Electrical and Computer Engineering

EE3613 Processor Systems Laboratory

LAB 10

Advanced AVR Programming in Assembly

Fall 2019

OBJECTIVES:

- To learn how to use **EEPROM** in AVR Microcontrollers
- To learn how to operate **Checksum**
- To learn how to read external data and write to EEPROM

REFERENCES:

Mazidi and Naimi, “The AVR Microcontroller and Embedded Systems,” Chapter 6

MATERIALS:

Atmel Studio 7, Atmel Xplained PRO board

BACKGROUND INFORMATION:

1. EEPROM in AVR

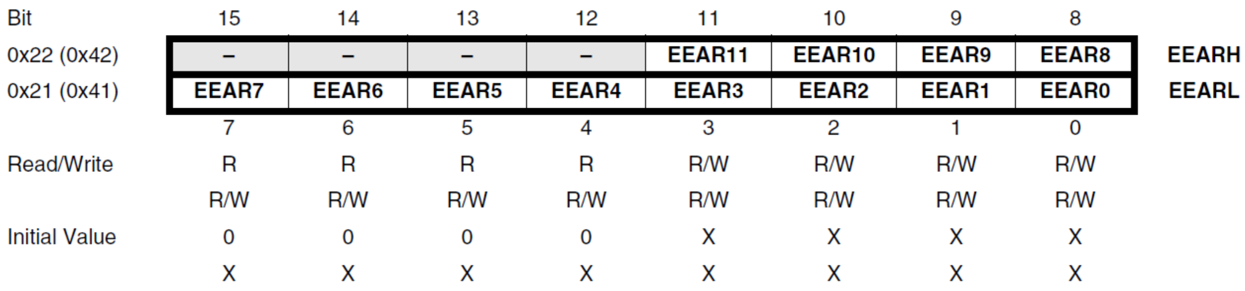
EEPROM: *Electrically Erasable Programmable Read-Only Memory.*

AVR Microcontroller has several different sections of memory. We have used program and data memory which includes General Purpose Registers, I/O Registers, Flash, SRAM, etc. All these memory locations usually wipe out the data during power loss of the system. EEPROM is a special part of AVR memory locations which can retain the saved data even if there is any power loss. Usually the EEPROM has an endurance of at least 100,000 write/erase cycles. To use this memory, we use three registers: 1. EEPROM Address Register (EEARH and EEARL), 2.

EEPROM Control Register (EECR), and 3. EEPROM Data Register (EEDR). The registers' structures are shown in Figure 1, 2, and 3 from the ATmega324PB datasheet.

EEPROM Address Registers, EEAR:

We use EEAR 9:0 bits to address the location because the ATmega324PB model has 1KB space in the memory and the data bytes are addressed between 0 and 1023. The initial value of EEAR is undefined. A proper value must be written before the EEPROM may be accessed. Bits 15:12 are unused bits in the ATmega324PB and will always read as zero.

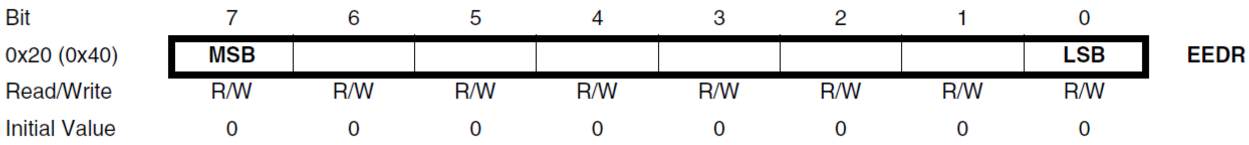


Bit	15	14	13	12	11	10	9	8	
0x22 (0x42)	–	–	–	–	EEAR11	EEAR10	EEAR9	EEAR8	EEARH
0x21 (0x41)	EEAR7	EEAR6	EEAR5	EEAR4	EEAR3	EEAR2	EEAR1	EEAR0	EEARL
	7	6	5	4	3	2	1	0	
Read/Write	R	R	R	R	R/W	R/W	R/W	R/W	
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	X	X	X	X	
	X	X	X	X	X	X	X	X	

Figure 1: EEPROM Address Registers (9:0)

EEPROM Data Registers, EEDR:

For the EEPROM write operation, the EEDR Register will contain the data to be written to the EEPROM in the address defined by the EEAR Register. For the EEPROM read operation, the EEDR will contain the data read out from the EEPROM at the address defined by EEAR.



Bit	7	6	5	4	3	2	1	0	
0x20 (0x40)	MSB							LSB	EEDR
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

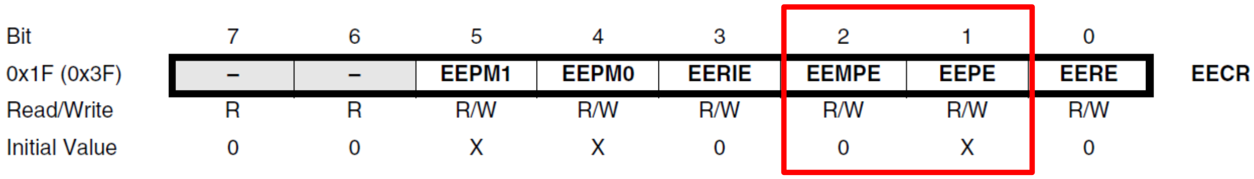
Figure 2: EERPOM Data Register

EEPROM Control Registers, EECR:

Bits 7:6 are unused bits in the ATmega324PB and will always read as zero.

Bits 5:4 - EEPM1 and EEPM0 are **E**EPROM **P**rogramming **M**ode bits. The Programming mode bit setting defines which programming action will be triggered when writing **EEPE** (**P**rogramming **E**nable bit). While EEPE is set, any write to EEPM0 or EEPM1 will be ignored.

During reset, the EEPM1 and EEPM0 bits will be reset to ‘00’ unless the EEPROM is busy programming.



Bit	7	6	5	4	3	2	1	0	
0x1F (0x3F)	–	–	EEPM1	EEPM0	EERIE	EEMPE	EEPE	EERE	EECR
Read/Write	R	R	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	X	X	0	0	X	0	

Figure 3: EEPROM Control Register

Bit 3 - EERIE: EEPROM Ready Interrupt Enable bit. Writing EERIE to ‘1’ enables the EEPROM Ready Interrupt if the I (Bit 7) in Status Register (Global Interrupt Enable) is set. Writing EERIE to zero disables the interrupt. The EEPROM Ready interrupt generates a constant interrupt when EEPE is cleared.

Bit 2 – EEMPE: EEPROM Master Programming Enable bit. The EEMPE bit determines whether setting EEPE to ‘1’ causes the EEPROM to be written. When EEMPE is set (‘1’), setting EEPE within four clock cycles will write data to the EEPROM at the selected address If EEMPE is ‘0’, setting EEPE will have no effect.

Bit 1 – EEPE: EEPROM Programming Enable bit. EEPE is the write strobe to the EEPROM. When address and data are correctly set up, the EEPE bit must be written to ‘1’ to write the value into the EEPROM. The EEMPE bit must be written to ‘1’ before a logical ‘1’ is written to EEPE, otherwise no EEPROM write takes place. The following procedure should be followed when writing the EEPROM (the order of steps 2 and 3 is not essential):

1. Wait until EEPE becomes zero.
2. Write new EEPROM address to EEAR (when necessary).
3. Write new EEPROM data to EEDR (when necessary).
4. Set the EEMPE bit to one while writing a zero to EEPE in EECR.
5. Within four clock cycles after setting EEMPE, set EEPE to one.

Bit 0 – EERE: EEPROM Read Enable bit. EERE is the read strobe to the EEPROM. When the correct address is set up in the EEAR Register, the EERE bit must be written to a logic one to trigger the EEPROM read.

EEPROM Write Error Issue: VCC is likely to rise or fall slowly on power-up/down. This causes the device for some period to run at a voltage lower than specified as minimum for the clock frequency used. To prevent unintentional EEPROM writes, a specific write procedure must be followed. When the EEPROM is read, the CPU is halted for four clock cycles before the next instruction is executed. When the EEPROM is written, the CPU is halted for two clock cycles before the next instruction is executed.

Example: Write an AVR program to store ‘G’ into location 0x005F of EEPROM and then read it back and send it to PORTA .

```
.INCLUDE <M324PBDEF.INC>
LDI R16,0xFF
OUT DDRA,R16
WAIT:
;wait for last write to finish
SBIC EECR,EEPE          ;check EEPE to see if last write is finished
RJMP WAIT               ;wait more
LDI R18,0               ;load high byte of address to R18
LDI R17,0x5F            ;load low byte of address to R17
OUT EEARH, R18          ;load high byte of address to EEARH
OUT EEARL, R17          ;load low byte of address to EEARL
LDI R16,'G'             ;load 'G' to R16
OUT EEDR,R16            ;load R16 to EEPROM Data Register
SBI EECR,EEMPE          ;set Master Write Enable to one
SBI EECR,EEPE           ;set Write Enable to one
SBI EECR,EERE           ;set Read Enable to one
IN R15,EEDR             ;load EEPROM Data Register to R16
OUT PORTA,R15           ;out R16 to PORTB
here: jmp here
```

2. CHECKSUM OPERATION

During the DATA transfer or storage, errors can happen due to the condition of wiring or unusual event during system startup. Checksum is one of the ways of checking the data integrity. A good checksum algorithm will usually output a significantly different value, even for small changes made to the input.

We can create a checksum byte for a set of data easily using the following steps:

1. Add all data together
2. Find 2’s complement of the value after carry drop
3. Define the value as a checksum byte

To check whether the data it is corrupted after being written to EEPROM or not, we can add the checksum byte to the sum of the data transferred to the new memory location and drop the carry. If the result is “00” after carry drop, we can determine that, the data is transferred correctly. If not, the data has been corrupted during the write or read process.

Example:

Data to be written: $45, $32 and $24

Checksum byte: ($45+$32+$24) 2's complement after drop the carry bit = $65

To verify the transferred data:

Correct case [($45+$32+$24) + $65 = 00 after the carry bit drop]

Incorrect case [($45+$33+$24) + $65 ≠ 00 after the carry bit drop]

ACTIVITY I

Write a program to write three sets of data in the flash memory, then transfer the data to the data memory and EEPROM at the following location.

- Flash Memory Location starts at 0x200
- DATA Memory Location starts at 0x200
- EEPROM Memory Location starts at 0x200

Follow the steps shown below:

[Step 1] Write three data sets shown in the flash memory.

Data Set 1: [$12, $36, $83, $A5, $5D, $29]

Data Set 2: ['T','E','M','P','L','E']

Data Set 3: ["Engineering"]

[Step 2] Generate three checksum bytes of the original datasets and load them to the registers, R20, R21, and R22.

[Step 3] Transfer the Data Set 1 to the data memory location at 0x200.

[Step 4] Transfer the Data Set 1 from the data memory location to the EEPROM memory location at 0x200.

[Step 5] Show the checksum byte of Data Set 1 through PORT A. Take pictures of the LED outputs and use them as your result.

[Step 6] Take the screenshots of the flash memory, data memory, and EEPROM memory locations at 0x200 and use them for your result.

[Step 7] Repeat Steps 3 – 6 for Data Set 2 and 3. The operation must run infinitely.

ACTIVITY 2

Write a program to transfer data from the flash memory to the EEPROM memory location. In this activity, we write correct and incorrect data sets in the flash memory and transfer them to the EEPROM. Also, we generate the checksum bytes of the correct data set and load them to some registers to check the data integrity.

Follow the steps shown below:
[Step 1] Generate and load the checksum bytes of the data set 1, 2, and 3 to R20, R21, and R22.

Data set values are followings:

Data Set 1: [$12, $36, $83, $A5, $5D, $29]
Data Set 2: [‘T’,’E’,’M’,’P’,’L’,’E’]
Data Set 3: [“Engineering”]

[Step 2] Write the correct and incorrect data sets shown in the flash memory.

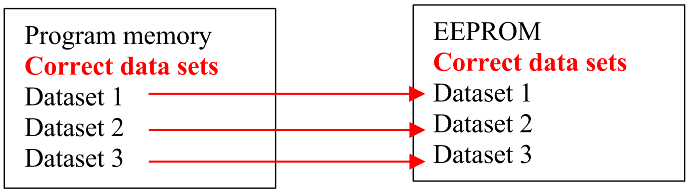
Correct Data Sets in the flash memory at 0x200
Data Set 1: [$12, $36, $83, $A5, $5D, $29]
Data Set 2: [‘T’,’E’,’M’,’P’,’L’,’E’]
Data Set 3: [“Engineering”]

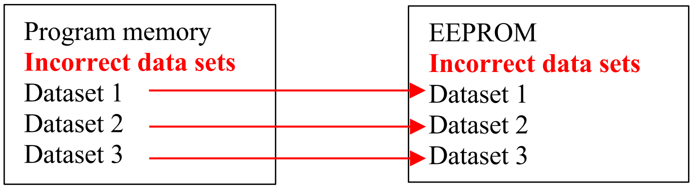
Incorrect Data Sets in the flash memory at 0x300
Data Set 1: [$12, $36, **$33**, $A5, $5D, $21]
Data Set 2: [‘T’,’E’,’M’,’**B**’,’L’,’E’]
Data Set 3: [“**e**ngineering”]

[Step 3] Read the PORTB switch bit 0

[Step 4] Based on the switch input, select the operation shown in Table 1

TABLE 1. Logic Operation based on the switch inputs

Switch bit 0	Operation
0	• Transfer the correct data set to EEPROM location at 0x100 for Data set 1, 0x150 for Dataset 2, and 0x200 for Dataset 3 • check the data integrity test  Program memory Correct data sets Dataset 1 Dataset 2 Dataset 3 EEPROM Correct data sets Dataset 1 Dataset 2 Dataset 3

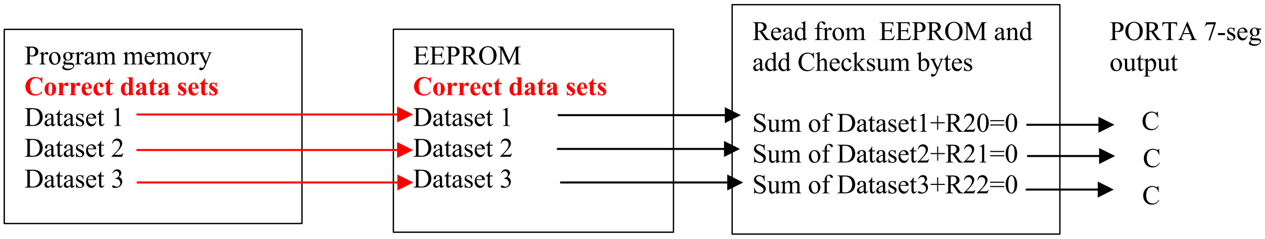
1	• Transfer the incorrect data set to EEPROM location at 0x100 for Data set 1, 0x150 for Dataset 2, and 0x200 for Dataset 3 • check the data integrity test  Program memory Incorrect data sets Dataset 1 Dataset 2 Dataset 3 EEPROM Incorrect data sets Dataset 1 Dataset 2 Dataset 3

[Step 5] Read each data set from EEPROM and check the data integrity of the data sets in the EEPROM location using the checksum bytes generated in Step 1

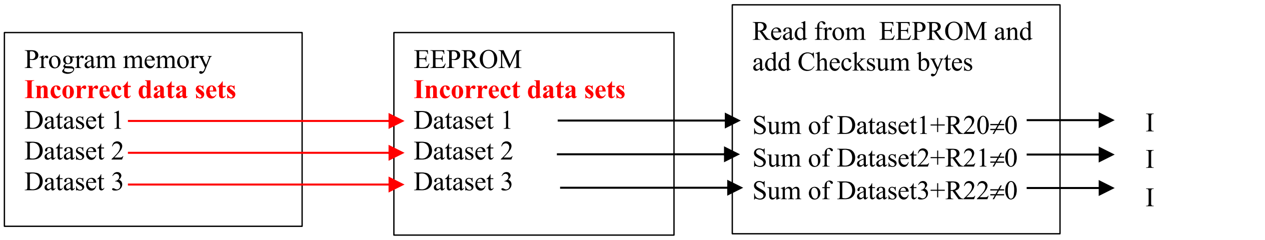
[Step 6] Display 'C' if the data is correctly transferred to EEPROM using the 7-segment display, or display 'I' if the data is incorrectly transferred to EEPROM using the 7-segment display (use 1-second time delay for each display)

<TOTAL PROCESS OF ACTIVITY 2>

If the correct data sets transferred, SW0=0:



If the incorrect data sets transferred, SW0=1:



[Step 7] Keep monitor the switch input and switch the operation after the data transfer is completed

[Step 8] Infinite loop operation is required

REPORT

Lab 10 requires the formal full-length report. The report must include the following sections and details:

1. Cover page (5pts): your information must be shown
2. Introduction (10pts)

- Objectives
- Any related knowledge – what is EEPROM and checksum operation

3. Procedure (30pts)
    - Flowchart: activity 1 and activity 2
    - Method: description of the given tasks based on your understanding and your approaches to make the solutions / Explanation of the required procedure and the expected result after you follow the procedure
4. Results (30pts)
    - Tables, screenshots, and pictures: you must explain the result with your own words
        - Activity 1 – Memory contents – flash memory, data memory, and EEPROM, operation result (Pictures or screenshots of LED outputs)
        - Activity 2 – Memory contents – flash memory, EEPROM, registers
          7-Segment Display Result and Time Delay, switch input
    - The physical result –video links for each activity
5. Discussion (20pts)
    - Verification of the answers and discuss the results by comparing them with the expected result.
6. Conclusion (10pts)
    - Summary of key concepts of this lab
7. Appendix (10pts)
    - Code for Activity 1
    - Code for Activity 2

Your codes must be saved in a .txt file, too. Then, submit the .txt file to the LAB 10 Code submission part separately. The lab report must be submitted by November 16, 2019, 11:59 pm (section 1) / November 19, 2019, 11:59 pm (section 2).

IMPORTANT: DO NOT COPY FROM LAB MANUAL OR ANY OTHER SOURCES >>

ECE3613 Processor System Laboratory Rubric
Lab #: 10
Section: 001 / 002
Name: ________________________

Report Section	Activity	Contents	Full Points	Earned Points	Comment
Cover Page			5		Lab report Cover page
Introduction			10		Objective and background information (EEPROM 5pts and CHECKSUM 5pts)
Procedure	1	Flowchart	5		
		Method	15		• Outline of the steps do complete the activities(5 pts) • Description of the code lines that are related to the main operation (10 pts)
	2	Flowchart	5		
		Method	15		• Outline of the steps do complete the activities(5 pts) • Description of the code lines that are related to the main operation (10 pts)
	Subtotal		**55**		
Result	1	Result	20		• Screenshot of the contents in the memory locations (10pts) • Pictures of LEDs outputs with description (10pts)
		Video	5		Video link
	2	Result	20		• Screenshot of the contents in the memory locations (10pts) • Pictures of 7-segment display with description (10pts)
		Video	5		Video link
	Subtotal		**50**		
Discussion	1	Evaluation	10		Result Verification and Analysis
	2	Evaluation	10		Result Verification and Analysis
Conclusion			5		Summary of the lab
Appendix	1	Code	5		Full comments
	2	Code	5		Full comments
	Subtotal		**35**		
Total			**140**		