Generating Morse Code Using DDS

Bryan Cheung

Kishan Patel

Josh Skow

Aditya Somani

Raina Tanazzum

ECE 2031, Digital Design Lab

Section L06

Georgia Institute of Technology

School of Electrical and Computer Engineering

Submitted

November 16, 2012

# Executive Summary

This design report details technical information regarding a project that enables Morse code speech to be displayed on an LCD and heard through a speaker. To successfully accomplish the goals of this project, a Direct Digital Synthesis (DDS) system will be used to generate varying frequencies based on the tone that is requested. Robots would find it useful to output messages to people around it, alerting the people to the condition of the robot. Morse code is one solution to this problem, as it could be generated and used to communicate the status of the robot, without the additional complications of speech synthesis. This project will take values from set inputs, either through switches or pushbuttons, and produce a Morse code audio signal through the speaker jack. Since only the DE2 inputs are provided for this project, switches and pushbuttons simulate data that could be collected from other robot peripherals, showing how a Morse code system could be beneficial as a way to relay messages to those around the robot. The DDS used to create the audio signals will have a variable tuning word which will allow for the frequency of output signals to be changed with ease. Tone changes will be simulated for various situations, such as when a robot gets closer to a wall, the pitch will increase. These simulations will be generated through data assigned to switches on the DE2 board, allowing for straightforward testing and clear demonstration of the project.

Generating Morse Code Using DDS

# Introduction

The goal of this project is to create an application that will allow a robot to communicate with people around it. One solution is to use Morse code as the means of communication. This project requires the use of I/O (input and output) peripherals, implementing Direct Digital Synthesis (DDS) techniques, and using audio codec to convert the digital signal to analog form. The output frequencies will be varied to simulate different situations, such as an increase in frequency when the robot approaches a wall.

# Technical Approach

## Design Requirements

The Morse code translator project will operate under the following requirements in order to accomplish the desired goals:

1. Accept text input from a PS/2 keyboard.

2. Store the text in the LCD’s memory location.

3. Translate the text to binary code that represents Morse code.

4. Generate tuning words for a DDS device through switches on the DE2.

5. Use pushbuttons to enter and reset text on the LCD.

## Context for Hardware Devices Used

Since this project is being designed for an existing prototype robot, using a Field-programmable Gate Array (FPGA) to implement the logical functions is the economical choice. The DE2 FPGA was chosen for the hardware flexibility that it offers, as well as the ability to interface with many I/O devices. The usage of Hardware Description Language (HDL) allows for rapid modification of the robot’s functionality, which facilitates testing and debugging of new features, at low-recurring cost.

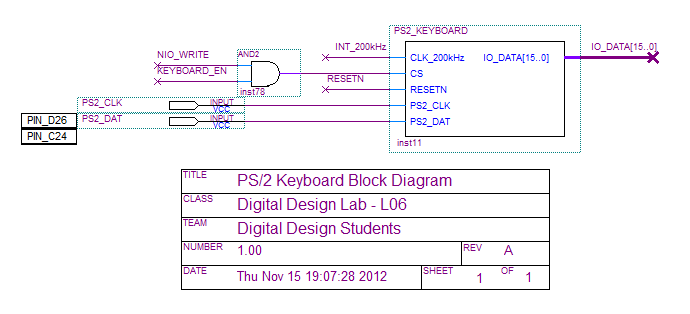
The DE2 houses a LCD screen, a seven-segment display array, and a LED array. These output devices can be used to display data and feedback from the robot to the user. The user can also provide input to the robot through the switches and pushbuttons located on the board. These input devices allow the user to provide commands to the robot, or can represent data provided by external sensors that may be added to the robot. The DE2 also includes standardized ports (such as PS/2 and USB) for additional devices that may further expand the robot’s functionality or user-interface.

Internally, the DE2 will run HDL to simulate a Simple Computer (SCOMP). This computer will act as the liaison between all I/O devices on the robot. Although the robot is intended to understand high-level commands such as “read input message” and “output sound”, the SCOMP can only execute low-level commands such as “add” or “logical AND”. Thus, the robot must be programmed using Assembly code to break down the complex commands into simple instructions that read data from inputs, perform calculations on the data, and output desirable data to other peripherals.

**Additional Capabilities**

In order to improve the user interface while adding functionality to the robot, the group will add a PS/2 keyboard as an I/O peripheral. Adding a keyboard will vastly improve the efficiency and ease of message input over using the switches and pushbuttons of the DE2. The keyboard will allow for users to intuitively enter messages into the system, which will be translated into Morse code.

In addition, having a separate peripheral over a standardized port maintains the portability of the robot. Users can attach their own keyboards for message input, and then can detach the keyboard before sending the robot en route, eliminating the issues associated with attaching a physical peripheral to the robot. Figure 1 shows a block diagram of the proposed PS/2 keyboard input.

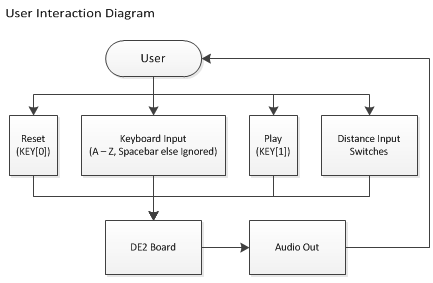


**Figure 1.** Proposed PS/2 Keyboard block diagram added as an I/O peripheral into the SCOMP.

## Project Details for the Morse Code Production

First, the user will type out a message using the PS/2 keyboard input, connected to the DE2. As each key is pressed, the SCOMP takes the corresponding letter and writes it to the LCD display for the user to see, as well as for storage purposes. During the input process, the SCOMP also checks the pushbuttons Key[0] and Key[1] for additional input. Key[0] will be used to reset the display, thereby clearing the current message. Key[1] will be used to signal the end of the message input from the PS/2 keyboard. If Key[1] is pressed, the SCOMP will stop reading input from the PS/2 keyboard and begin the output phase. Figure 2 depicts how the user can interact with the DE2 and provided I/O devices.

During the output phase, the SCOMP will read the message that is stored in the LCD display. As the SCOMP reads the message, it will convert the characters into Morse code, represented by a long binary string (zeros to represent silence, ones to represent a tone). A longer sequence of ones represents a longer duration of tone, while a longer sequence of zeros represents a longer duration of silence. The SCOMP then steps through this binary string and, at each high value, selects a tuning word based on the value of the switches and sends this tuning word to the DDS unit. At each low value in the message string, the SCOMP will select a tuning word of zero and sends it to the DDS. With these inputs, the DDS unit will then generate the tone (or silence), and pass this signal to the audio codec to be converted and played through the audio output port. The SCOMP will continue to loop through the stored message and repeat the audio signal. While the SCOMP loops, it will also monitor the DE2 switches, which are used to represent a distance from an ultrasonic sensor input. If the distance changes, then the SCOMP will select a different tuning word so that the DDS system will output a different tone. For a visual look at this process, a program flow is available in Appendix A.

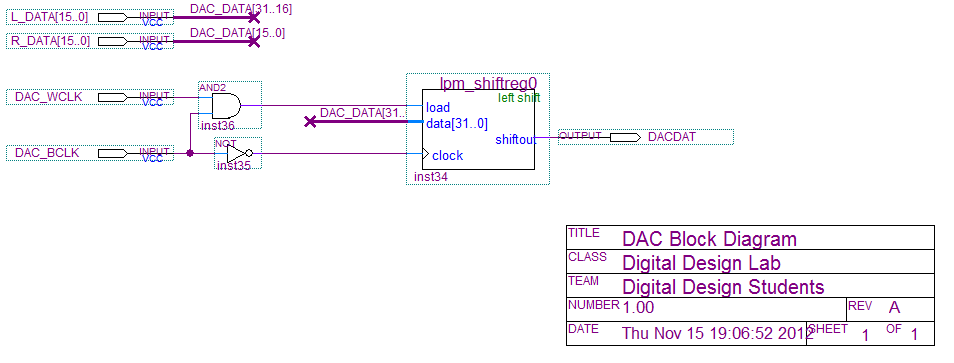


**Figure 2.** User interaction flow chart showing how user inputs and receives outputs to project.

## Using the DE2 Hardware

User input from the keyboard I/O will generate a message that is read by the SCOMP. This message will be read and translated one character a time into Morse code, which will help decide the tuning word to be sent to the DDS. The DDS system connects directly to the audio codec, which allows for simple creation of audio signals that will be outputted through the DE2 audio jack. The audio codec will be used as a digital-to-analog converter (DAC), so that the DDS frequency can be translated into analog sounds to be outputted through the DE2. The DDS will be connected to the DAC through a 15 bit wide data bus, which is sent to the left and right inputs of the DAC. The left and right inputs are concatenated, with the left input as bits 31 to 16 and the right input as bits 15 to 0. These bits are then put into a shifter, enabled by the clock signals. The DAC will take in two clock signals, one 48 kHz clock which is used to regulate the output of the DDS and ensure that the DAC is taking in the DDS data values at regular intervals. The other clock will be a 12 MHz clock that serves as the master clock for the audio codec chip. The audio codec outputs a signal called DACDAT, as depicted in figure 3, which is converted and outputted to the analog audio output.

The hardware on the DE2 will be controlled by a user as depicted in figure 2. A PS/2 keyboard will input a message to be translated to Morse code, and the switches on the DE2 will change the tuning word and thus the frequency of the output of the Morse code audio message. These switches simulate possible data from other potential robot inputs. The DE2 hardware will allow for debugging to take place using the logic analyzer. The DE2 will run HDL describing the SCOMP, which will be executing the instructions for the robot. The logic analyzer will be connected to the DE2 in order to verify that the SCOMP is correctly running the assembly code.

**Figure 3**. Block diagram of the DAC from the Audio Codec.

## Minimum Demonstrated

When the project is completed, the simulation will be done on the DE2 board. The demonstration will consist of entering the desired message on the PS/2 keyboard, and then displaying the message on the LCD. After displaying the text on the LCD, the SCOMP will generate the Morse code and give the tuning word to the DDS, which will then pass the audio samples to the DAC to create an audible tone of the Morse code message. The switches will be changed so that the Morse code can be played at different pitches. The pushbuttons will also demonstrate that the message can be reset, and that the message can be entered after being typed on the PS/2 keyboard.

# Management Plan

The Gantt chart in Appendix B shows the plan for designing and producing the project. Each part of the project will be split up amongst the members of the design team, with each member also editing and contributing to all other parts of the project. Since parts of the project are going to be designed by one member of a team at first, this member can thoroughly test and simulate their particular module to ensure that it will function individually before being integrated into the project as a whole. Should the design plan not go according to the plan laid out in the Gantt chart provided, the backup plan will be to remove parts of the project that are not completely necessary for the project requirements. In this situation, only the DDS audio output will be made sure to be complete, and the other goals including the LCD and LED outputs will be scrapped for the sake of time.

Appendix A: Program Flow Diagram

Appendix B: Gantt Chart