Using Atmel's Serial DataFlash™

Introduction

In the past, engineers have struggled to use Flash memory for data storage applications. The traditional Flash memory devices, with their large page sizes of 4K to 128K bytes, make it difficult to change a few bytes. Typically, system designers must include external RAM buffers to shadow the Flash memory page's contents to make the data modifications.

Atmel's Serial DataFlash is a new Flash family designed specifically for data storage applications. Its small page size of 264 bytes for densities of 8M and smaller, provides the system designer with a high level of flexibility and completely simplifies the process of data modifications. Furthermore, the Serial DataFlash incorporates a simple serial interface which facilitates hardware layout, increases system reliability, and minimizes switching noise.

The Serial DataFlash is perfectly suited for digital voice-, image-, and data-storage applications, especially where low power consumption is required. In these storage applications, the Serial DataFlash's small page size not only makes it easier to manipulate data, it also increases storage efficiency. The list below shows some typical applications for the Serial DataFlash.

Digital Voice Storage Applications:

- Digital answering machines
- Voice memo functions in cellular phones
- Voice storage in pagers
- Portable voice memo recorders
- Portable dictation recorders

Image Storage Applications:

- Image storage for digital cameras
- Scanned fax storage for delayed fax sending/receiving

Data Storage Applications:

- "Saved game and high score" data for video game systems
- Phone number and text message storage in pagers
- Data storage in PDAs
- Data acquisition systems

Serial DataFlash™

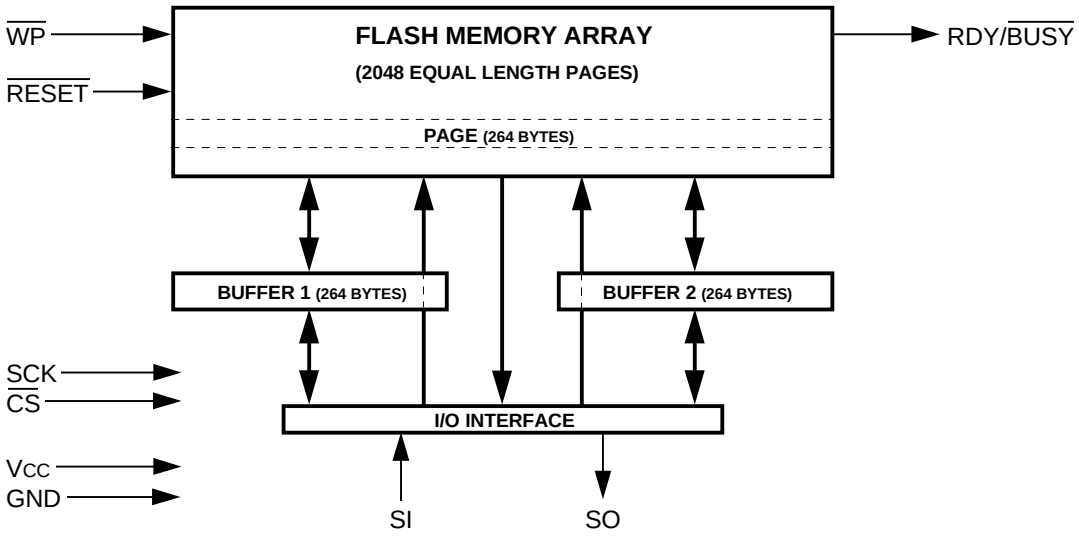
Application Note (AN-4)

Rev. 0842A-A–01/98

Functional Description

The block diagram of Atmel's Serial DataFlash memory shows that the device consists of a Flash memory array, two data buffers, and a simple I/O interface (Figure 1). Unlike conventional Flash memories that are accessed randomly with multiple address lines and a parallel interface, the Serial DataFlash uses a serial interface to sequentially access its data. The serial interface, used to transfer both address and data information, provides a true upgrade/downgrade path

Figure 1. Atmel's AT45D041 Serial DataFlash consists of a Flash memory array, two buffers, and a simple I/O interface.



Serial DataFlash Array

The Flash memory array of the AT45D041, the first device in the Serial DataFlash family, comprises 2048 equal length pages. The AT45D041's page size is 264 bytes, rather than 256 bytes. Therefore, the total density (4,325,376 bits) of the device is 128K bits larger than 4M bits.

System designers can use all of a page's 264 bytes for storing data. Alternatively, the 8 extra bytes per page can be used for error detection and correction mechanisms (EDC) or associated control information, such as pointers, flags, and phone message routing directions. This information, which is potentially vital to the operation of the system, can be used by the microcontroller or processor to determine how to utilize the data stored within the associated page.

For example, in a digital voice messaging system, the AT45D041 pages store the compressed data of digitized voice messages. A page's control information could contain a mailbox number indicating which user can access the voice message stored within that page. Additionally, the system could set a priority level flag to denote that the message is urgent. If the message spans more than one Flash page, the system can store a pointer to indicate which page contains the next portion of the message. The extra 8 bytes of each AT45D041 page provides an area to store this type of control information, greatly enhancing the capability of the system.

The device also provides the ability to securely store critical, infrequently updated information via the Write Protect pin ($\overline{WP}$). When $\overline{WP}$ is held low, the first 256 pages of the array are protected and cannot be programmed. Infrequently updated information, from the digital voice messaging example described above, could be user identification or parameters such as the permitted message length, a message's retention period, and the number of messages a particular user can have. The protected area could also be used to store information such as voice menu prompts, time/date information, secure data or coefficient look-up tables.

Data Buffers Increase Performance

The AT45D041 incorporates two on-chip, bi-directional buffers to expedite the flow of data to and from the device. These buffers provide a built-in "pseudo" cache memory and allow the AT45D041 device to receive data during erase/program operations. Each buffer is 264 bytes long, the same size as a Flash page, and function independently from each other. The system may also use the data buffers as "scratch pad" memory for reads and writes. Therefore, these on-chip buffers may eliminate the need to use off-

chip RAM or RAM contained within the microcontroller or processor.

The AT45D041's buffers are static RAMs (SRAM) and therefore, data stored within the buffers is not guaranteed if the supply voltage drops below the specified minimum operating level. However, the buffers' static nature eliminates the need for refreshing, and data will not change until new data is loaded into the buffers. When loading new data, only those bytes specified to be overwritten will change; the remaining bytes are unaffected. For example, if the user loads only 200 bytes into a buffer, the remaining 64 bytes will still retain their previous values.

Serial Interface Simplifies Upgradability

The sequential access, serial interface scheme employed through the Serial DataFlash's pinout enables a practically limit-free upgrade path for either density or word-width. Conventional random access, parallel interface Flash must use dedicated address pins to interface to the system microcontroller or processor. As density requirements increase, address lines must be added, in turn increasing the number of pins and the size of the device's package. Likewise, a parallel interface Flash requires dedicated I/O pins, and as word-widths increase to 16 or 32 bits, I/O pins must be added, again impacting the package's size.

The Serial DataFlash interfaces with other devices using only seven signal leads, three of which are dedicated to the serial bus (SCK, SI, and SO). The remaining signal leads on the Serial DataFlash include a chip select ($\overline{CS}$), chip reset input ($\overline{RESET}$), a write protect input ($\overline{WP}$), and a ready/busy output (RDY/$\overline{BUSY}$).

Functional Operation of the SPI

The Serial DataFlash can be used with any type of microcontroller, but the interface of the device is also compatible with SPI modes 0 and 3 to provide simple interconnections with the increasingly popular SPI microcontrollers.

SPI is a serial interface protocol, utilizing 8-bit words, useful in communicating with external devices such as serial EEPROMs and the Serial DataFlash. Prior to the availability of SPI EEPROMs, engineers used the standard Microwire EEPROMs to interface with the SPI port on microcontrollers. While the SPI port on microcontrollers was capable of running at 2.1 MHz, it was limited by the EEPROM's 1 MHz operating rate. It wasn't until recently that EEPROMs, such as Atmel's AT25010/020/040 Serial CMOS EEPROMs, became available with the SPI standard interface. Due to SPI's faster clock speed and interface compatibility, this EEPROM device is increasing in popularity; the same applies to the Serial DataFlash.

Controlling Data Flow

The Serial Data Clock (SCK) input pin of the Serial DataFlash must be generated by the master microcontroller or processor, or in some instances a free-running oscillator. All programming cycles in the Serial DataFlash are completely self-timed, so the SCK signal only controls the clocking of data into and out of the device.

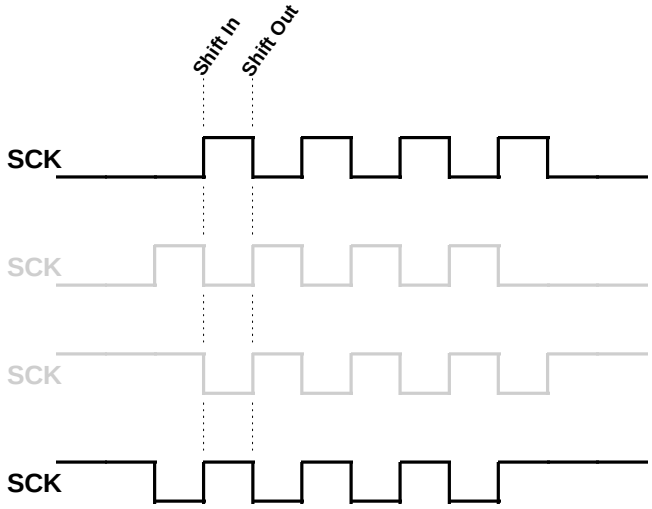
The $\overline{CS}$ pin on the Serial DataFlash functions the same as that of a chip select pin on any memory device. Driving $\overline{CS}$ LOW selects the device; driving $\overline{CS}$ HIGH deselects the device and puts the device into a quiescent state. When $\overline{CS}$ is deselected, the Serial DataFlash ignores any data present on the Serial Data Input (SI) pin, and the Serial Data Output (SO) pin remains in a high-impedance state. The $\overline{CS}$ pin also functions as a trigger to initiate the internal self-timed read and write sequences. The specifics for each sequence and how $\overline{CS}$ relates to them are discussed in the "Read Operations" and "Program Operations" section of the application note.

Table 1. Features of the Serial DataFlash Interface

Maximum Bus Speed	10 MHz
Number of Active Pins	3 or 4
Maximum Memory Size	N/A
Data Size	8 bits
Block Write Capability	Yes
Sequential Read Capability	Yes
Number of Devices on Bus	Limited by Port Pins
Supported SPI Modes	0 and 3

Figure 2. SPI mode determines which edge of the clock signal controls the data transfer direction.

SPI MASTER CONFIGURATION BITS		
CPOL	CPHA	SPI MODE
0	0	0
0	1	1
1	0	2
1	1	3



Note: The shifting out of data does not occur on the falling edge of the same clock cycle as data shifting in, but rather the falling edge of the next clock cycle.

SPI Operating Modes

SPI has four operating modes: 0, 1, 2, and 3. The SPI operating mode determines the clock phase and polarity for transmitting or receiving data. In other words, the mode determines which edge of the clock signal controls the direction of data transfer (Figure 2).

The Serial DataFlash only supports the most commonly used SPI modes, 0 and 3. With these modes, the rising edge of the SCK signal always clocks data in, while the falling edge always clocks data out. Supporting only modes 0 and 3 eliminates the need to integrate special mode select registers within the Serial DataFlash. Examining the clock waveforms in Figure 2, observe that the difference between modes 0 and 3 is the level where SCK starts. When the Serial DataFlash sees a rising edge transition on SCK, this is the indication to latch data in. As long as the designer follows these clock signal conventions, any type of microcontroller or processor may be used as the SPI master — *the Serial DataFlash is not limited to interfacing with SPI-compatible devices*.

Interfacing the Serial DataFlash to a Microcontroller

Atmel's AT89S8252 is an MCS-51 compatible microcontroller with a Serial Peripheral Interface. It supports full-duplex, 3-wire synchronous data transfer with a 6 MHz maximum bit frequency. By enabling the AT89S8252's SPI feature, port 1 pins P1.5-P1.7 can be connected to the Serial DataFlash. This microcontroller contains a series of Special Function Registers (SFRs). Among these SFRs is the SPI Control Register located at SFR address D5H (Table 2). Bits CPOL and CPHA control the SPI mode, while bits SPR0 and SPR1 control the data rate (Table 3).

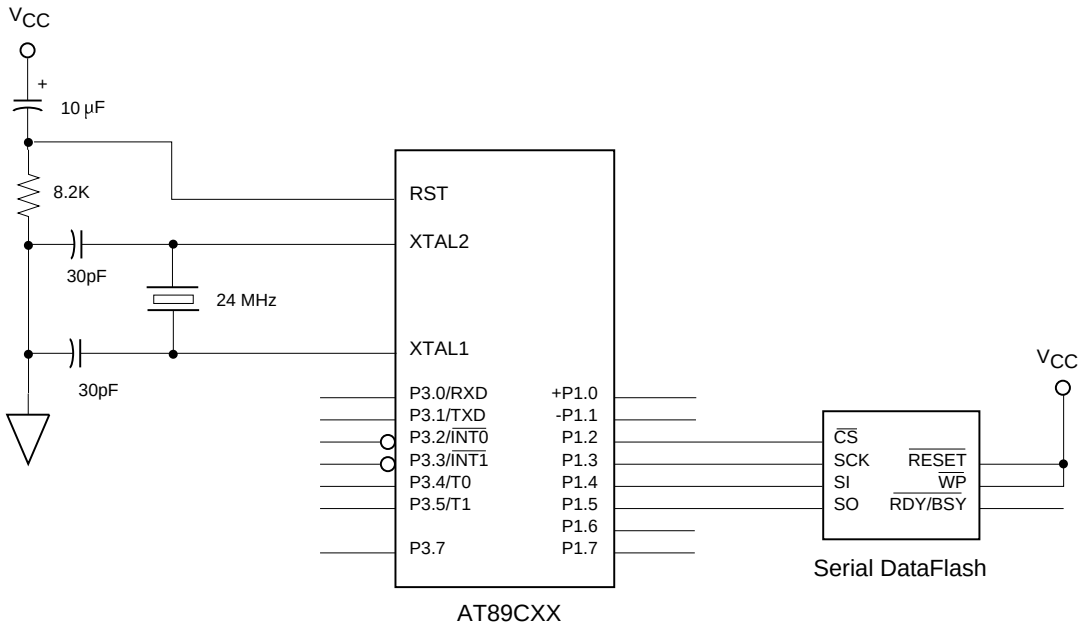
Table 2. SPI Control Register

AT89S8252 Microcontroller, SPI Control Register: SFR Address D5H								
Bit	SPIE	SPE	DORD	MSTR	CPOL	CPHA	SPR1	SPR0
	7	6	5	4	3	2	1	0

Table 3. SPI Control Register Bit Definitions

Symbol	Function
SPIE	SPI Interrupt Enable
SPE	SPI Enable: SPI = 1 enables the SPI channel and connects SS, MOSI, and SC to pins P1.4-P1.7 SPI = 0 disables the SPI channel
DORD	Data Order
MSTR	Master/Slave Select
CPOL	Clock Polarity: CPOL = 1, SCK is high when idle CPOL = 0, SCK of the master device is low when not transmitting
CPHA	Clock Phase. The CPHA bit, together with the CPOL bit, controls the clock and data relationship between master and slave
SPR0 SPR1	SPI Clock Rate Select

Figure 3. The Serial DataFlash can be connected to any microcontroller with the ability to provide a clock signal.



Alternatively, any microcontroller with individually controlled port pins can provide the proper serial interface for the Serial DataFlash. As shown in Figure 3, Atmel's AT89CXX can be connected to the Serial DataFlash. The microcontroller can clock data into the Serial DataFlash at rates up to 10 MHz; whereas the limit on a strict SPI implementation is only 2.1 MHz.

The Reset Function

The Serial DataFlash has a reset function that causes any operation currently in progress to be terminated and forces the device's internal state machine into an idle state. The reset function is activated by holding the device's $\overline{\text{RESET}}$ pin LOW. This feature can be used as a safeguard against system power glitches or when the system supply monitor

(continued)

circuitry detects the supply voltage going below the minimum operating limit. Resetting the Serial DataFlash during these operating conditions prevents any erroneous operations which could result in data corruption (for more information see section on “Write Protection Mechanisms”).

If the Serial DataFlash is reset before the completion of a page program/erase operation, then the data in the page being programmed or erased cannot be guaranteed; the Serial DataFlash must finish the entire operation in order for all data in the page to be valid. If the user wants to ensure that a valid program/erase operation has been performed before resetting the device, then the system must either wait the maximum t_{EP} or t_P time, poll the RDY/$\overline{\text{BUSY}}$ pin, or poll the status register (see section on “Status Register”) to determine the completion of the program/erase operation.

If the system must service a higher level interrupt and must reset the Serial DataFlash before the completion of the program/erase operation, then the system can later program the Flash page again with the same data. Resetting the device will not alter the contents of the internal RAM buffers, so the buffer used to perform the initial program/erase operation before the device was reset will still retain the same data. Therefore, a simple Buffer to Main Memory Page Program with Built-In Erase command can be issued to reprogram the Flash page again.

The Command Interface

In addition to the basic Flash memory functional blocks, the Serial DataFlash device consists of a Command User Interface (CUI) and a state machine that controls all internal operations. The CUI interfaces the system to the Serial DataFlash’s internal state machine. The CUI receives the user’s software commands, translates them into state machine operations, and determines the commands validity.

Status Register

The state machine contains a Status Register that provides feedback on device functions (Table 4). To read the Status Register, begin by loading a Status Register Read command (opcode 57H) into the Serial DataFlash. Next, read eight bits of data from the SO pin. It is not possible to write data into the Status Register, so data will be output after the last bit of the opcode is clocked into the device.

The first bit to be output from the Status Register will be bit 7, the most-significant bit (MSB). Valid data will continue being output through bit 3, while bits 2, 1, and 0 will have unknown values since they are reserved for future use. After bit 0 of the Status Register has been output, the sequence will repeat itself (as long as $\overline{\text{CS}}$ remains LOW) starting again with bit 7. The data in the Status Register is always being updated, so each repeating sequence will contain new data.

You can use the Status Register to determine if the Serial DataFlash is busy or not. The part will be busy during a Main Memory Page to Buffer Transfer, Main Memory Page to Buffer Compare, Buffer to Main Memory Page Program with Built-In Erase, Buffer to Main Memory Page Program without Built-In Erase, Main Memory Page Program, or an Auto Page Rewrite operation. The first bit (MSB) out of the Status Register indicates the ready/busy status, which is derived from the operational status of the internal state machine. If this bit is a 0, the Serial DataFlash is busy performing one of the operations listed above; if this bit is a 1, the part is not busy and is ready to accept a new command. You can also use the Serial DataFlash’s RDY/$\overline{\text{BUSY}}$ pin to determine the same information.

The second bit out of the Status Register indicates the outcome of the most recent Main Memory Page to Buffer Compare operation. If the data in the main memory page matches the data in the buffer, this bit will be a 0; if at least one bit of the data does not match, this will be a 1.

The next three bits out of the Status Register indicate the device density of the Serial DataFlash being used. The three bits represent a code relating to different Serial DataFlash densities, allowing a total of eight density combinations. Please refer to Table 5 for the list of codes and densities.

Table 4. Status Register Bit Definitions

Order shifted out	1	2	3	4	5	6	7	8
Status Register Bit	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
	RDY/$\overline{\text{BUSY}}$	Compare	Density Code			Reserved for future use		

Table 5. Status Register Density Codes

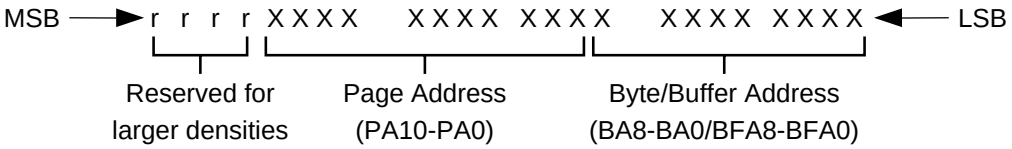
BIT 5	BIT 4	BIT 3	DEVICE DENSITY
0	0	0	512K
0	0	1	1M
0	1	0	2M
0	1	1	4M
1	0	0	8M
1	0	1	16M
1	1	0	32M
1	1	1	64M

Command Table

To begin an operation on the Serial DataFlash, the system must send a command to the device. Table 6 shows the bit sequence to follow for each Serial DataFlash operation. All operations (except for Status Register Read) start with an opcode followed by three address bytes that are clocked into the Serial DataFlash.

The three address bytes (24 bits) are used to address the memory array or buffers for the AT45D041. As shown in Figure 4, the four MSB bits are *Reserved* bits; bits 5-15 denote a page number; bits 16-24 denote a specific byte address within the 264-byte page or buffer. This 24-bit addressing scheme allows the system to address up to 64M bytes. The four *Reserved* bits of the AT45D041 will be address bits for larger density devices and should be zero to ensure upwards compatibility.

Figure 4. Command Sequence for AT45D041 Read/Write Operations (Except Status Register Read).



Note: 1. "r" designates bits reserved for larger densities.
2. It is recommended that "r" be a logical "0" for densities of 4M bit or smaller.
3. For densities larger than 4M bit, the "r" bits become the most significant Page Address bit for the appropriate density.

Table 6.

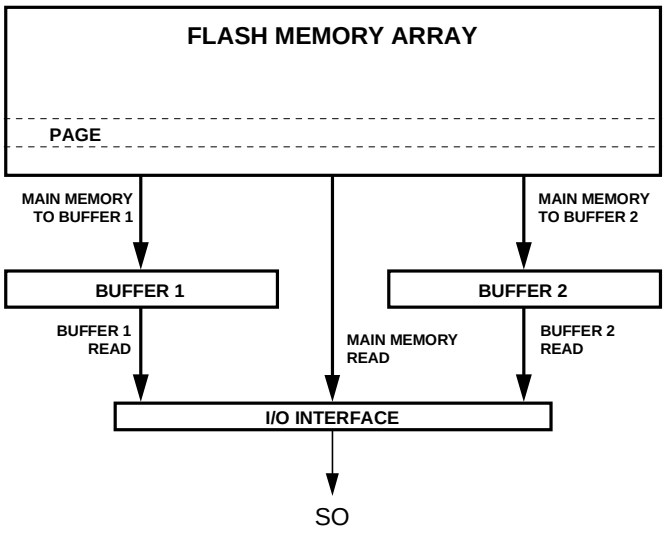
Main Memory Page Read	Buffer 1 Read	Buffer 2 Read	Main Memory Page to Buffer 1 Transfer	Main Memory Page to Buffer 2 Transfer	Main Memory Page to Buffer 1 Compare	Main Memory Page to Buffer 2 Compare	Buffer 1 Write	Buffer 2 Write
Opcode								
52H	54H	56H	53H	55H	60H	61H	84H	87H
0	0	0	0	0	0	0	1	1
1	1	1	1	1	1	1	0	0
0	0	0	0	0	1	1	0	0
1	1	1	1	1	0	0	0	0
0	0	0	0	0	0	0	0	0
0	1	1	0	1	0	0	1	1
1	0	1	1	0	0	0	0	1
0	0	0	1	1	0	1	0	1
r	r	r	r	r	r	r	r	r
r	r	r	r	r	r	r	r	r
r	r	r	r	r	r	r	r	r
r	r	r	r	r	r	r	r	r
PA10	r	r	PA10	PA10	PA10	PA10	r	r
PA9	r	r	PA9	PA9	PA9	PA9	r	r
PA8	r	r	PA8	PA8	PA8	PA8	r	r
PA7	r	r	PA7	PA7	PA7	PA7	r	r
PA6	r	r	PA6	PA6	PA6	PA6	r	r
PA5	r	r	PA5	PA5	PA5	PA5	r	r
PA4	r	r	PA4	PA4	PA4	PA4	r	r
PA3	r	r	PA3	PA3	PA3	PA3	r	r
PA2	r	r	PA2	PA2	PA2	PA2	r	r
PA1	r	r	PA1	PA1	PA1	PA1	r	r
PA0	r	r	PA0	PA0	PA0	PA0	r	r
BA8	BFA8	BFA8	X	X	X	X	BFA8	BFA8
BA7	BFA7	BFA7	X	X	X	X	BFA7	BFA7
BA6	BFA6	BFA6	X	X	X	X	BFA6	BFA6
BA5	BFA5	BFA5	X	X	X	X	BFA5	BFA5
BA4	BFA4	BFA4	X	X	X	X	BFA4	BFA4
BA3	BFA3	BFA3	X	X	X	X	BFA3	BFA3
BA2	BFA2	BFA2	X	X	X	X	BFA2	BFA2
BA1	BFA1	BFA1	X	X	X	X	BFA1	BFA1
BA0	BFA0	BFA0	X	X	X	X	BFA0	BFA0
X	X	X						
X	X	X						
X	X	X						
X	X	X						
X	X	X						
X	X	X						
X	X	X						
X	X	X						
•								
•								
•								
X (64th bit)								

X (Don't Care)
r (reserved bits)

Buffer 1 to Main Memory Page Program with Built-In Erase	Buffer 2 to Main Memory Page Program with Built-In Erase	Buffer 1 to Main Memory Page Program without Built-In Erase	Buffer 2 to Main Memory Page Program without Built-In Erase	Main Memory Page Program Through Buffer 1	Main Memory Page Program Through Buffer 2	Auto Page Rewrite Through Buffer 1	Auto Page Rewrite Through Buffer 2	Status Register
Opcode								
83H	86H	88H	89H	82H	85H	58H	59H	57H
1	1	1	1	1	1	0	0	0
0	0	0	0	0	0	1	1	1
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	1	1	1
0	0	1	1	0	0	1	1	0
0	1	0	0	0	1	0	0	1
1	1	0	0	1	0	0	0	1
1	0	0	1	0	1	0	1	1
r	r	r	r	r	r	r	r	
r	r	r	r	r	r	r	r	
r	r	r	r	r	r	r	r	
r	r	r	r	r	r	r	r	
PA10	PA10	PA10	PA10	PA10	PA10	PA10	PA10	
PA9	PA9	PA9	PA9	PA9	PA9	PA9	PA9	
PA8	PA8	PA8	PA8	PA8	PA8	PA8	PA8	
PA7	PA7	PA7	PA7	PA7	PA7	PA7	PA7	
PA6	PA6	PA6	PA6	PA6	PA6	PA6	PA6	
PA5	PA5	PA5	PA5	PA5	PA5	PA5	PA5	
PA4	PA4	PA4	PA4	PA4	PA4	PA4	PA4	
PA3	PA3	PA3	PA3	PA3	PA3	PA3	PA3	
PA2	PA2	PA2	PA2	PA2	PA2	PA2	PA2	
PA1	PA1	PA1	PA1	PA1	PA1	PA1	PA1	
PA0	PA0	PA0	PA0	PA0	PA0	PA0	PA0	
X	X	X	X	BA8	BA8	X	X	
X	X	X	X	BA7	BA7	X	X	
X	X	X	X	BA6	BA6	X	X	
X	X	X	X	BA5	BA5	X	X	
X	X	X	X	BA4	BA4	X	X	
X	X	X	X	BA3	BA3	X	X	
X	X	X	X	BA2	BA2	X	X	
X	X	X	X	BA1	BA1	X	X	
X	X	X	X	BA0	BA0	X	X	

X (Don’t Care)
r (reserved bits)

Figure 5. Data can be read directly from main memory or though the buffers.



Read Operations for the AT45D041

By specifying the appropriate opcode, data can be read from the main memory or from either one of the two buffers (Figure 5).

Main Memory Page Read

A main memory read allows the user to read data directly from any one of the 2048 pages, bypassing both of the data buffers and leaving the buffer contents unchanged. To start a page read, the Main Memory Page Read command (52H) is clocked into the device, followed by four *Reserved* bits, 20 address bits, and 32 don't care bits. The four *Reserved* bits will be address bits used for future expansion and should be zero to ensure upwards compatibility. The 32 don't care bits are sent to give the Serial DataFlash's state machine time to initialize.

When data is read from the main memory, you must specify the page address and the address of the first byte to be read within the page. Specifying the page address requires 11 bits. Specifying the first byte to be read within the page requires nine bits. While reading data from main memory, if the end of the page is reached, the Serial DataFlash will wrap around back to the beginning of the page.

After a high to low transition occurs on the $\overline{CS}$ pin, toggling the SCK pin loads the eight opcode bits, four *Reserved* bits, 20 address bits, and 32 don't care bits from the SI pin; at this point data can be read serially from the SO pin. The $\overline{CS}$ pin must remain low during this entire sequence; a low to high transition of the $\overline{CS}$ pin will terminate the read operation and tri-state the SO pin.

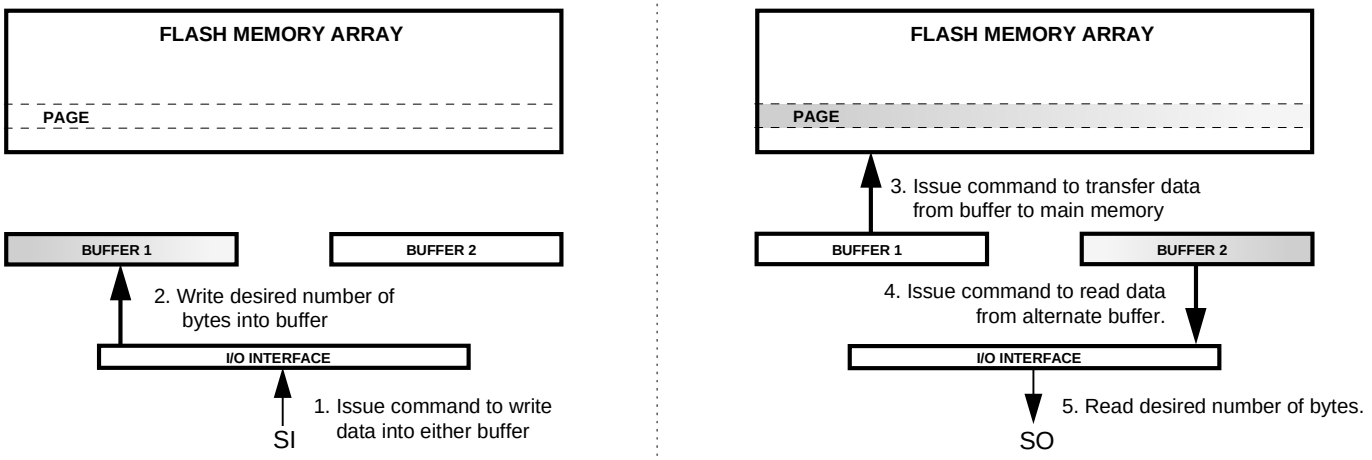
Buffer Read

A buffer read allows the user to read data directly from either of the two buffers. To start a buffer read, a Buffer Read command (54H for buffer 1, 56H for buffer 2) is clocked into the device, followed by 15 *Reserved* bits, nine address bits, and eight don't care bits. The 15 *Reserved* bits may be used for future expansion and should be zero to ensure upwards compatibility. Nine address bits are required to specify the first byte of data to be read from the 264-byte buffer. The eight don't care bits are sent to give the Serial DataFlash's state machine time to initialize. While reading data from a buffer, if the end of the buffer is reached, the Serial DataFlash will wrap around back to the beginning of the buffer.

After a high to low transition occurs on the $\overline{CS}$ pin, toggling the SCK pin loads the eight opcode bits, 15 *Reserved* bits, nine address bits, and eight don't care bits from the SI pin; at this point data can be read serially from the SO pin. The $\overline{CS}$ pin must remain low during this entire sequence; a low to high transition of the $\overline{CS}$ pin will terminate the read operation, and tri-state the SO pin.

Note: You can read from one buffer while the Serial DataFlash's state machine is transferring data from the other buffer into main memory (Figure 6).

Figure 6. Serial DataFlash supports virtual read-while-write operations.



Main Memory Page to Buffer Transfer

The Serial DataFlash's state machine can automatically transfer data in a main memory page to either buffer 1 or buffer 2. This allows the user to modify one or more bytes of data in a main memory page and then write the modified buffer contents back into main memory.

To start a data transfer, a Main Memory Page to Buffer Transfer command (53H for buffer 1, 55H for buffer 2) is followed by four *Reserved* bits, 11 address bits, and nine don't care bits. The four *Reserved* bits may be used for future expansion and should be zero to ensure upwards compatibility. 11 address bits are required to specify the page in main memory that is to be transferred to the selected buffer.

After a high to low transition occurs on the $\overline{CS}$ pin, toggling the SCK pin loads the eight opcode bits, four *Reserved* bits, 11 address bits, and nine don't care bits from the SI pin. The data transfer begins when there is a low to high transition on the $\overline{CS}$ pin. You can use the RDY/$\overline{BUSY}$ pin or the RDY/$\overline{BUSY}$ bit in the Status Register to determine whether the state machine has completed the transfer. Refer to the section on the Status Register for details on how to access and interpret the Status Register.

Main Memory Page to Buffer Compare

The Serial DataFlash's internal state machine can be used to automatically compare the data in a main memory page to the data in either buffer 1 or buffer 2. This operation is useful after performing a Buffer to Main Memory Page Program or a Main Memory Page Program Command, for verifying that the Serial DataFlash successfully programmed the buffer contents into a main memory page.

To start the compare operation, a Main Memory Page to Buffer Compare command (60H for buffer 1, 61H for buffer 2) is followed by four *Reserved* bits, 11 address bits, and nine don't care bits. The four *Reserved* bits may be used for future expansion and should be zero to ensure upwards compatibility. 11 address bits are required to specify the page in main memory that is to be compared with the selected buffer.

After a high to low transition occurs on the $\overline{CS}$ pin, toggling the SCK pin loads the eight opcode bits, four *Reserved* bits, 11 address bits, and nine don't care bits from the SI pin. The compare operation begins when there is a low to high transition on the $\overline{CS}$ pin. You can use the RDY/$\overline{BUSY}$ pin or the RDY/$\overline{BUSY}$ bit in the Status Register to determine whether the state machine has completed the compare separation. Refer to the section on the Status Register for details on how to access and interpret the Status Register.

Note: On completion of the compare operation, the state machine updates the second MSB of the Status Register with the result of the compare.

Program Operations for the AT45D041

By specifying the appropriate opcode, data can be written to main memory or to either one of the two buffers (Figure 7).

Buffer Write

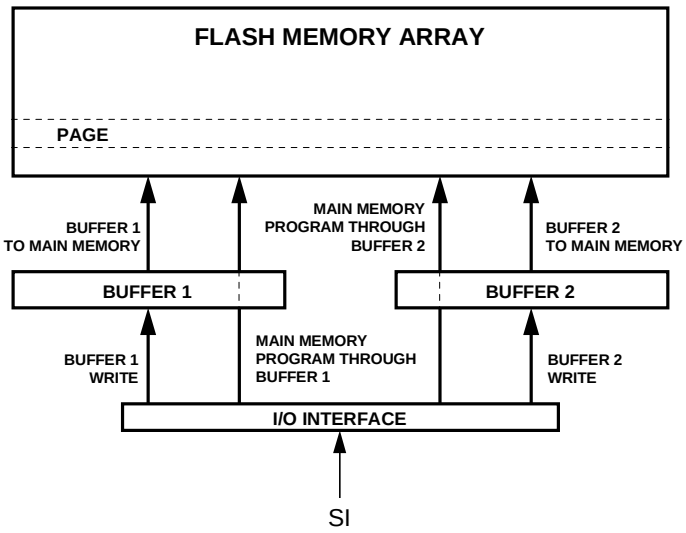
A buffer write allows the user to write data directly into either of the two buffers. To start a buffer write, a Buffer Write command (84H for buffer 1, 87H for buffer 2) is followed by 15 *Reserved* bits and nine address bits. The 15 *Reserved* bits may be used for future expansion and should be zero to ensure upwards compatibility. Nine address bits are required to specify the first byte of data to be written in the 264-byte buffer.

After a high to low transition occurs on the $\overline{CS}$ pin, toggling the SCK pin loads the eight opcode bits, 15 *Reserved* bits, and nine address bits from the SI pin; at this point data can

be written serially from the SI pin. While writing data to a buffer, if the end of the buffer is reached, the Serial DataFlash will wrap around back to the beginning of the buffer. The $\overline{CS}$ pin must remain low during this entire sequence; a low to high transition of the $\overline{CS}$ pin will terminate the write operation.

Note: Buffer locations not written to will remain unchanged from their previous values. Any unused buffer locations should be written with “known” data before performing a Buffer to Main Memory Page operation. Atmel recommends writing ones to unused locations to lower energy consumption.

Figure 7. Serial DataFlash write operations.



Buffer to Main Memory Page Program with Built-In Erase

The Serial DataFlash’s state machine can automatically erase a main memory page and then transfer data from either buffer 1 or buffer 2 into that main memory page. This allows the user to quickly write data to a buffer and not have to issue separate commands to pre-erase a page in the Flash array. You can also use the Buffer to Main Memory Page Program with Built-In Erase command to leave a page in the erased state; to do this, write all 1’s to the pages.

Before performing the Buffer to Main Memory Page Program operation, use the Buffer Write operation to write the desired data to either buffer. To start the Buffer to Main Memory Page Program with Built-In Erase command, an 8-bit opcode (83H for buffer 1, 86H for buffer 2) is followed by four *Reserved* bits, 11 address bits, and nine don’t care bits. The four *Reserved* bits may be used for future expansion and should be zero to ensure upwards compatibility. 11 address bits are required to specify the page in main memory that is to be erased and then written with the buffer contents.

After a high to low transition occurs on the $\overline{CS}$ pin, toggling the SCK pin loads the eight opcode bits, four *Reserved* bits, 11 address bits, and nine don’t care bits from the SI pin. The erase/program operation begins when there is a low to high transition on the $\overline{CS}$ pin. You can use the RDY/$\overline{BUSY}$ pin or the RDY/$\overline{BUSY}$ bit in the Status Register to determine whether the state machine has completed the self-timed operation. Refer to the section on the Status Register for details on how to access and interpret the Status Register.

Note: While the state machine is busy transferring data from one buffer to the main memory page, the other buffer may be read from or written to.

Buffer to Main Memory Page Program without Built-In Erase

The Serial DataFlash’s state machine can automatically transfer data from either buffer 1 or buffer 2 into a main memory page that has been previously erased. This operation allows the user to quickly write data to a buffer and not have to wait for relatively long with Built-In Erase Flash memory erase time. Note that the Buffer to Main Memory Page Program without Built-In Erase is approximately 30% faster than the Buffer to Main Memory Page Program with Built-In Erase operation.

Before you use the Buffer to Main Memory Page Program without Built-In Erase operation, it is necessary that the main memory page that is being programmed has been previously erased (to erase a page, use the Buffer to Main Memory Page Program with Built-In Erase operation and program in all 1’s).

Before performing the Buffer to Main Memory Page Program without Built-In Erase operation, use the Buffer Write operation to write the desired data to either buffer. To start

the Buffer to Main Memory Page Program without Built-In Erase command, an 8-bit opcode (88H for buffer 1, 89H for buffer 2) is followed by four *Reserved* bits, 11 address bits, and nine don't care bits. The four *Reserved* bits may be used for future expansion and should be zero to ensure upwards compatibility. 11 address bits are required to specify the page in main memory that is to be written with the buffer contents.

After a high to low transition occurs on the $\overline{CS}$ pin, toggling the SCK pin loads the eight opcode bits, four *Reserved* bits, 11 address bits, and nine don't care bits from the SI pin. The program operation begins when there is a low to high transition on the $\overline{CS}$ pin. You can use the RDY/$\overline{BUSY}$ pin or the RDY/$\overline{BUSY}$ bit in the Status Register to determine whether the state machine has completed the self-timed operation. Refer to the section on the Status Register for details on how to access and interpret the Status Register.

Note: While the state machine is busy transferring data from one buffer to the main memory page, the other buffer may be read from or written to.

Main Memory Page Program

The Main Memory Page Program operation allows you to write to the buffer and transfer the buffer contents to the specified main memory page using a single command. The operation also erases the main memory page before the buffer's data is transferred.

To start the program operation in main memory, a Main Memory Page Program command (82H for buffer 1, 85H for buffer 2) is followed by four *Reserved* bits, and 20 address bits. The four *Reserved* bits may be used for future expansion and should be zero to ensure upwards compatibility. When referencing main memory data, you must specify the page address and the address of the first byte to be written within the buffer. Specifying the page address requires 11 bits. Specifying the first byte to be written within the buffer requires nine bits.

After a high to low transition occurs on the $\overline{CS}$ pin, toggling the SCK pin loads the eight opcode bits, four *Reserved* bits, and 20 address bits from the SI pin. The Serial DataFlash is now ready to take data from the SI pin and store it in the selected data buffer. While writing data to a buffer, if the end of the buffer is reached, the Serial DataFlash will wrap around back to the beginning of the buffer. The $\overline{CS}$ pin must remain low during this entire sequence; a low to high transition of the $\overline{CS}$ pin will initiate the erase of the selected main memory page and then program the data stored in the buffer to that page.

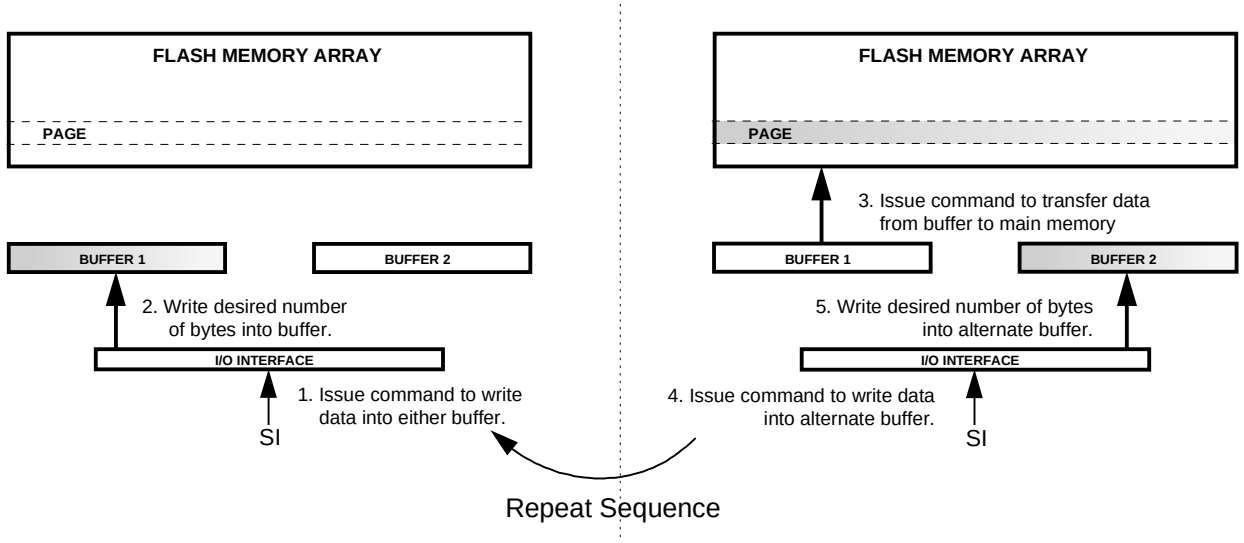
Once the main memory page erase/program has begun, you can use the RDY/$\overline{BUSY}$ pin or the RDY/$\overline{BUSY}$ bit in the Status Register to determine whether the state machine has completed the self-timed operation. Refer to the section on the Status Register for details on how to access and interpret the Status Register.

Note: While the state machine is busy transferring data from one buffer to the main memory page, the other buffer may be read from or written to.

Virtual Continuous Write Buffer Operation

The previous sections note that while the state machine is busy transferring data from one buffer to the main memory page, the other buffer may be read from or written to. This feature of the Serial DataFlash allows a virtually continuous write operation provided each of the buffers is not filled faster than the maximum page erase and program time (data cannot be clocked in at a rate in which the time to fill a buffer is less than the maximum t_{EP} time specified in the data sheet). Figure 8 shows the sequence of events that system software can use.

Figure 8. The buffers support a virtually continuous write operation.



- Step 1: Issue a Buffer Write command to write data into either buffer.
- Step 2: Write the desired number of bytes into the buffer.
- Step 3: Issue a Buffer to Main Memory Page Program command.
- Step 4: Issue a Buffer Write command to write data into the alternate buffer.
- Step 5: Write the desired number of bytes into the alternate buffer.
- Step 6: Monitor RDY/$\overline{\text{BUSY}}$ and when the status indicates that the Serial DataFlash is not busy, issue a Buffer to Main Memory Page Program command (for alternate buffer). Return to step 1.

Auto Page Rewrite Command

The Auto Page Rewrite operation allows the Serial DataFlash to automatically rewrite the contents of a main memory page. This operation is a combination of two operations: Main Memory Page to Buffer Transfer and Buffer to Main Memory Page Program with Built-In Erase.

To start the rewrite operation, an Auto Page Rewrite command (58H for buffer 1, 59H for buffer 2) is followed by four *Reserved* bits, 11 address bits, and nine don't care bits. The four *Reserved* bits may be used for future expansion and should be zero to ensure upwards compatibility. 11 address bits are required to specify the page in main memory.

After a high to low transition occurs on the $\overline{\text{CS}}$ pin, toggling the SCK pin loads the eight opcode bits, four *Reserved* bits, 11 address bits, and nine don't care bits from the SI pin. When a low to high transition occurs on the $\overline{\text{CS}}$ pin, the Serial DataFlash transfers data from the page in main memory to the specified buffer, and then programs the data in the buffer back into the same page of main memory. Once the Auto Page Rewrite operation has begun, you can use the RDY/$\overline{\text{BUSY}}$ pin or the RDY/$\overline{\text{BUSY}}$ bit in the Status Register to determine whether the state machine has completed the self-timed operation. Refer to the section on the Status Register for details on how to access and interpret the Status Register.

Extended Reprogramming

To improve the reprogramming ability of the Serial DataFlash for write intensive applications that do not write in a cyclical, sequential manner, certain guidelines must be followed to preserve the integrity of data stored within the Flash array. A write intensive application can be defined as any application in which thousands of cumulative reprogram (erase/program) operations are performed throughout the course of the product's life cycle.

If the reprogram operations occur in a cyclical, sequential manner, then no special guidelines need to be followed. That is, if the Flash pages are updated/rewritten beginning with a specific page (e.g., page 6) and continuing sequentially through the next 2047 pages (e.g., pages 7-2047 and 0-5), and cycled again starting back at the original page (e.g., page 6), then no additional algorithms need to be incorporated into the system's microcontroller or microprocessor software.

However, if the reprogram operations occur in a random fashion in which any number of pages is updated in a random order, then the system must ensure that each page of the Serial DataFlash memory array be updated/rewritten at least once within every 10,000 cumulative page reprogram operations. Depending on the type of application, different methodologies can be used to accomplish the updating of the Flash array.

One method requires that every reprogram operation of a single page be followed by an additional page update. In this scenario, a software controlled pointer would be used to designate which additional page of the Flash array is to be updated. For example, the pointer would initially point to page 0. When the system reprograms a page, say page 12, the system would then issue the Auto Page Rewrite command for page 0 after the completion of the page 12 erase/program operation (t_{EP}). The pointer would then be incremented to point to page 1. When the system reprograms another page, the process would be repeated. When the pointer reaches 2048, it would be reset back to 0. Figure 9 illustrates this example.

Another method for updating the Flash array is somewhat similar to the previous method, but would accommodate reprogram operations of multiple pages. With this method, a software controlled pointer would again be implemented with the addition of a software controlled counter. Like the previous example, the pointer would initially point to page 0 and the counter would be set to 0. When the system reprograms multiple pages, the counter would be incremented for each page reprogram operation. After the system completes the programming of the multiple pages (e.g., pages 4, 18, 23 and 25), the system would then issue the Auto Page Rewrite command for page 0. Once the rewrite operation of page 0 is complete, the pointer would be incremented to point to page 1, and the counter would decrement from 4 to 3. The system would continue by issuing the Auto Page Rewrite command for page 1, and the rewrite process would repeat until the counter decremented back to 0. When the pointer reaches 2048, it would be reset back to 0. Figure 10 illustrates this example.

The final method for updating the Flash array would allow 10,000 reprogram operations to occur before the any Auto Page Rewrite command needs to be issued. With this method, a software controlled counter needs to be implemented. The counter would initially be set to 0, and after every reprogram operation, the counter would be incremented. Once the counter reaches 10,000, the system would begin the Auto Page Rewrite process by issuing the Auto Page Rewrite command for page 0. Once the rewrite operation of page 0 is complete, the system would continue by rewriting pages 1, 2, 3, and so on until all 2048 pages have been rewritten. After the entire Flash array has been rewritten, the counter would be reset back to 0. Figure 11 illustrates this example.

Figure 9.

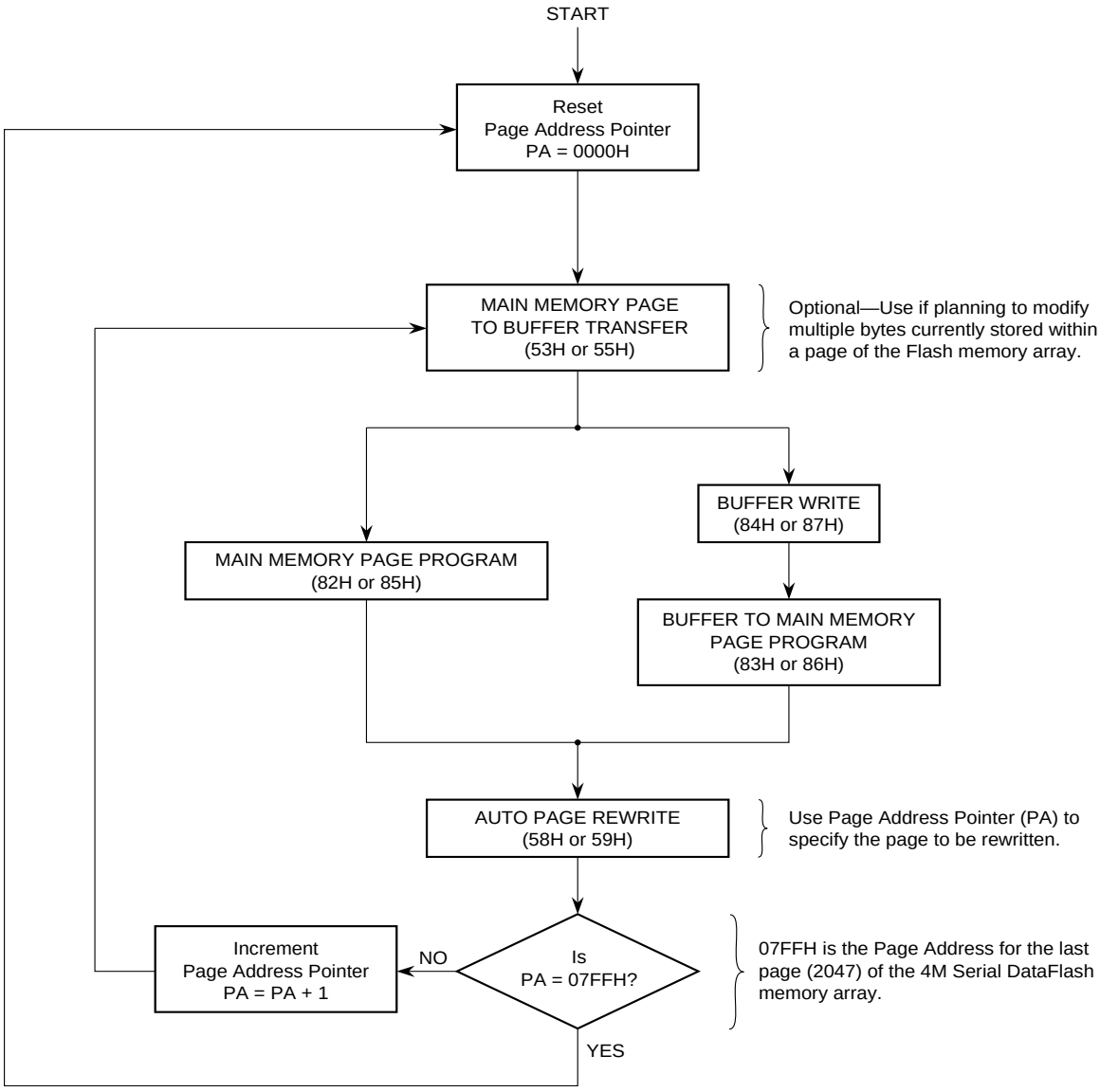


Figure 10.

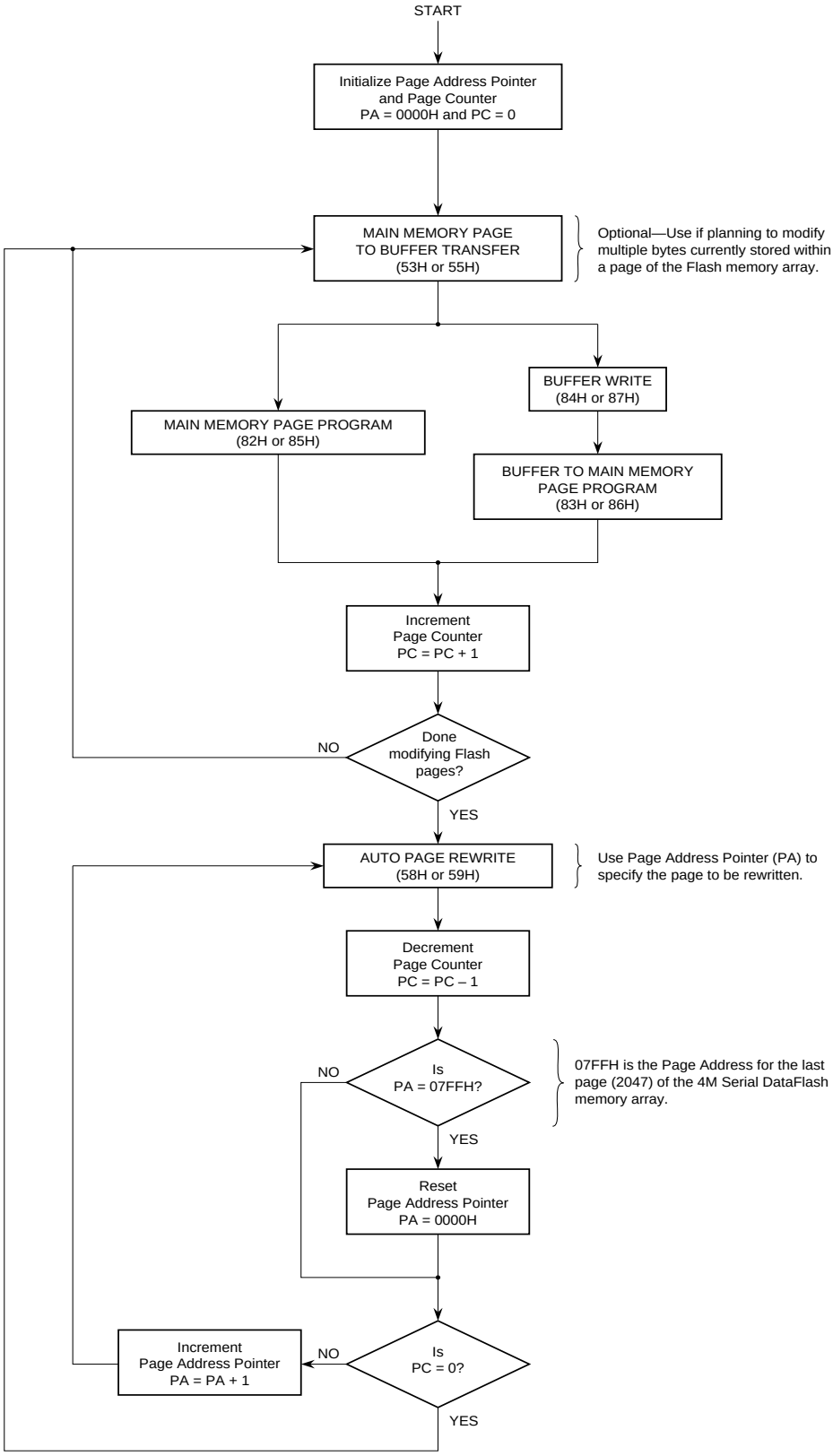

START
Initialize Page Address Pointer and Page Counter PA = 0000H and PC = 0
MAIN MEMORY PAGE TO BUFFER TRANSFER (53H or 55H)
Optional—Use if planning to modify multiple bytes currently stored within a page of the Flash memory array.
BUFFER WRITE (84H or 87H)
MAIN MEMORY PAGE PROGRAM (82H or 85H)
BUFFER TO MAIN MEMORY PAGE PROGRAM (83H or 86H)
Increment Page Counter PC = PC + 1
Done modifying Flash pages?
NO
YES
AUTO PAGE REWRITE (58H or 59H)
Use Page Address Pointer (PA) to specify the page to be rewritten.
Decrement Page Counter PC = PC – 1
Is PA = 07FFH?
NO
YES
07FFH is the Page Address for the last page (2047) of the 4M Serial DataFlash memory array.
Reset Page Address Pointer PA = 0000H
Is PC = 0?
NO
YES
Increment Page Address Pointer PA = PA + 1

Figure 11.

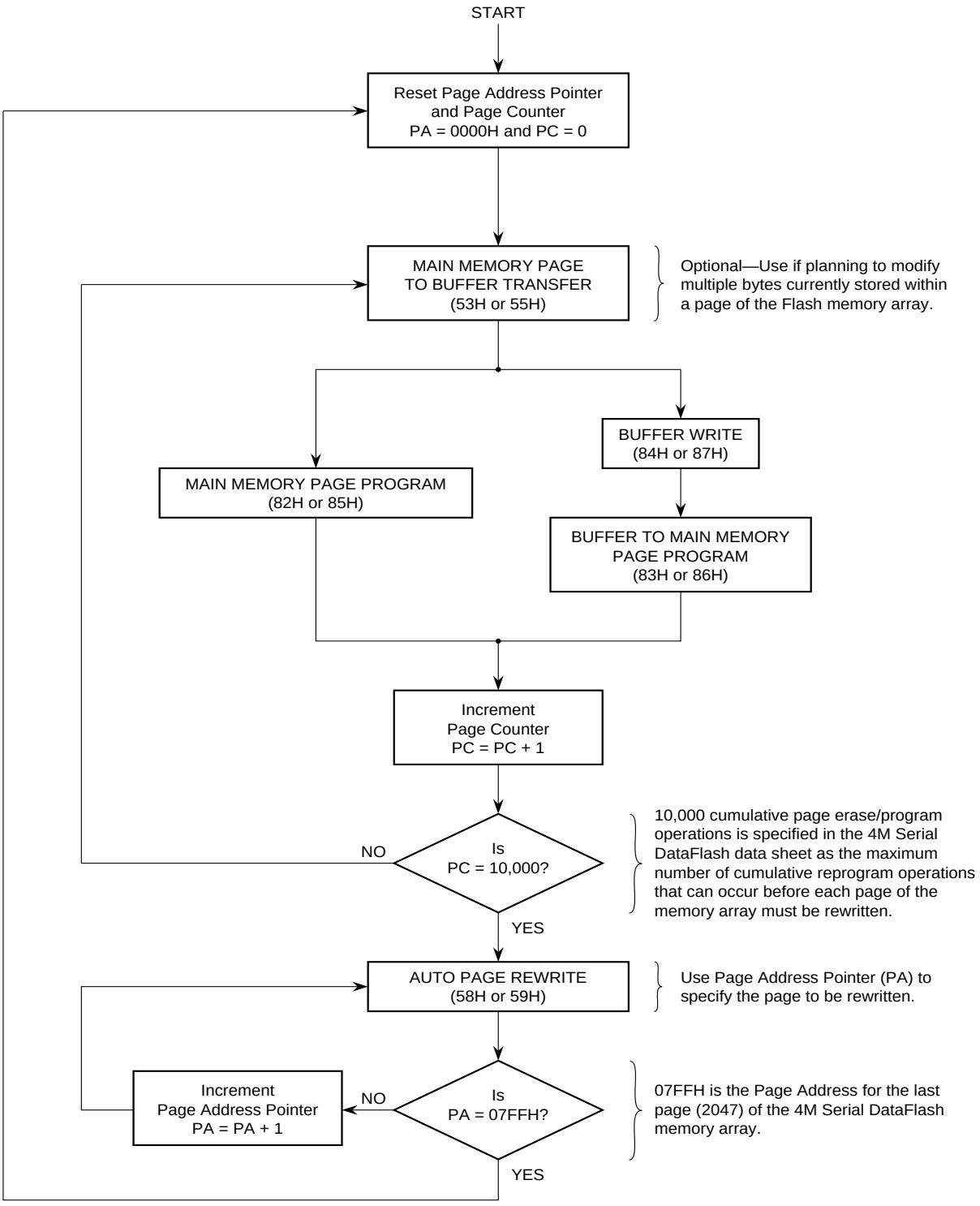
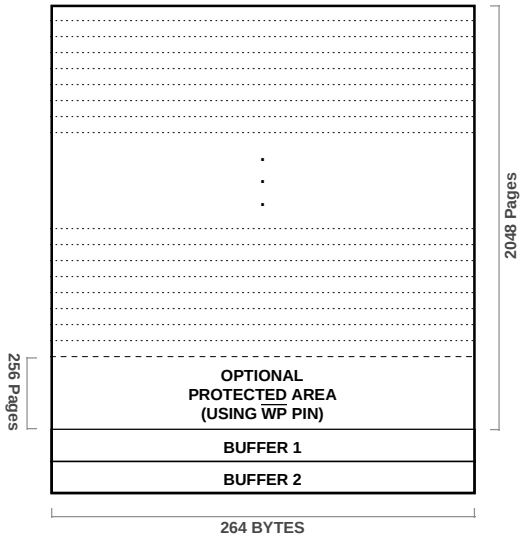

START
Reset Page Address Pointer and Page Counter PA = 0000H and PC = 0
MAIN MEMORY PAGE TO BUFFER TRANSFER (53H or 55H)
Optional—Use if planning to modify multiple bytes currently stored within a page of the Flash memory array.
BUFFER WRITE (84H or 87H)
MAIN MEMORY PAGE PROGRAM (82H or 85H)
BUFFER TO MAIN MEMORY PAGE PROGRAM (83H or 86H)
Increment Page Counter PC = PC + 1
NO
Is PC = 10,000?
10,000 cumulative page erase/program operations is specified in the 4M Serial DataFlash data sheet as the maximum number of cumulative reprogram operations that can occur before each page of the memory array must be rewritten.
YES
AUTO PAGE REWRITE (58H or 59H)
Use Page Address Pointer (PA) to specify the page to be rewritten.
Increment Page Address Pointer PA = PA + 1
NO
Is PA = 07FFH?
07FFH is the Page Address for the last page (2047) of the 4M Serial DataFlash memory array.
YES

Figure 12. The first 256 pages of the Serial DataFlash can be hardware write protected.



Data Protection Mechanisms

A system designer needs to be aware of the possibility of data corruption caused by inadvertent data writes. The Serial DataFlash can have data corruption problems due to glitches, noise spikes, bus contention, etc., which may initiate a false program or erase cycle. The Serial DataFlash provides several mechanisms that can be used to prevent data corruption.

The Write Protect Signal

As shown in Figure 12, the $\overline{WP}$ pin of the Serial DataFlash provides hardware-controlled write protection for the first 256 pages of the Flash memory array (address locations 00000H to 1FF07H). When the $\overline{WP}$ pin is LOW, any attempts (intentional or accidental) to write to the protected region will not affect the previously stored data. However, the erroneous write attempt causes the Serial DataFlash to perform a "dummy" write cycle (as though a normal write operation had occurred). A HIGH level on the $\overline{WP}$ pin disables the write protection feature, allowing the system to write to all pages of the Flash array.

Note: When this write protect feature is enabled, the first 256 pages of the Flash memory array do not have to undergo the rewrite procedure (as described in the "Extended Reprogramming" section) as long as $\overline{WP}$ has been held low during the cumulative reprogramming of the main array.

Using the $\overline{RESET}$ Signal for Data Protection

The Serial DataFlash's $\overline{RESET}$ pin can be connected to the system's reset line which will keep $\overline{RESET}$ held low and the Serial DataFlash inactive, until the power supply is within tolerance. When using this approach, you must ensure that the memory wakes up before the CPU issues memory read cycles to it. The Serial DataFlash also incorporates an internal power-on reset circuit; therefore, it is not required to hold $\overline{RESET}$ low during power-on sequences.

The $\overline{RESET}$ pin is level sensitive and can also be used to protect the entire memory array from inadvertent writes during power outages. The $\overline{RESET}$ pin can also be used to terminate any operation in progress. When terminating erase or program operations before the specified completion time, the data being erased or programmed cannot be guaranteed.

Summary

The Serial DataFlash was designed to provide a new nonvolatile memory device to easily and efficiently handle large amounts of frequently changing data. The need for the Serial DataFlash arose from frustrated system designers who struggled for years trying to use large sectored Flash to store and manipulate data. With the Serial DataFlash's small page sizes, built-in internal RAM buffers, simple serial interface, and flexible software commands, users now have a Flash family that can meet their nonvolatile data storage requirements.

The Serial DataFlash series of devices is the first offering in the DataFlash product line. Larger density Serial DataFlash devices will be introduced in the future as the DataFlash product line continues to grow. In addition, new DataFlash architectures will be introduced that will have differing memory segmentations, new features, new command sets, and new interface methods. As the DataFlash products evolve, they will be tailored toward specific applications and market segments to provide the easiest and most useful devices for system designers.