

Lab #8

TECH 3233

Ver 1.6

Overview of Lab:

We are going to use Timer Output Compare (OCR1A) and its associated pin (PB1) to generate the song “The Jeopardy Theme”. The program will adjust the frequency for each note and the time that note is played.

Here is the sheet music for the song:

JEOPARDY THEME
from the Television Game Show

Music by MERV GRIFFIN

Moderately



Copyright © 1984 Sony/ATV Songs LLC
This arrangement Copyright © 2005 Sony/ATV Songs LLC
All Rights Administered by Sony/ATV Music Publishing, 8 Music Square West, Nashville, TN 37203
International Copyright Secured All Rights Reserved

First we need to determine what notes are used in the song. Use the table below to mark each note in the sheet music above with the note.

Treble Clef



C4 D4 E4 F4 G4 A4 B4 C5 D5 E5 F5 G5 A5 B5 C6

So in the case of the first note (for Jeopardy) it would be an C5.

Once that is done, now you can determine the frequency (in Hz) of the signal you need to generate is for each one of the notes using the table below (all Frequencies are in Hz):

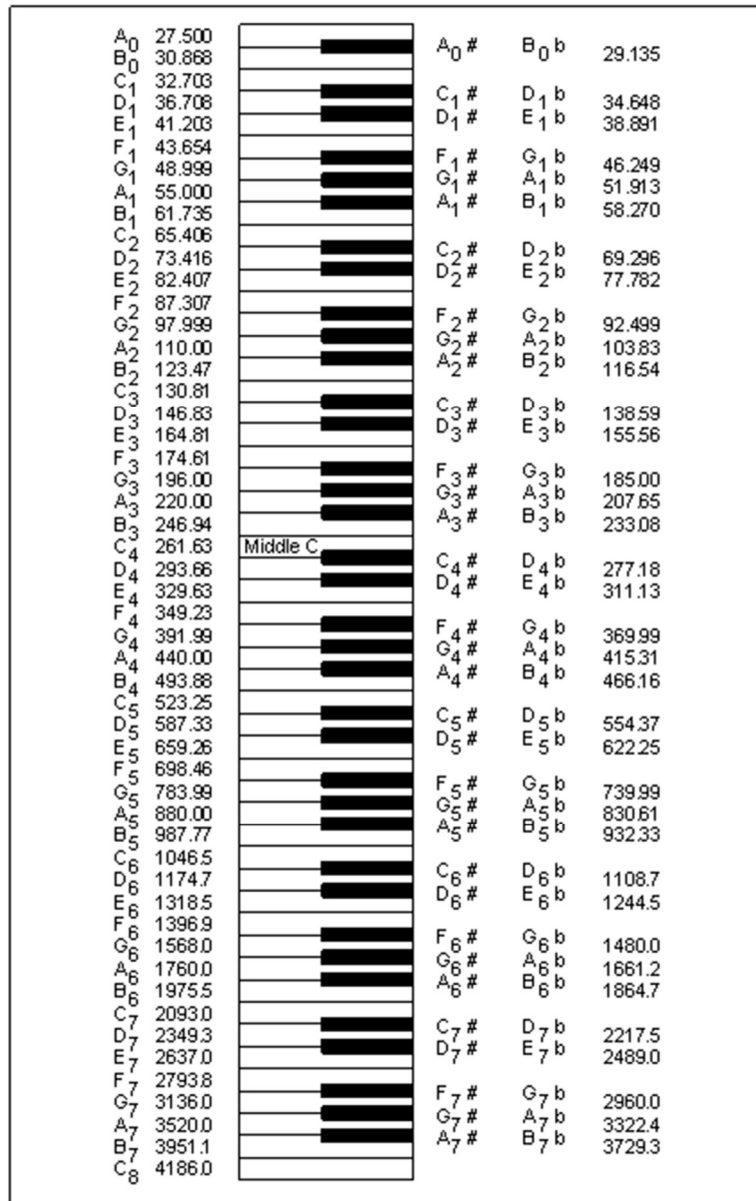
























Figure 13-5. Piano Note Frequencies

The last piece of information is how long each note needs to be held for. This is determined by the type of note as shown in the sheet music as per the note types below:

<u>Note</u>		<u>Rest</u>	Dotted Notes	
	Whole		Dotted whole note	 =  + 
	Half		Dotted half note	 =  + 
	Quarter		Dotted quarter note	 =  + 
	Eighth		Dotted eighth note	 =  + 
	Sixteenth		<p>A dot increases the length of a note by 1/2 its value. Dotted whole note = 6 beats. Dotted 1/2 note = 3 beats. Dotted 1/4 note = 1 1/2 beats. Dotted 8th note = 3/4 beat.</p>	

Note since a whole note is 8x as long as an eighth note, and an eighth note is the shortest note in the piece, it is a good idea to set a constant for the time of a the quarter note, so if the song is going too fast or too slow, you can change it in one place to change the speed of the song.

Now, knowing the frequencies you need to generate, figure out what pre scaler you need for the Atmel238p timer.

Calculate the number of counts needed to generate the 50% duty cycle square wave necessary to generate the frequency of each note.

Create two arrays, one for the timer counts and one for the length (1=eighth note, 2=quarter note, 4=half note and 8 being full note). It is a good idea, for readability of the code, to define the counts for each note and use that name when filling the array:

```
#define c6 1568
#define d6 1174

freq[4]={c6,d6,c6,c6};
time[4]={2,4,4,1};
```

NOTE – for the above example the notes in the above were defined as Hz not as counts!

Lastly, since the note to play (ie the frequency to generate) automatically gets generated by the Output Compare of the Atmel IC, we just have to pause the program for how long the note needs to play. It would be easy to just do a `_delay_ms(tempo*time)` to do this, but UNFORTUNATELY, the way Atmel Studio does the `_delay_ms()` function, it must be a constant in the `()`. So you need to make your own function that does the base (tempo) time delay for a number of times then returns to solve this issue. So create a function that does this and name it `t_delay` and pass it an int.

To test, connect the Arduino output to a set of computer speakers (Gnd on Arduino to Ground on the 3.5mm Jack and PB1 to the Right or Left Input on the 3.5mm jack).



Figure 1 - 3.5mm Jack Connections