EEPROM IN AVR

hroodaki@kntu.ac.ir

ACCESSING EEPROM IN AVR

- Every member of the AVR microcontrollers has some amount of on-chip EEPROM.
- The data in SRAM will be lost if the power is disconnected. However, we need a place to save our data to protect them against power failure.
- EEPROM memory can save stored data even when the power is cut off.

EEPROM registers

- There are three I/O registers that are directly related to EEPROM.
  - EECR (EEPROM Control Register)
  - EEDR (EEPROM Data Register)
  - EEARH-EEARL (EEPROM Address Register High-Low).

EEPROM Data Register (EEDR)

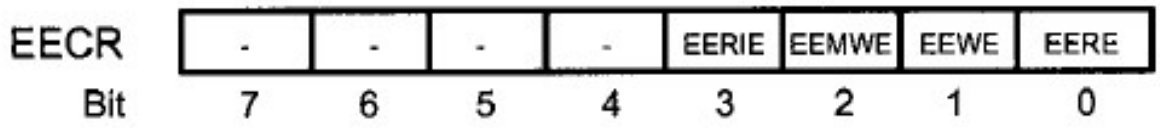
- To write data to EEPROM, you have to write it to the EEDR register and then transfer it to EEPROM.
- Also, if you want to read from EEPROM you have to read from EEDR.
- EEDR is a bridge between EEPROM and CPU.

EEPROM Address Register (EEARH and EEARL)

- The EEARH:EEARL registers together make a 16-bit register to address each location in EEPROM memory space.
- When you want to read from or write to EEPROM, you should load the EEPROM location address in EEARs.
  - Only 10 bits of the EEAR registers are used in ATmega32. BecauseATmega32 has 1024-byte EEPROM locations, we need 10 bits to address each location in EEPROM space.
  - In ATmega 16, 9 bits of the EEAR registers are used becauseATmegal6 has 512 bytes of EEPROM, and to address 512 bytes we need a 9-bit address.

EEPROM Control Register (EECR)

- The EECR register is used to select the kind of operation to perform on.
  - Read
  - write



EEPROM Control Register (EECR)

- EEPROM Write Enable (EEWE) and EEPROM Master Write Enable (EEMWE)
  - When EEMWE is set, setting EEWE within four clock cycles will start a write operation.
  - If EEMWE is zero, setting EEWE to one will have no effect.

EEPROM Control Register (EECR)

- Notice that you cannot start read or write operations before the last write operation is finished.
- You can check for this by polling the EEWE bit.
- If EEWE is zero it means that EEPROM is ready to start a new read or write operation

Programming the AVR to write on EEPROM

- Wait until EEWE becomes zero.
- Write new EEPROM address to EEAR.
- Write new EEPROM data to EEDR.
- Set the EEMWE bit to one (in EECR register).
- Within four clock cycles after setting EEMWE, set EEWE to one.

Example 6-28

Write an AVR program to store 'G' into location 0x005F of EEPROM .

Solution:

```
.INCLUDE "M16DEF.INC"

WAIT:              ;wait for last write to finish
SBIC  EECR,EEWE    ;check EEWE 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,EEMWE   ;set Master Write Enable to one
SBI   EECR,EEWE    ;set Write Enable to one
```

Run and simulate the code on AVR Studio to see how the content of the EEPROM changes after the last line of code. Enter four NOP instructions before the last line, change the 'G' to 'H', and run the code again. Explain why the code doesn't store 'H' at location 0x005F of EEPROM.

EEPROM Control Register (EECR)

- EEPROM Read Enable (EERE)
  - Setting this bit to one will cause a read operation if EEWE is zero.
    - When a read operation starts, one byte of EEPROM will be read into the EEPROM Data Register (EEDR).
    - The EEAR register specifies the address of the desired byte

Programming the AVR to read from EEPROM

- Wait until EEWE becomes zero.
- Write new EEPROM address to EEAR.
- Set the EERE bit to one.
- Read EEPROM data from EEDR.

Example 6-29

Write an AVR program to read the content of location 0x005F of EEPROM into PORTB.

Solution:

```
.INCLUDE "M16DEF.INC"
      LDI   R16,0xFF
      OUT   DDRB,R16
WAIT:                    ;wait for last write to finish
      SBIC  EECR,EEWE    ;check EEWE 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
      SBI   EECR,EERE    ;set Read Enable to one
      IN    R16,EEDR     ;load EEPROM Data Register to R16
      OUT   PORTB,R16    ;out R16 to PORTB
```

Initializing EEPROM

- If we write .ESEG before a definition, the variable will be located in the EEPROM, whereas .CSEG before a definition causes the variable to be allocated in the code (program) memory.
  - By default the variables are located in the program memory.
- We can initialize the EEPROM using the .DB directive.
- For example, the following code allocates locations $10 and $11 of EEPROM for DATAI and DATA2, and initializes them with $95 and $19, respectively.

```
.ESEG
.ORG $10
        DATAl:.DB $95
        DATA2: .DB $19
```