

TMS320C1x Evaluation Module Analog Interface

Application Report



TMS320C1x Evaluation Module Analog Interface Application Report

Product Application

Chaucer Kuo
Digital Signal Processing Products — Semiconductor Group
Texas Instruments

SPRA029
January 1993



important notice

Texas Instruments Incorporated (TI) reserves the right to make changes to its products or to discontinue any semiconductor product or service without notice, and advises its customers to obtain the latest version of relevant information to verify, before placing orders, that the information being relied on is current.

TI warrants performance of its semiconductor products and related software to current specifications 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.

Please be aware that TI products are not intended for use in life-support appliances, devices, or systems. Use of TI product in such applications requires the written approval of the appropriate TI officer. Certain applications using semiconductor devices may involve potential risks of personal injury, property damage, or loss of life. In order to minimize these risks, adequate design and operating safeguards should be provided by the customer to minimize inherent or procedural hazards. Inclusion of TI products in such applications is understood to be fully at the risk of the customer using TI devices or systems.

TI assumes no liability for applications assistance, customer product design, software performance, or infringement of patents or services described herein. Nor does TI 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 semiconductor products or services might be or are used.

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

<i>Title</i>	<i>Page</i>
1 Introduction	1
2 Hardware Description	1
3 Analog Interface Circuit (AIC)	1
4 Serial-to-Parallel Conversion	3
5 Communication With the EVM	3
6 Hardware Interface	3
7 Power Requirements	6
8 Software Description	7
9 Initializing the TMS320C16 EVM	7
10 Communication With the AIC Board	7
11 TLC32040 Secondary Communication	7
12 Summary	7

Appendices

<i>Title</i>	<i>Page</i>
13 Appendix A — TMS320C16 EVM PAL Equations	9
14 Appendix B — Software Flowcharts for AIC Initialization and Communication	10
15 Appendix C — Software Listings	13
16 Appendix D — Parts List for the Analog Interface Board (AIB)	26

List of Illustrations

<i>Figure</i>	<i>Title</i>	<i>Page</i>
1	The Two Stages of AIC Input Circuitry	2
2	Serial-to-Parallel Conversion Circuit (Synchronous Mode)	4
3	PAL Pinouts	9
4	Main Routine Flowchart	10
5	Primary Interrupt Routine	11
6	Secondary Communications 1	11
7	Secondary Communications 2	12

List of Tables

<i>Table</i>	<i>Title</i>	<i>Page</i>
1	EVM Connector Pinouts	5
2	TMS320C16 EVM Switch Settings for I/O Address	6

Software Listings

<i>Example</i>	<i>Title</i>	<i>Page</i>
1	Initializing Interrupt Source Code	13
2	Main Program With FIR Routine	13
3	Main Program With IIR Routine	17
4	Linker Command File (FIR)	20
5	FIR Subroutine	21
6	FIR Coefficients File	22
7	IIR Subroutine	24
8	IIR Coefficients File	25

Introduction

The TMS320C16 Evaluation Module (EVM) is a low-cost PC-AT plug-in board that can be used to evaluate performance characteristics of TMS320C1x digital signal processors (DSPs). The TMS320C1x EVM contains the following:

- PC Interface
- TMS320C1x DSP
- assembler/linker
- debugger software

This application report describes how to build a supplemental analog interface board (AIB) that provides the analog capabilities commonly used by a DSP chip. The AIB is an analog-to-digital/digital-to-analog conversion board that can be used with the TMS320C1x EVM as a code development target system. The schematic, parts list, cable connects, power requirements, and software necessary to build this board are all discussed in this document. You can build your own board or use this information to convert an existing AIB to the TMS320C16 EVM.

Hardware Description

Analog Interface Circuit (AIC)

The conversion component integral to the AIB is the analog interface circuit (AIC). For this application, the AIC is the Texas Instruments TLC32040, which provides a single-channel input/output voice-quality analog interface. AIC features include D/A and A/D conversions with 14 bits of dynamic range, variable DAC and ADC sampling rate, and up to 19.2-kHz sampling rate filtering.

The master input clock to the AIC is provided by a 5.184-MHz crystal oscillator. The AIC is hard wired for 16-bit word mode operation.

The AIC accepts analog input and produces analog output within specific fixed voltage and power ranges only. The analog circuitry allows the system to process the type of signals that are often encountered in general-purpose signal processing.

The AIC input circuitry comprises two stages of TL072 operational amplifiers as shown in Figure 1.

1. Stage one (U6) is configured as a noninverting, high-input impedance, unity gain amplifier. Its purpose is to buffer the input signal from a microphone or some similar input source without drawing any appreciable load from the source. Its output is connected to the AIC **IN+** (pin 24) input.
2. Stage two provides gain for the input signal. A nominal gain of ten supports low-voltage signals common on most microphones. Its output is connected to the AIC **AUX IN+** (pin 24) input.

The AIC's output is designed to drive a minimum load impedance of 300 ohms. This enables it to drive telecommunications circuitry, which commonly exhibits impedances of 600 ohms. In contrast, most speakers exhibit impedances of between four and sixteen ohms.

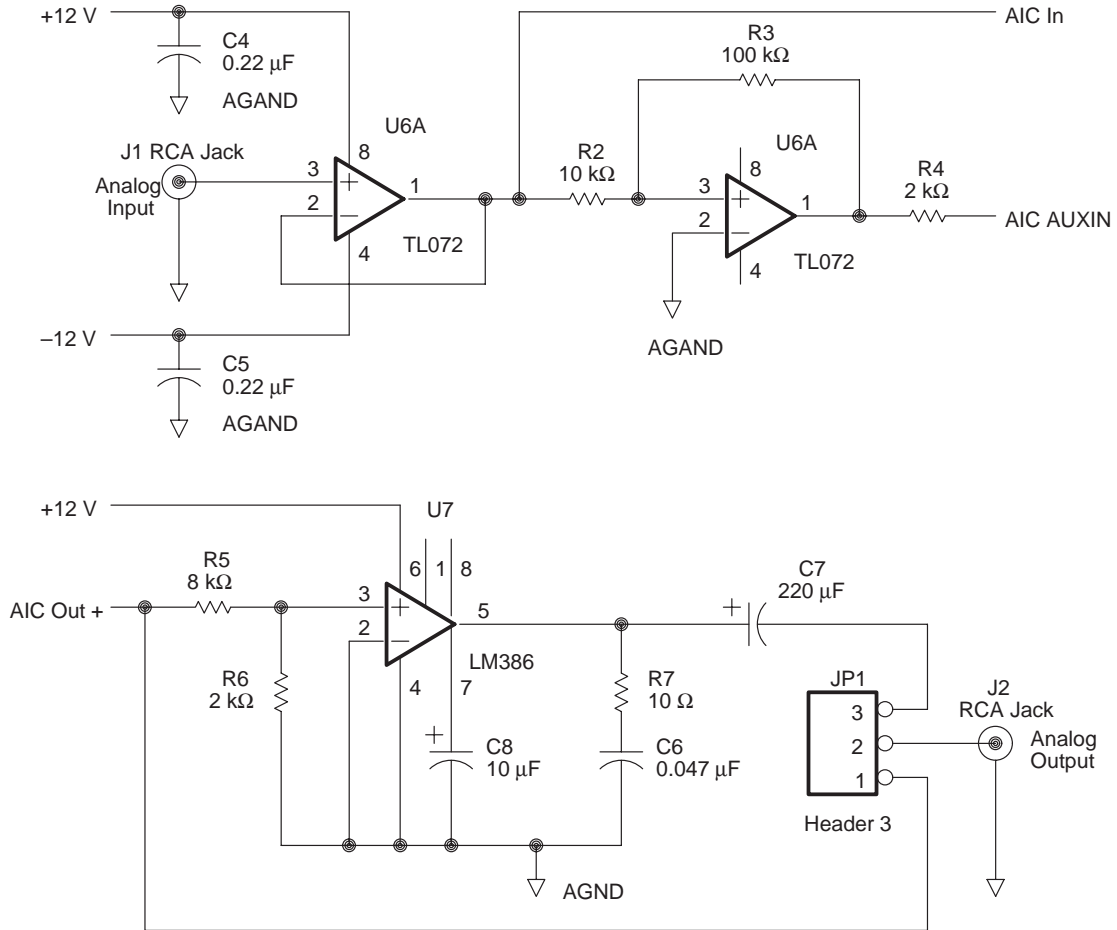
Accordingly, an LM386 audio power amplifier (U7) is used to drive speaker impedances and provide power gain. Because the LM386 has an inherent gain of 20, a voltage divider is used at the AIC's output to avoid overdriving the input to this amplifier.

With a two-to-ten voltage divider, the gain of the output amplifier circuit is four. An AIC output voltage of 1.5 volts peak to peak corresponds to an amplifier output of approximately 6 volts peak to peak (unclipped) when driving an 8-ohm load impedance. Refer to the LM386 data sheet for detailed information on its output characteristics.

A JP1 jumper allows you to select between AIC direct output and LM386 output.

The TMS320C16 EVM with the TLC32040 analog interface capabilities are suited to many applications, including audio data processing. Most speakers and microphones connect directly to the AIC analog input (IN+) and output (OUT+). An inexpensive microphone, amplifier, and speakers are sufficient; however, higher quality audio is achieved by using a better grade of stereo amplifier (with a volume control) and speakers.

Figure 1. The Two Stages of AIC Input Circuitry



Serial-to-Parallel Conversion

The TLC32040 AIC interface to the EVM requires a small amount of glue logic because the EVM has no serial port. The circuit accomplishes the serial-to-parallel conversion, operating in synchronous mode, as shown in Figure 2.

A PAL16L8–10 programmable array logic (PAL) device decodes the addresses of the ports to which the TLC32040 and the interface logic are mapped. The TMS320C16 uses the IN instruction at port 0 and the OUT instruction at port 1. Because the interface circuits are addressed only when the TMS320C16 executes an IN or an OUT instruction, the PAL equations are required to read from and write to the shift registers on these instructions only.

The $\overline{\text{FSR}}$ (pin 4) and $\overline{\text{FSX}}$ (pin 14) in the AIC have the same timing if the AIC is in the synchronous mode. The $\overline{\text{FSX}}$ is used for both transmit and receive frame syncs. During the serial transmission mode, the $\overline{\text{FSR}}$ and $\overline{\text{FSX}}$ are held low as the AIC starts to convert the analog signal to a digital signal with 14-bit resolution. The conversion is tied from DR (pin 5) of the AIC to the SN74LS299 shift registers. The contents of these two registers shift to the AIC through DX (pin 12).

Communication With the EVM

After receiving and/or transmitting, both $\overline{\text{FSR}}$ and $\overline{\text{FSX}}$ go high, and an end-of-data receive signal, $\overline{\text{EODR}}$ (pin 3), and end-of-data transmit signal, $\overline{\text{EODX}}$ (pin 11), go low. Both signals have the same timing because of the synchronous mode. For example, you can select $\overline{\text{EODX}}$ to interrupt the EVM through the TARGET_INT/ of DSUB connector (pin 14) upon completion of serial communication.

In an application program, you should use the IN instruction with the interrupt service routine to read data from port 0 (that is, to read data from the two SN74LS299 shift registers). Similarly, write data with the OUT instruction before the next $\overline{\text{FSX}}$ signal goes low. This period is dependent upon the DAC and ADC conversion rate. All data transmission is synchronous with the SHIFT CLK (pin 10).

Hardware Interface

The 74LS74 flip-flop ensures the setup and hold times of the SN74LS299 shift registers. It uses the SRCLK from the PAL along with the DR from the AIC.

The PAL also provides a XRESET signal that connects to the CLR pins of the SN74LS299 shift registers, as well as to the XRESET pin of the TLC32040 AIC. This ensures that the registers are clear when the AIC begins to transfer data; it also minimizes the possibility that the AIC will shift in bad data that could cause the AIC to shut down or behave in an unexpected manner.

The XRESET signal provided by the PAL comes from two sources:

- hardware reset with DIP SWITCH, or
- programmable software with the TMS320C16 OUT instruction at port 2 implemented via a SN74LS74.

This allows the application program to control the AIC board.

The TMS320C16 EVM supports external I/O operations via a standard 37-pin DSUB connector. All the I/O lines are buffered to protect the TMS320C16. Table 1 lists the I/O expansion connector pinouts between the TMS320C16 EVM and the TLC32040 AIC board.

Figure 2. Serial-to-Parallel Conversion Circuit (Synchronous Mode)

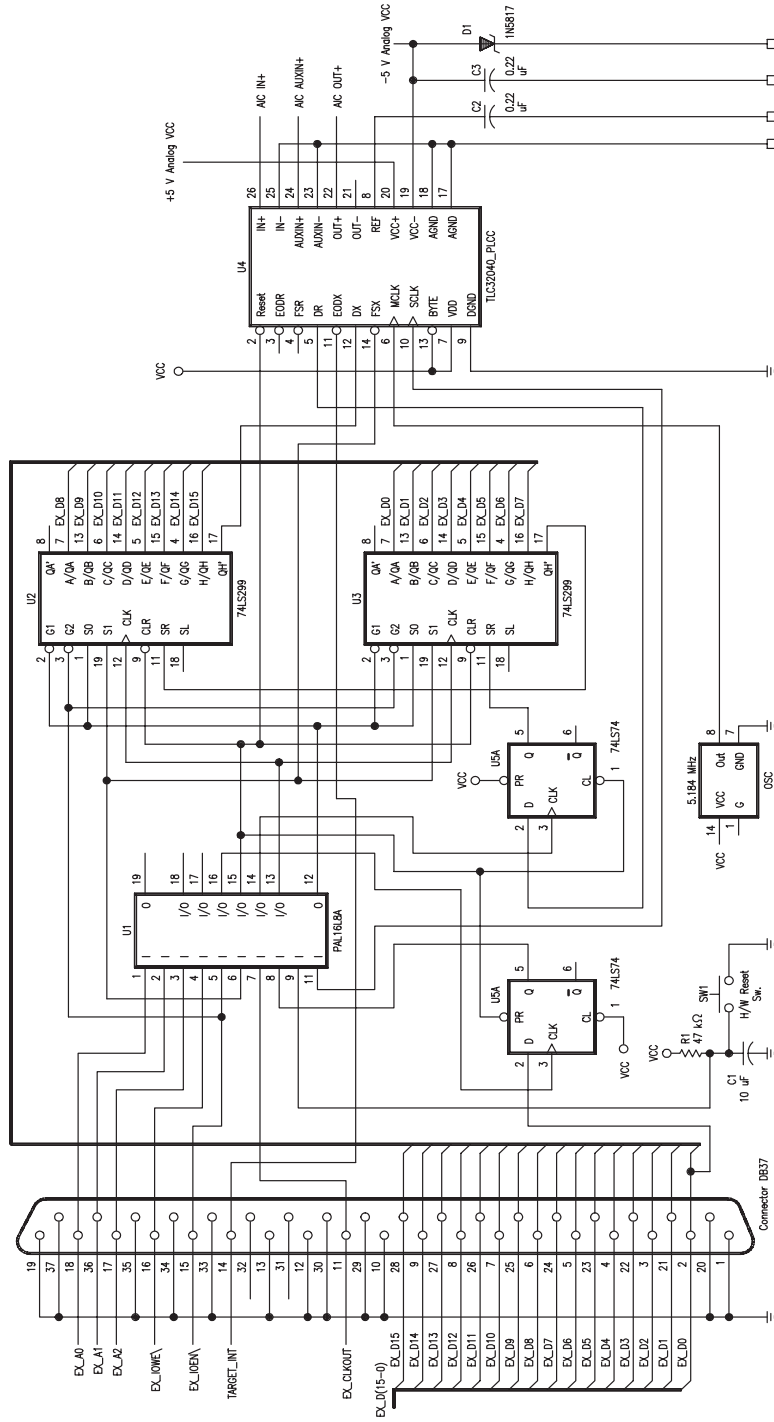


Table 1. EVM Connector Pinouts

Pin Number	Signal Name	Signal Type	Signal Description
1, 10, 12, 13, 19, 20, 29, 33, 34, 35, 37	GND	Output	Ground
2	EX_D0	I/O	Buffered Data Bit 0
3	EX_D1	I/O	Buffered Data Bit 1
4	EX_D2	I/O	Buffered Data Bit 2
5	EX_D3	I/O	Buffered Data Bit 3
6	EX_D4	I/O	Buffered Data Bit 4
7	EX_D5	I/O	Buffered Data Bit 5
8	EX_D6	I/O	Buffered Data Bit 6
9	EX_D7	I/O	Buffered Data Bit 7
21	EX_D8	I/O	Buffered Data Bit 8
22	EX_D9	I/O	Buffered Data Bit 9
23	EX_D10	I/O	Buffered Data Bit 10
24	EX_D11	I/O	Buffered Data Bit 11
25	EX_D12	I/O	Buffered Data Bit 12
26	EX_D13	I/O	Buffered Data Bit 13
27	EX_D14	I/O	Buffered Data Bit 14
28	EX_D15	I/O	Buffered Data Bit 15
18	EX_A0	Output	Buffered Address Bit 0
36	EX_A1	Output	Buffered Address Bit 1
17	EX_A2	Output	Buffered Address Bit 2
16	EX_IOWE\	Output	Buffered I/O Write Strobe
15	EX_IOEN\	Output	Buffered I/O Read Strobe
11	EX_CLKOUT	Output	TMS320C16 Buffered CLKOUT
31	EX_RESET\	Output	TMS320C16 Reset
14	TARGET_INT\	Input	TMS320C16 Interrupt
32	TARGET_BIO\	Input	TMS320C16 BIO

NOTE: The TMS320C16 EVM has two interrupt sources. One is from the PC, and the other one is from external I/O. You can select the latter through the control register at one of four I/O base address ranges by setting the EVM switches 1 and 2 as shown in Table 2.

Table 2. TMS320C16 EVM Switch Settings for I/O Address

I/O Address Space	Switch	
	1	2
0x0240 – 0x025F	ON	ON
0x0280 – 0x029F	ON	OFF
0x0320 – 0x033F	OFF	ON
0x0340 – 0x035F	OFF	OFF

The control register offset is 0x0002. Be sure that the EVM interrupt source is external before communicating with the AIC board. Example 1 of the software listings in Appendix C shows a segment of code that sets the interrupt source to external.

Power Requirements

+12 V	@ 40 mA
-12 V	@ 40 mA
+ 5 V	@ 15 mA
- 5 V	@ 15 mA

Software Description

The communication program flowcharts and the TMS320C16 EVM program listings are shown in Appendix B. These programs are also available on the Texas Instruments TMS320 DSP Bulletin Board Service (BBS) at (713) 274-2323.

Initializing the TMS320C16 EVM

As shown in the communication program flowcharts, the program begins with an initialization routine that clears the DATA RAM, resets the AIC board, clears the transmit/receive end flag and the secondary communication flag, and stores the address of the interrupt subroutines. The program uses the MPYK and PAC instruction sequence to load data memory locations with the 12-bit address of the subroutines. This sequence is necessary only if the subroutines reside in program memory locations larger than 0x0FF. Otherwise, you can use the instructions LACK and SACL to initialize the subroutine address storage locations.

Communication With the AIC Board

After the storage registers and status register have been initialized, the interrupt is enabled and control is passed to your application routine (i.e., the FIR and IIR filter routine). The program ignores the first interrupt that occurs after interrupts are enabled. This allows the AIC to stabilize after a reset. The application routine should not write to the shift registers while data is moving into (or out of) them. In addition, it should ensure that no primary data is written to the shift registers between a primary and secondary data communication pair.

The number of instruction cycles between data transfers can be calculated from the conversion frequency. By counting instruction cycles in the application program, it is possible to determine whether the data transfer will conflict with the OUT instruction to the shift registers. The second objective can be accomplished by monitoring SNDFLG in the application program. If SNDFLG equals 0x00FF, then secondary communication has not been completed.

When the processor receives an interrupt, the program counter is pushed onto the hardware stack and then the program counter is set to 0x0002, the location of the interrupt service routine. The interrupt service routine then saves the contents of the accumulator and the status register and calls the interrupt subroutine to which XVECT points. If secondary communication is to follow the upcoming primary communication, XVECT is set by the application program to refer to SINT1; otherwise, XVECT defaults to NINT (the normal interrupt routine).

TLC32040 Secondary Communication

To write to the control register of the AIC or configure any of the AIC internal counters, you must use the application program to initiate a primary/secondary communication pair. To do this, place a data word in which bits 0 and 1 are both high into DXMT, place the secondary control word in D2ND, and place the address of the secondary communication subroutine, SINT1, in XVECT. When the next interrupt occurs, the interrupt subroutine will call routine SINT1. SINT1 reads A/D information from the shift registers and writes the secondary communication word to the shift register.

Summary

This report shows how to build an analog interface to the TMS320C16 EVM. The AIB consists of glue logic, shift registers, and an AIC that performs the A/D and D/A conversions as well as filtering functions.

Complete schematics and software are included in this report. The paper provides a theory of operation as well as the practical implementation details.

Appendices

Appendix A — TMS320C16 EVM PAL Equations

equations

PAL16L8

EVM & INTERFACE GLUE LOGIC PAL EQUATIONS MMI

EX_A0 EX_A1 EX_A2 EX_IOWE EX_IOEN XFSX CLKOUT XRESETIN XRESETDRV GND
SHFCLK SOG1 CLK299 SRCLK XRESET XIOWCLK NC NC NC VCC

IF(VCC) /SOG1 = XFSX*/EX_A0*/EX_A1*/EX_A2 + EX_IOWE*XFSX

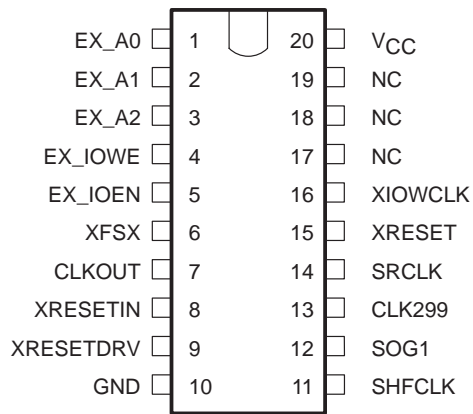
IF(VCC) /CLK299 = XFSX*EX_A0*/EX_A1*/EX_A2*CLKOUT*/EX_IOWE+/XFSX*/SHFCLK

IF(VCC) /SRCLK = SHFCLK + XFSX

IF(VCC) /XRESET = /XRESETIN + /XRESETDRV

IF(VCC) /XIOWCLK = /EX_IOWE*EX_A0*/EX_A1*EX_A2*/CLKOUT

Figure 3. PAL Pinouts



Appendix B —Software Flowcharts for AIC Initialization and Communication

Figure 4. Main Routine Flowchart

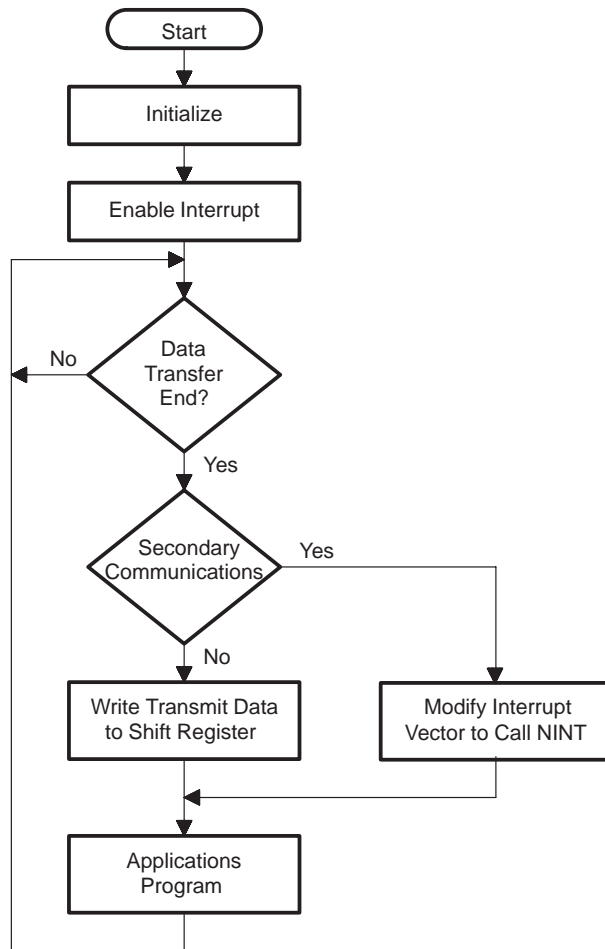


Figure 5. Primary Interrupt Routine

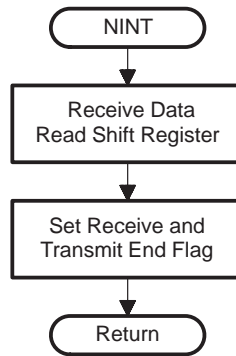


Figure 6. Secondary Communications 1

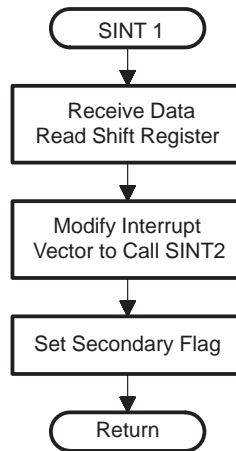
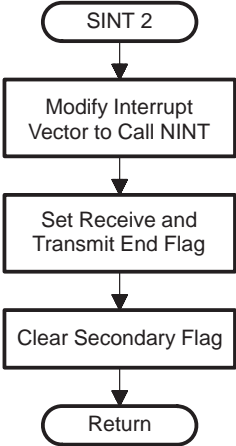


Figure 7. Secondary Communications 2



Appendix C — Software Listings

Example 1. Initializing Interrupt Source Code

```
MOV     DX,IO_PORT    ;port of EVMC16 depend on your selection
                        ;it may be 0x242,0x282,0x322 or 0x342
IN      AX,DX         ;Get the data of EVMC16 Control Register
MOV     BX,07FFFh
AND     AX,BX         ;Bit-15 set to 0, select external as
                        ;interrupt source
OUT     DX,AX
INT     3
```

Example 2. Main Program With FIR Routine

```
;File name == mainfir.asm
;
;*****
;      .TITLE   "EVMC16 & TLC32040 COMMUNICATION PROGRAM"
;*****
;
;   WRITTEN BY:  CHAUCER KUO
;   REVISION:   1.0      Dec 1992          **
;
;   DATA RAM PAGE0= IIR COEFF          00 ~ 7F
;   DATA RAM PAGE1= VARIABLE          80 ~ FF
;
;
;   .length 60
;   .width 120
;   .option X
;
;   .GLOBAL START,MAIN,MAIN1,INTSVC,NINT,SINT1,SINT2,IGINT
;   .DEF     RXDR
;   .REF     FIR_FITR
;   .DEF     FIRCOEF,FIRTBL
;   .DEF     DXMT
;   .DEF     XNDN
;
;The following VARIABLES will be stored in DATA memory
;
;   .data
;   LO      .SET    0F0h
;   HIGH    .SET    0F1h
;   RXEFLG  .SET    0F2H
;   SNDFLG  .SET    0F3H
;   RXDR    .SET    0F4H
;   XNDN    .SET    0F5H
;   D2ND    .SET    0F6H
;   INTVECT .SET    0F7H
;   ACHSTK  .SET    0F8h
;   ACLSTK  .SET    0F9h
;   SSTSTK  .SET    0FAh
;   ANINT   .SET    0FBh
;   ASINT1  .SET    0FCh
;   ASINT2  .SET    0FDh
;   TMP0    .SET    0FEh
;   DXMT    .SET    0FEh
;   AND_BUFR .SET    0EEh
;
;   ZERO    .SET    0
;   ONE     .SET    01
;   SET     .SET    0FFh
;
;   FIRL    .SET    093
```

Example 2. Main Program With FIR Routine (Continued)

```

INIT_DATA    .SET    0h
              .text
              B      START
              B      INTSVC

              .PAGE
; *****
; ***   Initializations routine after RESET EVMC16   ***
; *****

START:
              DINT
;Clear the DATA MEMORY
              SOVM          ;DISABLE OVERFLOW MODE
              LARK    0,255 ;AR0
              ZAC          ;CLEAR Accumulator
              LARP    0      ;AR0 AS POINTER AND LOOP CONTROL
              LACK    INIT_DATA

LOOP:
              SACL    *
              BANZ   LOOP

;*****
;   RESET THE AIC BOARD
;*****
;
              LACK    ZERO
              SACL    LO
              OUT     LO,PA2
;*****
              LDPK    ONE
              LACK    ONE
              SACL    TMP0      ; TMP0=1
              LT     TMP0      ; T Reg=1

              MPYK    NINT      ; (TMP0) * (NINT) --> (P)
              PAC          ; (ACC) <-- (P)
              SACL    ANINT     ;Save primary com. program starting address

              MPYK    SINT1
              PAC
              SACL    ASINT1    ;Save secondary com. program starting address1
              MPYK    SINT2
              PAC
              SACL    ASINT2    ;Save secondary com. program starting address2

;***** FIRST INTERRUPT VECTOR SETTING *****
              MPYK    IGINT     ;Save ignore interrupt once after master reset
              PAC
              SACL    INTVECT

              ZAC
              SACL    RXEFLG,0   ;RXEFLG=0 with 0 shift
              SACL    SNDFLG,0   ;SNDFLG=0 with 0 shift

***** READ COEFFICIENT TABLE*****
*
RDTABLE:
              LT     TMP0          ; TMP0 = 1
              MPYK    FIRTBL
              PAC          ;(ACC)<--the start address of COE
              MPYK    1            ;(P) <-- 1
              LARP    AR0
              LARK    AR0,FIRL-1

```

```
                                LARP   AR1           ;use AR1 as starting address
                                LARK   AR1,FIRL+FIRCOEF;move FIR coefficients
LOOP1:
                                LARP   1
                                TBLR  *,0
```

Example 2. Main Program With FIR Routine (Continued)

```

APAC                                ;(ACC) <(ACC)+(P Reg=1)
BANZ    LOOP1LT    TMP0
MPYK    AND_DATA
PAC
TBLR    AND_BUFR

; RELEASE THE AIC BOARD FROM RESET STATUS
LACK    SET
SACL    HIGH      ; HIGH=1
OUT     HIGH,PA2
EINT

;*****MAIN ROUTINE START HERE *****
MAIN:
LAC     RXEFLG,0    ;(acc) <-- RXEFLG
BZ      MAIN       ; Wait for interrupt

;***** DATA MANIPULATION *****
LAC     SNDFLG,0    ;Skip OUT instruction during
BNZ     MAIN1      ;secondary commuication

CALL    FIR_FITR

;*****
; Output to the AIC

LAC     XNDN
AND     AND_BUFR    ;Make sure the bit 0 and bit 1 is 0
SACL    DXMT
OUT     DXMT,PA1    ;Write Xmit data to shift Reg.

MAIN1:
ZAC
SACL    RXEFLG      ;Clear ACC
                          ;Store to RXEFLG
B       MAIN

;*****
;*** WE HAVE TO SET THE INTERRUPT VECTOR AT LOCATION 2 ***
;*****

INTSVC:
DINT
SST     SSTSTK
SACH    ACHSTK
SACL    ACLSTK
LAC     INTVECT,0    ; INTVECT == IGNIT
CALA    ; (PC) <-- (ACC)
ZALH    ACHSTK
OR      ACLSTK
LST     SSTSTK      ;(SST Reg. ) <-- (SSTSTK)
EINT
RET

IGINT:
LAC     ANINT
SACL    INTVECT      ;Modify the interrupt vector
RET

NINT:
IN      XNDN,PA0
LACK    SET
SACL    RXEFLG      ;Set receive FLAG
RET

;***** Secondary Communication Interrupt Services Routine *****
SINT1:

```

```
IN      XNDN,PA0
OUT     D2ND,PA1      ;Write secondary data to sht reg
LAC     ASINT2,0
SACL    INTVECT      ;the new int. vector is SINT2
LACK    SET
SACL    SNDFLG,0     ;Set the secondary commu. flag
RET
```

Example 2. Main Program With FIR Routine (Continued)

```
SINT2:          LAC      ANINT
                SACL     INTVECT
                LACK     SET
                SACL     RXEFLG
                ZAC
                SACL     SNDFLG,0
                RET

                .COPY    "4BANDFIR.COE"
FIRCOEF         .usect  "coefs",FIRL
AND_DATA        .WORD   0FFFCH
                .PAGE
                .END
;***** End the program *****
```


Example 3. Main Program With IIR Routine

```

;File name == mainfir.asm
;
;*****
;      .TITLE  "EVMC16 & TLC32040 COMMUNICATION PROGRAM"
;*****
;
;  WRITTEN BY:  CHAUCER KUO
;  REVISION:   0.1    Dec 1992          **
;
;      DATA RAM PAGE0= IIR COEFF      00 ~ 7F
;      DATA RAM PAGE1= VARIABLE      80 ~ FF
;
;
;      .length 60
;      .width 120
;      .option X
;
;      .GLOBAL START,MAIN,MAIN1,INTSVC,NINT,SINT1,SINT2,IGINT
;
;      .DEF    RXDR
;      .REF    IIR_FITR
;      .DEF    IIRCOEF,IIRTLB
;      .DEF    DN,XNDN,DXMT
;The following VARIABLES will be stored in DATA memory
;      .data
LO          .SET    0F0h
HIGH       .SET    0F1h
RXEFLG    .SET    0F2h
SNDFLG    .SET    0F3h
RXDR      .SET    0F4h
D2ND      .SET    0F6h
INTVECT   .SET    0F7h
ACHSTK    .SET    0F8h
ACLSTK    .SET    0F9h
SSTSTK    .SET    0FAh
ANINT     .SET    0FBh
ASINT1    .SET    0FCh
ASINT2    .SET    0FDh
TMP0      .SET    0FEh
AND_BUFR  .SET    0EFh
;
XNDN      .SET    0EEh
DXMT      .SET    0EDh
DN        .SET    0ECh
;
ZERO      .SET    0
ONE       .SET    1
INIT_DATA .SET    0h
SET       .SET    0Fh
;
IIR_NUMBER .SET    04H
DN_BUFR   .usect  "DNN",4*2
;
;      .text
;      B      START
;      B      INTSVC
;
;      .COPY  "iirbp.coe"
IIRCOEF   .usect  "coefs",IIRLEN
;
;      .PAGE
; *****
; ***  Initializations routine after RESET EVMC16  ***
; *****

```

START:

DINT

Example 3. Main Program With IIR Routine (Continued)

```

;*****
;Clear the DATA MEMORY
      SOVM          ;DISABLE OVERFLOW MODE
      LARK          0,255 ;AR0
      ZAC           ;CLEAR Accumulator
      LARP          0 ;AR0 AS POINTER AND LOOP CONTROL
      LACK          INIT_DATA
LOOP:
      SACL          *
      BANZ          LOOP
;*****
; RESET THE AIC BOARD
;*****
;
      LACK          ZERO
      SACL          LO
      OUT           LO,PA2
;*****
      LDPK          ONE
      LACK          ONE
      SACL          TMP0 ; TMP0=1
      LT           TMP0 ; T Reg=1

      MPYK          NINT ; (TMP0) * (NINT) --> (P)
      PAC           ; (ACC) <-- (P)
      SACL          ANINT ;Save primary com. program starting address

      MPYK          SINT1
      PAC
      SACL          ASINT1 ;Save secondary com. program starting
address1

      MPYK          SINT2
      PAC
      SACL          ASINT2 ;Save secondary com. program starting
address2

      MPYK          IGINT ;Save ignore interrupt once after master
reset
      PAC
      SACL          INTVECT

      ZAC
      SACL          RXEFLG,0 ;RXEFLG=0 with 0 shift
      SACL          SNDFLG,0 ;SNDFLG=0 with 0 shift

***** READ PARAMETER TABLE*****
*
RDTABLE:
      LT           TMP0 ; TMP0 = 1
      MPYK          IIRTBL
      PAC           ;(ACC)<--the start address of COE
      MPYK          1 ;(P) <-- 1

      LARP          AR0
      LARK          AR0,IIRLEN-1
      LARP          AR1 ;use AR1 as starting address
      LARK          AR1,IIRCOEF ;move IIR coefficients
LOOP1:
      LARP          1
      TBLR          *,0
      APAC          ;(ACC) <(ACC)+(P Reg=1)
      BANZ          LOOP1

```

; RELEASE THE AIC BOARD FROM RESET STATUS

LT TMP0
MPYK AND_DATA
PAC
TBLR AND_BUFR

Example 3. Main Program With IIR Routine (Continued)

```

LACK      SET
SACL      HIGH
OUT       HIGH,PA2
EINT

;*****
MAIN:
          LAC      RXEFLG,0      ;(acc) <-- RXEFLG
          BZ       MAIN         ; Wait for interrupt

;***** DATA MANIPULATION *****
TEST_LOOP:
          LACK     IIR_NUMBER
          LARK     AR1, DN_BUFR
          LARK     AR0, IIRCOEF
          LARP     1

NEXT_ORDER:
          PUSH     ;(ACC(11-0)) --> TOS
          CALL    IIR_FITR
          DMOV    *           ;D(n-2) <-- D(n-1)
          LAC     DN
          SACL    *+          ;D(n-1) <-- D(n)
          LAC     *+          ;Point to next DN buffer,Dummy read
          ZAC     ;Clear ACC
          POP
          SUB     TMP0        ;(TMP0)=1
          BNZ     NEXT_ORDER

;*****
          LAC     SNDFLG,0    ;Skip OUT instruction during
          BNZ     MAIN1       ;secondary communication
;*****

; Output to the AIC
          LAC     XNDN
          AND     AND_BUFR
          SACL    DXMT
          OUT     DXMT,PA1    ;Write Xmit data to shift Reg.

MAIN1:
          ZAC     ;Clear ACC
          SACL    RXEFLG     ;Store to RXEFLG

          B       MAIN

;*****
INTSVC:
          DINT
          SST     SSTSTK
          SACH    ACHSTK
          SACL    ACLSTK
          LAC     INTVECT,0   ; INTVECT == IGNIT
          CALA   ; (PC) <-- (ACC)
          ZALH   ACHSTK
          OR     ACLSTK
          LST     SSTSTK     ;(SST Reg. ) <-- (SSTSTK)
          EINT
          RET

IGINT:
          LAC     ANINT
          SACL    INTVECT    ;Modify the interrupt vector
          RET

NINT:
          IN      XNDN,PA0
          LACK    SET

```

SACL RXEFLG ;Set receive FLAG
RET

Example 3. Main Program With IIR Routine (Continued)

```
;*****
SINT1:
        IN      RXDR,PA0
        OUT     D2ND,PA1      ;Write secondary data to shift reg
        LAC     ASINT2,0
        SACL    INTVECT      ;the new int. vector is SINT2
        LACK    SET
        SACL    SNDFLG,0     ;Set the secondary comm. flag
        RET

SINT2:
        LAC     ANINT
        SACL    INTVECT
        LACK    SET
        SACL    RXEFLG
        ZAC
        SACL    SNDFLG,0
        RET

AND_DATA      .WORD      0FFFCH
              .PAGE
              .END
;***** End the program *****
```

Example 4. Linker Command File (FIR)

```
;File name == mainfir.cmd

-e main
-o mainfir.out      /* replace fir with iir for other prog */
-m mainfir.map

mainfir.obj
fir.obj

MEMORY
{
    PAGE 0: PROG:   ORIGIN = 0x0, LENGTH = 10000H
    PAGE 1: DATA:  ORIGIN = 0, LENGTH = 100H
}

SECTIONS
{
    .text : {} > PROG
    .data : {} > DATA
    .coefs : {} > DATA
}

```


Example 6. FIR Coefficients File

```

; File Name == 4BANKFIR.COE
;*****
;** MAIN FILTER   FIR MULTIBAND           **
;**                                                     **
;** FINITE INPULSE RESPONSE (FIR)       **
;**                                                     **
;** FILTER TYPE: MULTIBAND               **
;**;FILTER LENGTH   =           93-TAP FIR FILTER **
;**;SAMPLING FREQUENCY=   8.000 (KHZ)      **
;**           BAND 1   BAND 2   BAND 3   BAND 4   **
;**LOWER BAND EDGE   .0000   1.0000   1.8000   2.7000 **
;**UPPER BAND EDGE   .7000   1.5000   2.5000   4.0000 **
;**NOMINAL GAIN       1.0000   .0000   2.0000   1.0000 **
;**NOMINAL RIPPLE     .0200   .0200   .0200   .0200 **
;**MAXIMUM RIPPLE     .0071   .0114   .0112   .0092 **
;**RIPPLE IN DB       .0616  -38.8863  -5.9721   .0794 **
;**                                                     **
;** GENERATED BY: CHAUCER KUO           **
;** USING THE DIGITAL FILTER DESIGN PACKAGE AVAILBLE **
;** FROM ATLANTA SIGNAL PROCESSING INC. **
;**                                                     **
;** REVISION:      1.0           Dec 1992 **
;*****

.page
* FILTER PARAMETER DEFINITION
*
* FIR COEFFICIENTS
  .DEF   FIRTBL

  .text
FIRTBL:

.WORD  -17           ;C092
.WORD  -64           ;C091
.WORD  -17           ;C090
.WORD   21           ;C089
.WORD   30           ;C088
.WORD   34           ;C087
.WORD  -33           ;C086
.WORD  -24           ;C085
.WORD  114           ;C084
.WORD   55           ;C083
.WORD -139           ;C082
.WORD -124           ;C081
.WORD  -30           ;C080
.WORD   57           ;C079
.WORD  172           ;C078
.WORD   14           ;C077
.WORD -167           ;C076
.WORD  111           ;C075
.WORD  247           ;C074
.WORD  -99           ;C073
.WORD -267           ;C072
.WORD -234           ;C071
.WORD  -91           ;C070
.WORD  384           ;C069
.WORD  354           ;C068
.WORD -296           ;C067
.WORD -138           ;C066
.WORD  420           ;C065
.WORD  141           ;C064

```

```
.WORD -253 ;C063
.WORD -516 ;C062
.WORD -703 ;C061
```

Example 6. FIR Coefficients File (Continued)

```
.WORD 346 ;C060
.WORD 1239 ;C059
.WORD 34 ;C058
.WORD -682 ;C057
.WORD 300 ;C056
.WORD 463 ;C055
.WORD 138 ;C054
.WORD -507 ;C053
.WORD -2596 ;C052
.WORD -1325 ;C051
.WORD 4109 ;C050
.WORD 3644 ;C049
.WORD -2293 ;C048
.WORD -2162 ;C047
.WORD 16998 ;C046
.WORD -2162 ;C045
.WORD -2293 ;C044
.WORD 3644 ;C043
.WORD 4109 ;C042
.WORD -1325 ;C041
.WORD -2596 ;C040
.WORD -507 ;C039
.WORD 138 ;C038
.WORD 463 ;C037
.WORD 300 ;C036
.WORD -682 ;C035
.WORD 34 ;C034
.WORD 1239 ;C033
.WORD 346 ;C032
.WORD -703 ;C031
.WORD -516 ;C030
.WORD -253 ;C029
.WORD 141 ;C028
.WORD 420 ;C027
.WORD -138 ;C026
.WORD -296 ;C025
.WORD 354 ;C024
.WORD 384 ;C023
.WORD -91 ;C022
.WORD -234 ;C021
.WORD -267 ;C020
.WORD -99 ;C019
.WORD 247 ;C018
.WORD 111 ;C017
.WORD -167 ;C016
.WORD 14 ;C015
.WORD 172 ;C014
.WORD 57 ;C013
.WORD -30 ;C012
.WORD -124 ;C011
.WORD -139 ;C010
.WORD 55 ;C009
.WORD 114 ;C008
.WORD -24 ;C007
.WORD -33 ;C006
.WORD 34 ;C005
.WORD 30 ;C004
.WORD 21 ;C003
```

.WORD -17 ;C002
.WORD -64 ;C001
.WORD -17 ;C000
.PAGE

Example 7. IIR Subroutine

```
; File name == IIR.ASM
;*****
; The following equations implement an IIR filter
;
; d(n) = x(n)+ d(n-1) * a1 + d(n-2) * a2
; y(n) = d(n) * b0 + d(n-1) * b1 + d(n-2) * b2
;
;*****

      .DEF   IIR_FITR
      .REF   DN,XNDN,DXMT

;** THE AIC INPUT store at RXDR
; AR0= The pointer of Coefficient
; AR1= The pointer of Dn,Dn-1,Dn-2
; Dn= The common buffer of Dn for 4 section
; XNDN= The common buffer of Yn for 4 section
      .TEXT
IIR_FITR:
      LAC   XNDN,15      ;XNDN=x(n) for each input
      LT    *,0          ; d(n-1) * a1
      MPY   *,1          ; A1
      LTA   *,0          ;(ACC)<--(ACC)+(P)<== x(n)+d(n-1)*a1
      MPY   *,0          ;(P) <-- d(n-2) * A2
      APAC          ;ACC<--(ACC)+(P) <==> d(n)
      SACH  DN,1         ;d(n)=x(n)+d(n-1)*a1+d(n-2)*a2
      ZAC
      MPY   *,1          ; (T) still remain d(n-2)
      LTA   *,0          ;(ACC) <== d(n-2)*b2
      MPY   *,0          ;(P) <==d(n-1)*b1
      LTA   DN          ;(T)=DN,(ACC)=d(n-1)*b1+d(n-2)*a2
      MPY   *,1          ;(P)==d(n)*b0,point to next Coe.
      APAC          ;(ACC)==(P)+(ACC)
      SACH  XNDN,1       ;y(n)=d(n)*b0+d(n-1)*b1+d(n-2)*b2
      RET

;After return from this sub-routine the AR0 point to next starting
;address and AR1 point to D(n-1)
; AR point to AR1
```

Example 8. IIR Coefficients File

```
; file name == iirbp.coe
;*****
;** 4-STAGE RECURSIVE FILTER **
;** SECOND ORDER SECTIONS **
;** Filter type: BANDPASS **
;** Approximation type: ELLIPTIC **
;** Sampling freq: 8000 HZ **
;** FILTER ORDER = 8 IIR FILTER **
;** BAND 1 BAND 2 BAND 3 **
;** LOWER BAND EDGE .00000 2.00000 3.50000 **
;** UPPER BAND EDGE 1.00000 3.00000 4.00000 **
;** NOMINAL GAIN .00000 1.00000 .00000 **
;** NOMINAL RIPPLE .00500 .01000 .00500 **
;** MAXIMUM RIPPLE .00134 .00364 .00134 **
;** RIPPLE IN dB -57.45283 .03154 -57.45144 **
;** GENERATED BY: CHAUCER KUO **
;** USING THE DIGITAL FILTER DESIGN PACKAGE AVAILBLE **
;** FROM ATLANTA SIGNAL PROCESSING INC. **
;** **
;** REVISION: 1.0 Dec 1992 **
```

```
;**
;*****
. DEF IIRLEN
. DEF IIRTBL

IIRLEN .SET 20
. text
IIRTBL:
. WORD -9807
. WORD -17223
. WORD 6440
. WORD -11850
. WORD 6400

. WORD -30971
. WORD -19439
. WORD 13921
. WORD 27447
. WORD 13921

. WORD 3975
. WORD -26263
. WORD 10827
. WORD -14338
. WORD 10827

. WORD -45143
. WORD -28302
. WORD 24272
. WORD 45273
. WORD 24273

. PAGE
```

Appendix D — Parts List for the Analog Interface Board (AIB)

Item	Quantity	Reference	Part
1	2	C1, C8	10 μ F
2	4	C2, C3, C4, C5	.22 μ F
3	1	C6	.047 μ F
4	1	C7	220 μ F
5	1	R1	47 k Ω
6	1	R2	10 k Ω
7	1	R3	100 k Ω
8	2	R4, R6	2 k Ω
9	1	R5	8 k Ω
10	1	R7	10 Ω
11	1	U1	PAL16L8A
12	2	U2, U3	74LS299
13	1	U4	TLC32040
14	1	U5	74LS74
15	1	U6	TL072
16	1	U7	LM386
17	1	OSC	5.184 MHz
18	1	D1	1N5817
19	1	SW1	Push Button
20	2	J1, J2	RCA Jack
21	1	JP1	Header 3
22	1	CON1	Connector DB37

