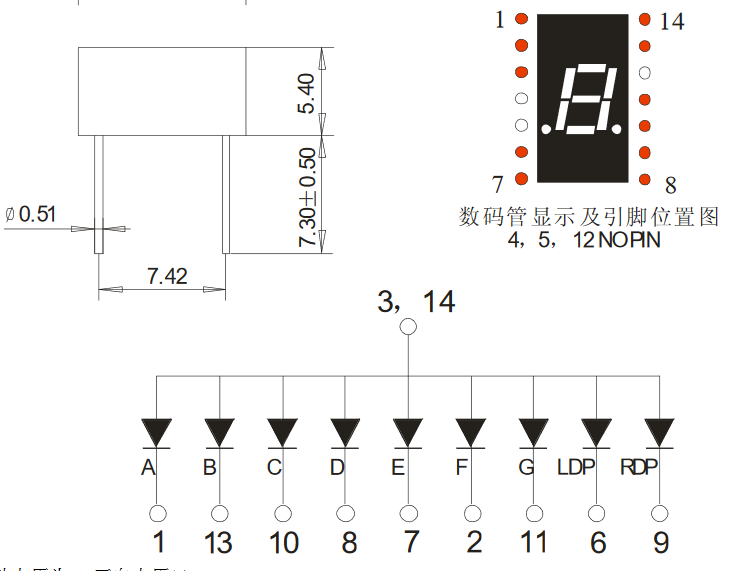
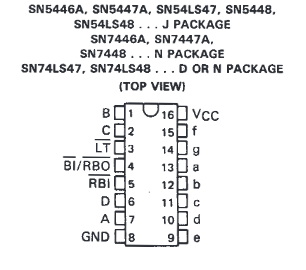
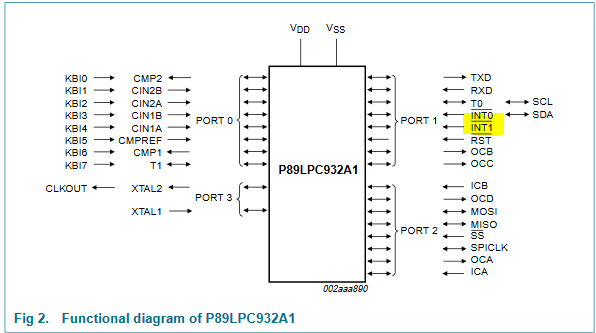
Project 3 Individual Report

Josh Cash

Individual Feature Design and Implementation:

For the Team portion of the project I worked as a head programmer. I had my hand in fixing and helping correct all the functions. I took more of an overseer role for the overall coding aspect. I personally wrote the following functions: keyboard2, displaySong, timerover, ExternInterrupt, clearScreen, playSong1, and song1Repeat. I collaborated on other functions as well. Again, I was the main consultant for our group when it came to the programming. I also spearheaded the circuit board. For the bread boarding we used a Common Anode 7-Segment Display(LSD3221-11) and a BCD – 7-Segment Display Decoder(7447).



I just used the calculations we did in the excel file (which is included with the project files and broken down in the team report) to assign what notes to play for song1. I used the pinout of the 932 to assign an onboard button to trigger the External Interrupt. I also went through and modularized the code to save on space. Such as adopting the holdNote function from Alan into the playNote function since playNote had something similar in it. The 7-Segment display uses the pins P3.0 and P3.1 to send a binary signal to the decoder to then forward the correct signals to the 7-Segment. The decoder uses 4 signals to create a 4 bit word but since I was only able to get those two pins to behave, we ground the two most significant bits. This allows us to display 0-3 on our 7-Segment Display. (code for the functions will be at the end of the document)

**Individual**

For my individual part I added another mode. This mode allowed the user to play a keyboard imitation with the computer’s keyboard using the asdf keys. It uses the calculations already made and functions already made to achieve the imitation. Each was assigned a note to play when the uart\_get receive that corresponding button input. This uses no timers, but only the serial communication via UART.

**CODE**

/\*

\* Description: Display Function That will send the Song name given via UART to Terminal.

\* Requires: Array of characters of song name and size of the array.

\* Returns: None.

\*/

void displaySong(unsigned char song[], unsigned char size) {

// Display Song

display(song, size);

// Move to the next line

uart\_transmit('\r');

uart\_transmit('\n');

}

/\*

\* Description: Expects Timer 0 to not be in use. For Timer Load Values greater than 65535.

\* Requires: Timer Load Value that the actual desired value is 225 \* value but value is less than 16 bit size.

\* Returns: None.

\*/

void timerover(unsigned int t) {

  // Looping over causes inexact timing due to loop nature.

  unsigned char loop = 225;

  // Make sure not to override the Timer1 settings.

  //(Hopefully this makes sure it doesn't mess up if it is running).

  unsigned int currentTMOD = TMOD >> 4 << 4;

  // Attach the timer 0 mode we want.

  currentTMOD += 0x01;

  // Set TMOD to that mode.

  TMOD = currentTMOD;

  // Run the pause.

  for(; loop > 0; loop--) {

    TH0 = -t >> 8;

    TL0 = -t;

    TR0 = 1;

    while(TF0 == 0);

TR0 = 0;

    TF0 = 0;

  }

}

/\*

\* Description: Function to play redunant tune for Song 1.

\* Requires: None.

\* Returns: None.

\*/

void song1Repeat() {

displayNote(4);

playNote(notes4[4], 2, 0);

displayNote(2);

playNote(notes4[2], 2, 0);

displayNote(0);

playNote(notes4[0], 3, 0);

}

/\*

\* Description: Song #1 - Hot Cross Buns.

\* Requires: None.

\* Returns: None.

\*/

void playSong1() {

// Measure 1.

song1Repeat();

// Measure 2.

song1Repeat();

// Measure 3.

displayNote(0);

playNote(notes4[0], 1, 0);

displayNote(0);

playNote(notes4[0], 1, 0);

displayNote(0);

playNote(notes4[0], 1, 0);

displayNote(0);

playNote(notes4[0], 1, 0);

displayNote(2);

playNote(notes4[2], 1, 0);

displayNote(2);

playNote(notes4[2], 1, 0);

displayNote(2);

playNote(notes4[2], 1, 0);

displayNote(2);

playNote(notes4[2], 1, 0);

// Measure 4.

song1Repeat();

// Only Delay the Next Play of the song if We are not switching modes.

if(!flag)

timerover(SEC);

}

/\*

\* Description: Function to handle when the mode switch button has been pressed.

\*\*\* Is wired up to recieve press signal from SW9 on the board.

\* Requires: EX1 to be enabled.

\* Returns: None.

\*/

void ExternInterrupt() interrupt 2 {

// Set our Global Flag to let everything else to gracefully switch modes.

flag = 1;

// Multi Button Protection.

timerover(BUTTONPAUSE);

}

/\*

\* Description: Clears the Terminal Via UART Excape characters.

\* Requires: None.

\* Returns: None.

\*/

void clearScreen() {

// Google is my Friend.

// Excape Character.

uart\_transmit(27);

// Clear Screen Code.

display("[1J", 3);

// Excape Character.

uart\_transmit(27);

// Curser to top Code.

display("[H", 2);

}

/\*

\* Description: Keyboard Function 2. Allows User to use 'ASDF' as buttons to play notes via Serial Communication Via UART.

\* Requires: None.

\* Returns: None.

\*/

void keyboard2() {

// Stays here till mode changes.

while(!flag) {

// Get note to play and play the assigned note.

switch(uart\_get()) {

case 'a':

turnOnLED(11);

holdNote(notes4[11]);

break;

case 's':

turnOnLED(0);

holdNote(notes5[0]);

break;

case 'd':

turnOnLED(2);

holdNote(notes5[2]);

break;

case 'f':

turnOnLED(4);

holdNote(notes5[4]);

default:

break;

}

shutOffLED();

}

}