Text

Description automatically generated

Embedded System Design Lab 3 Report

Date: 24th October 2021

Name: Dhiraj Bennadi

**Objective:**

* Learn how to configure the 8051 for serial communication and how to write serial device drivers.
* Learn how to use internal XRAM and external XRAM (using the NVSRAM).
* Begin learning how to use a compiler (e.g. SDCC) to develop C programs.
* Learn how to initialize hardware properly in C.
* Continue learning about the ARM architecture and development environment

**Summary of Tasks:**

* Configured the Hardware to use NVSRAM as additional XRAM data memory for the system. The NVSRAM has control signals from the microcontroller. A15 address line, Write Pin and Read Pin from the microcontroller was connected to Chip Enable, Write Enable and Output Enable Pins of the NVSRAM.
* Enabled the XRS1 and XRS0 bit fields in the AUX Register to increase the space available on XRAM to 1024 bytes.
* Programmed the Processor with PAULMON2 monitor program along with extra.asm and verified the working.
* Verified the locations of XRAM from 0x0000 to 0x7FFF for write values with various input values.
* Understood the working of SDCC compiler to generate hex records with various other compiler options.
* Developed and executed a program in C to allocate a heap of 5000 bytes and accept various input from the user to execute commands to allocate memory, free memory, print the statistics of the heap, display the hex dump of the buffer allocated in the memory, empty the buffer.
* Implemented the debug port to write values to memory locations and observing the output.
* Developed and executed a program to C on MSP432 Platform to use the inbuilt peripherals such UART, Timer as PWM, ADC to measure temperature and accept user inputs to alter the PWM signal, Variation of the Duty Cycle based on the temperature measurements, Variation of the duty cycle based on the switch press and displaying of information on the terminal window.
* Understood the functionality of Programmable Counter Array and generated PWM signals, Idle Mode, Power Down Mode, Max and Minimum frequency of the peripheral clock
* Implemented the Optional Challenge to write a function in assembly and call the function in a C file.
* Implemented the Optional Challenge to implement a Interrupt Driven Serial Port Driver based on UART, and compared the efficiency with the Polled I/O approach.
* Implemented the Optional Challenge to understand the heap management and the issues of memory fragmentation.

**Critical Readings**

**Part 1**

Verification of flashing of Paulmon Monitor Program and extra.asm and writing a value to the locations from 0x0000-0x7FFF

Text

Description automatically generated

Value of the maximum baud rate for which the Paulmon Program was modified.

Diagram

Description automatically generated

The Paulmon Program provides the option to select the baud rate or allow for a automatic detection of baud rate and different rates.

The working of the program was verified on both the automatic detection and higher speeds.

![Text

Description automatically generated with medium confidence]()

**Part 2:**

The elements of the program developed was showcased to the TA during the signoff.

The program had a user menu to select among the various options to allocate memory in the heap, free the allocated memory, add/delete new buffers onto the delete, provide the heap statistics, provide a hexdump of the buffer and delete all the allocated buffers.

**Part 3:**

The elements of the program developed was showcased to the TA during the signoff.

The program demonstrated the working of UART, PWM, ADC. The program had options to alter the duty cycle by providing commands from the keyboard over UART, to display the reading of Duty Cycle, temperature in degrees, Fahrenheit and Kelvin. The Duty cycle also changed for every 0.5 degree change in the temperature.

**Initial Duty Cycle of 55 %**

Graphical user interface, diagram

Description automatically generated

**Increase of Duty Cycle by user command**

Graphical user interface, diagram

Description automatically generated

**Decrease of Duty Cycle by User command**

Graphical user interface, diagram

Description automatically generated

**Increase of Period by User command**

![Text

Description automatically generated]()

**Decrease of Period by User command**

![Text

Description automatically generated]()

**Variations of Duty Cycle due to Period Change**

Graphical user interface, diagram

Description automatically generated

Graphical user interface, diagram

Description automatically generated

**Default Duty Cycle by user command**

![Text

Description automatically generated]()

Graphical user interface, diagram

Description automatically generated

**Temp Variation by user command**

![Text

Description automatically generated]()

**Temperature Displayed in Degree, Celsius and Kelvin on user command**

![Text

Description automatically generated with medium confidence]()

**Supplemental Element**

The Supplemental Elements to change the units of temperature measurements, to alter the duty cycle of the PWM cycle and Period of the duty cycle is also shown in the above images.

**Generation of PWM using the PCA of Atmel AT89C51RC2**

Diagram

Description automatically generated with medium confidence

**FCLK PERIPHERAL Maximum Frequency as supported by the CKRL Register**

Graphical user interface

Description automatically generated

**FCLK PERIPHERAL Minimum Frequency as supported by the CKRL Register**

Diagram

Description automatically generated

**IDLE Mode**

In the IDLE Mode the ALE and PSEN are high

Graphical user interface

Description automatically generated

Graphical user interface

Description automatically generated

**Power Down Mode**

In the Power down mode the oscillator is stopped.

Graphical user interface

Description automatically generated with medium confidence

**Optional Elements:**

The Optional Elements for the Assembly Interfacing with C and Heap Management and the issue of memory fragmentation was shown to the TA during the Signoff.

The System Performance between Polling and Interrupt based Serial Port Driver was demonstrated with the glowing of LED and terminal screenshots.

On the terminal, a character (In this case ‘z’) was continuously printed, and the user was asked to input characters on the terminal. In case of Polling the system was less responsive to the inputs from the user as the system was busy continuously polling the busy flag of UART. In case of Interrupt based the system was more responsive to the inputs of the user.

**Polling Based Driver**

A picture containing text

Description automatically generated

**Interrupt Based Driver**

A picture containing text

Description automatically generated

**Key Learnings from TA Comments**

1. Deallocation of memory
2. Heap Management
3. Timer usage to set sample rate for ADC
4. PCA Modes