   Wyatt Tack

Kalon Bienz  
   EE 329-01 F'24

Group H

   2024-Oct-25

**EE 329 A7**

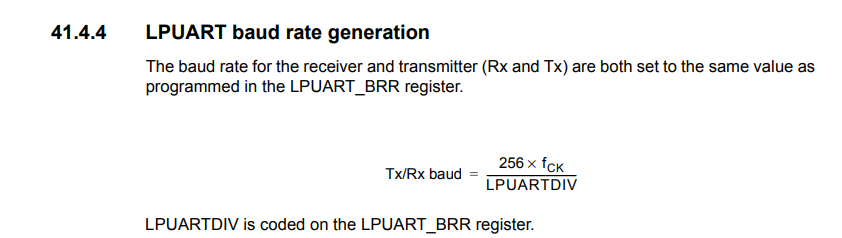
This code is designed to use the UART peripheral to communicate across USB to the targeted device, using a VT100 Terminal. The STM32 produces text that acts as a rudimentary game, displaying a splash screen with a bubbles graphic, and a boarder and character that move, and scroll to the next screen when jumped. This game was used with the terminal PuTTY in mind.

**Link to YouTube Presentation:**

<https://youtu.be/cXt5tV1ykHU>

**Calculations:**

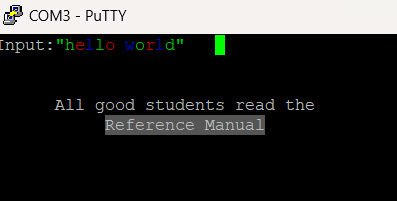
Equation A7.a(a): Baud Rate From Reference Manual:

****

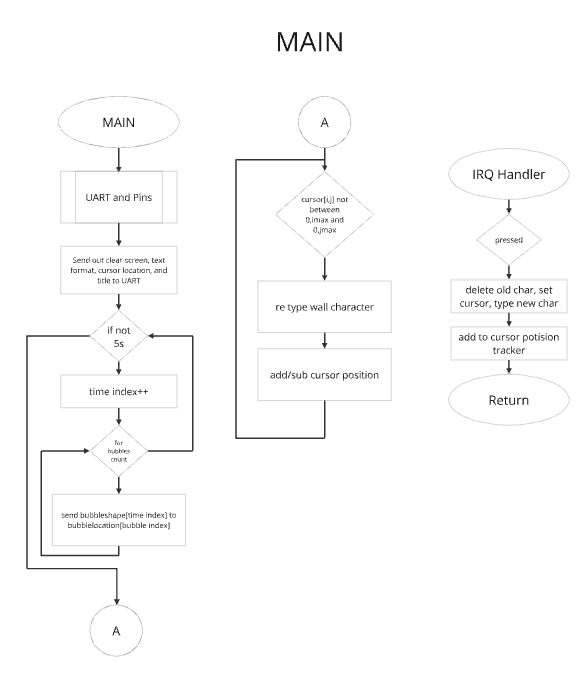
**LPUART->DIV** = 256\*4e6/115200= **8889**

**Captures:**

Figure A7.a(a): Sample Home Screen



**Pseudocode Flow Chart:**

****

**Formatted Source Code main.h:**

/\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* @file : main.h

\* project : EE329 Lab A7

\* author : Wyatt Tack (wwt) - wtack@calpoly.edu

\* date : 10/12/2024

\* firmware : ST-Link V1

\* @attention : Copyright (c) 2024 STMicroelectronics. All rights reserved.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*

\* main header for defines for C and stm32 headers/hal

\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*/

#ifndef \_\_MAIN\_H

#define \_\_MAIN\_H

#ifdef \_\_cplusplus

extern "C" {

#endif

/\* Created defines and function prototypes -----------------------------------\*/

void Execute(const char \*messages[]);

void Bubbles(void);

void LPUART1\_IRQHandler( void );

/\* Includes ------------------------------------------------------------------\*/

#include "stm32l4xx\_hal.h"

/\* Exported functions prototypes ---------------------------------------------\*/

void SystemClock\_Config(void);

void Error\_Handler(void);

#ifdef \_\_cplusplus

}

#endif

#endif

**Formatted Source Code main.c:**

/\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* @file : main.c

\* project : EE329 Lab A7

\* author : Wyatt Tack (wwt) - wtack@calpoly.edu

\* date : 10/12/2024

\* firmware : ST-Link V1

\* @attention : Copyright (c) 2024 STMicroelectronics. All rights reserved.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*

\* Device uses UART and a VT100 terminal to act as a very simple game. WASD

\* keys are used as control, and taking character off screen places it as if

\* screen scrolled over. Uses a baud rate of 115200.

\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*/

#include "main.h"

#include "delay.h"

#include "uart.h"

volatile uint16\_t cursor[2] ={0,0};

int main(void)

{

//Initialize clock, keypad, spi config

HAL\_Init();

SystemClock\_Config();

SysTick\_Init();

UART\_Init();

//homescreen lines and commands

char \*homescreen[] = {

"\e[2J", //clear screen

"\e[10;30H", //set cursor in middle

"\e[1m", //set cursor bold

"\e[31m", //set font red

"EE329 ADVENTURE", //title

NULL};

//execute homescreen text and bubbles graphic

Execute(homescreen);

Bubbles();

//screen boarder for game

char \*gamescreen[] = {

"\e[2J", //clear screen

"\e[H",

"\e[36m", //set font cyan

"\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\e[2;0H"

"| |\e[3;0H"

"| |\e[4;0H"

"| |\e[5;0H"

"| |\e[6;0H"

"| |\e[7;0H"

"| |\e[8;0H"

"| |\e[9;0H"

"| |\e[10;0H"

"| |\e[11;0H"

"| |\e[12;0H"

"| |\e[13;0H"

"| |\e[14;0H"

"| |\e[15;0H"

"| |\e[16;0H"

"| |\e[17;0H"

"| |\e[18;0H"

"| |\e[19;0H"

"|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|"

"\e[10;30H", //set cursor in middle

"\e[1m", //set cursor bold

"\e[35m", //set font purple

"&", //character

NULL};

Execute(gamescreen);

cursor[0] = 10; //set cursor

cursor[1] = 30;

//boundaries and character

while (1){

//if collision detected, repair wall and send character to other side

if (cursor[0] < 2) {

LPUART\_Print("\e[36m\e[D\_\e[D");

LPUART\_Print("\e[17B");

LPUART\_Print("\e[1m\e[35m&"); //print char

cursor[0] = 18;

}

if (cursor[0] > 18) {

LPUART\_Print("\e[36m\e[D\_\e[D");

LPUART\_Print("\e[17A");

LPUART\_Print("\e[1m\e[35m&"); //print char

cursor[0] = 2;

}

if (cursor[1] < 2) {

LPUART\_Print("\e[36m\e[D|\e[D");

LPUART\_Print("\e[58C");

LPUART\_Print("\e[1m\e[35m&"); //print char

cursor[1] = 59;

}

if (cursor[1] > 59) {

LPUART\_Print("\e[36m\e[D|\e[D");

LPUART\_Print("\e[58D");

LPUART\_Print("\e[1m\e[35m&"); //print char

cursor[1] = 2;

}

}

}

//------------------------- Movement in IRQ Handler -------------------------------

void LPUART1\_IRQHandler( void ) {

uint8\_t charRecv;

if (LPUART1->ISR & USART\_ISR\_RXNE) {

charRecv = LPUART1->RDR;

switch ( charRecv ) {

case 'w':

LPUART\_Print("\e[D \e[D");

LPUART\_Print("\e[A"); //move up

LPUART\_Print("\e[1m\e[35m&"); //print char

cursor[0] = cursor[0]-1; //move cursor

break;

case 'a':

LPUART\_Print("\e[D \e[D");

LPUART\_Print("\e[D"); //move left

LPUART\_Print("\e[1m\e[35m&"); //print char

cursor[1] = cursor[1]-1; //move cursor

break;

case 's':

LPUART\_Print("\e[D \e[D");

LPUART\_Print("\e[B"); //move down

LPUART\_Print("\e[1m\e[35m&"); //print char

cursor[0] = cursor[0]+1; //move cursor

break;

case 'd':

LPUART\_Print("\e[D \e[D");

LPUART\_Print("\e[C"); //move right

LPUART\_Print("\e[1m\e[35m&"); //print char

cursor[1] = cursor[1]+1; //move cursor

break;

default:

LPUART\_Print\_Char(charRecv); // echo char to terminal

break;

}

}

}

//---------------------------------------------------------------------------

void Execute(const char \*messages[]){

//executes all messages in script

for(uint8\_t idx = 0; messages[idx] != NULL; idx++){

LPUART\_Print(messages[idx]);

}

}

void Bubbles(void){

//Time index for home screen, bubble locations colors and time stamps

char \*bubble[] = {".","o","0","O","()","( )","\*"," "};

int timex = 0;

int bSeed[] = {1, 3, 8, 12, 16, 18, 22, 27};

char \*sSeed[] = {"\e[11;47H\e[32m", "\e[12;25H\e[36m", "\e[7;45H\e[35m", "\e[8;22H\e[33m",

"\e[8;49H\e[34m", "\e[7;25H\e[35m", "\e[12;44H\e[36m", "\e[13;35H\e[32m"};

//print bubbles around middle of screen for 5 seconds

for(int timer = 0; timer < 40; timer++){

for (int bubblex = 0; bubblex < 8; bubblex++){

if(timex > bSeed[bubblex] && timex < bSeed[bubblex]+8){

LPUART\_Print(sSeed[bubblex]);

LPUART\_Print(bubble[timex-bSeed[bubblex]]);

LPUART\_Print(" ");

LPUART\_Print("\e[H");

}

}

timex++;

delay\_us(125000);

}

}

//----------------------------- System --------------------------------------

void SystemClock\_Config(void)

{

RCC\_OscInitTypeDef RCC\_OscInitStruct = {0};

RCC\_ClkInitTypeDef RCC\_ClkInitStruct = {0};

/\*\* Configure the main internal regulator output voltage

\*/

if (HAL\_PWREx\_ControlVoltageScaling(PWR\_REGULATOR\_VOLTAGE\_SCALE1) != HAL\_OK)

{

Error\_Handler();

}

/\*\* Initializes the RCC Oscillators according to the specified parameters

\* in the RCC\_OscInitTypeDef structure.

\*/

RCC\_OscInitStruct.OscillatorType = RCC\_OSCILLATORTYPE\_MSI;

RCC\_OscInitStruct.MSIState = RCC\_MSI\_ON;

RCC\_OscInitStruct.MSICalibrationValue = 0;

RCC\_OscInitStruct.MSIClockRange = RCC\_MSIRANGE\_6;

RCC\_OscInitStruct.PLL.PLLState = RCC\_PLL\_NONE;

if (HAL\_RCC\_OscConfig(&RCC\_OscInitStruct) != HAL\_OK)

{

Error\_Handler();

}

/\*\* Initializes the CPU, AHB and APB buses clocks

\*/

RCC\_ClkInitStruct.ClockType = RCC\_CLOCKTYPE\_HCLK|RCC\_CLOCKTYPE\_SYSCLK

|RCC\_CLOCKTYPE\_PCLK1|RCC\_CLOCKTYPE\_PCLK2;

RCC\_ClkInitStruct.SYSCLKSource = RCC\_SYSCLKSOURCE\_MSI;

RCC\_ClkInitStruct.AHBCLKDivider = RCC\_SYSCLK\_DIV1;

RCC\_ClkInitStruct.APB1CLKDivider = RCC\_HCLK\_DIV1;

RCC\_ClkInitStruct.APB2CLKDivider = RCC\_HCLK\_DIV1;

if (HAL\_RCC\_ClockConfig(&RCC\_ClkInitStruct, FLASH\_LATENCY\_0) != HAL\_OK)

{

Error\_Handler();

}

}

void Error\_Handler(void)

{

\_\_disable\_irq();

while (1)

{

}

}

#ifdef USE\_FULL\_ASSERT

void assert\_failed(uint8\_t \*file, uint32\_t line)

{

}

#endif

**Formatted Source Code uart.h:**

/\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* @file : uart.h

\* project : EE329 Lab A7

\* author : Wyatt Tack (wwt) - wtack@calpoly.edu

\* date : 10/13/2024

\* firmware : ST-Link V1

\* @attention : Copyright (c) 2024 STMicroelectronics. All rights reserved.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*

\* main header for defines for uart.h

\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*/

#ifndef INC\_UART\_H\_

#define INC\_UART\_H\_

#include "stm32l4xx\_hal.h"

#define BAUD\_RATE (8889) //256 \* 4MHz / 115.2kb/s = 8888.8

void UART\_Init(void);

void LPUART\_Print( const char\* message );

//void LPUART1\_IRQHandler( void );

#endif /\* INC\_UART\_H\_ \*/

**Formatted Source Code uart.c:**

/\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* @file : uart.c

\* project : EE329 Lab A7

\* author : Wyatt Tack (wwt) - wtack@calpoly.edu

\* date : 10/13/2024

\* firmware : ST-Link V1

\* @attention : Copyright (c) 2024 STMicroelectronics. All rights reserved.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*

\* Functions for UART module, set up as LPUART1 through GPIOG.

\* PTG-7 -> Tx

\* PTG-8 -> Rx

\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*/

#include "uart.h"

// -----------------------------------------------------------------------------

void UART\_Init(void){

//Power and Clock

PWR->CR2 |= (PWR\_CR2\_IOSV); // power avail on PG[15:2] (LPUART1)

RCC->AHB2ENR |= (RCC\_AHB2ENR\_GPIOGEN); // enable GPIOG clock

RCC->APB1ENR2 |= RCC\_APB1ENR2\_LPUART1EN; // enable LPUART clock bridge

//GPIO Ports - AF8, no PU/PD, fast (despite uart being slow)

GPIOG->MODER &= ~(GPIO\_MODER\_MODE7 | GPIO\_MODER\_MODE8);

GPIOG->MODER |= (GPIO\_MODER\_MODE7\_1 | GPIO\_MODER\_MODE8\_1);

GPIOG->OTYPER &= ~(GPIO\_OTYPER\_OT7 | GPIO\_OTYPER\_OT8);

GPIOG->PUPDR &= ~(GPIO\_PUPDR\_PUPD7 | GPIO\_PUPDR\_PUPD8);

GPIOG->OSPEEDR |= ((3 << GPIO\_OSPEEDR\_OSPEED7\_Pos) | (3 << GPIO\_OSPEEDR\_OSPEED8\_Pos));

GPIOG->AFR[0] &= ~(0x000F << GPIO\_AFRL\_AFSEL7\_Pos);

GPIOG->AFR[0] |= (0x0008 << GPIO\_AFRL\_AFSEL7\_Pos);

GPIOG->AFR[1] &= ~(0x000F << GPIO\_AFRH\_AFSEL8\_Pos);

GPIOG->AFR[1] |= (0x0008 << GPIO\_AFRH\_AFSEL8\_Pos);

//LPUART

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

LPUART1->BRR = (BAUD\_RATE);

/\* USER: set baud rate register (LPUART1->BRR) \*/

NVIC->ISER[2] = (1 << (LPUART1\_IRQn & 0x1F)); // enable LPUART1 ISR

\_\_enable\_irq();

}

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

}

}

void LPUART\_Print\_Char (uint8\_t charRecv){

while( !(LPUART1->ISR & USART\_ISR\_TXE) );// wait for empty TX buffer

LPUART1->TDR = charRecv; // send char to terminal

}