Supervisory Control and Data Acquisition System

Camilo Loaiza

11 December 2022

Table of Contents

1. System Description

This SCADA System Project utilizes an Atmega8515 microcontroller and is developed in assembly language to manage system modes effectively. It begins by setting up the stack pointer to optimize program execution and initializes various hardware interfaces including UART, LCD, ports, and registers through dedicated initialization subroutines in the "**init**" section. Upon power-up, the "**poweru_banner**" subroutine displays a "SCADA Mon v1.0" startup message on the LCD and terminal, identifies the system ID, and clears the display after 4 seconds. The duration of the banner display is controlled by delay mechanisms integrated into the subroutine, ensuring the message remains visible for a brief period of time before clearing the display and terminal. The "**mode_sel"** subroutine determines operational mode based on the input from a switch connected to PINB, directing the system to either supervisor or node mode.

In node mode, the system utilizes the ADC to monitor analog input voltages ranging from 0 to 5V. The terminal connected via UART reflects the system's mode state and displays the related prompt ("Node>"). Single-letter commands entered into the terminal trigger specific actions:

- **'A' Command:** Initiates ADC sampling, acquiring 256 samples at 2ms intervals, storing them in a buffer named "samples". Each sample triggers a 10us pulse and is categorized as either below (L) or above (H) a default trim level of 2.5V. The count values (in hexadecimal) for L and H are displayed on the second line of the LCD, providing real-time feedback.

- **'S' Command:** Calculates and displays the sum of all samples stored in the "samples" buffer. Handles overflow conditions by incrementing a counter for each carry detected during addition. Results are shown on both the LCD and terminal.

- **'T' Command:** Allows users to set a new ADC trim level by entering two hexadecimal digits via the terminal. The new trim level is converted from ASCII to hexadecimal and stored in register R19, adjusting subsequent ADC operations accordingly. Confirmation and status messages are relayed to both the LCD and terminal.

These commands facilitate real-time interaction with the microcontroller, enabling precise control and monitoring capabilities tailored for multiple embedded system applications while ensuring visibility and feedback through multiple output channels.

2. Circuit Diagram

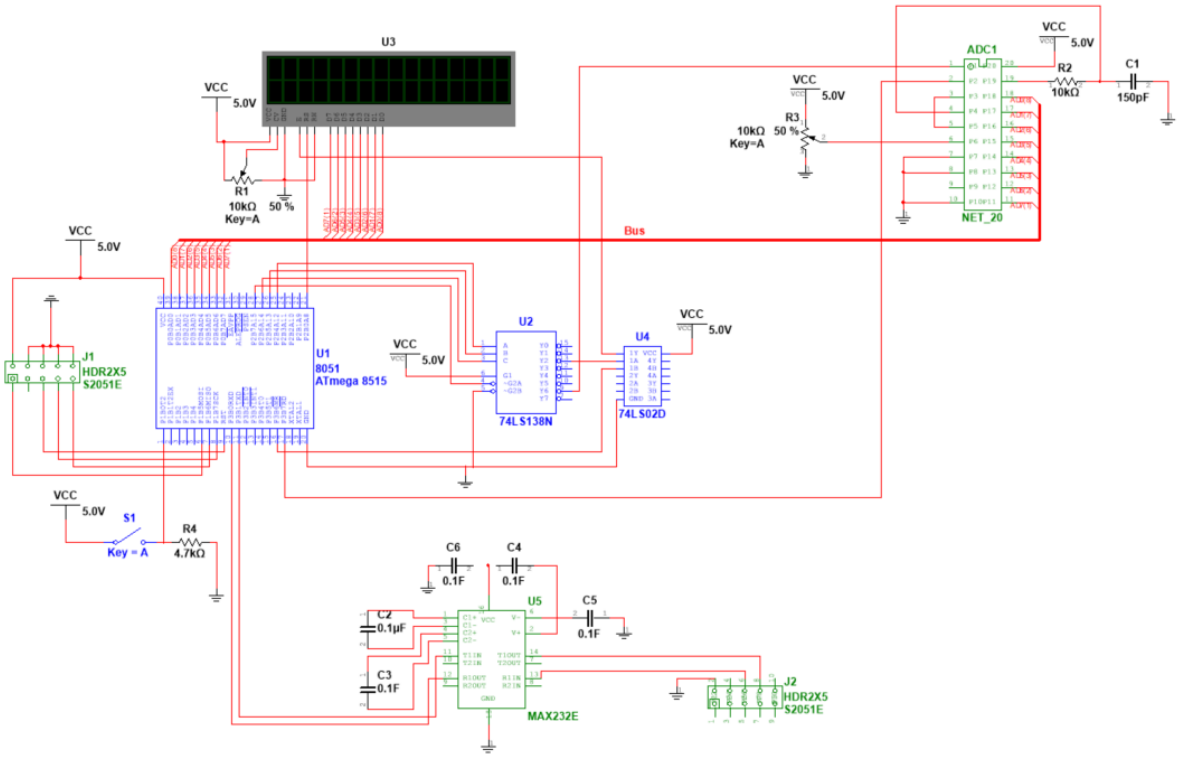

Figure 1: Circuit Diagram

The circuit diagram highlights the critical hardware setup required for this project. The Atmega8515 is at the center of the circuit, interfacing with an LCD display to provide visual output. A 74LS138N 3-to-8 line decoder and a 74LS02D quad 2-input NOR gate are included to handle various logic operations and expand the number of output lines from the microcontroller.

The circuit features an ADC0804 for converting analog signals to digital data. A variable resistor (R3) connected to the ADC serves as the input being monitored, allowing for adjustable voltage input ranging from 0 to 5V. Additional components in the ADC signal conditional circuit include another variable resistor (R2) and a capacitor (C1), which help stabilize and filter the input signal for accurate readings.

For serial communication, the MAX232E is utilized to interface with RS-232, with capacitors ensuring proper voltage level shifting. The entire circuit is powered by a 5V supply, providing stable voltage to all components for reliable operation.

This setup enables the Atmega8515 microcontroller to effectively manage the system, with various peripherals supporting its functionality and allowing real-time monitoring and control of the input signal via the ADC.

3. System Output Showcase

- **Powerup Banner:** Upon powering up, the system displays a distinctive startup banner on both the interface and the LCD screen that serves as an initial visual confirmation of successful initialization.

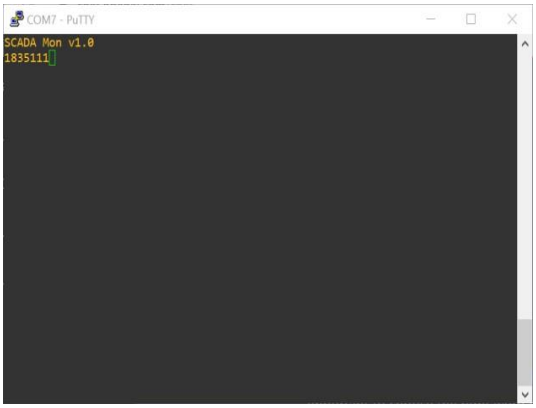

Figure 2: Powerup banner on terminal

Figure 3: Powerup banner on L

- **Standby Mode:** The LCD screen alternates between displaying "Node Mode" and "Mode Standby" to indicate the current operational state of the SCADA system.

Figure 4: Mode Standby

- **Acquire samples:** Figure 5 depicts the terminal interface in Node Mode where the command "A" initiates ADC sampling. The system promptly confirms execution with "Command Completed!", showcasing its real-time responsiveness and ability to perform continuous analog data acquisition crucial for embedded system applications. Figure 6 and 7 reflect the results of the sample when it is lower or higher than the trim level respectively.

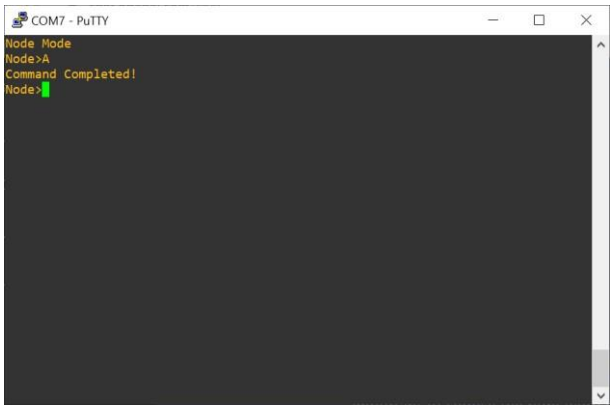

Figure 5: Acquire command on terminal

Figure 6: Acquire sample when sample lower than trim level

Figure 7: Acquire sample when samples higher than trim level

- **Display sum of samples:**

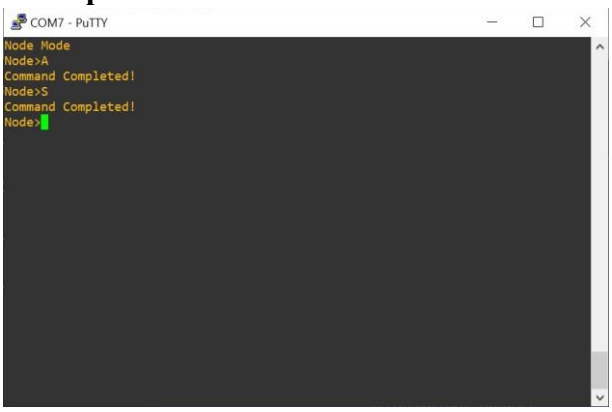

Figure 8: Sum command on terminal

Figure 9: Sum command when sample lower than trim level

Figure 10: Sum command when sample higher than trim level

- **Set new ADC trim level:**

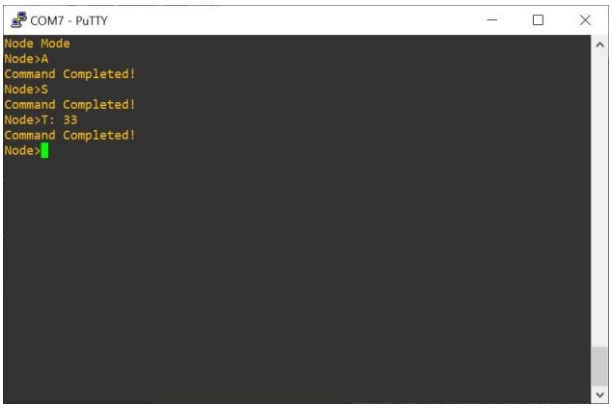

Figure 11: Trim command on terminal

4. Discussion

- **Initialization:** This code starts by initializing the stack pointer by setting the low and high memory addresses to the end of the RAM memory, this will allow calling the correct use by the processor of calls and returns. The init section calls several subroutines to initialize multiple components of the microcontroller, such as the external memory interface, UART, the LCD, PORTs and registers.

- **Powerup Banner:** After the external memory interface and LCD initialization, the "poweru_banner" subroutine is called which is responsible for calling internal subroutines that load the power-up message stored in flash and displaying it, placing the LCD's cursor on the second line, displaying the ID number, displaying the entire powerup message for four seconds, and finally clearing the display and terminal.

- **Mode Select:** The "mode_sel" subroutine determines the operation mode of the system, node mode or supervisor mode, depending on the input on PINB, which is read from a switch connected to PB0. If the input is low, the subroutine selects node mode, and if the input is high, the subroutine selects supervisor mode. After the mode is selected, the subroutine performs different and specific actions for the selected mode.

- **Node Mode:** In node mode, the ADC is used to monitor an analog input voltage ranging from 0 to 5V.The terminal will be expecting a single-letter command to perform different actions. When a command is performed, a message will notify it on the terminal, as well as when an invalid input is detected.

- **Acquire samples:** When commanded from the serial terminal, with the command "A", the ADC, found at memory the $2000 memory address, will acquire 256 samples every 2 milliseconds, and store them in a memory buffer called "samples" while generating a 10us pulse every time a new sample is acquired. Then, the same buffer will be compared with the trim level value to determine if is lower or higher, incrementing a counter stored in register to conclude the number of samples for each aspect. The values of these counters (in hex) are converted to ASCII using the hex2asc subroutine found in the library "numio.inc" provided by the teacher. The number of samples will be then displayed on the LCD.

- **Display sum of samples:** When the 'S' command is entered, the program will load each value of the memory buffer and perform an addition with the previous value until the whole buffer (255 samples) is processed. The loop will check each time for a carry flag, if the addition sets the flag, it will branch to a different section where a counter will be increment which represents the carry of the sum. After the loop is finished, the program will once again convert the hex values into ASCII using the "numio.inc" library. Then, it will clear the second line of the LCD, and display the sum of the samples.

- **Set new ADC trim level:** The trim level is set at start-up to $87 (in theory 2.5V is $7F, but since it is calibrated to my specific ADC, the value I obtained was $87 which is within the +/-8 range), and its value, which is always stored in register R19, does not change throughout the program unless the “T” command is entered. This command expects two hexadecimal digits that represent the new trim level of the program. The “new_trim” subroutine gets the ASCII characters input by the user, converts them into ASCII using a combination of AND and ROL instructions, add them together, and stores the hexadecimal value in R19.