

Playing The Imperial March

From Texas Instruments Wiki

The #evp parser function was deprecated in EmbedVideo 2.0. Please convert your parser function tag to #ev.

In this tutorial we will learn how to generate sounds and tunes with the Texas Instruments MSP430 Launchpad.

Contents

- 1 Quick Start
- 2 Development
 - 2.1 Schematic
 - 2.2 Code
- 3 Theoretical Background
- 4 Future Ideas
- 5 References
- 6 Related Projects

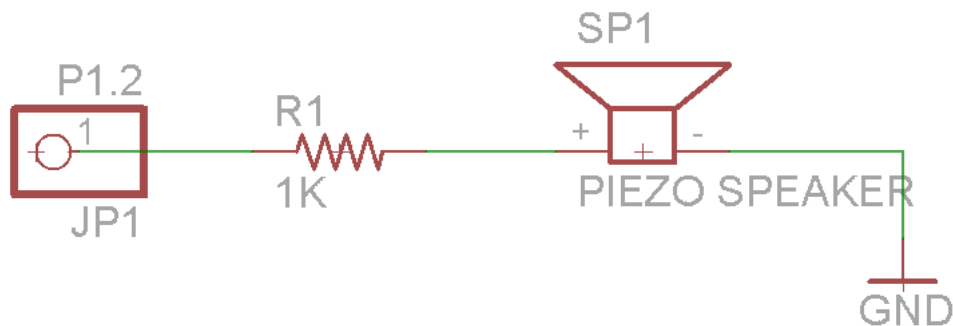
Quick Start

- Get a piezo speaker, like the ones that can be found in an old computer.
- Get a 1KOhm resistor.
- Download Code/Build the circuit.
- Enjoy the John Williams' masterpiece!

Development

Schematic

The circuit is really simple to build. Solder the 1KOhm resistor to the positive lead of the speaker (red wire) and connect it to P1.2 on the Launchpad. Then connect the black wire to the GND pin on the board. You can even remove the R1, but the sound will be very loud and annoying, and the speaker may be damaged. You can add a 1K trimmer instead of the resistor to make the volume adjustable.



Code

Here is the fully commented code for this tutorial. I have ported it to the MSP430G2231 from an Arduino example made by naneau (<http://www.youtube.com/watch?gl=AU&hl=en-GB&v=uLPGTMISJIY>).

File:Lauchpad Imperial March.zip

```
#include "msp430g2231.h"

//Definition of the notes' frequencies in Hertz.
#define c 261
#define d 294
#define e 329
#define f 349
#define g 391
#define gS 415
#define a 440
#define aS 455
#define b 466
#define cH 523
#define cSH 554
#define dH 587
#define dSH 622
#define eH 659
#define fH 698
#define fSH 740
#define gH 784
#define gSH 830
#define aH 880
```

```

/* This two functions stop the main thread for a certain number of milli -or- microseconds.
   They are based on trial and error, but they work fine for the out-of-the-box Launchpad board.
   TI should really add this types of functions as default, just like Arduino does :) .
*/
void delay_ms(unsigned int ms )
{
    unsigned int i;
    for (i = 0; i<= ms; i++)
        __delay_cycles(500); //Built-in function that suspends the execution for 500 cycles
}

void delay_us(unsigned int us )
{
    unsigned int i;
    for (i = 0; i<= us/2; i++)
        __delay_cycles(1);
}

//This function generates the square wave that makes the piezo speaker sound at a determinated frequency.
void beep(unsigned int note, unsigned int duration)
{
    int i;
    long delay = (long)(10000/note); //This is the semiperiod of each note.
    long time = (long)((duration*100)/(delay*2)); //This is how much time we need to spend on the note.
    for (i=0;i<time;i++)
    {
        P1OUT |= BIT2; //Set P1.2...
        delay_us(delay); //...for a semiperiod...
        P1OUT &= ~BIT2; //...then reset it...
        delay_us(delay); //...for the other semiperiod.
    }
    delay_ms(20); //Add a little delay to separate the single notes
}

//This is the Imperial March code.
//As you can see, there are lots of beeps at different frequencies and durations, and some delays to separate the various bits of this wonderful song.
void play()
{
    beep(a, 500);
    beep(a, 500);
    beep(a, 500);
    beep(f, 350);
    beep(cH, 150);
    beep(a, 500);
    beep(f, 350);
    beep(cH, 150);
    beep(a, 650);

    delay_ms(150);
    //end of first bit

    beep(eH, 500);
    beep(eH, 500);
    beep(eH, 500);
    beep(fH, 350);
    beep(cH, 150);
    beep(gS, 500);
    beep(f, 350);
    beep(cH, 150);
    beep(a, 650);

    delay_ms(150);
    //end of second bit...

    beep(aH, 500);
    beep(a, 300);
    beep(a, 150);
    beep(aH, 400);
    beep(gSH, 200);
    beep(gH, 200);
    beep(fSH, 125);
    beep(fH, 125);
    beep(fSH, 250);

    delay_ms(250);

    beep(aS, 250);
    beep(dSH, 400);
    beep(dH, 200);
    beep(cSH, 200);
    beep(cH, 125);
    beep(b, 125);
    beep(cH, 250);

    delay_ms(250);

    beep(f, 125);
    beep(gS, 500);
    beep(f, 375);
    beep(a, 125);
    beep(cH, 500);
    beep(a, 375);
    beep(cH, 125);
    beep(eH, 650);

    //end of third bit... (Though it doesn't play well)
    //Let's repeat it

    beep(aH, 500);
    beep(a, 300);
    beep(a, 150);
    beep(aH, 400);
    beep(gSH, 200);
    beep(gH, 200);
}

```

```

beep(fSH, 125);
beep(fH, 125);
beep(fSH, 250);

delay_ms(250);

beep(aS, 250);
beep(dSH, 400);
beep(dH, 200);
beep(cSH, 200);
beep(cH, 125);
beep(b, 125);
beep(cH, 250);

delay_ms(250);

beep(f, 250);
beep(gS, 500);
beep(f, 375);
beep(cH, 125);
beep(a, 500);
beep(f, 375);
beep(cH, 125);
beep(a, 650);
//end of the song
}

int main( void )
{
    WDTCTL = WDTPW + WDTHOLD; //Disable Watchdog Timer
    P1DIR|=BIT2;              // P1.2 output
    while(1)
    {
        play();
        delay_ms(2000);      //Add a 2 sec. delay to avoid replaying right after the end.
    }
}

```

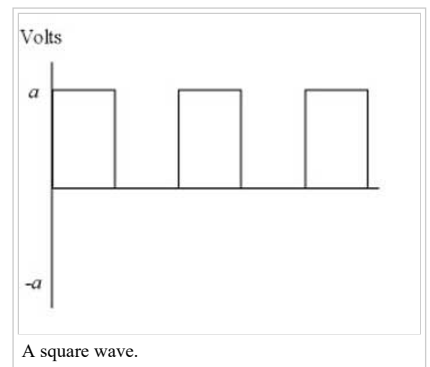
Theoretical Background

A Pulse Width Modulation (PWM (http://en.wikipedia.org/wiki/Pulse-width_modulation)) output can be used to play tones on a piezo speaker. With this, musical scales and simple songs can be played on the piezo speaker.

Piezo speakers operate by the converse piezoelectric effect (<http://en.wikipedia.org/wiki/Piezoelectricity>): when a voltage is applied across the terminals, the piezoelectric material in the speaker deflects in one direction. Applying an alternating voltage, such as a square wave, will cause the material to vibrate and create a sound. A constant voltage will not produce a sound. Also sine, triangle and sawtooth waves can be used to drive a piezo speaker, resulting in different pitches of the notes.

As you can see from the code, the beep() function was used to provide the alternating voltage to drive the speaker. The period (do you remember delay_us() ?) of the square wave determines the note that will be played.

You can have some problems about the clock and timings with this project, I'm still figuring out the MSP430 Clock System as it is the first time I use them.



Future Ideas


- You might want to play sounds from an SD Card, like it has been done here (<http://www.diyliife.com/2008/04/25/make-a-talking-msp430-microcontroller-part-3/>). (I'm working on it, but the sound is really distorted, I'm still wondering why...)

References

1. MSP430x2xx Family User Guide (<http://www.ti.com/litv/pdf/slau144e>)
2. MSP430G2x31 Datasheet (<http://www.ti.com/lit/gpn/msp430g2231>)
3. Naneau's Arduino Imperial March (<http://www.youtube.com/watch?gl=AU&hl=en-GB&v=uLPGTMISJIY>)
4. Make a talking MSP430 MCU (<http://www.diyliife.com/2008/04/25/make-a-talking-msp430-microcontroller-part-3/>)

Related Projects

- MSP430 LaunchPad Learning Community
- MSP430 LaunchPad Drive LED

 <p>Engage in the TI E2E Community Ask questions, share knowledge, explore ideas and help solve problems with fellow engineers</p>	<p>For technical support on MSP430 please post your questions on The MSP430 Forum (http://e2e.ti.com/support/microcontrollers/msp43016-bit_ultra-low_power_mcus/default.aspx). Please post only comments about the article <i>Playing The Imperial March</i> here.</p>
---	--

Links

Amplifiers & Linear
(http://www.ti.com/lstds/ti/analog/amplifier_and_linear.page)
Audio (http://www.ti.com/lstds/ti/analog/audio/audio_overview.page)
Broadband RF/IF & Digital Radio
(<http://www.ti.com/lstds/ti/analog/rfif.page>)
Clocks & Timers
(http://www.ti.com/lstds/ti/analog/clocksandtimers/clocks_and_timers.page)
Data Converters
(http://www.ti.com/lstds/ti/analog/dataconverters/data_converter.page)

DLP & MEMS (<http://www.ti.com/lstds/ti/analog/mems/mems.page>)
High-Reliability (http://www.ti.com/lstds/ti/analog/high_reliability.page)
Interface (<http://www.ti.com/lstds/ti/analog/interface/interface.page>)
Logic (http://www.ti.com/lstds/ti/logic/home_overview.page)
Power Management
(http://www.ti.com/lstds/ti/analog/powermanagement/power_portal.page)

Processors
(http://www.ti.com/lstds/ti/processors/processors_overview.page)

- ARM Processors (http://www.ti.com/lstds/ti/processors/arm_processors_overview.page)
- Digital Signal Processors (http://www.ti.com/lstds/ti/processors/dsp_processors_overview.page)
- Microcontrollers (http://www.ti.com/lstds/ti/processors/microcontrollers_overview.page)

Retrieved from "http://processors.wiki.ti.com/index.php?title=Playing_The_Imperial_March&oldid=182027"

Categories: MSP430 | Microcontroller Projects - MSP430

- This page was last modified on 21 July 2014, at 08:39.
- This page has been accessed 15,993 times.
- Content is available under Creative Commons Attribution-ShareAlike unless otherwise noted.