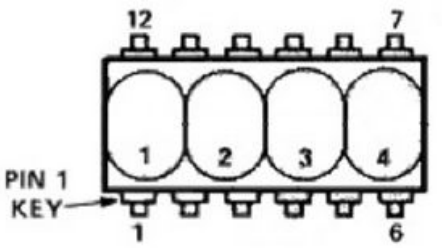
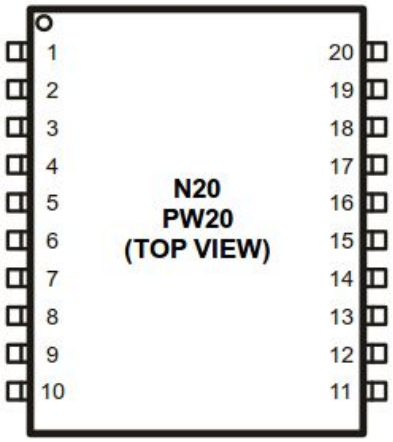
Jonah Groendal
CS2230

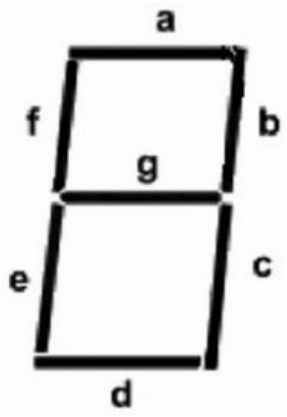
A6 Documentation

PINOUT





2 : bubble display pin no. 8 (ANODE d)
3 : serial communications
4 : serial communications
3 : tied to button (P1.3) input
6 : bubble display pin no. 11 (ANODE b)
7 : bubble display pin no. 3 (ANODE c)
8 : bubble display pin no. 9 (ANODE f)
9 : bubble display pin no. 7 (ANODE g)
10 : bubble display pin no. 12 (ANODE a)
11 : bubble display pin no. 10 (CATHODE 2)
12 : bubble display pin no. 4 (CATHODE 3)
13 : bubble display pin no. 6 (CATHODE 4)
14 : PWM output for speaker
15 : bubble display pin no. 2 (ANODE e)
18 : RGB LED red output (high)
19 : RGB LED greed output (high)
20 : RGB LED ground



GAME OVERVIEW

Using the bubble display, speaker, RGB LED and the onboard button, a game is created. The bubble display will count up from 000 to 999, during which the player must attempt to stop the counting at certain numbers. These are even numbers, then powers

of two, then prime numbers. Once the player successfully presses the button on one of the first set of numbers, the counter resets. The player then attempts to do the same on the second set, then the third. After the player successfully stops the timer on all three, he/she will progress to the next level, which increments the timer at shorter intervals (it gets faster). If, at any time, the player presses the button when an incorrect number is on the screen, a life (or attempt) is taken away. By default Each reset allows the player three lives. The goal is to reach the highest level possible.

USER INTERFACE

The user will be able to see the current number on the bubble display. He or she will know when there is a correct or incorrect button push from audio feedback using the speaker. The RGB LED will display the current number of lives left (red is one, orange is two, and green is three or more). Serial output and input is also used to communicate with the user. A command line interface is used to display the game rules as well as the current player statistics such as latest number press, current number of lives, and the current goal (or current set of numbers). The shell can also input commands from the user.

SOFTWARE DESIGN

Timers and interrupts:
TIMER0 is used for the PWM output for the speaker. The frequency can be adjusted by adjusting the global variable “frequency.”

TIMER1 is used to call the interrupt that controls the bubble displays and also the RGB LED. Within the interrupt method, the digits to display are calculated from the global counter variable “display_number.” Then, one digit is displayed each time the interrupt is triggered, changing the display each time. The “display_number” counter is also incremented here.

BUTTON P1.3 triggers the port 1 interrupt. Here, the user input (current number) is calculated and audio feedback is played. This also triggers serial I/O feedback.

Serial I/O:
A menu is included, which accepts commands to customize the game. Feedback, such as current goal, number of lives, and completed levels is also displayed.

There are three code files used in this program:

a6.c - contains the main method, global variables, and interrupts. Also contains code for the shell and all serial I/O
numberDisplay.c - contains code for displaying a digit on a given display
PWMControllerA0.c - controls PWM output on P1.6 (for the speaker)

DEVELOPMENT SCHEDULE

1. Plan pinout organization and configure wiring.
2. Plan and code timers, interrupts, and PWM for the speaker.
3. Configure shell and add commands.
4. Finish header text, serial output messages, and game mechanics.
5. Complete final testing and debugging.