

CSE 379
Lab #3
Part 1
Spring 2022

Objective

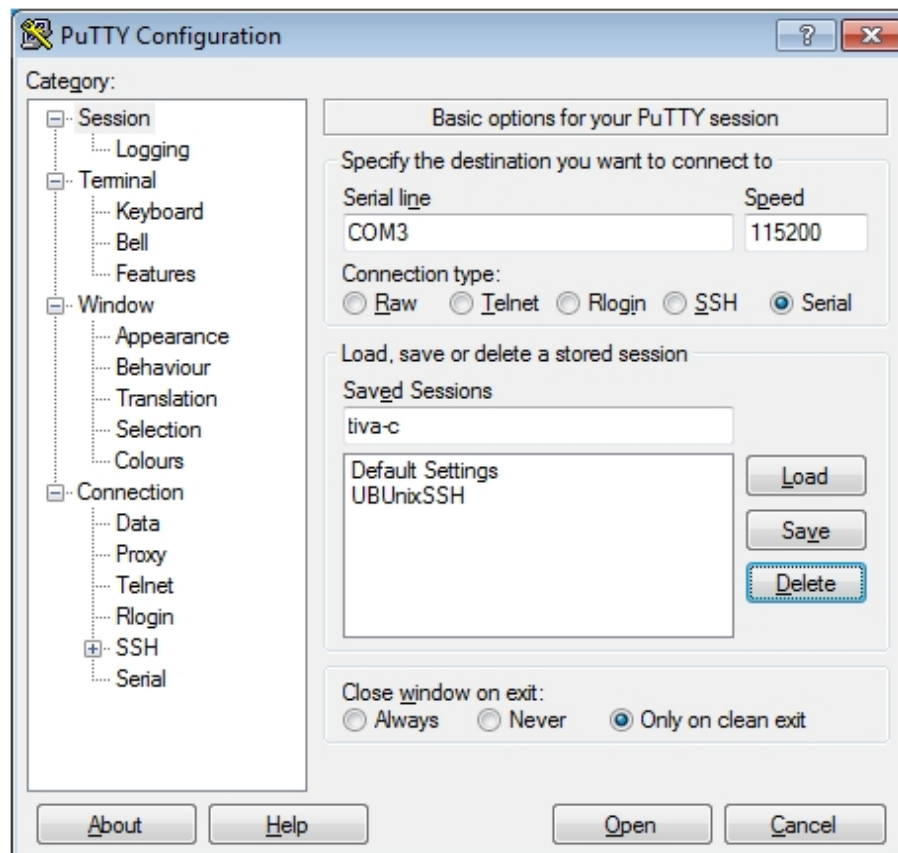
In this lab, you will learn how to communicate with the ARM processor via a serial connection using the UART (universal asynchronous receiver transmitter). In doing so, you will also learn how to write and use assembly language subroutines.

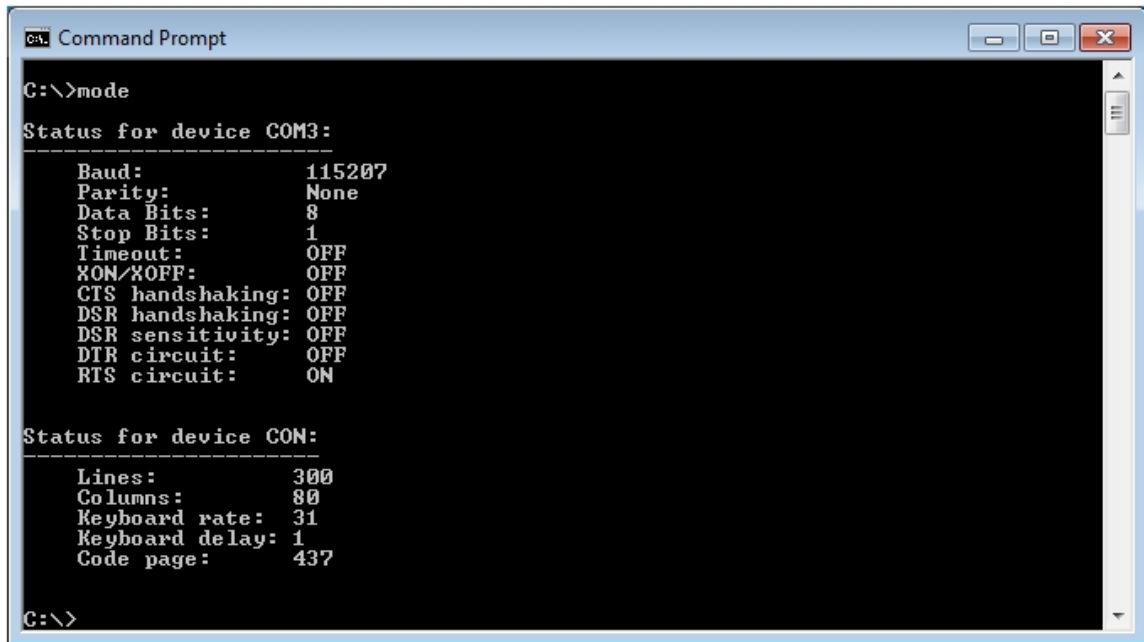
Description

Write two (2) ARM assembly language subroutines, called *output_character* and *read_character*. These subroutines will allow a user to enter a character and display the character in *PuTTY* via the UART. All user input should be echoed back to the display.

PuTTY Setup

Open *PuTTY* (*Start* → *All Programs* → *PuTTY* → *PuTTY*) to communicate with the board. Select *Serial* as the connection type. To determine which COM port the board is connected to, go to *Start*, and in the search bar type *cmd* and press *Enter*. In the command line, type *mode* and press *Enter*. Typically, the highest numbered COM port corresponds to the ARM board. In *PuTTY* enter *COMx* as the serial line, where *x* is the number of the COM port you determined using *mode* command in the previous step. Set the baud rate to 115,200 in the *speed* field. Save the session by entering *Tiva-c* under *Saved Sessions* and then hit *Save*. Load the session called *Tiva-c* by selecting *Tiva-c* from the list and clicking on the *Load* button. The figures below will help you navigate the setup.





```
cmd. Command Prompt

C:\>mode

Status for device COM3:
-----
Baud:          115200
Parity:         None
Data Bits:      8
Stop Bits:      1
Timeout:        OFF
XON/XOFF:       OFF
CTS handshaking: OFF
DSR handshaking: OFF
DSR sensitivity: OFF
DTR circuit:    OFF
RTS circuit:    ON

Status for device CON:
-----
Lines:          300
Columns:        80
Keyboard rate:  31
Keyboard delay: 1
Code page:      437

C:\>
```

Testing

The following C program can be used to test your code. The program first initializes the UART by calling *serial_init*, and then calls *lab3*, which is the assembly language subroutine that contains your code. This code is available on the *Labs* page of the course website. The routine *lab3* should call *read_character* followed by *output_character* to allow the user to enter a character and then echo that character back to the display.

```
#include <stdint.h>
extern int lab3(void);
void serial_init(void);

void serial_init(void)
{
    /******
    /* When translating the following to assembly */
    /* it is advised to use LDR and STR as opposed */
    /* to LDRB and STRB. */
    /******
    /* Provide clock to UART0 */
    (*(volatile uint32_t *) (0x400FE618)) = 1;
    /* Enable clock to PortA */
    (*(volatile uint32_t *) (0x400FE608)) = 1;
    /* Disable UART0 Control */
    (*(volatile uint32_t *) (0x4000C030)) = 0;
    /* Set UART0_IBRD_R for 115,200 baud */
    (*(volatile uint32_t *) (0x4000C024)) = 8;
    /* Set UART0_FBRD_R for 115,200 baud */
    (*(volatile uint32_t *) (0x4000C028)) = 44;
    /* Use System Clock */
    (*(volatile uint32_t *) (0x4000CFC8)) = 0;
    /* Use 8-bit word length, 1 stop bit, no parity */
}
```

```

    (*(volatile uint32_t *) (0x4000C02C)) = 0x60;
    /* Enable UART0 Control */
    (*(volatile uint32_t *) (0x4000C030)) = 0x301;
    /******
    /* The OR operation sets the bits that are OR'ed */
    /* with a 1. To translate the following lines */
    /* to assembly, load the data, OR the data with */
    /* the mask and store the result back. */
    /******
    /* Make PA0 and PA1 as Digital Ports */
    (*(volatile uint32_t *) (0x4000451C)) |= 0x03;
    /* Change PA0,PA1 to Use an Alternate Function */
    (*(volatile uint32_t *) (0x40004420)) |= 0x03;
    /* Configure PA0 and PA1 for UART */
    (*(volatile uint32_t *) (0x4000452C)) |= 0x11;
}

int main()
{
    serial_init();
    lab3();
}

```

Skeleton Code

The following ARM assembly language skeleton code can be used to get you started. It is available online on the *Labs* page of the course website.

```

.text

.global lab3

U0FR:    .equ 0x18    ; UART0 Flag Register

lab3:
    PUSH {lr}    ; Store lr to stack

    ; Your code is placed here.

    POP {lr}    ; Restore lr from stack
    mov pc, lr

output_character:
    PUSH {lr}    ; Store register lr on stack

    ; Your code to output a character to be displayed in PuTTY
    ; is placed here. The character to be displayed is passed
    ; into the routine in r0.

    POP {lr}
    mov pc, lr

```

```

read_character:
    PUSH {lr}    ; Store register lr on stack

    ; Your code to receive a character obtained from teh keyboard
    ; in PuTTY is placed here.  The character is received in r0.

    POP {lr}
    mov pc, lr

.end

```

Partners

You will work with partner you selected during the week you worked on Lab #2. ***One partner is responsible for writing `output_character` and the other is responsible for writing `read_character`. They must be done individually! It is recommended that you test `output_character` first, and then use `output_character` to echo the character back when testing `read_character`. When testing, both partners should be present and participate in the debugging process. It is also important to note that both partners must understand both routines!***

Demonstrations & Submissions

You MUST demonstrate to your TA that your *output_character* and *read_character* routines work properly before starting the second part of this lab. The due date for getting *output_character* and *read_character* working is the end of your lab session on February 21 or 22 (depending upon which day you have lab).