of the power amplifier is connected to RCA connectors. Since the amplifier has a differential output, both terminals of the connection contain the amplified signal. Ensure that neither of these contacts is connected to ground. The power amplifier also contains an LED, labeled CLIP in the silkscreen on the PCB. The LED illuminates when the maximum output power is exceeded or when one of the outputs is shorted to ground.

Six switches are used to program the digital equalization of the TAS3001 via the MSP430 microcontroller. The microcontroller provided with this board is preprogrammed for five different EQ settings that can be selected via different combinations of the switches. The switches and EQ settings are described in Section 3.2. This microcontroller, in conjunction with the six switches, can also adjust the volume, bass, and treble settings of the TAS3001. The settings for changing the volume, bass, and treble levels are described in Section 5.2.

The MSP430 is placed in a socket so that it can be replaced with a 50-Ω resistor between pins 7 and 13 to reset the board. The board can then be controlled through the DB25 connector. A 25-pin DB25 (male at both ends) is then connected between the EVM and a PC. The cable must have all 25 wires connected (not an RS232 extender cable) and the parallel port on the PC must be set to the EPP mode. If the EPP mode is not available, the bidirectional or PS-2 mode will work in most cases. The TAS3001 EVM is shipped with the 50-Ω resistor installed. If stand-alone operation is desired, an MSP430 microcontroller can be requested from the digital speaker group at Texas Instruments (214–480–3362).

This system receives and transmits digital data at a 48-kHz sampling frequency.

1.2 Features

The speaker equalization board has the following features.

- 12–V to 18-V-V dc power supply operation
- Onboard 3.3-V regulator
- TAS3001 stereo audio digital equalizer
- Operation via either a preprogrammed MSP430 microcontroller or via a PC through a DB25 male/male cable and software program
- Six switches used to control volume, bass, treble, and equalization
- Mute control
- LED indicates mode of operation
- A 12.288-MHz master clock frequency
- 1-Vrms line input and output analog stereo signals sampled at 48 kHz
- 24-watts-per-channel stereo-differential-output audio power amplifier
- Diagnostic indicator (CLIP LED)

1.3 Environmental Working Conditions

The circuit is designed to operate in an office-type environment.

1.4 Description of Inputs

The speaker equalization board uses a 3.5-mm stereo jack for its input. It requires a stereo analog signal input. The analog input ranges from ground to a maximum of 1 Vrms.

1.5 Description of Outputs

1.5.1 Line Output

The line output uses a 3.5-mm stereo jack. The output is a stereo analog audio signal with a 1-Vrms level. The output can drive amplified speakers or the line input of a stereo amplifier.

1.5.2 Power Amplifier Output

The power amplifier output uses two RCA-type connectors. The power amplifier output signal is capable of driving 4-Ω to 8-Ω speakers. The maximum power amplifier output is 24 watts per channel.

1.6 Power Supply

The power supply for the speaker equalization board is 12 V to 18 V (15 V nominal). The board draws an average current of about 0.250 A when driving speakers, and less than 100 mA with no speakers connected. The board can draw greater than 2.5 A peak current when driving speakers at a high volume.

Chapter 2

System Components

This chapter presents an overview of the system components.

Topic **Page**

2.1 Power Supplies and Decoupling

The system uses a TL78M05 linear regulator to convert the 12 V dc input to 5 V dc to power the operational amplifiers, and a TPS7233 LDO to convert the 5-V power to 3.3 V to power the MSP430, the TAS3001, and the TLC320AD77.

All power input pins are decoupled with 0.1 μF capacitors.

2.2 Input Amp and Bias

The input to the TLC320AD77 codec is a differential signal and the input to the speaker equalization board is a single-ended stereo signal. The analog front-end buffers the input signal and provides 1.6 V dc for the ADC. The section input amplifier and bias on the schematic (Figure 4–1) describe the analog-front-end circuit and a single-pole low-pass antialiasing filter. The antialiasing filter attenuates unwanted frequencies out of the range of the ADC.

The analog front end is independently preformed on the left and right channels via a TLV2362PWR dual high-performance, low voltage operational amplifier. The antialiasing filter is created through a combination of resistors and capacitors on the input of the ADC.

2.3 TLC320AD77—ADC/DAC Functions

The speaker equalization board takes an analog stereo audio input signal and converts it to digital so that digital equalization can be performed. The digital signal is then converted back into analog and sent to an external device. The analog-to-digital (A/D) and digital-to-analog (D/A) functions are performed by the TLC320AD77 stereo audio ADA. The TLC320AD77 is a stereo A/D and D/A 24-bit delta-sigma converter. The TLC320AD77 operates at a 48-kHz sampling frequency. The TLC320AD77 serial input option is set to MODE 5, 24-bit I2S operation.

2.4 TAS3001—Digital Equalization

The TAS3001 stereo audio digital equalizer is a 32-bit digital audio signal processor. It provides parametric equalization, bass, treble, and volume control, and dynamic range compression. On the speaker equalization board, the TAS3001 operates in slave mode at a 48-kHz sampling frequency. As with the TLC320AD77, the serial audio format is I2S.

2.5 Clock Generation

A crystal oscillator provides the master clock for the speaker equalization board. The oscillator runs at 12.288 MHz and provides a 50% duty cycle square wave at 3.3 V peak amplitude. The other IIS timing signals are provided by circuitry in the TAS3001.

2.6 System Microcontroller

An EPROM version of the MSP430E112 with 4K bytes of memory is used to control the system because it can be erased and reprogrammed.

Future versions of this board will contain the flash memory version of the MSP430.

Chapter 3

Board Operation

This chapter describes the equalization board operation.

Topic **Page**

3.1 Power-Up Sequence

The following is the power-up sequence for the speaker equalization board:

- ❑ Insert either the MSP430 microcontroller or the *DB25* PC interface cable into the socket provided on the board, verifying the position of pin 1.
- ❑ Connect 15 V dc to the positive terminal of the power supply jack and connect a ground to the negative terminal.
- ❑ Turn on the power supply and verify that the board draws a current of less than 150 mA.
- ❑ Turn off the power.
- ❑ Connect a 1-Vrms analog signal to the 3.5-mm stereo input jack. This input can be provided either through a signal generator or through a CD player or other sound generation equipment (some CD players provide up to 2 Vrms, which causes distortion).
- ❑ Connect the output to amplified stereo speakers via the 3.5-mm stereo output jack. This output can be measured by a signal analysis or played through speakers. Alternately, connect passive speakers to the amplifier's output terminals (RCA connectors).
- ❑ Turn on the power to the speaker equalization board and verify again that it draws less than 150 mA of current after the board is initialized by the microcontroller or by the software GUI.
- ❑ Turn on the power to the input and output devices.
- ❑ Depending on the selected mode of operation—either through the MSP430 microcontroller or through the software GUI program and the *DB25 cable*—introduce equalization to the input analog signal. The instructions for the microcontroller mode of operation are described in section 3.2, and the instructions for the software are described in section 3.3**.**

3.2 Switch Functions for Digital Equalization

When the speaker equalization board is used with an MSP430 microcontroller, the seven onboard switches allow users to adjust the digital equalization, the volume, the base, and the treble of the audio input signal.

The MSP430 microcontroller is preprogrammed with five different equalization effects—original or no EQ, flat EQ, jazz EQ, rock EQ, and voice EQ. Since these equalizations are created for a specific type of speaker, they must be reprogrammed for your speakers. The microcontroller is also preprogrammed to increase or decrease the system volume, bass, and treble by 1 dB. There is also a preprogrammed mode that independently resets the bass and the treble to 0 dB. Users can toggle back and forth between EQ mode and volume/bass/treble mode by alternately depressing SW1 and SW6 simultaneously. The LED is on when the board is in EQ mode and off when the board is in volume/bass/treble mode.

The first switch on the left performs the shift function. It switches the microcontroller between volume/bass/treble and equalization modes. Alternately, the shift function can be performed by pressing switches #1 and #6.

Table 3–1 describes the operation of the switches and the LED.

Table 3–1. Switch Functions for Speaker Equalization

	SW1	SW2	SW3	SW4	SW5	SW6	LED
VOL/BASS/TREBLE MODE							
Volume up: 1 dB	X						Off
Volume down: –1 dB		X					Off
Bass up: 1 dB			X				Off
Bass down: –1 dB				X			Off
Bass reset: 0 dB			X	X			Off
Treble up: 1 dB					X		Off
Treble down: –1 dB						X	Off
Treble reset: 0 dB					X	X	Off
SHIFT	X					X	On
EQ MODE							
Mute	X						On
Original equalization		X					On
Flat equalization			X				On
Jazz equalization				X			On
Rock equalization					X		On
Voice equalization						X	On

3.3 Software GUI

When the MSP430 microcontroller is replaced by the *DB25 cable*, the programming of the TAS3001 equalization can now be done through the software GUI program via a PC. In the software GUI operation mode, the six onboard switches are disabled and the six bands of digital equalization, volume control, bass, and treble are programmed via the software. Refer to the software GUI user's manual for instructions on the use of this feature.

3.4 Software Installation

1) Set the parallel port on the PC to the EPP mode. (If the EPP mode is not available, bidirectional or PS-2 mode works in most cases.)

2) Create a directory on the PC and install the contents of the supplied disk in it.

3) Run the .exe file and follow the steps on the screen.

Chapter 4

TAS3001 EVM Block Diagram

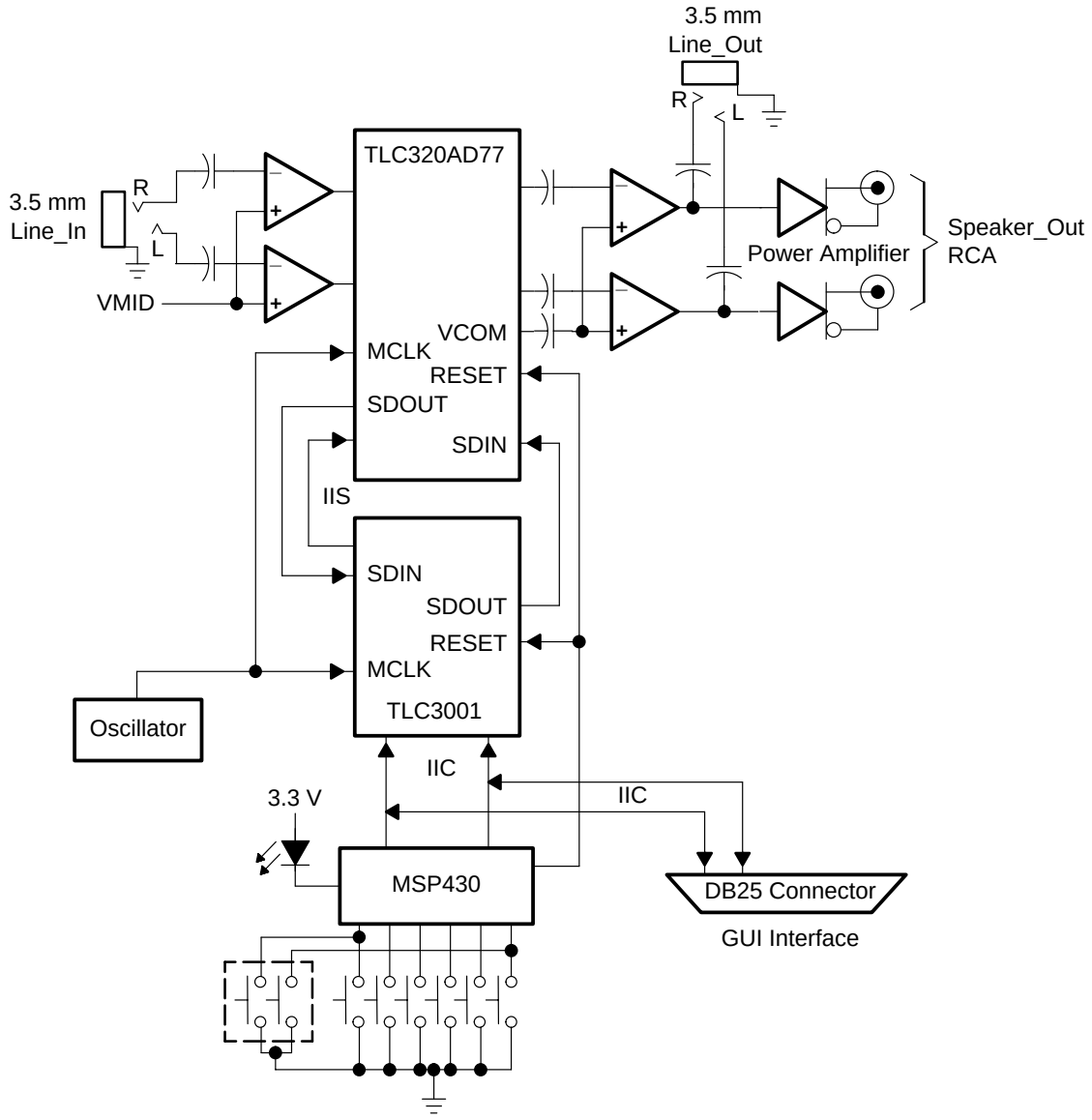
This chapter presents the TAS3001 EVM block diagram and the printed-circuit board (PCB) silk screen.

Topic **Page**

4.1 TAS3001 EVM Block Diagram

Figure 4–1 shows the TAS3001 block diagram.

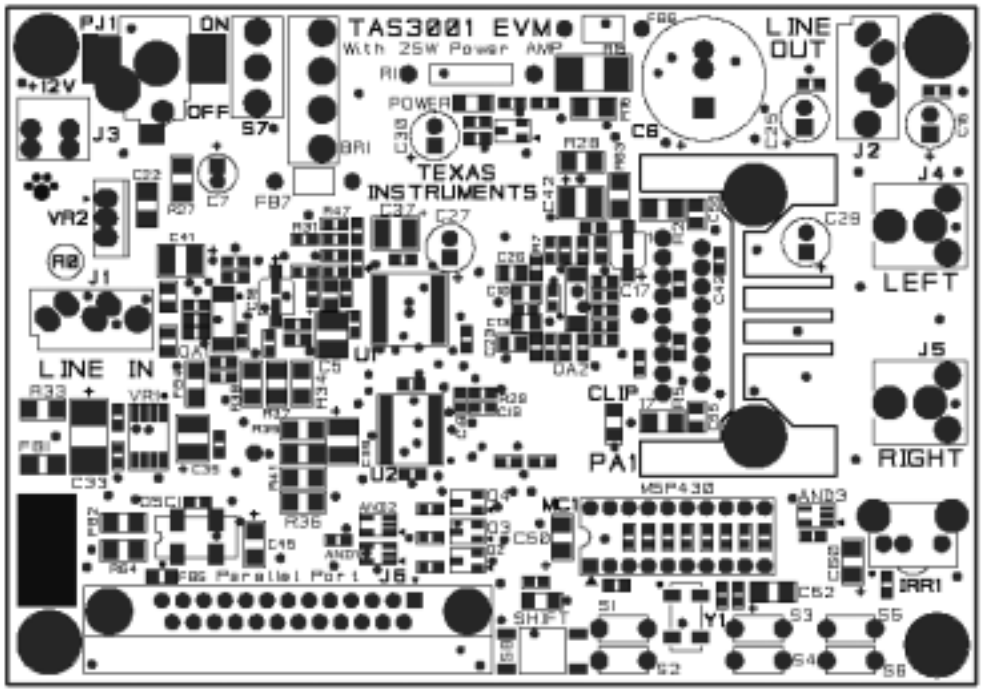
Figure 4–1. TAS3001 EVM Block Diagram



4.2 PCB Silkscreen

Figure 4–2 shows the printed-circuit board silkscreen.

Figure 4–2. PCB Silk Screen



Top Silkscreen

Chapter 5

MSP430 Microcode Example

```
_CPU_   .set  5         ; ID11X
    .copy "STD_DEF.ASM" ; link standard definitions file
;*******************************************************************************
;* MSP430 to Digital TAS3001 EVM board         **
;*                                             **
;* Brian Merritt                                     **
;* MSP430 Applications                               **
;* rev 2.4G 10/99                                    **
;*                                             **
;*******************************************************************************
; Purpose of program to initialize Digital Speaker part via I2C; perform
; start up of the equalizer board; and act as interface between multifunction
; key switches and Digital Speaker part. 430 does not have to read data from
; DS part (other than acknowledge) so read routines deleted to save memory
; space. Arbitration also deleted since 430 is the only part transmitting
; data.
;
;  rev 2.4G - unchanged delay changes in rev F back to rev C, including
;  additional delay0 added in F, after finding typo in rev F.
;
;*******************************************************************************
;  Definitions
;*******************************************************************************

;*********************   11x standard locations
Stack     .EQU  00210h       ;stackpointer
I_vectors .EQU  0FFFFh       ;Interrupt vectors in ROM
Main      .EQU  0F000h       ;ROM start on x112
;*********************   I2C pins and I/O registers
SCL       .EQU  010h         ;P1.4 functions as SCL line (pull-up required)
SDA       .EQU  020h         ;P1.5 functions as SDA line (pull-up required)
SCLIN     .EQU  020h         ;clock in = P1 input register (P1IN)
SDAIN     .EQU  020h         ;data in = P1 input register (P1IN)
SCLDAT    .EQU  021h         ;clock out = P1 output register (P1OUT)
SDADAT    .EQU  021h         ;data out = P1 output register (P1OUT)
SCLEN     .EQU  022h         ;clock enable = P1 direction register (P1DIR)
SDAEN     .EQU  022h         ;data enable = P1 direction register (P1DIR)
SCLFUNC   .EQU  026h         ;fuction = P1 function select register (P1SEL)
SDAFUNC   .EQU  026h         ;fuction = P1 function select register (P1SEL)
```

```
;********************** I2C register
Tbl_Pntr     .EQU  R4    ;pointer for position in total data table
Strng_Pntr   .EQU  R5    ;pointer for position in current byte string
Mask         .EQU  R6    ;used to test level of bits in data byte
;******************** DCO calibration definitions
Count1       .set  R7    ;Counter for SW DCO calibration
Count2       .set  R8    ;Counter for SW DCO calibration
Delta        .set  150   ;SMCLK = 1,228,800 Hz
;******************** Speaker Equal. board definitions
OSC_OE       .set  080h  ;P1.7 functions as oscill enable (active high)
RST          .set  001h  ;P1.0 functions as MIC board reset (active low).
                         ;"RESET" already used elsewhere in this program.
EXP          .set  008h  ;P2.3 functions as expand (active high)
;*********************   Key switch/LED/ pins & registers
SW1          .equ  08h   ;P1.3 equates to switch 1 (active low)
SW2          .equ  04h   ;P1.2       "  "   2       "
SW3          .equ  02h   ;P1.1    "  "      3       "
SW4          .equ  01h   ;P2.0    "  "      4       "
SW5          .equ  02h   ;P2.1    "  "      5       "
SW6          .equ  04h   ;P2.2    "  "      6       "
SW1_3IF      .equ  023h  ;SW1-3 int flags = P1 int flag
                         ;register (P1IFG)
SW4_6IF      .equ  02Bh  ;SW4-6 int flags = P2 int flag register (P2IFG)
SW1_3IE      .equ  025h  ;SW1-3 int enables = P1 int enable register (P1IE)
SW4_6IE      .equ  02Dh  ;SW4-6 int enables = P2 int enable register (P1IE)
SW1_3IN      .equ  020h  ;SW1-3 inputs = P1 input register (P1IN)
SW4_6IN      .equ  028h  ;SW4-6 inputs = P2 input register (P1IN)
LED          .equ  020h  ;P2.5 equates to LED pin (active low)
LED_OUT      .equ  029h  ;LED data out = P2 output register (P2OUT)
Vol_Pntr     .equ  R9    ;pointer for volume setting = data table position
Bass_Pntr    .equ  R10   ;pointer for bass setting = data table position
Trbl_Pntr    .equ  R11   ;pointer for treble setting = data table position
MUTE         .equ  R12   ;current mute setting. 0=mute off, 1=mute on.
Trbl_Setpnt  .equ  R13   ;used to track last manual setting of treble
                         ;needed since treble reset to original with 2 EQs
Bass_Setpnt  .equ  R14   ;used to track last manual setting of bass

                         ;needed since bass reset to original with 2 EQs
;***************************************************************************
;  Main Program
;***************************************************************************
             .sect "MAIN",Main
RESET  MOV   #Stack,SP       ;initialize stackpointer
             CALL  #Setup    ;call device setup routine
       CALL  #Set_DCO        ;sw calibration of DCO

Begin  MOV   #1669,Tbl_Pntr  ;set pointer to start of set up data
       CALL  #Loop           ;call routine to send I2C
Finish       BIC.b #EXP,&P2OUT ;set shutdown pin low after I2C code sent
       EINT                  ;interrupts enabled after setup complete
LPM    BIS   #LPM3,SR        ;go into low pwr mode 3 after DS part is set up
       JMP   LPM             ;back to low power mode after interrupt serviced
```

```
;*******************************************************************************
;  Set Up 11x
;*******************************************************************************
Setup  MOV    #WDTPW+WDTHOLD,&WDTCTL      ;stop watchdog timer
   MOV      #TASSEL1+CLR,&TACTL         ;clock source for 16-bit
                                        ;timer of Timer_A = SMCLK,
   MOV.b    #XTOFF+DIVA1+RSEL2+RSEL0,&BCSCTL1  ;ACLK/4 for DCO cal,
                                               ;current into DCO
                                               ; (Rsel=6)

   BIC.b    #0FFh,&P1SEL    ;set P1 function = I/O
   BIC.b    #0FFh,&P2SEL    ;set P2 function = I/O
   BIC.b    #0FEh,&P1OUT    ;set P1 output data = low, except reset
   BIS.b    #0C1h,&P1DIR    ;set P1 pins-I2C & switches = inputs
   BIS.b    #RST,&P1OUT     ;set RESET data = high
   BIS.b    #OSC_OE,&P1OUT  ;set oscill enable data = high
   BIC.b    #0FFh,&P2OUT    ;set P2 output data = low
   ;setting for Expand = 0 = off
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
   CALL  #DELAY2            ;delay to let oscill stabilize
   BIC.b #RST,&P1OUT        ;set RESET data = low. pin goes
                            ;low since it is enabled
   CALL  #NOP9              ;delay RESET low to high
   BIS.b #RST,&P1OUT        ;set RESET pin high
;*********************   Key switch pins & registers
   BIC.b #SW1+SW2+SW3,&P1DIR   ;switch pins on P1 set to inputs
   BIC.b #SW4+SW5+SW6,&P2DIR   ;switch pins on P2 set to inputs
   BIS.b #SW1+SW2+SW3,&P1IES   ;SW1-3 interrupt edge select = HI to LO
   BIS.b #SW4+SW5+SW6,&P2IES   ;SW4-6 interrupt edge select = HI to LO
   BIS.b #SW1+SW2+SW3,&SW1_3IE ;SW1-3 interrupts enabled
   BIS.b #SW4+SW5+SW6,&SW4_6IE ;SW4-6 interrupts enabled
   BIS.b #EXP+LED,&P2DIR       ;set expand & LED pins to outputs
   BIS.b #LED,&LED_OUT         ;LED initial state = off (active low)

   MOV   #180,Vol_Pntr         ;initial setting for volume
   MOV   #415,Bass_Pntr        ;initial setting for bass
   MOV   #415,Bass_Setpnt      ;initial setting for bass
   MOV   #510,Trbl_Pntr        ;initial setting for treble
   MOV   #510,Trbl_Setpnt      ;initial setting for treble
   CLR   MUTE                  ;set mute = 0 = initial state of mute = off
   BIC.b #SW1+SW2+SW3,&SW1_3IF ;clear SW1-3 interrupt flags (P1IFG)
   BIC.b #SW4+SW5+SW6,&SW4_6IF ;clear SW4-6 interrupt flags (P2IFG)
   RET
;*******************************************************************************
;  Delay routine
;*******************************************************************************
DELAY0    push  #03000h         ;SW Delay using Stack
          JMP   DELAY3
DELAY     push  #06000h         ;SW Delay using Stack
          JMP   DELAY3
DELAY2    push  #09000h            ;SW Delay using Stack
DELAY3    dec   0(SP)              ;
```

```
          jnz    DELAY3                ;
          incd      SP                 ;
          RET
;-------------------------------------------------------------------------
;   Set_DCO Subroutine: Set DCO to selected frequency.
;   Count1 = R8 and Count2 = R9 are used and not saved.
;-------------------------------------------------------------------------
Set_DCO
       ;      JMPskipdco
        ;>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
       BIS       #MC1,&TACTL           ;start timer—continous Mode
       CLR       Count1                ;clear counter
       MOV       #CCIS0+CM0+CAP,&CCTL2 ;Define CCR2: capture input signal
                                       ;=ACLK, capture on rising edge,
                                       ;set to capture mode
Test_DCO  BIT    #CCIFG,&CCTL2         ;intrpt flag set? yes=value captured
          JZ     Test_DCO              ;flag not set = loop again
          BIC    #CCIFG,&CCTL2         ;clear interrupt flag
AdjDCO    MOV    &CCR2,Count2          ;contents of cap/com reg2 ->Count2
          SUB    Count1,Count2         ;Count2-Count1 --> Count2
          MOV    &CCR2,Count1          ;contents of cap/com reg2 ->Count1
          CMP    #Delta,Count2         ;Delta=Count2? Delta = SMCLK/(32768/4)
          JL     IncDCO                ;jump if Count2 < Delta
          JEQ    DoneDCO               ;jump if Count2 = Delta
DecDCO            DEC.b &DCOCTL        ;decrease DCO frequency
          JMP    Test_DCO              ;jump to test new setting
IncDCO            INC.b &DCOCTL        ;increase DCO frequency
          JMP    Test_DCO              ;jump to test new setting
DoneDCO           CLR &CCTL2           ;DCO calibrated. stop CCR2
          ;skipdco
;>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
       RET          ;return from setup / cal routine
;-------------------------------------------------------------------------
;  I2C send & test for end of data
;-------------------------------------------------------------------------
Loop   XOR.b #LED,&LED_OUT   ;LED momentarily swithed to indicate I2C
                             ;data being sent
Loop2  CMP.b #0,Data(Tbl_Pntr)  ;first byte in string is no. of data bytes
                             ;to follow before next stop is issued
       JZ LP_END             ;if first byte in string = 0 then end
       MOV.b Data(Tbl_Pntr),Strng_Pntr ;set string pointer = (table pointer)
       CALL  #I2C         ;call routine to send Dig Spkr data byte via I2C
       JMP   Loop2        ;loop to send next data byte string
LP_END    CALL  #DELAY
          XOR.b #LED,&LED_OUT   ;LED switched back to original state = off at
                                ;start up, on in shift (EQ) mode
          RET                   ;return from call
;-------------------------------------------------------------------------
;  I2C subroutines
;-------------------------------------------------------------------------
;******************* send start, set up mask
I2C    BIS.B #SDA,&SDAEN     ;start Condition: set SDA low
       INC   Tbl_Pntr        ;increment to first data byte in string
```

```
      CALL   #NOP9           ;delay before SCL for start low
I2C_Send  MOV.b #80h,Mask    ;bit mask, MSB (bit) first

;******************** test bit, send if low
I2C_Send1 BIT.B  Mask,Data(Tbl_Pntr)   ;bit = 1?
          JC     I2C_Send2             ;jump if bit = 1
          BIS.B  #SCL,&SCLEN           ;data is 0, SCL = low
          CALL   #NOP9                 ;delay low portion of clock
          BIS.B  #SDA,&SDAEN           ;set SDA = low
          BIC.B  #SCL,&SCLEN           ;set SCL (clock) = high
          ;      jmp skip1
;>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
DS_busy           BIT.b #SCL,&SCLIN     ;test if DS part busy (C=0)
          JNC    DS_busy               ;yes, loop again
   ;skip1
;>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
          JMP    I2C_Send3
;******************** send high bit
I2C_Send2 BIS.B  #SCL,&SCLEN           ;data bit = 1: set SCL = low
          CALL   #NOP9                 ;delay low portion of clock
          BIC.B  #SDA,&SDAEN           ;set SDA = high
          CALL   #NOP9                 ;delay
          BIC.B  #SCL,&SCLEN           ;set SCL = high
          ;      jmp skip2
;>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
DS_busy2  BIT.b  #SCL,&SCLIN           ;test if DS part busy (C=0)
          JNC    DS_busy2              ;yes, loop again
       ;skip2
;>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
;******************** shift mask, test for end of byte
I2C_Send3 CLRC                         ;clear carry bit
          RRC.B  Mask                  ;shift 0 into MSB, shift ”1” right
          JNC I2C_Send1                ;”1” shifted into carry?
                                       ; No = jump send1
          CALL   #NOP9                 ;extend SCL pulse before ackn
          ;  JMP  pointers
;>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>

;******************** check acknowledge signal from DS part
I2C_Ackn  BIS.B  #SCL,&SCLEN  ;set SCL = low for Acknowledge Bit
          BIC.B  #SDA,&SDAEN  ;set SDA = high, master releases SDA
          CALL   #NOP8        ;extend SCL low before ackn.
          BIC.B  #SCL,&SCLEN  ;set SCL = high, master releases SCL
Ackn_Test BIT.B  #SDA,&SDAIN  ;test for ackn in carry bit (0=Ackn.)
          JC     Ackn_Test    ;no ackn (C=1), test again for acknowledge
          BIS.b  #SDA,&SDAEN  ;hold SDA line low after Ackn.

;******************** change pointers, check for end of string
   ;pointers
;>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
          INC Tbl_Pntr         ;increment table pointer next byte in table
          DEC Strng_Pntr       ;decrement string pointer
          JNZ I2C_Send         ;if not 0 then send next byte
          ;  jmp tempend
;>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
```

```
          ;  RET
   ;>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
;******************** stop Condition
I2C_Stop  BIS.B #SDA,&SDAEN  ;set SDA = low
          BIS.B #SCL,&SCLEN  ;set SCL = low
          CALL  #NOP9        ;Delay 18 cycles before issuing stop
          CALL  #NOP9
          BIC.B #SCL,&SCLEN  ;set SCL = high
DS_busy3  BIT.b #SCL,&SCLIN  ;test if DS part busy (C=0)
          JNCDS_busy3        ;yes, loop again
          CALL  #NOP9
          BIC.B #SDA,&SDAEN  ;set SDA = high
       ;tempend
;>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
       RET
;******************** delays routines
NOP9      NOP                ; 9 cycles delay
NOP8      RET                ; 8 cycles delay
;***************************************************************************
;  Key switch interrupt service routine
;***************************************************************************
SW_ISR             ;start of interrupt service routine for key switches
          DINT     ;disable interrupts

          CALL  #DELAY2         ;delay for switch debounce
SW1_FUNC                        ;test for SW1+6 pressed=shift (EQ) mode
          BIT.b #SW1,&SW1_3IF   ;SW1 pressed?
          JZ    SW2_EQ          ;NO, jump to next SW
          CALL  #DELAY2         ;delay before testing 2nd SW
          BIT.b #SW6,&SW4_6IF   ;SW6 pressed also?
          JNZ   TOG_LED         ;YES - jump to toggle LED
          BIT.b #LED,&LED_OUT   ;LED on? (=shift mode)
          JZ    SW1_MUTE        ;YES - jump to mute
SW1_FUNC2 BIC.b #LED,&LED_OUT   ;turn LED on
          CALL  #DELAY          ;delay before testing SW again
          BIC.b #SW1,&SW1_3IF    clear SW1 flag
          CMP   #360,Vol_Pntr   ;volume = maximum?
          JEQ   SW1_TEST        ;Yes - SW1_test
          ADD   #10,Vol_Pntr    ;increase pointer to higher setting
          MOV   Vol_Pntr,Tbl_Pntr  ;set table pointer = Vol up data
          CALL  #Loop           ;call I2C send routine
SW1_TEST  BIT.b #SW1,P1IN       ;SW1 still pressed?
          JZ    SW1_FUNC2       ;YES (active low), increase volume
          BIS.b #LED,&LED_OUT   ;turn LED off
          JMP   SW_DONE         ;jump to end of SW routine
TOG_LED   XOR.b #LED,&LED_OUT   ;toggle LED when SW1&6 pressed
                                ;= shift mode on/off
          JMP       SW_DONE     ;jump to end of SW routine
SW1_MUTE
          XOR   #1,MUTE         ;toggle mute setting (0 = mute off)
          JZ    MUTE_OFF        ;jump if mute=0 - turn mute off
          CALL  #MUTE_ON        ;call mute function (changed to function
                                ;so it could be used elsewhere in program)
```

```
          JMP   SW_DONE         ;jump to end of SW routine
MUTE_OFF  MOV   Vol_Pntr,Tbl_Pntr  ;set table pointer = current Volume
          CALL  #Loop           ;call I2C send routine
          CALL  #NOP9
          CALL  #NOP9
          MOV   #1655,Tbl_Pntr  ;set table pointer to unmute mix data
          CALL  #Loop           ;call I2C send routine
          JMP   SW_DONE         ;jump to end of SW routine
MUTE_ON                         ;turn mute on
          MOV   #1645,Tbl_Pntr  ;set table pointer = mute location
          CALL  #Loop           ;call I2C send routine
          CALL  #NOP9
          CALL  #NOP9
          MOV   #1662,Tbl_Pntr  ;set table pointer to mute mix data
          CALL  #Loop           ;call I2C send routine
          RET                   ;return from call

SW2_EQ                          ;send EQ
          BIT.b #SW2,&SW1_3IF   ;SW2 pressed?
          JZ    SW3_EQ          ;NO, jump to next SW
          BIT.b #LED,&LED_OUT   ;LED on? (=shift mode)
          JNZ   SW2_FUNC        ;NO - jump to SW2 function
          CALL  #MUTE_ON        ;mute before sending EQ
          CALL  #NOP9           ;delay before sending EQ
          MOV   #560,Tbl_Pntr   ;set table pointer to start of SW2 EQ
          CALL  #Loop           ;jump to I2C send routine
          CALL  #DELAY0
          MOV   #415,Tbl_Pntr   ;set table pointer to startup bass
          CALL  #Loop           ;jump to I2C send routine
          MOV   #510,Tbl_Pntr   ;set table pointer to startup treb
          CALL  #Loop           ;jump to I2C send routine
          CALL  #DELAY
          JMP   MUTE_OFF        ;jump to reset volume
SW2_FUNC
          BIC.b #LED,&LED_OUT    ;turn LED on
          CALL  #DELAY          ;delay before testing SW again
          BIC.b #SW2,&SW1_3IF   ;clear SW2 flag
          CMP   #0,Vol_Pntr     ;volume = minimum?
          JEQ   SW2_TEST        ;Yes - jump to SW2_test
          SUB   #10,Vol_Pntr    ;decrease pointer to lower setting
          MOV   Vol_Pntr,Tbl_Pntr  ;set table pointer = Vol down data
          CALL  #Loop           ;call I2C send routine
SW2_TEST  BIT.b #SW2,&P1IN      ;SW2 still pressed?
          JZ    SW2_FUNC        ;YES (active low), decrease volume
          BIS.b #LED,&LED_OUT   ;turn LED off
          JMP   SW_DONE         ;jump to end of SW routine
SW3_EQ                          ;send EQ
          BIT.b #SW3,&SW1_3IF   ;SW3 pressed?
          JZ    SW4_EQ          ;NO, jump to next SW
          CALL  #DELAY2         ;delay before testing 4th SW
          BIT.b #SW4,&SW4_6IF   ;SW4 pressed also?
          JNZ   BASS_RST        ;YES - jump to reset bass
          BIT.b #LED,&LED_OUT   ;LED on? (=shift mode)
```

```
          JNZ   SW3_FUNC        ;NO - jump SW3 function
          CALL  #MUTE_ON        ;mute before sending EQ
          CALL #NOP9                    ;delay before sending EQ
          MOV   #777,Tbl_Pntr           ;set table pointer to start of SW3 EQ
          CALL  #Loop                   ;jump to I2C send routine
          CALL  #DELAY0
          MOV   Bass_Setpnt,Bass_Pntr ;set bass pointer back to last
                                        ;manual setting
          MOV   Bass_Pntr,Tbl_Pntr    ;
          CALL  #Loop                   ;call I2C send routine
          MOV   Trbl_Setpnt,Trbl_Pntr ;set treb pointer back to last
                                        ;manual setting
          MOV   Trbl_Pntr,Tbl_Pntr ;
          CALL  #Loop                   ;call I2C send routine
          CALL  #DELAY
          JMP   MUTE_OFF                ;jump to rest volume
SW3_FUNC
          BIC.b #LED,&LED_OUT           ;turn LED on
          CALL  #DELAY                  ;delay before checking switch again
          BIC.b #SW3,&SW1_3IF           ;clear SW3 flag
          CMP   #460,Bass_Pntr          ;bass = maximum?
          JEQ   SW3_TEST                ;Yes - jump to SW3_test
          ADD   #5,Bass_Pntr            ;increase pointer to higher setting
          MOV   Bass_Pntr,Bass_Setpnt ;bass set point = bass pointer
                                        ;tracks last manual bass setting
          MOV   Bass_Pntr,Tbl_Pntr    ;set table pointer = bass up data
          CALL  #Loop                   ;call I2C send routine
SW3_TEST  BIT.b #SW3,&P1IN              ;SW3 still pressed?
          JZ    SW3_FUNC                ;YES (active low), increase bass
          BIS.b #LED,&LED_OUT           ;turn LED off
          JMPSW_DONE                    ;jump to end of SW routine
BASS_RST  MOV   #415,Bass_Pntr          ;set bass pointer to startup bass
          MOV   Bass_Pntr,Tbl_Pntr    ;set table pointer = bass startup
          CALL  #Loop                   ;jump to I2C send routine
          JMP   SW_DONE                 ;jump to end of SW routine

SW4_EQ                                  ;send EQ
          BIT.b #SW4,&SW4_6IF           ;SW4 pressed?
          JZ    SW5_EQ                  ;NO, jump to next SW
          CALL  #DELAY2                 ;delay before checking SW6
          BIT.b #SW3,&SW1_3IF           ;SW3 pressed also?
          JNZ   BASS_RST                ;YES - jump to reset bass
          BIT.b #LED,&LED_OUT           ;LED on? (=shift mode)
          JNZ   SW4_FUNC                ;NO - jump SW4 function
          CALL  #MUTE_ON                ;mute before sending EQ
          CALL #NOP9                    ;delay before sending EQ
          MOV   #994,Tbl_Pntr           ;set table pointer to start of SW4 EQ
          CALL  #Loop                   ;jump to I2C send routine
          CALL  #DELAY0
          MOV   Bass_Setpnt,Bass_Pntr ;set bass pointer back to last
                                        ;manual setting
          MOV   Bass_Pntr,Tbl_Pntr    ;
          CALL  #Loop                   ;call I2C send routine
```

```
          MOV   Trbl_Setpnt,Trbl_Pntr ;set treb pointer back to last
                                      ;manual setting
          MOVTrbl_Pntr,Tbl_Pntr ;
          CALL  #Loop                 ;call I2C send routine
          CALL  #DELAY
          JMP   MUTE_OFF              ;jump to reset volume
SW4_FUNC
          BIC.b #LED,&LED_OUT         ;turn LED on
          CALL  #DELAY                ;delay before checking switch again
          BIC.b #SW4,&SW4_6IF         ;clear SW4 flag
          CMP   #370,Bass_Pntr        ;bass = minimum?
          JEQ   SW4_TEST              ;Yes - jump to SW4_test
          SUB   #5,Bass_Pntr          ;decrease pointer to lower setting
          MOV   Bass_Pntr,Bass_Setpnt ;bass set point = bass pointer
                                      ;tracks last manual bass setting
          MOV   Bass_Pntr,Tbl_Pntr    ;set table pointer = bass down data
          CALL  #Loop                 ;call I2C send routine
SW4_TEST  BIT.b #SW4,&P2IN            ;SW4 still pressed?
          JZ    SW4_FUNC              ;YES (active low), decrease bass
          BIS.b #LED,&LED_OUT         ;turn LED off
          JMP   SW_DONE               ;jump to end of SW routine
SW5_EQ                                ;send EQ
          BIT.b #SW5,&SW4_6IF         ;SW5 pressed?
          JZ    SW6_EQ                ;NO, jump to next SW

          CALL  #DELAY2               ;delay before checking SW6
          BIT.b #SW6,&SW4_6IF         ;SW6 pressed also?
          JNZ   TRBL_RST              ;YES - jump to reset treble
          BIT.b #LED,&LED_OUT         ;LED on? (=shift mode)
          JNZ   SW5_FUNC              ;NO - jump SW5 function
          CALL  #MUTE_ON              ;mute before sending EQ
          CALL #NOP9                  ;delay before sending EQ
          MOV   #1211,Tbl_Pntr        ;set table pointer to start of SW5 EQ
          CALL  #Loop                 ;jump to I2C send routine
          CALL  #DELAY0
          MOV   Bass_Setpnt,Bass_Pntr ;set bass pointer back to last
                                      ;manual setting
          MOV   Bass_Pntr,Tbl_Pntr    ;
          CALL  #Loop                 ;call I2C send routine
          MOV   Trbl_Setpnt,Trbl_Pntr ;set treb pointer back to last
                                      ;manual setting
          MOV   Trbl_Pntr,Tbl_Pntr    ;
          CALL  #Loop                 ;call I2C send routine
          CALL  #DELAY
          JMP   MUTE_OFF              ;jump to reset volume
SW5_FUNC
          BIC.b #LED,&LED_OUT         ;turn LED on
          CALL  #DELAY                ;delay before checking switch again
          BIC.b #SW5,&SW4_6IF         ;clear SW5 flag
          CMP   #555,Trbl_Pntr        ;treble = maximum?
          JEQ   SW5_TEST              ;Yes - jump to SW5_test
          ADD   #5,Trbl_Pntr          ;increase pointer to higher setting
          MOV   Trbl_Pntr,Trbl_Setpnt ;treble set point = treble pointer
```

```
                                    ;tracks last manual treble setting
          MOV   Trbl_Pntr,Tbl_Pntr    ;set table pointer = treble up data
          CALL  #Loop               ;call I2C send routine
SW5_TEST  BIT.b #SW5,P2IN           ;SW5 still pressed?
          JZ    SW5_FUNC            ;YES (active low), increase treble
          BIS.b #LED,&LED_OUT       ;turn LED off
          JMP   SW_DONE             ;jump to reset volume
TRBL_RST  MOV   #510,Trbl_Pntr      ;set table pointer to startup treble
          MOV   Trbl_Pntr,Tbl_Pntr ;set table pointer = treb startup
          CALL  #Loop               ;jump to I2C send routine
          JMP   SW_DONE             ;jump to end of SW routine
SW6_EQ                              ;test for SW6 OR SW1+6 pressed
          BIT.b #SW6,&SW4_6IF       ;SW6 pressed?
          JZ    SW_DONE             ;NO - jump to SW ISR done

          CALL  #DELAY2             ;delay before testing 2nd SW
          BIT.b #SW1,&SW1_3IF       ;SW1 pressed also?
          JNZ   TOG_LED             ;YES - jump to toggle LED
          CALL  #NOP9               ;delay before checking SW5
          BIT.b #SW5,&SW4_6IF       ;SW5 pressed also?
          JNZ   TRBL_RST            ;YES - jump to reset treble
          BIT.b #LED,&LED_OUT       ;LED on? (=shift mode)
          JNZ   SW6_FUNC            ;NO - jump SW6 function
          CALL  #MUTE_ON            ;mute before sending EQ
          CALL #NOP9                ;delay before sending EQ
          MOV   #1428,Tbl_Pntr      ;set table pointer to start of SW6 EQ
          CALL  #Loop               ;call I2C send routine
          CALL  #DELAY0
          MOV   #415,Tbl_Pntr       ;set table pointer to startup bass
                                    ;setting
          CALL  #Loop               ;jump to I2C send routine
          MOV   #510,Tbl_Pntr       ;set table pointer to startup treble
                                    ;setting
          CALL  #Loop               ;jump to I2C send routine
          CALL  #DELAY
          JMP   MUTE_OFF            ;jump to reset volume
SW6_FUNC
          BIC.b #LED,&LED_OUT       ;turn LED on
          CALL  #DELAY              ;delay before checking switch again
          BIC.b #SW6,&SW4_6IF       ;clear SW6 flag
          CMP   #465,Trbl_Pntr      ;treble = minimum?
          JEQ   SW6_TEST            ;Yes - jump to SW6_test
          SUB   #5,Trbl_Pntr        ;decrease pointer to lower setting
          MOV   Trbl_Pntr,Trbl_Setpnt  ;treble set point = treble pointer
                                    ;tracks last manual treble setting
          MOV   Trbl_Pntr,Tbl_Pntr ;set table pointer = treble down data
          CALL  #Loop               ;call I2C send routine
SW6_TEST  BIT.b #SW6,&P2IN          ;SW6 still pressed?
          JZ    SW6_FUNC            ;YES (active low), decrease bass
          BIS.b #LED,&LED_OUT       ;turn LED off
SW_DONE                             ;complete interrupt sevice routine
          CALL  #DELAY              ;delay for switch debounce when released
          BIC.b #SW1+SW2+SW3,&SW1_3IF  ;clear SW1-3 interrupt flags (P1IFG)
```

```
         BIC.b #SW4+SW5+SW6,&SW4_6IF  ;clear SW4-6 interrupt flags (P2IFG)
         EINT                    ;enable interrupts
       RETI           ;return from ISR
;*******************************************************************************
;  Data to be sent to Digital Speaker part via I2C
;*******************************************************************************
         .EVEN          ;Align following section on even address
Data     .byte 008h     ;vol setting = 0 = mute
         .byte 068h
         .byte 004h
         .byte 000h
         .byte 000h
         .byte 000h
         .byte 000h
         .byte 000h
         .byte 000h
         .byte 000h     ;IMPORTANT-this line must be included

         .byte 008h     ;line 10
         .byte 068h
         .byte 004h
         .byte 000h
         .byte 004h
         .byte 08Dh
         .byte 000h
         .byte 004h
         .byte 08Dh
         .byte 000h     ;IMPORTANT-this line must be included
         .byte 008h     ;vol setting = 2      line 20
         .byte 068h
         .byte 004h
         .byte 000h
         .byte 005h
         .byte 01Ch
         .byte 000h
         .byte 005h
         .byte 01Ch
         .byte 000h     ;IMPORTANT-this line must be included
         .byte 008h     ;line 30
         .byte 068h
         .byte 004h
         .byte 000h
         .byte 005h
         .byte 0BBh
         .byte 000h
         .byte 005h
         .byte 0BBh
         .byte 000h     ;IMPORTANT-this line must be included
         .byte 008h     ;vol setting = 4
         .byte 068h
         .byte 004h
         .byte 000h
         .byte 006h
```

```
.byte 06Eh
.byte 000h
.byte 006h
.byte 06Eh
.byte 000h      ;IMPORTANT-this line must be included
.byte 008h      ;vol setting =     line 50
.byte 068h
.byte 004h
.byte 000h
.byte 007h
.byte 037h
.byte 000h
.byte 007h
.byte 037h
.byte 000h      ;IMPORTANT-this line must be included
.byte 008h      ;vol setting = 6
.byte 068h
.byte 004h
.byte 000h
.byte 008h
.byte 018h
.byte 000h
.byte 008h
.byte 018h
.byte 000h      ;IMPORTANT-this line must be included
.byte 008h      ;vol setting =
.byte 068h
.byte 004h
.byte 000h
.byte 009h
.byte 015h
.byte 000h
.byte 009h
.byte 015h
.byte 000h      ;IMPORTANT-this line must be included
.byte 008h      ;vol setting = 8      line 80
.byte 068h
.byte 004h
.byte 000h
.byte 00Ah
.byte 031h
.byte 000h
.byte 00Ah
.byte 031h
.byte 000h      ;IMPORTANT-this line must be included
.byte 008h      ;vol setting =
.byte 068h
.byte 004h
.byte 000h
.byte 00Bh
.byte 06Fh
.byte 000h
```

```
.byte 00Bh
.byte 06Fh
.byte 000h      ;IMPORTANT-this line must be included
.byte 008h      ;vol setting = 10     line 100
.byte 068h
.byte 004h
.byte 000h
.byte 00Ch
.byte 0D5h
.byte 000h
.byte 00Ch
.byte 0D5h
.byte 000h      ;IMPORTANT-this line must be included
.byte 008h      ;vol setting = 11 = -25 dB
.byte 068h
.byte 004h
.byte 000h
.byte 00Eh
.byte 065h
.byte 000h
.byte 00Eh
.byte 065h
.byte 000h      ;IMPORTANT-this line must be included
.byte 008h      ;vol setting = 12     line 120
.byte 068h
.byte 004h
.byte 000h
.byte 010h
.byte 027h
.byte 000h
.byte 010h
.byte 027h
.byte 000h      ;IMPORTANT-this line must be included
.byte 008h      ;vol setting =
.byte 068h
.byte 004h
.byte 000h
.byte 012h
.byte 020h
.byte 000h
.byte 012h
.byte 020h
.byte 000h      ;IMPORTANT-this line must be included
.byte 008h      ;vol setting = 14     line 140
.byte 068h
.byte 004h
.byte 000h
.byte 014h
.byte 056h
.byte 000h
.byte 014h
.byte 056h
```

```
.byte 000h      ;IMPORTANT-this line must be included
.byte 008h      ;vol setting =
.byte 068h
.byte 004h
.byte 000h
.byte 016h
.byte 0D1h
.byte 000h
.byte 016h
.byte 0D1h
.byte 000h      ;IMPORTANT-this line must be included
.byte 008h      ;vol setting = 16     line 160
.byte 068h
.byte 004h
.byte 000h
.byte 019h
.byte 09Ah
.byte 000h
.byte 019h
.byte 09Ah
.byte 000h      ;IMPORTANT-this line must be included
.byte 008h      ;vol setting =
.byte 068h
.byte 004h
.byte 000h
.byte 01Ch
.byte 0B9h
.byte 000h
.byte 01Ch
.byte 0B9h
.byte 000h      ;IMPORTANT-this line must be included
                ;***** line 180 **********************
.byte 008h      ;vol setting = 18 = starting point ****
.byte 068h
.byte 004h
.byte 000h
.byte 020h
.byte 03Ah
.byte 000h
.byte 020h
.byte 03Ah
.byte 000h      ;IMPORTANT-this line must be included
.byte 008h      ;vol setting =
.byte 068h
.byte 004h
.byte 000h
.byte 024h
.byte 029h
.byte 000h
.byte 024h
.byte 029h
.byte 000h      ;IMPORTANT-this line must be included
```

```
.byte 008h     ;vol setting = 20     line 200
.byte 068h
.byte 004h
.byte 000h
.byte 028h
.byte 093h
.byte 000h
.byte 028h
.byte 093h
.byte 000h     ;IMPORTANT-this line must be included
.byte 008h     ;vol setting =
.byte 068h
.byte 004h
.byte 000h
.byte 02Dh
.byte 086h
.byte 000h
.byte 02Dh
.byte 086h
.byte 000h     ;IMPORTANT-this line must be included
.byte 008h     ;vol setting = 22     line 220
.byte 068h
.byte 004h
.byte 000h
.byte 033h
.byte 014h
.byte 000h
.byte 033h
.byte 014h
.byte 000h     ;IMPORTANT-this line must be included
.byte 008h     ;vol setting =
.byte 068h
.byte 004h
.byte 000h
.byte 039h
.byte 050h
.byte 000h
.byte 039h
.byte 050h
.byte 000h     ;IMPORTANT-this line must be included
.byte 008h     ;vol setting = 24     line 240
.byte 068h
.byte 004h
.byte 000h
.byte 040h
.byte 04Eh
.byte 000h
.byte 040h
.byte 04Eh
.byte 000h     ;IMPORTANT-this line must be included
.byte 008h     ;vol setting =
.byte 068h
```

```
.byte 004h
.byte 000h
.byte 048h
.byte 027h
.byte 000h
.byte 048h
.byte 027h
.byte 000h      ;IMPORTANT-this line must be included
.byte 008h      ;vol setting = 26     line 260
.byte 068h
.byte 004h
.byte 000h
.byte 050h
.byte 0F4h
.byte 000h
.byte 050h
.byte 0F4h
.byte 000h      ;IMPORTANT-this line must be included
.byte 008h      ;vol setting =
.byte 068h
.byte 004h
.byte 000h
.byte 05Ah
.byte 0D5h
.byte 000h
.byte 05Ah
.byte 0D5h
.byte 000h      ;IMPORTANT-this line must be included
.byte 008h      ;vol setting = 28     line 280
.byte 068h
.byte 004h
.byte 000h
.byte 065h
.byte 0EAh
.byte 000h
.byte 065h
.byte 0EAh
.byte 000h      ;IMPORTANT-this line must be included
.byte 008h      ;vol setting =
.byte 068h
.byte 004h
.byte 000h
.byte 072h
.byte 05Ah
.byte 000h
.byte 072h
.byte 05Ah
.byte 000h      ;IMPORTANT-this line must be included
.byte 008h      ;vol setting = 30     line 300
.byte 068h
.byte 004h
.byte 000h
```

```
.byte 080h
.byte 04Eh
.byte 000h
.byte 080h
.byte 04Eh
.byte 000h      ;IMPORTANT-this line must be included
.byte 008h      ;vol setting =
.byte 068h
.byte 004h
.byte 000h
.byte 08Fh
.byte 0F6h
.byte 000h
.byte 08Fh
.byte 0F6h
.byte 000h      ;IMPORTANT-this line must be included
.byte 008h      ;vol setting = 32     line 320
.byte 068h
.byte 004h
.byte 000h
.byte 0A1h
.byte 086h
.byte 000h
.byte 0A1h
.byte 086h
.byte 000h      ;IMPORTANT-this line must be included
.byte 008h      ;vol setting =
.byte 068h
.byte 004h
.byte 000h
.byte 0B5h
.byte 03Ch
.byte 000h
.byte 0B5h
.byte 03Ch
.byte 000h      ;IMPORTANT-this line must be included
.byte 008h      ;vol setting = 34     line 340
.byte 068h
.byte 004h
.byte 000h
.byte 0CBh
.byte 059h
.byte 000h
.byte 0CBh
.byte 059h
.byte 000h      ;IMPORTANT-this line must be included
.byte 008h      ;vol setting =
.byte 068h
.byte 004h
.byte 000h
.byte 0E4h
.byte 029h
```

```
.byte 000h
.byte 0E4h
.byte 029h
.byte 000h      ;IMPORTANT-this line must be included
.byte 008h      ;vol setting = 36 = max = 0 dB  line 360
.byte 068h
.byte 004h
.byte 001h
.byte 000h
.byte 000h
.byte 001h
.byte 000h
.byte 000h
.byte 000h      ;IMPORTANT-this line must be included
                ;***** END OF VOLUME DATA ***********
                ;***** line 370 *********************
.byte 003h      ;bass setting = 0 = minimum = -9 db
.byte 068h
.byte 006h
.byte 05Dh
.byte 000h      ;IMPORTANT-this line must be included
.byte 003h      ;bass setting =
.byte 068h
.byte 006h
.byte 05Ah
.byte 000h      ;IMPORTANT-this line must be included
.byte 003h      ;bass setting = 2     line 380
.byte 068h
.byte 006h
.byte 058h
.byte 000h      ;IMPORTANT-this line must be included
.byte 003h      ;bass setting =
.byte 068h
.byte 006h
.byte 055h
.byte 000h      ;IMPORTANT-this line must be included
.byte 003h      ;bass setting = 4     line 390
.byte 068h
.byte 006h
.byte 053h
.byte 000h      ;IMPORTANT-this line must be included
.byte 003h      ;bass setting =
.byte 068h
.byte 006h
.byte 04Fh
.byte 000h      ;IMPORTANT-this line must be included
.byte 003h      ;bass setting = 6     line 400
.byte 068h
.byte 006h
.byte 04Bh
.byte 000h      ;IMPORTANT-this line must be included
.byte 003h      ;bass setting =
```

```
.byte 068h
.byte 006h
.byte 046h
.byte 000h     ;IMPORTANT-this line must be included
.byte 003h     ;bass setting = 8     line 410
.byte 068h
.byte 006h
.byte 042h
.byte 000h     ;IMPORTANT-this line must be included
               ;***** line 415 ****************************
.byte 003h     ;bass setting = 9 = mid point = starting point
.byte 068h
.byte 006h
.byte 03Eh
.byte 000h     ;IMPORTANT-this line must be included
.byte 003h     ;bass setting = 10    line 420
.byte 068h
.byte 006h
.byte 03Bh
.byte 000h     ;IMPORTANT-this line must be included
.byte 003h     ;bass setting =
.byte 068h
.byte 006h
.byte 038h
.byte 000h     ;IMPORTANT-this line must be included
.byte 003h     ;bass setting = 12 line 430
.byte 068h
.byte 006h
.byte 035h
.byte 000h     ;IMPORTANT-this line must be included
.byte 003h     ;bass setting =
.byte 068h
.byte 006h
.byte 031h
.byte 000h     ;IMPORTANT-this line must be included
.byte 003h     ;bass setting = 14 line 440
.byte 068h
.byte 006h
.byte 02Eh
.byte 000h     ;IMPORTANT-this line must be included
.byte 003h     ;bass setting =
.byte 068h
.byte 006h
.byte 02Bh
.byte 000h     ;IMPORTANT-this line must be included
.byte 003h     ;bass setting = 16 line 450
.byte 068h
.byte 006h
.byte 028h
.byte 000h     ;IMPORTANT-this line must be included
.byte 003h     ;bass setting =
.byte 068h
.byte 006h
```

```
.byte 025h
.byte 000h     ;IMPORTANT-this line must be included
               ;***** line 460 *********************
.byte 003h     ;bass setting = 18 = maximum = +9 dB
.byte 068h
.byte 006h
.byte 021h
.byte 000h     ;IMPORTANT-this line must be included
               ;***** END OF TREBLE DATA ************
               ;***** line 465 *********************
.byte 003h     ;treble setting 0 = minimum = -9 dB
.byte 068h
.byte 005h
.byte 084h
.byte 000h     ;IMPORTANT-this line must be included
.byte 003h     ;treble setting =
.byte 068h
.byte 005h
.byte 082h
.byte 000h     ;IMPORTANT-this line must be included
.byte 003h     ;treble setting = 2   line 475
.byte 068h
.byte 005h
.byte 080h
.byte 000h     ;IMPORTANT-this line must be included
.byte 003h     ;treble setting =
.byte 068h
.byte 005h
.byte 07Eh
.byte 000h     ;IMPORTANT-this line must be included
.byte 003h     ;treble setting = 4   line 485
.byte 068h
.byte 005h
.byte 07Ch
.byte 000h     ;IMPORTANT-this line must be included
.byte 003h     ;treble setting =
.byte 068h
.byte 005h
.byte 07Ah
.byte 000h     ;IMPORTANT-this line must be included
.byte 003h     ;treble setting = 6   line 495
.byte 068h
.byte 005h
.byte 078h
.byte 000h     ;IMPORTANT-this line must be included
.byte 003h     ;treble setting =
.byte 068h
.byte 005h
.byte 076h
.byte 000h     ;IMPORTANT-this line must be included
.byte 003h     ;treble setting = 8   line 505
.byte 068h
```

```
.byte 005h
.byte 074h
.byte 000h     ;IMPORTANT-this line must be included
            ;***** line 510 ******************************
.byte003h ;treble setting = 9 = starting point **********
.byte 068h
.byte 005h
.byte 072h
.byte 000h     ;IMPORTANT-this line must be included
.byte 003h     ;treble setting = 10
.byte 068h
.byte 005h
.byte 070h
.byte 000h     ;IMPORTANT-this line must be included
.byte 003h     ;treble setting =     line 520
.byte 068h
.byte 005h
.byte 06Dh
.byte 000h     ;IMPORTANT-this line must be included
.byte 003h     ;treble setting = 12
.byte 068h
.byte 005h
.byte 06Bh
.byte 000h     ;IMPORTANT-this line must be included
.byte 003h     ;treble setting =     line 530
.byte 068h
.byte 005h
.byte 068h
.byte 000h     ;IMPORTANT-this line must be included
.byte 003h     ;treble setting = 14
.byte 068h
.byte 005h
.byte 065h
.byte 000h     ;IMPORTANT-this line must be included
.byte 003h     ;treble setting =     line 540
.byte 068h
.byte 005h
.byte 062h
.byte 000h     ;IMPORTANT-this line must be included
.byte 003h     ;treble setting = 16
.byte 068h
.byte 005h
.byte 05Eh
.byte 000h     ;IMPORTANT-this line must be included
.byte 003h     ;treble setting =     line 550
.byte 068h
.byte 005h
.byte 05Ah
.byte 000h     ;IMPORTANT-this line must be included
               ;***** line 555 *********************
.byte 003h     ;treble setting = 18 = maximum = +9 dB
.byte 068h
```

```
.byte 005h
.byte 055h
.byte 000h      ;IMPORTANT-this line must be included
                ;***** END OF TREBLE DATA ***********


                ;***** line 560 *********************
.byte 011h      ;start of EQs for SW2 -- ORIGINAL
.byte 068h
.byte 00Ah
.byte 010h
.byte 000h
.byte 000h
.byte 000h
.byte 000h
.byte 000h
.byte 000h
.byte 000h
.byte 000h
.byte 000h
.byte 000h
.byte 000h
.byte 000h
.byte 000h
.byte 011h      ;line 578
.byte 068h
.byte 00Bh
.byte 010h
.byte 000h
.byte 000h
.byte 000h
.byte 000h
.byte 000h
.byte 000h
.byte 000h
.byte 000h
.byte 000h
.byte 000h
.byte 000h
.byte 000h
.byte 000h
.byte 000h
.byte 000h

.byte 011h      ;line 596
.byte 068h
.byte 00Ch
.byte 010h
.byte 000h
.byte 000h
.byte 000h
.byte 000h
```

```
        .byte 000h
        .byte 000h
        .byte 000h
        .byte 000h
        .byte 000h
        .byte 000h
        .byte 000h
        .byte 000h
        .byte 000h
        .byte 000h

        .byte 011h
        .byte 068h
        .byte 00Dh
        .byte 010h
        .byte 000h
        .byte 000h
        .byte 000h
        .byte 000h
        .byte 000h
        .byte 000h
        .byte 000h
        .byte 000h
        .byte 000h
        .byte 000h
        .byte 000h
        .byte 000h
        .byte 000h
        .byte 000h
        .byte 000h
        .byte 000h

        .byte 011h
        .byte 068h
        .byte 00Eh
        .byte 003h
        .byte 085h
        .byte 0EAh
        .byte 007h
        .byte 00Bh
        .byte 0D5h
        .byte 003h
        .byte 085h
        .byte 0EAh
        .byte 0FBh
        .byte 014h
        .byte 036h
        .byte 003h
        .byte 003h
        .byte 076h

        .byte 011h
        .byte 068h
        .byte 00Fh
```

```
.byte 00Fh
.byte 0DAh
.byte 043h
.byte 0E0h
.byte 04Bh
.byte 07Ah
.byte 00Fh
.byte 0DAh
.byte 043h
.byte 0E0h
.byte 04Bh
.byte 0D3h
.byte 00Fh
.byte 0B4h
.byte 0DFh  ;end of left EQ for SW2

.byte 011h
.byte 068h
.byte 013h
.byte 010h
.byte 000h
.byte 000h
.byte 000h
.byte 000h
.byte 000h
.byte 000h
.byte 000h
.byte 000h
.byte 000h
.byte 000h
.byte 000h
.byte 000h
.byte 000h
.byte 000h

.byte 011h
.byte 068h
.byte 014h
.byte 010h
.byte 000h
.byte 000h
.byte 000h
.byte 000h
.byte 000h
.byte 000h
.byte 000h
.byte 000h
.byte 000h
.byte 000h
.byte 000h
.byte 000h
.byte 000h
.byte 000h
```

```
.byte 000h
.byte 011h
.byte 068h
.byte 015h
.byte 010h
.byte 000h
.byte 000h
.byte 000h
.byte 000h
.byte 000h
.byte 000h
.byte 000h
.byte 000h
.byte 000h
.byte 000h
.byte 000h
.byte 000h
.byte 000h
.byte 000h
.byte 011h
.byte 068h
.byte 016h
.byte 010h
.byte 000h
.byte 000h
.byte 000h
.byte 000h
.byte 000h
.byte 000h
.byte 000h
.byte 000h
.byte 000h
.byte 000h
.byte 000h
.byte 000h
.byte 000h
.byte 000h
.byte 000h
.byte 000h
.byte 011h
.byte 068h
.byte 017h
.byte 003h
.byte 085h
.byte 0EAh
.byte 007h
.byte 00Bh
.byte 0D5h
.byte 003h
.byte 085h
.byte 0EAh
.byte 0FBh
.byte 014h
.byte 036h
```

```
.byte 003h
.byte 003h
.byte 076h
.byte 011h
.byte 068h
.byte 018h
.byte 00Fh
.byte 0DAh
.byte 043h
.byte 0E0h
.byte 04Bh
.byte 07Ah
.byte 00Fh
.byte 0DAh
.byte 043h
.byte 0E0h
.byte 04Bh
.byte 0D3h
.byte 00Fh
.byte 0B4h
.byte 0DFh  ;end of right EQ for SW2
            ;***************************
.byte 000h  ;IMPORTANT-this line must be included


            ;***** line 777 **************
.byte 011h  ;start of EQs for SW3 -- FLAT2
.byte 068h
.byte 00Ah
.byte 00Fh
.byte 056h
.byte 0E8h
.byte 0E2h
.byte 0A7h
.byte 053h
.byte 00Eh
.byte 045h
.byte 0E6h
.byte 0E2h
.byte 0A7h
.byte 053h
.byte 00Dh
.byte 09Ch
.byte 0CFh
.byte 011h
.byte 068h
.byte 00Bh
.byte 00Fh
.byte 0E0h
.byte 079h
.byte 0E0h
.byte 07Fh
```

```
.byte 091h
.byte 00Fh
.byte 0A1h
.byte 020h
.byte 0E0h
.byte 07Fh
.byte 091h
.byte 00Fh
.byte 081h
.byte 09Ah

.byte 011h
.byte 068h
.byte 00Ch
.byte 00Fh
.byte 064h
.byte 0F1h
.byte 0ECh
.byte 0A5h
.byte 01Ch
.byte 00Dh
.byte 052h
.byte 0BEh
.byte 0ECh
.byte 0A5h
.byte 01Ch
.byte 00Ch
.byte 0B7h
.byte 0B0h

.byte 011h
.byte 068h
.byte 00Dh
.byte 018h
.byte 0CEh
.byte 0A8h
.byte 0EEh
.byte 01Bh
.byte 033h
.byte 006h
.byte 093h
.byte 085h
.byte 0FAh
.byte 065h
.byte 075h
.byte 003h
.byte 017h
.byte 0ECh

.byte 011h
.byte 068h
.byte 00Eh
```

```
.byte 010h
.byte 00Eh
.byte 098h
.byte 0E0h
.byte 010h
.byte 066h
.byte 00Fh
.byte 0E1h
.byte 059h
.byte 0E0h
.byte 010h
.byte 066h
.byte 00Fh
.byte 0EFh
.byte 0F2h

.byte 011h
.byte 068h
.byte 00Fh
.byte 010h
.byte 010h
.byte 02Bh
.byte 0E0h
.byte 015h
.byte 0F6h
.byte 00Fh
.byte 0DAh
.byte 070h
.byte 0E0h
.byte 015h
.byte 0F6h
.byte 00Fh
.byte 0EAh
.byte 09Bh  ;end of left EQ for SW3

.byte 011h
.byte 068h
.byte 013h
.byte 00Fh
.byte 056h
.byte 0E8h
.byte 0E2h
.byte 0A7h
.byte 053h
.byte 00Eh
.byte 045h
.byte 0E6h
.byte 0E2h
.byte 0A7h
.byte 053h
.byte 00Dh
.byte 09Ch
```

```
        .byte 0CFh

        .byte 011h
        .byte 068h
        .byte 014h
        .byte 00Fh
        .byte 0E0h
        .byte 079h
        .byte 0E0h
        .byte 07Fh
        .byte 091h
        .byte 00Fh
        .byte 0A1h
        .byte 020h
        .byte 0E0h
        .byte 07Fh
        .byte 091h
        .byte 00Fh
        .byte 081h
        .byte 09Ah
        .byte 011h
        .byte 068h
        .byte 015h
        .byte 00Fh
        .byte 064h
        .byte 0F1h
        .byte 0ECh
        .byte 0A5h
        .byte 01Ch
        .byte 00Dh
        .byte 052h
        .byte 0BEh
        .byte 0ECh
        .byte 0A5h
        .byte 01Ch
        .byte 00Ch
        .byte 0B7h
        .byte 0B0h
        .byte 011h
        .byte 068h
        .byte 016h
        .byte 010h
        .byte 000h
        .byte 000h
        .byte 000h
        .byte 000h
        .byte 000h
        .byte 000h
        .byte 000h
        .byte 000h
        .byte 000h
        .byte 000h
        .byte 000h
```

```
.byte 000h
.byte 000h
.byte 000h
.byte 000h
.byte 011h
.byte 068h
.byte 017h
.byte 010h
.byte 00Eh
.byte 098h
.byte 0E0h
.byte 010h
.byte 066h
.byte 00Fh
.byte 0E1h
.byte 059h
.byte 0E0h
.byte 010h
.byte 066h
.byte 00Fh
.byte 0EFh
.byte 0F2h
.byte 011h
.byte 068h
.byte 018h
.byte 010h
.byte 010h
.byte 02Bh
.byte 0E0h
.byte 015h
.byte 0F6h
.byte 00Fh
.byte 0DAh
.byte 070h
.byte 0E0h
.byte 015h
.byte 0F6h
.byte 00Fh
.byte 0EAh
.byte 09Bh  ;end of right EQ for SW3
            ;***************************
.byte 000h  ;IMPORTANT-this line must be included


            ;***** line 994 *********************
.byte 011h  ;start of EQs for SW4 -- JAZZ
.byte 068h
.byte 00Ah
.byte 00Fh
.byte 08Bh
.byte 038h
.byte 0E1h
```

```
.byte 0E8h
.byte 00Fh
.byte 00Eh
.byte 0DDh
.byte 012h
.byte 0E1h
.byte 0E4h
.byte 099h
.byte 00Eh
.byte 06Bh
.byte 0C0h
.byte 011h
.byte 068h
.byte 00Bh
.byte 00Fh
.byte 011h
.byte 08Bh
.byte 0EDh
.byte 024h
.byte 072h
.byte 00Dh
.byte 015h
.byte 0FCh
.byte 0EDh
.byte 012h
.byte 028h
.byte 00Ch
.byte 039h
.byte 0D2h

.byte 011h
.byte 068h
.byte 00Ch
.byte 010h
.byte 004h
.byte 04Dh
.byte 0E0h
.byte 00Bh
.byte 034h
.byte 00Fh
.byte 0F1h
.byte 08Ah
.byte 0E0h
.byte 00Bh
.byte 035h
.byte 00Fh
.byte 0F5h
.byte 0D6h

.byte 011h
.byte 068h
.byte 00Dh
```

```
.byte 010h
.byte 000h
.byte 0E3h
.byte 0E0h
.byte 002h
.byte 0B7h
.byte 00Fh
.byte 0FDh
.byte 010h
.byte 0E0h
.byte 002h
.byte 0B7h
.byte 00Fh
.byte 0FDh
.byte 0F4h

.byte 011h
.byte 068h
.byte 00Eh
.byte 045h
.byte 04Ch
.byte 047h
.byte 0FAh
.byte 08Bh
.byte 06Dh
.byte 0DFh
.byte 04Eh
.byte 061h
.byte 00Ch
.byte 02Ah
.byte 0BEh
.byte 002h
.byte 0FBh
.byte 055h

.byte 011h
.byte 068h
.byte 00Fh
.byte 010h
.byte 000h
.byte 000h
.byte 000h
.byte 000h
.byte 000h
.byte 000h
.byte 000h
.byte 000h
.byte 000h
.byte 000h
.byte 000h
.byte 000h
.byte 000h
.byte 000h
```

```
.byte 000h  ;end of left EQ for SW4

.byte 011h
.byte 068h
.byte 013h
.byte 00Fh
.byte 08Bh
.byte 038h
.byte 0E1h
.byte 0E8h
.byte 00Fh
.byte 00Eh
.byte 0DDh
.byte 012h
.byte 0E1h
.byte 0E4h
.byte 099h
.byte 00Eh
.byte 06Bh
.byte 0C0h

.byte 011h
.byte 068h
.byte 014h
.byte 00Fh
.byte 011h
.byte 08Bh
.byte 0EDh
.byte 024h
.byte 072h
.byte 00Dh
.byte 015h
.byte 0FCh
.byte 0EDh
.byte 012h
.byte 028h
.byte 00Ch
.byte 039h
.byte 0D2h
.byte 011h
.byte 068h
.byte 015h
.byte 010h
.byte 004h
.byte 04Dh
.byte 0E0h
.byte 00Bh
.byte 034h
.byte 00Fh
.byte 0F1h
.byte 08Ah
.byte 0E0h
```

```
.byte 00Bh
.byte 035h
.byte 00Fh
.byte 0F5h
.byte 0D6h
.byte 011h
.byte 068h
.byte 016h
.byte 010h
.byte 000h
.byte 0E3h
.byte 0E0h
.byte 002h
.byte 0B7h
.byte 00Fh
.byte 0FDh
.byte 010h
.byte 0E0h
.byte 002h
.byte 0B7h
.byte 00Fh
.byte 0FDh
.byte 0F4h
.byte 011h
.byte 068h
.byte 017h
.byte 045h
.byte 04Ch
.byte 047h
.byte 0FAh
.byte 08Bh
.byte 06Dh
.byte 0DFh
.byte 04Eh
.byte 061h
.byte 00Ch
.byte 02Ah
.byte 0BEh
.byte 002h
.byte 0FBh
.byte 055h
.byte 011h
.byte 068h
.byte 018h
.byte 010h
.byte 000h
.byte 000h
.byte 000h
.byte 000h
.byte 000h
.byte 000h
.byte 000h
```

```
.byte 000h
.byte 000h
.byte 000h
.byte 000h
.byte 000h
.byte 000h
.byte 000h  ;end of right EQ for SW4
            ;**************************
.byte 000h  ;IMPORTANT-this line must be included


            ;***** line 1211 ********************
.byte 011h  ;start of EQs for SW5 -- ROCK
.byte 068h
.byte 00Ah
.byte 00Fh
.byte 0EDh
.byte 016h
.byte 0E0h
.byte 025h
.byte 0D4h
.byte 00Fh
.byte 0EDh
.byte 016h
.byte 0E0h
.byte 025h
.byte 0EAh
.byte 00Fh
.byte 0DAh
.byte 043h
.byte 011h
.byte 068h
.byte 00Bh
.byte 00Fh
.byte 08Bh
.byte 038h
.byte 0E1h
.byte 0E8h
.byte 00Fh
.byte 00Eh
.byte 0DDh
.byte 012h
.byte 0E1h
.byte 0E4h
.byte 099h
.byte 00Eh
.byte 06Bh
.byte 0C0h

.byte 011h
.byte 068h
.byte 00Ch
```

```
        .byte 00Fh
        .byte 011h
        .byte 08Bh
        .byte 0EDh
        .byte 024h
        .byte 072h
        .byte 00Dh
        .byte 015h
        .byte 0FCh
        .byte 0EDh
        .byte 012h
        .byte 028h
        .byte 00Ch
        .byte 039h
        .byte 0D2h

        .byte 011h
        .byte 068h
        .byte 00Dh
        .byte 010h
        .byte 010h
        .byte 02Bh
        .byte 0E0h
        .byte 015h
        .byte 0F6h
        .byte 00Fh
        .byte 0DAh
        .byte 070h
        .byte 0E0h
        .byte 015h
        .byte 0F6h
        .byte 00Fh
        .byte 0EAh
        .byte 09Bh

        .byte 011h
        .byte 068h
        .byte 00Eh
        .byte 010h
        .byte 00Eh
        .byte 098h
        .byte 0E0h
        .byte 010h
        .byte 066h
        .byte 00Fh
        .byte 0E1h
        .byte 059h
        .byte 0E0h
        .byte 010h
        .byte 066h
        .byte 00Fh
        .byte 0EFh
```

```
        .byte  0F2h

        .byte  011h
        .byte  068h
        .byte  00Fh
        .byte  045h
        .byte  04Ch
        .byte  047h
        .byte  0FAh
        .byte  08Bh
        .byte  06Dh
        .byte  0DFh
        .byte  04Eh
        .byte  061h
        .byte  00Ch
        .byte  02Ah
        .byte  0BEh
        .byte  002h
        .byte  0FBh
        .byte  055h   ;end of left EQ for SW5

        .byte  011h
        .byte  068h
        .byte  013h
        .byte  00Fh
        .byte  0EDh
        .byte  016h
        .byte  0E0h
        .byte  025h
        .byte  0D4h
        .byte  00Fh
        .byte  0EDh
        .byte  016h
        .byte  0E0h
        .byte  025h
        .byte  0EAh
        .byte  00Fh
        .byte  0DAh
        .byte  043h

        .byte  011h
        .byte  068h
        .byte  014h
        .byte  00Fh
        .byte  08Bh
        .byte  038h
        .byte  0E1h
        .byte  0E8h
        .byte  00Fh
        .byte  00Eh
        .byte  0DDh
        .byte  012h
```

```
.byte 0E1h
.byte 0E4h
.byte 099h
.byte 00Eh
.byte 06Bh
.byte 0C0h
.byte 011h
.byte 068h
.byte 015h
.byte 00Fh
.byte 011h
.byte 08Bh
.byte 0EDh
.byte 024h
.byte 072h
.byte 00Dh
.byte 015h
.byte 0FCh
.byte 0EDh
.byte 012h
.byte 028h
.byte 00Ch
.byte 039h
.byte 0D2h
.byte 011h
.byte 068h
.byte 016h
.byte 010h
.byte 010h
.byte 02Bh
.byte 0E0h
.byte 015h
.byte 0F6h
.byte 00Fh
.byte 0DAh
.byte 070h
.byte 0E0h
.byte 015h
.byte 0F6h
.byte 00Fh
.byte 0EAh
.byte 09Bh
.byte 011h
.byte 068h
.byte 017h
.byte 010h
.byte 00Eh
.byte 098h
.byte 0E0h
.byte 010h
.byte 066h
.byte 00Fh
```

```
.byte 0E1h
.byte 059h
.byte 0E0h
.byte 010h
.byte 066h
.byte 00Fh
.byte 0EFh
.byte 0F2h
.byte 011h
.byte 068h
.byte 018h
.byte 045h
.byte 04Ch
.byte 047h
.byte 0FAh
.byte 08Bh
.byte 06Dh
.byte 0DFh
.byte 04Eh
.byte 061h
.byte 00Ch
.byte 02Ah
.byte 0BEh
.byte 002h
.byte 0FBh
.byte 055h  ;end of right EQ for SW5
            ;***************************
.byte 000h  ;IMPORTANT-this line must be included


            ;***** line 1428 ********************
.byte 011h  ;start of EQs for SW6 -- VOICE
.byte 068h
.byte 00Ah
.byte 010h
.byte 000h
.byte 000h
.byte 000h
.byte 000h
.byte 000h
.byte 000h
.byte 000h
.byte 000h
.byte 000h
.byte 000h
.byte 000h
.byte 000h
.byte 000h
.byte 000h
.byte 000h
.byte 000h
.byte 011h
.byte 068h
.byte 00Bh
```

```
        .byte 010h
        .byte 000h
        .byte 000h
        .byte 000h
        .byte 000h
        .byte 000h
        .byte 000h
        .byte 000h
        .byte 000h
        .byte 000h
        .byte 000h
        .byte 000h
        .byte 000h
        .byte 000h
        .byte 000h

        .byte 011h
        .byte 068h
        .byte 00Ch
        .byte 010h
        .byte 000h
        .byte 000h
        .byte 000h
        .byte 000h
        .byte 000h
        .byte 000h
        .byte 000h
        .byte 000h
        .byte 000h
        .byte 000h
        .byte 000h
        .byte 000h
        .byte 000h
        .byte 000h

        .byte 011h
        .byte 068h
        .byte 00Dh
        .byte 00Fh
        .byte 09Ah
        .byte 01Fh
        .byte 0E0h
        .byte 0CBh
        .byte 0C2h
        .byte 00Fh
        .byte 09Ah
        .byte 01Fh
        .byte 0E0h
        .byte 0CEh
        .byte 04Ah
        .byte 00Fh
        .byte 036h
```

```
        .byte 0C7h

        .byte 011h
        .byte 068h
        .byte 00Eh
        .byte 001h
        .byte 01Dh
        .byte 0D2h
        .byte 002h
        .byte 03Bh
        .byte 0A5h
        .byte 001h
        .byte 01Dh
        .byte 0D2h
        .byte 0EDh
        .byte 0FAh
        .byte 026h
        .byte 006h
        .byte 07Dh
        .byte 023h

        .byte 011h
        .byte 068h
        .byte 00Fh
        .byte 039h
        .byte 06Eh
        .byte 0CAh
        .byte 099h
        .byte 027h
        .byte 059h
        .byte 02Eh
        .byte 08Eh
        .byte 0A5h
        .byte 0E6h
        .byte 098h
        .byte 02Ah
        .byte 00Ah
        .byte 08Ch
        .byte 09Fh   ;end of left EQ for SW6

        .byte 011h
        .byte 068h
        .byte 013h
        .byte 010h
        .byte 000h
        .byte 000h
        .byte 000h
        .byte 000h
        .byte 000h
        .byte 000h
        .byte 000h
        .byte 000h
```

```
        .byte 000h
        .byte 000h
        .byte 000h
        .byte 000h
        .byte 000h
        .byte 000h

        .byte 011h
        .byte 068h
        .byte 014h
        .byte 010h
        .byte 000h
        .byte 000h
        .byte 000h
        .byte 000h
        .byte 000h
        .byte 000h
        .byte 000h
        .byte 000h
        .byte 000h
        .byte 000h
        .byte 000h
        .byte 000h
        .byte 000h
        .byte 000h
        .byte 000h
        .byte 011h
        .byte 068h
        .byte 015h
        .byte 010h
        .byte 000h
        .byte 000h
        .byte 000h
        .byte 000h
        .byte 000h
        .byte 000h
        .byte 000h
        .byte 000h
        .byte 000h
        .byte 000h
        .byte 000h
        .byte 000h
        .byte 000h
        .byte 000h
        .byte 000h
        .byte 011h
        .byte 068h
        .byte 016h
        .byte 00Fh
        .byte 09Ah
        .byte 01Fh
        .byte 0E0h
        .byte 0CBh
        .byte 0C2h
```

```
.byte 00Fh
.byte 09Ah
.byte 01Fh
.byte 0E0h
.byte 0CEh
.byte 04Ah
.byte 00Fh
.byte 036h
.byte 0C7h
.byte 011h
.byte 068h
.byte 017h
.byte 001h
.byte 01Dh
.byte 0D2h
.byte 002h
.byte 03Bh
.byte 0A5h
.byte 001h
.byte 01Dh
.byte 0D2h
.byte 0EDh
.byte 0FAh
.byte 026h
.byte 006h
.byte 07Dh
.byte 023h
.byte 011h
.byte 068h
.byte 018h
.byte 039h
.byte 06Eh
.byte 0CAh
.byte 099h
.byte 027h
.byte 059h
.byte 02Eh
.byte 08Eh
.byte 0A5h
.byte 0E6h
.byte 098h
.byte 02Ah
.byte 00Ah
.byte 08Ch
.byte 09Fh  ;end of right EQ for SW6
            ;************************************
.byte 000h  ;IMPORTANT-this line must be included
            ;***** line 1645 ***********************
.byte 008h  ;vol setting = MUTE
.byte 068h
.byte 004h
.byte 000h
```

```
.byte 000h
.byte 000h
.byte 000h
.byte 000h
.byte 000h
.byte 000h  ;IMPORTANT-this line must be included

            ;***** line 1655 ************************
            ;data for unmute of mix1
.byte 005h  ;mix1 = -15 dB
.byte 068h
.byte 007h
.byte 002h
.byte 0D8h
.byte 062h
.byte 000h  ;IMPORTANT-this line must be included
            ;***** line 1662 ************************
            ;data for unmute of mix1
.byte 005h  ;mix1 = -15 dB
.byte 068h
.byte 007h
.byte 000h
.byte 000h
.byte 000h
.byte 000h  ;IMPORTANT-this line must be included
            ;***** line 1669 ************************
            ;beginning of data to be sent at start up
.byte 003h  ;hex number of bytes before next stop cmd
.byte 068h
.byte 001h
.byte 069h  ;(normal mode I2S 18 bit)
.byte 011h  ;hex number of bytes before next stop cmd
.byte 068h
.byte 00Ah
.byte 010h
.byte 000h
.byte 000h
.byte 000h
.byte 000h
.byte 000h
.byte 000h
.byte 000h
.byte 000h
.byte 000h
.byte 000h
.byte 000h
.byte 000h
.byte 000h
.byte 000h
.byte 000h
.byte 011h  ;hex number of bytes before next stop cmd
.byte 068h
.byte 00Bh
```

```
.byte 010h
.byte 000h
.byte 000h
.byte 000h
.byte 000h
.byte 000h
.byte 000h
.byte 000h
.byte 000h
.byte 000h
.byte 000h
.byte 000h
.byte 000h
.byte 000h
.byte 000h

.byte 011h  ;hex number of bytes before next stop cmd
.byte 068h
.byte 00Ch
.byte 010h
.byte 000h
.byte 000h
.byte 000h
.byte 000h
.byte 000h
.byte 000h
.byte 000h
.byte 000h
.byte 000h
.byte 000h
.byte 000h
.byte 000h
.byte 000h
.byte 000h
.byte 000h

.byte 011h  ;hex number of bytes before next stop cmd
.byte 068h
.byte 00Dh
.byte 010h
.byte 000h
.byte 000h
.byte 000h
.byte 000h
.byte 000h
.byte 000h
.byte 000h
.byte 000h
.byte 000h
.byte 000h
.byte 000h
.byte 000h
.byte 000h
.byte 000h
.byte 000h
```

```
.byte 000h

.byte 011h  ;hex number of bytes before next stop cmd
.byte 068h
.byte 00Eh
.byte 003h
.byte 085h
.byte 0EAh
.byte 007h
.byte 00Bh
.byte 0D5h
.byte 003h
.byte 085h
.byte 0EAh
.byte 0FBh
.byte 014h
.byte 036h
.byte 003h
.byte 003h
.byte 076h

.byte 011h  ;hex number of bytes before next stop cmd
.byte 068h
.byte 00Fh
.byte 00Fh
.byte 0DAh
.byte 043h
.byte 0E0h
.byte 04Bh
.byte 07Ah
.byte 00Fh
.byte 0DAh
.byte 043h
.byte 0E0h
.byte 04Bh
.byte 0D3h
.byte 00Fh
.byte 0B4h
.byte 0DFh  ;end of left eq’s

.byte 011h  ;hex number of bytes before next stop cmd
.byte 068h
.byte 013h
.byte 010h
.byte 000h
.byte 000h
.byte 000h
.byte 000h
.byte 000h
.byte 000h
.byte 000h
.byte 000h
```

```
.byte 000h
.byte 000h
.byte 000h
.byte 000h
.byte 000h
.byte 000h

.byte 011h  ;hex number of bytes before next stop cmd
.byte 068h
.byte 014h
.byte 010h
.byte 000h
.byte 000h
.byte 000h
.byte 000h
.byte 000h
.byte 000h
.byte 000h
.byte 000h
.byte 000h
.byte 000h
.byte 000h
.byte 000h
.byte 000h
.byte 000h
.byte 011h  ;hex number of bytes before next stop cmd
.byte 068h
.byte 015h
.byte 010h
.byte 000h
.byte 000h
.byte 000h
.byte 000h
.byte 000h
.byte 000h
.byte 000h
.byte 000h
.byte 000h
.byte 000h
.byte 000h
.byte 000h
.byte 000h
.byte 000h
.byte 000h
.byte 011h  ;hex number of bytes before next stop cmd
.byte 068h
.byte 016h
.byte 010h
.byte 000h
.byte 000h
.byte 000h
.byte 000h
.byte 000h
```

```
.byte 000h
.byte 000h
.byte 000h
.byte 000h
.byte 000h
.byte 000h
.byte 000h
.byte 000h
.byte 000h
.byte 011h  ;hex number of bytes before next stop cmd
.byte 068h
.byte 017h
.byte 003h
.byte 085h
.byte 0EAh
.byte 007h
.byte 00Bh
.byte 0D5h
.byte 003h
.byte 085h
.byte 0EAh
.byte 0FBh
.byte 014h
.byte 036h
.byte 003h
.byte 003h
.byte 076h
.byte 011h  ;hex number of bytes before next stop cmd
.byte 068h
.byte 018h
.byte 00Fh
.byte 0DAh
.byte 043h
.byte 0E0h
.byte 04Bh
.byte 07Ah
.byte 00Fh
.byte 0DAh
.byte 043h
.byte 0E0h
.byte 04Bh
.byte 0D3h
.byte 00Fh
.byte 0B4h
.byte 0DFh  ;end of right eq’s
.byte 003h  ;treble = 0 dB
.byte 068h
.byte 005h
.byte 072h
.byte 003h  ;bass = 0 dB
.byte 068h
.byte 006h
```

```
          .byte 03Eh
          .byte 008h   ;volume = -18 dB
          .byte 068h
          .byte 004h
          .byte 000h
          .byte 020h
          .byte 03Ah
          .byte 000h
          .byte 020h
          .byte 03Ah
          .byte 005h   ;mix1 = -15 dB
          .byte 068h
          .byte 007h
          .byte 002h
          .byte 0D8h
          .byte 062h
          .byte 005h   ;mix2 = mute
          .byte 068h
          .byte 008h
          .byte 000h
          .byte 000h
          .byte 000h

          .byte 000h   ;IMPORTANT - this ”000h” indicates end of
                       ;data table it MUST be included for program
                       ;to fuction correctly
;***************************************************************************
;  Interrupt vectors
;***************************************************************************
        .even ; Following section must be evenly aligned
        .sect ”Int_Vect”,I_vectors-31
        .word RESET              ; no source
        .word RESET              ; no source
        .word SW_ISR             ; P1.x
        .word SW_ISR             ; P2.x
        .word RESET              ; no source
        .word RESET              ; no source
        .word RESET              ; no source
        .word RESET              ; no source
        .word RESET              ; Timer_AX
        .word RESET              ; Timer_A0
        .word RESET              ; Watchdog/Timer, Timer mode
        .word RESET              ; no source
        .word RESET              ; no source
        .word RESET              ; no source
        .word RESET              ; NMI, Osc. fault
        .word RESET              ; POR, ext. Reset, Watchdog
```

Appendix A

Schematics

This appendix includes a schematic diagram of the TAS3001 EVM.