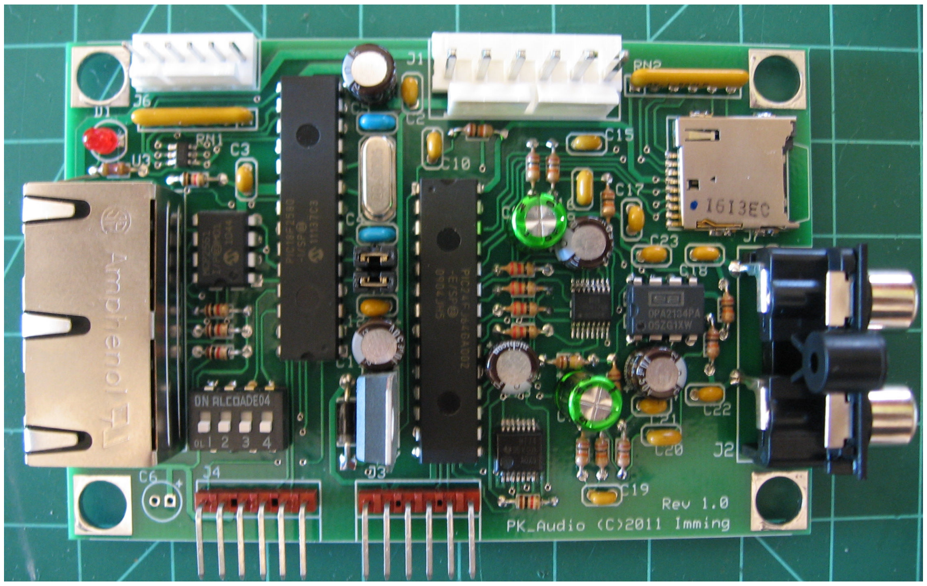
PinKit Audio Board

The PinKit Audio Board plays back audio files from a MicroSD memory card to generate high (near CD) quality audio. It is the audio generation board in the set of controller boards that comprise the PinKit controller system. The audio board connects to the PinCAN bus and processes sound commands. It mixes up to three tracks of digital audio and converts it to high-quality stereo line-level outputs. A standard audio amplifier can be used to amplify this to drive a set of speakers.

For details on the complete PinKit system, see http://pinkit.planetimming.com . While this board is intended to be connected as part of a PinCAN system, the board has dedicated switch inputs and an RS-232 interface option that allows it to be used as a stand-alone audio board.



Actual board size is 3.8" x 2.5"

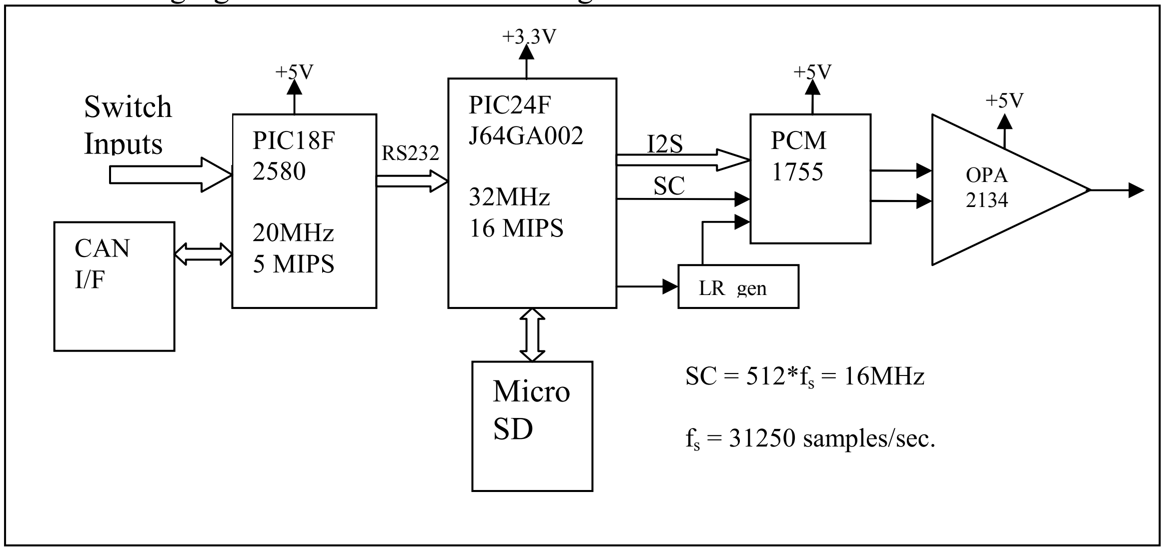
Features:

1. Three track audio with per-track volume control (music, voice, effects)
2. 31250 Hz audio sample rate
3. 4-bit ADPCM audio data storage. ITU G.721 Adaptive Pulse Code Modulation algorithm based on the Interactive Multimedia Association's reference algorithm
4. High-performance audio DAC (digital-to-analog converter)
5. High-performance audio filter, 23 KHz low-pass
6. Line out to external amplifier
7. Optional headphone output
8. MicroSD audio storage supporting FAT-16 or FAT-32 ; a 2GB MicroSD card provides over 35 hours of audio.
9. Software controlled volume level

10. RS-232 serial interface (115,200 baud, 8 data bits, no parity, 1 stop bit) for use as a stand-along audio board

Block Diagram

The following figure shows a block level diagram of the PinKit audio board.



The CAN/IF block is the RJ45 connector and CAN transceiver that provides the connection to the CAN bus that is used for system communication. The CAN bus controller is onboard the PIC18F2580 processor. The CAN I/F and PIC18 is common to all of the PinKit controllers. The output of the PIC18 is an RS-232 interface that controls the sound portion of the audio card. Jumpers are provided to disconnect the CAN bus logic so that RS-232 commands can be sent directly into the audio controller.

The PIC24F processor handles all of the audio processing; reading data from the microSD card, decoding the ADPCM data stream, mixing the audio tracks, and transmitting the I2S data to the audio DAC. Since the processor does not directly support the I2S data format, the LR_Gen logic generates the Left/Right data signal needed for I2S from the SPI stream transmitted by the processor. The final block is the OPA2134 audio amplifier that is configured as a low-pass filter. The bulk of the discrete components are part of this low-pass filter.

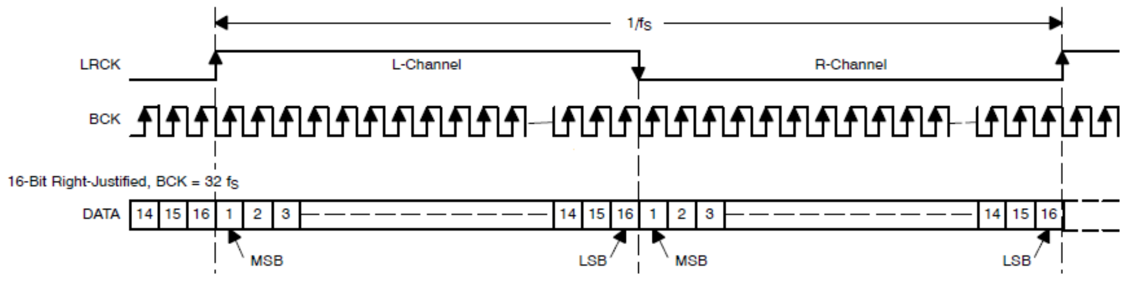
Bandwidth Table

Interface	Bandwidth Required	Notes
I2S	125 KB/sec	Exact bandwidth determined by two 16-bit sound values every 32 us
SPI/MicroSD	46,875 Bytes/sec	Three channels of 4-bit ADPCM data. Additional bandwidth will be needed for file system operations. Theoretical max available is 49.2 KB/sec.
Command	very low, ~ 360 bits/second	Typically about one command every 0.1 seconds. Available bandwidth is 115,200 baud

I2S Interface

The Inter-Integrated Circuit Sound (I^2S) interface is the most common interface for audio digital-to-analog converters (DAC). It is also a very demanding interface in terms of bit rate and timing requirements, plus it requires a continuous, un-interrupted stream of audio data. At a sample rate of 31,250 samples, one bit of audio data must be provided every 1 microsecond. This requires that the serial clock toggles every 500 ns. This means that a microcontroller running 16 MIPS, or 62.5 ns per instruction, must change the state of I/O pins every 8 instructions. This does not leave enough instructions to receive and format data without some type of hardware assist.

The following figure shows the timing of the I2S bus. For completeness, it must be pointed out that this is not fully I2S compliant, it's a variant called 16-bit right-justified. The only difference is in the alignment of the LRCK signal, so this does not relax the timing requirements.



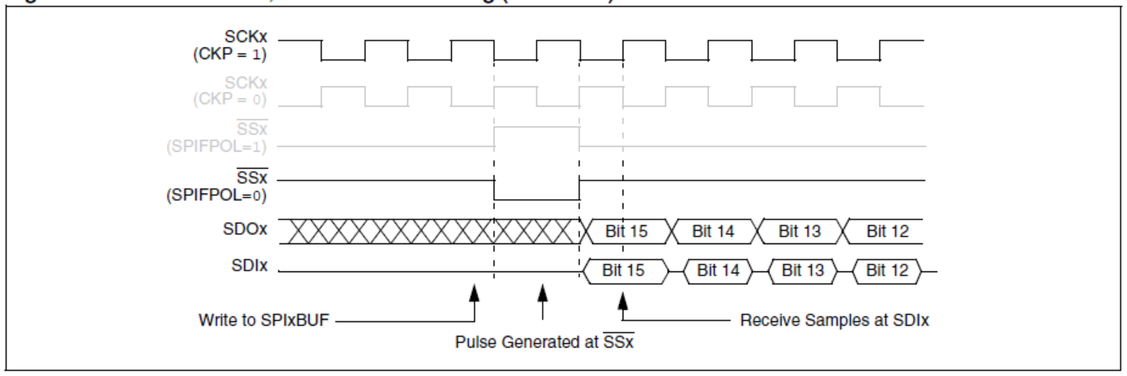
From this figure, LRCK is the Left/Right channel clock and indicates whether the current data is for the left or right audio channel. BCK is the bit clock; DATA is captured on the rising edge of BCK. DATA is the audio sample, 16-bits provided with most significant bit (MSB) first.

Although this seems like a common problem that would have been solved previously, a significant amount of research turned up no solutions. Checking chip suppliers and electronics retailers turned up no chips that interface a standard bus (e.g. SPI) to I2S.

Existing designs that were found use a programmable logic device like a MAX II CPLD. The solution invented for the PinKit Audio board relies on several special hardware features of the Microchip PIC24F microcontrollers. The solution is described here.

The PIC24F provides an SPI mode called "Framed SPI." It is intended to connect to devices together with an SPI bus in a master/slave configuration. As part of this support, the PIC24F can generate a pulse on the SS pin at the start of every SPI transfer as shown in the following figure from the SPI section of the PIC24F family reference manual.

Figure 23-10: SPI Master, Frame Master Timing (SPIFE = 0)


SCKx (CKP = 1)
SCKx (CKP = 0)
$\overline{\text{SSx}}$ (SPIFPOL=1)
$\overline{\text{SSx}}$ (SPIFPOL=0)
SDOx
SDIx
Bit 15
Bit 14
Bit 13
Bit 12
Write to SPIxBUF
Pulse Generated at $\overline{\text{SSx}}$
Receive Samples at SDIx

This SS signal is used as the clock for a 7474 positive edge triggered D flip-flow configured as a toggle flip-flop. The RESET pin of the flip-flop is used to initialize the flip-flop at start-up so the phase of the I2S LRCLK signal is correct.

Internal Serial Interface

The interface between the PIC18 and the PIC24F is a standard serial (RS-323 like) UART interface at 115,200 baud. The format is 8-bits, no parity, one stop bit. Optional jumpers are provided on the board in case there are applications that want to control this interface directly. All audio functions are controlled through this interface.

Why serial? After looking at several options, any parallel command options required too many microcontroller pins. SPI would be a good option, but the two available hardware controllers are already in use on the PIC24F. Performance requirements did not justify inventing a unique 4-bit parallel bus when a standard UART interface would work. In the end, this also may allow additional uses for the audio board as there are many options for driving a standard serial bus.

Serial Commands

Commands on the serial bus are 1 upper case character followed by one or more hexadecimal digits in lower case. All commands are sent as ASCII characters. The following table lists the available commands.

Command	Data	Description
M	CNN	Master Volume control. Set left/right channel or both channels to volume NN. Left: C='0' Right: C='1' Both: C='3' NN is in the range of 00 to "ff", where "ff" is maximum volume. The PCM1755 has a soft-mute function that prevents pops and clicks during volume changes.
V	CTV	Track Volume control. Set left/right channel or both channel track T attenuation to V, T is 0-2 and V is 0-f. C is the same as above. 0 is the minimum volume and 'f' is the maximum volume. This attenuation is done in software during the mixing of the three channels of sound.
P	TNNN	Play sound NNN on track T
Q	TNNN	Queue sound NNN on track T to play next.
R	TNNN	Play and Repeat sound NNN on track T
K	TNNN	Skip forward NNN/10 seconds on track T
S	T	Stop playing sound on track T, stops after current sound has completed. T is 0-2. Setting T = 'f' stops all tracks.
A	T	Abort playing sound on track T, stops immediately. T is 0-2. Setting T = 'f' aborts all tracks.

Example 1: Set left and right channel volume to maximum.
```
Mffff
```

Example 2: Play sound 001 on track 1:
```
P1001
```

Return Code:
PIC24F returns "OK" when a command is successfully received. If any problem is encountered, PIC24F returns "EN", where N is a single character representing the type of error[1].

Sound file naming

Sound files are named "SNDxxx.ad4", where "xxx" is the three hex digit value (NNN) used in the serial commands to play the file.

For example, the sound command "P1001" will play file "SND001.ad4" on track 1.

Sound file banks

The PinKit PLAY command can address up to 256 sound files. While normally this is adequate, a banking option is available to expand this to 3136. A bank register is provided that expands the sound indices by one hexadecimal digit. The default value is

[1] This is optional and not currently enabled in the PinKit usage.

0x0 which makes sound files 0x000 – 0x0FF available by default. The first 64 sound values are automatically mapped into every bank. In other words, with bank 1 selected, playing sound 0x3F will play sound 0x03F, but playing sound 0x40 will play sound file 0x140. The suggested use is to keep frequently used sounds in the first 64 indices and using the bank select to switch between the banks. The following table shows the sound index mapping.

Index	Bank 0	Bank 1	Bank 2	…	Bank 0xF
0-63	0x000 – 0x03F				
64-255	0x040 0x041 0x042 … 0x0FF	0x140 0x141 0x142 … 0x1FF	0x240 0x241 0x242 … 0x2FF		0xF40 0xF41 0xF042 … 0xFFF

Sound file format

Sound files must be pre-stored on the MicroSD card in the root directory. Sub-directories are not supported. The sound file format is raw 4-bit ADPCM per the ITU G.721 reference algorithm.

ADPCM provides a 4:1 compression in data size while maintaining very high quality (near 16-bit) sound. Compression is not needed for storage on the MicroSD, but it lowers the bandwidth requirements so multiple tracks can be played simultaneously.

Sound file caching

The first 64 sound files (xxx <= 0x40) have their File Allocation Table (FAT) cluster list prefetched into the PIC24F memory. This is necessary to avoid repeated searches in the FAT that are required to stream through multiple open files simultaneously. The cluster list is compressed into a run-length encoded format (start + length) that is extremely efficient for an unfragmented file structure.

Adaptive Differential Pulse Code Modulation (ADPCM)

ADPCM combines several techniques to obtain high compression ratio while maintaining a high-quality audio result.

Audio is encoded digitally by sampling the analog waveform and recording the value at each sample point. These samples are later used to reconstruct the original waveform for playback. The faster the samples are taken the higher the frequencies that can be accurately reproduced. The more resolution (bits) that is used for each sample, the more accurately the original waveform can be reproduced. Increasing either (or both) of these increases the amount data and the data rate that must be maintained to play back the audio. Compact Disc (CD) audio is stored with 16-bit samples at 44.1KHz sample rate.

Pulse Code Modulation – Each sample represents the magnitude of the waveform at that time. Playback involves creating a pulse of that magnitude for every sample.
Differential – Since adjacent samples are typically of similar magnitude, sample size can be reduced by only storing the difference between samples
Predictive – Sample size can be further reduced by predicting a value for the next sample and storing the difference from the predicted sample
Adaptive – Still further savings can be realized by changing, or adapting, the step size represented by the sample based on the previous samples

4-bit ADPCM uses all of these techniques to maximize the data quality while minimizing data storage.

Creating Sound Files

The basic steps to create a sound file for the audio boards are:

1. Create or copy a WAV file in PC format
2. Convert the WAV file to 16-bit mono at 31,250 samples per second
3. Extract the audio samples from the new WAV file (i.e. delete the first 44 bytes)
4. Convert the raw audio data to ADPCM

This section provides a step-by-step process to do this using a free set of tools. Windows batch files that help automate some of these steps are available and described in the next section.

Wavosaur audio editor, http://www.wavosaur.com/
Sox Sound Exchange, http://sourceforge.net/projects/sox/
PCspeech, provided with support files for Microchip Application Note 643, but patched some data types to get correct results.

Step 1 & 2
Start Wavosaur,
Open the wav file to convert.
Process -> Resample, set sample rate to 31250
Process -> Convert to Mono -> Mix all channels
Process -> Normalize -> 0 db
File -> Save As -> sndxxx.wav, where "xxx" is the name you choose

Note that Wavosaur can also record PC sounds by pressing the [record] button. After recording, highlight the section of audio you want to save:
Edit -> Copy
Edit -> Other Paste -> Paste into new file

Step 3
At windows command prompt:
```
sox -s -2 -L -c 1 -r 31250 snd001.wav snd001.raw
```

Step 4
At windows command prompt:
```
pcspeech e snd001.raw snd001.ad4
```

Note: sox can create .ima files, which should be equivalent to the G.721 ADPCM files generated by pcspeech, but the produced file is different. There may be more options to correct this, but it would need to be investigated.

The generated file can be checked by converting it back to a WAV file.

1. pcspeech d snd001.ad4 snd001new.raw
2. sox -s -2 -L -c 1 -r 31250 snd001new.wav snd001new.wav

The new WAV file should be identical to the original snd001.wav file.

Sound Utilities – One Time Setup

The sound utilities run from a Windows command line. Create a working directory and unpack the contents of pk_tools.zip into the directory. Download Wavosaur from http://www.wavosaur.com/download/files/Wavosaur.1.0.5.0(en).zip This can be installed in the same directory.

Windows Batch Files

Two batch files help with the conversion. Change directory into the directory you created prior to running the following commands.

To convert a wave file to AD4 format required by the audio card, use:

```
C:\> pk_convert FILE.wav
```

where FILE is the name of the wave file, e.g. SND001

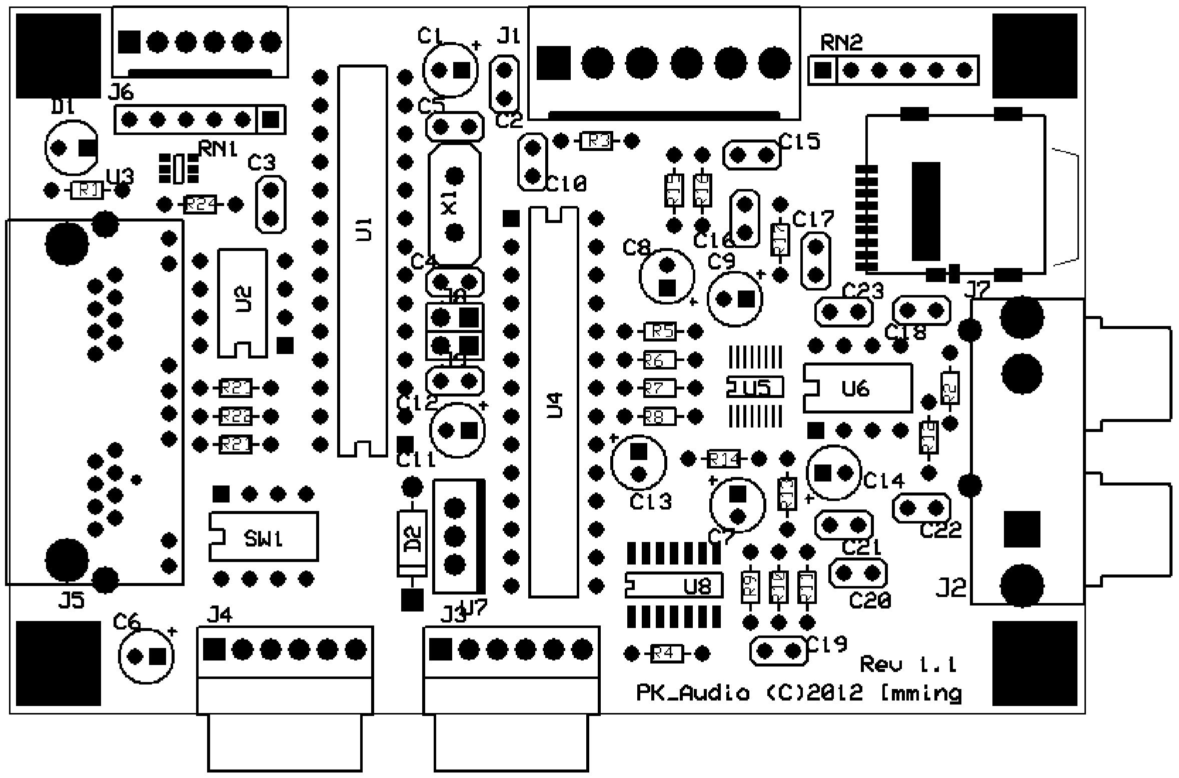
Edit the copyfiles.bat file to run the specific copy commands for all the sound files needed for your application.

After the AD4 files are created, plug the MicroSD card into a USB port and run:

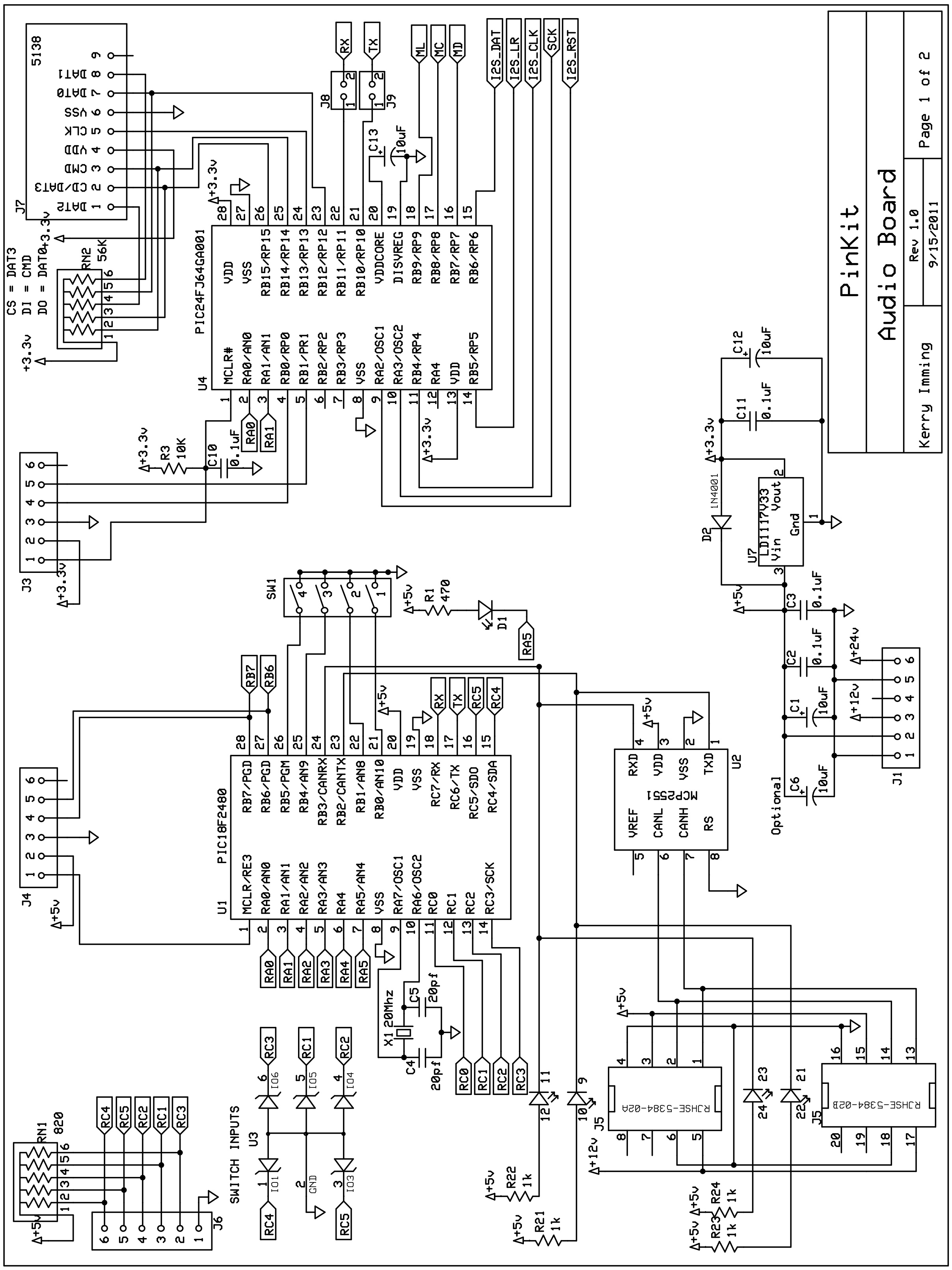
```
C:\> copyfiles
```

This will prompt for the drive letter of the MicroSD card. The card will be formatted and the sound files will be copied onto the card.

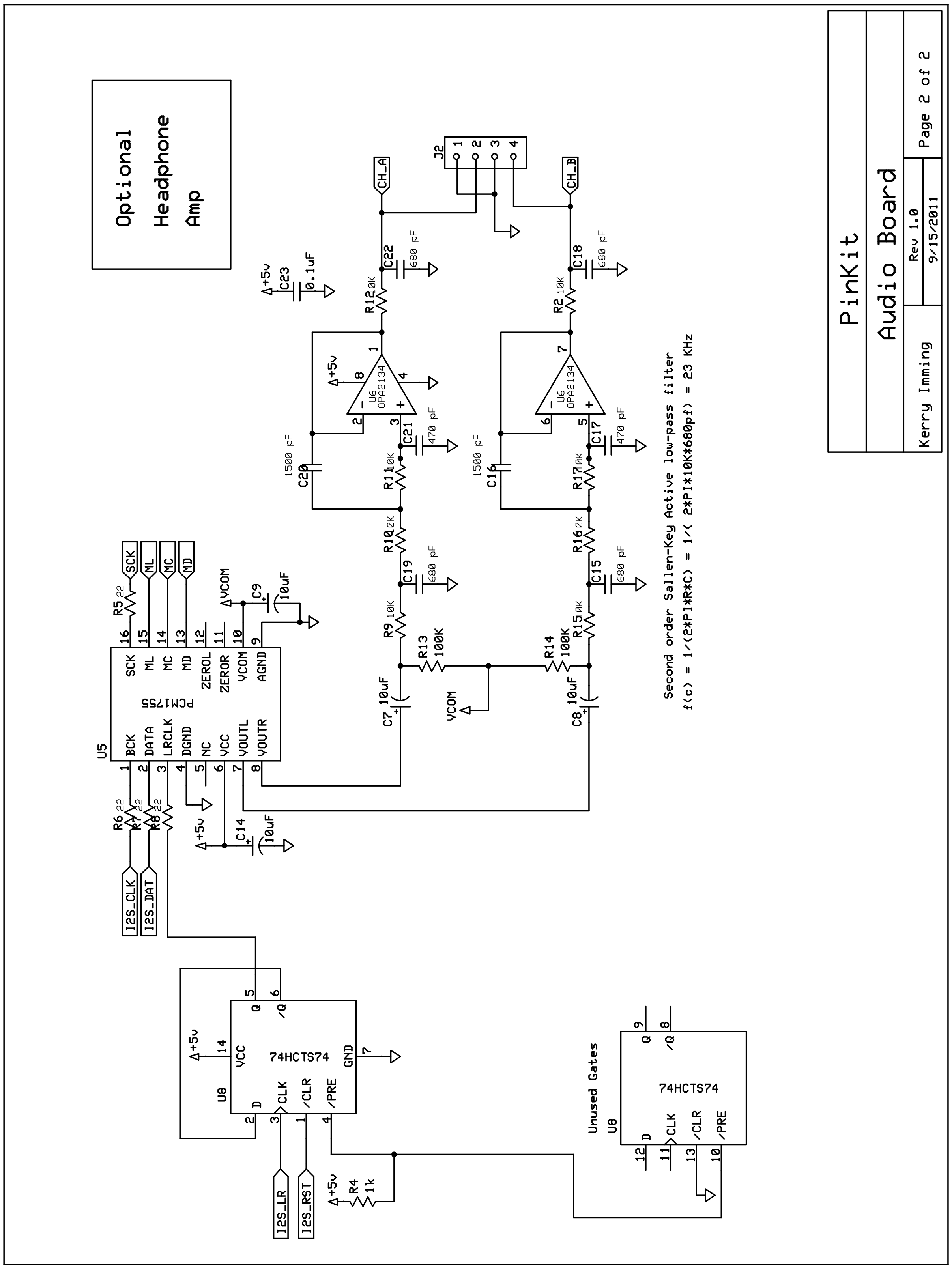
PCB Layout


C1
J1
RN2
J6
D1
RN1
C3
U3
R1
R24
C5
C2
R3
C15
C10
U1
X1
C16
C17
U2
C8
C9
C4
J8
C23
J7
C18
R5
R6
R7
R8
U5
U6
J9
C12
U4
C11
R14
C13
C7
C14
C22
SW1
D2
U8
C21
C20
J2
J5
C6
J4
J3
U7
R4
C19
Rev 1.1
PK_Audio (C)2012 Lmming

Schematics (1 of 2)



Schematics (2 of 2)



Bill of Materials

Reference	Component	Part #	Qty
C1, C9, C12, C13, C14	10 uF, 50V, 105C (0.098”)	493-3281-ND	5
C2, C3, C10, C11, C23	0.1 uF (0.1”)	399-4264-ND	5
C23	0.015 uF (0.1”)	C320C153K5R5TA	1
C4, C5	20 pf (0.098”)	490-3703-ND	2
C7, C8	10uF, Bipolar	UES1V100MEM-ND	2
C15, C18, C19, C22	680 pF	399-4191-ND	4
C16, C20	1500 pF	399-4269-ND	2
C17, C21	470 pF	399-4182-ND	2
D1	LED, red T-1	67-1066-ND	1
D2	1N4004	1N4004FSCT	1
J1	0.156” Header, 6 pin	WM4624-ND	1
J2	RCA Jack, dual, red/white	RCJ-2123	1
J3, J4 (ICSP – optional)	0.100” rt angle header, 6 pin	WM4104-ND	2
J5	RJ45 1x2 w/LEDs	RJHSE-5381-02-ND	1
J6	0.100” Header, 6 pin	WM4204-ND	1
J7	MicroSD Socket	Kyocera 5138	1
R1	470 ohm, 1/8 Watt	CF18JT470RCT-ND	1
R2, R3, R9-R12, R15, R16, R17	10K ohm, 1/8 Watt	CF18JT10K0CT-ND	9
R4, R21-R24	1K ohm, 1/8 Watt	CF18JT1K00CT-ND	5
R5-R8	22 ohm, 1/8 Watt	CF18JT22R0CT-ND	4
R13, R14	100K ohm, 1/8 Watt	CF18JT100KCT-ND	2
RN1	820 ohm RES – 6 SIP	4606X-1-821LF-ND	1
RN2	56K ohm RES – 6 SIP	4606X-101-563LF	1
SW1	4-pos DIP Switch	450-1364-ND	1
U1	PIC18F2580	PIC18F2580-I/SP-ND	1
U2	MCP2551 CAN HI-SPD 8-DIP	MCP2551-I/P-ND	1
U3	ESD Protection Array – ST Micro ESDA6V1-5SC6 (SOT-23-6)	497-7747-1-ND	1
U4	PIC24FJ64GA001 (DIP-28)	PIC24FJ64GA002-E/SP-ND	1
U5	PCM1755DBQ (SSOP-16)	296-16653-5-ND	1
U6	OPA2134 (PDIP-8)	OPA2134PA-ND	1
U7	LD1117V33, 3.3V regulator	497-1491-5-ND	1
U8	74HCTS74[2] (SOIC-14)		1
X1	20 Mhz crystal	300-8507-ND	1
PCB	PCB, 3.8 x 2.5”	PK_Audio_v1p1	1

[2] SSOP-14 on pcb rev 1.0

Pin Assignments

Pin	Description	Notes
J1-1	Ground	
J1-2	+5V	Card Logic & Switches
J1-3	+12V	Lamps (Unused)
J1-4	NC	
J1-5	Ground	
J1-6	+24V	Coils (Unused)
J2-1	Ground	
J2-2	Channel A	Line out
J2-3	Ground	
J2-4	Channel B	Line out
J3	ICSP	PIC24F programming I/F, PicKit2 compatible
J4	ICSP	PIC18 programming I/F, PicKit2 compatible
J5A	RJ-45	PinCAN
J5B	RJ-45	PinCAN

Errata

PK_Audio Revision 1.0 Errata

Rev 1.0 of the PK_Audio board is fully functional, but the following problems are corrected in Rev 1.1.

1. Missing +5V connection from J5 to bulk of supply distribution. A jumper wire must be added from J4 pin 2 to U2 pin 3 to support PinCAN powering of audio board.
2. Solder mask on side support tabs of J7 (MicroSD) socket removed. On 1.0 these must be scraped before mounting the socket.
3. Solder tabs for pins extended to make it easier to solder manually.
4. Change U8 from SSOP-14 to SOIC-14 to make it easier to solder manually.
5. Increase spacing between C12 and U7
6. Increase spacing between C8 and C9

Engineering Notes:

4-bit ADPCM = 2 samples/byte

31250 samples/sec * bytes/(2 samples) = 15625 bytes / sec * 3 channels = 46875 bytes/sec

SparkFun Electronics makes an “RS232 Shifter SMD”, sku: PRT-00449, that can interface standard RS-232 voltage levels to the 3.3V levels used on PK-Audio. Schematics are provided if you want to build one, but theirs is reasonably priced. http://www.sparkfun.com/products/449

Interface Levels:
The PinKit Audio board is powered by 5V DC, but an on-board regulator is used to power the PIC24 and MicroSD card at 3.3V. The following table summarizes the I/O voltage levels to check for proper signaling between the various components.

Parameter	**PIC18[3](5V)**	**PIC24F(3.3V)**	**74HCT(5V)**	**PCM1755(5V)**	
Vil	0.8V	0.67 V (0.2*vdd)	0.8V	0.8V	
Vih	2.0V	2.64 V (0.8*vdd)	2.0V	2.0V	
Vol	≤0.4V	≤ 0.4 V	≤0.4V	≤0.4V	
Voh	≥2.4V	≥ 2.75 V	≥2.4V	≥2.4V	
f(max)	40MHz	32MHz	22MHz	32MHz?	

References

1. I2S Bus Specification, Phillips Semiconductors, June 5, 1996 Revision.
2. Connecting the Atmel ARM-based Serial Synchronous Controller (SSC) to an I2S-compatible Serial Bus
   http://atmel.com/dyn/resources/prod_documents/doc6020.pdf
3. SPI to I2S Using MAX II CPLDs http://www.altera.com/literature/an/an487.pdf
4. Interfacing an I2S Device to an MSP430 Device
   http://www.ti.com/lit/an/slaa449a/slaa449a.pdf
5. SD Specifications Part 1, Physical Layer Simplified Specification Version 2.00, September 25, 2006
6. PIC24F Family Reference Manual
   This is only available as a set of individual section documents. Reference online

[3] PIC18F2580 datasheet, section 28.3

by searching for “PIC24F Family Reference Manual” on www.microchip.com. Or for more direct access, web search for “PIC24F Family Reference Manual”.

7. “PIC24FJ64GA004 Family Data Sheet”, Microchip, DS39881D, 2010
8. “Adaptive Differential Pulse Code Modulation using PICmicro™ Microcontrollers”, Application note AN643, Microchip, http://ww1.microchip.com/downloads/en/AppNotes/00643c.pdf