# Contents

1.0 Introduction .................................................................................................................. 1

2.0 Benefits of CFI ............................................................................................................. 3
   2.1 Upgrades ............................................................................................................... 3
   2.2 Second Sources .................................................................................................... 3

3.0 How to Use CFI Effectively ..................................................................................... 4
   3.1 Read Query String ................................................................................................. 4
   3.2 Read Electronic Databook Information .................................................................. 6
   3.3 Read Vendor-Specific Extended Query Table ...................................................... 9
   3.4 Software Branch to Appropriate Routines ........................................................... 11

4.0 How to Use Command Sets Effectively .................................................................... 11

5.0 Conclusion .................................................................................................................. 11

Appendix A  Memory Hueristics ................................................................................................ 12
   A1 Memory Hueristics Flowchart .............................................................................. 12
   A2 CFI Query Flowchart ........................................................................................... 13
   A3 Read JEDEC ID Flowchart .................................................................................. 14

Appendix B  Basic Command Set ..................................................................................... 15
   B1 Clear Status Register Flowchart .......................................................................... 15
   B2 Read Array Flowchart .......................................................................................... 15
   B3 Block Erase Flowchart ......................................................................................... 16
   B4 Erase Suspend/Resume Flowchart ....................................................................... 17
   B5 Single Byte Program (Write) Flowchart ............................................................... 17
   B6 Full Status Check Flowchart ................................................................................ 18

Appendix C  Scaleable Command Set ............................................................................... 19
   C1 Write to Buffer Flowchart .................................................................................... 19
   C2 Program (Write) Suspend/Resume Flowchart ..................................................... 20
   C3 Block Lock Bit Set Flowchart ............................................................................... 20
   C4 Block Lock Bit Reset Flowchart .......................................................................... 21
## Revision History

<table>
<thead>
<tr>
<th>Date of Revision</th>
<th>Number</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-001</td>
<td>-001</td>
<td>Original version</td>
</tr>
<tr>
<td>06/01/97</td>
<td>-002</td>
<td>Added ANSI 'C' code to Appendix A</td>
</tr>
<tr>
<td>12/01/97</td>
<td>-003</td>
<td>Removed routine codes from the Appendices</td>
</tr>
<tr>
<td>04/01/00</td>
<td>-004</td>
<td>Reformatted document</td>
</tr>
</tbody>
</table>
1.0 Introduction

This application note defines Common Flash Interface (CFI), Basic Command Set (BCS), and Scaleable Command Set (SCS), as well as discusses their benefits and details how best to use them.

Common Flash Interface (CFI) is a published, standardized data structure that may be read from a flash memory device. CFI allows system software to query the installed device (on board component, PC [PCMCIA] Card, or Miniature Card) to determine configurations, various electrical and timing parameters, and functions supported by the device.

The Basic Command Set (BCS) is a group of commands that have been used for years on Intel’s and other vendors’ legacy products. This command set is also commonly referred to as the 28F008, or simply the 008 command set. These commands include Read Array, Read ID, Read Status Register, Clear Status Register, Program (Write), Block Erase, Erase Suspend, and Confirm/Resume. The BCS is the “Standard Command Set” used by Intel in its CFI implementations.

Scaleable Command Set (SCS) is the “Extended Command Set” that Intel uses to control the functions of most CFI-enabled flash devices. CFI allows the vendor to specify a command set that should be used with the component. SCS is the command set that will be used by Intel on most of its CFI enabled devices. SCS includes all commands available in the BCS, as well as some new advanced commands that have been designed to take advantage of Intel’s next generation optimized flash devices. These new commands include Set and Clear Lock Bits, CFI Query, Write to Buffer, Program Suspend, Status Configuration, and Full Chip Erase. With many new capabilities being designed into flash products today, these new commands were necessary to take full advantage of the improvements.

CFI is used to allow the system to learn how to interface to the flash device most optimally. The BCS and SCS are used to then command the device to perform the desired flash functions.

Figure 1. How CFI, SCS, and BCS Fit Together
Figure 2. CFI Allows Easy Upgrades and Use of Second Sources

Compatibility among removable media

Silicon updates without software changes

Removable Media

Embedded Products
2.0 Benefits of CFI

The two primary benefits of using CFI are ease of upgrading and second source availability. Both are concerns when an OEM or end-user (the consumer) purchases a product.

2.1 Upgrades

In order to take advantage of increased densities (or speeds, etc.) on memory devices and cards, an easy upgrade path is desirable. Care is generally taken to ensure that hardware footprints are pin-for-pin compatible or that flexible layouts may be used when upgrading a product. However, thought is seldom given to software compatibility. CFI allows many new and improved products to be used in place of their older versions without modifications of system software.

Because CFI allows the system to learn about the features, parameters, and timings of a flash device, the system can take full advantage of these improvements. For instance, if the timeout for a block erase to occur was cut in half, the system software could take advantage of that fact by changing its internal timers. Also, a 32-Mbit device can be replaced by a 64-Mbit device and vice versa because the device can tell the system what size it is.

With CFI, when upgrading a flash memory design, it is no longer necessary to re-optimize low level software drivers to take advantage of the new features. Simply program the system initially to accept CFI enabled devices, and allow the software to upgrade itself.

2.2 Second Sources

Particularly in the card environment, second sourcing is a primary concern. Because the end-user of a PC Card or Miniature Card could be a consumer, care must be taken to ensure compatibility among all flash cards that may be installed into the same sockets. For instance, when purchasing a replacement or spare memory card for their digital camera, consumers do not want to have to worry that they can only purchase a certain vendor’s card or a particular version of what seems to be a similar card.

This is analogous to a consumer purchasing floppy disks for their computer or film for their camera. Any vendor’s product works. That is the goal of CFI—complete and simple interchange between vendors in a card application. The hardware inside the PC Card or Miniature Card does not have to operate identically; the software takes care of the differences as long as the devices are CFI-compliant.

CFI allows the system to determine the manufacturer of the card, its operating parameters, its configuration, and any special command codes that the card may accept. With this knowledge, the system can optimize its use of the card by using appropriate timeout values, optimal voltages, and commands necessary to use the card to its full advantage.
3.0 How to Use CFI Effectively

To use CFI effectively, system software must be written to take advantage of the flexibility provided by the specification. The software must be capable of modifying timeouts, adjusting to different memory sizes, accommodating varying block erase characteristics, and branching to vendor-specific code sections. The following paragraphs outline several steps which system software must transition through to read a CFI-enabled device. Flowcharts are included in Appendix A.

3.1 Read Query String

Not all devices installed into a flash memory socket will be CFI-enabled. To determine if a device is CFI-capable, the system software must write a 98h to location 55h within the memory (see CFI Query Flowchart included in Appendix A). The flash device may or may not have an address sensitive query command; the Intel devices do not. The low-level driver, however, should supply the 55h address even though the flash device may choose to ignore the address bus and enter the query mode if 98h is on the data bus only. If three consecutive maximum device bus width reads beginning at location 10h in the flash array return the ASCII equivalent “Q,” “R,” and “Y,” then the device is CFI-compliant.

Although there are other configuration possibilities, there are currently three CFI array configurations that are of primary interest. These configurations must be tested and accounted for in the software. These configurations are:

- single chip operating in a x16 mode (16-bit data bus)—chips may be capable of 8-bit accesses, but are operating only with 16-bit bus accesses
- two chips each capable of 8- and 16-bit data bus accesses, but each only operating in a x8 mode (8-bit data bus on each chip with a total array bus width of 16-bits)
- two chips each only capable of 8-bit data bus accesses operating only in a x8 mode (8-bit data bus per chip with a total array bus width of 16 bits)

Each of these configurations are shown in Figure 3, “Possible Flash Array Configurations” on page 5. Table 1, “CFI Query Read” on page 6 indicates the addressing necessary to read the CFI query table for each of these configurations, along with some other possible device configurations. The table also includes what the query data will look like to the host processor in byte or word addressing. Note that the query data (ASCII “Q,” “R,” and “Y,” as well as the electronic databook information discussed in the next section) may be doubled or even quadrupled depending on the array configuration. The software must be able to determine the correct array configuration based on the number of “Q”s returned to accurately calculate the array size and read and write to the array properly. The CFI Query Flowchart is located in Appendix A. The QueryCFI routine heuristically determines the configuration of the array, calculates the appropriate data, and indicates to the higher level routines how to communicate with the CFI-enabled devices.

If the device does not respond with the “QRY” string, the device is not CFI-compliant and the software must then attempt to read the device’s JEDEC ID. (See the Memory Heuristics Flowchart included in Appendix A.) The software must write a 90h to the first location in the device. If the device returns a Manufacturer’s ID and Component ID, the flash device may be accessed as it has been in the past, based on the Manufacturer and Component ID. If the device does not return a Manufacturer and Component ID, then the device is not a flash memory and other routines are necessary to determine what type of device is installed. (See the Memory Heuristics Flowchart included in Appendix A.)
Figure 3. Possible Flash Array Configurations

Chips may be x16 only or x8/x16 capable

Single x16 device (x16 capable device operating in a 16-bit mode)

Paired x8/x16 devices (two x8/x16 capable devices operating in an 8-bit mode)

Paired x8 devices (two x8 only devices operating in an 8-bit mode)

Flash Array

x16 or x8/x16 capable device

16-bit data bus

Lower 8 bits of 16-bit data bus

Upper 8 bits of 16-bit data bus

Flash Array

x8/x16 capable device

x8/x16 capable device

Flash Array

x8 only capable device

x8 only capable device

Lower 8 bits of 16-bit data bus

Upper 8 bits of 16-bit data bus
3.2 Read Electronic Databook Information

In a CFI-enabled device, following the “QRY” string is a list of device specific parameters and vendor-specific information—the “Electronic Databook.” Table 2, “CFI Query Identification String” on page 7, Table 3, “System Interface Information” on page 8, and Table 4, “Flash Geometry Information” on page 9 outline the data provided by the device during the CFI query. Software routine QueryCFI (the flowchart is included in Appendix A) reads the following information from the device:

Table 1. CFI Query Read

<table>
<thead>
<tr>
<th>Device Type and Data Bus Operating Mode</th>
<th>Query Start Location in Maximum Device Buswidth Addresses</th>
<th>Query Data with Maximum Device Buswidth Addressing (&quot;x&quot; = ASCII equivalent)</th>
<th>Query Start Address in Bytes</th>
<th>Query Data With Byte Addressing</th>
</tr>
</thead>
<tbody>
<tr>
<td>x8 device operating in 8-bit mode</td>
<td>10h</td>
<td>10h: 51h &quot;Q&quot;</td>
<td>10h</td>
<td>10h: 51h &quot;Q&quot;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>11h: 52h &quot;R&quot;</td>
<td></td>
<td>11h: 52h &quot;R&quot;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>12h: 59h &quot;Y&quot;</td>
<td></td>
<td>12h: 59h &quot;Y&quot;</td>
</tr>
<tr>
<td>two x8 devices operating in 8-bit mode</td>
<td>10h</td>
<td>10h: 0051h &quot;Q&quot;</td>
<td>20h</td>
<td>20h: 51h &quot;Q&quot;</td>
</tr>
<tr>
<td>(paired chip configuration)</td>
<td></td>
<td>11h: 0052h &quot;R&quot;</td>
<td></td>
<td>21h: 51h &quot;Q&quot;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>12h: 0059h &quot;Y&quot;</td>
<td></td>
<td>22h: 52h &quot;R&quot;</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>23h: 52h &quot;R&quot;</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>24h: 59h &quot;Y&quot;</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>25h: 59h &quot;Y&quot;</td>
</tr>
<tr>
<td>x16 device operating in 16-bit mode</td>
<td>10h</td>
<td>10h: 0051h &quot;Q&quot;</td>
<td>20h</td>
<td>20h: 51h &quot;Q&quot;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>11h: 0052h &quot;R&quot;</td>
<td></td>
<td>21h: 00h null</td>
</tr>
<tr>
<td></td>
<td></td>
<td>12h: 0059h &quot;Y&quot;</td>
<td></td>
<td>22h: 52h &quot;R&quot;</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>23h: 00h null</td>
</tr>
<tr>
<td>x16 device operating in 8-bit mode</td>
<td>N/A(1)</td>
<td></td>
<td>20h</td>
<td>20h: 51h &quot;Q&quot;</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>21h: 51h &quot;Q&quot;</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>22h: 52h &quot;R&quot;</td>
</tr>
<tr>
<td>two x16 devices operating in 8-bit mode</td>
<td>N/A(1)</td>
<td></td>
<td>40h</td>
<td>40h: 51h &quot;Q&quot;</td>
</tr>
<tr>
<td>(paired chip configuration)</td>
<td></td>
<td></td>
<td></td>
<td>41h: 51h &quot;Q&quot;</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>42h: 51h &quot;Q&quot;</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>43h: 51h &quot;Q&quot;</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>44h: 52h &quot;R&quot;</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>45h: 52h &quot;R&quot;</td>
</tr>
<tr>
<td>x32 device operating in 32-bit mode</td>
<td>10h</td>
<td>10h: 00000051h &quot;Q&quot;</td>
<td>40h</td>
<td>40h: 51h &quot;Q&quot;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>11h: 00000052h &quot;R&quot;</td>
<td></td>
<td>41h: 00h null</td>
</tr>
<tr>
<td></td>
<td></td>
<td>12h: 00000059h &quot;Y&quot;</td>
<td></td>
<td>42h: 00h null</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>43h: 00h null</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>44h: 52h &quot;R&quot;</td>
</tr>
<tr>
<td>x32 device operating in 8-bit mode</td>
<td>N/A(1)</td>
<td></td>
<td>40h</td>
<td>40h: 51h &quot;Q&quot;</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>41h: 51h &quot;Q&quot;</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>42h: 51h &quot;Q&quot;</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>43h: 51h &quot;Q&quot;</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>44h: 52h &quot;R&quot;</td>
</tr>
</tbody>
</table>

NOTE:
1. The system must drive the lowest order addresses to access all the device’s array data when the device is configured in x8 mode; therefore, word addressing where these lower addresses are not toggled by the system is “Not Applicable” for x8-configured devices.
NOTES:
1. Offset is the location in memory when using maximum device bus width addressing.
2. The CFI specification allows for replacement of all or part of the standard query table contents. If the vendor primary (or alternate) algorithm extended query table address (P or A) points to any address between 10h and the end of Table 4, “Flash Geometry Information” on page 9, the standard query table contents from that point on are assumed to be replaced by the information defined by the vendor primary (or alternate) algorithm. Thus, some or all of the standard query may be replaced. For example, a vendor primary (or alternate) algorithm extended query table address of 27h means that the standard device geometry definition has been replaced by something which has been defined by the vendor. The System Interface information at locations 1Bh to 26h may be assumed valid, but the ultimate definition must be described by the particular vendor algorithm. If the vendor primary (or alternate) algorithm extended query table address points to an address beyond the end of Table 4, “Flash Geometry Information,” a new table of data is included at that address. The contents of this table are defined by the corresponding vendor primary (or alternate) algorithm.

### Table 2. CFI Query Identification String

<table>
<thead>
<tr>
<th>Offset</th>
<th>Length (bytes)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>10h</td>
<td>03h</td>
<td>Query-unique ASCII string “QRY”</td>
</tr>
<tr>
<td>13h</td>
<td>02h</td>
<td>Primary Vendor Command Set and Control Interface ID Code 16-bit ID code defining specific Vendor-specified algorithm [Refer to CFI Publication 100 for definition of the ID codes]</td>
</tr>
<tr>
<td>15h</td>
<td>02h value = P</td>
<td>Address for Primary Algorithm extended Query table Note: Address 0000h means that no extended table exists</td>
</tr>
<tr>
<td>17h</td>
<td>02h</td>
<td>Alternate Vendor Command Set and Control Interface ID Code second vendor-specified algorithm supported by the device [Refer to CFI Publication 100 for definition of the ID codes] Note: ID Code = 0000h means that no alternate algorithm is employed</td>
</tr>
<tr>
<td>19h</td>
<td>02h value = A</td>
<td>Address for Alternate Algorithm extended Query table Note: Address 0000h means that no alternate extended table exists</td>
</tr>
</tbody>
</table>
### Table 3. System Interface Information

<table>
<thead>
<tr>
<th>Offset</th>
<th>Length (bytes)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1Bh</td>
<td>01h</td>
<td>$V_{CC}$ Logic Supply Minimum Program/Erase voltage</td>
</tr>
<tr>
<td></td>
<td></td>
<td>bits 7–4 BCD value in volts</td>
</tr>
<tr>
<td></td>
<td></td>
<td>bits 3–0 BCD value in 100 millivolts</td>
</tr>
<tr>
<td>1Ch</td>
<td>01h</td>
<td>$V_{CC}$ Logic Supply Maximum Program/Erase voltage</td>
</tr>
<tr>
<td></td>
<td></td>
<td>bits 7–4 BCD value in volts</td>
</tr>
<tr>
<td></td>
<td></td>
<td>bits 3–0 BCD value in 100 millivolts</td>
</tr>
<tr>
<td>1Dh</td>
<td>01h</td>
<td>$V_{PP}$ [Programming] Supply Maximum Program/Erase voltage</td>
</tr>
<tr>
<td></td>
<td></td>
<td>bits 7–4 HEX value in volts</td>
</tr>
<tr>
<td></td>
<td></td>
<td>bits 3–0 BCD value in 100 millivolts</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Note: This value must be 0000h if no $V_{PP}$ pin is present</td>
</tr>
<tr>
<td>1Eh</td>
<td>01h</td>
<td>$V_{PP}$ [Programming] Supply Maximum Program/Erase voltage</td>
</tr>
<tr>
<td></td>
<td></td>
<td>bits 7–4 HEX value in volts</td>
</tr>
<tr>
<td></td>
<td></td>
<td>bits 3–0 BCD value in 100 millivolts</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Note: This value must be 0000h if no $V_{PP}$ pin is present</td>
</tr>
<tr>
<td>1Fh</td>
<td>01h</td>
<td>Typical timeout per single byte/word write (buffer write count = 1), $2^n \mu s$</td>
</tr>
<tr>
<td>20h</td>
<td>01h</td>
<td>Typical timeout for maximum-size buffer write, $2^n \mu s$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(if supported; 00h = not supported)</td>
</tr>
<tr>
<td>21h</td>
<td>01h</td>
<td>Typical timeout per individual block erase, $2^n$ ms</td>
</tr>
<tr>
<td>22h</td>
<td>01h</td>
<td>Typical timeout for full chip erase, $2^n$ ms (if supported; 00h = not supported)</td>
</tr>
<tr>
<td>23h</td>
<td>01h</td>
<td>Maximum timeout for byte/word write, $2^n$ times typical (offset 1Fh)</td>
</tr>
<tr>
<td>24h</td>
<td>01h</td>
<td>Maximum timeout for buffer write, $2^n$ times typical (offset 20h)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(00h = not supported)</td>
</tr>
<tr>
<td>25h</td>
<td>01h</td>
<td>Maximum timeout per individual block erase, $2^n$ times typical (offset 21h)</td>
</tr>
<tr>
<td>26h</td>
<td>01h</td>
<td>Maximum timeout for chip erase, $2^n$ times typical (offset 22h)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(00h = not supported)</td>
</tr>
</tbody>
</table>
3.3 Read Vendor-Specific Extended Query Table

Using data from addresses 15h (address for primary algorithm extended query table) and possibly 19h (address for alternate algorithm extended query table), the system software can read more specific information about the flash device (see the CFI Query Flowchart, included in Appendix A). Each vendor will have specific data that should be read from the extended query table. Intel defines this data with its SCS. Also, each vendor may locate this table in a different location, so it is important that the software reads the location of the tables from offsets 15h and 19h to determine where (if at all) the extended query data is stored. The Vendor Command Set definition (Intel’s SCS) will indicate what data is stored in the extended query table. Table 5, “Intel Primary Algorithm Extended Query Table” on page 10 shows the extended table for the Intel devices implementing CFI (all devices implementing CFI will use the extended query table regardless of the command set being used).

---

Table 4. Flash Geometry Information

<table>
<thead>
<tr>
<th>Offset</th>
<th>Length (bytes)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>27h</td>
<td>01h</td>
<td>Device Size = $2^n$ in number of bytes.</td>
</tr>
<tr>
<td>28h</td>
<td>02h</td>
<td>Flash Device Interface description [Refer to CFI Publication 100]</td>
</tr>
<tr>
<td>2Ah</td>
<td>02h</td>
<td>Maximum number of bytes in buffer write = $2^n$.</td>
</tr>
</tbody>
</table>

| 2Ch    | 01h            | Number of erase block regions within device |
|        |                | bits 7–0 = x = number of erase block regions |
| Notes: |                | 1. x = 0 means no erase blocking, i.e., the device erases at once in “bulk.” |
|        |                | 2. x specifies the number of regions within the device containing one or more contiguous erase blocks of the same size. For example, a 128-KB device (1 Mb) having blocking of 16-KB, 8-KB, four 2-KB, two 16-KB, and one 64-KB is considered to have five erase block regions. Even though two regions both contain 16-KB blocks, the fact that they are not contiguous means they are separate erase block regions. |
|        |                | 3. By definition, symmetrically-blocked devices have only one blocking region. |

| 2Dh    | 04h            | Erase block region information |
|        |                | bits 31–16 = z, where the erase block(s) within this region are (z) times 256 bytes in size. |
|        |                | The value z = 0 is used for 128-byte block size. |
|        |                | e.g., for 64-KB block size, z = 0100h = 256 => 256 * 256 = 64K |
|        |                | bits 15–0 = y, where y+1 = number of erase blocks of identical size within the erase block region, e.g.: |
|        |                | y = D15:D0 = FFFFh => y+1 = 64K blocks [maximum number] |
|        |                | y = 0 means no blocking (# blocks = y+1 = “1 block”) |
| Note:  |                | y = 0 value must be used with # of block regions of one as indicated by (x) = 0 |

| 31h to | 04h per entry  | Additional erase block region information, 4 bytes per region |
| (k-1)h |                | Notes: |
|        |                | 1. The total number of blocks multiplied by individual block size must add up to the device size. |
|        |                | 2. The address K is next available query address at end of the device geometry structure. It is the first possible starting address of the optional vendor-specific query table(s) (i.e., Address “P,” the primary vendor-specific extended query table offset, must be ≥ k to not overwrite the existing tables). See note 2 under Table 2, “CFI Query Identification String” on page 7 for more information. |
Table 5. Intel Primary Algorithm Extended Query Table

<table>
<thead>
<tr>
<th>Offset</th>
<th>Length (bytes)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>(P + 0)h</td>
<td>03h</td>
<td>Primary extended Query table unique ASCII string “PRI”</td>
</tr>
<tr>
<td>(P + 1)h</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(P + 2)h</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(P + 3)h</td>
<td>01h</td>
<td>Major version number, ASCII</td>
</tr>
<tr>
<td>(P + 4)h</td>
<td>01h</td>
<td>Minor version number, ASCII</td>
</tr>
<tr>
<td>(P + 5)h</td>
<td>04h</td>
<td>Optional Feature &amp; Command Support (1=yes, 0=no)</td>
</tr>
<tr>
<td>(P + 6)h</td>
<td></td>
<td>bits 9-31 are reserved; undefined bits are “0”. If bit 31 is “1,” then another 31-bit field of optional features follows at the end of the bit-30 field.</td>
</tr>
<tr>
<td>(P + 7)h</td>
<td></td>
<td>bit 0 Chip erase supported</td>
</tr>
<tr>
<td>(P + 8)h</td>
<td></td>
<td>bit 1 Suspend erase supported</td>
</tr>
<tr>
<td>(P + 9)h</td>
<td>01h</td>
<td>Supported functions after suspend</td>
</tr>
<tr>
<td>(P + A)h</td>
<td></td>
<td>Read array, status, and query are always supported during suspended erase or program operation. This field defines other operations supported.</td>
</tr>
<tr>
<td>(P + B)h</td>
<td></td>
<td>bit 0  Program supported after Erase Suspend (1=yes, 0=no)</td>
</tr>
<tr>
<td>(P + C)h</td>
<td>01h</td>
<td>Block status register mask</td>
</tr>
<tr>
<td>(P + D)h</td>
<td>01h</td>
<td>$V_{CC}$ logic supply optimum program/erase voltage (highest performance)</td>
</tr>
<tr>
<td>(P + E)h</td>
<td></td>
<td>$V_{PP}$ [programming] supply optimum program/erase voltage</td>
</tr>
</tbody>
</table>

Note: If no $V_{PP}$ pin is present, this value must be 0000h.
3.4 **Software Branch to Appropriate Routines**

Using the information read from location 13h (Primary Vendor Command Set and Control Interface ID code) and possibly from 17h (Alternate Vendor Command Set and Control Interface ID Code), the system determines which set(s) of commands are recognizable by the flash device. Every vendor has defined at least one set of commands that their flash devices accept. Any commands other than those defined will either be rejected or cause unexpected behavior and should therefore be avoided.

Intel has two such sets of commands defined, BCS (Basic Command Set) and SCS (Scaleable Command Set). SCS includes all of the BCS commands plus some new enhanced ones. The low level reference code provided by Intel includes both of these command sets. By using this reference code, current and future flash devices may be controlled using the same system software drivers.

4.0 **How to Use Command Sets Effectively**

Flowcharts for most of the driver software routines necessary to implement the Basic and Scaleable Command Sets are located in Appendix B and Appendix C respectively. All of the reference codes may be downloaded from the Intel web site from the Electronic Tools Catalog (ETC). The ETC can be accessed from the Intel Developer’s site from:

http://developer.intel.com/design/flash/swtools/etc.htm

Link to the ETC: Flash Memory components set “Tool Type” to “Software-Templates” and select the button “find my selection” from the list select the common flash interface, or CFI entry. If all works well, it can be directly linked at:


To ensure compatibility with future CFI compliant devices, all functions of the reference code should be included in a design.

5.0 **Conclusion**

CFI and SCS have been created to allow a system designer the flexibility to design products now that can use both current and future flash memory devices, as well as the security of knowing that second source products may be used without system software modifications.
Appendix A  Memory Hueristics

A1  Memory Hueristics Flowchart

Start

Read and Store Data from Device Address 0

Issue CFI Query
Write 98h to Address 55h

Data = "QRY"?

Yes

Read CFI Parameters from Device (See CFI Query)

Return, CFI Device

No

Issue Read JEDEC ID
Write 90h to Address 0h

Data at 0 = 90h?

Yes

Restore Original Data

Determine Size of RAM

Load Software Routine Addresses from Tables

Return, SRAM Device

No

Read IDs?

Yes

Issue AMD Read JEDEC ID
Write AA55h to Address 0h

Read IDs?

Yes

Read JEDEC parameters from System Table (See JEDEC Query)

Return, JEDEC Flash Device

No

Load Software Routine Addresses from Tables

Return, ROM Device
**CFI Query Flowchart**

**Start**

Issue CFI Query, to address 55h

**Paired x?**

Yes

**Read Erase Block Size**

- Read Number of Erase Regions
- Read Device Size
- Read Max Write Buffer Size
- Read Max and Typical Voltages and Timeouts
- Read Address of Extended Query Table
- Determine Features Supported by Device

**Map to Next Flash Device**

**Put Device into Read Array Mode, Write FFh**

- **Data = QRY?**
  - Yes
    - Issue CFI Query Command to Address 55h
  - No
    - **Put Device into Read Array Mode, Write FFh**

**Bus Operation**

<table>
<thead>
<tr>
<th>Command</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Write CFI Query</td>
<td>Data = 98h &lt;br&gt;Addr = 55h</td>
</tr>
<tr>
<td>Read ASCII Q, R, and Y</td>
<td></td>
</tr>
<tr>
<td>Read Block Size</td>
<td></td>
</tr>
<tr>
<td>Read Number of Erase Regions</td>
<td></td>
</tr>
<tr>
<td>Read Device Size</td>
<td></td>
</tr>
<tr>
<td>Read Max Write Buffer Size</td>
<td></td>
</tr>
<tr>
<td>Read Max and Typical Voltages and Timeouts</td>
<td></td>
</tr>
<tr>
<td>Read Address of Extended Table</td>
<td></td>
</tr>
<tr>
<td>Read Features Supported by Device</td>
<td></td>
</tr>
<tr>
<td>Write CFI Query</td>
<td>Data = 98h &lt;br&gt;Addr = 55h</td>
</tr>
<tr>
<td>Read ASCII Q, R, Y</td>
<td></td>
</tr>
<tr>
<td>Write Read Array Data = FFh &lt;br&gt;Addr = X</td>
<td></td>
</tr>
</tbody>
</table>
A3  Read JEDEC ID Flowchart

Start

Issue Read ID, Write 90h to Address 0

Read Location 0 and 1 in Flash Array (Manufacturer and Device ID)

JEDEC ID in Table?

Load geometric parameters from Software Tables

Load Software Routine Addresses from Tables

Map in Next Flash Device

Return, Unknown Device

JEDEC ID?

Issue Read JEDEC ID Command (90h)

Return, JEDEC Flash Device

JEDEC ID?

Issue Read Array Command (FFh)

Bus Operation | Command | Comments
---|---|---
Write | Read ID | Data = 90h Addr = 0
Read | Data = Manufacturer ID Addr = 0
Read | Data = Device ID Addr = 1
Write | Read ID | Data = 90h Addr = 0
Write | Read Array | Data = FFh Addr = X

Start

Write Read ID Data = 90h Addr = 0

Read Data = Manufacturer ID Addr = 0

Read Data = Device ID Addr = 1

Write Read ID Data = 90h Addr = 0

Write Read Array Data = FFh Addr = X

Start

Read Location 0 and 1 in Flash Array (Manufacturer and Device ID)

JEDEC ID in Table?

Load geometric parameters from Software Tables

Load Software Routine Addresses from Tables

Map in Next Flash Device

Return, Unknown Device

JEDEC ID?

Issue Read Array Command (FFh)

Return, JEDEC Flash Device

JEDEC ID?

Issue Read Array Command (FFh)
Appendix B  Basic Command Set

B1  Clear Status Register Flowchart

![Clear Status Register Flowchart Diagram]

<table>
<thead>
<tr>
<th>Bus Operation</th>
<th>Command</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Write</td>
<td>Clear Status Register</td>
<td>Data = 50h</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Addr = X</td>
</tr>
</tbody>
</table>

B2  Read Array Flowchart

![Read Array Flowchart Diagram]

<table>
<thead>
<tr>
<th>Bus Operation</th>
<th>Command</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Write</td>
<td>Read Array</td>
<td>Data = FFh</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Addr = Address to read</td>
</tr>
<tr>
<td>Standby</td>
<td>Check SR.7</td>
<td>1 = WSM ready</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 = WSM busy</td>
</tr>
<tr>
<td>Read</td>
<td>Read array</td>
<td>locations</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Standby Check SR.7

Need to Resume?

Call Resume Routine

Return, Data Read
### B3 Block Erase Flowchart

#### Flowchart:

1. **Start**
   - **Device Supports Queuing?**
     - **No**
       - **Set Timeout**
         - **Issue Block Queue Erase Command 28h and Block Address**
         - **Read Extended Status Register**
           - **XSR.7 = ?**
             - **= 0, No**
               - **Another Block Erase?**
                 - **Yes**
                   - **Issue Block Queue Erase Command 28h and Block Address**
                   - **Read Extended Status Register**
                     - **XSR.7 = ?**
                       - **= 1, Yes**
                         - **Write Confirm D0h and Block Address**
                           - **Another Block Erase?**
                             - **Yes**
                               - **Read Status Register**
                                 - **SR.7 = ?**
                                   - **= 0, No**
                                     - **Suspend Erase Loop**
                                   - **= 1, Yes**
                                     - **Full Status Check if Desired**
                                       - **Erase Flash Block(s) Complete**
     - **Yes**
       - **Set Timeout**
         - **Issue Block Queue Erase Command 28h and Block Address**
         - **Read Extended Status Register**
           - **XSR.7 = ?**
             - **= 0, No**
               - **Another Block Erase?**
                 - **Yes**
                   - **Issue Block Queue Erase Command 28h and Block Address**
                   - **Read Extended Status Register**
                     - **XSR.7 = ?**
                       - **= 1, Yes**
                         - **Write Confirm D0h and Block Address**
                           - **Another Block Erase?**
                             - **Yes**
                               - **Read Status Register**
                                 - **SR.7 = ?**
                                   - **= 0, No**
                                     - **Suspend Erase Loop**
                                   - **= 1, Yes**
                                     - **Full Status Check if Desired**
                                       - **Erase Flash Block(s) Complete**
   - **Yes**
     - **Set Timeout**
       - **Issue Block Queue Erase Command 28h and Block Address**
       - **Read Extended Status Register**
         - **XSR.7 = ?**
           - **= 0, No**
             - **Another Block Erase?**
               - **Yes**
                 - **Issue Block Queue Erase Command 28h and Block Address**
                 - **Read Extended Status Register**
                   - **XSR.7 = ?**
                     - **= 1, Yes**
                       - **Write Confirm D0h and Block Address**
                         - **Another Block Erase?**
                           - **Yes**
                             - **Read Status Register**
                               - **SR.7 = ?**
                                 - **= 0, No**
                                   - **Suspend Erase Loop**
                                 - **= 1, Yes**
                                   - **Full Status Check if Desired**
                                     - **Erase Flash Block(s) Complete**
   - **No**
     - **Erase Block Available?**
       - **Yes**
         - **Issue Block Queue Erase Command 28h and Block Address**
         - **Read Extended Status Register**
           - **XSR.7 = ?**
             - **= 0, No**
               - **Another Block Erase?**
                 - **Yes**
                   - **Issue Block Queue Erase Command 28h and Block Address**
                   - **Read Extended Status Register**
                     - **XSR.7 = ?**
                       - **= 1, Yes**
                         - **Write Confirm D0h and Block Address**
                           - **Another Block Erase?**
                             - **Yes**
                               - **Read Status Register**
                                 - **SR.7 = ?**
                                   - **= 0, No**
                                     - **Suspend Erase Loop**
                                   - **= 1, Yes**
                                     - **Full Status Check if Desired**
                                       - **Erase Flash Block(s) Complete**
         - **No**
           - **Issue Single Block Erase Command 20h and Block Address**
           - **Write Confirm D0h and Block Address**
           - **Another Block Erase?**
             - **Yes**
               - **Read Status Register**
                 - **SR.7 = ?**
                   - **= 0, No**
                     - **Suspend Erase Loop**
                   - **= 1, Yes**
                     - **Full Status Check if Desired**
                       - **Erase Flash Block(s) Complete**

#### Table:

<table>
<thead>
<tr>
<th>Bus Operation</th>
<th>Command</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Write</td>
<td>Erase Block</td>
<td>Data = 28h or 20h, Addr = Block Address</td>
</tr>
<tr>
<td>Read</td>
<td>XSR.7 = valid, Addr = X</td>
<td></td>
</tr>
<tr>
<td>Standby</td>
<td>Check XSR.7, 1 = Erase queue available, 0 = No erase queue available</td>
<td></td>
</tr>
<tr>
<td>Write</td>
<td>Erase Block</td>
<td>Data = 28h, Addr = Block Address</td>
</tr>
<tr>
<td>Read</td>
<td>SR.7 = valid, SR.6-0 = X, CE# and OE# low updates SR, Addr = X</td>
<td></td>
</tr>
<tr>
<td>Standby</td>
<td>Check XSR.7, 1 = Erase queue available, 0 = No erase queue available</td>
<td></td>
</tr>
<tr>
<td>Write</td>
<td>Erase Confirm</td>
<td>Data = D0h, Addr = X</td>
</tr>
<tr>
<td>Read</td>
<td>Status Register data, CE# and OE# low updates SR, Addr = X</td>
<td></td>
</tr>
<tr>
<td>Standby</td>
<td>Check SR.7, 1 = WSM ready, 0 = WSM busy</td>
<td></td>
</tr>
</tbody>
</table>

1. The Erase Confirm byte must follow Erase setup when the erase queue status (XSR.7) = 0. Full status check can be done after all Erase and Program sequences complete. Write FFh after the last operation to reset the device to read array mode.
B4  Erase Suspend/Resume Flowchart

Start

Issue Suspend Command, Write 0BH

Read Status Register

SR.7 = 0

SR.6 = 0

Erase Complete

Read Array Data, Read SR, or Write Data

No

Issue Resume command, Write D0h

Erase Resumed

Yes

Issue Read Command, Write FFh

Return, Nothing to Suspend

Bus Operation | Command | Comments
---|---|---
Write | Erase Suspend | Data = B0h
| | Addr = X
Read | SR.7 valid; SR.6-0=X
| CE# & OE# low updates SR
| Addr = X
Standby | Check SR.7
| 1 = WSM ready
| 0 = WSM busy
Standby | Check SR.6
| 1 = Erase suspended
| 0 = Erase in progress or complete
Write | Read Array, Read SR, or Write
| Data = FFh
| Addr = X
Read/Write | Read/Write array locations other than those being erased or Read SR
Write | Resume
| Data = D0h
| Addr = X

B5  Single Byte Program (Write) Flowchart

Start

Issue Program Command, 40h or 10h and Address

Write Data to Address

Read Status Register

SR.7 = 0

Full Status Check if Desired

Byte / Word Program Complete

Bus Operation | Command | Comments
---|---|---
Write | Setup Byte / Word Program
| Data = 40h or 10h
| Addr = Location to Be Programmed
Write | Byte / Word Program
| Data = Data to Be Programmed
| Addr = Location to Be Programmed
Read | Status Register Data

Standby | Check SR.7
| 1 = WSM Ready
| 0 = WSM Busy

Repeat for subsequent program operations.
Full status register check can be done after each program operation or after a sequence of Programming.
Write FFh after the last program operation to place the device in read array mode.
Full Status Check Flowchart

SR.5, SR.4, SR.3, and SR.1 are only cleared by the Clear Status Register command. If error is detected, clear the status register before attempting retry or other error recovery.

<table>
<thead>
<tr>
<th>Bus Operation</th>
<th>Command</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Standby</td>
<td>Check SR.3</td>
<td>1 = Programming Voltage Error Detect</td>
</tr>
<tr>
<td>Standby</td>
<td>Check SR.1</td>
<td>1 = Device Protect Detect</td>
</tr>
<tr>
<td>Standby</td>
<td>Check SR.4,5</td>
<td>Both 1 = Command Sequence Error</td>
</tr>
<tr>
<td>Standby</td>
<td>Check SR.5</td>
<td>1 = Clear Block Lock-Bits Error</td>
</tr>
</tbody>
</table>

Start
Read Status Register

SR.1 = 1
Device Protect Error

SR.2 = 1
Write Is Suspended

SR.3 = 1
Voltage Range Error

SR.4 = 1
Write or Block Lock Bit Set Error

SR.5 = 1
Clear Block Lock Bit or Erase Error

SR.6 = 1
Erase Suspended

SR.4, 5 = 1
Command Sequence Error

Return, Status Checked
Appendix C  Scaleable Command Set

C1  Write to Buffer Flowchart

### Bus Operation

<table>
<thead>
<tr>
<th>Command</th>
<th>Comments</th>
</tr>
</thead>
</table>
| **Write** | Write to Buffer  
Data = E8h  
Addr = Block Address |
| **Read** | XSR.7 = valid; XSR.8-0 = X;  
Addr = X |
| **Standby** | Check XSR.7  
1 = Write buffer ready  
0 = No write buffer ready |
| **Write** (Notes 1,2) | Data = N = word/byte count  
N = 0 corresponds to count = 1  
Addr = Block Address |
| **Write** (Notes 3,4) | Data = write buffer data  
Addr = device address |
| **Write** (Notes 5,6) | Data = write buffer data  
Addr = device address |
| **Write** | Write Confirm  
Data = D0h  
Addr = X |
| **Read** | Status Register Data  
CE# and OE# low updates SR  
Addr = X |
| **Standby** | Check SR.7  
1 = WSM ready  
0 = WSM busy |

### Notes
1. Byte/word count values on DQ 0-7 are loaded into the count register.
2. The device now outputs the status register when read (XSR is no longer available).
3. Write Buffer contents will be programmed at the device start address or destination flash address.
4. Align the start address on a Write Buffer boundary for maximum programming performance.
5. The device aborts the Write to Buffer command if the current address is outside of the original block address.
6. The status register indicates an “improper command sequence” if the Write to Buffer command is aborted. Follow this with a Clear Status Register command.

Full status check can be done after all Erase and Write sequences complete. Write FFh after the last operation to reset the device to read array mode.
C2  Program (Write) Suspend/Resume Flowchart

Start
Issue Suspend Command
Write B0h
Read Status Register
SR.7 =

Read Array Data
Issue Read Command
Write FFh

No

Read?

Yes

Issue Resume command
Write D0h
Write Resumed

Return, Nothing to Suspend

Write Complete

Bus Operation | Command | Comments
---|---|---
Write | Write Suspend | Data = B0h, Addr = X
Read | SR.7=valid; SR.6-0=0->X | CEA & OE# low updates SR
| | Addr = X |
Standby | Check SR.7 | 1 = WSM ready
| | 0 = WSM busy |
Standby | Check SR.2 | 1 = Write suspended
| | 0 = Write complete |
Write | Read Array | Data = FFh, Addr = X
Read | Read array locations other than those being written |
Write | Resume | Data = D0h, Addr = X

C3  Block Lock Bit Set Flowchart

Start
Write 60h, Block Address
Write 01h, Block Address
Read Status Register
SR.7 =

Issue Read Command
Write FFh

Full Status Check if Desired
Set Lock-Bit
Complete

Bus Operation | Command | Comments
---|---|---
Write | Set Block-Lock Bit | Data = 60h, Addr = Block Address (Block)
Write | Set Block-Lock Bit Confirm | Data = 01h, Addr = Block Address (Block)
Read | Status Register Data |
Standby | Check SR.7 | 1 = WSM Ready
| | 0 = WSM Busy |

Repeat for subsequent lock-bit set operations.
Full status check can be done after each lock-bit set operation or after a sequence of lock-bit set operations.
Write FFh after the last lock-bit set operation to place device in read array mode.
C4    Block Lock Bit Reset Flowchart

**Flowchart Diagram:**

```
Start
  ↓ Write 60h
  ↓ Write DO
  ↓ Read Status Register
      ↓ SR.7 = 0
          ↓ Full Status Check if Desired
                ↓ Clear Block Lock-Bits Complete
                1
          ↓ SR.7 = 1
```

**Bus Operation Table:**

<table>
<thead>
<tr>
<th>Bus Operation</th>
<th>Command</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Write</td>
<td>Clear Block Lock-Bits Setup</td>
<td>Data = 60h, Addr = X</td>
</tr>
<tr>
<td>Write</td>
<td>Clear Block Lock-Bit Confirm</td>
<td>Data = DOh, Addr = X</td>
</tr>
<tr>
<td>Read</td>
<td>Status Register Data</td>
<td></td>
</tr>
<tr>
<td>Standby</td>
<td>Check SR.7</td>
<td>1 = WSM Ready, 0 = WSM Busy</td>
</tr>
</tbody>
</table>

Write FFh after the last lock-bit set operation to place device in read array mode.