Microcontroller Project Autograder Specifications

Tarang Janawalkar

July 17, 2024

Contents

1 Terminology

The following terms will be used throughout this document to refer to various aspects of the autograding process. Their definitions are provided below:

- A **test** refers to one execution of the emulator. This is different from the tests displayed on Gradescope as some Gradescope tests may contain multiple **tests**.
- **Test output** refers to the text displayed under each **test** on Gradescope.
- An **input event** refers to the inputs provided to the emulator. These inputs can affect the position of the potentiometer, the state of a pushbutton, or the state of the UART RX net, resulting in a character being transmitted to the ATtiny1626 via UART.
- A **gameplay input** refers to an input to the Simon game. Such inputs can either be correct (match Simon), or incorrect (fail to match Simon).
- An **output event** refers to an output detected by the emulator. Namely, this includes outputs from the buzzer and 7-segment display, and characters transmitted from the ATtiny1626 via UART.
- An **event** refers to a group of buzzer and display **output events** that were produced by Simon or the user.

2 Program Emulation

As usual, the autograder uses an emulator to simulate user input via various **input events**. The autograder then reads and parses all **output events** generated by the emulator to evaluate a program.

2.1 The Tests

A total of 16 tests will be executed, with some tests grouped under the same Gradescope test. These tests may execute the emulator with a different set of parameters. That is,

- Each test may execute the emulator with a different timeout.
- Each test may execute the emulator with a different set of inputs.

A list of all tests and their timeouts is provided below, under their corresponding Gradescope test names[1].

1. **Gameplay Test**: 5505 ms
2. **Pushbutton Hold Test**: 7405 ms

[1] Actual times may vary by ±200 ns when using the default clock prescalar.

3. **Pushbutton Debouncing Test**: 620 ms
4. **Sequencing Test**: 31 300 ms
5. **Playback Delay Test**:
   A. 4737.5 ms
   B. 8237.5 ms
   C. 11 737.5 ms
   D. 15 237.5 ms
6. **Playback Frequency Test**: 9550 ms
7. **UART Gameplay Test**: 5630 ms
8. **UART Octave Test**:
   A. 2500 ms
   B. 2500 ms
9. **High Score Test**: 30 700 ms
10. **Reset Test**: 2555 ms
11. **Seed Test**: 5500 ms
12. **Mixed Input Test**: 12 710 ms

An efficient program should produce the necessary output in response to all inputs, within the timeouts prescribed above.

2.2 The Events

Each test will simulate some pre-determined inputs that may expect an output. The start time and duration of these outputs are compared against an ideal program which receives the same inputs. Each test will output a series of events to provide feedback on each output event. This is shown below:

Example of events

```
[EVENT 1 - Simon] [PLAYBACK DELAY: 250 ms]
PASS. Buzzer produced a 358.10 Hz tone at 0 ms for a duration of 125 ms.
PASS. Display produced "|   " at 0 ms for a duration of 125 ms.

[EVENT 2 - User Input: (S1 (15 ms))]
PASS. Buzzer produced a 358.10 Hz tone at 350 ms for a duration of 125 ms.
PASS. Display produced "|   " at 350 ms for a duration of 125 ms.
      [INPUT SEQUENCE CORRECT]
PASS. Display produced "88" at 500 ms for a duration of 250 ms.
```

Note that some tests may log less or more information depending on what is being evaluated.

2.3 Dealing with Timing Errors

To account for variations in programs, the autograder allows small tolerances between output events, and adds delays before all gameplay inputs (i.e., pushbutton inputs or serial functions S1-S4). The following outputs will not be accepted:

- A buzzer or display output event that is active for a duration with relative error greater than 15 %.
- A buzzer or display output event that occurs before a gameplay input is simulated.

In some cases, future outputs may be required to calculate the duration of current outputs. For example, if a program has not yet implemented displaying the user score on the 7-segment display, and an incorrect gameplay input is simulated, then, after displaying the **FAIL** pattern, the autograder may fail to determine the duration of this pattern.

3 Emulator Inputs

All valid inputs are logged in the test output, next to the relevant event.

3.1 Potentiometer Updates

Changes to the position of the potentiometer result in a change to the *playback delay*. This change is emphasised by the following text:

Example test output from potentiometer position update

```
[EVENT 1 - Simon] [PLAYBACK DELAY: 250 ms]
```

All subsequent events will use this *playback delay*, until it is changed again.

3.2 Pushbutton Inputs

Pushbutton presses result in an output from the buzzer and display, when pressed for a sufficiently long duration, accounting for mechanical switch bounce. These gameplay inputs are provided in a list as follows:

Example test output for pushbutton inputs

```
// S1 pressed for 15 ms
[EVENT 2 - User Input: (S1 (15 ms))]

// S1 pressed for 15 ms and S3 pressed for 15 ms
[EVENT 4 - User Input: (S1 (15 ms), S3 (15 ms))]
```

In this test output, pushbutton presses are listed with their corresponding press durations.

For events with multiple gameplay inputs, inputs after the first are simulated after the *playback delay* and a short delay have elapsed, simulating the time a person takes to transition between the pushbuttons.

3.3 UART Inputs

Characters transmitted over UART perform various actions. These characters will be shown for UART related tests and their corresponding actions will be provided next to the relevant event, when appropriate. An example of UART input is shown below:

Example test output for UART inputs

```
UART input:
  250 ms: 'k'
  350 ms: 'q'
 1350 ms: ','
 1450 ms: '1'
 1700 ms: '3'
```

An example of the UART functions is shown below:

Example test output for UART functions

```
[EVENT 2 - User Input: ('q')]              // S1
[EVENT 4 - User Input: ('1', 'e')]         // S1 (alt) and S3
[EVENT 2 - User Input: ('q')] [OCTAVE: -1] // DEC FREQ
[EVENT 2 - User Input: ('q')] [OCTAVE: 1]  // INC FREQ
[EVENT 4 - Simon] [RESET]                  // RESET
[EVENT 3 - Simon] [SEED: 0x2DF599AC]       // SEED
```

Note that non-gameplay related inputs will be passed while the program is waiting for user input, before gameplay inputs are simulated.

To account for alternative keys, some UART inputs will be randomised.

4 Emulator Outputs

For this assignment, the autograder will focus on **buzzer** and **display** output events. When enabled, these will be logged at the start of a test's output, as follows:

Example buzzer/display outputs from Gameplay Test

```
Detected buzzer events:
    0 ms:  358.10 Hz, 50.00 % duty cycle
  125 ms:    0.00 Hz,  0.00 % duty cycle
  300 ms:  358.10 Hz, 50.00 % duty cycle
  425 ms:    0.00 Hz,  0.00 % duty cycle
...

Detected display events:
    0 ms: "|   "
  125 ms: "  "
  300 ms: "|   "
  425 ms: "  "
...
```

The autograder may also provide UART output at the end of a test's output, logging all characters transmitted before emulation timeout. An example is provided below:

Example UART output from UART Gameplay Test

```
UART output:
SUCCESS
1
GAME OVER
2
Enter name:
Patrick 2
SUCCESS
1
GAME OVER
2
Enter name:
Patrick 2
Jackson 2
```

4.1 Buzzer Output

When evaluating buzzer output events, the start time, frequency, duty cycle, and duration of a tone must be within tolerance. The duration of a tone is calculated as:

$$\delta t = t_{\text{off}} - t_{\text{on}}$$

where t_{on} is the first occurrence of the expected tone, and t_{off} is the first occurrence of a 0 Hz, 0 % duty cycle tone, found after t_{on}.

If a tone is produced with an incorrect frequency, the autograder will skip over that tone until it finds a tone with the correct frequency. This may lead to inconsistencies in timings between the buzzer and display, as shown below:

Example of incorrect buzzer tone

```
Detected buzzer events:
    0 ms:  358.10 Hz, 50.00 % duty cycle
  125 ms:    0.00 Hz,  0.00 % duty cycle
  350 ms:  478.00 Hz, 50.00 % duty cycle // expected 358.10 Hz tone at 350 ms
  475 ms:    0.00 Hz,  0.00 % duty cycle
  725 ms:  358.10 Hz, 50.00 % duty cycle // found tone at the wrong time
  850 ms:    0.00 Hz,  0.00 % duty cycle
...

Detected display events:
    0 ms: "|   "
  125 ms: "  "
  350 ms: "|   "
  475 ms: "  "
  475 ms: "88"
  725 ms: "  "
  725 ms: "|   "
  850 ms: "  "
...

[EVENT 1 - Simon] [PLAYBACK DELAY: 250 ms]
PASS. Buzzer produced a 358.10 Hz tone at 0 ms for a duration of 125 ms.
PASS. Display produced "|   " at 0 ms for a duration of 125 ms.

[EVENT 2 - User Input: ('1')]
FAIL. Buzzer produced a 358.10 Hz tone at 725 ms for a duration of 125 ms.
PASS. Display produced "|   " at 350 ms for a duration of 125 ms.
      [INPUT SEQUENCE CORRECT]
PASS. Display produced "88" at 475 ms for a duration of 250 ms.
```

In this example, the first buzzer event under event 2 fails because it was not detected at the expected time of 350 ms.

Likewise, if no buzzer output is detected, or if a tone does not cease after 50% of the playback delay, testing will terminate after that event.

4.2 Display Output

When evaluating display output events, the start time and duration of a pattern must be within tolerance. The start time and duration of a pattern are calculated similarly to buzzer output events.

As the display is multiplexed between two digits, patterns that require both digits to be illuminated will produce transient output before the emulator detects a steady state from the 7-segment display. This behaviour is shown in the output below:

Example of transcience in the output from the 7-segment display

```
Detected display events:
    0 ms: "|   "
  125 ms: "  "
  350 ms: "|   "
  475 ms: "  "
  475 ms: "8 " // transience (LHS)
  480 ms: " 8" // transience (RHS)
  485 ms: "88" // steady state
  725 ms: "  "
  725 ms: "|   "
...
```

In this example, the 7-segment display refreshes every 5 m s, switching between both digits, for an equal duration, allowing the emulator to detect a steady state after 10 m s.

If the 7-segment display is not properly multiplexed, the emulator may fail to detect a steady state. The output of such a program is shown below:

Example of improperly configured display multiplexing

```
Detected display events:
    0 ms: "|   "
  125 ms: "  "
  350 ms: "|   "
  475 ms: "  "
  475 ms: "8 "
  475 ms: " 8"
  480 ms: "8 "
  480 ms: " 8"
  485 ms: "8 "
  585 ms: " 8"
  590 ms: "8 "
  590 ms: " 8"
...
```

In this example, the 7-segment display refreshes twice successively every 5 m s, rather than once every 5 m s, where the left digit is illuminated for less than 1 m s, and the right digit is illuminated for 5 m s. As these intervals are not equal, the emulator will not log any steady state output.

4.3 UART Output

UART output is captured when the emulator terminates, and as such, the autograder only tests whether the correct output is produced. A program should follow all formatting specifications exactly to ensure that UART output tests succeed. Caution should be taken when submitting a program with debugging `printf` statements, as they can significantly impact the efficiency of a program. Such statements should be removed upon submission.