COMP 2659 Lab 5 – I/O Programming (Sound)

Copy the `PSG.C` file from the `lab 5 code` subfolder (under the course resources folder) into your working directory. Launch STeem, start the emulation and run Gulam.

You will be reading and modifying code which controls the Atari ST's PSG[1] (Programmable Sound Generator) chip. If you have a pair of headphones, plug them into the output jack of the PC's sound card so that you can hear the results!

1. Briefly review the provided code. What it does and how it works will hopefully be clear by the end of this lab.

   Compile the program, run it and listen (if using a lab PC, you must have headphones plugged into the output jack of the PC's sound card). **WHENEVER USING HEADPHONES, START WITH THE VOLUME AT A LOW SETTING BEFORE YOU BEGIN TO PLAY TONES.**

   You should hear a continuous tone. To end the program, type any key.

   To understand this program, a few basic facts about the PSG must be understood:

   - The PSG has a large number of registers. Instead of mapping each to a separate address (which would require a large number of "register select" pins), the PSG chip designers used a trick: they only mapped two addresses. The first is a "register select" register. The value written here determines which other PSG register will be accessible at the second address.
   - The PSG has three tone generators onboard, named channels A, B and C. Each as a fine tune and a course tune register, which together determine the channel's pitch.
   - Each channel can be switched on or off at the PSG's mixer.
   - Each channel has a volume register.
   - The `Super` system call is used to switch the 68000 between user and supervisor mode. The memory mapped registers are only accessible when the 68000 is in supervisor mode.

   With the above in mind, study the existing code more completely so that you have at least a general understanding of how it works.

   Note: the PSG has additional registers for generating "noise" and shaping a channel's "envelope". These features are often used for generating sound effects, but are not used in the original version of this example program.

2. Experiment with commenting out the `Super` calls. What happens at run-time? Explain why. When finished, uncomment the calls so that the code will work again.

[1] This type of chip is also called an SSG (Software-controlled Sound Generator).

3. On the ST, the PSG is a YM2149 chip. Consult the provided documentation in the appropriate reference subfolder of the course resources folder. There are multiple documents, but begin by skimming `YM2149 Application Manual.pdf`. Focus on the following sections:

• section 1 (introduction)	← page 1
• section 2 (overview of functions)	← pages 2-4
o mainly subsections 2-1 to 2-3	
• section 3 (description of operation)	← pages 13-14, 16-18, 23
o mainly subsections 3-1, 3-3, 3-4 and 3-8	
o ignore the I/O ports described in 3-3	
• start of section 4 (interface)	← first half of page 24

   As you read this material, again review the code in `PSG.C`. Confirm that you understand each detail of the program precisely. If anything is unclear, consult with your instructor before continuing.

   For later, also note the immense usefulness of section 5 (page 31) and 6 (page 34).

4. Consult one of the provided Atari ST memory maps, also found in the reference folder. Locate the section of the memory map which states the addresses of the PSG's registers within the ST's address space.

5. Experiment with different fine tune values. Make the pitch higher. Once you can do this, make the volume quieter.

6. Experiment with changing the coarse tune value. Which has a more significant effect – changing the fine or coarse tune setting?

7. In your own words, how could you write a program that plays a simple, cyclic melody? Describe the basic idea.

8. Modify the code so that some noise is mixed in with the tone (note section 3-2).

9. Instead of playing sounds at a constant volume, modify the code to use the envelope generator (note sections 3-5 and 3-6). Experiment with different envelope frequencies and shapes.

10. **Challenge:** modify the existing code so that it plays a C major chord (musical notes C, E, G played simultaneously).

11. **Challenge:** modify the existing code so that it play the sound of an explosion (i.e. noise that begins loudly and quickly tapers off through the application of an appropriate envelope).