   Wyatt Tack

Yazmin Corona  
   EE 329-01 F'24

Group H

   2024-Nov-15

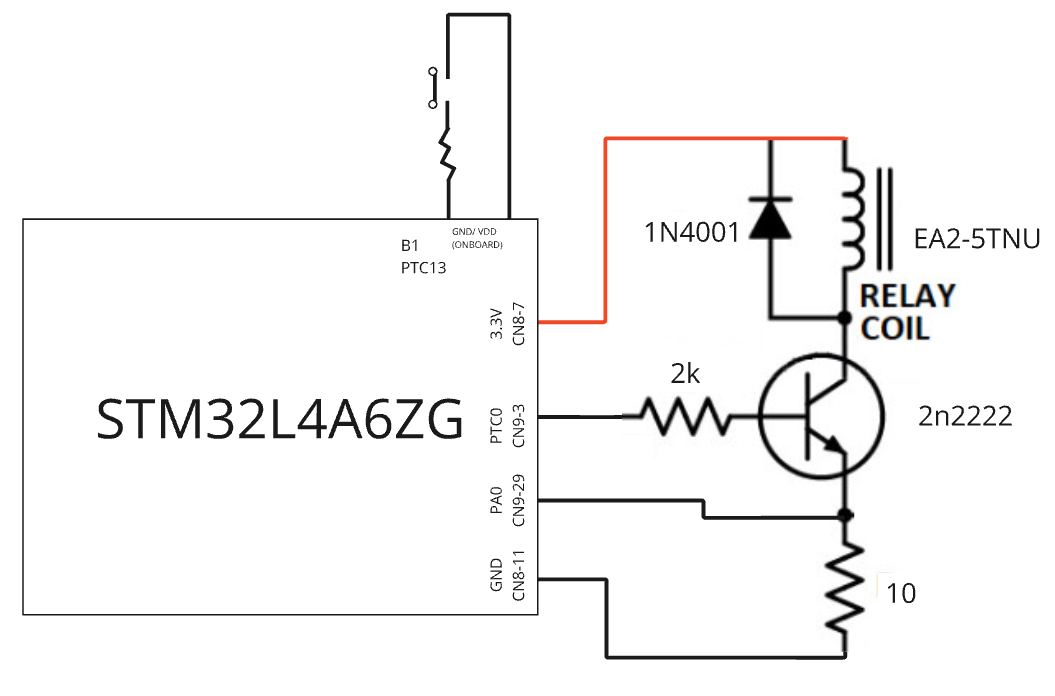
**EE 329 A8**

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:**

ADC Resolution for FSR 3.3V =

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).



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:

|  |  |
| --- | --- |
| **VCESAT** | **0.1658 V** |
| **VBE** | **0.7472 V** |
| **VRE** | **0.2671 V** |
| **IB** | **1.1531 mA** |
| **Vcoil** | **4.6127 V** |
| **IC** | **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

}