Using Atmel's DataFlash®

DataFlash®

Application Note

(AN-4)

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 DataFlash is a feature-rich Flash family utilizing NOR technology designed specifically for data and code storage applications. Its small page sizes of 264 bytes for densities of 8M and smaller, 528 bytes for 16M and 32M, 1056 bytes for 64M, 128M and up to 2112 bytes for 256M and 512M, provide the system designer with a high level of flexibility and completely simplifies the process of data modifications. Furthermore, the DataFlash incorporates a simple serial interface which facilitates hardware layout, increases system reliability and minimizes switching noise. Densities of 64M and above support a dual interface. Designers can use the serial interface, the sequential access parallel interface (transporting 8 bits at a time for increased throughput) or both to optimize system performance and cost.

The DataFlash is perfectly suited for digital voice-, image-, code- and data-storage applications, especially where low power consumption is required. In these storage applications, the 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 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

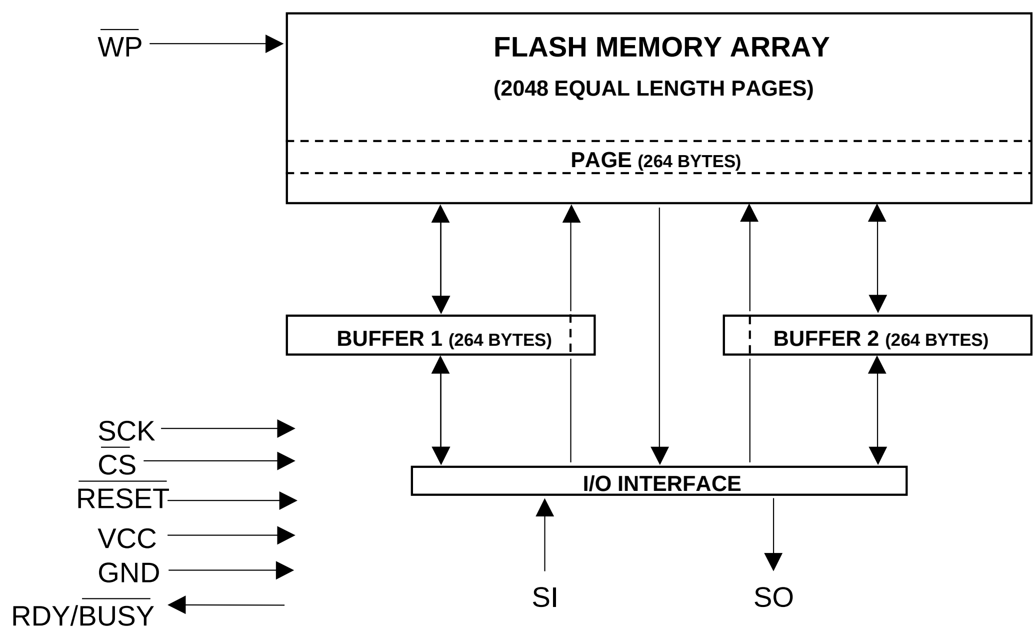
Rev. 0842D–DFLSH–11/02

- Code Storage Applications:
  – Portable test and measurement systems
  – Set top boxes
  – Hard disk drives
  – Network interface cards
  – Video graphics cards
  – Network hubs and routers
  – 802.11b wireless LAN
  – Home energy meters
  – DSL modems
  – FPGA Configuration

Functional Description

The block diagram of Atmel's 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 DataFlash uses a SPI compatible 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 AT45DB041B DataFlash Consists of a Flash Memory Array, Two Buffers and a Simple I/O Interface



DataFlash Array

The Flash memory array of the AT45DB041B, comprises 2048 equal length pages, 264 bytes each, 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 AT45DB041B 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 AT45DB041B 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 AT45DB041B 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 AT45DB041B 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 AT45DB041B'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 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 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 DataFlash include a chip select ($\overline{\text{CS}}$), chip reset input ($\overline{\text{RESET}}$), a write protect input ($\overline{\text{WP}}$), and a ready/busy output (RDY/$\overline{\text{BUSY}}$).

Functional Operation of the SPI

The 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 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 DataFlash.

Controlling Data Flow

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

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

Table 1. Features of the DataFlash Interface

Maximum Bus Speed	20 MHz
Number of Active Pins	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

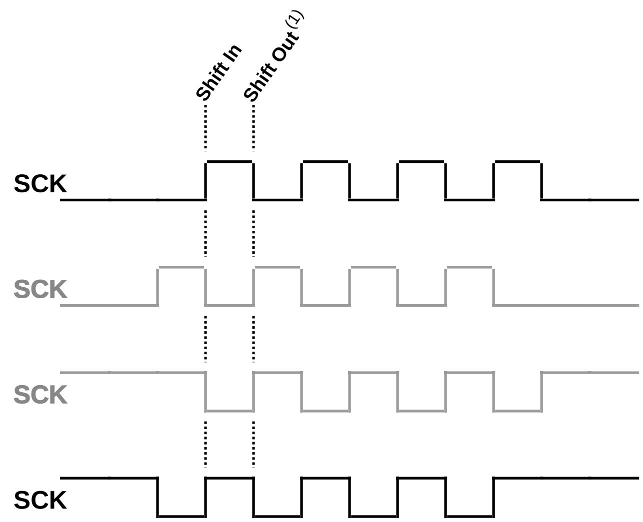
SPI Operating Modes

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 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 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 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 DataFlash is not limited to interfacing with SPI-compatible devices.

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: 1. 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.

Interfacing the 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 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).

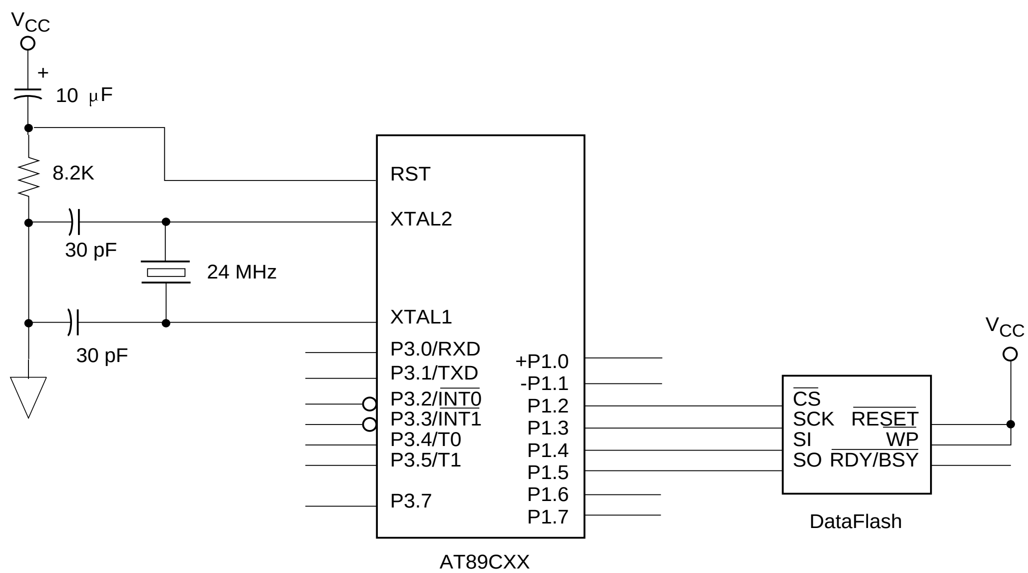
Table 2. SPI Control Register

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 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 DataFlash. As shown in Figure 3, Atmel's AT89CXX can be connected to the DataFlash. The microcontroller can clock data into the DataFlash at rates up to 20 MHz; whereas the limit on a strict SPI implementation is only 2.1 MHz.

The Reset Function

The 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 circuitry detects the supply voltage going below the minimum operating limit. Resetting the 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 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 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 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 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 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 or D7H) into the 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 2, while bits 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 and SCK is being toggled) 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 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 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 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 DataFlash being used. The three bits represent a code relating to different 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

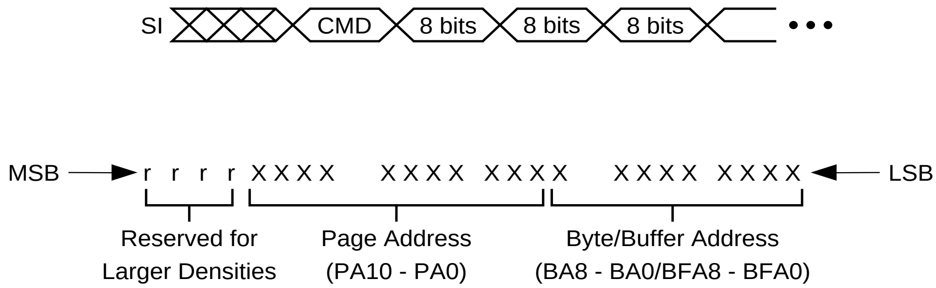
BIT 5	BIT 4	BIT 3	BIT 2	Device Density
0	0	1	1	1M
0	1	0	1	2M
0	1	1	1	4M
1	0	0	1	8M
1	0	1	1	16M
1	1	0	1	32M
1	1	1	1	64M
0	1	0	0	128M
0	1	1	0	256M
1	0	0	0	512M

Command Table

To begin an operation on the DataFlash, the system must send a command to the device. Table 6 on page 17 shows the bit sequence to follow for each DataFlash operation. All operations (except for Status Register Read) start with an opcode followed by three address bytes that are clocked into the DataFlash.

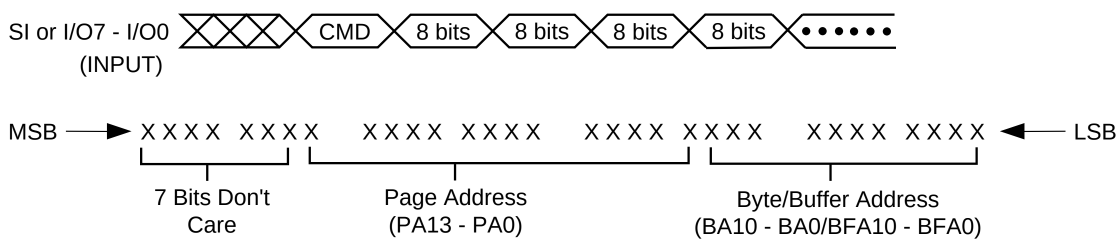
The three address bytes (24 bits) are used to address the memory array or buffers for the AT45DB041B. As shown in Figure 4, the four MSB bits are Reserved bits; bits 5-15 denote a page number; bits 16-24 denote a specific byte address within the 264-byte page or buffer. This 24-bit addressing scheme allows the system to address up to 64M bytes. The four Reserved bits of the AT45DB041B will be address bits for larger density devices and should be zero to ensure upwards compatibility.

Figure 4. Command Sequence for AT45DB041B Read/Write Operations (Except Status Register Read)



Notes:
1. "r" designates bits reserved for larger densities.
2. It is recommended that "r" be a logical "0" for densities of 4-Mbit or smaller.
3. For densities larger than 4-Mbit, the "r" bits become the most significant Page Address bit for the appropriate density.
4. Densities of 64M and below have three address bytes.

Figure 5. Command Sequence for Read/Write Operation of Densities 128M or Greater (Except Status Register Read)

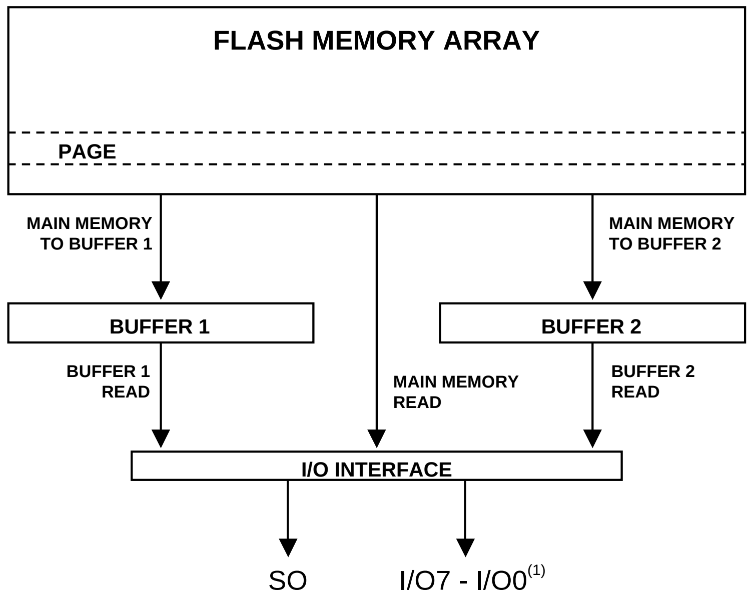


Notes:
1. For densities larger than 128M, the don't care bits become the most significant page address bit for the appropriate density.
2. The addresses shown in the example are for AT45DB1282. Densities of 128M and higher have four address bytes.

Read Operations for the AT45DB041B

By specifying the appropriate opcode, data can be read from the main memory or from either one of the two buffers (Figure 6). The DataFlash supports two categories of read modes in relation to the SCK signal. The differences between the modes are in respect to the inactive state of the SCK signal as well as which clock cycle data will begin to be output. The two categories, which are comprised of four modes total, are defined as Inactive Clock Polarity Low or Inactive Clock Polarity High and SPI Mode 0 or SPI Mode 3. A separate opcode (refer to Table 1 for a complete list) is used to select which category will be used for reading.

Figure 6. Data Can be Read Directly from Main Memory or through the Buffers



Note: 1. I/O7 - I/O0 interface is only offered in densities of 64M or higher.

CONTINUOUS ARRAY READ: By supplying an initial starting address for the main memory array, the Continuous Array Read command can be utilized to sequentially read a continuous stream of data from the device by simply providing a clock signal; no additional addressing information or control signals need to be provided. The DataFlash incorporates an internal address counter that will automatically increment on every clock cycle, allowing one continuous read operation without the need of additional address sequences. To perform a continuous read, an opcode of 68H or E8H must be clocked into the device followed by 24 address bits and 32 don't care bits. The first four bits of the 24-bit address sequence are reserved for upward and downward compatibility to larger and smaller density devices (see Notes under "Command Sequence for Read/Write Operations" diagram). The next 11 address bits (PA10 - PA0) specify which page of the main memory array to read, and the last nine bits (BA8 - BA0) of the 24-bit address sequence specify the starting byte address within the page. The 32 don't care bits that follow the 24 address bits are needed to initialize the read operation. Following the 32 don't care bits, additional clock pulses on the SCK pin will result in serial data being output on the SO (serial output) pin.

The $\overline{CS}$ pin must remain low during the loading of the opcode, the address bits, the don't care bits, and the reading of data. When the end of a page in main memory is reached during a Continuous Array Read, the device will continue reading at the beginning of the next page with no delays incurred during the page boundary crossover (the crossover from the end of one page to the beginning of the next page). When the last bit in the main memory array has been read, the device will continue reading back at the beginning of the first page of memory. As with crossing over page boundaries, no delays will be incurred when wrapping around from the end of the array to the beginning of the array.

A low-to-high transition on the $\overline{CS}$ pin will terminate the read operation and tri-state the SO pin. The maximum SCK frequency allowable for the Continuous Array Read is defined by the f_{CAR} specification. The Continuous Array Read bypasses both data buffers and leaves the contents of the buffers unchanged.

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, an opcode of 52H or D2H 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 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 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.

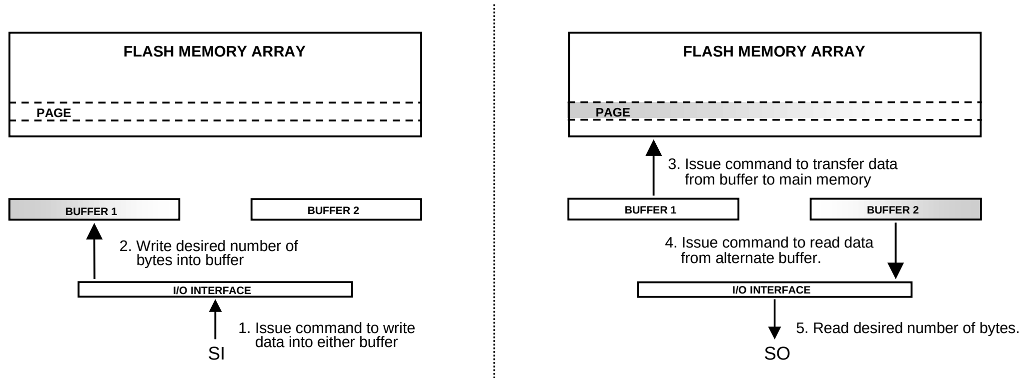
Buffer Read

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 or D4H for buffer 1, 56H or D6H 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 DataFlash's state machine time to initialize. While reading data from a buffer, if the end of the buffer is reached, the 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 DataFlash's state machine is transferring data from the other buffer into main memory (Figure 7).

Figure 7. DataFlash Supports Virtual Read-while-write Operations



Main Memory Page to Buffer Transfer

The 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. The 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 $\overline{\text{CS}}$ 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. The data transfer begins when there is a low-to-high transition on the $\overline{\text{CS}}$ pin. 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 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 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 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. The 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 $\overline{\text{CS}}$ 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. The compare operation begins when there is a low-to-high transition on the $\overline{\text{CS}}$ pin. 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 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 AT45DB041B

By specifying the appropriate opcode, data can be written to main memory or to either one of the two buffers (Figure 8).

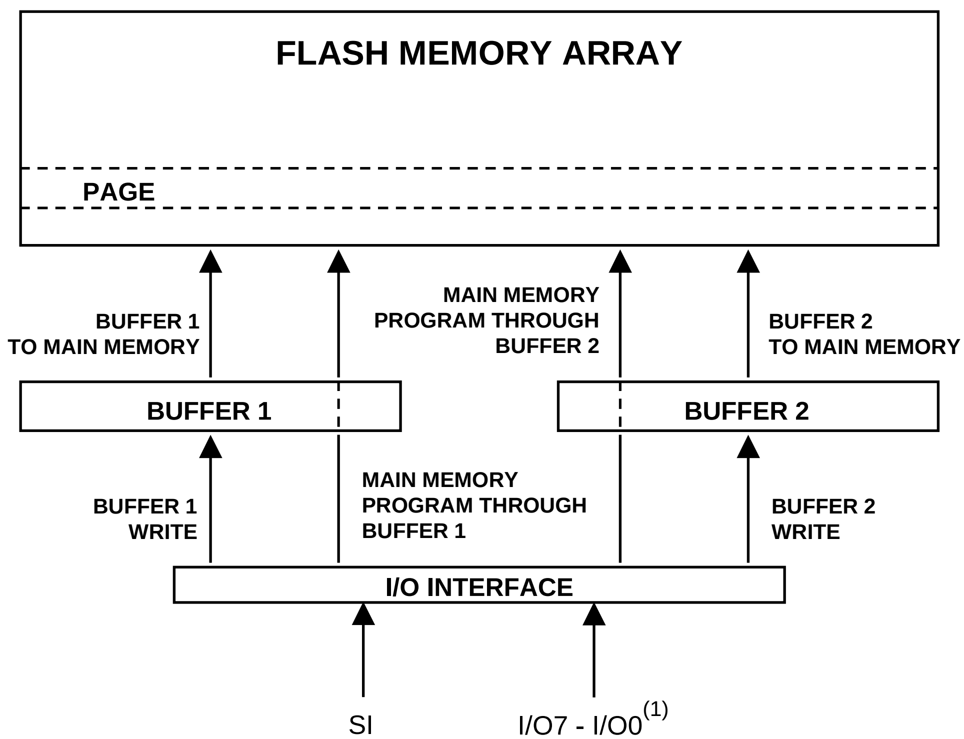
Buffer Write

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 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 8. DataFlash Write Operations



Note: 1. I/O7 - I/O0 interface is only offered in densities of 64M or higher.

Buffer to Main Memory Page Program with Built-In Erase

The 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 1s 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. The 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 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 1s).

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. The 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.

PAGE ERASE: The optional Page Erase command can be used to individually erase any page in the main memory array allowing the Buffer to Main Memory Page Program without Built-in Erase command to be utilized at a later time. To perform a Page Erase, an opcode of 81H must be loaded into the device, followed by four reserved bits, 11 address bits (PA10 - PA0), and nine don't care bits. The 11 address bits are used to specify which page of the memory array is to be erased. When a low-to-high transition occurs on the $\overline{CS}$ pin, the part will erase the selected page to 1s. The erase operation is internally self-timed and should take place in a maximum time of t_{PE}. During this time, the status register will indicate that the part is busy.

BLOCK ERASE: A block of eight pages can be erased at one time allowing the Buffer to Main Memory Page Program without Built-in Erase command to be utilized to reduce programming times when writing large amounts of data to the device. To perform a Block Erase, an opcode of 50H must be loaded into the device, followed by four reserved bits, eight address bits (PA10 - PA3), and 12 don't care bits. The eight address bits are used to specify which block of eight pages is to be erased. When a low-to-high transition occurs on the $\overline{CS}$ pin, the part will erase the selected block of eight pages to 1s. The erase operation is internally self-timed and should take place in a maximum time of t_{BE}. During this time, the status register will indicate that the part is busy.

Block Erase Addressing – AT45DB041B

PA10	PA9	PA8	PA7	PA6	PA5	PA4	PA3	PA2	PA1	PA0	Block
0	0	0	0	0	0	0	0	X	X	X	0
0	0	0	0	0	0	0	1	X	X	X	1
0	0	0	0	0	0	1	0	X	X	X	2
0	0	0	0	0	0	1	1	X	X	X	3
•	•	•	•	•	•	•	•	•	•	•	•
•	•	•	•	•	•	•	•	•	•	•	•
•	•	•	•	•	•	•	•	•	•	•	•
1	1	1	1	1	1	0	0	X	X	X	252
1	1	1	1	1	1	0	1	X	X	X	253
1	1	1	1	1	1	1	0	X	X	X	254
1	1	1	1	1	1	1	1	X	X	X	255

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 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 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 to all 1s and then program the data stored in the buffer to that page.

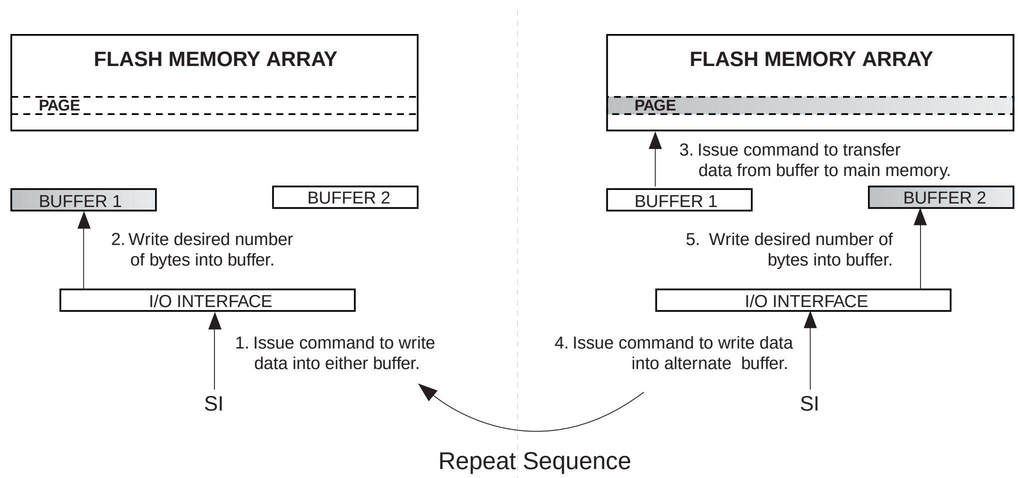
Once the main memory page erase/program has begun, you can use the RDY/$\overline{BUSY}$ pin or the RDY/$\overline{BUSY}$ bit in the Status Register to determine whether the state machine has completed the self-timed operation. Refer to the section on the Status Register for details on how to access 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 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 datasheet). Figure 9 shows the sequence of events that system software can use.

Figure 9. The Buffers Support a Virtually Continuous Write Operation



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

Table 6. Commands

Read Commands	SCK Mode	Opcode
Continuous Array Read	Inactive Clock Polarity Low or High	68H
	SPI Mode 0 or 3	E8H
Main Memory Page Read	Inactive Clock Polarity Low or High	52H
	SPI Mode 0 or 3	D2H
Buffer 1 Read	Inactive Clock Polarity Low or High	54H
	SPI Mode 0 or 3	D4H
Buffer 2 Read	Inactive Clock Polarity Low or High	56H
	SPI Mode 0 or 3	D6H
Status Register Read	Inactive Clock Polarity Low or High	57H
	SPI Mode 0 or 3	D7H
Programs and Erase Commands	**SCK Mode**	**Opcode**
Buffer 1 Write	Any(1)	84H
Buffer 2 Write	Any	87H
Buffer 1 to Main Memory Page Program with Built-in Erase	Any	83H
Buffer 2 to Main Memory Page Program with Built-in Erase	Any	86H
Buffer 1 to Main Memory Page Program without Built-in Erase	Any	88H
Buffer 2 to Main Memory Page Program without Built-in Erase	Any	89H
Page Erase	Any	81H
Block Erase	Any	50H
Main Memory Page Program through Buffer 1	Any	82H
Main Memory Page Program through Buffer 2	Any	85H
Additional Commands	**SCK Mode**	**Opcode**
Main Memory Page to Buffer 1 Transfer	Any	53H
Main Memory Page to Buffer 2 Transfer	Any	55H
Main Memory Page to Buffer 1 Compare	Any	60H
Main Memory Page to Buffer 2 Compare	Any	61H
Auto Page Rewrite through Buffer 1	Any	58H
Auto Page Rewrite through Buffer 2	Any	59H

Note: 1. "Any" denotes any one of the four modes of operation (Inactive Clock Polarity Low, Inactive Clock Polarity High, SPI Mode 0, or SPI Mode 3).

Table 7. Detailed Bit-level Addressing Sequence for the AT45DB041B (Three-byte Address)

Opcode	Opcode	Address Byte								Address Byte								Address Byte								Additional Don't Care Bytes Required
		Reserved	Reserved	Reserved	Reserved	PA10	PA9	PA8	PA7	PA6	PA5	PA4	PA3	PA2	PA1	PA0	BA8	BA7	BA6	BA5	BA4	BA3	BA2	BA1	BA0	
50H	0 1 0 1 0 0 0 0	r	r	r	r	**P**	**P**	**P**	**P**	**P**	**P**	**P**	**P**	x	x	x	x	x	x	x	x	x	x	x	x	N/A
52H	0 1 0 1 0 0 1 0	r	r	r	r	**P**	**P**	**P**	**P**	**P**	**P**	**P**	**P**	**P**	**P**	**P**	**B**	**B**	**B**	**B**	**B**	**B**	**B**	**B**	**B**	4 Bytes
53H	0 1 0 1 0 0 1 1	r	r	r	r	**P**	**P**	**P**	**P**	**P**	**P**	**P**	**P**	**P**	**P**	**P**	x	x	x	x	x	x	x	x	x	N/A
54H	0 1 0 1 0 1 0 0	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	**B**	**B**	**B**	**B**	**B**	**B**	**B**	**B**	**B**	1 Byte
55H	0 1 0 1 0 1 0 1	r	r	r	r	**P**	**P**	**P**	**P**	**P**	**P**	**P**	**P**	**P**	**P**	**P**	x	x	x	x	x	x	x	x	x	N/A
56H	0 1 0 1 0 1 1 0	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	**B**	**B**	**B**	**B**	**B**	**B**	**B**	**B**	**B**	1 Byte
57H	0 1 0 1 0 1 1 1	N/A								N/A								N/A								N/A
58H	0 1 0 1 1 0 0 0	r	r	r	r	**P**	**P**	**P**	**P**	**P**	**P**	**P**	**P**	**P**	**P**	**P**	x	x	x	x	x	x	x	x	x	N/A
59H	0 1 0 1 1 0 0 1	r	r	r	r	**P**	**P**	**P**	**P**	**P**	**P**	**P**	**P**	**P**	**P**	**P**	x	x	x	x	x	x	x	x	x	N/A
60H	0 1 1 0 0 0 0 0	r	r	r	r	**P**	**P**	**P**	**P**	**P**	**P**	**P**	**P**	**P**	**P**	**P**	x	x	x	x	x	x	x	x	x	N/A
61H	0 1 1 0 0 0 0 1	r	r	r	r	**P**	**P**	**P**	**P**	**P**	**P**	**P**	**P**	**P**	**P**	**P**	x	x	x	x	x	x	x	x	x	N/A
68H	0 1 1 0 1 0 0 0	r	r	r	r	**P**	**P**	**P**	**P**	**P**	**P**	**P**	**P**	**P**	**P**	**P**	**B**	**B**	**B**	**B**	**B**	**B**	**B**	**B**	**B**	4 Bytes
81H	1 0 0 0 0 0 0 1	r	r	r	r	**P**	**P**	**P**	**P**	**P**	**P**	**P**	**P**	**P**	**P**	**P**	x	x	x	x	x	x	x	x	x	N/A
82H	1 0 0 0 0 0 1 0	r	r	r	r	**P**	**P**	**P**	**P**	**P**	**P**	**P**	**P**	**P**	**P**	**P**	**B**	**B**	**B**	**B**	**B**	**B**	**B**	**B**	**B**	N/A
83H	1 0 0 0 0 0 1 1	r	r	r	r	**P**	**P**	**P**	**P**	**P**	**P**	**P**	**P**	**P**	**P**	**P**	x	x	x	x	x	x	x	x	x	N/A
84H	1 0 0 0 0 1 0 0	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	**B**	**B**	**B**	**B**	**B**	**B**	**B**	**B**	**B**	N/A
85H	1 0 0 0 0 1 0 1	r	r	r	r	**P**	**P**	**P**	**P**	**P**	**P**	**P**	**P**	**P**	**P**	**P**	**B**	**B**	**B**	**B**	**B**	**B**	**B**	**B**	**B**	N/A
86H	1 0 0 0 0 1 1 0	r	r	r	r	**P**	**P**	**P**	**P**	**P**	**P**	**P**	**P**	**P**	**P**	**P**	x	x	x	x	x	x	x	x	x	N/A
87H	1 0 0 0 0 1 1 1	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	**B**	**B**	**B**	**B**	**B**	**B**	**B**	**B**	**B**	N/A
88H	1 0 0 0 1 0 0 0	r	r	r	r	**P**	**P**	**P**	**P**	**P**	**P**	**P**	**P**	**P**	**P**	**P**	x	x	x	x	x	x	x	x	x	N/A
89H	1 0 0 0 1 0 0 1	r	r	r	r	**P**	**P**	**P**	**P**	**P**	**P**	**P**	**P**	**P**	**P**	**P**	x	x	x	x	x	x	x	x	x	N/A
D2H	1 1 0 1 0 0 1 0	r	r	r	r	**P**	**P**	**P**	**P**	**P**	**P**	**P**	**P**	**P**	**P**	**P**	**B**	**B**	**B**	**B**	**B**	**B**	**B**	**B**	**B**	4 Bytes
D4H	1 1 0 1 0 1 0 0	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	**B**	**B**	**B**	**B**	**B**	**B**	**B**	**B**	**B**	1 Byte
D6H	1 1 0 1 0 1 1 0	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	**B**	**B**	**B**	**B**	**B**	**B**	**B**	**B**	**B**	1 Byte
D7H	1 1 0 1 0 1 1 1	N/A								N/A								N/A								N/A
E8H	1 1 1 0 1 0 0 0	r	r	r	r	**P**	**P**	**P**	**P**	**P**	**P**	**P**	**P**	**P**	**P**	**P**	**B**	**B**	**B**	**B**	**B**	**B**	**B**	**B**	**B**	4 Bytes

Note: 1. r = Reserved Bit
P = Page Address Bit
B = Byte/Buffer Address Bit
x = Don't Care

Table 8. Detailed Bit-level Addressing Sequence for the AT45DB1282 (Four-byte Address)

Opcode	Opcode	Address Byte								Address Byte								Address Byte								Address Byte								Additional Don't Care Bytes*
									PA13	PA12	PA11	PA10	PA9	PA8	PA7	PA6	PA5	PA4	PA3	PA2	PA1	PA0	BA10	BA9	BA8	BA7	BA6	BA5	BA4	BA3	BA2	BA1	BA0	
50h	0 1 0 1 0 0 0 0	x	x	x	x	x	x	x	P	P	P	P	P	P	P	P	P	P	P	x	x	x	x	x	x	x	x	x	x	x	x	x	x	N/A
53h	0 1 0 1 0 0 1 1	x	x	x	x	x	x	x	P	P	P	P	P	P	P	P	P	P	P	P	P	P	x	x	x	x	x	x	x	x	x	x	x	N/A
54h	0 1 0 1 0 1 0 0	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	B	B	B	B	B	B	B	B	B	B	B	2*
55h	0 1 0 1 0 1 0 1	x	x	x	x	x	x	x	P	P	P	P	P	P	P	P	P	P	P	P	P	P	x	x	x	x	x	x	x	x	x	x	x	N/A
56h	0 1 0 1 0 1 1 0	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	B	B	B	B	B	B	B	B	B	B	B	2*
60h	0 1 1 0 0 0 0 0	x	x	x	x	x	x	x	P	P	P	P	P	P	P	P	P	P	P	P	P	P	x	x	x	x	x	x	x	x	x	x	x	N/A
61h	0 1 1 0 0 0 0 1	x	x	x	x	x	x	x	P	P	P	P	P	P	P	P	P	P	P	P	P	P	x	x	x	x	x	x	x	x	x	x	x	N/A
81h	1 0 0 0 0 0 0 1	x	x	x	x	x	x	x	P	P	P	P	P	P	P	P	P	P	P	P	P	P	x	x	x	x	x	x	x	x	x	x	x	N/A
84h	1 0 0 0 0 1 0 0	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	B	B	B	B	B	B	B	B	B	B	B	N/A
87h	1 0 0 0 0 1 1 1	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	B	B	B	B	B	B	B	B	B	B	B	N/A
88h	1 0 0 0 1 0 0 0	x	x	x	x	x	x	x	P	P	P	P	P	P	P	P	P	P	P	P	P	P	x	x	x	x	x	x	x	x	x	x	x	N/A
89h	1 0 0 1 1 0 0 1	x	x	x	x	x	x	x	P	P	P	P	P	P	P	P	P	P	P	P	P	P	x	x	x	x	x	x	x	x	x	x	x	N/A
98h	1 0 0 1 1 0 0 0	x	x	x	x	x	x	x	P	P	P	P	P	P	P	P	P	P	P	P	P	P	x	x	x	x	x	x	x	x	x	x	x	N/A
99h	1 0 0 1 1 0 0 1	x	x	x	x	x	x	x	P	P	P	P	P	P	P	P	P	P	P	P	P	P	x	x	x	x	x	x	x	x	x	x	x	N/A
D2h	1 1 0 1 0 0 1 0	x	x	x	x	x	x	x	P	P	P	P	P	P	P	P	P	P	P	P	P	P	B	B	B	B	B	B	B	B	B	B	B	3 or 19*
D4h	1 1 0 1 0 1 0 0	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	B	B	B	B	B	B	B	B	B	B	B	1
D6h	1 1 0 1 0 1 1 0	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	B	B	B	B	B	B	B	B	B	B	B	1
D7h	1 1 0 1 0 1 1 1	N/A								N/A								N/A								N/A								I/O or 1*
E8h	1 1 1 0 1 0 0 0	x	x	x	x	x	x	x	P	P	P	P	P	P	P	P	P	P	P	P	P	P	B	B	B	B	B	B	B	B	B	B	B	3 or 19*
E9h	1 1 1 0 1 0 0 1	x	x	x	x	x	x	x	P	P	P	P	P	P	P	P	P	P	P	P	P	P	B	B	B	B	B	B	B	B	B	B	B	3 or 19*

Notes: P = Page Address Bit
B = Byte/Buffer Address Bit
x = Don't Care
*First number is for serial interface, second is for parallel interface.

Auto Page Rewrite Command

The Auto Page Rewrite operation allows the 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. The 11 address bits are required to specify the page in main memory.

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. When a low-to-high transition occurs on the $\overline{CS}$ pin, the 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/$\overline{BUSY}$ 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.

Extended Reprogramming

To improve the reprogramming ability of the 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. Examples are described for sector 0b of 4M device, but the approach is applicable to any sector in the DataFlash family.

If the reprogram operations occur in a cyclical, sequential manner within the sector, 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 9) and continuing sequentially through the next 247 pages (e.g., pages10 - 255 and 8), and cycled again starting back at the original page (e.g., page 9), then no additional algorithms need to be incorporated into the system's microcontroller or microprocessor software.

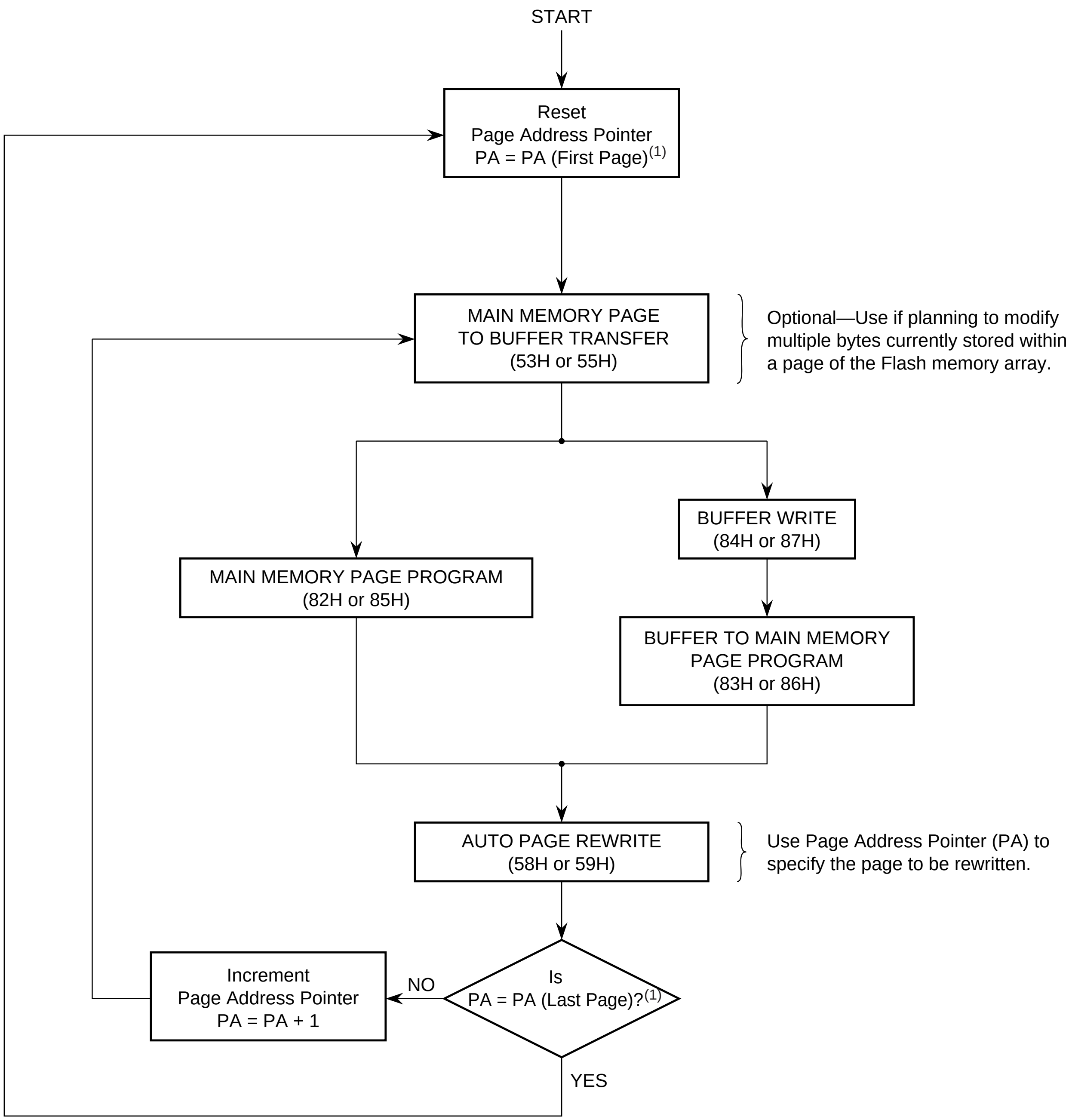
However, if the reprogram operations occur in a random fashion in which any number of pages is updated in a random order, then the system must ensure that each page of the DataFlash sector must be updated/refreshed at least once within every 10,000 cumulative page reprogram operations to other pages within the same sector. 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 8. When the system reprograms a page, say page 12, the system would then issue the Auto Page Rewrite command for page 8 after the completion of the page 12 erase/program operation (t_{EP}). The pointer would then be incremented to point to page 9. When the system reprograms another page, the process would be repeated. When the pointer reaches 255, it would be reset back to 8. Figure 10 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 8 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 8. Once the rewrite operation of page 8 is complete, the pointer would be incremented to point to page 9, and the counter would decrement, e.g., from 4 to 3. The system would continue by issuing the Auto Page Rewrite command for page 9, and the rewrite process would repeat until the counter decremented back to 0. When the pointer reaches 255, it would be reset back to 8. Figure 11 illustrates this example.

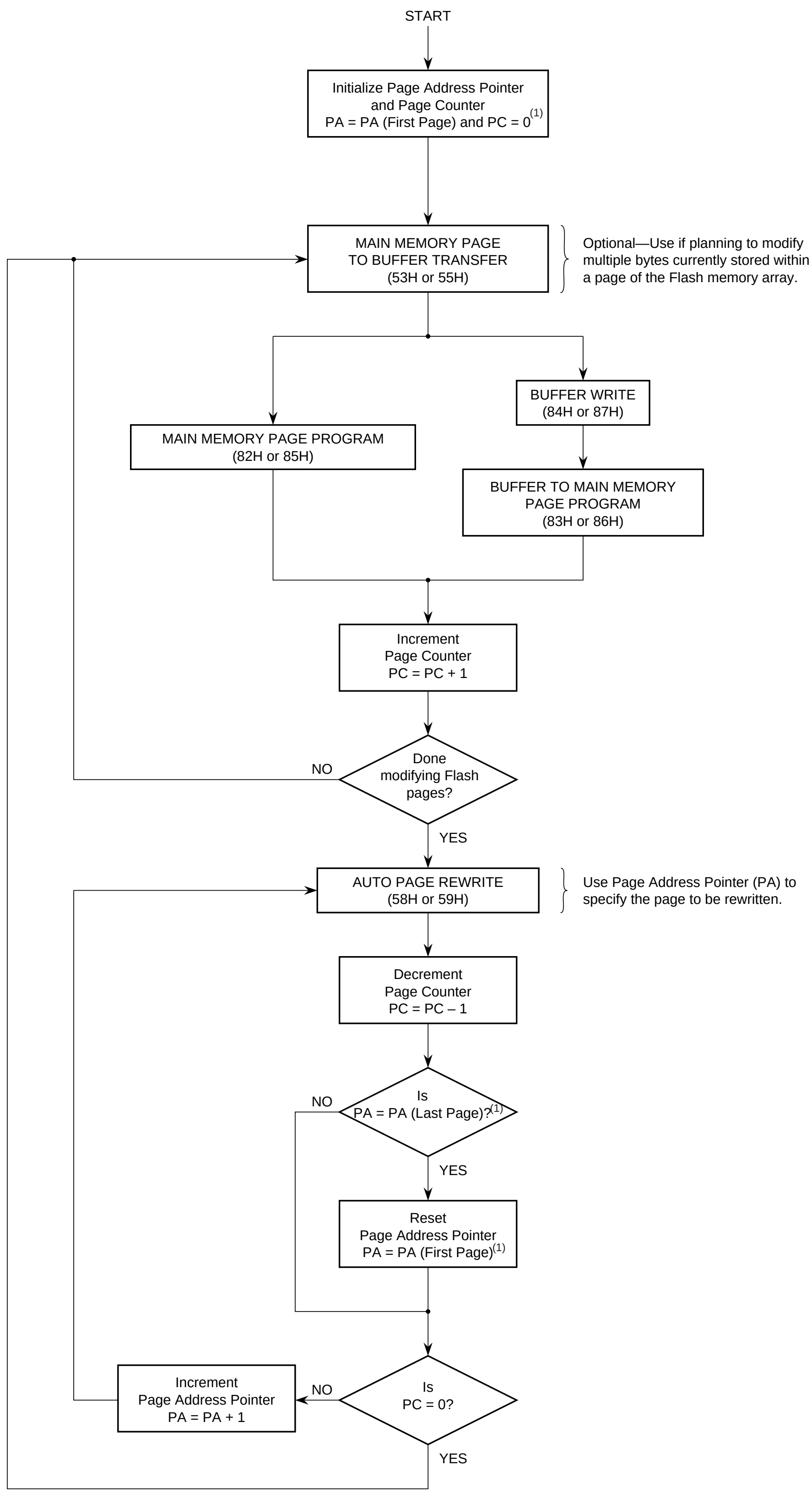
The final method for updating the Flash array would allow 10,000 reprogram operations within a sector to occur before the Auto Page Rewrite command needs to be issued. With this method, a software controlled counter needs to be implemented for each sector. The counter would initially be set to 0, and 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 8. Once the rewrite operation of page 8 is complete, the system would continue by rewriting pages 9, 10, 11, and so on until all 248 pages have been rewritten. After the entire Flash array has been rewritten, the counter would be reset back to 0. Figure 12 illustrates this example.

Figure 10. Extended Reprogramming – Single Page Rewrite



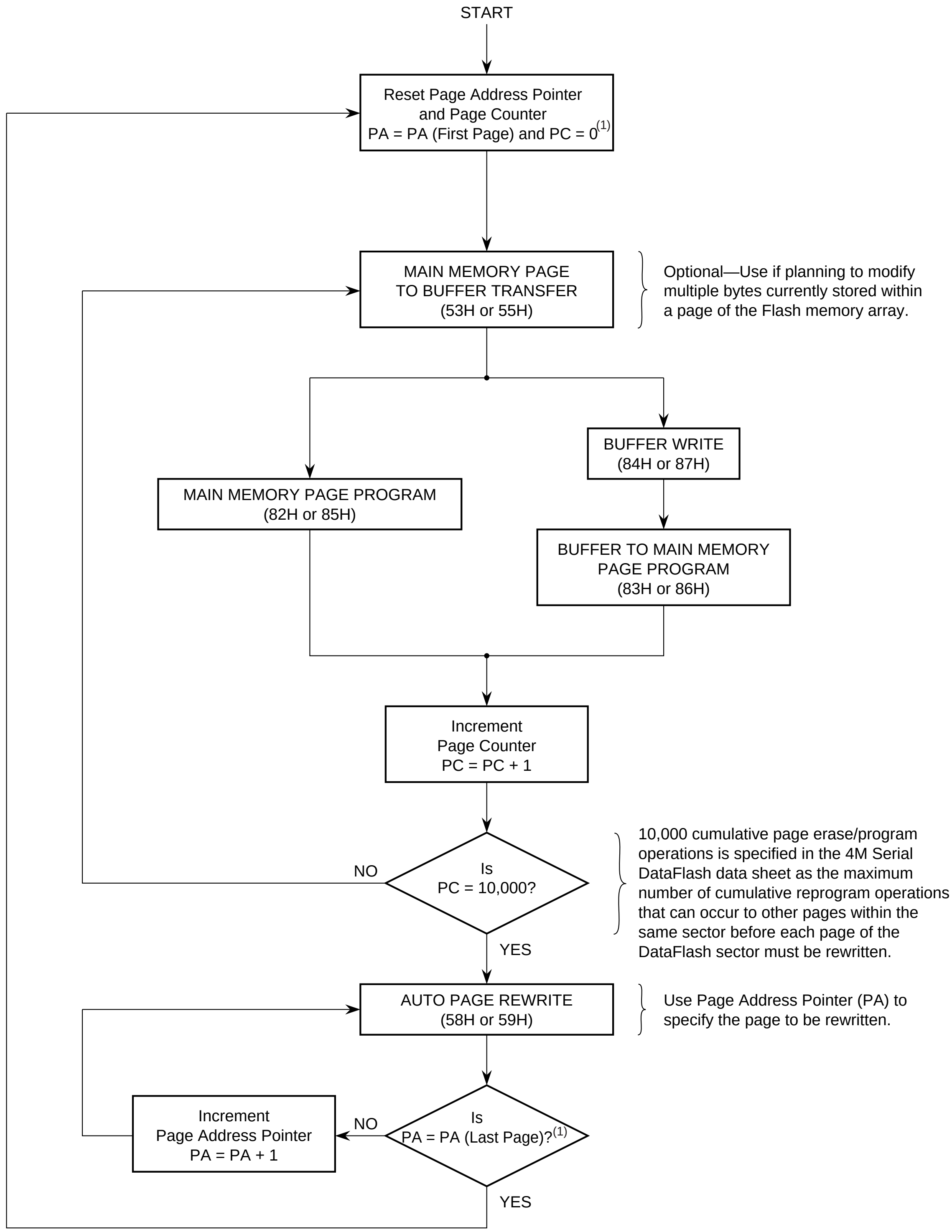
Note: 1. PA (First Page) and PA (Last Page) are page address for first and last page in the sector.

Figure 11. Extended Reprogramming – Multiple Page Rewrite



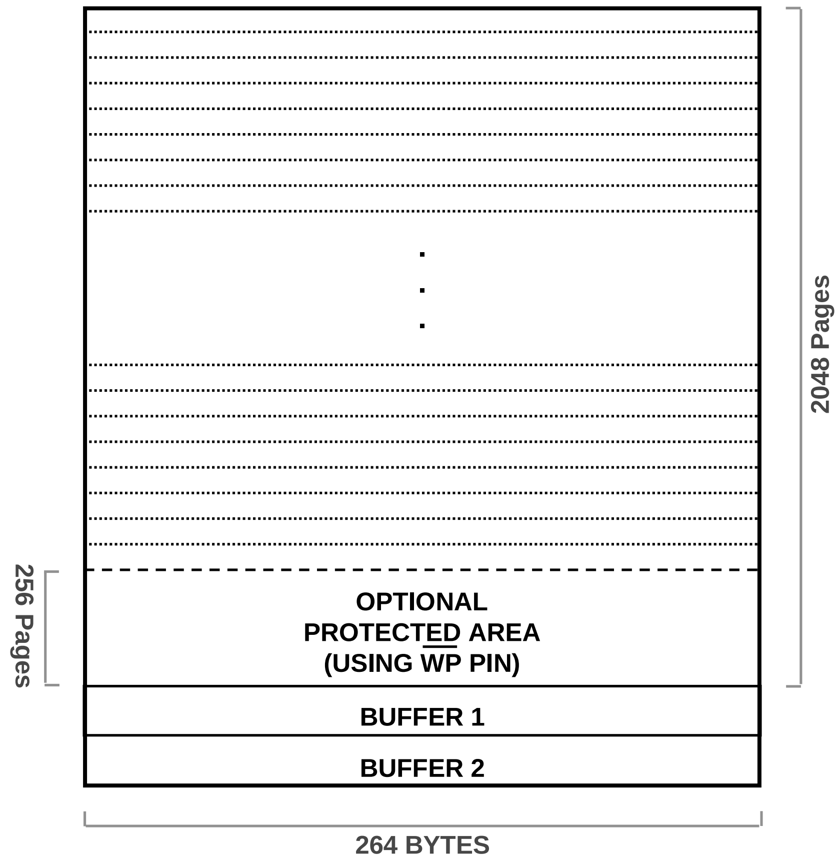
Note: 1. PA (First Page) and PA (Last Page) are page address for first and last page in the sector.

Figure 12. Extended Reprogramming – Sector Rewrite



Note: 1. PA (First Page) and PA (Last Page) are page address for first and last page in the sector.

Figure 13. The First 256 Pages of the 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 DataFlash can have data corruption problems due to glitches, noise spikes, bus contention, etc., which may initiate a false program or erase cycle. The DataFlash provides several mechanisms that can be used to prevent data corruption.

The Write Protect Signal

As shown in Figure 13, the $\overline{WP}$ pin of the 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 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 $\overline{\text{RESET}}$ Signal for Data Protection

The DataFlash's $\overline{\text{RESET}}$ pin can be connected to the system's reset line which will keep $\overline{\text{RESET}}$ held low and the 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 DataFlash also incorporates an internal power-on reset circuit; therefore, it is not required to hold $\overline{\text{RESET}}$ low during power-on sequences.

The $\overline{\text{RESET}}$ pin is level sensitive and can also be used to protect the entire memory array from inadvertent writes during power outages. The $\overline{\text{RESET}}$ 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.

Dual Interface

DataFlash devices of densities 64M, 128M, 256M offer a dual interface. They can be operated both in the serial and parallel mode.

The DataFlash device that has dual interface may be configured to utilize either its serial port or parallel port through the use of the serial/parallel control pin (SER/$\overline{\text{PAR}}$). When the SER/$\overline{\text{PAR}}$ pin is held high, the serial port (SI and SO) of the DataFlash will be used for all data transfers, and the parallel port (I/O7 - I/O0) will be in a high impedance state. Any data presented on the parallel port while SER/$\overline{\text{PAR}}$ is held high will be ignored. When the SER/$\overline{\text{PAR}}$ is held low, the parallel port will be used for all data transfers, and the SO pin of the serial port will be in a high impedance state. While SER/$\overline{\text{PAR}}$ is low, any data presented on the SI pin will be ignored.

Switching between the serial port and parallel port can be done at anytime, provided the following conditions are met:

1. $\overline{\text{CS}}$ should be held high during the switching between the two modes.
2. t_{SPH} (SER/$\overline{\text{PAR}}$ hold time) and t_{SPS} (SER/$\overline{\text{PAR}}$ setup time) requirements should be obeyed.

The SER/$\overline{\text{PAR}}$ pin is internally pulled high; therefore, if the parallel port is never to be used, then connection of the SER/$\overline{\text{PAR}}$ pin is not necessary. In addition, if the SER/$\overline{\text{PAR}}$ pin is not connected or if the SER/$\overline{\text{PAR}}$ pin is always driven high externally, then the parallel input/output pins (I/O7 - I/O0), the VCCP pin and the GNDP pin should be treated as "don't connects".

Having both a serial port and parallel port on the DataFlash allows the device to reside on two buses that can be connected to different processors. The advantage of switching between the serial and parallel ports is that while an internally self-timed operation such as an erase or program operation is started using either of the ports, a simultaneous operation such as a buffer read or buffer write can be started utilizing the other port.

Table 9. Detailed Bit-level Addressing Sequence of the AT45DB642 in the Parallel Mode

Opcode	Opcode	Address Byte								Address Byte								Address Byte								Additional Don't Care Bytes Required
		PA12	PA11	PA10	PA9	PA8	PA7	PA6	PA5	PA4	PA3	PA2	PA1	PA0	BA10	BA9	BA8	BA7	BA6	BA5	BA4	BA3	BA2	BA1	BA0	
50H	0 1 0 1 0 0 0 0	**P**	**P**	**P**	**P**	**P**	**P**	**P**	**P**	**P**	**P**	x	x	x	x	x	x	x	x	x	x	x	x	x	x	N/A
52H	0 1 0 1 0 0 1 0	**P**	**P**	**P**	**P**	**P**	**P**	**P**	**P**	**P**	**P**	**P**	**P**	**P**	**B**	**B**	**B**	**B**	**B**	**B**	**B**	**B**	**B**	**B**	**B**	60 Bytes
53H	0 1 0 1 0 0 1 1	**P**	**P**	**P**	**P**	**P**	**P**	**P**	**P**	**P**	**P**	**P**	**P**	**P**	x	x	x	x	x	x	x	x	x	x	x	N/A
54H	0 1 0 1 0 1 0 0	x	x	x	x	x	x	x	x	x	x	x	x	x	**B**	**B**	**B**	**B**	**B**	**B**	**B**	**B**	**B**	**B**	**B**	1 Byte
55H	0 1 0 1 0 1 0 1	**P**	**P**	**P**	**P**	**P**	**P**	**P**	**P**	**P**	**P**	**P**	**P**	**P**	x	x	x	x	x	x	x	x	x	x	x	N/A
56H	0 1 0 1 0 1 1 0	x	x	x	x	x	x	x	x	x	x	x	x	x	**B**	**B**	**B**	**B**	**B**	**B**	**B**	**B**	**B**	**B**	**B**	1 Byte
57H	0 1 0 1 0 1 1 1	N/A								N/A								N/A								N/A
58H	0 1 0 1 1 0 0 0	**P**	**P**	**P**	**P**	**P**	**P**	**P**	**P**	**P**	**P**	**P**	**P**	**P**	x	x	x	x	x	x	x	x	x	x	x	N/A
59H	0 1 0 1 1 0 0 1	**P**	**P**	**P**	**P**	**P**	**P**	**P**	**P**	**P**	**P**	**P**	**P**	**P**	x	x	x	x	x	x	x	x	x	x	x	N/A
60H	0 1 1 0 0 0 0 0	**P**	**P**	**P**	**P**	**P**	**P**	**P**	**P**	**P**	**P**	**P**	**P**	**P**	x	x	x	x	x	x	x	x	x	x	x	N/A
61H	0 1 1 0 0 0 0 1	**P**	**P**	**P**	**P**	**P**	**P**	**P**	**P**	**P**	**P**	**P**	**P**	**P**	x	x	x	x	x	x	x	x	x	x	x	N/A
68H	0 1 1 0 1 0 0 0	**P**	**P**	**P**	**P**	**P**	**P**	**P**	**P**	**P**	**P**	**P**	**P**	**P**	**B**	**B**	**B**	**B**	**B**	**B**	**B**	**B**	**B**	**B**	**B**	60 Bytes
69H	0 1 1 0 1 0 0 1	**P**	**P**	**P**	**P**	**P**	**P**	**P**	**P**	**P**	**P**	**P**	**P**	**P**	**B**	**B**	**B**	**B**	**B**	**B**	**B**	**B**	**B**	**B**	**B**	60 Bytes
81H	1 0 0 0 0 0 0 1	**P**	**P**	**P**	**P**	**P**	**P**	**P**	**P**	**P**	**P**	**P**	**P**	**P**	x	x	x	x	x	x	x	x	x	x	x	N/A
82H	1 0 0 0 0 0 1 0	**P**	**P**	**P**	**P**	**P**	**P**	**P**	**P**	**P**	**P**	**P**	**P**	**P**	**B**	**B**	**B**	**B**	**B**	**B**	**B**	**B**	**B**	**B**	**B**	N/A
83H	1 0 0 0 0 0 1 1	**P**	**P**	**P**	**P**	**P**	**P**	**P**	**P**	**P**	**P**	**P**	**P**	**P**	x	x	x	x	x	x	x	x	x	x	x	N/A
84H	1 0 0 0 0 1 0 0	x	x	x	x	x	x	x	x	x	x	x	x	x	**B**	**B**	**B**	**B**	**B**	**B**	**B**	**B**	**B**	**B**	**B**	N/A
85H	1 0 0 0 0 1 0 1	**P**	**P**	**P**	**P**	**P**	**P**	**P**	**P**	**P**	**P**	**P**	**P**	**P**	**B**	**B**	**B**	**B**	**B**	**B**	**B**	**B**	**B**	**B**	**B**	N/A
86H	1 0 0 0 0 1 1 0	**P**	**P**	**P**	**P**	**P**	**P**	**P**	**P**	**P**	**P**	**P**	**P**	**P**	x	x	x	x	x	x	x	x	x	x	x	N/A
87H	1 0 0 0 0 1 1 1	x	x	x	x	x	x	x	x	x	x	x	x	x	**B**	**B**	**B**	**B**	**B**	**B**	**B**	**B**	**B**	**B**	**B**	N/A
88H	1 0 0 0 1 0 0 0	**P**	**P**	**P**	**P**	**P**	**P**	**P**	**P**	**P**	**P**	**P**	**P**	**P**	x	x	x	x	x	x	x	x	x	x	x	N/A
89H	1 0 0 0 1 0 0 1	**P**	**P**	**P**	**P**	**P**	**P**	**P**	**P**	**P**	**P**	**P**	**P**	**P**	x	x	x	x	x	x	x	x	x	x	x	N/A
D2H	1 1 0 1 0 0 1 0	**P**	**P**	**P**	**P**	**P**	**P**	**P**	**P**	**P**	**P**	**P**	**P**	**P**	**B**	**B**	**B**	**B**	**B**	**B**	**B**	**B**	**B**	**B**	**B**	60 Bytes
D4H	1 1 0 1 0 1 0 0	x	x	x	x	x	x	x	x	x	x	x	x	x	**B**	**B**	**B**	**B**	**B**	**B**	**B**	**B**	**B**	**B**	**B**	1 Byte
D6H	1 1 0 1 0 1 1 0	x	x	x	x	x	x	x	x	x	x	x	x	x	**B**	**B**	**B**	**B**	**B**	**B**	**B**	**B**	**B**	**B**	**B**	1 Byte
D7H	1 1 0 1 0 1 1 1	N/A								N/A								N/A								N/A
E8H	1 1 1 0 1 0 0 0	**P**	**P**	**P**	**P**	**P**	**P**	**P**	**P**	**P**	**P**	**P**	**P**	**P**	**B**	**B**	**B**	**B**	**B**	**B**	**B**	**B**	**B**	**B**	**B**	60 Bytes
E9H	1 1 1 0 1 0 0 1	**P**	**P**	**P**	**P**	**P**	**P**	**P**	**P**	**P**	**P**	**P**	**P**	**P**	**B**	**B**	**B**	**B**	**B**	**B**	**B**	**B**	**B**	**B**	**B**	60 Bytes

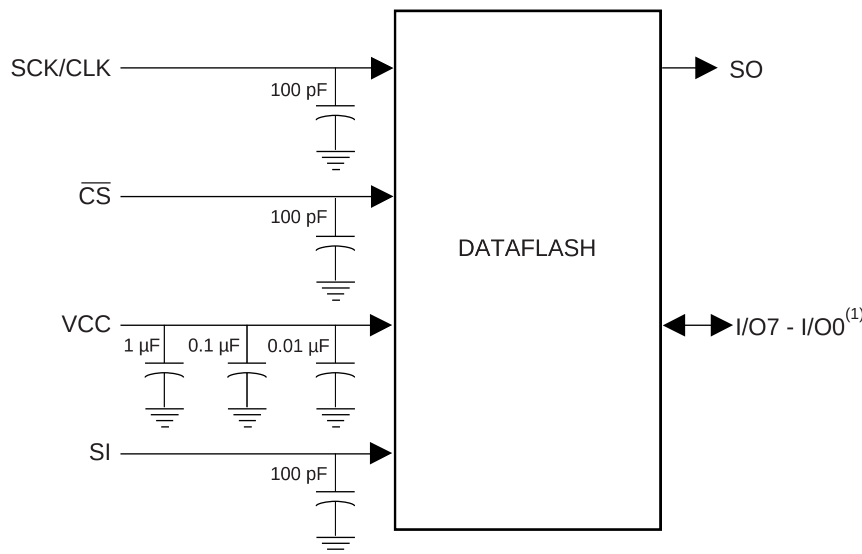
Note: P = Page Address Bit
B = Byte/Buffer Address Bit
x = Don't Care

System Considerations

DataFlash is controlled by the clock SCK/CLK and chip select $\overline{CS}$ pins. These signals must rise and fall monotonically and be free from noise. Excessive noise or ringing on these pins can be misinterpreted as multiple edges and cause improper operation of the device. The PC board traces must be kept to a minimum distance or appropriately terminated to ensure proper operation. If necessary, decoupling capacitors can be added on these pins to provide filtering against noise glitches.

Adding a capacitor of 100 pF on the SCK/CLK, $\overline{CS}$ and SI pins can minimize noise present on the SCK/CLK, $\overline{CS}$ and SI signals. To filter the different levels of noise present on the V_{CC} signal, capacitors of values 1 µF, 0.1 µF and 0.01 µF can be used in parallel. The user can choose a different value of the capacitor to be added on the SCK/CLK, $\overline{CS}$, SI, V_{CC} signals depending on the system requirement. If there is no noise present on the SCK/CLK, V_{CC}, $\overline{CS}$ and SI signals, the DataFlash will function correctly.

Figure 14. System Considerations



Note: 1. I/O7 - I/O0 interface is only offered in densities of 64M or higher.

Table 10. Recommended Capacitance Values on the SCK/CLK, $\overline{CS}$, VCC and SI Pins

Signal	Capacitance
SCK/CLK	100 pF
$\overline{CS}$	100 pF
SI	100 pF
V_{CC}	1 µF + 0.1 µF +0.01 µF (To be Connected in Parallel)

As system complexity continues to increase, voltage regulation is becoming more important. A key element of any voltage regulation scheme is its current sourcing capability. Like all Flash memories, the peak currents of DataFlash occur during the programming and erase operations. The peak current during programming or erase of a DataFlash can be 70 mA to 80 mA. The regulator needs to supply this peak current requirement. An under specified regulator can cause current starvation. Besides increasing system noise, current starvation during programming or erase can lead to improper operation and possible data corruption.

Summary

The 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 DataFlash arose from frustrated system designers who struggled for years trying to use large sectored Flash to store and manipulate data. With the 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 entire nonvolatile data and code storage requirements.