Lab 1 – Introduction to USART

EE390 – Smart Sensor Systems

ECET Department

Minnesota State University, Mankato

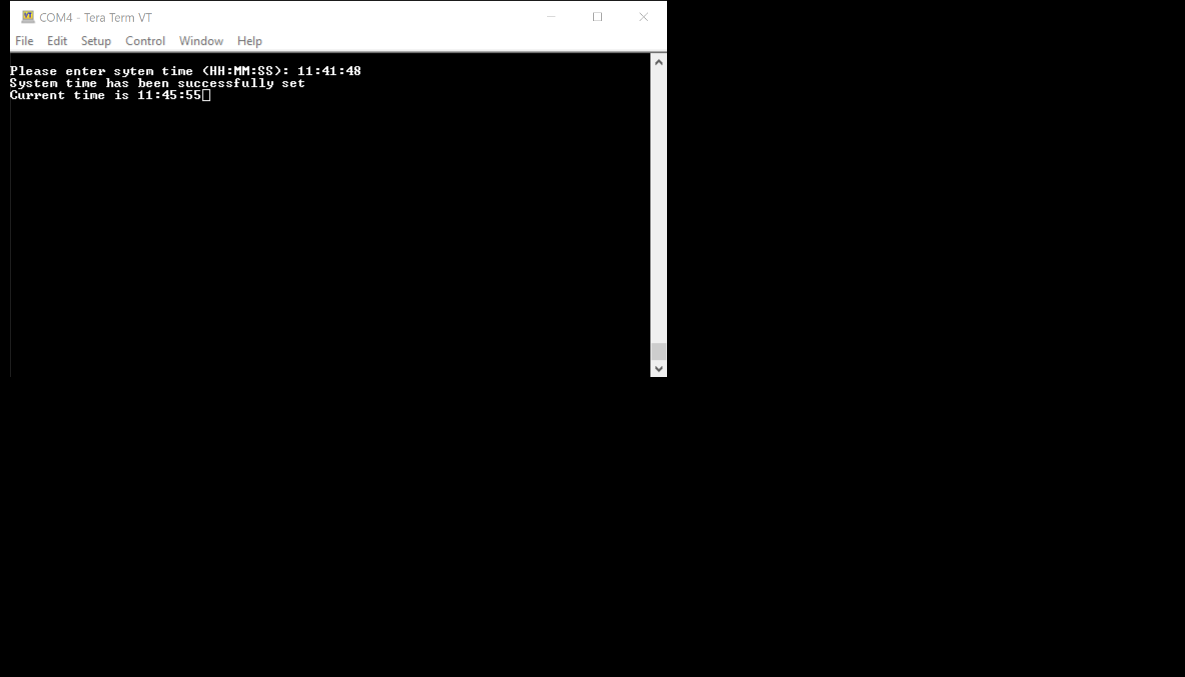
Colin Roskos

1/27/2023

Purpose:

This lab is designed to reintroduce to the STM Discovery board and the ST32L475 processor through USART.

Program Description:



The program enters the main function through the file main.c, we initialize the system clock entering sysClkInit(), we initialize USART using initUART(), then we enter the user interface program portion of the code entering setSysTime(), then printing this system time using printSystemTime(). After the user input phase we enter a forever loop where we wait 1000ms when calling secTick() and then reprinting the time portion using rePrintSystemTime(). Further descriptions are given below.

**sysClkInit() – Sets the system clock to 80MHz using the HSI clock.**

void sysClkInit()

{

system\_clock\_init\_HSI\_80MHz();

}

void system\_clock\_init\_HSI\_80MHz(void){

// Enable High Speed Internal Clock (HSI = 16 MHz)

RCC->CR |= ((uint32\_t)RCC\_CR\_HSION);

// wait until HSI is ready

while ( (RCC->CR & (uint32\_t) RCC\_CR\_HSIRDY) == 0 );

PWR->CR1 &= ~PWR\_CR1\_VOS;

PWR->CR1 |= PWR\_CR1\_VOS\_0;

while(PWR->SR2 & PWR\_SR2\_VOSF);

FLASH->ACR |= FLASH\_ACR\_LATENCY\_4WS; /\* Frequency is above 64MHz, 4 WS required \*/

FLASH->ACR |= FLASH\_ACR\_PRFTEN; /\* prefetch enable \*/

RCC->CR &= ~(RCC\_CR\_PLLON);

while(RCC->CR & RCC\_CR\_PLLRDY)

;

// choose HSI as PLL Input

RCC->PLLCFGR = (RCC\_PLLCFGR\_PLLSRC\_HSI + RCC\_PLLCFGR\_PLLN\_4 +

RCC\_PLLCFGR\_PLLN\_2 + RCC\_PLLCFGR\_PLLR\_0);

// Wait till PLL is used as system clock source

RCC->CR |= RCC\_CR\_PLLON;

while ((RCC->CR & RCC\_CR\_PLLRDY) == 0)

;

RCC->PLLCFGR = RCC\_PLLCFGR\_PLLREN;

// Select PLL as system clock source

RCC->CFGR &= ~(RCC\_CFGR\_SW + RCC\_CFGR\_HPRE);

RCC->CFGR |= (uint32\_t)RCC\_CFGR\_SW\_PLL + RCC\_CFGR\_HPRE\_1; //11: PLL oscillator used as system clock, divide clock by 1

while ((RCC->CFGR & (RCC\_CFGR\_SW\_PLL + RCC\_CFGR\_HPRE\_1)) == 0 ); //11: PLL oscillator used as system clock, divide clock by 1

SystemCoreClockUpdate();

}

**initUART() – initializes uart using USART1 and a 115200 baud rate, 8 bits of data, no parity, and 1 stop.**

void initUART ()

{

RCC->AHB2ENR |= RCC\_AHB2SMENR\_GPIOBSMEN;

RCC->APB2ENR |= RCC\_APB2ENR\_USART1EN;

// Configure PB6 & PB7 for alternate function

GPIOB->MODER &= ~(GPIO\_MODER\_MODE6 + GPIO\_MODER\_MODE7); // clear bits bit 12~15

GPIOB->MODER |= GPIO\_MODER\_MODE6\_1 + GPIO\_MODER\_MODE7\_1;

// Assign PB6 & PB7 pins to alternate function 7

GPIOB->AFR[0] &= ~(GPIO\_AFRL\_AFSEL6 + GPIO\_AFRL\_AFSEL7); // clear bits 24~31

GPIOB->AFR[0] |= (GPIO\_AFRL\_AFSEL6\_2 + GPIO\_AFRL\_AFSEL6\_1 + GPIO\_AFRL\_AFSEL6\_0 + \

GPIO\_AFRL\_AFSEL7\_2 + GPIO\_AFRL\_AFSEL7\_1 + GPIO\_AFRL\_AFSEL7\_0);

USART1->CR1 &= ~(USART\_CR1\_M1 + USART\_CR1\_M0 + USART\_CR1\_OVER8); // select 8 data bits

USART1->BRR = 0x2B6; // set baud rate to 115,200 baud

USART1->CR2 &= ~(USART\_CR2\_STOP); // select one stop bit

USART1->CR3 &= ~(USART\_CR3\_ONEBIT); // use 3 samples to detect start bit, determine bit value

// enable USART1, enable transmitter, receiver

USART1->CR1 |= USART\_CR1\_UE + USART\_CR1\_TE + USART\_CR1\_RE;

}

**setSysTime() – Prompts the user for a system time in format HH:MM:SS, prompting the user of successful and unsuccessful inputs.**

void setSysTime(){

char\* msgBuf;

msgBuf = (char\*) malloc(sizeof(char) \* 128);

putsUART(SYSTIME\_REQUEST\_INPUT);

getsUART(msgBuf);

putsUART(processMsg(msgBuf)); // process message and respond feedback to the user

free(msgBuf);

}

char\* processMsg(char\* pmsg)

{

uint8\_t numPort = 0; // numeric portion for input processing

// buffers for System Time input checking

uint8\_t hrs = 0;

uint8\_t min = 0;

uint8\_t sec = 0;

// start input check with fisrt character in pmsg (char at index 0)

uint8\_t i = 0;

char msgChar = pmsg[i];

while(i < SYSTIME\_TIME\_FORMAT\_WIDTH && msgChar)

{

// check that characters passed are valid

if (msgChar > ASCII\_COLON) return SYSTIME\_INVALID\_CHARACTER;

else if (msgChar < ASCII\_ZERO) return SYSTIME\_INVALID\_CHARACTER;

else{} // rest of code in loop

// an ASCII number is 0x30 <= Num < 0x3A, the 0x30 case is taken care of

if (msgChar != ASCII\_COLON)

{

if(numPort++ > 1) return SYSTIME\_INVALID\_FORMAT;

if(i < 2) // if in hours section

{

hrs = hrs \* 10; // if second digit, shift existing digit to tens place

hrs += msgChar - ASCII\_ZERO; // assign ones place

}

else if(i < 5) // if in minutes seciton

{

min = min \* 10; // if second digit, shift existing digit to tens place

min += msgChar - ASCII\_ZERO; // assign ones place

}

else // in seconds section

{

sec = sec \* 10; // if second digit, shift existing digit to tens place

sec += msgChar - ASCII\_ZERO; // assign ones place

}

}

// if msg char is ":" it must have exactly 2 numbers preceding it

else

{

if (numPort != 2) return SYSTIME\_INVALID\_FORMAT;

numPort = 0;

}

// incrementors

msgChar = pmsg[++i];

}

// check time

if (hrs > 23 || min > 60 || sec > 60) return SYSTIME\_INVALID\_TIME;

// set valid time to systemTime

systemTime.hrs = hrs;

systemTime.min = min;

systemTime.sec = sec;

return SYSTIME\_SUCSESS;

}

**printSystemTime() – Prompts the user for a system time in format HH:MM:SS, prompting the user of successful and unsucces**

void setSysTime(){

char\* msgBuf;

msgBuf = (char\*) malloc(sizeof(char) \* 128);

putsUART(SYSTIME\_REQUEST\_INPUT);

getsUART(msgBuf);

putsUART(processMsg(msgBuf)); // process message and respond feedback to the user

free(msgBuf);

}

char\* processMsg(char\* pmsg)

{

uint8\_t numPort = 0; // numeric portion for input processing

// buffers for System Time input checking

uint8\_t hrs = 0;

uint8\_t min = 0;

uint8\_t sec = 0;

// start input check with fisrt character in pmsg (char at index 0)

uint8\_t i = 0;

char msgChar = pmsg[i];

while(i < SYSTIME\_TIME\_FORMAT\_WIDTH && msgChar)

{

// check that characters passed are valid

if (msgChar > ASCII\_COLON) return SYSTIME\_INVALID\_CHARACTER;

else if (msgChar < ASCII\_ZERO) return SYSTIME\_INVALID\_CHARACTER;

else{} // rest of code in loop

// an ASCII number is 0x30 <= Num < 0x3A, the 0x30 case is taken care of

if (msgChar != ASCII\_COLON)

{

if(numPort++ > 1) return SYSTIME\_INVALID\_FORMAT;

if(i < 2) // if in hours section

{

hrs = hrs \* 10; // if second digit, shift existing digit to tens place

hrs += msgChar - ASCII\_ZERO; // assign ones place

}

else if(i < 5) // if in minutes seciton

{

min = min \* 10; // if second digit, shift existing digit to tens place

min += msgChar - ASCII\_ZERO; // assign ones place

}

else // in seconds section

{

sec = sec \* 10; // if second digit, shift existing digit to tens place

sec += msgChar - ASCII\_ZERO; // assign ones place

}

}

// if msg char is ":" it must have exactly 2 numbers preceding it

else

{

if (numPort != 2) return SYSTIME\_INVALID\_FORMAT;

numPort = 0;

}

// incrementors

msgChar = pmsg[++i];

}

// check time

if (hrs > 23 || min > 60 || sec > 60) return SYSTIME\_INVALID\_TIME;

// set valid time to systemTime

systemTime.hrs = hrs;

systemTime.min = min;

systemTime.sec = sec;

return SYSTIME\_SUCSESS;

}

void printSystemTime(void)

{

putsUART(SYSTIME\_OUTPUT\_PREFIX);

displaySystemTime();

}

void displaySystemTime(void)

{

char \* msgBuf;

msgBuf = (char\*) malloc(sizeof(char) \* 9);

msgBuf[0] = int2char(systemTime.hrs/10); //10s place

msgBuf[1] = int2char(systemTime.hrs);

msgBuf[2] = ASCII\_COLON;

msgBuf[3] = int2char(systemTime.min/10); //10s place

msgBuf[4] = int2char(systemTime.min);

msgBuf[5] = ASCII\_COLON;

msgBuf[6] = int2char(systemTime.sec/10); //10s place

msgBuf[7] = int2char(systemTime.sec);

msgBuf[8] = ASCII\_EOL;

putsUART(msgBuf);

free(msgBuf);

}

**secTick() – Waits 1ms using the wait\_ms\_80MHz() call then updates the system time to the correct H, M, S**

void secTick(void)

{

wait\_ms\_80MHz(1000);

systemTime.sec++;

if(systemTime.sec == 60)

{

systemTime.sec = 0;

systemTime.min++;

}

if(systemTime.min == 60)

{

systemTime.min = 0;

systemTime.hrs++;

}

if(systemTime.hrs == 24)

{

systemTime.hrs = 0;

}

}

**rePrintSystemTime() – prints the updated system time every 1s in HH:MM:SS format, deleting the 8 characters for the system time output**

void rePrintSystemTime(void)

{

backspaceUART(8); // backup output by 8 chars

displaySystemTime(); // print system time HH:MM:SS

}

**Program List:**

main.c/.h

systime.c/.h

usart.c/.h

sysclk.c/.h

system.c/.h

startup\_stm32l475xx.s

system\_stm32l4xx.c

stm32l475xx.h

