Department of Computer Engineering
Faculty of Engineering, University of Peradeniya

CO321 Embedded Systems - 2018

Lab 8 - EEPROM

AVR EEPROM programming in C language

Introduction

EEPROM Data Memory

EEPROM (Electrically Erasable Programmable Read-Only Memory) is a type of non-volatile memory that allows individual bytes to be erased and reprogrammed. EEPROMs can be programmed and erased in-circuit, by applying special programming signals.

The EEPROM in ATMEGA328P has a capacity of 1 KB. It can endure at least 100,000 write/erase cycles. The access between the EEPROM and the CPU is described in the following, specifying the EEPROM Address Registers, the EEPROM Data Register, and the EEPROM Control Register.

EEPROM Read/Write Access

The EEPROM Access Registers are accessible in the I/O space. If the user code contains instructions that write the EEPROM, some precautions must be taken. In heavily filtered power supplies, VCC is likely to rise or fall slowly on power up/down. This causes the device for some period of time to run at a voltage lower than specified as a minimum for the clock frequency used. In order 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.

Register Description

EEARH and EEARL – The EEPROM Address Register

Bit	15	14	13	12	11	10	9	8	
0x22 (0x42)	–	–	–	–	–	–	EEAR9(1)	EEAR8(1)	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	R	R	R/W	
	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	X	
	X	X	X	X	X	X	X	X	

- **Bits 9:0 – EEAR[9:0]: EEPROM Address**

The EEPROM Address Registers – EEARH and EEARL specify the EEPROM address in the 256/512/512/1k Bytes EEPROM space. The EEPROM data bytes are addressed linearly between 0 and 255/511/511/1023. The initial value of EEAR is undefined. A proper value must be written before the EEPROM may be accessed.

EEDR – The EEPROM Data Register

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

EECR – The EEPROM Control Register

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	

.

Bits 5, 4 – EEPM1 and EEPM0: EEPROM Programming Mode Bits

The EEPROM Programming mode bit setting defines which programming action that will be triggered when writing EEPE. It is possible to program data in one atomic operation (erase the old value and program the new value) or to split the Erase and Write operations in two different operations. While EEPE is set, any write to EEPMn will be ignored. During reset, the EEPMn bits will be reset to 0b00 unless the EEPROM is busy programming.

EEPROM Mode Bits

EEPM1	EEPM0	Programming Time	Operation
0	0	3.4ms	Erase and Write in one operation (Atomic Operation)
0	1	1.8ms	Erase Only
1	0	1.8ms	Write Only
1	1	–	Reserved for future use

Bit 3 – EERIE: EEPROM Ready Interrupt Enable

Writing EERIE to one enables the EEPROM Ready Interrupt if the I bit in SREG is set. Writing EERIE to zero disables the interrupt. The EEPROM Ready interrupt generates a constant interrupt when EEPE is cleared. The interrupt will not be generated during EEPROM write or SPM.

Bit 2 – EEMPE: EEPROM Master Write Enable

The EEMPE bit determines whether setting EEPE to one causes the EEPROM to be written. When EEMPE is set, setting EEPE within four clock cycles will write data to the EEPROM at the selected address If EEMPE is zero, setting EEPE will have no effect. When EEMPE has been written to one by software, hardware clears the bit to zero after four clock cycles.

Bit 1 – EEPE: EEPROM Write Enable

The EEPROM Write Enable Signal EEPE is the write strobe to the EEPROM. When address and data are correctly set up, the EEPE bit must be written to one to write the value into the EEPROM. The EEMPE bit must be written to one before a logical one is written to EEPE, otherwise no EEPROM write takes place.

Bit 0 – EERE: EEPROM Read Enable

The EEPROM Read Enable Signal 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. The EEPROM read access takes one instruction, and the requested data is available immediately. When the EEPROM is read, the CPU is halted for four cycles before the next instruction is executed. The user should poll the EEPE bit before starting the read operation. If a write operation is in progress, it is neither possible to read the EEPROM, nor to change the EEAR Register.

When the write access time has elapsed, the EEPE bit is cleared by hardware. The user software can poll this bit and wait for a zero before writing the next byte. When EEPE has been set, the CPU is halted for two cycles before the next instruction is executed.

Accessing EEPROM using AVR C

Writing to EEPROM

The following procedure should be followed when writing the EEPROM (the order of steps 3 and 4 is not essential):

1. Wait until a previous EEPROM write is finished (till EEPE becomes zero)
2. Write new EEPROM address
3. Write new EEPROM data
4. Write a logical one to the EEMPE bit while writing a zero to EEPE in EECR
5. Within four clock cycles after setting EEMPE, write a logical one to EEPE

Caution: An interrupt between step 5 and step 6 will make the write cycle fail, since the EEPROM Master Write Enable will time-out. If an interrupt routine accessing the EEPROM is interrupting another EEPROM access, the EEAR or EEDR Register will be modified, causing the interrupted EEPROM access to fail. It is recommended to have the Global Interrupt Flag cleared during all the steps to avoid these problems.

Reading from EEPROM

1. Wait until a previous EEPROM write is finished (till EEPE becomes zero)
2. Write new EEPROM address
3. Start EEPROM read by writing to EERE
4. Read the EEPROM data from data register

Exercises

1. Write a function called EEPROMwrite() that takes the EEPROM address and the data as arguments and write to the EEPROM. Write a function called EEPROMread() that takes the EEPROM address as the arguments and returns the data in that memory location. (You don't need to show this part to the instructors but you need to use these functions for the next questions).

2. Write a program that takes an ASCII sentence (ending with the carriage return \r) through the serial port and then writes to the EEPROM. Then write another program that reads and prints all the data in the first 1024 addresses in the EEPROM via the serial port.

3. In this exercise you have to develop an encryption system using the Caesar cipher.
   For this you have to use the LCD, keypad and the AVR MCU provided.
   When you run the program on the AVR, it should ask whether you want to encrypt or change the secret key. You should be able to make the choice using the provided keypad.

Once the key is set, then it should ask for the plain text (for simplicity, we'll use 10 characters). Also, the key should be stored in EEPROM. The plain text should be entered using the keypad. It should be shown on the LCD in real time. Once the plain text is entered, it should be encrypted, and displayed on the LCD.

You can fine tune the system in any manner you like. (e.g.: Like what happens after an encryption process).
Also no communication should not be done with the PC other than uploading the program

Sample code

```
void EEPROMwrite(unsigned int address, char data){

    //wait for completion of previous write
    while(EECR & (1<<EEPE));

    //set up address and data regs
    EEAR = address;
    EEDR = data;

    //write logical 1 to EEMPE
    EECR |= (1<<EEMPE);

    //start eeprom write
    EECR |= (1<<EEPE);

}

char EEPROMread(unsigned int address){

    //wait for completion of writes
    while(EECR & (1<<EEPE));

    //set up address
    EEAR = address;

    //start eeprom read
    EECR |= (1<<EERE);

    //return data
    return EEDR;
}
```