Wyatt Tack
Yazmin Corona
EE 329-01 F'24
Group H
2024-Nov-15

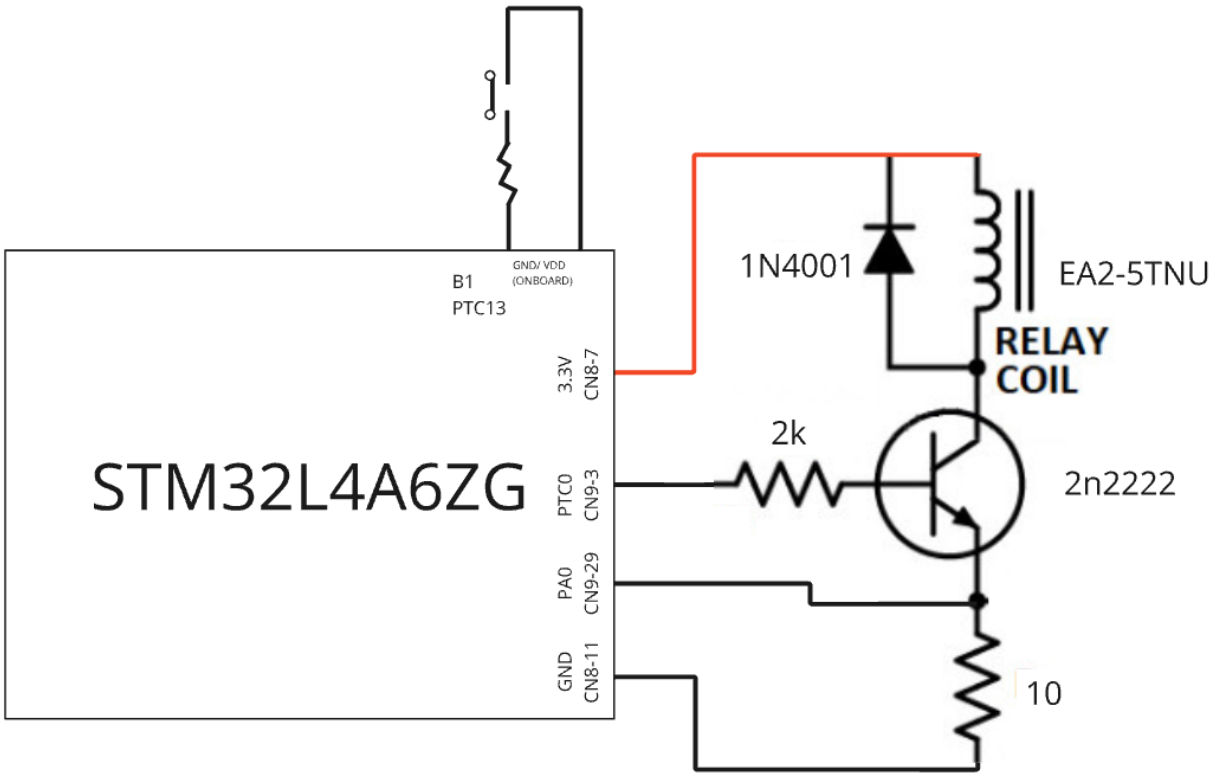
EE 329 A8

This code is designed to use the built in STM32's Analog to Digital Converter (ADC) to measure the emitter voltage of a BJT used to drive a relay, and thus attempt to calculate the current through the relay coil knowing some previous parameters. The code works and is able to use a UART terminal to display the changing voltage average, max, and min of each sample batch, and show the current estimation.

Video Link:

https://youtu.be/WX_pdkR84_8

Device Wiring Diagram:



Data:

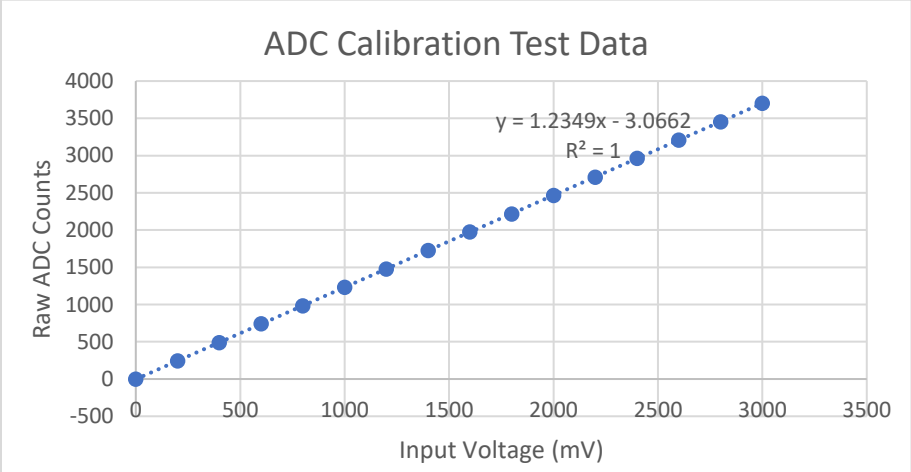
$$\text{ADC Resolution for FSR 3.3V} = \frac{3.3V}{2^{12}} = \mathbf{0.806mV}$$

Table ADC vs Voltage:

Calibration:

Vin (Volts)	ADC Counts
0	0
0.2	243
0.4	489
0.6	742
0.8	981
1	1231
1.2	1477
1.4	1726
1.6	1976
1.8	2218
2	2467
2.2	2712
2.4	2962
2.6	3208
2.8	3453
3	3704

Graph ADC vs Voltage:



Adjustment:

Used trendline to put mV as an expression of ADC counts. Multiplied regression constants by 10k to remove decimal values (all values are left in millivolt integers).

```
minVolt = (minVal(ADC_Samples)*10000 + 30662) / 12349;
```

Table ADC values for sample times:

	Clock Cycles = 47.5		Clock Cycles = 640.5	
	Counts	Volts	Counts	Volts
Min	1838	1.490	1851	1.501
Max	1853	1.503	1858	1.507
Avg	1845	1.496	1854	1.503

Relay Circuitry Measurements:

V_{CESAT}	**0.1658 V**
V_{BE}	**0.7472 V**
V_{RE}	**0.2671 V**
I_B	**1.1531 mA**
V_{coil}	**4.6127 V**
I_C	**24.432 mA**

Formatted Source Code main.h:

```
/**
  ******************************************************************************
  * @file                  : main.h
  * project               : EE329 Lab A8
  * author                : Wyatt Tack (wwt) - wtack@calpoly.edu
  * date                  : 11/6/2024
  * 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

#include <stdint.h>

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


#define RESISTOR (10113)  //in milliohms
#define IBASE (1153)              //in microamps
#define BUTTON_PORT GPIOC
#define LED_PORT GPIOB
#define BUTTON_PRESS ((((~(GPIOC->IDR)) & GPIO_PIN_13) >> 13))
void Led_Config(void);
void Button_Config(void);
void Output_Config(void);

uint16_t maxVal (uint16_t* nums );
uint16_t minVal (uint16_t* nums );
uint16_t avgVal (uint16_t* nums );
void HomeScreen(void);
void sendHexData(uint16_t minVal, uint16_t maxVal, uint16_t avgVal);
void sendVoltData(uint16_t minVal, uint16_t maxVal, uint16_t avgVal);
void parseSendHex(uint16_t value, uint8_t row);
void parseSendVolt(uint16_t value, uint8_t row);
void sendCurrentData(uint16_t value);

/* 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 A8
  * author        : Wyatt Tack (wwt) - wtack@calpoly.edu
  * date          : 11/6/2024
  * firmware      : ST-Link V1
  * @attention    : Copyright (c) 2024 STMicroelectronics. All rights reserved.
  ******************************************************************************
  *
  *       Device uses ADC1 at GPIO A0 to measure relay current through
  *       established parameters along with measuring emitter voltage
  *       of the BJT used to drive the circuitry.
  *
  *       PA0       - ADC at emitter of BJT
  *       PC0       - Base driver output
  *       PC13 - Button input to drive software
  *
  *       Wire Relay coil +/- to Flyback diode anode/cathode respectively
  *       Relay coil + to 3.3V, relay - to BJT Collector
  *       BJT base to 2k resistor into PC0
  *       BJT emitter to 10 resistor into gnd
  *       PA0 at BJT emitter
  *
  ******************************************************************************
  */

#include "main.h"
#include "delay.h"
#include "uart.h"
#include "adc.h"

volatile uint16_t ADC_Samples [20];
volatile uint16_t ADC_SampleCount;

int main(void)
{
          //Initialize clock
          HAL_Init();
          SystemClock_Config();
          SysTick_Init();
          UART_Init();
          ADC_init();
          HomeScreen();
          Button_Config();
          Output_Config();
          uint16_t minVolt, maxVolt, avgVolt, current;

          while (1){
                    if(!BUTTON_PRESS)            //if pushed
                              GPIOC->BSRR |= (GPIO_PIN_0); //turn on relay
                    else
                              GPIOC->BRR |= (GPIO_PIN_0);
                    ADC1->CR |= ADC_CR_ADSTART;  //start ADC
                    while(ADC_SampleCount <= 19);         //wait for array to fill
                    delay_us(500000);
          sendHexData(minVal(ADC_Samples), maxVal(ADC_Samples), avgVal(ADC_Samples));
                    minVolt = (minVal(ADC_Samples)*10000 + 30662) / 12349;
                    maxVolt = (maxVal(ADC_Samples)*10000 + 30662) / 12349;
                    avgVolt = (avgVal(ADC_Samples)*10000 + 30662) / 12349;
          sendVoltData(minVolt, maxVolt, avgVolt);
                    current = (1000*avgVolt*1000/RESISTOR - IBASE)/1000;
                              //(nV/mOhms - uA) --> mA
                              sendCurrentData(current);
          }
}
```

Formatted Source Code main.c (Continued):

```
//--------------------------- Screen --------------------------------------

void HomeScreen(void){
            char *homescreen[] = {
                                    "\e[2J",                            //clear screen
                                    "\e[?25l",                          //hide cursor
                                    "\e[3;5H",                          //set cursor home
                                    "\e[33m",                           //set text yellow
                                    "ADC counts volts \e[4;5H",
                                    "MIN  0000  0.000 V \e[5;5H",
                                    "MAX  0000  0.000 V \e[6;5H",
                                    "AVG  0000  0.000 V \e[7;5H",
                                    "coil current = 0.000 A",
                                    NULL};
            for(uint8_t idx = 0; homescreen[idx] != NULL; idx++)
                        LPUART_Print(homescreen[idx]);
}


void sendHexData(uint16_t minVal, uint16_t maxVal, uint16_t avgVal){
            parseSendHex(minVal, 1);
            parseSendHex(maxVal, 2);
            parseSendHex(avgVal, 3);
}
void sendVoltData(uint16_t minVal, uint16_t maxVal, uint16_t avgVal){
            parseSendVolt(minVal, 1);
            parseSendVolt(maxVal, 2);
            parseSendVolt(avgVal, 3);
}
void sendCurrentData(uint16_t value){
            uint8_t valTho = (value/1000);
            uint8_t valHun = (value - valTho*1000)/100;
            uint8_t valTen = (value - valTho*1000 - valHun*100)/10;
            uint8_t valOne = (value - valTho*1000 - valHun*100 - valTen*10);
            LPUART_Print("\e[7;20H");
            LPUART_Print_Char(valTho+0x30);                 //send A.BDC
                        LPUART Print("\e[C");                          //skip period
            LPUART_Print_Char(valHun+0x30);
            LPUART_Print_Char(valTen+0x30);
            LPUART_Print_Char(valOne+0x30);
}

void parseSendHex(uint16_t value, uint8_t row){
            uint8_t valTho = (value/1000);
            uint8_t valHun = (value - valTho*1000)/100;
            uint8_t valTen = (value - valTho*1000 - valHun*100)/10;
            uint8_t valOne = (value - valTho*1000 - valHun*100 - valTen*10);
            switch (row){                                                           //pick max/min/avg row
            case 1:
                        LPUART_Print("\e[4;10H");
            break;
            case 2:
                        LPUART_Print("\e[5;10H");
            break;
            case 3:
                        LPUART_Print("\e[6;10H");
            break;
            }
            LPUART_Print_Char(valTho+0x30);                 //send ABCD
            LPUART_Print_Char(valHun+0x30);
            LPUART_Print_Char(valTen+0x30);
            LPUART_Print_Char(valOne+0x30);
}

void parseSendVolt(uint16_t value, uint8_t row){
            uint8_t valTho = (value/1000);
            uint8_t valHun = (value - valTho*1000)/100;
            uint8_t valTen = (value - valTho*1000 - valHun*100)/10;
            uint8_t valOne = (value - valTho*1000 - valHun*100 - valTen*10);
            switch (row){                                                           //pick max/min/avg row
            case 1:
                        LPUART_Print("\e[4;16H");
            break;
            case 2:
                        LPUART_Print("\e[5;16H");
            break;
            case 3:
                        LPUART_Print("\e[6;16H");
            break;
            }
            LPUART_Print_Char(valTho+0x30);                 //send A.BDC
                        LPUART_Print("\e[C");                          //skip period
            LPUART_Print_Char(valHun+0x30);
            LPUART_Print_Char(valTen+0x30);
            LPUART_Print_Char(valOne+0x30);
}
```

Formatted Source Code main.c (Continued):

```
//--------------------------- GPIO ----------------------------------------

void Led Config(void)
{                               // configure GPIO pin PB14 for:
                                // output mode, push-pull, no pull up or pull down, high speed
  RCC->AHB2ENR   |=  (RCC_AHB2ENR_GPIOBEN);
  LED_PORT->MODER   &= ~(GPIO_MODER_MODE14);
  LED_PORT->MODER   |=  (GPIO_MODER_MODE14_0);
  LED_PORT->OTYPER  &= ~(GPIO_OTYPER_OT14);
  LED PORT->PUPDR   &= ~(GPIO PUPDR PUPD14);
  LED PORT->OSPEEDR |=  (3 << GPIO OSPEEDR OSPEED14 Pos);
  LED PORT->BRR |= (GPIO PIN 14);
}

void Output Config(void)
{                               // configure GPIO pin PB14 for:
                                // output mode, push-pull, pull down, high speed
  RCC->AHB2ENR   |=  (RCC_AHB2ENR_GPIOCEN);
  GPIOC->MODER   &= ~(GPIO MODER MODE0);
  GPIOC->MODER   |=  (GPIO MODER MODE0 0);
  GPIOC->OTYPER  &= ~(GPIO OTYPER OT0);
  GPIOC->PUPDR   &= ~(GPIO PUPDR PUPD0);
  GPIOC->PUPDR   |= (GPIO PUPDR PUPD0 1);
  GPIOC->OSPEEDR |=  (3 << GPIO OSPEEDR OSPEED0 Pos);
  GPIOC->BRR |= (GPIO PIN 0);
}

void Button Config(void)
{                               // configure GPIO pin PC13 for:
                                // Button pin initialize - Input, on board PU
                RCC->AHB2ENR   |=  (RCC AHB2ENR GPIOCEN);
                BUTTON PORT->MODER   &= ~(GPIO MODER MODE13);
                BUTTON PORT->PUPDR   &= ~(GPIO PUPDR PUPD13);
}

//--------------------------- MATH ----------------------------------------
uint16_t maxVal (uint16_t* nums ){
                uint16 t max = nums[0];
                for (int index = 0; index < sizeof(nums)/sizeof(nums[0]); index++)
                                 if (nums[index] > max)
                                     max = nums[index];
                return max;
}

uint16_t minVal (uint16_t* nums ){
                uint16_t min = nums[0];
                for (int index = 0; index < sizeof(nums)/sizeof(nums[0]); index++)
                                  if (nums[index] < min)
                                      min = nums[index];
                return min;
}

uint16_t avgVal (uint16_t* nums ){
                uint16_t avg = 0;
                uint16_t arraySize = sizeof(nums)/sizeof(nums[0]);
                for (int index = 0; index < arraySize; index++)
                                avg = avg + nums[index];
                avg = avg/arraySize;
                return avg;
}

//---------------------------- 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 adc.h:

```
/**
******************************************************************************
***
  * @file               : adc.h
  * project            : EE329 Lab A8
  * author             : Wyatt Tack (wwt) - wtack@calpoly.edu
  * date               : 11/6/2024
  * firmware           : ST-Link V1
  * @attention         : Copyright (c) 2024 STMicroelectronics. All rights
reserved.

******************************************************************************
***
  *
  *     main header for defines for adc.h
  *

******************************************************************************
***
  */
#ifndef SRC_ADC_H_
#define SRC_ADC_H_

#include "stm32l4xx_hal.h"

void ADC_init(void);
void ADC1_2_IRQHandler(void);

extern volatile uint16_t ADC_Samples [20];
extern volatile uint16_t ADC_SampleCount;

#endif /* SRC_ADC_H_ */
```

Formatted Source Code adc.c:

```
/**
  ******************************************************************************
  * @file             : adc.c
  * project           : EE329 Lab A8
  * author            : Wyatt Tack (wwt) - wtack@calpoly.edu
  * date              : 10/28/2024
  * firmware          : ST-Link V1
  * @attention        : Copyright (c) 2024 STMicroelectronics. All rights reserved.
  ******************************************************************************
  *
  *        Functions for ADC. Initializes ADC1 at PA0 (channel 5). Interrupt
  *        flag samples for 20 samples, global array must be included in main along
  *        with global sample count. Once array fulfilled ADC stops sampling.
  *
  ******************************************************************************
  */

#include "adc.h"
#include "delay.h"


void ADC_init(void) {
RCC->AHB2ENR |= RCC_AHB2ENR_ADCEN;          // turn on clock for ADC
// power up & calibrate ADC
ADC123_COMMON->CCR |= (1 << ADC_CCR_CKMODE_Pos); // clock source = HCLK/1
ADC1->CR &= ~(ADC_CR_DEEPPWD);              // disable deep-power-down
ADC1->CR |= (ADC_CR_ADVREGEN);              // enable V regulator - see RM 18.4.6
delay_us(20);                               // wait 20us for ADC to power up
ADC1->DIFSEL &= ~(ADC_DIFSEL_DIFSEL_5);     // PA0=ADC1_IN5, single-ended
ADC1->CR &= ~(ADC_CR_ADEN | ADC_CR_ADCALDIF); // disable ADC, single-end calib
ADC1->CR |= ADC_CR_ADCAL;                   // start calibration
while (ADC1->CR & ADC_CR_ADCAL) {;}         // wait for calib to finish
// enable ADC
ADC1->ISR |= (ADC_ISR_ADRDY);               // set to clr ADC Ready flag
ADC1->CR |= ADC_CR_ADEN;                    // enable ADC
while(!(ADC1->ISR & ADC_ISR_ADRDY)) {;}     // wait for ADC Ready flag
ADC1->ISR |= (ADC_ISR_ADRDY);               // set to clr ADC Ready flag
// configure ADC sampling & sequencing
ADC1->SQR1  |= (5 << ADC_SQR1_SQ1_Pos);     // sequence = 1 conv., ch 5
ADC1->SMPR1 |= (0x7 << ADC_SMPR1_SMP5_Pos);  // ch 5 sample time = 640.5 clocks
ADC1->CFGR  &= ~( ADC_CFGR_CONT  |          // single conversion mode
                  ADC_CFGR_EXTEN |          // h/w trig disabled for s/w trig
                  ADC_CFGR_RES   );         // 12-bit resolution
// configure & enable ADC interrupt
ADC1->IER |= ADC_IER_EOCIE;                 // enable end-of-conv interrupt
ADC1->ISR |= ADC_ISR_EOC;                   // set to clear EOC flag
NVIC->ISER[0] = (1<<(ADC1_2_IRQn & 0x1F)); // enable ADC interrupt service
__enable_irq();                             // enable global interrupts
// configure GPIO pin PA0
RCC->AHB2ENR  |= (RCC_AHB2ENR_GPIOAEN);     // connect clock to GPIOA
GPIOA->AFR[0] &= ~(GPIO_AFRL_AFSEL0);                       // clear alt. function select
GPIOA->AFR[0] |= (7 << GPIO_AFRL_AFSEL0_Pos);          // choose AF 7 (PA0=ADC1_IN5)
GPIOA->MODER  |= (GPIO_MODER_MODE0);        // analog mode for PA0 (set MODER last)
ADC1->CR |= ADC_CR_ADSTART;
}


void ADC1_2_IRQHandler(void){
           if(ADC1->ISR & ADC_ISR_EOC){
                     uint16_t adcData = ADC1->DR;
                     if(ADC_SampleCount <= 19){
           //if array not full
                               ADC_Samples [ADC_SampleCount] = adcData;    //fill array
                               ADC_SampleCount++;
                     //incriment to next sample
                               ADC1->CR |= ADC_CR_ADSTART;
           //start next conv
                     }
                     else {
                               ADC_SampleCount = 0;
           //if array full reset index
                               ADC1->ISR &= ~(ADC_ISR_EOC);                              //clear flag
                     }
           }
}
```

Formatted Source Code uart.h:

```
/**

******************************************************************************
***
  * @file               : uart.h
  * project            : EE329 Lab A4
  * 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 LPUART_Print_Char (uint8_t charRecv);


#endif /* INC_UART_H_ */
```

Formatted Source Code uart.c:

```
/**
  ******************************************************************************
  * @file                    : uart.c
  * project                  : EE329 Lab A4
  * 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
                    }
```