Evaluation Board for the AD74111 Mono Codec

EVAL-AD74111EB

FEATURES

Full-featured evaluation board for the AD74111
Buffered and filtered analog input and output
On-board power supply regulation

INTRODUCTION

This data sheet describes the evaluation board for the AD74111 audio codec. Full data on the codec is available in the AD74111 data sheet and should be consulted in conjunction with this data sheet when using the evaluation board.

The evaluation board includes an AD74111 mono codec, and additional analog circuitry is provided to buffer and filter the input and output audio signals. The digital section contains an on-board reset circuit, 12.288 MHz MCLK oscillator, and an edge connector that connects to the digital interface pins of the AD74111. Multiple link options are also available to facilitate testing different modes of operation.

FUNCTIONAL BLOCK DIAGRAM

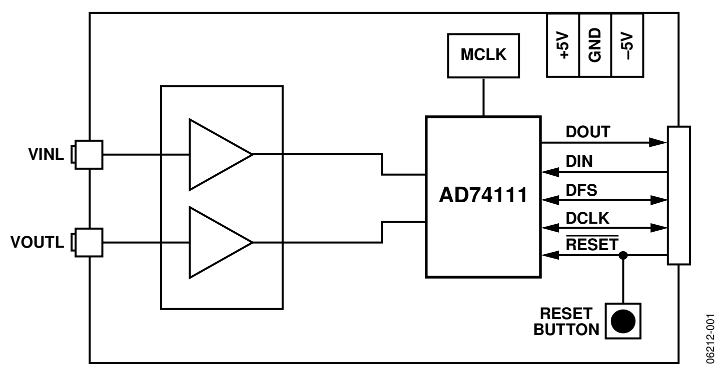


Figure 1.

One Technology Way, P.O. Box 9106, Norwood, MA 02062-9106, U.S.A.
Tel: 781.329.4700 **www.analog.com**
Fax: 781.461.3113

TABLE OF CONTENTS

REVISION HISTORY

12/06—Revision 0: Initial Version

EVALUATION BOARD HARDWARE

OPERATING THE AD741111 EVALUATION BOARD

Power Supplies

The evaluation board requires a supply of +5 V, GND, and −5 V. Voltage regulators on the evaluation board regulate these supplies to produce an analog and digital 2.5 V and digital 3 V for the codec. The ±5 V supplies are used to power the op amps.

All the supplies are decoupled with 10 μF tantalum and 0.1 μF ceramic capacitors.

Extensive ground planes are used on this board to minimize the effect of high frequency noise interference. There are two ground planes, AGND and DGND. These are connected at one location close to the AD74111.

The AD74111 evaluation board uses shorted links to allow additional testing, such as current measurement or the use of separate supplies. The links are not populated on the boards, and if it is required to do, for example, a supply current measurement, the PCB track underneath the link can be cut and an ammeter placed between the two pads to complete the circuit. When the measurement is no longer required, a shorted link can be used to replace the broken track. This is shown in Figure 2, Figure 3, and Figure 4.

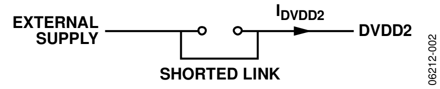


Figure 2. Default Link Setting

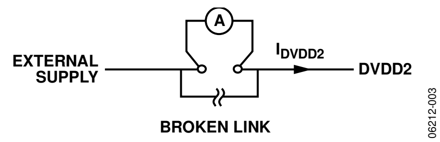


Figure 3. Board Modified for Current Measurement

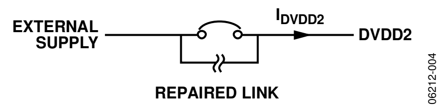


Figure 4. Link Used to Bypass Cut Track

LINK SETTINGS

The AD74111 evaluation board has a number of link options that configure the board for various modes of operation. The links are listed in Table 1 along with their default conditions and a description of their function. Note that some links are not populated by default.

Table 1.

Link	Default Position	Description
LK1	Not Populated	This link can be used to separate the AVDD supply from the digital power supply regulators.
LK2	Not Populated	This link can be used to separate the DVDD1 that is used for the interface circuitry from the DVDD1 that the AD74111 codec uses.
LK3	Not Populated	This link can be used to separate DVDD2 from the 2.5 V regulator, U3.
LK4	Not Populated	This link can be used to separate AVDD1 from the 2.5 V regulator, U2.
LK5	B	This link selects between the on-board crystal oscillator and an external MCLK signal.
LK6	B	This link enables or disables the MCLK divider of U7-B.
LK7	IN	This link uses MCLK/2 as the reference clock source for DCLK.
LK8	B	This link selects whether the AD74111 powers up as a master device (Position B) or as a slave device (Position A).
LK9, LK12	Not Populated	These links can be used to bypass the capacitive decoupling of the input signals.
LK10, LK13	Not Populated	These links can be used to bypass the op-amp stage of the analog input sections.
LK11	A	This link selects the source of the reset signal for the AD74111.
LK14	Not Populated	This link can be used if a bias voltage other than REFCAP is required for the op amp stages.

EVALUATION BOARD SOFTWARE

DSP EXAMPLES

The following text gives some code examples showing methods of interfacing the AD74111 to Analog Devices, Inc.'s fixed-point DSPs. The flowcharts are generic and should be adaptable to any type of DSP, but the code samples are specifically for the ADSP-218x DSPs.

INTERRUPTS

The code examples that follow make use of the serial port (SPORT) transmit and receive interrupts that allow the processor to service the codec only when required. The transmit section of the ADSP-218x SPORT consists of a transmit register, Tx0 for SPORT0 and Tx1 for SPORT1, and a serial shift register, Rx0 for SPORT0 and Rx1 for SPORT1. Transmission of a data-word is initiated by loading it to the appropriate Tx register. The DSP begins transmitting this word by copying it to the serial shift register where it is clocked out one bit at a time dependant on the SCLK rate. A transmit interrupt occurs when it is safe to load the Tx register with the next value to be transmitted. The DSP will not allow the second word to be transmitted until the first word is completely transmitted. A transmit interrupt occurs when the second bit of the data-word is transmitted.

A receive interrupt occurs when a complete data-word is received into the Receive Register Rx0 or the Receive Register Rx1. The receive section of the SPORT has a similar serial shift register to the transmit section. It is important that any data received in the Rx register be read as soon as possible because a new word can be read into the serial shift register and overwrite the Rx register once the entire word is received.

16-BIT MIXED MODE, 16-BIT DATA, MASTER MODE

The flowchart in Figure 6 is an example of how to interface the DSP to the AD74111 in master mode. In master mode, the codec generates the DFS and DCLK signals so the TFS1/RFS1 and SCLK1 are inputs to the DSP. The codec is operating in its default 16-bit mixed mode with the data-word length set to 16 bits. In this mode, the AD74111 generates two DFS pulses per sample interval, one for the control register/status information, and one for the DAC/ADC data.

These software examples take the ADC results and send them back to the DAC to create a loopback effect. The DAC output is therefore one sample interval behind the ADC input.

The software uses three buffers to control the flow of information:

1. The control buffer contains the control register words that need to be sent to the AD74111.
2. The Tx buffer contains a control register value and a value for the DAC.
3. The Rx buffer stores the status information and the ADC result.

Figure 5 shows how the buffers are arranged in memory.

When the program is running, the transmit interrupt reads the next value in the Tx buffer and transmit it when it receives a transmit interrupt. The receive interrupt does most of the work in this example. When a receive interrupt is generated, the program reads the value from the Rx1 register into the AY1 register immediately storing it for later use. It then decrements and checks a counter value to determine how many receive interrupts have been received in the current sample interval. If the value is not zero, the control word just finishes transmitting, and the status word is just received. The status information is not required in this example; therefore, the program returns from the interrupt subroutine and waits for the next interrupt event.

When the program determines that two receive interrupts have occurred (control counter = 0), it will get the next control word to be sent to the codec from control buffer and load it to the CTRL memory location in the Tx buffer. The ADC value that was just received is copied to the DAC memory location to create a loopback operation, and this value is transmitted to the DAC in the next sample interval. The program resets the control counter variable and returns from the interrupt subroutine to wait for the next interrupt event.

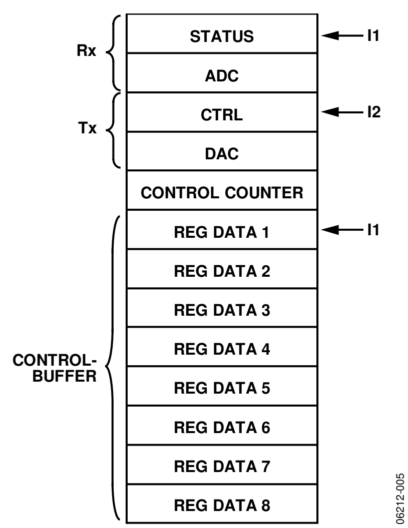


Figure 5. Buffer Memory Map

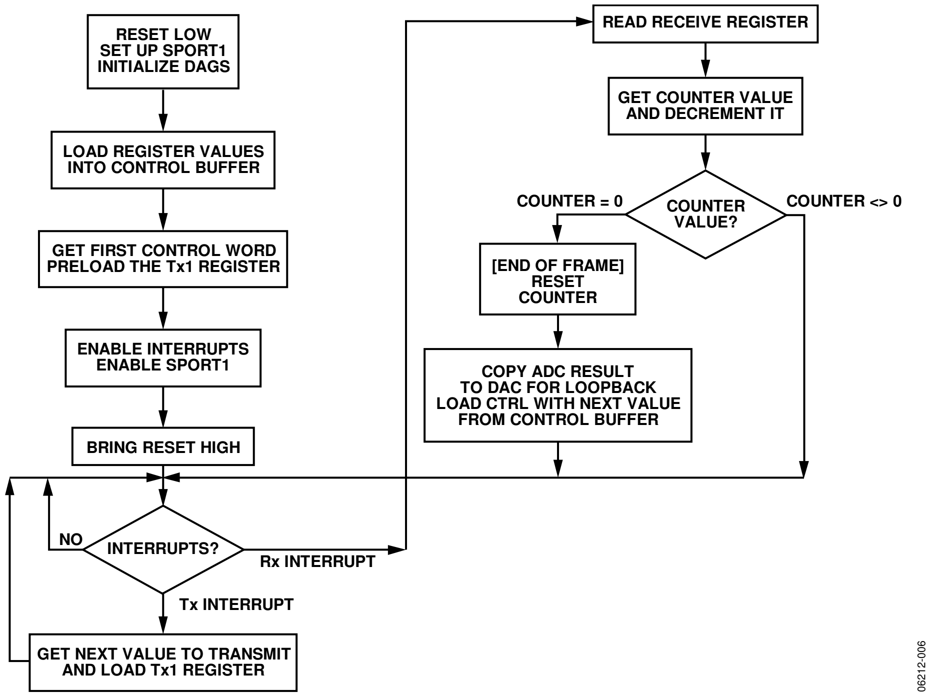


Figure 6. Mixed Mode Flowchart

16-Bit Mixed Mode, 16-Bit Data, Master Mode Programming Code

```
/ *
This program creates a loopback effect by copying the ADC result to the DAC.
The codec is used in 16-bit mixed-master mode.
* /
.section/data data1;
#define N 7 //           number of control words to program
#define CTRL             TX
#define DAC              TX+1
#define STATUS           RX
#define ADC              RX+1
.var/circ                TX[2];
.var/circ                RX[2];
.var/circ                CONTROLBUFFER[N];
.var CONTROLCOUNTER;
#include <header.h>
.section/pm program;
ISR_RESET:               jump START;RTI; RTI; RTI;
ISR_IRQ2:                RTI;RTI; RTI; RTI;
ISR_IRQ1:                RTI;RTI; RTI; RTI;
ISR_IRQ0:                RTI;RTI; RTI; RTI;
ISR_S0_TRANSMIT:         RTI;RTI; RTI; RTI;
ISR_S0_RECEIVE:          RTI;RTI; RTI; RTI;
ISR_IRQE:                RTI;RTI; RTI; RTI;
ISR_BDMA:                RTI;RTI; RTI; RTI;
ISR_S1_TRANSMIT:         JUMP TXINT;RTI; RTI; RTI;
ISR_S1_RECEIVE:          JUMP RXINT;RTI; RTI; RTI;
ISR_TIMER:               RTI;RTI; RTI; RTI;
ISR_POWERDOWN:           RTI;RTI; RTI; RTI;
```

```
START:      //program starts here
            reset fl1; //reset low until everything is set up
            //set up the serial port (SPORT1)
            //rfs, tfs & sclk are inputs
            //frame syncs required
            ax0=0x280f;
            dm(SPORT1_CTRL_REG)=ax0;
            imask=0; ifc=0xff;

            //initialise DAGs
            i0=CONTROLBUFFER; m0=1; l0=N;
            i1=CTRL; m1=1; l1=2;
            i2=STATUS; m2=1; l2=2;

            //fill control buffer with control register values
            ax0=0x807c; dm(i0,m0)=ax0;      //cra
            ax0=0x8a00; dm(i0,m0)=ax0;      //crb
            ax0=0x9000; dm(i0,m0)=ax0;      //crc
            ax0=0x9809; dm(i0,m0)=ax0;      //crd NOTE Mixed Master Mode
            ax0=0xa000; dm(i0,m0)=ax0;      //cre
            ax0=0xa800; dm(i0,m0)=ax0;      //crg
            ax0=0x3000; dm(i0,m0)=ax0;      //crg read

            ax0=2; dm(CONTROLCOUNTER)=ax0;

            //reinitialise DAG
            i0=CONTROLBUFFER;
            //preload the TX register
            ax0=0x807c;
            tx1=ax0;
            //bring reset high and wait for interrupts
            call MASTERRESET;
            //set fl1;
            //enable the SPORT
            ax0=0x1c00; dm(SYS_CTRL_REG)=ax0;
            //enable RX and TX interrupts
            imask= 0x06; ifc=0xff;
WAIT1:      jump WAIT1;

TXINT:      ax0=dm(i1,m1);            //get next value to transmit
            tx1=ax0;                  //transmit it
            rti;                      //done
RXINT:

            //read the RX register straight away
            ay1=rx1;
            ax0=dm(CONTROLCOUNTER);
            ar=ax0-1;
            if eq jump ZERO;

ONE:        //status word received
            dm(CONTROLCOUNTER)=ar;          //store counter value
            //do whatever with status word
            rti;

ZERO:       //ADC Word Received - End Of Frame
            //loopback operation
            dm(ADC)=ay1;
            dm(DAC)=ay1;
            //get next control word
            ax0=dm(i0,m0); dm(CTRL)=ax0;
            //reset the counter
            ax0=2; dm(CONTROLCOUNTER)=ax0;
            rti;
```

```
MASTERRESET:
            //this subroutine keeps DIN low during reset for Master Mode
            ax0=0x0800; dm(SYS_CTRL_REG)=ax0;
            reset FLAG_OUT;
            cntr=200; do KT1 until ce;
KT1:        nop;        //kill time
            set fl1;    //reset high
            cntr=200; do KT2 until ce;
KT2:        nop;        //kill time
            rts;        //return
```

16-BIT DATA MODE, 16-BIT DATA, MASTER MODE

This example demonstrates how to program the AD74111 in mixed mode and then switch to data mode when the programming is completed. The program uses a data buffer to store the control words that are sent to the AD74111 and the initial values that are sent to the DAC. The DAC is programmed with a midscale value until the AD74111 is put into data mode. At that point, the ADC results are sent to the DAC to create a loopback effect. Because the number of control words and DAC data-words is known, the program checks the value of Address Pointer I0 at each received interrupt. If the value is less than the end of the buffer, then a control or DAC value is taken from the control buffer and transmitted to the AD74111. Address Pointer I0 is automatically incremented by this procedure. If the address pointer is equal to the last address in the buffer, the program reads the ADC result that was just received and sends it to the DAC to create a loopback effect. Because no value was read from the control buffer, the address pointer is not incremented, and it will equal the last address of the control buffer for subsequent interrupts. Figure 7 shows the flowchart for this program.

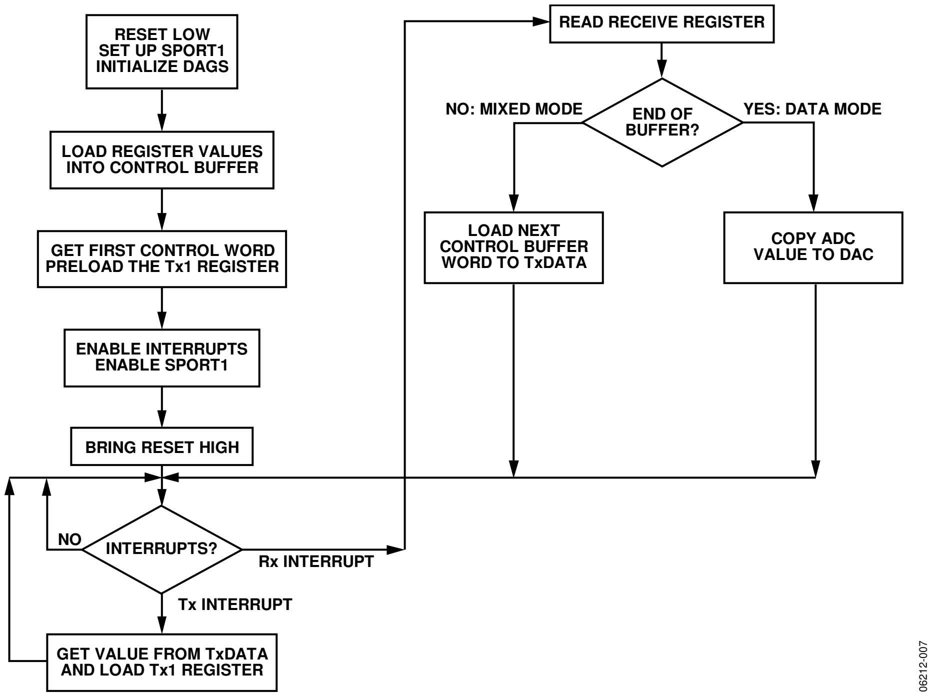


Figure 7. Mixed Mode Flowchart

16-Bit Data Mode, 16-Bit Data, Master Mode Programming Code

```
/ *
This program creates a loopback effect by copying the ADC result to the DAC.
Once the codec has been programmed, it is put into data mode.
The codec is used in 16-bit mixed-master mode.
* /
.section/data data1;
#define N 18        //number of words transmitted before going into data mode
#define             CONTROLEND CONTROLBUFFER + N-1
.var                TXDATA;
.var                RXDATA;
.var/circ           CONTROLBUFFER[N];

#include <header.h>
.section/pm program;
ISR_RESET:          jump START; RTI; RTI; RTI;
ISR_IRQ2:           RTI; RTI; RTI; RTI;
ISR_IRQ1:           RTI; RTI; RTI; RTI;
ISR_IRQ0:           RTI; RTI; RTI; RTI;
ISR_S0_TRANSMIT:    RTI; RTI; RTI; RTI;
ISR_S0_RECEIVE:     RTI; RTI; RTI; RTI;
ISR_IRQE:           RTI; RTI; RTI; RTI;
ISR_BDMA:           RTI; RTI; RTI; RTI;
ISR_S1_TRANSMIT:    JUMP TXINT; RTI; RTI; RTI;
ISR_S1_RECEIVE:     JUMP RXINT; RTI; RTI; RTI;
ISR_TIMER:          RTI; RTI; RTI; RTI;
ISR_POWERDOWN:      RTI; RTI; RTI; RTI;
```

```
START:       //program starts here
             reset fl1;                //reset low until everything is set up
             //set up the serial port (SPORT1)
             //rfs, tfs & sclk are inputs
             //frame syncs required
             ax0=0x280f;
             dm(SPORT1_CTRL_REG)=ax0;
             imask=0; ifc=0xff;
             //initialise DAGs
             i0=CONTROLBUFFER; m0=1; l0=N;
             //fill control buffer with control register values
             ax0=0x807c; dm(i0,m0)=ax0;      //cra
             ax0=0x0000; dm(i0,m0)=ax0;      //mid-scale DAC value
             ax0=0x807c; dm(i0,m0)=ax0;      //cra
             ax0=0x0000; dm(i0,m0)=ax0;      //mid-scale DAC value
             ax0=0x807c; dm(i0,m0)=ax0;      //cra
             ax0=0x0000; dm(i0,m0)=ax0;      //mid-scale DAC value
             ax0=0x807c; dm(i0,m0)=ax0;      //cra
             ax0=0x0000; dm(i0,m0)=ax0;      //mid-scale DAC value
             ax0=0x8a00; dm(i0,m0)=ax0;      //crb
             ax0=0x0000; dm(i0,m0)=ax0;      //mid-scale DAC value
             ax0=0x9000; dm(i0,m0)=ax0;      //crc
             ax0=0x0000; dm(i0,m0)=ax0;      //mid-scale DAC value
             ax0=0xa000; dm(i0,m0)=ax0;      //cre
             ax0=0x0000; dm(i0,m0)=ax0;      //mid-scale DAC value
             ax0=0xa800; dm(i0,m0)=ax0;      //crg
             ax0=0x0000; dm(i0,m0)=ax0;      //mid-scale DAC value
             ax0=0x9801; dm(i0,m0)=ax0;      //crd NOTE Data Master Mode
             ax0=0x0000; dm(i0,m0)=ax0;      //mid-scale DAC value
             //reinitialise DAG
             i0=CONTROLBUFFER;
             //preload the TX register
             ax0=dm(i0,m0);
             tx1=ax0;
             ax0=dm(i0,m0);
             dm(TXDATA)=ax0;
             //bring reset high and wait for interrupts
             call MASTERRESET;
             //enable the SPORT
             ax0=0x1c00; dm(SYS_CTRL_REG)=ax0;
             //enable RX and TX interrupts
             imask= 0x06; ifc=0xff;
WAIT1:       jump WAIT1;
TXINT:       ax0=dm(TXDATA);     //get next value to transmit
             tx1=ax0;            //transmit it
             rti;                //done
RXINT:       ay0=rx1;                  //get received value
ax0=CONTROLEND;
ay1=i0;
ar=ax0-ay1;
if eq jump DATAMODE;
//still in mixed mode
ax0=dm(i0,m0);
dm(TXDATA)=ax0;
toggle fl0;
rti;
```

```
DATAMODE:    //in data mode
             dm(TXDATA)=ay0; //loopback
             dm(RXDATA)=ay0;
             rti;
MASTERRESET:       //this subroutine keeps DIN low during reset for Master Mode
             ax0=0x0800; dm(SYS_CTRL_REG)=ax0;
             reset FLAG_OUT;
             cntr=200; do KT1 until ce;
KT1:         nop;         //kill time
             set fl1;     //reset high
             cntr=200; do KT2 until ce;
KT2:         nop;         //kill time
             rts;         //return
```

SCHEMATICS AND BILL OF MATERIALS

SCHEMATICS

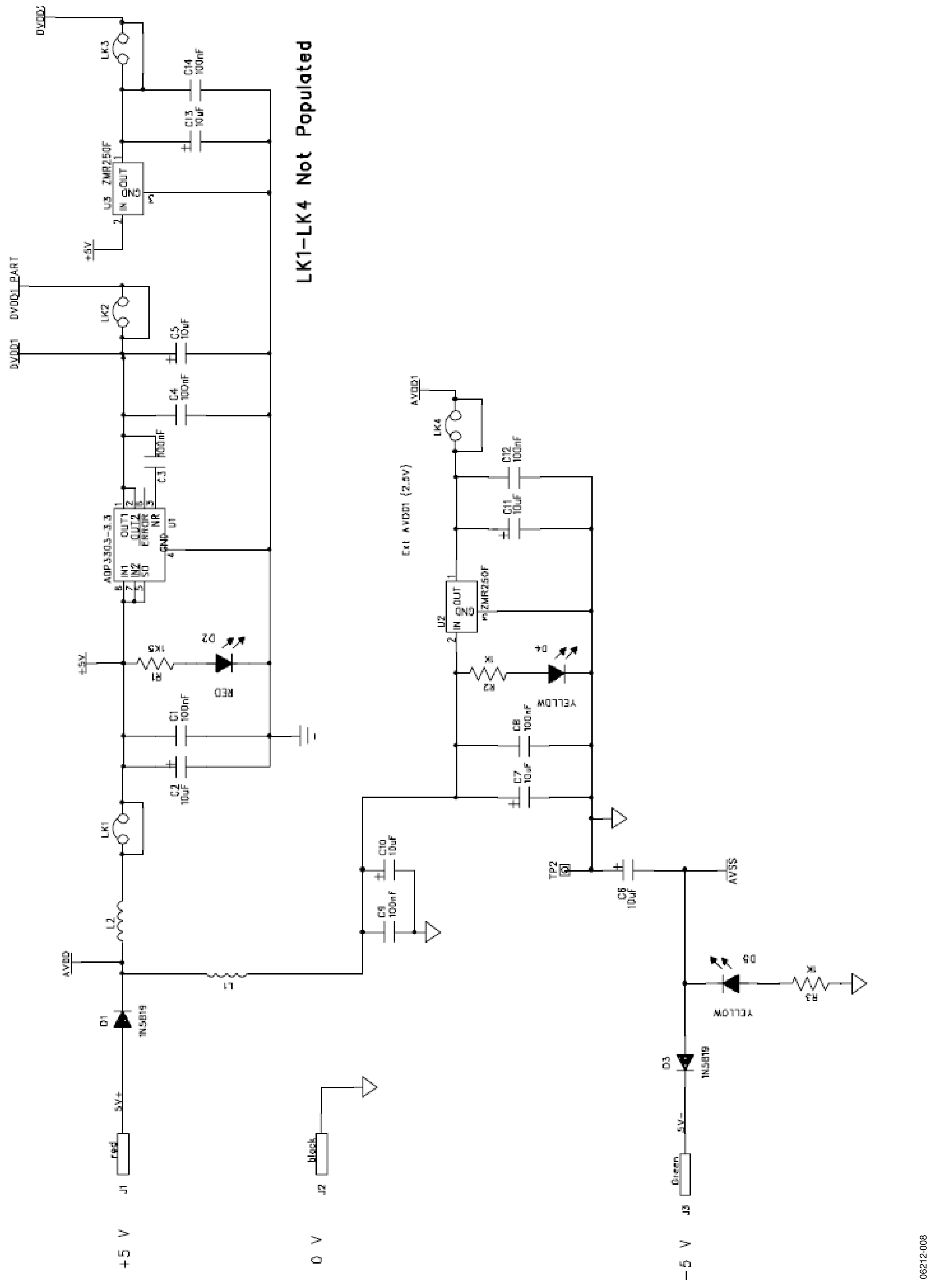


Figure 8. Evaluation Board Schematic (1 of 3)

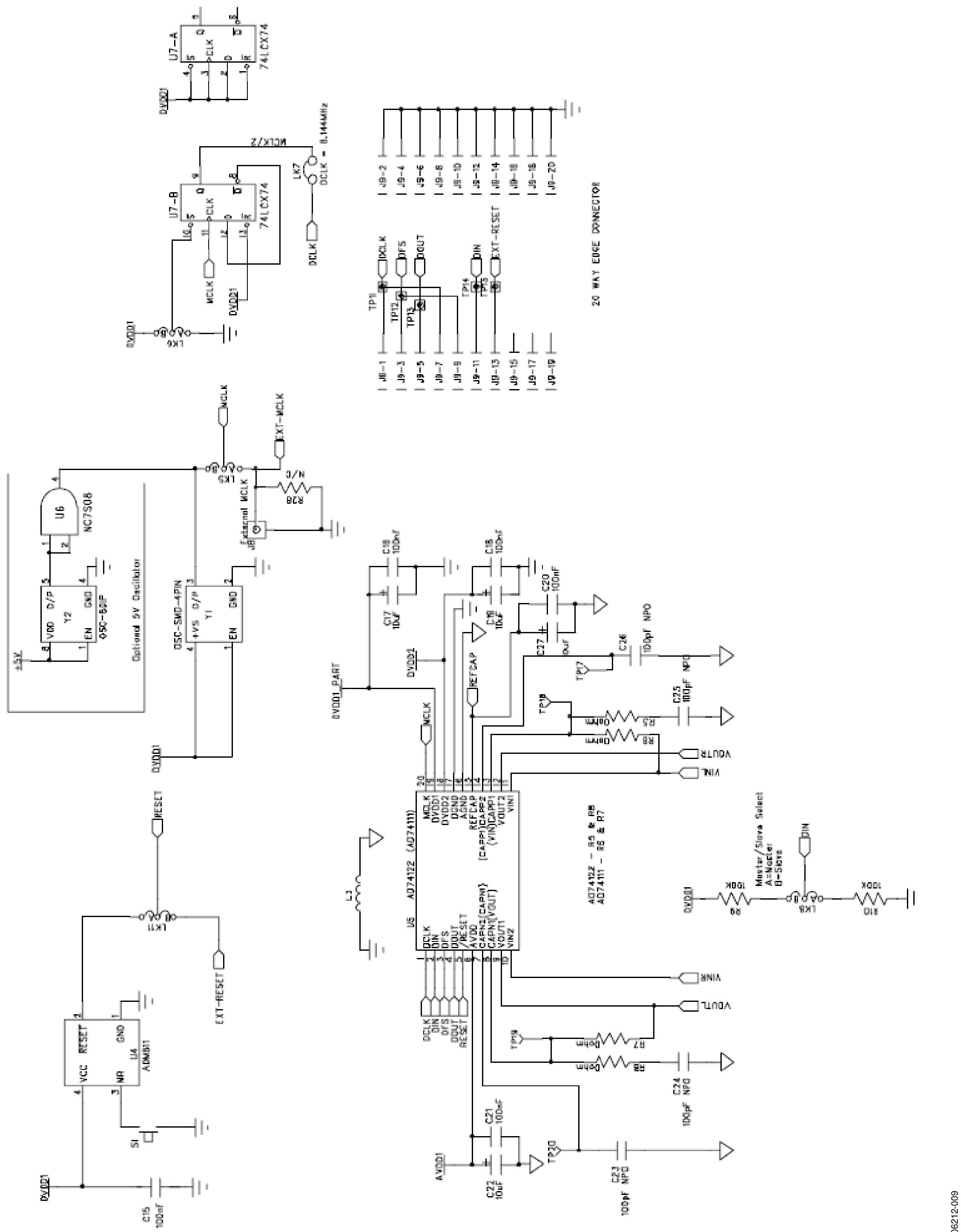
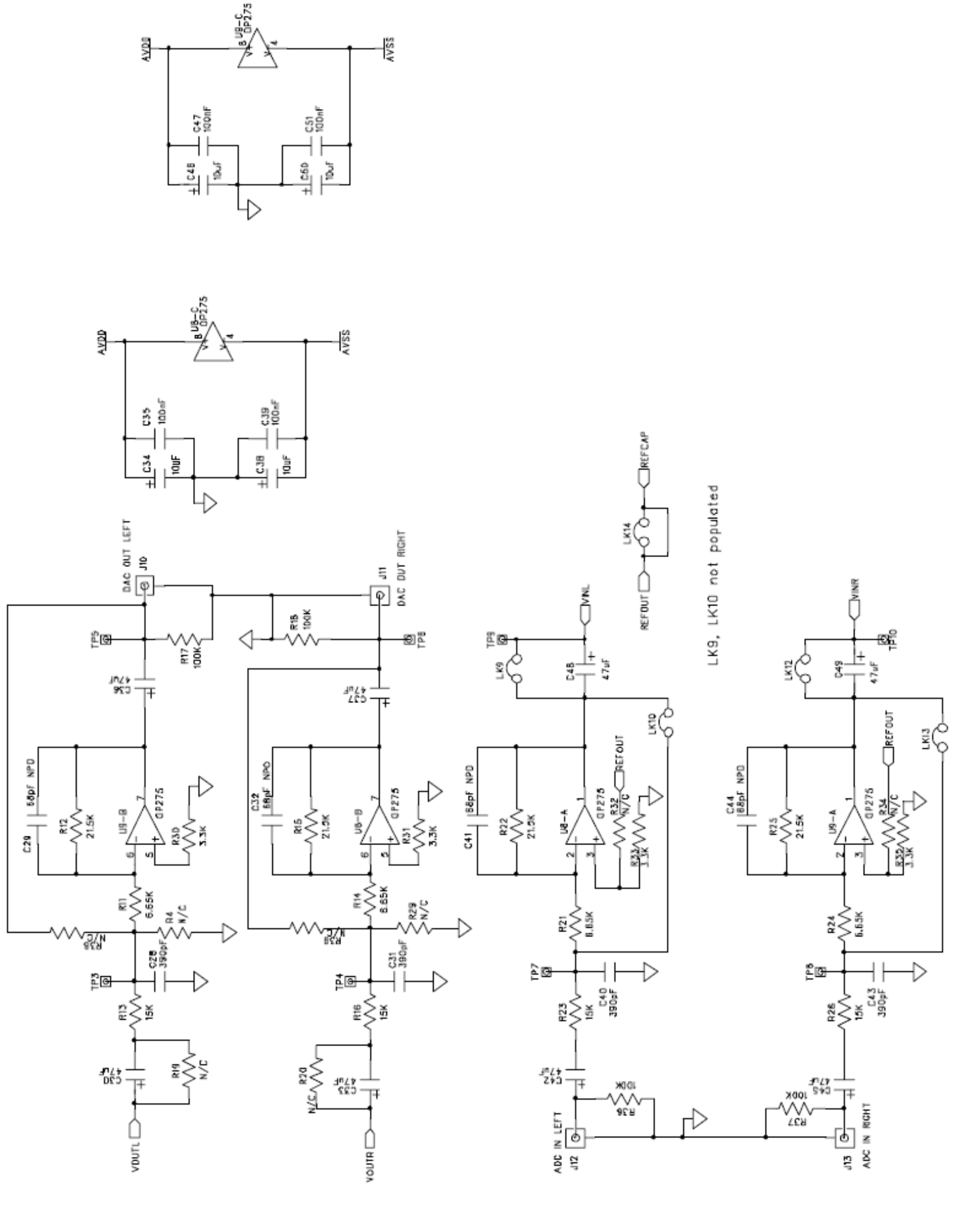


Figure 9. Evaluation Board Schematic (2 of 3)



06212-010

Figure 10. Evaluation Board Schematic (3 of 3)

BILL OF MATERIALS

Table 2.

Reference Designator	Type	Value
C1, C3, C4, C8, C9, C12, C14 to C16, C18, C20, C21, C35, C39, C47, C51	Capacitor	100 nF
C2, C5 to C7, C10, C11, C13, C17, C19, C22, C27, C34, C38, C46, C50	Capacitor	10 µF, 10 V
C23 to C26	Capacitor	100 pF, NPO
C28, C31, C40, C43	Capacitor	390 pF
C29, C32, C41, C44	Capacitor	68 pF, NPO
C30, C33, C36, C37, C42, C45, C48, C49	Capacitor + ELEK	47 µF, 35 V
D1, D3	DIODE-1N5819	
D2	LED	RED
D4, D5	LED	YELLOW
J1	Banana	RED
J2	Banana	BLACK
J3	Banana	GREEN
J8	SMA	
J9	HEADER20	
J10 to J13	Phono	
L1 to L3	Bead	600 Ω @ 100 MHz
LK1 to LK4, LK7, LK9, LK10, LK12 to LK14	Jumper	
LK5 to LK6, LK8, LK11	JUMPER2\SIP3	
R1	Resistor	1.5 kΩ
R2, R3	Resistor	1 kΩ
R4, R19, R20, R28, R29, R32, R34, R38	Resistor	N/C
R5, R6, R7, R8	Resistor	0 Ω
R9, R10, R17, R18, R36, R37	Resistor	100 kΩ
R11, R14, R21, R24	Resistor	6.65 kΩ
R12, R15, R22, R25	Resistor	21.5 kΩ
R13, R16, R23, R26	Resistor	15 kΩ
R30, R31, R33, R35	Resistor	3.3 kΩ
S1	SW-PUSH-SMD	
TP2 to TP15	Wire Wrap	
TP17 to TP20	Test Point	
U1	ADP3303-3.3	
U2, U3	ZMR250F	
U4	ADM811	
U5	AD74122	
U6	NC7S08	
U7	74LVX74	
U8, U9	OP275	
Y1	OSC-SMD-4PIN	12.288 MHz
Y2	OSC-8DIP	12.288 MHz

ORDERING INFORMATION

ORDERING GUIDE

Model	Package Description
EVAL-AD74111EB	Evaluation Board

ESD CAUTION

ESD (electrostatic discharge) sensitive device. Charged devices and circuit boards can discharge without detection. Although this product features patented or proprietary protection circuitry, damage may occur on devices subjected to high energy ESD. Therefore, proper ESD precautions should be taken to avoid performance degradation or loss of functionality.

NOTES