ENSE 452 - Lab2
CLI on the Target

Trevor Douglas
Software Systems Lab Instructor
University of Regina

1 Objective

The objective is to set up a Command-Line Interface (CLI) through which you can communicate with your target board. Such a tool is extremely useful for run time communication with your target.

First you will be enabling an onboard USART and establishing simple polled serial communication with a terminal program (e.g. Putty, or TeraTerm) running on the host machine.

Next, you will design a set of commands and responses such that you can type the commands at your CLI prompt, and expect to see various behaviours on the target board as well as textual responses inside the CLI. One useful exercise will incorporate LED control into your CLI.

At each step of the development, we will be paying attention to good software design principles. This means creating a src directory and an include directory for any header files needed.

2 Equipment and Software Requirements

- PC for developing as well as viewing Zoom Presenations.
- Keil uVision
- Nucleo-64 development kit and USB cable.
- Com client such as Putty.
- Git (Git client or use command line).

3 Procedure

- Create a new Lab2 subdirectory inside your repository on your local machine.
- Create a new STM32 project and add and commit the project to your repository.

To get USART 2 working (Semi-hosting), you will need to configure the follwing registers:

- APB1ENR
- APB2ENR
- CRL OF PORTA
- USART2 CRR
- USART2 CR1
- USART2 CR2

3.1 Phase 1: Get the Serial Port Working

To communicate with the target we will setup USART 2. This USART is already connected to the target via the USB cable and used for debugging. Sharing this connection with our own serial connection is called semi hosting. The I/O Pins that need to be setup are Pin PA2 for Tx and Pin PA 3 for Rx. You should create a usart.c and usart.h for this phase.

- Ensure the the Port A clock is enabled.
- Ensure the USART 2 clock is enabled.
- Configure PA2 for alternate function output Push-pull mode, max speed 50 MHz.
- Configure PA3 for Input with pull-up / pull-down.
- Enable the USART Tx and Rx in the USART Control register.
- Configure USART 2 for 115200 bps, 8-bits-no parity, 1 stop bit. (Peripheral clock is 36MHz).
- Create a loop to send a character to the host and make sure that character is printed. This will test the Tx.
- Create a function to receive a character and echo that back to the host to make sure Rx is working.

Your Usart code should have functions like below: (include <stdint.h>and create usart.c and usart.h)

```
/** Configure and enable the device. */
void serial_open(void);

/**
Undo whatever serial_open() did, setting the peripheral back to
its reset configuration.
```

```
*/
void serial_close(void);
/**

Send an 8-bit byte to the serial port, using the configured
bit-rate, # of bits, etc. Returns 0 on success and non-zero on
failure.
@param b the 8-bit quantity to be sent.
@pre must have already called serial_open()
*/
int sendbyte(uint8_t b);

/**
Gets an 8-bit character from the serial port, and returns it.
@pre must have already called serial_open()
*/
uint8_t getbyte(void)
```

3.2 Phase 2: Improve Writing and Reading to the USART

USART 2 is working but we should create some functions that will allow us to read and write larger buffers. Create a routine to Transmit and another to Receive like this:

```
void CLI_Transmit(uint8_t *pData, uint16_t Size);

void CLI_Receive(uint8_t *pData, uint16_t Size);
```

Also, as you type characters in the host terminal, the target should echo the characters back so the user can see what they type. You should create a CLI.h and CLI.c for this section.

3.3 Phase 3: Implement the CLI and commands

The CLI should print some sort of prompt. Blank lines merely repeat the prompt. The user must be able to use the backspace key to correct typing errors. All commands must produce some textual output as confirmation the command worked, or an error message if the command was malformed. You should implement commands to do the following:

1. Turn off or on the LED.
2. Query the state of the LED.
3. "help" should print a screen of help explaining the commands.

Put this functionality into the CLI entity.

4 What To Submit

On URCourses put the link to your git repository in the submission link. The TA will checkout the version (date) submitted at the deadline.