Brayden Daly, Zachary Lee

Lab A7

EE 329-05

2025 May 7

A7

Introduction:

The code works as intended, with the initial screen showing the correct message with part of the message blinking and being able to type in different colors. Additionally, the screen successfully transitioned to the game while being forced to stay within the boundary and jump to the opposite side with no known bugs. One lesson learned during development was implementing the LPUART1 to not only send a message to the terminal and utilize the echo with the keyboard, but also use it to create a basic game within the terminal.

YouTube demo: https://youtu.be/XUNZTRecJj8

Baud Rate Calculation:

$$\frac{256*4*10^6}{115.2x10^3} = 8888.89$$

Round up to integer value: 8889 (0x22B9)

Appendix

---

main.h

```
/* USER CODE BEGIN Header */
/**
 ******************************************************************************
 * @file           : main.h
 * @brief          : Header for main.c file.
 *                   This file contains the common defines of the application.
 ******************************************************************************
 * @attention
 *
 * Copyright (c) 2025 STMicroelectronics.
 * All rights reserved.
 *
 * This software is licensed under terms that can be found in the LICENSE file
 * in the root directory of this software component.
 * If no LICENSE file comes with this software, it is provided AS-IS.
 *
 ******************************************************************************
 */
/* USER CODE END Header */
#ifndef __MAIN_H
#define __MAIN_H
#ifdef __cplusplus
extern "C" {
#endif
#include "stm32l4xx_hal.h"
void Error_Handler(void);
#ifdef __cplusplus
}
#endif
#endif /* __MAIN_H */
```

---

main.c

```
/* USER CODE BEGIN Header */
/**
 ******************************************************************************
 * @file           : main.c
 * @brief          : Main program body
 ******************************************************************************
```

```
 * @attention
 *
 * Copyright (c) 2025 STMicroelectronics.
 * All rights reserved.
 *
 * This software is licensed under terms that can be found in the LICENSE file
 * in the root directory of this software component.
 * If no LICENSE file comes with this software, it is provided AS-IS.
 *
 ******************************************************************************
 */
/* USER CODE END Header */
/* Includes ------------------------------------------------------------------*/
#include "main.h"
#include <UART.h>
#include <GV.h>
//variable to check if game is started
int start_game = 0;
void SystemClock_Config(void);
/* ----------------------------------------------------------------------------
* function : main
* INs      : none
* OUTs     : int
* action   : Run Part one, switch to game title screen, start game and create
*            borders
* authors  : Brayden Daly, Zachary Lee
* version  : 0.3
* date     : 250507
* ---------------------------------------------------------------------------- */
int main( void ) {
   HAL_Init();
   LPUART_init(); //Initialize UART
   //Print Title Screen for Part 1
   LPUART_Print( "\x1B[3B\x1B[5C" );
   LPUART_Print("All good students read the" );
   LPUART_Print( "\x1B[1B\x1B[21D" );
   LPUART_Print( "\x1B[5m" );
   LPUART_Print( "Reference Manual" );
   LPUART_Print( "\x1B[H" );
   LPUART_Print( "\x1B[0m" );
   LPUART_Print( "Input: " );
   //stall until the game is started ('P' Pressed)
   while ( !start_game )
```

```
   {
      ;
   }
   //start title screen of game
   Splash_Screen();
   //Create Border for game
   Create_Border();
}
```

---

uart.h

```
/*
* UART.h
*
*  Created on: May 7, 2025
*      Author: bdaly
*/
#ifndef UART_H_
#define UART_H_
#include "main.h"
#include "stm32l4xx.h"
#define UART_PORT GPIOG    // Define the UART port as GPIOG
#define LEFT_BORDER 20     //Define Left border position
#define RIGHT_BORDER 140   //Define Right border position
#define UPPER_BORDER 35    //define upper border position
#define LOWER_BORDER 5     //define lower border position
// Function Prototypes
void LPUART_init(void);
void LPUART_Print(const char* message);
void LPUART1_IRQHandler(void);
void LPUART_ESC_Print(const char* esc_code, const char* message);
void Splash_Screen(void);
void Create_Border(void);
#endif /* UART_H_ */
```

---

uart.c

```
/*
* UART.c
*
*  Created on: May 7, 2025
*      Author: bdaly, Zachary Lee
*/
```

```c
#include <main.h>
#include <UART.h>
#include <GV.h>
//global variables to hold the position and start game interrupt
uint32_t x = 0;
uint32_t y = 0;
extern int start_game;
/* -----------------------------------------------------------------------------
* function : LPUART_init
* INs      : none
* OUTs     : none
* action   : initializes UART GPIO (PORT G 7,8), sets baud rate
*
* authors  : Brayden Daly, Zachary Lee
* version  : 0.3
* date     : 250507
* ---------------------------------------------------------------------------- */
void LPUART_init( void ) {
   PWR->CR2 |= (PWR_CR2_IOSV);  // power avail on PG[15:2] (LPUART1)
   RCC->AHB2ENR |= (RCC_AHB2ENR_GPIOGEN);  // enable UART_PORT clock
   RCC->APB1ENR2 |= RCC_APB1ENR2_LPUART1EN; // enable LPUART clock bridge
   /* USER: configure UART_PORT registers MODER/PUPDR/OTYPER/OSPEEDR then
    select AF mode and specify which function with AFR[0] and AFR[1] */
   //RECEIVER (RX) PG8
   UART_PORT->MODER &= ~(GPIO_MODER_MODE7 | GPIO_MODER_MODE8);
   //TRANSMITTER (TX) PG7
   UART_PORT->MODER |= (GPIO_MODER_MODE7_1 | GPIO_MODER_MODE8_1); // Set AF mode
   UART_PORT->OTYPER &= ~(GPIO_OTYPER_OT7);
   //pullups
   UART_PORT->PUPDR &= 0; // Enable pull-up for PG7 and PG8
   // Set very high output speed for PC7, 8
   UART_PORT->OSPEEDR |=  ((3 << GPIO_OSPEEDR_OSPEED7_Pos) |
                           (3 << GPIO_OSPEEDR_OSPEED8_Pos));
   // preset PC0, PC1, PC2, PC3 to 0
   UART_PORT->BRR = (GPIO_PIN_0 | GPIO_PIN_1 | GPIO_PIN_2 | GPIO_PIN_3);
   UART_PORT->AFR[0] &= ~(0x000F << GPIO_AFRL_AFSEL7_Pos); // clear PG7 nibble AF
   UART_PORT->AFR[0] |= (0x0008 << GPIO_AFRL_AFSEL7_Pos); //set PG7 AF = LPUART1_TX
   UART_PORT->AFR[1] &= ~(0x000F << GPIO_AFRH_AFSEL8_Pos); // clear PG8 nibble AF
   UART_PORT->AFR[1] |= (0x0008 << GPIO_AFRH_AFSEL8_Pos); //set PG8 AF = LPUART1_RX
   LPUART1->CR1 &= ~(USART_CR1_M1 | USART_CR1_M0); // 8-bit data
   LPUART1->CR1 |= USART_CR1_UE;                   // enable LPUART1
   LPUART1->CR1 |= (USART_CR1_TE | USART_CR1_RE);  // enable xmit & recv
   LPUART1->CR1 |= USART_CR1_RXNEIE;        // enable LPUART1 recv interrupt
```

```
   LPUART1->ISR &= ~(USART_ISR_RXNE);       // clear Recv-Not-Empty flag
   /* USER: set baud rate register (LPUART1->BRR) */
   LPUART1->BRR = 0x22B9; //8889
   NVIC->ISER[2] = (1 << (LPUART1_IRQn & 0x1F));   // enable LPUART1 ISR
   __enable_irq();
   //clear screen and move cursor to top left
   LPUART_Print( "\x1b[2J" );
   LPUART_Print( "\x1b[H" );
}
/* -----------------------------------------------------------------------
* function : Move_Left_Clear
* INs      : none
* OUTs     : none
* action   : move the cursor to the left and clear character in previous place
*
* authors  : Brayden Daly, Zachary Lee
* version  : 0.3
* date     : 250507
* ----------------------------------------------------------------------- */
void Move_Left_Clear( void ) {
   LPUART_Print( "\x1b[1D" );
   LPUART_Print( " " );
   LPUART_Print( "\x1b[1D" );
   LPUART_Print( "\x1b[1D" );
}
/* -----------------------------------------------------------------------
* function : LPUART_Print
* INs      : string
* OUTs     : none
* action   : print a string to the screen
*
* authors  : Brayden Daly, Zachary Lee
* version  : 0.3
* date     : 250507
* ----------------------------------------------------------------------- */
void LPUART_Print( const char* message ) {
   uint16_t iStrIdx = 0;
   while ( message[iStrIdx] != 0 ) {
      while(!(LPUART1->ISR & USART_ISR_TXE)) // wait for empty xmit buffer
         ;
      LPUART1->TDR = message[iStrIdx];       // send this character
   iStrIdx++;                               // advance index to next char
   }
```

```
}
/* ------------------------------------------------------------------------
* function : LPUART_IRQHandler
* INs      : none
* OUTs     : none
* action   : Interrupt Handler to check for ESC key presses
*
* authors  : Brayden Daly, Zachary Lee
* version  : 0.3
* date     : 250507
* ------------------------------------------------------------------------ */
void LPUART1_IRQHandler( void ) {
   //set variable to hold which character was received
   uint8_t charRecv;
   //check if there was an interrupt
   if (LPUART1->ISR & USART_ISR_RXNE) {
      //set charRecv as the input from the UART interrupt
      charRecv = LPUART1->RDR;
      //switch to character inputted
      switch ( charRecv ) {
      //If R pressed, make font color red
      case 'R':
         LPUART_Print( "\x1b[31m" );
         break;
      //If P pressed, set start_game to 1
      case 'P':
         start_game = 1;
         break;
      //If B pressed, make font color blue
      case 'B':
         LPUART_Print( "\x1b[34m" );
         break;
      //If G pressed, make font color Green
      case 'G':
         LPUART_Print( "\x1b[32m" );
         break;
      //If W pressed, make font color white
      case 'W':
         LPUART_Print( "\x1b[37m" );
         break;
      //If C pressed, Clear the Screen and move cursor top left
      case 'C':
         LPUART_Print( "\x1b[2J" );
```

```
   LPUART_Print( "\x1b[H" );
//If i pressed, move character up
case 'i':
   //if reach top border
   if ( y == UPPER_BORDER - 1 ) {
      //erase character
      Move_Left_Clear();
      //move down to bottom border
      LPUART_Print( "\x1b[27B" );
      //print character
      LPUART_Print( "💩" );
      //update y coordinate
      y = LOWER_BORDER + 2;
   }
      //otherwise, move character up and increment y
   else {
      Move_Left_Clear();
      LPUART_Print( "\x1b[1A" );
      LPUART_Print( "💩" );
      y++;
   }
   break;
//if j pressed move left
case 'j':
   //If character hits the left border
   if ( x == LEFT_BORDER + 1 ) {
      //clear character
      Move_Left_Clear();
      //move down to right border - 2
      LPUART_Print( "\x1b[117C" );
      //print character
      LPUART_Print( "💩" );
      //update x coordinate
      x = RIGHT_BORDER - 2;
   }
   //otherwise, move character left and decrement x
   else {
      Move_Left_Clear();
      LPUART_Print( "\x1b[1D" );
      LPUART_Print( "💩" );
      x--;
   }
   break;
```

```
//If k pressed, move character down
case 'k':
   //if character hits the lower border
   if ( y == LOWER_BORDER + 2 ) {
      Move_Left_Clear();
      //move down to top border - 2
      LPUART_Print( "\x1b[27A" );
      //print character
      LPUART_Print( "💩" );
      //update y coordinate
      y = UPPER_BORDER - 1;
   }
   //otherwise, move character down and decrement y
   else {
      //clear character
      Move_Left_Clear();
      //move character down one
      LPUART_Print( "\x1b[1B" );
      //print character
      LPUART_Print( "💩" );
      //decrement y
      y--;
    }
   break;
//If l pressed move right and increment x
case 'l':
   //If character hits right border
   if ( x == RIGHT_BORDER - 2 ) {
      //clear character
      Move_Left_Clear();
      //move to left boarder - 2
      LPUART_Print( "\x1b[117D" );
      //print character
      LPUART_Print( "💩" );
      //update x coordinate
      x = LEFT_BORDER + 1;
   }
   //Otherwise, move character right and increment x
   else {
      //clear character
      Move_Left_Clear();
      //move character right one
      LPUART_Print( "\x1b[1C" );
```

```
            //print character
            LPUART_Print( "💩" );
            //increment x
            x++;
         }
         break;
      //for default, wait for TX buffer
      default:
         while ( !(LPUART1->ISR & USART_ISR_TXE) )
         {
              ;    // wait for empty TX buffer
         }
         //reset start game
         start_game = 0;
         LPUART1->TDR = charRecv;  // echo char to terminal
      }  // end switch
   }
}
/* ---------------------------------------------------------------------------
* function : Splash_Screen
* INs      : none
* OUTs     : none
* action   : Prints Title Screen for Game
*
* authors  : Brayden Daly, Zachary Lee
* version  : 0.3
* date     : 250507
* ------------------------------------------------------------------------- */
void Splash_Screen( void ) {
   //set temporary variable to toggle for moving stars
   uint8_t tempvar = 0;
   //clear screen
   LPUART_Print( "\x1b[2J" );
   //move cursor to top left
   LPUART_Print( "\x1b[H" );
   //move down 15
   LPUART_Print( "\x1b[15B" );
   //move 60 to the right
   LPUART_Print( "\x1b[60C" );
   //print name of game
   LPUART_Print( "WELCOME TO 329 PRISON 😨" );
   //iterate to toggle stars
   for ( uint8_t i = 0; i < 10; i++ ) {
```

```
//switch statement for toggling stars
switch(tempvar) {
   //if 0, print stars in certain position
   case (0):
      //move cursor to top left
      LPUART_Print( "\x1b[H" );
      //move cursor down 10
      LPUART_Print( "\x1b[10B" );
      //move cursor 50 to the right
      LPUART_Print( "\x1b[50C" );
      //print the stars
      LPUART_Print( "🌟 🌟 🌟 🌟 🌟 🌟 🌟 🌟 🌟 🌟 🌟 🌟 🌟 🌟 🌟 " );
      //move cursor upper left
      LPUART_Print( "\x1b[H" );
      //move down 20
      LPUART_Print( "\x1b[20B" );
      //move 50 to the right
      LPUART_Print( "\x1b[50C" );
      //print stars
      LPUART_Print( "🌟 🌟 🌟 🌟 🌟 🌟 🌟 🌟 🌟 🌟 🌟 🌟 🌟 🌟 🌟 " );
      break;
   //otherwise, print stars in another position to make them twinkle
   case(1):
      //move cursor to top left
      LPUART_Print( "\x1b[H" );
      //move cursor down 10
      LPUART_Print( "\x1b[10B" );
      //move cursor 51 to the right
      LPUART_Print( "\x1b[51C" );
      //print stars
      LPUART_Print( "🌟 🌟 🌟 🌟 🌟 🌟 🌟 🌟 🌟 🌟 🌟 🌟 🌟 🌟 🌟 " );
      //move cursor upper left
      LPUART_Print( "\x1b[H" );
      //move 20 down
      LPUART_Print( "\x1b[20B" );
      //move 51 right
      LPUART_Print( "\x1b[51C" );
      //print stars again
      LPUART_Print( "🌟 🌟 🌟 🌟 🌟 🌟 🌟 🌟 🌟 🌟 🌟 🌟 🌟 🌟 🌟 " );
   break;
}
//delay small amount for animation
delay_us(200000);
```

```
      //toggle to next case statement
      tempvar ^= 1;
   }
   //go to top left
   LPUART_Print( "\x1b[H" );
   //move to upper left border
   LPUART_Print( "\x1b[21C" );
   LPUART_Print( "\x1b[34B" );
   //delay small amount of time
   delay_us(500000);
   //animation to have character run across the screen
   //iterate until it is done running
   for ( uint32_t i = 0; i < 120; i++ ) {
      //clear character
      Move_Left_Clear();
      //print character
      LPUART_Print( "💩" );
      //move character right one
      LPUART_Print( "\x1b[1C" );
      //small delay
      delay_us(50000);
   }
   //another delay
   delay_us(500000);
   //clear
   LPUART_Print( "\x1b[2J" );
   //go to top left
   LPUART_Print( "\x1b[H" );
   //set start coordinates to middle of screen
   x = 80;
   y = 20;
}
/* ------------------------------------------------------------------------
* function : Create_Border
* INs      : none
* OUTs     : none
* action   : Creates the border for the game
*
* authors  : Brayden Daly, Zachary Lee
* version  : 0.3
* date     : 250507
* ------------------------------------------------------------------------ */
void Create_Border( void ) {
```

```
//clear
LPUART_Print( "\x1b[2J" );
//go to top left
LPUART_Print( "\x1b[H" );
//set cursor to LEFT_BORDER and upper border
//move to upper left border
LPUART_Print( "\x1b[20C" );
LPUART_Print( "\x1b[5B" );
//print the ceiling of the border
for ( uint32_t i = 0; i < (RIGHT_BORDER - LEFT_BORDER); i++ ) {
   LPUART_Print("-");
}
//move to upper left border
LPUART_Print( "\x1b[H" );
//move 20 to the right
LPUART_Print( "\x1b[20C" );
//move 5 down
LPUART_Print( "\x1b[5B" );
// Draw the side borders
for (uint32_t i = 0; i < (UPPER_BORDER - LOWER_BORDER); i++) {
   LPUART_Print( "|" );          //Print Left border
   LPUART_Print( "\x1b[1D" );       // Move back to the left 1 column
   LPUART_Print( "\x1b[120C" );        // Move back to the right 30 columns
   LPUART_Print( "|" );          //print right border
   LPUART_Print( "\x1b[1B" );       // Move down 1 row
   LPUART_Print( "\x1b[121D" );        // Move back to the left 1 columns
}
//go to top left
LPUART_Print( "\x1b[H" );
//move to upper left border
LPUART_Print( "\x1b[21C" );
LPUART_Print( "\x1b[34B" );
//print the floor of the border
for ( uint32_t i = 0; i < (RIGHT_BORDER - LEFT_BORDER - 1); i++ ) {
   LPUART_Print( "-" );
}
//go to top left
LPUART_Print( "\x1b[H" );
//move the cursor to center of screen
LPUART_Print( "\x1b[20B" );
LPUART_Print( "\x1b[80C" );
//print character
LPUART_Print( "💩" );
```

```
}
```

---

delay.h

```
/*
* delay.h
*
*  Created on: Apr 21, 2025
*      Author: bdaly
*/
#ifndef INC_DELAY_H_
#define INC_DELAY_H_
#include "stm32l4xx_hal.h"
void SysTick_Init( void );
void Delay_Us( const uint32_t time_us );
#endif /* INC_DELAY_H_ */
```

---

delay.c

```
#include "main.h"
#include "Delay.h"
#include "stm32l4xx.h"
#include <stdint.h>
/* ---------------------------------------------------------------------------
* function : SysTick_Init(void);
* INs      : none
* OUTs     : none
* action   : Configures the ARM Cortex-M SysTick timer for microsecond delays.
*            Disables interrupts and sets it to use the processor clock.
* authors  : Brayden Daly, Tyler Wong
* version  : 0.3
* date     : 253004
* ------------------------------------------------------------------------- */
void SysTick_Init( void ) {
   SysTick->CTRL |=  (SysTick_CTRL_ENABLE_Msk |       // Enable SysTick
                      SysTick_CTRL_CLKSOURCE_Msk);    // Use processor clock
   SysTick->CTRL &= ~(SysTick_CTRL_TICKINT_Msk);     // Disable SysTick interrupt
}
/* ---------------------------------------------------------------------------
* function : delay_us(uint32_t time_us);
* INs      : time_us - number of microseconds to delay
* OUTs     : none (blocking delay)
* action   : Uses SysTick countdown to delay for specified number of microseconds.
```

```
*            Note: small values may result in longer-than-expected delay.
* authors  : Brayden Daly
* version  : 0.3
* date     : 253004
* ------------------------------------------------------------------------- */
void delay_us( const uint32_t time_us ) {
   // Calculate number of clock cycles for the desired delay
   SysTick->LOAD = (uint32_t)((time_us * (SystemCoreClock / 1000000)) - 1);
   SysTick->VAL = 0;                                          // Reset SysTick counter
   SysTick->CTRL &= ~(SysTick_CTRL_COUNTFLAG_Msk);            // Clear count flag
   while (!(SysTick->CTRL & SysTick_CTRL_COUNTFLAG_Msk)); // Wait for countdown
}
```

GV.h

```
/*
* GV.h
*
*  Created on: May 7, 2025
*      Author: bdaly
*/
#ifndef GV_H_
#define GV_H_
extern int start_game;
#endif /* INC_GV_H_ */
```