

## Re: 8051: writing to memory in the program

---

*Source:* <http://coding.derkeiler.com/Archive/General/comp.arch.embedded/2007-01/msg00558.html>

---

- *From:* Roberto Waltman <[usenet@xxxxxxxxxxxxxx](mailto:usenet@xxxxxxxxxxxxxx)>
  - *Date:* Thu, 11 Jan 2007 11:17:19 -0500
- 

"wanwan" wrote:

I'm learning the 8051 type microcontroller now ...

I'd like my program to be able to write some data to memory while running, so these data can be remembered when power down and can be retrieved at power up.

There are several ways to do it, the following are generic answers that may or not be applicable to your environment.

You can save the data to:

(A) Non-volatile memory. (typically FLASH or EEPROM, although other technologies are available.)

(B) Volatile memory (RAM), with back-up power to keep its contents intact while the rest of the system is unpowered.

And that memory can be:

(1) On-chip.

(2) Off-chip, on an external device.

(B-1) Volatile/on-chip. The simplest way – Some microprocessors have the capability to go into a low-power mode or shut down completely, while still providing power to the internal RAM.

If following this approach, you need to make sure the RAM area you allocate for this purpose can not be overwritten or corrupted by the normal startup/ shut down processes.

(A-1) Non-volatile/on-chip Not so simple anymore. In most micros with internal FLASH memory you can program it under software control, but you can not access the contents of FLASH memory while it is being programmed.

## Re: 8051: writing to memory in the program

That means you can not run the code that programs the FLASH from the FLASH itself. (Or at least, not from the sector that is being programmed)

If all you have is one big sector of FLASH memory, the programming code needs to be copied to internal RAM and run from there.

Interrupts need to be disabled while programming if there are ISRs using vectors and/or code in the FLASH memory

This problem disappears, of course, if you have both FLASH and EEPROM and use the EEPROM for data storage.

Two additional caveats:

(1) You can not erase/reprogram individual bytes in FLASH memory, only whole sectors. To change only a few bytes you will have to save the contents of the whole sector in RAM, or keep a spare FLASH sector copying/modifying into it. (EEPROM also allows erasure of individual bytes.)

(2) FLASH and EEPROM devices have a limit on how many times they can be erased. This limit tends to be lower in on-chip devices since the manufacturing process is not optimized for FLASH or EEPROM due to presence of other circuitry on the chip (CPU, RAM, etc.)

It is very easy to exceed this limit with careless programming.

(B-2) Volatile/off-Chip – Simple as B-1. May be simpler, since you can buy memories with a built-in battery, or chips with both volatile and non volatile areas that can be copied one on top of the other as a single operation.

(A-2) Non-volatile/off-chip – With FLASH, no problems with "disappearing" sectors (Well, you still can not read from FLASH while programming, but the programming code could run from internal FLASH)

With EEPROM: A common solution for small amounts of data is to use a serial interface (I2C, SPI) to communicate with an EEPROM chip. They are easy to use, both hardware and software-wise.

The best solution for you will depend on what hardware is available in your system, how much data you want to store, etc.

Roberto Waltman

.