F-RAM™ as One-Chip Solution for Code and Data Memory Applications

Author: Harsha Medu

AN101 reviews the memory requirements for both code and data, and the design considerations for combining them in a single F-RAM device.

1 Overview

Applications for memory technologies are generally divided between executable code and data tasks. Executable code requires that the memory is nonvolatile and retains the code under all conditions. Data tasks require that the memory access for read and write operations are fast, simple, and unlimited. Memory for data tasks can be either volatile or nonvolatile depending on the application. The traditional choice of memory for executable code has been ROM-based technologies, while for data tasks, RAM-based technologies. F-RAM, one of the nonvolatile RAM products from Cypress, provides a unique advantage of combining both executable code and data in a single memory.

This application note compares code and data memory requirements when each is considered separately and when both are combined in a single-system memory solution. It also proposes a design solution for protecting the code portion of the F-RAM from accidental overwrite in a combined code-and-data memory application.

2 Memory Requirements for Code Storage

- Nonvolatile
- Appropriate density
- Read-access time
- Ability to prevent inadvertent writes
- Field-programmable in parts or sections
- Programmable with concurrent read access

The basic requirement of code storage is that the memory should be nonvolatile and retain its state under all conditions. Code memory is read-only and does not involve any run-time updates.

The memory space required to hold the code is set by the application. Often, 20%-30% of additional memory space is included for possible future system modifications.

Code execution time is determined by the memory's read-access time; the faster the access time, the less impact on the controller overhead.

Code memory is read-only. Any inadvertent write should be prevented since it can result in application malfunction. There are very few occasions when code memory is written. One such case is the periodic updates to bring in new features or fix firmware bugs. The ability to reprogram the code memory in the field is a key advantage and capability. Reprogramming often only impacts parts or sections of the code, and that not doing a total code reinstall is preferred. During reprogram, write-access timing is a factor in determining how soon the part can be reprogrammed.

Code memory can contain the erase/program code which is responsible for planned field programmability. During field programming, it is important to have concurrent write and read access to memory bytes or sections to avoid unnecessary complications of copying the erase/program code to some other memory (RAM) and executing from there.

NAND flash is the common code memory solution, but in some applications where the code memory is quite small and applications need symmetrical write and read access times (data logging, energy harvesting, or long-life battery operation) F-RAM is the more appropriate choice.

3 Memory Requirements for Data Storage

- Fast write access
- Large number of writes allowed
- Simple write protocol (no protocol is preferable)
- Byte-addressable writes
- Nonvolatile (some applications)
- Ability to serve volatile and nonvolatile needs

In many respects, the requirements for data are the opposite of those of code. Data storage is a far more diverse task, requiring flexibility and easy write-access. Data storage applications typically need fast read/write access and it may be either nonvolatile or volatile depending on the type of data or application.

Due to different application needs, data memory tends to have a large number of writes. Some are periodic writes while some are based on events. It is desirable to have fast write-access to store the data quickly. Therefore, data memory should have unlimited number of writes and fast access. It should also have easy and simple access due to the large number of transactions involved.

Byte-writable memory is another important aspect of data memory. Because the data can include a single variable getting updated multiple times, byte-writable access is preferred without disturbing the other content.

Some applications require the data to be nonvolatile to save the settings over power cycles. It gives the flexibility to the user to modify the settings and yet save it without any complexity.

In volatile applications, RAM technologies are the typical data storage devices. In nonvolatile applications, data storage is a challenge for the mainstream memory technologies. ROM-based technologies, such as flash, make poor data memories as they are not flexible for writing and are more suited for static configurations. EEPROM is another choice for nonvolatile data memory. However, it offers very slow write times (milliseconds compared to nanoseconds in F-RAM) and permits fewer write-cycles before wearing out.

F-RAM is optimized for nonvolatile data storage. F-RAM writes occur at the same speed as the reads. It is nonvolatile, and offers a very large ($>10^{14}$) number of access cycles. No special algorithms or protocols are needed to write the memory, and it is byte-addressable. Therefore, F-RAM provides the most flexible solution for data storage.

4 Single-Chip Code and Data Requirements

- Nonvolatile with fast write and read access
- Large number of writes
- Ability to prevent inadvertent writes to memory blocks
- Flexibility to change the code and data memory size

Combining code and data in a single device can mean getting one memory to provide almost mutually exclusive services for both code and data.

Code memory requires nonvolatile storage and the ability to support occasional writes. Upgrading the code will never call for a large number of write operations (cycles). The write time is usually not important. In some respects, the more difficult it is to write a code memory the better, because an accidental write could be catastrophic. However, the read-access time is important to run the code faster.

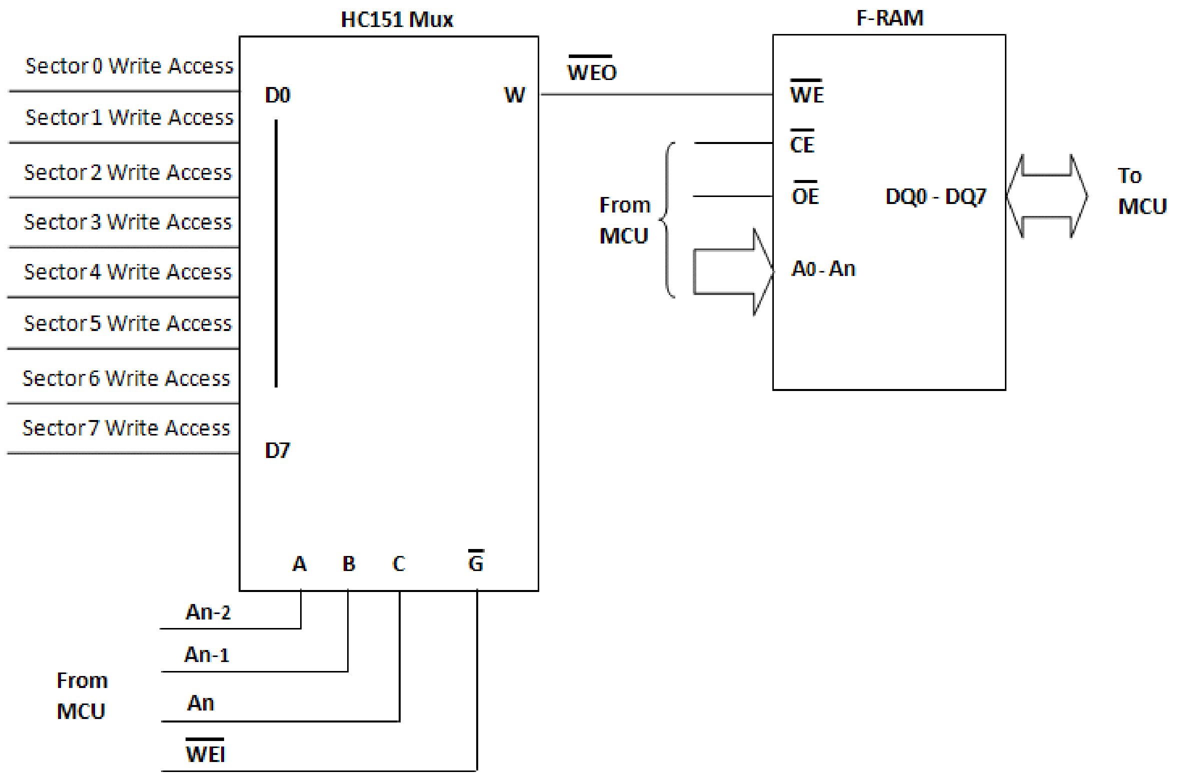
Data memory often needs a mix of volatile and nonvolatile storage, or at least a nonvolatile storage that can be written without restriction. It requires relatively large numbers of writes/reads, and write/read access time can be an important factor. It also requires that a large, even an unlimited, number of writes, is allowed. At the same time, the code memory requires it to be difficult to write. A single-chip solution should be flexible enough to satisfy both these conditions.

In applications like a data logger, the code size required to measure and collect the data can be comparably small with respect to the amount of memory required to store the data. Having a separate code memory might result in under utilization of memory. A single-chip solution will work better for this case. It should be flexible enough to partition the memory for code and data as per the application needs.

The F-RAM technology is currently optimized for data memory applications. Its superior write performance makes it preferable to flash or EEPROM, and its nonvolatile aspect makes it preferable to RAM. Since it is nonvolatile, it can serve as code memory as well. The main limitation of using F-RAM in a single-chip memory application is the available density. Currently, in both parallel and serial (SPI) formats Cypress F-RAM's largest density is 4 Mbit. In applications (specifically battery-operated or energy harvesting) that require less code size and more data size, F-RAMs can be the choice for a single-chip solution for code and data.

When F-RAM is used for code storage, it is critical to ensure that the system does not accidentally write to an area of the memory that is being used for executable code. F-RAM is easily adapted for this by creating a simple write-protection circuit. This logic can be used to prevent inadvertent writes to code areas (or even data areas when required). An example circuit is shown in Figure 1. It provides a programmable block-write protection feature for F-RAM or any other RAM solution. Hardwiring the Sector Write Access inputs creates a fixed block-write protection scheme, while connecting them to logic or a microcontroller provides a method to dynamically change the write protection.

Figure 1. Sector-Write-Protection Circuit



5 Operating the Write-Protection Scheme

The circuit in Figure 1 uses an ordinary CMOS multiplexer such as the HC151 to create an address-dependent write enable that can be connected directly to the F-RAM device. The truth table of the HC151 is provided in Table 1.

Table 1. HC151 Truth Table

Select Inputs			**Strobe: $\overline{G}$ [$\overline{WEI}$]**	**W [$\overline{WEO}$]**
C [An]	**B [An-1]**	**A [An-2]**		
X	X	X	H	H
L	L	L	L	$\overline{D0}$
L	L	H	L	$\overline{D1}$
L	H	L	L	$\overline{D2}$
L	H	H	L	$\overline{D3}$
H	L	L	L	$\overline{D4}$
H	L	H	L	$\overline{D5}$
H	H	L	L	$\overline{D6}$
H	H	H	L	$\overline{D7}$

Sector write-access values set to 1 will allow writes to that sector. Sector write-access values set to 0 will block writes to that sector.

The write input signal from the microcontroller is labeled $\overline{WEI}$, which is connected to the $\overline{G}$ strobe of the HC151 mux. The write output signal from the HC151 mux connected to the F-RAM is labeled $\overline{WEO}$. When the $\overline{WEI}$ signal is HIGH, the $\overline{WEO}$ output from HC151 remains HIGH. When $\overline{WEI}$ goes LOW, the current address and the corresponding write-access input determine if the $\overline{WEO}$ to the F-RAM is permitted to go LOW as explained later in this section.

Assuming that the three most-significant memory address lines (An, An-1, An-2) are connected to HC151, the memory space is divided into eight equal sectors. For each sector, the Dn input of HC151 determines if the sector is write-protected. The incoming address will select one of the eight sectors depending on the three MSB address lines. If the corresponding Dn input is HIGH, then the $\overline{WEO}$ from HC151 will go LOW, provided that $\overline{WEI}$ is LOW. Thus, these sectors can be written. If the selected Dn is LOW, then the $\overline{WEO}$ output will remain HIGH for all addresses in that sector, regardless of the state of the $\overline{WEI}$ input.

This simple circuit allows the system to add programmable block-write protection to any RAM memory including F-RAM. One option is to hard-wire the settings for write access to fixed values based on the locations of code and data. This prevents inadvertent writes but eliminates the possibility of making a field upgrade of the code without making a circuit board change. A minor variation of hard wiring is to use jumpers. This would allow for changes but still require human intervention. A more flexible alternative involves connecting the write-access settings to a microcontroller (possibly via other logic). If the power-on-reset state of these inputs is set to LOW, then the default condition is to write-protect the entire device. Under software control, the system can then alter these settings to either provide limited full-time write access to certain sectors or to dynamically open and close access to all sectors. This scheme allows write-protection of critical data areas and code.

6 Write Protection in Serial F-RAMs

Serial F-RAMs can also be used as code and data memory. Most of the SPI F-RAMs have the block-protect feature. Using this feature, the code memory is write-protected. It can protect the upper quarter, upper half, or the full memory. The block-protect bits (BP1 and BP0) are defined in the status register. Table 2 provides the details of block protection.

Table 2. Block Memory Write Protection

BP1	BP0	Protected Area
0	0	None
0	1	Upper-quarter memory
1	0	Upper-half memory
1	1	Full memory

Serial F-RAMs also define a hardware write-protect pin for extra security against inadvertent writes

7 Summary

AN101 reviews the memory requirements for code, data and both together. It discusses the advantages and disadvantages of using the existing solutions such as Flash and EEPROM against F-RAM. It provides the design considerations for combining code and data in a single F-RAM device. F-RAM is clearly seen as a better single-chip solution for code and data due to a simple system design using write-protect circuit and faster memory access.

Document History

Document Title: AN101 - F-RAM™ as One-Chip Solution for Code and Data Memory Applications

Document Number: 001-87060

Revision	ECN	Orig. of Change	Submission Date	Description of Change
**	4018287	MEDU	06/07/2013	New Spec.
*A	4575174	MEDU	11/20/2014	Added “Write Protection in Serial F-RAMs”.
*B	5293268	MEDU	06/02/2016	Updated to new template.
*C	5836868	HARA	08/17/2017	Updated logo and copyright.

Worldwide Sales and Design Support

Cypress maintains a worldwide network of offices, solution centers, manufacturer's representatives, and distributors. To find the office closest to you, visit us at Cypress Locations.

Products

ARM$^{®}$ Cortex$^{®}$ Microcontrollers	cypress.com/arm
Automotive	cypress.com/automotive
Clocks & Buffers	cypress.com/clocks
Interface	cypress.com/interface
Internet of Things	cypress.com/iot
Memory	cypress.com/memory
Microcontrollers	cypress.com/mcu
PSoC	cypress.com/psoc
Power Management ICs	cypress.com/pmic
Touch Sensing	cypress.com/touch
USB Controllers	cypress.com/usb
Wireless Connectivity	cypress.com/wireless

PSoC® Solutions

PSoC 1 | PSoC 3 | PSoC 4 | PSoC 5LP | PSoC 6

Cypress Developer Community

Forums | WICED IOT Forums | Projects | Videos | Blogs | Training | Components

Technical Support

cypress.com/support

PSoC is a registered trademark and PSoC Creator is a trademark of Cypress Semiconductor Corporation.

Cypress Semiconductor
198 Champion Court
San Jose, CA 95134-1709