

# Series III Programmer

**Operation Manual** 



### **Table of Contents**

| Hardware Background     | 2  |
|-------------------------|----|
| Sample Applications     | 4  |
| Acronymns/Abbreviations | 20 |



## Hardware Background

Datakey Serial Memory devices each contain a commercially available memory chip. The memory chip is designed to:

- operate within a specified supply voltage range,
- communicate via a specific protocol (SPI Flash, SPI EEPROM, I2C, or Microwire), and
- have a specified memory capacity (i.e. size) usually measured in bits.

The portable memory devices are available are available in several different form factors: key, Slimline token, Plug token, and Bar token. Each shape has a corresponding receptacle that allows for quick insertion/removal and electrical connection to the device. See the appropriate protocol's interface specification document for the various device shapes, their associated receptacle pin numbering conventions, and signal locations.

The Series III Programmer is a microcontroller-based product that contains the firmware and interface circuitry necessary to allow a host computer to communicate (via USB) with a Datakey key or token that uses the SPI Flash, SPI EEPROM, I<sup>2</sup>C, or Microwire protocol. Once power is applied to the Series III Programmer (via the host computer's USB port), its two-color LED will illuminate. The LED will be red while reading, writing, or erasing of the key is in progress; green, otherwise. If the token is withdrawn from the Series III Programmer while the LED is red, data corruption might occur.

Datakey memory devices have a family name embossed on them that indicates their communications protocol, memory technology and form factor. In addition, they have a size code, a voltage code, and a date code embossed on them. See chart on page 3.

| KEY<br>SQUARE HEAD            | KEY<br>Round head                    | SLIMLINE<br>TOKEN                  |
|-------------------------------|--------------------------------------|------------------------------------|
|                               |                                      |                                    |
| DK                            | IIK, ISK,<br>LCK, SFK,<br>SFK5V, SSK | IIT, IST,<br>LCS, SFT,<br>SLT, SST |
| EXTENDED<br>SLIMLINE<br>TOKEN | PLUG                                 | BAR                                |
|                               |                                      |                                    |
| IIX, ISX, LCX,<br>SFX, SSX    | ISP, KSD, SSP                        | ISB, LCB, SSB                      |



# Key/Token/Plug Size Codes

| CODE | SIZE IN BITS | BYTES     |
|------|--------------|-----------|
| А    | 1 Kb         | 128 bytes |
| В    | 2 Kb         | 256 bytes |
| С    | 4 Kb         | 512 bytes |
| D    | 8 Kb         | 1 KB      |
| Е    | 16 Kb        | 2 KB      |
| F    | 32 Kb        | 4 KB      |
| G    | 64 Kb        | 8 KB      |
| Н    | 128 Kb       | 16 KB     |
| 1    | 256 Kb       | 32 KB     |
| J    | 512 Kb       | 64 KB     |
| K    | 1 Mb         | 128 KB    |
| L    | 2 Mb         | 256 KB    |
| М    | 4 Mb         | 512 KB    |
| N    | 8 Mb         | 1 MB      |
| 0    | 16 Mb        | 2 MB      |
| Р    | 32 Mb        | 4 MB      |
| Q    | 64 Mb        | 8 MB      |
| R    | 128 Mb       | 16 MB     |
| S    | 256 Mb       | 32 MB     |
| Т    | 512 Mb       | 64 MB     |
| U    | 1 Gb         | 128 MB    |
| V    | 2 Gb         | 256 MB    |
| W    | 4 Gb         | 512 MB    |
| Х    | 8 Gb         | 1 GB      |
| Y    | 16 Gb        | 2 GB      |
| Z    | 32 Gb        | 4 GB      |

**SIZE IN BITS** 

1 Gb

2 Gb 4 Gb

8 Gb

16 Gb

32 Gb

64 Gb

128 Gb

256 Gb

512 Gb

1 Tb

2 Tb

4 Tb

8 Tb

**BYTES** 

128 MB

256 MB

512 MB

1 GB

2 GB

4 GB

8 GB

16 GB

32 GB

64 GB

128 GB

256 GB

512 GB

1 TB

CODE

В

С

D

Е

G

J K

Μ

KEY MARKING



EXTENDED TOKEN MARKING



RUGGEDRIVE TOKEN MARKING



#### Date Codes

| N | IONTH |   |      |      |      |
|---|-------|---|------|------|------|
| 1 | lan   | 1 | 2001 | 2011 | 2021 |
|   | Jan   |   | 2001 | 2011 | ZUZI |
| 2 | Feb   | 2 | 2002 | 2012 | 2022 |
| 3 | Mar   | 3 | 2003 | 2013 | 2023 |
| 4 | Apr   | 4 | 2004 | 2014 | 2024 |
| 5 | May   | 5 | 2005 | 2015 | 2025 |
| 6 | Jun   | 6 | 2006 | 2016 | 2026 |
| 7 | Jul   | 7 | 2007 | 2017 | 2027 |
| 8 | Aug   | 8 | 2008 | 2018 | 2028 |
| 9 | Sep   | 9 | 2009 | 2019 | 2029 |
| 0 | Oct   | 0 | 2010 | 2020 | 2030 |
| N | Nov   |   |      |      |      |
| D | Dec   |   |      |      |      |

Although the memory chip within a device has a range of voltages over which it can operate, the Series III Programmer can only be configured to apply either 3.3 volts or 5 volts to the device.

MONTH DATE CODE

AR DATE CODE

MONTH DATE CODE

YEAR DATE CODE

FAMILY N

MEMORY SIZE

VOLTAGE COD

LEGACY INFORMATION: NO MARKS OR BLACK TOP IS 1 KB, 5 V MICROMPER INTERACE

licate

PLUG MARKING

TOKEN MARKING

VOLTAGE CODE

EMORY SIZE

LEGACY TOKEN



BAR MARKING



#### Voltage Codes

| CODE  | VOLTAGE RANGE |
|-------|---------------|
| Blank | 4.5 - 5.5 V   |
| 5     | 3.0 - 3.6 V   |
| 4     | 2.7 - 5.5 V   |
| 3     | 2.7 - 3.6 V   |
| 2     | 1.8 - 3.6 V   |

3



# **Sample Applications**

There are two different designs within the sample applications included on the flash drive/download: the C++ implementation and the C#/Visual Basic implementation. The applications have many features in common and some features that are unique to one or the other. The operation of each of the applications is described separately. Refer to the correct corresponding section.

### **C++ Implementation**

To start the application, do the following:

- attach the Series III Programmer to the host computer's USB 2.0 port, then
- double-click the Series III Programmer desktop icon on the host computer.

### Configuration

Each time the sample application is started, it will prompt the user to configure the attached Series III Programmer for the type of device that is to be expected. This is a crucial step, it tells the Series III Programmer:

- what protocol is to be used for communication,
- what supply voltage to apply to the device (either 3.3 or 5 volts), and
- the memory capacity of the device.

The sample application cannot query this information from an attached device, it must be told what to expect. The Series III Programmer does remember the type of device for which it was last configured and the corresponding 5-character code is displayed in the **Code** text box of the **Configure Programmer** form.

Expand the appropriate protocol family and select the specific device to be used in the Series III Programmer. Note that the same 5-character code may be assigned to more than one particular device in the list. That's because the electrical interface to the devices may be the same, even though their physical shapes are different.

Once the desired configuration is selected, click **OK**. In this example, the Series III Programmer is being configured for an SFK 2Mb device.

Note that if the Series III Programmer is configured for a Serial Flash device, a message will be displayed, reminding the user to manually erase the device before overwriting any existing data. EEPROM devices implement an automatic erase feature whenever new data is written; Serial Flash does not.

All of the Datakey devices that utilize Serial Flash memory have family names that begin with the letters "SF" (e.g., SFK, SFT, SFX).



| Configure Programmer  | ×           |
|---|-------------|
| Make a selection from the list below or type the 5 character cod<br>the help menu if your part is not in the list.  | e from      |
| <ul> <li>⊕: Microwine (DK, KSD, LCB, LCK, LCS, LCA)</li> <li>⊕: I2C (I0T, ISB, ISK, ISP, IST, ISA)</li> <li>⊕: I2C Senialized (IIK, IIT, IIA)</li> <li>⊕: SPI EBERPON (ISGT, SLK, SLT, SSB, SSK, SSM, SSP, SST, SSA)</li> <li>⊕: SPI Flash Standard 3.37 (SPK, SFT, SPA)</li> <li>⊕: SPI Flash 5V (SFKSV, SFTSV, SFASV)</li> </ul>  |             |
| Туре:   | -           |
| Product:  |             |
| Shape:  |             |
| Voltage:  |             |
| Code: SF3GL   |             |
| OK. Cancel  |             |
| Configure Programmer Make a selection from the list below or type the 5 character cod<br>the help menu if your part is not in the list<br>SPI Flash Standard 3.3V (SFK, SFT, SFX)<br>SFK Mub<br>SFK 8Mb<br>SFK 8Mb<br>SFK 16Mb<br>SFK |             |
| Product: SFK 2Mb  |             |
| Shape: Key-Hound Head   | _           |
| Voltage: 3V   |             |
|   |             |
|   |             |
| SeriesIIIProgrammerSampleCppApp   | X           |
| •   |             |
| Remember to erase your Serial Flash device before   | re-writing. |



### **Overview**

The Sample Cpp App pictured below has:

- a menu bar at the top that offers a choice of File, Configure, Function, and Help,
- a message window that shows the results of operations recently performed and the general status of the application,
- a **Data** group box where address limits and data patterns can be specified and where read, write, and compare operations can be performed,
- an Erase group box where individual sectors or the entire device can be targeted for erasure,
- an Electronic Signature group box where the signature of a Serial Flash device can be displayed,
- a Block Protect (SPI) group box where the block protect bits of an SPI device can be read and set/cleared,
- a KeyLink III group box where the Series III Programmer can be reconfigured or reset and where the on-line/off-line status of the Series III Programmer can easily be determined,
- a status bar that indicates success (green), failure (red), or progress (blue) of the pending operation, and
- an information bar where information about the software, firmware, configuration, and hardware status are displayed.

| TTING PROGRAMMER<br>member to erase ;<br>3GL<br>Data | CONFIGURATI<br>your Serial | ON<br>Flash devic                     | e before re-wri | ting.                 |            |                 |
|--|----------------------------|---------------------------------------|-----------------|-----------------------|------------|-----------------|
| Data   |                            |                                       |                 |                       |            |                 |
| Data   |                            |                                       |                 |                       |            |                 |
|  |                            |                                       |                 | Erase                 |            |                 |
| Start Address 0                                      | ☐ Hex                      | Write                                 | Read            | Sector Erase          | 0          |                 |
| Number of Bytes 128                                  | ☐ Hex                      | Verify                                | Page Size       | BULK Erase            |            |                 |
| Data Pattern ABC                                     | DEF01 THex                 | Page Write                            | 256 bytes       | -Electronic Signature |            | Datakey.        |
|  |                            | File Write                            |                 | Read                  |            |                 |
| Block Protect (SPI)                                  |                            |                                       |                 | KeyLink III           |            |                 |
|  |                            | - 1                                   | high bytes      | Configure             | Programmer | www.datakey.com |
| BP1 BP0  | read DP Dits               |                                       |                 | Reset                 | Detected   |                 |
|  | Write BP Bits              | -1                                    | low bytes       |                       |            |                 |
|  |                            | red = protected<br>black = unprotecte | d               |                       |            |                 |
|  |                            |                                       |                 |                       |            |                 |

Note that not all controls are available or appropriate for all types of devices.



### Insertion

When a key is inserted into the receptacle, an image of a key will be displayed within the **KeyLink III** group box. (Note that for an actual key type of device, the key must be inserted into the receptacle and turned a quarter turn clockwise before the image of a key will appear.) Behind the scenes, the application is polling the status of the LOFO (last on, first off) signal and when it goes low, the image is displayed.

| Configure |  |
|-----------|--|
| Reset     |  |
| nesei     |  |
|           |  |

| =ADDR=== ====<br>0000 0000:41 4<br>0000 0010:41 4<br>0000 0020:41 4<br>0000 0030:41 4 | 12 43 44<br>12 43 44<br>12 43 44<br>12 43 44<br>12 43 44 | 45 46<br>45 46<br>45 46<br>45 46<br>45 46 | === HEX<br>30 31 4<br>30 31 4<br>30 31 4<br>30 31 4<br>30 31 4 | 1 42 4<br>1 42 4<br>1 42 4<br>1 42 4<br>1 42 4<br>1 42 4 | 3 44<br>3 44<br>3 44<br>3 44<br>3 44<br>3 44 | 45<br>45<br>45<br>45<br>45 | 46<br>46<br>46<br>46 | 30<br>30<br>30<br>30<br>30 | 31<br>31<br>31<br>31<br>31<br>31 | == ASCII DATA ==<br>ABCDEF01ABCDEF01<br>ABCDEF01ABCDEF01<br>ABCDEF01ABCDEF01<br>ABCDEF01ABCDEF01<br>ABCDEF01ABCDEF01 | •                 |
|---|--|---|--|--|--|----------------------------|----------------------|----------------------------|----------------------------------|--|-------------------|
| 0000 0050:41 4<br>0000 0060:41 4<br>0000 0070:41 4                                    | 12 43 44<br>12 43 44<br>12 43 44                         | 45 46<br>45 46<br>45 46                   | 30 31 4<br>30 31 4<br>30 31 4                                  | 1 42 4<br>1 42 4<br>1 42 4                               | 3 44<br>3 44<br>3 44                         | 45<br>45<br>45             | 46<br>46<br>46       | 30<br>30<br>30             | 31<br>31<br>31                   | ABCDEF01ABCDEF01<br>ABCDEF01ABCDEF01<br>ABCDEF01ABCDEF01   |                   |
| Data<br>Start Address<br>Number of Bytes<br>Data Pattern                              | 0<br>128<br>ABCDEF                                       | — Не<br>— Не<br>01 — Не                   | x Wi<br>x Ve<br>x Page<br>File V                               | ite<br>ify<br>Write<br>/rite                             | Page<br>256                                  | Rea<br>Size                | d<br>byti            | es                         |                                  | Erase 0 BULK Erase 0 Electronic Signature Read   | Datakey.          |
| Block Protect (SPI)<br>BP1 BP0  | Read   | BP Bits<br>BP Bits                        | red = prot<br>black = ur                                       | - hij<br>- lor<br>acted<br>protected                     | gh byte:<br>v bytes                          | 5                          |                      |                            |                                  | KeyLink III Configure Reset  | www.distalary.com |

Utilizing the default values in the **Data** group box and clicking the **Write** button and then the **Read** button, a repeating data pattern is first written to and then read back from the inserted key. As per the default values, writing/reading begins at address zero and continues sequentially for 128 bytes.

Because none of the **Hex** checkboxes in the Data group box were checked, the specified **Start Address** and **Number of Bytes** are interpreted as decimal values and the **Data Pattern** is converted to ASCII data.



| Se | ries III | Programn    | ner Sa | mple  | Срр  | App  |      |      |       |         |     |    |     |      |     |    |    |                      |
|----|----------|-------------|--------|-------|------|------|------|------|-------|---------|-----|----|-----|------|-----|----|----|----------------------|
| Fi | le Co    | onfigure    | Funct  | ion H | lelp |      |      |      |       |         |     |    |     |      |     |    |    |                      |
| I  | ==ADI    | DR=== =     |        |       | ===: |      |      |      | = HEX | x ==    |     |    |     |      |     |    |    | == ASCII DATA ==     |
|    | 0000     | 0000:7      | AB C   | DEF   | 01   | AB   | CD   | EF   | 01    | AB      | CD  | EF | 01  | AB   | CD  | EF | 01 |                      |
|    | 0000     | 0010:7      | ABC    | DEF   | 01   | AB   | CD   | EF   | 01    | AB      | CD  | EF | 01  | AB   | CD  | EF | 01 |                      |
|    | 0000     | 0020.7      | AB C   | DEF   | 01   | AB   | CD   | EF   | 01    | AB      | CD  | EF | 01  | AB   | CD  | EF | 01 |                      |
|    | 0000     | 0040:#      | AB C   | DEF   | 01   | AB   | CD   | EF   | 01    | AB      | CD  | EF | 01  | AB   | CD  | EF | 01 |                      |
|    | 0000     | 0050:7      | AB C   | D EF  | 01   | AB   | CD   | EF   | 01    | AB      | CD  | EF | 01  | AB   | CD  | EF | 01 |                      |
|    | 0000     | 0060:7      | ABC    | DEF   | 01   | AB   | CD   | EF   | 01    | AB      | CD  | EF | 01  | AB   | CD  | EF | 01 |                      |
|    | 0000     | 0070.2      | 1D C.  | о Бе  | 01   | AD   | CD   | 51   | 01    | AD      | CD  | D1 | 01  | AD   | CD  | 51 | 01 |                      |
| Ľ  |          |             |        |       |      |      |      |      |       |         |     |    |     |      |     |    |    |                      |
|    | Dat      | a           |        |       |      |      |      |      |       |         |     |    |     |      |     |    |    | Erase                |
|    | S        | tart Addres | s      | 0     |      | Г    | He   | ex _ | ١     | Write   |     |    |     | Rea  | d   |    |    | Sector Erase 0       |
|    | N        | umber of B  | lytes  | 128   |      | ſ    | Ηe   | ex _ | 1     | /erify  |     |    | age | Size |     |    |    | BULK Erase           |
|    | D        | ata Pattern | 1      | ABC   | DEFO | II F | ▼ Не | ex   | Pag   | ge Wr   | ite |    | 256 |      | byt | es | Γ  | Electronic Signature |
|    |          |             |        |       |      |      |      |      | File  | e Write | ə   |    |     |      |     |    |    | Read                 |

Utilizing the same default values in the **Data** group box, but checking the **Hex** checkbox associated with **Data Pattern** and then clicking the **Write** button and then the **Read** button, would display the data shown above. Note that in this case the **Data Pattern** would be interpreted as a sequence of hexadecimal values and written directly to the key as binary, without being converted to its equivalent ASCII codes.

**CAUTION:** Attempting to send general text as the **Data Pattern** and telling the application to interpret it as hexadecimal characters will yield seemingly unpredictable behavior.

If, within the **Data** group box, the **Page Write** button is clicked instead of the **Write** button, the **Number of Bytes** to be written must be less than or equal to the **Page Size** of the device. The **Page Size** is displayed within the **Data** group box.

If, within the **Data** group box, the **File Write**... button is clicked instead of the **Write** button, the application will prompt for and allow browsing to the file to be written.



# Manual Erasure (Serial Flash only)

Here's an example of the type of behavior that can be expected if a Serial Flash device is not manually erased before being rewritten. It is assumed that the first example of writing 128 bytes of data to the key was, in fact, performed.

| Series III Programmer Sample Cpp  | Арр   |  |  |  |  |  |  |  |  |
|---|---|--|--|--|--|--|--|--|--|
| File Configure Function Help  |   |  |  |  |  |  |  |  |  |
| 0000 0020:41 42 43 44<br>0000 0030:41 42 43 44<br>0000 0040:41 42 43 44<br>0000 0050:41 42 43 44<br>0000 0050:41 42 43 44<br>0000 0060:41 42 43 44<br>================================= | 45       46       30       31       41       42       43       44       45       46       30       31       A1         45       46       30       31       41       42       43       44       45       46       30       31       A1         45       46       30       31       41       42       43       44       45       46       30       31       A1         45       46       30       31       41       42       43       44       45       46       30       31       A1         45       46       30       31       41       42       43       44       45       46       30       31       A1         45       46       30       31       41       42       43       44       45       46       30       31       A1         45       46       30       31       41       42       43       44       45       46       30       31       A1         45       46       30       31       41       42       43       44       45       46       30       31       A1 | BCDEF01ABCDEF01<br>BCDEF01ABCDEF01<br>BCDEF01ABCDEF01<br>BCDEF01ABCDEF01<br>BCDEF01ABCDEF01<br>BCDEF01ABCDEF01<br>==================================== |  |  |  |  |  |  |  |
|   |   |  |  |  |  |  |  |  |  |
| Start Address 64  | Hex Write Read  | Sector Erase 0   |  |  |  |  |  |  |  |
| Number of Bytes 64  | Hex Verify Page Size  | BULK Erase   |  |  |  |  |  |  |  |
| Data Pattern TestData   | Hex Page Write 256 bytes  | ctronic Signature  |  |  |  |  |  |  |  |
|   | File Write  | Read   |  |  |  |  |  |  |  |

Without erasing the initial data, perform the following additional operations:

- enter the values shown above in the Data group box for Start Address, Number of Bytes, and Data Pattern,
- click the **Write** button,

| Series III Programmer Sample Cpp App   |   |
|--|---|
| File Configure Function Help   |   |
|  | - |
| 0000 0000:41 42 43 44 45 46 30 31 41 42 43 44 45 46 30 31 ABCDEF01ABCDEF0  | 1 |
| 0000 0010:41 42 43 44 45 46 30 31 41 42 43 44 45 46 30 31 ABCDEF01ABCDEF0  | 1 |
| 0000 0020:41 42 43 44 45 46 30 31 41 42 43 44 45 46 30 31 ABCDEF01ABCDEF0  | 1 |
| 0000 0030:41 42 43 44 45 46 30 31 41 42 43 44 45 46 30 31 ABCDEF01ABCDEF0  |   |
| 0000 0040:40 40 43 44 44 40 30 21 40 40 43 44 44 40 30 21 @@CDD@0:@@CDD@0<br>0000 0050.40 40 43 44 44 40 30 21 40 40 43 44 44 40 30 21 @@CDD@0!@@CDD@0 |   |
| 0000 0060:40 40 43 44 44 40 30 21 40 40 43 44 44 40 30 21 @@CDD@0!@@CDD@0  |   |
| 0000 0070:40 40 43 44 44 40 30 21 40 40 43 44 44 40 30 21 @@CDD@0!@@CDD@0  | ! |
|  |   |
| Data   |   |
| Start Address 0  | ) |
| Number of Bytes 128 THex Verify Page Size  |   |
| Data Pattern TestData Hex Page Write 256 bytes Electronic Signature  |   |
| File Write Read  |   |

- enter the values shown above in the Data group box for Start Address and Number of Bytes, and
- click the **Read** button.



Now look at the 128 bytes of data that was read from the device. The first 64 bytes still exhibit the same repeating pattern of data that was written to them initially, but the final 64 bytes show neither the initial nor the subsequent data pattern. What's going on here?

The answer is that the following rules are obeyed when overwriting data in a Serial Flash device:

- binary ones in the initial data can remain as binary ones in the subsequent data,
- binary ones in the initial data can be demoted to binary zeroes in the subsequent data,
- binary zeroes in the initial data can remain as binary zeroes in the subsequent data, but
- binary zeroes in the initial data cannot be promoted to binary ones in the subsequent data.

More often than not, what is wanted is to manually erase the previous data before writing new data to any given memory location. That is the only way to promote binary zeroes in the initial data back to binary ones. When any byte is erased, all its bits go to binary ones.

The scope of an erasure can be limited to a single sector at a time or applied to the entire memory space simultaneously. This can be done using the controls in the **Erase** group box. To erase a single sector, specify the sector number and then click the **Sector Erase** button. To erase the entire memory space, click the **Bulk Erase** button.

### **Block Protect (SPI only)**

Sections of an SPI EEPROM/Flash memory space can be protected from writing and erasure by utilizing the Block Protect bits, BP0 and BP1. Depending on the memory capacity of the device, there may even be a third Block Protect bit, BP2. Only SPI memory offers Block Protect bits.

SPI EEPROM and SPI Flash memory devices have their memory space organized into sectors (i.e., blocks or chunks) of uniform size. That is, the size is uniform throughout any given device, but the number of sectors and the size of a sector may vary from one device to another – depending on the total memory capacity of the device. Setting various combinations of the Block Protect bits write-protects various combinations of sectors in the memory space. When all of the Block Protect bits are cleared, none of the sectors is writeprotected. Setting all of the available Block Protect bits writeprotects the entire memory space of the device. The table to the right shows which sectors are write-protected (for devices of various memory capacities) based on the combination of Block Protect bits that are set/cleared.



| Block | Protec | t Bits                | Protected Sectors |          |            |            |              |  |
|-------|--------|-----------------------|-------------------|----------|------------|------------|--------------|--|
|       |        |                       | ≤ 2-Mbit          | 4-Mbit   | 8-Mbit     | 32-Mbit    | 64-Mbit      |  |
| BP2   | BP1    | BP1 BP0 Device Device | Device            | Device   | Device     | Device     |              |  |
| 0     | 0      | 0                     | None              | None     | None       | None       | None         |  |
| 0     | 0      | 1                     | 3                 | 7        | 15         | 63         | 126 and 127  |  |
| 0     | 1      | 0                     | 2 and 3           | 6 and 7  | 14 and 15  | 62 and 63  | 124 thru 127 |  |
| 0     | 1      | 1                     | All               | 4 thru 7 | 12 thru 15 | 60 thru 63 | 120 thru 127 |  |
| 1     | 0      | 0                     | n/a               | All      | All        | 56 thru 63 | 112 thru 127 |  |
| 1     | 0      | 1                     | n/a               | All      | All        | 48 thru 63 | 96 thru 127  |  |
| 1     | 1      | 0                     | n/a               | All      | All        | 32 thru 63 | 64 thru 127  |  |
| 1     | 1      | 1                     | n/a               | All      | All        | All        | All          |  |

Sector numbering begins at zero – corresponding to the lowest memory address. Sectors are protected in a top-down fashion. The top-most sector(s) represents the smallest portion of the total memory that can be protected.

Note that when initially displayed, the **Block Protect (SPI)** group box indicates that neither (none) of the Block Protect bits is set – this indication is not necessarily accurate. Click on the **Read BP Bits** button to update the status of the BP0, BP1 (and BP2) checkboxes so that they will be accurate. A checked box indicates that the corresponding bit is set (i.e., high, one) and an unchecked box indicates that the corresponding bit is cleared (i.e., low, zero). The status of the Block Protect bits can be set manually by checking/clearing the BP0, BP1 (and BP2) checkboxes as desired and then clicking the **Write BP Bits** button.

Note that when a checkbox is checked/cleared, the graphics are updated immediately, but the Block Protect bits remain unaltered until the **Write BP Bits** button is clicked.

### **Address Pattern**

If no data is placed within the **Data Pattern** text box, the application will automatically generate a sequence of increasing data values for use as the data pattern and will re-label the text box **Address Pattern**.

== ASCII DATA == 0000 0000:30 30 30 30 30 30 30 30 30 30 30 30 36 30 30 30 30 000000000060000 0000 0010:30 43 30 30 30 30 31 32 30 30 30 30 31 38 30 30 0C00001200001800 0000 0020:30 30 31 45 30 30 30 30 32 34 30 30 30 30 32 41 001E00002400002A 0000 0030:30 30 30 30 33 30 30 30 30 30 30 33 36 30 30 30 30 0000300000360000 0000 0040:33 43 30 30 30 30 34 32 30 30 30 30 34 38 30 30 3C00004200004800 0000 0050:30 30 34 45 30 30 30 30 35 34 30 30 30 35 41 004E00005400005A 0000 0060:30 30 30 30 36 30 30 30 30 30 36 36 30 30 30 30 0000600000660000 0000 0070:36 43 30 30 30 30 37 32 30 30 30 30 37 38 30 30 6C00007200007800

Shown here is a sample of the address pattern that the application automatically generates.

## **Electronic Signature (Serial Flash only)**

Only Serial Flash devices have electronic signatures. To read the electronic signature of a Serial Flash device, click the **Read** button within the **Electronic Signature** group box. (See the SPI Flash Interface Specification for more information.)

| Read | 17 (0x11 hex) |
|------|---------------|
|------|---------------|

10









#### Menu Bar

From within the **File** menu, the only two options are clearing the message window (**Clear Screen**) or closing the Sample Cpp App (**Exit**).

From within the **Configure** menu, the 5-character configuration code (which is always visible in the information bar at the bottom of the GUI) can be displayed in the message window by clicking **View Programmer Configuration**.

Clicking **View Firmware Revision** will display a message similar to what is shown here.

From within the **Function** menu, the usual operations as well as some additional operations are available. **Read Serial Number** is only applicable to IIK/IIT devices. **Write** will permit entering a longer sequence of values, be they ASCII or hexadecimal (see the next screen shot).

This is the window displayed by clicking **Write** within the **Function** menu. An example of a longer ASCII character string is shown.

For information about the Sample Cpp App, click **About** within the **Help** menu (see the next screen shot). To display a list of all device descriptors and their associated 5-character codes, click **Codes**.

Clicking About in the Help menu will display a message similar to this.

| File | Configure Funct         | ion Help |  |  |  |  |
|------|-------------------------|----------|--|--|--|--|
|      | Clear Screen            |          |  |  |  |  |
|      | Exit                    |          |  |  |  |  |
| File | Configure Function Help |          |  |  |  |  |
|      | View Programmer Config  | uration  |  |  |  |  |
|      | View Firmware Revision  |          |  |  |  |  |

READING FIRMWARE REVISION . . . Keylink III V01.20 Copyright(C) 2008-2010 DKE Inc

| File Configure | Function Help          |
|----------------|------------------------|
|                | Read<br>Write<br>Erase |
|                | Read Size              |
|                | Read Status            |
|                | Read Serial Number     |



| File Configure Function   | on Help  |  |  |  |  |  |  |  |
|---|--|--|--|--|--|--|--|--|
|   | About  |  |  |  |  |  |  |  |
|   | Codes  |  |  |  |  |  |  |  |
| About Series III Programmer Sample Cpp App                          |  |  |  |  |  |  |  |  |
|   |  |  |  |  |  |  |  |  |
| Datakey Series III Programmer Sample                                | Datakey Series III Programmer Sample Cop App VS2022, Version 2.2.0.0 |  |  |  |  |  |  |  |
| Keylink III DLL Version 5.0.0.0                                     |  |  |  |  |  |  |  |  |
| Copyright (0) 2024 ATEK Access Technologies LLC All rights received |  |  |  |  |  |  |  |  |
|   | ninologios, EEG. Airlights rosorroo.                                 |  |  |  |  |  |  |  |
| ОК  |  |  |  |  |  |  |  |  |



### **C# and Visual Basic Implementation**

To start the application, do the following:

- attach the Series III Programmer to the host computer's USB 2.0 port, then
- double-click the Series III Programmer desktop icon on the host computer.

### Configuration

Each time the sample application is started, it will prompt the user to configure the attached Series III Programmer for the type of device that is to be expected. This is a crucial step, since it tells the Series III Programmer:

- what protocol is to be used for communication,
- what supply voltage to apply to the device (either 3.3 or 5 volts), and
- the memory capacity of the device.

The sample application cannot query this information from an attached device, it must be told what to expect. The Series III Programmer does remember the type of device for which it was last configured and the corresponding 5-character code is displayed in the **Code** text box of the **Configure Programmer** form.

Expand the appropriate protocol family and select the specific device to be used in the Series III Programmer. Note that the same 5-character code may be assigned to more than one particular device in the list. That's because the electrical interface to the devices may be the same, even though their physical shapes are different.

Once the desired configuration has been selected, click the **OK** button. In this example the Series III Programmer is being configured for an SFK 2Mb device.

Note that if the Series III Programmer is configured for a Serial Flash device, a message will be displayed, reminding the user to manually erase the device before overwriting any existing data. EEPROM devices implement an automatic erase feature whenever new data is written; Serial Flash does not.

| lease Note:                         |                           |
|-------------------------------------|---------------------------|
| Remember to erase your Serial Flash | device before re-writing. |
|                                     |                           |

All of the Datakey devices that utilize Serial Flash memory have family names that begin with the letters "SF" (e.g., SFK, SFT, SFX).



| Make a selection   | keycodes list if you  | ur part is not in the list.   | code nom die |
|--|---|---|--------------|
| <ul> <li>Microwire (DB</li> <li>I2C (IOT, ISE</li> <li>I2C Serialize</li> <li>SPI EEPROM (S</li> <li>SPI Flash State</li> <li>SPI Flash State</li> </ul> | <ul> <li>K, KSD, LCB, LG</li> <li>S, ISK, ISP, IS</li> <li>ed (IIK, IIT, IS</li> <li>SGT, SLK, SLT,</li> <li>andard 3.3V (SP</li> <li>(SFK5V, SFT5V,</li> </ul> | TK, LCS, LCX)<br>ST, ISX)<br>HIX)<br>SSB, SSK, SSM, SSP,<br>FK, SFT, SFX)<br>SFX5V) | SST, SSX)    |
| Type:<br>Product:<br>Shape:  |   |   |              |
| Voltage:<br>Code:  | SF3GL   | List Keycodes   |              |
|  |   |   |              |

| Make a selection<br>k | from the list below<br>eycodes list if you | or type the 5 character code<br>r part is not in the list. | from the |
|-----------------------|--|--|----------|
| SPI Flash Sta         | ndard 3.3V (SF                             | K, SFT, SFX)   |          |
| -SFK 1Mb              |  | (SF3GK)  |          |
| SFK 2Mb               |  | (SF3GL)  | -        |
| -SFK 4Mb              |  | (SF3GM)  |          |
| -SFK 8Mb              |  | (SF3GN)  |          |
| -SFK 16Mb             |  | (SF3GO)  |          |
| -SFK 32Mb             | (SF3GP)                                    |  |          |
| -SFT 1Mb              |  | (SF3GK)  |          |
| -SFT 2Mb              |  | (SF3GL)  | -        |
| Type:                 | SPI Flash Stand                            | ard 3.3V   | 1        |
| Product:              | SFK 2Mb                                    |  | -        |
| Shape:                | Key - Round He                             | ad   | 1        |
| Voltage:              | 3V   |  | 1        |
| Code:                 | SF3GL                                      | List Keycodes  | 1        |



#### **Overview**

| 🣕 Series III Progra   | ammer Sample \   | /B App - Connected  | 1                                  |   | - 🗆 X                         |
|---|--|---|------------------------------------|---|-------------------------------|
| Transcript W<br>Selected con<br>Type:<br>Product<br>Shape:<br>Voltage<br>Code:<br>Success writ<br>Remember to | indow:<br>figuration:<br>SPI Flas<br>: SFK 2Mb<br>Key - Ro<br>: 3V<br>SF3GL<br>ing new cor<br>erase your | ch Standard 3<br>ound Head<br>figuration:<br>Serial Flash | .3V<br>SF3GL<br>device before re-w | writing.  |                               |
| Data<br>Start Address<br>Number of Bytes<br>Data Pattern  | 0<br>128<br>ABCDEF01   | Hex Wr  | te Read<br>Write Verify<br>Vrite   | Erase<br>Sector Erase<br>Bulk Erase<br>Transcript |                               |
| Write Extended D  | Data Hex   | Block Protect   | 0 Read BP Bits<br>Write BP Bits    | KeyLink III<br>Set Config<br>Reset Detect         | nmer<br>ted                   |
|   |  |   |                                    |   | www.datakey.com               |
| VB App VS2022   | Version  | 2.2.0.0   | Copyright © 2024                   | DLL Version 5.0.0.0                               | Firmware Keylink III V01.20.0 |
| Config Code: SF3GL  | Key Abs  | ent   | Block Prot: none                   | Elec Sig: none                                    | Serial No: none               |
| Family: SPI   | Memory   | : Flash   | Voltage: 3 volts                   | Page Size: 256 bytes                              | 0 bytes                       |

The **Sample VB App** pictured above has:

- a **Transcript Window** at the top that shows the results of operations recently performed and the general status of the application,
- a **Data** group box where address limits and data patterns can be specified and where read, write, and compare operations can be performed,
- an Erase group box where individual sectors or the entire device can be targeted for erasure,
- a Block Protect group box where the block protect bits of an SPI device can be read and set/cleared,
- a **KeyLink III** group box where the Series III Programmer can be reconfigured or reset and where the on-line/off-line status of the Series III Programmer can easily be determined,
- an Extended Data window where larger amounts of data can be entered and written to the device, and
- a status grid where information about the software, firmware, configuration, and hardware status are displayed including an electronic signature for Serial Flash devices (see the SPI Flash Interface Specification).

Note that not all controls are available or appropriate for all types of devices.



### Insertion

When a key is inserted into the receptacle, an image of a key will be displayed within the **KeyLink III** group box. (Note that for an actual key type of device, the key must be inserted into the receptacle and turned a quarter turn clockwise before the image of a key will appear.) Behind the scenes, the application is polling the status of the LOFO (last on, first off) signal and when it goes low, the image is displayed.

| 📜 Series III Programmer Sam  | nple VB App - Connected   |  |   |                                  |
|--|---|--|---|----------------------------------|
| Success writing net<br>Remember to erase y<br>Read Data Fields: 9           Writing Datakey det<br>Read Data Fields: 9           =ADDR==           0000 0000:41 42 43           0000 0020:41 42 43           0000 0020:41 42 43           0000 0030:41 42 43           0000 0050:41 42 43           0000 0050:41 42 43           0000 0070:41 42 43           0000 0070:41 42 43           0000 0070:41 42 43           0000 0070:41 42 43           0000 0070:41 42 43           0000 0070:41 42 43           0000 0070:41 42 43           0000 0070:41 42 43 | <pre>w configuration: SF3 your Serial Flash de Start Address = 0 By vice: succeeded Start Address = 0 By =========== HEX == 44 45 46 30 31 41 44 45 46 30 31 41 44 45 46 30 31 41 44 45 46 30 31 41 44 45 46 30 31 41 44 45 46 30 31 41 vice: succeeded</pre> | GL<br>vice before re-writi<br>te Count = 128<br>te Count = 128<br>42 43 44 45 46 30 31<br>42 43 44 45 46 30 31 | ng.<br>= == ASCII DATA ==<br>ABCDEF01ABCDEF01<br>ABCDEF01ABCDEF01<br>ABCDEF01ABCDEF01<br>ABCDEF01ABCDEF01<br>ABCDEF01ABCDEF01<br>ABCDEF01ABCDEF01<br>ABCDEF01ABCDEF01 |                                  |
| Data<br>Start Address 0<br>Number of Bytes 128<br>Data Pattern ABCDE   | Image: Hex     Hex     Page       Image: Hex     Page       EF01     Image: Hex     File V       Block Protect     Block Protect  | te Read<br>Write Verify<br>Vrite   | Erase<br>Sector Erase 0<br>Bulk Erase<br>Transcript V Hex<br>KeyLink III  | Datakey.                         |
| Write Extended Data  | Hex DP1 BP  | Read BP Bits   | Reset   | www.dubley.com ww                |
| VB App VS2008  | Version 2.1.0.0   | Copyright © 2016   | DLL Version 4.0.0.0   | Firmware Keylink III V01.20 Copy |
| Config Code: SF3GL   | Key Present   | Block Prot: 0 0  | Elec Sig: 0x11  | Serial No: none                  |
| Family: SPI  | Memory: Flash   | Voltage: 3 volts   | Page Size: 256 bytes  | 262144 bytes                     |

Note that in addition to the key image, a green status bar is also displayed. Additionally, none of the status grid blocks are pink – most of them are now green.

Utilizing the default values in the **Data** group box and clicking the **Write** button and then the **Read** button, a repeating data pattern is first written to and then read back from the inserted key. As per the default values, writing/reading begins at address zero and continues sequentially for 128 bytes.

Because none of the **Hex** checkboxes in the **Data** group box were checked, the specified **Start Address** and **Number of Bytes** are interpreted as decimal values and the **Data Pattern** is converted to ASCII data.



| 🧧 Series III Program | mer Sample | VB App - | Connect | ted        |          |        |        | _ <b>D</b> X     |
|----------------------|------------|----------|---------|------------|----------|--------|--------|------------------|
| Transcript Wi        | ndow:      |          |         |            |          |        |        |                  |
| Read Data Fie        | lds: Sta   | rt Addı  | cess =  | = 0 Byte   | Count    | = 128  |        |                  |
| Writing Datak        | ey devic   | e: suco  | ceedeo  |            | Count    | - 100  |        |                  |
| Read Data Fie        | 105: Sta   | rt Addi  | ess =   | = 0 Byte   | Count    | = 128  |        | ASCII DATA       |
| 0000 0000:AB         | CD EF 01   | AB CD    | EF 01   | AB CD      | EF 01    | AB CD  | EF 01  | ASCII DAIA       |
| 0000 0010:AB         | CD EF 01   | AB CD    | EF 01   | AB CD      | EF 01    | AB CD  | EF 01  |                  |
| 0000 0020:AB         | CD EF 01   | AB CD    | EF 01   | AB CD      | EF 01    | AB CD  | EF 01  |                  |
| 0000 0030:AB         | CD EF 01   | AB CD    | EF 01   | AB CD      | EF 01    | AB CD  | EF 01  |                  |
| 0000 0040:AB         | CD EF 01   | AB CD    | EF 01   | AB CD      | EF 01    | AB CD  | EF 01  |                  |
| 0000 0050:AB         | CD EF 01   | AB CD    | EF 01   | AB CD      | EF 01    | AB CD  | EF 01  |                  |
| 0000 0060:AB         | CD EF 01   | AB CD    | EF 01   | AB CD      | EF 01    | AB CD  | EF 01  |                  |
| Dood 00/0:AB         | CD EF 01   | AB CD    | EF 01   | AB CD      | EF. 01   | AB CD  | EF. 01 |                  |
| Reading Datak        | ey devic   | e. succ  | Leeueu  |            |          |        |        |                  |
|                      |            |          |         |            |          |        |        |                  |
| Data                 |            |          |         |            |          |        |        | Erase            |
| Start Address        | 0          |          | Hex     | Write      |          | Read   | ו ר    | Sector Erase 0   |
|                      |            |          |         |            |          |        |        |                  |
| Number of Bytes      | 128        |          | Hex     | Page Write | <u> </u> | Verify |        | Bulk Erase       |
| Data Pattern         | ABCDEF01   | ₹        | Hex     | File Write |          |        |        | Transcript 🔽 Hex |

Utilizing the same default values in the **Data** group box, but checking the **Hex** checkbox associated with **Data Pattern** and then clicking the **Write** button and then the **Read** button, would display the data shown above. Note that in this case the **Data Pattern** would be interpreted as a sequence of hexadecimal values and written directly to the key as binary, without being converted to its equivalent ASCII codes.

**CAUTION:** Attempting to send "TestData" as the **Data Pattern** and telling the application to interpret it as hexadecimal characters will throw an unhandled exception. If this happens, click the **Continue** button.

If, within the **Data** group box, the **Page Write** button is clicked instead of the **Write** button, the **Number of Bytes** to be written must be less than or equal to the **Page Size** of the device. The **Page Size** is displayed in the bottom row of the status grid.

If, within the **Data** group box, the **File Write** button is clicked instead of the **Write** button, the application will prompt for and allow browsing to the file to be written.





# Manual Erasure (Serial Flash only)

| 🣕 Series III Programm | mer Sample VB Ap             | op - Connected |  |                   |
|-----------------------|------------------------------|----------------|--|-------------------|
| Read Data Fie         | lds: Start P                 | ddress = (     | 0 Byte Count = 128                                   | ▲                 |
| Read Data Fie         | ey device: s<br>lds: Start Z | ddress = 1     | Byte Count = $128$                                   |                   |
| ==ADDR=== ===         | ==========                   | ===== HEX      | <pre>&lt; ====================================</pre> | == ASCII DATA ==  |
| 0000 0000:41          | 42 43 44 45                  | 46 30 31       | 41 42 43 44 45 46 30 31                              | ABCDEF01ABCDEF01  |
| 0000 0010:41          | 42 43 44 45                  | 46 30 31       | 41 42 43 44 45 46 30 31                              | ABCDEF01ABCDEF01  |
| 0000 0020:41          | 42 43 44 45                  | 46 30 31       | 41 42 43 44 45 46 30 31                              | ABCDEF01ABCDEF01  |
| 0000 0040:41          | 42 43 44 45                  | 46 30 31       | 41 42 43 44 45 46 30 31                              | ABCDEF01ABCDEF01  |
| 0000 0050:41          | 42 43 44 45                  | 46 30 31       | 41 42 43 44 45 46 30 31                              | ABCDEF01ABCDEF01  |
| 0000 0060:41          | 42 43 44 45                  | 46 30 31       | 41 42 43 44 45 46 30 31                              | ABCDEF01ABCDEF01  |
| 0000 0070:41          | 42 43 44 45                  | 46 30 31       | 41 42 43 44 45 46 30 31                              | ABCDEF01ABCDEF01  |
| Reading Datak         | ey device: s                 | ddress =       | A Bute Count = 64                                    |                   |
| Writing Datak         | ev device: s                 | ucceeded       | by byte count - 64                                   |                   |
|                       |                              |                |  | -                 |
| Data                  |                              |                |  | Erase             |
| Start Address         | 64                           | □ Hex          | Write Read   | Sector Erase 0    |
| Number of Bytes       | 64                           | □ Hex          | Page Write Verify                                    | Bulk Erase        |
| Data Pattern          | TestData                     | □ Hex          | File Write   | Transcript IF Hex |

Here's an example of the type of behavior that can be expected if a Serial Flash device is not manually erased before being rewritten. It is assumed that the first example of writing 128 bytes of data to the key was, in fact, performed.

Now (without erasing the initial data) perform the following additional operations:

- enter the values shown above in the Data group box for Start Address, Number of Bytes, and Data Pattern,
- click the **Write** button,



| 📕 Series III Program | mer Sample VB Ap            | op - Connecte | d                       |                    |
|----------------------|-----------------------------|---------------|-------------------------|--------------------|
| 0000 0070:41         | 42 43 44 45                 | 46 30 31      | 41 42 43 44 45 46 30 31 | ABCDEF01ABCDEF01   |
| Reading Datak        | ey device: s                | ucceeded      |                         |                    |
| Read Data Fie        | Ids: Start A                | ddress =      | 64 Byte Count = $64$    |                    |
| Writing Datak        | lds: Start A                | ddress -      | 0 Bute Count - 129      |                    |
| ==ADDR=== ===        |                             | ====== HI     | 3 Byte count = 128      | = == ASCII DATA == |
| 0000 0000:41         | 42 43 44 45                 | 46 30 31      | 41 42 43 44 45 46 30 31 | ABCDEF01ABCDEF01   |
| 0000 0010:41         | 42 43 44 45                 | 46 30 31      | 41 42 43 44 45 46 30 31 | ABCDEF01ABCDEF01   |
| 0000 0020:41         | 42 43 44 45                 | 46 30 31      | 41 42 43 44 45 46 30 31 | ABCDEF01ABCDEF01   |
| 0000 0030:41         | 42 43 44 45                 | 46 30 31      | 41 42 43 44 45 46 30 31 | ABCDEF01ABCDEF01   |
| 0000 0040:40         | 40 43 44 44                 | 40 30 21      | 40 40 43 44 44 40 30 21 | 00CDD00!00CDD00!   |
| 0000 0050:40         | 40 43 44 44                 | 40 30 21      | 40 40 43 44 44 40 30 21 | 00CDD00!00CDD00!   |
| 0000 0060:40         | 40 43 44 44                 | 40 30 21      | 40 40 43 44 44 40 30 21 |                    |
| Reading Datak        | 40 43 44 44<br>ev device: e | 40 30 ZI      | 40 40 43 44 44 40 30 2  |                    |
| Reauting Datak       | ey device. S                | ucceeded      |                         | -                  |
| Data                 |                             |               |                         | Erase              |
| Start Address        | 0                           | □ Hex         | Write Read              | Sector Erase 0     |
| Number of Bytes      | Bytes 128                   |               | Page Write Verify       | Bulk Erase         |
| Data Pattern         | TestData                    | □ Hex         | File Write              | Transcript F Hex   |

- enter the values shown above in the Data group box for Start Address and Number of Bytes, and
- click the **Read** button.

Now look at the 128 bytes of data that was read from the device. The first 64 bytes still exhibit the same repeating pattern of data that was written to them initially, but the final 64 bytes show neither the initial nor the subsequent data pattern. What's going on here?

The answer is that the following rules are obeyed when overwriting data in a Serial Flash device:

- binary ones in the initial data can remain as binary ones in the subsequent data,
- binary ones in the initial data can be demoted to binary zeroes in the subsequent data,
- binary zeroes in the initial data can remain as binary zeroes in the subsequent data, but
- binary zeroes in the initial data cannot be promoted to binary ones in the subsequent data.

More often than not, what is wanted is to manually erase the previous data before writing new data to any given memory location. That is the only way to promote binary zeroes in the initial data back to binary ones. When any byte is erased, all its bits go to binary ones.

The scope of an erasure can be limited to a single sector at a time or applied to the entire memory space simultaneously. This can be done using the controls in the **Erase** group box. To erase a single sector, specify the sector number and then click the **Sector Erase** button. To erase the entire memory space, click the **Bulk Erase** button.



# **Block Protect (SPI only)**

Sections of an SPI EEPROM/Flash memory space can be protected from writing and erasure by utilizing the Block Protect bits, BPO and BP1. Depending on the memory capacity of the device, there may even be a third Block Protect bit, BP2. Only SPI memory offers Block Protect bits.

SPI EEPROM and SPI Flash memory devices have their memory space organized into sectors (i.e., blocks or chunks) of uniform size. That is, the size is uniform throughout any given device, but the number of sectors and the size of a sector may vary from one device to another – depending on the total memory capacity of the device. Setting various combinations of the Block Protect bits write-protects various combinations of sectors in the memory space. When all of the Block Protect bits are cleared, none of the sectors is write-protected. Setting all of the available

Block Protect bits write-protects the entire memory space of the device. The table to the right shows which sectors are write-protected (for devices of various memory capacities) based on the combination of Block Protect bits that are set/cleared.

Sector numbering begins at zero – corresponding to the lowest memory address. Sectors are protected in a top-down fashion. The top-most sector(s) represents the smallest portion of the total memory that can be protected.

Note that when initially displayed, the **Block Protect** group box indicates that neither (none) of the Block Protect bits is set – this indication is not necessarily accurate. Click on the **Read BP Bits** button to update the status of the BP0, BP1 (and BP2) checkboxes so that they will be accurate. A checked box indicates that the corresponding bit is set (i.e., high, one) and an unchecked box indicates that the corresponding bit is cleared (i.e., low, zero). The status of the Block Protect bits can be set manually by checking/ clearing the BP0, BP1 (and BP2) checkboxes as desired and then clicking the **Write BP Bits** button.

Note that when a checkbox is checked/cleared, the graphics are updated immediately, but the Block Protect bits remain unaltered until the **Write BP Bits** button is clicked.

| DIOCKTTOLOCI |     |               |
|--------------|-----|---------------|
| BP1          | BP0 | Read BP Bits  |
|              |     | Write BP Bits |

| Block | Protec  | t Bits | Protected Sectors |          |            |            |              |  |  |  |
|-------|---------|--------|-------------------|----------|------------|------------|--------------|--|--|--|
| 0.00  | BP2 BP1 |        | ≤ 2-Mbit          | 4-Mbit   | 8-Mbit     | 32-Mbit    | 64-Mbit      |  |  |  |
| BPZ   |         | BPO    | Device            | Device   | Device     | Device     | Device       |  |  |  |
| 0     | 0       | 0      | None              | None     | None       | None       | None         |  |  |  |
| 0     | 0       | 1      | 3                 | 7        | 15         | 63         | 126 and 127  |  |  |  |
| 0     | 1       | 0      | 2 and 3           | 6 and 7  | 14 and 15  | 62 and 63  | 124 thru 127 |  |  |  |
| 0     | 1       | 1      | All               | 4 thru 7 | 12 thru 15 | 60 thru 63 | 120 thru 127 |  |  |  |
| 1     | 0       | 0      | n/a               | All      | All        | 56 thru 63 | 112 thru 127 |  |  |  |
| 1     | 0       | 1      | n/a               | All      | All        | 48 thru 63 | 96 thru 127  |  |  |  |
| 1     | 1       | 0      | n/a               | All      | All        | 32 thru 63 | 64 thru 127  |  |  |  |
| 1     | 1       | 1      | n/a               | All      | All        | All        | All          |  |  |  |



### **Address Pattern**

If no data is placed within the **Data Pattern** text box, the application will automatically generate a sequence of increasing data values for use as the data pattern and will re-label the text box **Address Pattern**.

| Start Address   | 0   | □ Hex | Write      | Read   |
|-----------------|-----|-------|------------|--------|
| Number of Bytes | 128 | ⊢ Hex | Page Write | Verify |
| Address Pattern |     | □ Hex | File Write |        |

Shown here is a sample of the address pattern that the application automatically generates.

| Read  | Data  | Fie | lds | 3: 5 | Star | t I | ddr | ess | 5 =  | 0 By | te | Cou | int | = 1 | 28 |    |     |                  |
|-------|-------|-----|-----|------|------|-----|-----|-----|------|------|----|-----|-----|-----|----|----|-----|------------------|
| ==ADI | DR=== | === |     |      |      |     |     |     | = HE | X == |    |     |     |     |    |    | === | == ASCII DATA == |
| 0000  | 0000: | 30  | 30  | 30   | 30   | 30  | 30  | 30  | 30   | 30   | 30 | 30  | 36  | 30  | 30 | 30 | 30  | 0000000000060000 |
| 0000  | 0010: | 30  | 43  | 30   | 30   | 30  | 30  | 31  | 32   | 30   | 30 | 30  | 30  | 31  | 38 | 30 | 30  | 0C00001200001800 |
| 0000  | 0020: | 30  | 30  | 31   | 45   | 30  | 30  | 30  | 30   | 32   | 34 | 30  | 30  | 30  | 30 | 32 | 41  | 001E00002400002A |
| 0000  | 0030: | 30  | 30  | 30   | 30   | 33  | 30  | 30  | 30   | 30   | 30 | 33  | 36  | 30  | 30 | 30 | 30  | 0000300000360000 |
| 0000  | 0040: | 33  | 43  | 30   | 30   | 30  | 30  | 34  | 32   | 30   | 30 | 30  | 30  | 34  | 38 | 30 | 30  | 3C00004200004800 |
| 0000  | 0050: | 30  | 30  | 34   | 45   | 30  | 30  | 30  | 30   | 35   | 34 | 30  | 30  | 30  | 30 | 35 | 41  | 004E00005400005A |
| 0000  | 0060: | 30  | 30  | 30   | 30   | 36  | 30  | 30  | 30   | 30   | 30 | 36  | 36  | 30  | 30 | 30 | 30  | 0000600000660000 |
| 0000  | 0070: | 36  | 43  | 30   | 30   | 30  | 30  | 37  | 32   | 30   | 30 | 30  | 30  | 37  | 38 | 30 | 30  | 6C00007200007800 |

## **Extended** Data

The application's **Extended Data** window allows manual entry of larger amounts of data or a simple copy-and-paste operation. Click the **Write Extended Data** button to copy the window's contents to the key. The **Number of Bytes** written will automatically be calculated.

In order to see a plain text display of the data read from the key, clear the **Hex** checkbox in the **Erase** group box. That will prevent the application from displaying the address labels and hexadecimal values associated with the data it displays in the **Transcript Window**.

|   | Block Protect   |  | KeyLink II                                    | 20  |  |
|---|---|--|---|---|--|
|   | BP1 BP0   | Read BP Bits   | Set Config                                    | <b>2</b>                                      |  |
| Extended Data   | ГГ  | Write BP Bits  | Reset   |   |  |
| Here is a simple example<br>and then writing it to t<br>the Number of Bytes is a<br>data. | of typing alphr<br>the key. When the<br>automatically cal | umeric text into<br>Write Extended<br>culated - to fac | o the Extende<br>Data button<br>cilitate read | d Data window_<br>is clicked,<br>ing back the |  |

| Erase        |     |
|--------------|-----|
| Sector Erase | 0   |
| Bulk Erase   | ]   |
| Transcript   | Hex |

Read Data Fields: Start Address = 0 Byte Count = 248 Extended Write Datakey device: wrote 248 bytes from Extended Data starting at address 0x00000000 Read Data Fields: Start Address = 0 Byte Count = 248 Here is a simple example of typing alphnumeric text into the Extended Data window and then writing it to the key. When the Write Extended Data button is clicked, the Number of Bytes is automatically calculated - to facilitate reading back the data. Reading Datakey device: succeeded

Above is an example of what the plain text will look like in the Transcript Window.Click the **Read** button in the Data group box to display the text. To clear text from the **Transcript Window**, click the **Transcript** button in the **Erase** group box.



#### **Acronyms/Abbreviations**

| ASCII       | American Standard Code for Information Interchange  |
|-------------|---|
| EEPROM      | Electrically Erasable Programmable Read-Only Memory   |
| Gbits/bytes | Gigabits/bytes – may indicate a multiplier of 10 <sup>9</sup> (1,000,000,000) or 2 <sup>30</sup> (1,073,741,824) bits/bytes |
| GUI         | Graphical User Interface  |
| I2C         | Inter-Integrated Circuit  |
| Kbits/bytes | Kilobits/bytes – may indicate a multiplier of 10 <sup>3</sup> (1,000) or 2 <sup>10</sup> (1,024) bits/bytes                 |
| LED         | Light-Emitting Diode  |
| Mbits/bytes | Megabits/bytes – may indicate a multiplier of 10 <sup>6</sup> (1,000,000) or 2 <sup>20</sup> (1,048,576) bits/bytes         |
| SPI         | Serial Peripheral Interface   |
| USB         | Universal Serial Bus  |

#### 223-0100-000 Rev. F 6/24

Microsoft is a registered trademark of Microsoft Corporation. Windows is a trademarks of Microsoft Corporation. All other brand names and product names used in this manual are trademarks, registered trademarks, or trade names of their respective holders.

ATEK Access Technologies 10025 Valley View Road, Ste. 190 Eden Prairie, MN 55344 U.S.A. 1.800.523.6996 : 1.800.589.3705 +1.218.829.9797

www.atekaccess.com

©2024 ATEK Access Technologies, LLC. All Rights Reserved. Datakey images and marketing materials are protected by various patents, copyrights and/or trademarks. ATEK assumes no responsibility for any errors which may appear in this document, reserves the right to change devices or specifications detailed herein at any time without notice, and does not make any commitment to update the information contained herein. No licenses to patents or other intellectual property of ATEK are granted by the Company in connection with the sale of Datakey products, expressly or by implication.



Access the power of technology.