**ECEN – 5613 – EMBEDDED SYSTEM DESIGN**

**LAB\_2 WRITEUP – SRICHARAN KIDAMBI**

CONTENTS:

PART - I

1. Assembly program to blink an LED via External Addressing a 32KB NVRAM.
2. Writing SPLD WINCUPL firmware to enable access to external device.

PART – II

1. Disable External Addressing and Enabling the BLINK – LED operation using the 4KB ROM of the Microcontroller.
2. DEBUGGING the circuit via a DEBUG latch by attaching values for every jump to main and ISR routine in the above Assembly Program.
3. Basics of ARM programming in Embedded C to perform
   1. BLINK LED with 300ms
   2. TOGGLE BETWEEN LED in a fixed time and enabling a switch which controls the toggle operation or not.

**PART – I**

**Objective:** To write a basic assembly level programming for 8051 and configuring basic timers and interrupts to blink an LED at 0.39 second interval.

**Tools Required:**

1. M48Z35Y – NVRAM: A 32KB RAM.
2. ATF16V8B – SPLD to control the configuration of NVRAM.
3. Siemens Microcontroller (C51).
4. Soldering and Wire wrapping Tools.

**Procedure**:

1. Solder the RAM components and ensure the interconnections between MCU and RAM are neatly designed and labelled according to the below diagram.

Diagram, schematic

Description automatically generated

1. Write a basic SPLD configuration to ensure that the output enable of the NVRAM (/CE) is not grounded, rather controlled by A15 signal. The program should be like the below.

CE = A15

Please find the full code in the SRICHARANLAB2PART1.PLD file.

1. Once you successfully designed, compiled, simulated, and dumped to the SPLD code to the SPLD device. Start with writing the Assembly program.
2. Firmware Details of Assembly Program:
   1. Timer0 operated in Mode 1
   2. ISR vector addressing to 000Bh.
   3. 16- Bit counter UP Counter loaded with a value of approximately 39ms.
   4. Needed value = (65536 – load value) \* 1.085us.
   5. The above operation is made to function for 10 times after which the interrupt is told to address the main logic.
   6. Once addressed, the timer is again restarted with initial values.

Program found in LABPart.asm in the attached WINZIP file.

1. The LED should toggle at 0.39 msec intervals now.

**Observations**

LOGIC Analyzer is used to view the LAB results for this part.

**STATE MODE RESULTS**

Address transaction for address 0.

Graphical user interface

Description automatically generated

Address transaction for address E.

Graphical user interface, application

Description automatically generated

You can compare this with data present on LABPart.lst file.

**TIMING MODE RESULTS**

Timing Mode uses the clock of the MCU rather than the clock of Logic analyzer to view the results.

A screenshot of a computer

Description automatically generated

The above screenshot mentions that PSEN drops a bit later than ALE so that the bus cycle transaction happens seamlessly. Failing which, the MCU can be a cycle ahead before reaching the data from the previous cycle.

Theoretical Time = (1/11.0592) – 25ns = 90.4 – 25 = 65.4 ns

Tllpl time = PSEN – ALE ~= 100ns – Practical situations take a little bit more time. This may be due to:

1. Improper hardware setup.
2. Minor defects with microcontroller as well as LOGIC Analyzer.

**KEY LEARNINGS from LAB2\_Part1**

1. Understood how to configure timers and interrupts on assembly level language for 8051.
2. Obtained good insights on 8051 bus cycle and bus transactions.
3. How to calculate the time between ALE drop and PSEN drop in a program memory read cycle.
4. How to effectively program memory into an NVRAM and verify the contents using chip programmer to ensure that dumping is done correctly.
5. Learnt how to use a logic analyser and verify accurate results.
6. Learnt the difference between timing mode and state mode of logic analyser.

**RESULT:** The requirements are fulfilled, hardware and software devised properly, and parameters verified successfully.

**PART – II**

**OBJECTIVE:**

1. To disable external addressing and ensure that the assembly program mentioned above is working with internal memory of AT89C52 Atmel microcontroller.
2. Add a debug latch and verify its working by adding values to it.
3. Write basic ARM programs in embedded C to understand timers and interrupts on ARM Cortex M4 processor.

**TOOLS REQUIRED:**

1. MAX 232
2. RS 232
3. AT89C52
4. SN74LS374 DEBUG LATCH
5. 5 1uF Capacitors
6. 3 Pin header for EA
7. 5.1K to act as a pull-down resistor for PSEN.

**PROCEDURE:**

1. Ensure that connections are made right from the MAX232 to RS232 as well as MAX232 to Microcontroller.
2. Ensure that the Microcontroller is in bootloader mode. i.e RESET address = 00H and SRAM enabled.
3. Make connections according to below diagram.

Diagram, schematic

Description automatically generatedDiagram

Description automatically generated

1. Keep the MCU in reset mode. (Hold the reset signal, press the PSEN signal, leave the reset first, PSEN second – The system is now in bootloader mode.
2. Program the device using FLIP and ensure that LED blink is functioning properly.
3. Add a debug latch and add values to the latch each time the data passes through that.
4. Find the assembly code for this in Supplement.asm file.
5. Find the values using logic analyser.
6. Program the basic interrupts and timer program in ARM processor – Programs found in folder lab2 and lab2\_2.

**OBSERVATIONS**

**LOGIC ANALYZER OUTPUT**![A screenshot of a computer

Description automatically generated with medium confidence]()

**DSO OUTPUT**

**MCU INPUT – A15**

Chart

Description automatically generated with medium confidence

**SPLD INPUT – A15**

Diagram

Description automatically generated with medium confidence

**SPLD INPUT - WR**

Diagram

Description automatically generated with medium confidence

**SPLD OUTPUT – !A15 & !WR**

Diagram

Description automatically generated

**KEY LEARNINGS**

1. I am now comfortable with configuring basic timer and interrupts in ARM cortex M4 embedded device.
2. Soldering and wire wrapping comes handy.
3. Configuring DPTR and assigning it to address is something I am comfortable with after doing this lab work.
4. Understood how debug latches work, MAX232, Data terminal equipment to data communication equipment.
5. Obtained an idea on the scenarios where a C51 processor is implemented and an AT89C52 processor is implemented.
6. Understood how to program with flip.
7. Understood the concept of timing analysis, setup and hold time calculations.

**RESULT:** All requirements designed, coded, validated for this lab and executed successfully.

**SIGN OFF SHEETS**







