AVR EEPROM Handling – Tutorial #16

T.K. HAREENDRAN

AVR tutorial

Share this:

Share More

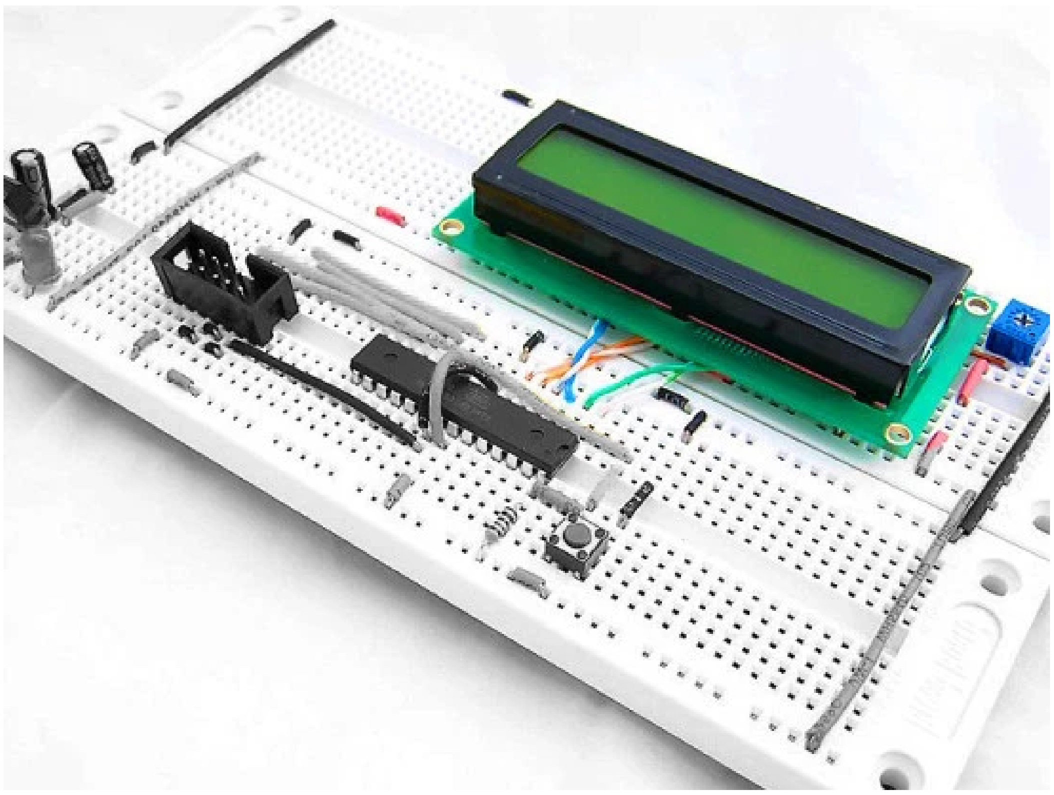
Almost all AVR microcontrollers have built-in Electrically Eraseable Programmable Read Only Memory (EEPROM).The advantage of EEPROM is that even when the microcontroller is turned off, data stored in the EEPROM will remain. This is essential in data-logging projects, and storing user-decided settings. In short, this internal EEPROM for semipermanent data storage, and like the flash memory, EEPROM can maintain its contents when electrical power is removed. However, note that in most variants of the AVR architecture, the internal EEPROM memory is not mapped into the microcontroller's addressable memory space. It can only be accessed the same way an external peripheral device is, using special pointer registers and read/write instructions.

Another important fact to note is that the AVR's internal EEPROM memory has a limited lifespan of 100,000 writes per EEPROM. Keep this in mind and always try to keep writes to a minimum, so that you only write the least amount of information required for your application each time you update the EEPROM.

The Atmega8 chip contain three types of memory. The program resides in the FLASH memory, which is programmable read-only memory (ROM).The content can be changed only by a programmer or a bootloader.

The next memory is the SRAM, which is a volatile memory holds data only when electric power is available. Note that in SRAM, the contents are erased when the chip is power cycled.

The third one is electrically erasable programmable read only memory-EEPROM. Contents written here are preserved across power cycles, and the user can manipulate this memory easily as a part of the program code.



ATMEGA8 & EEPROM

The AVR's internal EEPROM is accessed via special registers inside the AVR, which control the address to be written to (EEPROM uses byte addressing), the data to be written (or the data which has been read) as well as the flags to instruct the EEPROM controller to perform the requested read (R) or write (W) operation.

To use EEPROM in AVR studio with WINAVR, eeprom.h file can be used.For this, first include the avr/eeprom.h file as there are functions at eeprom.h to read and write a byte or a word which wroks for any storage variable like a char, an int, or a structure.

- eeprom_read_block();
- eeprom_write_block();

The variables that need to be written to EEPROM, or read from EEPROM need to be declare their storage location as EEMEM. This helps AVR Studio/WINAVR compiler to organize the memory. Defined by the same avr/eeprom.h header that gives us our access functions for the EEPROM, the EEMEM attribute instructs the compiler to assign your variable into the EEPROM address space, instead of the SRAM address space like a normal variable.

EEPROM STEPS – INDICATORS

- Include:

```
#include
```

- Write:

```
eeprom_write_byte ((uint8_t*) 23, 64); //  write the byte 64 to location 23 of the EEPROM
```

- Read:

```
uint8_t byteRead = eeprom_read_byte((uint8_t*)23); // read the byte in location 23
```

Before write, look in the datasheet to see how many bytes of EEPROM you have. For example, if it is 2kb of EEPROM, location can be anything up to 2000.

Now you learned the basics of AVR EEPROM. There is a lot more to EEPROM than what this part covers, I just skimmed the surface!

Update & Write functions?

Traditionally, there were only two types of EEPROM functions per data type; a write function, and a read function. In the case of the EEPROM write functions, these functions simply wrote out the requested data to the EEPROM without any checking performed, resulted in a reduced EEPROM lifetime if the data to be written already matches the current contents of the EEPROM cell. To reduce the wear on the AVR's limited lifetime EEPROM, the new update functions were added which only perform an EEPROM write if the data differs from the current cell contents. However, the old write functions are still kept around for compatibility with older applications.

Remember, EPROM is an older technology to implement rewritable non-volatile memory. It's normally used to store settings and other parameters between resets (power cycles). The Atmega8 have 512 bytes of EEPROM.

Here is an application note contains routines for access of the EEPROM memory in the AVR Microcontroller. Two types of Read/Write access has been implemented:

- **Random Read/Write:** The user must set up both data and address before calling the Read or Write routine.
- **Sequential Read/Write:** The user needs only to set up the data to be Read/Written. The current EEPROM address is automatically incremented prior toaccess. The address has to be set prior to writing the first byte in a sequense. The application note contains four routines

which are described in detail and routines for accessing the EEPROM in all AVR devices. You can view/download the PDF guide from here: www.atmel.in/Images/doc0932.pdf