

Entry AR1754: Media Player

Abstract

Media Player is an embedded hardware/software solution for 16-bit digital audio. The project has three major functional blocks:

- 1) LPC2138 micro controller
- 2) SD card interface
- 3) Audio DAC interface

The prototype (see photo 1) has been built around an LPC2138 evaluation board and two extra prototype boards: one for audio DAC (photo 2) and one for the SD card (photo 3).

System timing analysis:

To achieve the audio CD quality of sound (16 bit stereo with 44.1 kHz sampling frequency) the required bit rate is:

16 bit (resolution) x 2 channels (stereo) x 44.1 kHz (Fs) = 1411200 Hz

This is the clock to be generated by SSP, supplying data continuously to an audio DAC. The data has to be read from the file on an SD card. The file is in a PC compatible format. The data has to be read and processed in real time.

The processing required is very simple for uncompressed audio and more complex for compressed audio, like MP3, which requires complex decoding. As the above numbers plus the micro controller specifications show, it is definitely possible to implement a 16 bit audio replay from an SD card on an LPC2138 micro controller running at CLK=60MHz, for uncompressed audio (such as in WAV format).

There are commercial libraries for mp3 decoding on ARM7TDMI, but the author chose to check the performance of an open source code implementation.

Summary

The project shows that it is easy to implement the replay of uncompressed audio data of good quality on an LPC2138 micro controller.

The processing power of ARM7TDMI core allows for real time digital signal processing like mp3 decoding. With some extra RAM (available on LPC2148) a very good quality audio player can be built for a little cost.

The open source code, plus free GNU, or low cost compilers are very beneficial in terms of saving time and money.

Pictures

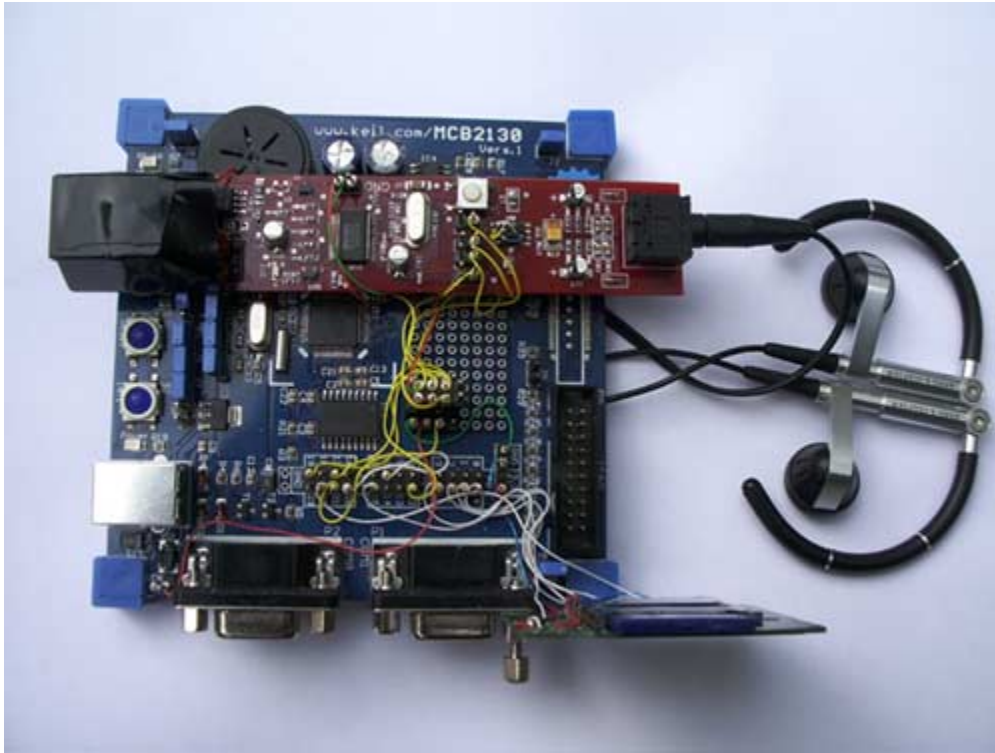


Photo 1: The working prototype of Media Player project AR1754

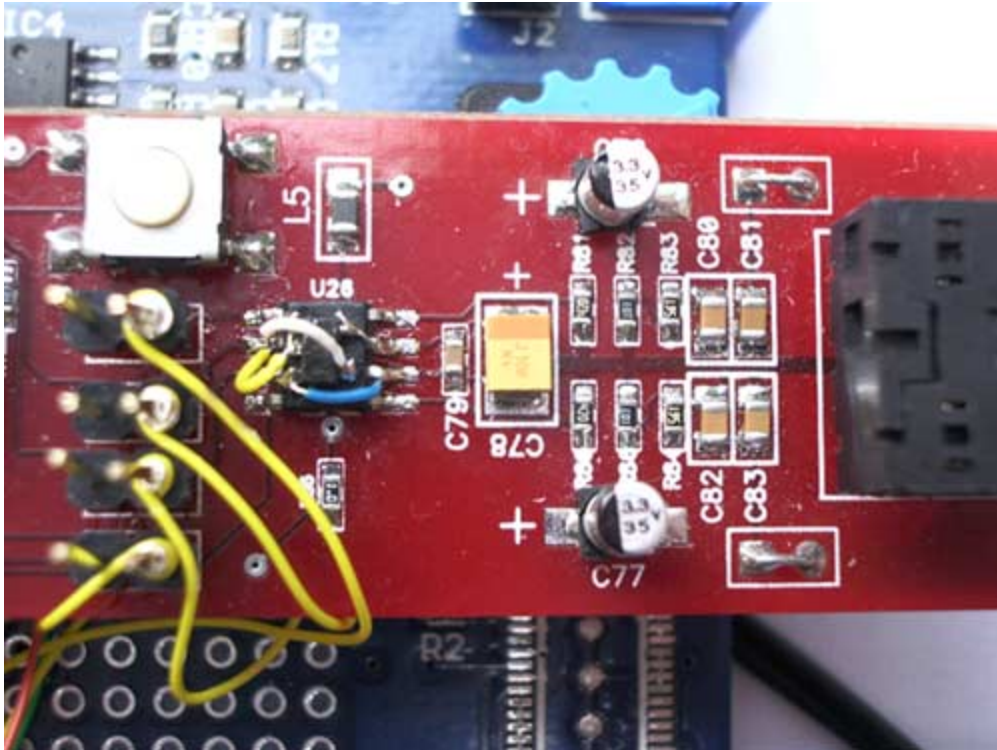
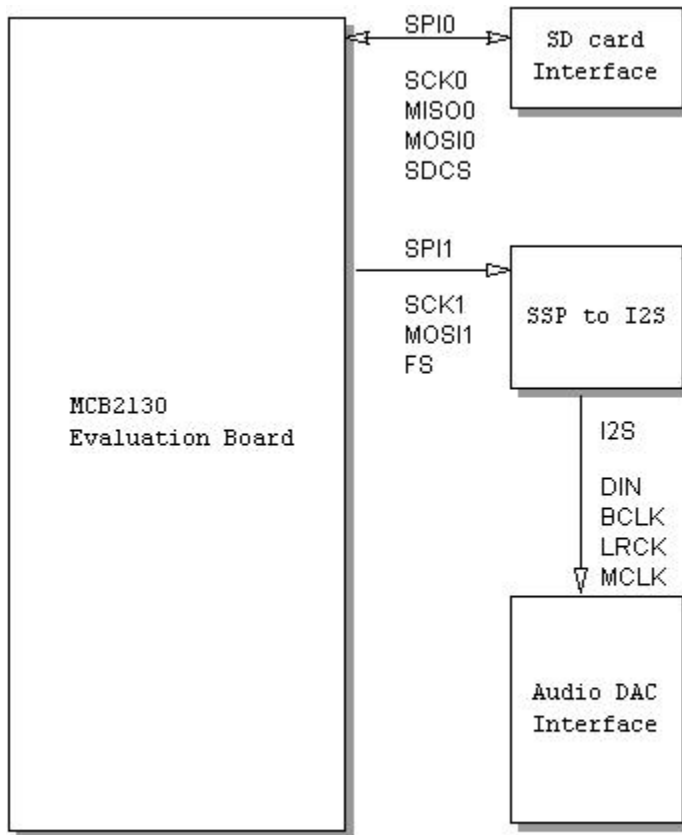


Photo 2: The DAC circuit based on CS4330 (NC7S14M5 single gate Schmitt inverter on top of it)



Photo 3: SD card interface

Block Diagram



Hardware Block Diagram

Sample Code

An interrupt for TX FIFO half empty is used. As *.wav data comes in little endian, the byte order is reversed here to feed 16 bit into SSP FIFO. Four 16-bit frames are loaded into FIFO (being 2 left and 2 right audio samples).

```
unsigned char bb[BIG_SIZE]; // big buffer - data written here from
                             // SD card and data are read from
                             // here for SSP (DAC)

typedef union { // used to change byte order
    unsigned char b8[4];
    unsigned short b16[2];
    unsigned int b32;
} wave_data;

wave_data left; // left audio sample
wave_data right; // right audio sample
wave_data left1;
wave_data right1;

void sspISR(void) __attribute__((interrupt ("IRQ")));

void sspISR(void)
{
    unsigned char c;
    c = SSPMIS;
    SSPICR = 0x03; // reset INT errors if any

    left.b8[0]=bb[play_index++];
    left.b8[1]=bb[play_index++];
    right.b8[0]=bb[play_index++];
    right.b8[1]=bb[play_index++];

    left1.b8[0]=bb[play_index++];
    left1.b8[1]=bb[play_index++];
    right1.b8[0]=bb[play_index++];
    right1.b8[1]=bb[play_index++];

    SSPDR = left.b16[0];
    SSPDR = right.b16[0];
    SSPDR = left1.b16[0];
    SSPDR = right1.b16[0];

    if(play_index >= BIG_SIZE) play_index = 0;

    VICVectAddr = 0; /* Update VIC priorities */
}
/*-----*/
/* End of Function: sspISR */
```

Schematic Diagram

