Lab 7: Output Timing and Making Music!

OBJECTIVES

- Learn how to use the C programming language on an ATMEL XMEGA device.
- Learn how to use XMEGA's timer system (output compare).
- Experiment with a timer and output compare system to generate digital waveforms (We'll use the timer system to make 'digital' music!)

REQUIRED MATERIALS

- EEL4744 Board kit and tools
- Working Keypad or Serial Interface (from previous labs)
- Working LCD circuits (from previous lab)
- 1 - Piezo Speaker
- XMEGA documents
  - doc8331 (Section 14)
  - doc8385
  - doc8045: *AVR1306: Using the XMEGA Timer/Counter*
- Digilent Analog Discovery (DAD) kit

YOU WILL NOT BE ALLOWED INTO YOUR LAB SECTION WITHOUT THE REQUIRED PRE-LAB.

PRELAB REQUIREMENTS

REMEMBER:
You must adhere to the *Lab Rules and* Policies document for **every** lab.

INTRODUCTION
Music is generated by combining a series of **tones**, each with a distinct **duration**. Each of the tones generated by most non-synthesized instruments has components of a multitude of frequencies, but generally one dominant frequency, which is referred to as the **note**. Band or orchestra music is created by combining the sounds created by each of several instruments (possibly including voices) to create a desirable sound.

In this lab, you will generate several tones by toggling a single pin to a speaker. Because we will generate square waves, the creation of each note will also produce odd harmonics of that note, e.g., if a 1 kHz note was generated, we would also hear successively lower volumes of notes at 3 kHz, 5 kHz, 7 kHz, etc.

In Part A of the lab, you will generate a single note that will play indefinitely while a switch input is true. In Part B, you will give a single note a distinct duration, and then combine notes with durations to make play a few songs (or scales).

PART A – TIMER & OUTPUT COMPARE

In this section you will learn how to use XMEGA's Timer/Counter (TC) system for Output Compare (OC) operations. Specifically, you will use an OC function to generate a G_6 musical note, which is simply a note with a frequency at 1567.98 Hz.

Musical notes can be generated by supplying a pulse train at the appropriate frequency to a speaker. The pulse train is composed of binary 0's and 1's (0V and +3.3V). The XMEGA's TC system can be used to generate a pulse train at a given frequency simply by toggling a pin at a constant rate. For our purposes, we will use a piezo speaker driven directly by the XMEGA's output pin.

Your XMEGA processor has two TCs per port (PORTC through PORTF), with a total of eight TCs. Each port has two types of TCs: Type 0 and Type 1. Both types have identical functions with exception that Type 0 has four capture/compare channels (CCA-CCD) while Type 1 has only two channels (CCA, CCB). In addition, TC of Type 0 can be split from 16-bits into two 8-bit counters; however this advanced feature will not be used in this lab. Each timer module is comprised of several configuration registers. We are most interested in the CTRL, CCx, INTCTRLx, CNT, CTRLyCLR/CTRLySET, and PER registers (where x= C, D, E, or F [for Port] and y=0, 1, or 2 [for Timer])

The CNT registers (e.g., TCF0_CNT) hold the actually count of the timer we are using. You have direct access to this timer which could prove useful for certain applications.

The CTRL registers (CTRLA through CTRLE) are used for configuring the timers. You will use these registers to select the speed of the timer's clock, the mode the timer will operate in, and to enable specific functions. CTRLB can be used as a sort of sound enable.

The CCx registers (CC = compare or capture) will only be used if the timer's CTRL registers are correctly set up to do so. You have to enable the compare registers and also select the correct waveform generation mode (FRQ or PWM). The CCx registers allow us to automatically generate a square wave with a specified frequency. In the FRQ mode, the value in the CCx registers is constantly compared to the CNT of the timer. When a match occurs, the corresponding OCx pin will be toggled and the CNT register will be reset to 0. This creates an easy way to generate a square wave at any frequency to create our music! No interrupt or processing is needed.

For precise time duration (or waveform generation, but not frequency generation), interrupts are required.

Lab 7: Output Timing and Making Music!

The INTCTRLx registers configure the interrupts associated with the timer module. We will need these to turn on and select the interrupt level for our interrupts.

The CTRLxCLR/CTRLxSET registers (e.g., TCF0_CTRLFSET and TCF0_CTRLGCLR) can be used to force a change in the timer. You can force an update of a register, reset the timer, and even change the direction of the timer count.

The PER registers (e.g., TCE1_PER) are used to set the "TOP" of the timer's count. Depending on the mode of operation, the timer's CNT registers are constantly compared to the PER register. (You can change which register acts as the "TOP" for the counter.) When a match occurs, the timer is reset to 0. An interrupt can be triggered on this matching event; it is enabled by turning on the overflow/underflow interrupt (in INTCTRLA). You will need to do this in PART2 to successfully complete this lab.

In this section, you will set the timer so that one of the output compare pins toggles at required frequency necessary to produce a musical note (tone).

1. Carefully read sections 14.1-14.4, 14.6, 14.8, and 14.12 in the XMEGA doc8331 manual. You may also consult doc8045 (*AVR1306: Using the XMEGA Timer/Counter*) for general TC information.
2. Choose any of the eight available TCs to be used in this lab. Consult doc8385, section 33.2, for pin-out information in order to find the corresponding OCx pin for your speaker. For example, OC1A will correspond to timer type 1 and its CCA register, and is available on ports C-F Note that when programming in C, you should refer to Timer/Counter registers as TCxy.REGISTER, where x stands for port (C-F) and y stands for type (0 or 1).
3. Connect a piezo speaker to your board. Unlike 8 Ω speakers, a piezo speaker can be driven directly from a pin on XMEGA (e.g., an OCx pin). Since the piezo speaker in your kit is polarized, connect the positive end (marked 'M') to the XMEGA timer output that was chosen in step 2 and the negative end (unmarked) to ground.
4. Configure the chosen TC to continuously play a 1567.98 Hz square wave (G_6) on the piezo speaker. There are several configuration registers you will have to be aware of.
   a. The actual 16-bit count value of the timer is held in the CNT (CNTH | CNTL) register. This value is incremented (or decremented) every TC clock cycle and is used by the TC module to perform compare/capture operations. You can access this register directly; this is useful in certain applications.
   b. Period register PER (PERH | PERL) holds the TOP value for the TC's count. During normal operation, the timer's CNT register is continuously compared to the PER register. Whenever counter reaches the TOP value (CNT = PER), an update condition occurs. The CNT register is then reset to zero (single slope operation) and TC starts counting again. Note that in dual slope operation, TC counts up and down with update occurring at the BOTTOM (CNT=0).
   c. The control registers (CTRLA through CTRLE) are used for configuring the TC. You will use these registers to select the clock source, the mode the timer will operate in, and to enable specific functions.
   d. The CCx registers (CC = compare or capture) are used to generate output waveforms (output compare) or measure input signals (input capture) when the CTRL registers are configured correctly. For this lab, you will use them in FRQ or PWM mode (see 14.8.2-14.8.4). In either mode, whenever counter value equals the compare value (CNT = CCx), a "match" condition occurs and corresponding output pin (OCx) is toggled.
   e. Whenever the match or update condition occurs, an interrupt may be issued. INTCTRLA and INTCTRLB registers are used for setting interrupt levels to various TC events. You can use this feature to create very accurate delays (as you will use in part B).

Before setting any TC registers, select an output pin and port (as described in step 2). See section 33.2 in the doc8385 to find the corresponding OCx pin that you wired to your piezo. **Don't forget to set this pin as an output in your initializations!**

Write a program that generates the **G_6** note using the TC system. The program should play the note continuously while a switch input is true. Verify the frequency of the note using your DAD using the Spectrum Analyzer and/or the oscilloscope function. Include a screenshot of the DAD oscilloscope output in the lab document.

PART B – PUTTING IT ALL TOGETHER

Remember those tiny musical keyboards that many of us enjoyed when we were young? Well, now it is time to make one of your own!

Table 1 shows the frequency assignment for each key in the keypad (or serial keyboard with TA permission). Pressing any of keys 1-D should play the note for 200 ms = 0.200 s. Keys '*' and '#' are "special keys." Pressing '*' should play the **ascending** C major scale (C_6, D_6, E_6, F_6, G_6, A_6, B_6, C_7), i.e., play each note in the sequence for 474.4 ms. Pressing '#' should play the **descending** C

Lab 7: Output Timing and Making Music!

major scale (C_7, B_6, A_6, G_6, F_6, E_6, D_6, C_6); producing each note for 370.1 ms.

If you want to check the tones with your cell phone, there are a bunch of applications (apps) that you can download. On a computer, the following website can be used to play a tone: http://onlinetonegenerator.com/. You can also use your DAD's digital output to send the correct frequency square wave to the piezo electric speaker. If your speaker is already attached to your board turn your power off first, attach ground to the ground of the piezo speaker and put the digital output on the other side. You could also use the analog output to see how different types of waveforms (Such as sign and Saw-tooth) sound.

A full table of frequencies for musical notes can be found at http://www.phy.mtu.edu/~suits/notefreqs.html.

You also have the option of making the special keys play any song you like instead of the ascending and descending scales. Sample sheet music for various songs can be found at the below URL.

http://www.gmajormusictheory.org/Freebies/freebies.html

Write your program in the following manner:

a) Determine CNT/PER values required to play all notes in the Table1 – this is just like part A. Remember that we are toggling the bit at twice the frequency (half the period).

b) Set up another TC to create precise note durations for the keypad (200ms, 370.1ms, 474.4ms). It is best to use interrupts for this portion of the program. This TC should trigger an interrupt via the overflow or underflow interrupt. It is recommended to run the timer in normal mode and use the PER registers. Do **NOT** use a delay subroutine. This is not a good approach and you will **NOT** receive credit for doing so.

c) Implement a polling routine to continuously scan the keypad (or an interrupt service routine to read the keyboard). If a key is pressed:

d) Clear the LCD. For keys 1 through D, play the note for 200ms while displaying the name and frequency of the note (as shown in the Table 1). For the special keys, display a message such as "Ascending Scale." Play notes for 474.4ms for ascending scales and 370.1ms for descending scales. For example, the LCD can display:

G#6/Ab6
1661.22 Hz

or

Ascending
Scale

For extra credit consideration (amount TBD), create custom note durations for songs as a replacement for the scales, e.g., "1812 Overture" by "P. I. Tchaikovsky." The LCD could then display something like:

1812 Overture
by Tchaikovsky

or

Queen's Bohemian
Rhapsody

Do not clear the LCD after displaying. It will be cleared when another key is pressed.

When the musical note, scale, or song has finished playing, set the output compare pin to "do nothing" and resume polling the keypad (or waiting for a new key press on the keyboard).

Keypad Key	Note	Frequency (Hz) or Description
1	C_6	1046.50
2	$C^{\#}_6/D^{b}_6$	1108.73
3	D_6	1174.66
4	$D^{\#}_6/E^{b}_6$	1244.51
5	E_6	1318.51
6	F_6	1396.91
7	$F^{\#}_6/G^{b}_6$	1479.98
8	G_6	1567.98
9	$G^{\#}_6/A^{b}_6$	1661.22
0	A_6	1760.00
A	$A^{\#}_6/B^{b}_6$	1864.66
B	B_6	1975.53
C	C_7	2093.00
D	$C^{\#}_7/D^{b}_7$	2217.46
*	--	Ascending scale or song 1
#	--	Descending scale or song 2

Table 1. Keypad keys with corresponding musical note names and corresponding frequencies. (Two musical symbol called sharp and flat look very much like "#" and "b." So $F^{\#}$ is known as "F-sharp" and G^{b} is known as G-flat.)

QUESTIONS

1. Draw a 10 kHz square wave with a 33% duty cycle. What is the period in milliseconds (ms)?
2. How does the prescaler affect the way the TC system counts per clock cycle? Where are the counts stored?
3. For part A, what is the limiting factor for the precision of your frequency generation? Can your XMEGA generate some frequency ranges with higher precisions than other frequency ranges? Explain.
4. Describe the difference(s) between the TC's Frequency Generation mode and its Single/Dual Slope PWM modes. Which mode(s) can be used to emulate the other(s)?How could you make a sine wave or other waveform using your XMEGA? Do you need to add any extra hardware?

PRELAB REQUIREMENTS SUMMARY

1. Build the piezo speaker circuit.

Lab 7: Output Timing and Making Music!

2. Use a TC module to play the requested single frequency note while a switch input is true.
3. Use two TC modules to implement a musical keypad.

IN-LAB REQUIREMENTS

1. If everything works, demonstrate your musical keypad to your TA. Show off your special keys!
2. If part B is not fully functional, demonstrate part A.