Monty Choy, Sereen Benchohra
EE329-03
Fri - 11/05/21

Assignment 6: UART

Baud Rate Calculations

f_{clk} = 4MHz
Baud Rate = 115.2 kbps

USART DIV = f_{clk}/Baud Rate
= 4MHz/115.2 kbps
USART DIV = 34.722 = 34

Oversample by 16, so OVER8 = 0 and BRR = USART DIV = 34x

Demo: https://photos.app.goo.gl/CzfWu4CJMxJatszD9

main.c

```
/**
  Monty Choy
  EE329-03
  Fri, 11/05/21

  Assignment 5: UART
    UART implementation to print characters to screen, echo transmitted content,
    and change text colors.
*/

/* Includes ------------------------------------------------------------------*/
#include "main.h"
#include "uart.h"


/* Macros --------------------------------------------------------------------*/


/* Function Prototypes  ------------------------------------------------------*/
void SystemClock_Config(void);
```

```c
int main(void) {
  // Pre-included config stuff
  HAL_Init();
  SystemClock_Config();

  // My init stuff
  initUART();

  // Part 2: Use VT100 Escape Codes
  sendESCCodeUART("[3B");
  sendESCCodeUART("[5C");
  printStringUART("All good students read the");

  sendESCCodeUART("[1B");
  sendESCCodeUART("[21D");
  sendESCCodeUART("[5m");
  printStringUART("Reference Manual");

  sendESCCodeUART("[H");
  sendESCCodeUART("[0m");
  printStringUART("Input: ");





  // Main loop
  while (1) {
    // Part 1: Interface the STM32L4 to a Serial Terminal
//    printStringUART("a", 1);

  } // main while loop

} // main()
```

```
  }
```

```c
/**
  * @brief System Clock Configuration
  * @retval None
  */
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();
  }
}

/**
  * @brief  This function is executed in case of error occurrence.
  * @retval None
  */
void Error_Handler(void)
{
  /* USER CODE BEGIN Error_Handler_Debug */
  /* User can add his own implementation to report the HAL error return state */
  __disable_irq();
  while (1)
  {
  }
  /* USER CODE END Error_Handler_Debug */
}


/************************ (C) COPYRIGHT STMicroelectronics *****END OF FILE****/
```

uart.h

```
/*
 * uart.h
 *
 *  Created on: November 05, 2021
 *      Author: Monty Choy
 */

#ifndef SRC_UART_H_
#define SRC_UART_H_



/* Macros --------------------------------------------------------------------*/
#define AF7 0x7
#define ESC 0x1B



/* "Public" Stuff ------------------------------------------------------------*/
void initUART(void);
void printStringUART(char*);
void sendESCCodeUART(char*);


/* "Private" Stuff -----------------------------------------------------------*/
void configGPIOs(void);
void printCharUART(char);


#endif /* SRC_UART_H_ */
```

uart.c

```c
/*
 * dac.c
 *
 *  Created on: November 05, 2021
 *      Author: Monty Choy
 *
 *  All functions to implement UART
 */

#include "main.h"
#include "uart.h"


/* "Public" Stuff ------------------------------------------------------------*/
void initUART() {
  /**
    * Init UART2 by config UART, GPIOs
    *
    * Params: None
    * Returns: None
  */

  // Enable peripheral clk for UART
  RCC->APB1ENR1 |= (RCC_APB1ENR1_USART2EN);

  // Define word length for 8 data bits: M[1:0] = 0b00
  USART2->CR1 &= ~(USART_CR1_M0 | USART_CR1_M1);

  // Set bitrate to 115.2 kbps by setting BRR to 34
  USART2->BRR = 34;

  // Set 1 stop by by seeing STOP[1:0] = 0b00
  USART2->CR2 &= ~(USART_CR2_STOP_0 | USART_CR2_STOP_1);

  // Enable USART
  USART2->CR1 |= (USART_CR1_UE);

  // Enable interrupt on recieve by setting RXNEIE
  USART2->CR1 |= (USART_CR1_RXNEIE);
  NVIC->ISER[1] = (1 << (USART2_IRQn & 0x1F));
  __enable_irq();

  // Clear receive interrupt flag
  USART2->ISR &= ~(USART_ISR_RXNE);


  configGPIOs();

  // Enable USART2 transmit
  USART2->CR1 |= (USART_CR1_TE);

  // Enable USART2 receive
  USART2->CR1 |= (USART_CR1_RE);


}


void printStringUART(char* str) {
  /**
    * Print string over UART
    *
    * Params: None
    * Returns: None
  */

  for (uint8_t i = 0; str[i] != '\0'; i++) {
    while (!(USART2->ISR & USART_ISR_TXE));
    printCharUART(str[i]);
  }

}
```

```
void sendESCCodeUART(char* str) {
  /**
    * Print an ESC Code over UART by sending an ESC code before sending the
    * string
    *
    * Params: None
    * Returns: None
  */

  printCharUART(ESC);
  printStringUART(str);

}


void USART2_IRQHandler(void) {
  /**
    * Handle interrupt when a character is received. Read the RDR and change
    * text color or echo if applicable.
    *
    * Params: None
    * Returns: None
  */

  // Check if USART2 RXNE caused the interrupt
  if (USART2->ISR & USART_ISR_RXNE) {
    // Read the received data
    uint8_t receivedChar = USART2->RDR;

    switch(receivedChar) {
      case 'R': {
        sendESCCodeUART("[31m");
        break;
      }
      case 'B': {
        sendESCCodeUART("[34m");
        break;
      }
      case 'G': {
        sendESCCodeUART("[32m");
        break;
      }
      case 'W': {
        sendESCCodeUART("[37m");
        break;
      }
      default: {
        printCharUART(receivedChar);
      }
    }

    // Clear ISR flag
    USART2->ISR &= ~(USART_ISR_RXNE);
  }

}



/* "Private" Stuff -----------------------------------------------------------*/
void configGPIOs() {
  /**
    * Config GPIOs for UART, TX (PA2) and RX (PA3)
    *
    * Params: None
    * Returns: None
  */

  // Enable peripheral clk for GPIOA
  RCC->AHB2ENR |= (RCC_AHB2ENR_GPIOAEN);


  // Set GPIOs to alternate function
  GPIOA->MODER |= (
    GPIO_MODER_MODE2_1 |
    GPIO_MODER_MODE3_1
  );
  GPIOA->MODER &= ~(
    GPIO_MODER_MODE2_0 |
    GPIO_MODER_MODE3_0
  );

  // Set AF to AF7, USART2
  GPIOA->AFR[0] |= (
    (AF7 << GPIO_AFRL_AFSEL2_Pos) |
    (AF7 << GPIO_AFRL_AFSEL3_Pos)
  );

}


void printCharUART(char charToPrint) {
  /**
    * Print single char over UART
    *
    * Params: None
    * Returns: None
  */

  // Wait until TX buffer is empty
  while (!(USART2->ISR & USART_ISR_TXE));
    USART2->TDR = charToPrint;
}
```