

# Flash Update Application for Nuvoton Embedded Controllers User Guide

July 2009  
Revision 1.6



## 1.0 Description

The Flash Update Application (`Flupdate.exe`) runs on either 16-bit DOS or 32-bit/64-bit Windows® XP, Vista® and Windows 7. It programs the Nuvoton WPC876xL, WPCE775x, WPCE78nx or NPCE78mx EC flash and, if the BIOS Model field (see Section 6.0) exists in either the binary image or the EC flash, the Flash Update Application verifies that the host BIOS model is the same as the BIOS model inside the binary image. (A warning is displayed if the host BIOS is newer than the BIOS inside the binary image.) The application programs the contents of the binary image, which resides in the Flash Update executable file, to the flash.

`Flupdate.exe` uses a flash configuration file, provided by the user, to recognize and control the various flash devices.

## 2.0 Features and Functions

- Supports 16-bit DOS and 32-bit/64-bit Windows XP / Vista / Windows 7 (contains automatic system detection)
- Automatically recognizes and supports shared flash systems
- Supports multiple flash types through the configuration file
- Configuration provided through command-line flags
- BIOS model and version recognition (see Section 6.0)

`Flupdate.exe` can be executed in either a DOS or Windows command line. It programs the contents of the binary file to the flash, identifying and programming any of the flashes defined in the `Flupdate.cfg` file (see Section 4).

On successful termination, the firmware jumps to the Boot Block. It is the user's responsibility to reset the EC after flash update completion.

## 3.0 Usage

`Flupdate.exe [options] <binary file>`

where `<binary file>` is the file containing the data to program into the flash, in binary format.

Option	Description
-h	Show program usage (help).
-v	Verbose operation - create a log file and report explicit error messages.
-shf	Assume the system has shared flash; i.e., BIOS and EC firmware (FW) are located on same flash device.
-nshf	Assume the system has no shared flash; i.e., only FW is on the EC flash. By default (i.e., without -shf or -nshf), the system type is recognized automatically.
-c <file name>	Flash Configuration file name. By default, the tool looks for <code>Flupdate.cfg</code> .
-x <exit type>	Select termination option: 0 = Normal Exit. Return to the main code (use only if the EC FW code was not changed). 1 = Reset EC. 2 = Go to EC FW Boot Block (default).
-d <file name>	Dump flash data to file <file name>.
-compare	Compare the given binary file with the flash data. In this mode, the flash is not programmed. If this flag is provided in addition to the -ofs flag (see below), only the bytes starting from <offset> are compared.
-bbl	Ignore protected sectors, as defined in the flash configuration file (see Section 4).
-noverify	Do not verify flash contents after programming.
-nocomp	Program the flash without first comparing its contents to the binary file data. By default, the contents of each section are compared to the source file, and each section is programmed only if different.
-noreboot	Do not prompt the user to restart the computer after the download is finished.
-nopause	Start the download/compare without user approval. In addition, the application terminates automatically after downloading the binary file to the flash (without restarting the computer).
-dislusb	For DOS only: disable the Legacy USB SMI (only in Intel® chipsets) during the flash update process.

Option	Description
-cba <address>	Set the Configuration Base Address of the device (in hex format); otherwise, try to locate the device at the following addresses (in this order): 0x164E, 0x2E, 0x4E.
-novr	Ignore BIOS model and version check.
-vrd	Verify the integrity of the load process using the read command rather than using the default checksum command. This flag is effective only when the host writes to the shared memory via the I/O space.
-rfp	Remove Flash Protection when the protection window registers are accessible (i.e., not locked).
-ofs <offset>	Load the provided binary file to the flash starting at offset <offset> from flash start. <offset> must be aligned with the start address of a flash sector. If this flag is provided in addition to the -compare flag, only the bytes starting from <offset> are compared.
-pfu	Enable EC FW to perform operations before the Flash Update process starts. This flag requires the EC FW to implement the command "Pre-Flash Update" (see the specific EC Software User Guide).
-ocv	Override Calibration Values in the flash with the values provided in the binary file. By default, the Flash Update Application automatically preserves the values when the EC contains an internal clock. Therefore, this flag should be used only when the existing values need to be overridden.
-vm <value>	For DOS only. This flag is used to change the default value of the video mode. It should be used only if there is a known graphic limitation in the system. Supported values are 18 (default), which uses 16 colors, and 17, which uses two colors.
-genbsf <TSF name>	Generates a Binary Script File (BSF) from a Text Script File (TSF). The BSF name is the same as the TSF, but with a different extension (*.bsf instead of *.tsf).
-runbsf <BSF name>	Runs the commands in the Binary Script File (BSF) during the Flash Update process. If the flupdate.bsf file exists in the Flash Update root directory, it is executed automatically (i.e., without the need for the input "-runbsf flupdate.bsf").

## 4.0 Flash Configuration File

The flash configuration file is provided to the Flash Update Application and determines which flash types are supported. A flash configuration file can define up to 100 flash configurations.

The configuration file format is as follows:

```
[START CONFIG]
# NexFlash
# -----
FLASH_ID           = 0x14EF      # Device ID & Manufacturer.
FLASH_SIZE         = 2048        # (2M) Flash size in Kbytes.
SECTOR_DEF         = 512:4K      # 512 sectors of 4K.
                                # The format is: n:sK,n:sK,...,n:sK.
                                # n - number of sectors.
                                # s - size of sectors in Kbytes.
BLOCK_SIZE         = 64          # (64K) block size in Kbytes. If block erase is
                                # not supported, block size should be equal
                                # to FLASH_SIZE.
PAGE_SIZE          = 256         # Page size in bytes.
PROG_SIZE          = 8           # Max number of bytes that can be programmed at
                                # one time. An FF value means PAGE_SIZE is used.
READ_DEV_ID_TYPE   = 0           # Read Device ID type.
                                # There are two ways to read the flash ID
                                # (depending on the flash type).
                                # A. Send the "Read ID" op-code, send 3 dummy
                                #    address bytes and then read the ID bytes.
                                # B. Send the "Read ID" op-code and immediately
                                #    read the ID bytes.
                                # When the value in this field is 0x47, method B
                                # is used. Otherwise (any other value) method A
```

```

# is used.

# Define SPI Flash commands
# -----
CMD_READ_DEV_ID      = 0x90      # Read Device ID command.
CMD_WRITE_STAT_EN    = 0x06      # Enable Write to Status Register.
CMD_WRITE_EN         = 0x06      # Write Enable.
CMD_READ_STAT        = 0x05      # Read Status Register.
CMD_WRITE_STAT       = 0x01      # Write Status Register.
CMD_PROG             = 0x02      # Page Program or Byte Program.
CMD_SECTOR_ERASE     = 0x20      # Sector Erase.
CMD_BLOCK_ERASE      = 0xD8      # Block Erase.
# Define device busy bits in status register
# -----
STATUS_BUSY_MASK     = 0x01      # Location (by mask) of busy bit located in
                                # the status register.
STATUS_REG_VAL        = 0x00      # Value of status register busy bit, when
                                # device is not busy. The device is
                                # considered not busy when:
                                # [status register value] & STATUS_BUSY_MASK = STATUS_REG_VAL.
# Define the protected sectors
# -----
PROTECTED_SECTORS    = 0-4      # Optional - Protected sectors in format sector
                                # number and/or range, separated by commas,
                                # e.g., 1,3,5-7. To avoid defining any protected
                                # sectors, make the line a remark line.

[END CONFIG]
[START CONFIG]
.
.
.
.
.
[END CONFIG]

```

## 5.0 Flupdate.cmd

This is an optional file that provides configuration information. When the Flash Update Application is invoked with no command line parameters, it tries to locate the `Flupdate.cmd` file in the current directory. If the file exists, the Flash Update Application reads its command line parameters from this file.

For example, if the contents of `Flupdate.cmd` content is:

```
-c flash.cfg flash.bin
```

the Flash Update Application runs as:

```
Flupdate.exe -c flash.cfg flash.bin
```

## 6.0 BIOS Model and Version

The BIOS Model and Version fields should be defined in the binary image file, as follows.

- The Model field starts with `$ML` and ends with `;`; for example:  
`$MLABC$` defines the BIOS model as "ABC".
- The Version field starts with `$VR` and ends with `;`; for example:  
`$VR1.23.45$` defines the BIOS version as "1.23.45" (the Version field is a string).

## 6.1 MODEL AND VERSION FIELD LOCATION

It is recommended that both the Model and Version fields be located between offsets 0x50 and 0xA0 of the binary image file. The value of each field has a maximum length of nine characters.

It is possible to change the location of the Model and Version fields using the Offset field, as follows.

- The Offset field starts with \$AD and ends with \$; for example:  
\$AD3040\$ defines the BIOS Model and Version fields as located between offsets 0x3040 and 0x3090 (the Offset field is a string).

## 7.0 Complementary Host Commands (CHC)

The CHC enables the user to execute host commands that write to I/O, memory or PCI before and after the flash update process.

### 7.1 COMMANDS

There are three commands: I/O Write, Memory Write and PCI Register Write.

#### 7.1.1 I/O Write

##### Syntax

IOW : <Width> <Address> <Data> [Mask]

##### Field Descriptions

<Width> - "b" for byte, "w" for word, "d" for double word.

<Address> - The address to be written.

<Data> - The data to be written.

[Mask] - Cleared bits in this parameter preserve the original value of the equivalent bits in the address. For example, if the current value in <Address> is 0xAA, <Data> is 0x55 and <Mask> is 0x0F, then after a successful execution of IOW, the new value is 0xA5. Note: This parameter is optional. If it is not provided, the default is 0xFFFFFFFF (i.e., all the bits of the value in the Address are changed, according to Width).

#### 7.1.2 Memory Write

##### Syntax

MEMW : <Width> <Address> <Data> [Mask]

##### Field Descriptions

Same as the IOW command.

#### 7.1.3 PCI Register Write

##### Syntax

PCIW : <Width> <Bus Number> <Device Number> <Function Number> <Register> <Data> [Mask]

##### Field Descriptions

<Width> - Same as the IOW command.

<Bus Number> - The bus number to be written.

<Device Number> - The device number to be written.

<Function Number> - The function number to be written.

<Register> - The register to be written.

<Data> - Same as the IOW command.

[Mask] - Same as the IOW command.

## 7.2 CHC INPUT FILES

In the CHC input files, define which CHC commands will be run and when they will be run (i.e., at what stage of the flash update process). There are two types of input files: Text Script File (TSF), which is the source file, and Binary Script File (BSF), which is compiled from the TSF.

Commands are first defined in the TSF. To run those commands, you must create a TSF, generate the BSF and then run the BSF.

### Binary Script File (\*.bsf)

This is a binary file that is compiled from a Text Script File (TSF).

The `-runbsf` flag (see below) tells Flash Update to execute the commands defined in this file.

### Text Script File (\*.tsf)

This is the source file used to create a BSF. The TSF syntax is defined below.

To generate a BSF from a TSF, use the `-genbsf` flag (see example below).

#### 7.2.1 TSF Syntax

##### Command Options

- MEMW
- IOW
- PCIW

##### Stages

- STAGE BEFORE FU – commands in this section run before flash update starts.
- STAGE AFTER FU – commands in this section run after the flash update ends.

##### Syntax

- Comment line - a line that starts with “#”
- Tag indicating the start of the STAGE BEFORE FU section: `[START STAGE BEFORE FU]`
- `MEMW / IOW / PCIW` command(s) – Up to 40 commands are supported.
- Tag indicating the end of the STAGE BEFORE FU section: `[END STAGE BEFORE FU]`
- Tag indicating the start of the STAGE AFTER FU section: `[START STAGE AFTER FU]`
- `MEMW / IOW / PCIW` command(s) – Up to 40 commands are supported.
- Tag indicating the end of the STAGE AFTER FU section: `[END STAGE AFTER FU]`

**7.2.2 TSF Sample File**

```

[START STAGE BEFORE FU]
#####
# Assign memory address 0xFF800000 into LPC generic memory range
# (PCH Bus0/Device31/Function0/Register98-9B)
#####
PCIW : d 0 31 0 98 FF800000
#####
# Enable LPC generic memory range
# (PCH Bus0/Device31/Function0/Register98-9B)
#####
PCIW : b 0 31 0 98 01 01 # note: the last byte is a mask parameter
#####
# Assign 0xFF800000 address into EC share windows 2 (index IO: 164e/164f)
#####
# Set SHAW1BA0-3 to 0xFF800000
IOW : b 164E F4
IOW : b 164F 00
IOW : b 164E F5
IOW : b 164F 00
IOW : b 164E F6
IOW : b 164F 80
IOW : b 164E F7
IOW : b 164F FF
#####
# Enable share window LPC memory transaction
#####
# Clear WIN_CFG.SHWINACC
IOW : b 164E F1
IOW : b 164F 00 40
[END STAGE BEFORE FU]
[START STAGE AFTER FU]
# No operation to be done after the flash update process is over
[END STAGE AFTER FU]

```

**7.3 CHC EXAMPLE**

- 1) Write a TSF file according to the syntax described above.
- 2) Generate a BSF file: `flupdate.exe -genbsf sample.tsf`  
This generates the `sample.bsf` file.
- 3) Run the BSF file: `flupdate.exe fw_image.bin -runbsf sample.bsf`

## 8.0 Dependencies

In systems with BIOS support for the Nuvoton EC, the system BIOS should configure the following:

- Shared Memory registers in the SHM logical device (LDN 0Fh):
  - Shared Memory Base Address registers (index F8h-FBh)
  - SHM\_CFG register (index F0h): field SHW\_FWH\_ID
  - WIN\_CFG register (index F1h): bit SHWIN\_ACC
- The chipset should be configured to enable access to:
  - The I/O range of the EC logical device
  - The memory or I/O address range of the shared memory window

For example, in platforms where the SPI flash memory containing the EC firmware is not shared with the system BIOS (this is one possible configuration with Intel's Ilex Peak PCH), the BIOS is required to configure the I/O addresses as specified in the EC Software User Guide, *Shared Access RAM Window* Section (see the sample code, which demonstrates how to map the Window 2 base address to I/O address 400h). Then, the Flash Update Application can be used, as described in Section 3.0 on page 1 (Flupdate.exe [options] <binary file>).

### Important Notice

Nuvoton products are not designed, intended, authorized or warranted for use as components in systems or equipment intended for surgical implantation, atomic energy control instruments, airplane or spaceship instruments, transportation instruments, traffic signal instruments, combustion control instruments, or for other applications intended to support or sustain life. Furthermore, Nuvoton products are not intended for applications wherein failure of Nuvoton products could result or lead to a situation wherein personal injury, death or severe property or environmental damage could occur.

Nuvoton customers using or selling these products for use in such applications do so at their own risk and agree to fully indemnify Nuvoton for any damages resulting from such improper use or sales.

#### Headquarters

No. 4, Creation Rd. 3,  
Science-Based Industrial Park,  
Hsinchu, Taiwan, R.O.C  
TEL: 886-3-5770066  
FAX: 886-3-5665577  
<http://www.nuvoton.com.tw/>

#### Nuvoton Technology Corporation America

2727 North First Street,  
San Jose, CA 95134, U.S.A.  
TEL: 1-408-544-1718  
FAX: 1-408-544-1787

#### Nuvoton Technology (Shanghai) Ltd.

27F, 2299 Yan An W. Rd.  
Shanghai, 200336 China  
TEL: 86-21-62365999  
FAX: 86-21-62365998

#### Taipei Office

9F, No.480, Rueiguang Rd.,  
Neihu District, Taipei, 114,  
Taiwan, R.O.C.  
TEL: 886-2-2658-8066  
FAX: 886-2-8751-3579

#### Winbond Electronics Corporation Japan

NO. 2 Ueno-Bldg., 7-18, 3-chome  
Shinyokohama Kohoku-ku,  
Yokohama, 222-0033  
TEL: 81-45-4781881  
FAX: 81-45-4781800

#### Nuvoton Technology (H.K.) Ltd.

Unit 9-15, 22F, Millennium City 2,  
378 Kwun Tong Rd.,  
Kowloon, Hong Kong  
TEL: 852-27513100  
FAX: 852-27552064

For Advanced PC Product Line information contact: [APC.Support@nuvoton.com](mailto:APC.Support@nuvoton.com)

Please note that all data and specifications are subject to change without notice.  
All trademarks of products and companies mentioned in this document belong to their respective owners.