Flash Magic

Product Manuals

Manual Revision 1.00

About This Manual

This manual follows some set conventions with the aim of making it easier to read. The following conventions are used:

0x	Hexadecimal (base 16) values are prefixed with “0x”.
italictext	Replace the text with the item it represents
[ ]	Items inside [and] are optional
a | b	a OR b may be used
…	One or more items may go here.

Contents

1. Introduction

NXP Semiconductors produce a range of Microcontrollers that feature both on-chip Flash memory and the ability to be reprogrammed using In-System Programming technology.
Flash Magic is Windows software from EmSA that allows easy access to all the ISP features provided by the devices. These features include:

- Erasing the Flash memory (individual blocks or the whole device)
- Programming the Flash memory
- Reading Flash memory
- Performing a blank check on a section of Flash memory
- Reading the signature
- Programming external flash
- Verification
- Firmware patching
- Production tasks
- Sending commands to place device in Bootloader mode
- Programming over serial port, SWD, CAN and Ethernet

Flash Magic provides a clear and simple user interface to these features and more as described in the following sections.

Under Windows, only one application may have access the COM Port at any one time, preventing other applications from using the COM Port. Flash Magic only obtains access to the selected COM Port when ISP operations are being performed. This means that other applications that need to use the COM Port, such as debugging tools, may be used while Flash Magic is loaded.

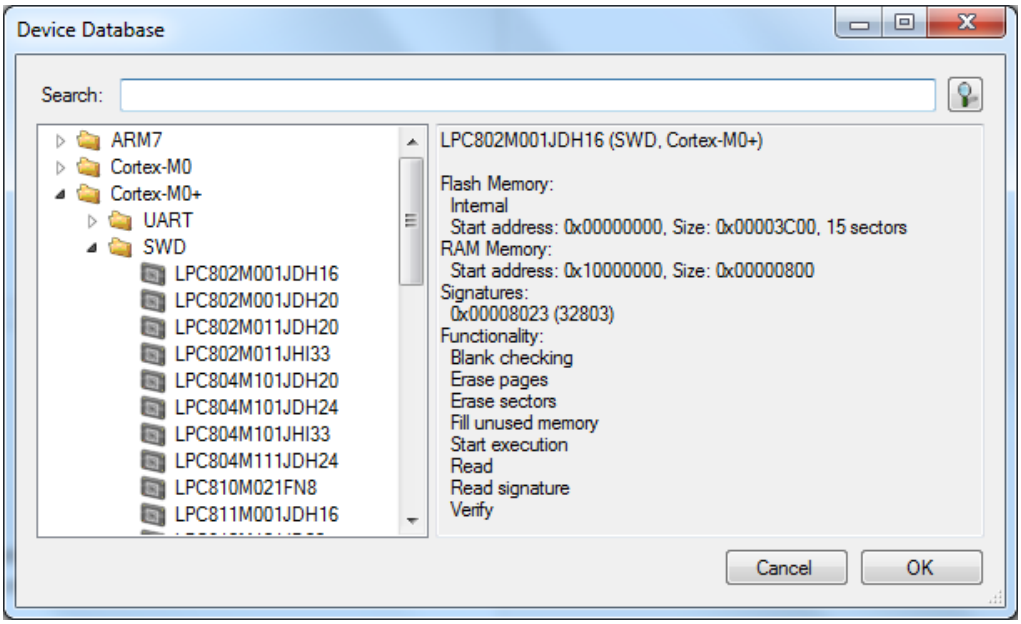
Note that in this manual third party Compilers are listed alphabetically. No preferences are indicated or implied.

2. Basic Usage

The main window is divided up into five sections from top left to bottom right. These sections contain the most common options and settings for erasing and programming a microcontroller.

2.1 Device

Choose the microcontroller you wish to use. Clicking on the Change… button opens the Device Database window. The Device Database shows all supported microcontrollers along with information about each one.



To look for a specific device enter the name (or part of a name) into the search box and click on the search button.

Microcontrollers are grouped according to the connection type (UART, SWD, CAN, Ethernet). Once selected the Device section of the main window will show configuration settings for the chosen connection type.

If a microcontroller supports more than one connection type then it will appear multiple times in the device database, once for each connection type.

If SWD connection type is chosen then Flash Magic will automatically detect any attached debug adapter or development board. See the SWD chapter for more details.

2.2 Erase

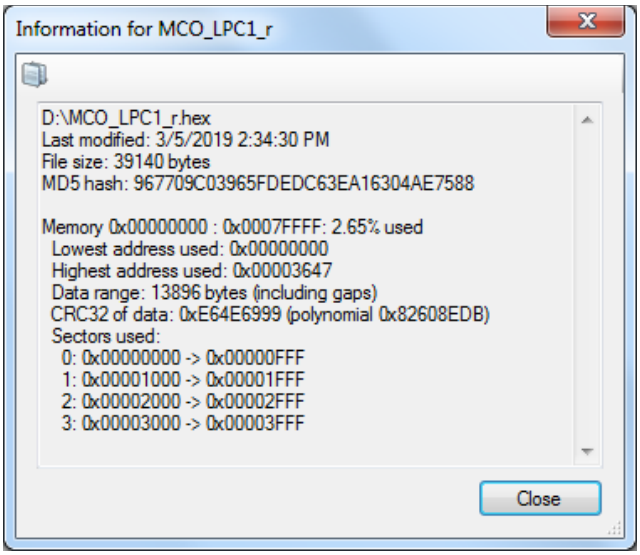
Choose the erase option to use and if desired the specific sectors to erase. Erase will happen before programming.

2.3 Firmware

Choose an Intel hex file or a binary file to program into the device. Binary files will be located at the start of the device memory space.

Below the input box the last modified date and time of the file is shown. This will update when the firmware is rebuilt by a compiler. It is not necessary to reselect the firmware file each time after rebuilding. Flash Magic does not keep the contents in memory and instead reloads the contents just before programming.

Also below the input box is a link “View File Information”. Clicking on this opens a dialog window that shows detailed information regarding the firmware file.



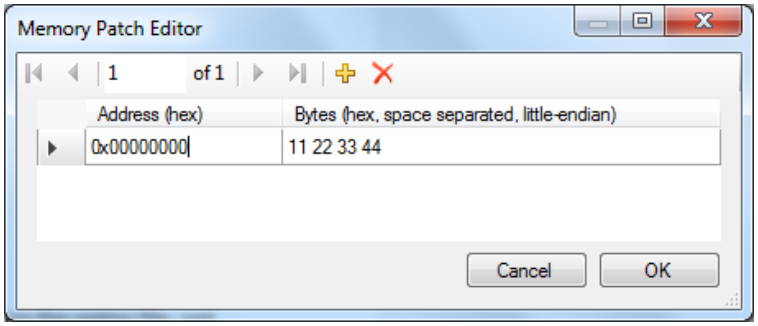
The MD5 hash can be recalculated using commonly available utilities and is based on the entire file, not just the firmware data in it.

Also displayed is a CRC32 of the firmware binary data, allowing regeneration by the firmware as a form of self-integrity checking.

2.4 Options

The options section provides configuration of what should happen before and after programming. For example verification. The available options depends on the functionality of the chosen microcontroller as well as the version of Flash Magic.

Located here is the patch editor. This allows changes to the firmware immediately before programming. To enable check the box and click on Settings… to open the editor.



To add a patch click on the yellow + button. Enter an address in hexadecimal and then enter hex data separated by spaces.

To remove a patch click on the red x button.

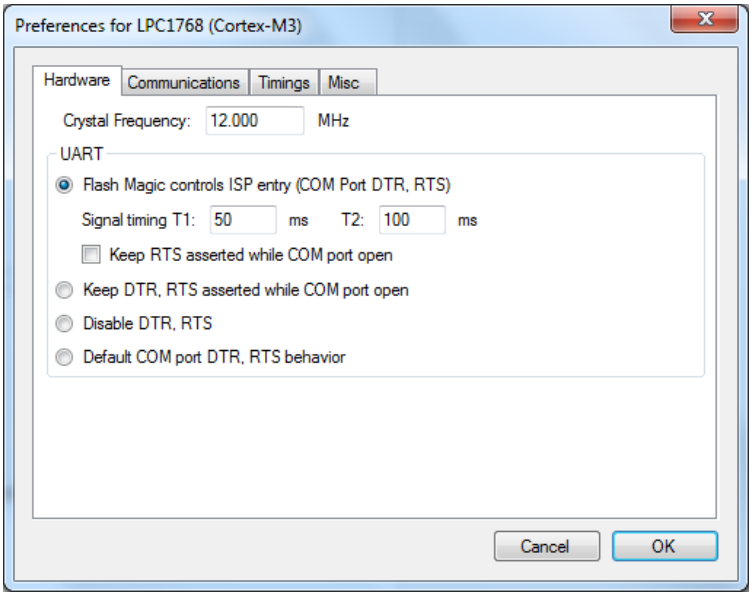
2.5 Start

Click on the button to start the erase-program sequence. This will perform all the chosen items in the main window in one go.

During operation the start button changes to a cancel button.

3. Advanced Settings

The advanced settings are available from the Preferences window. Open by clicking on the toolbar button or using the View menu.



The settings are divided into sections such as Hardware, Communications and Timings and each section may be subdivided into connection type, such as UART and CAN.

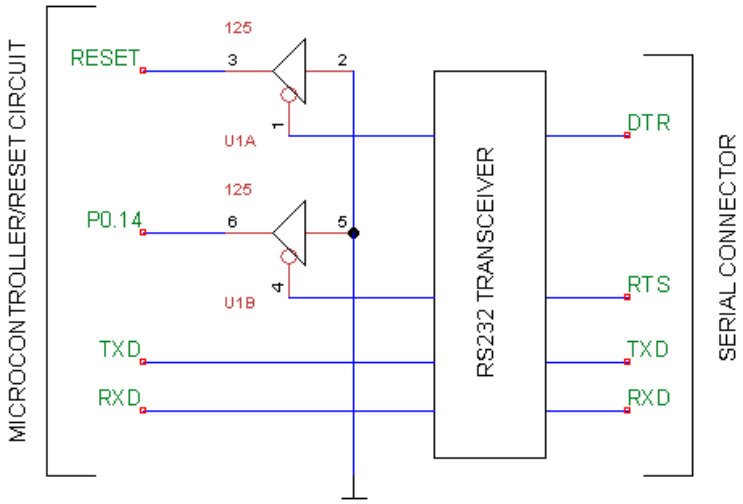
3.1 Hardware

The crystal frequency is required to allow microcontrollers to self-time, i.e. work out how long an erase operation will take. It is important to enter an accurate value in MHz.

For UART connections Flash Magic can optionally place the microcontroller into ISP mode and execute the new firmware afterwards. It does this by controlling the DTR and RTS pins on the PC serial port connector. To enable choose the option “Flash Magic control ISP entry”.

The relative timing of the signals can be adjusted to allow for varying capacitances. It is also possible to keep RTS asserted for the entire time the COM port is open, which is useful if used as a signal to other hardware that programming is taking place.

Support in hardware is required to allow Reset and ISP entry pins to be controlled by Flash Magic. Here is an example circuit for ARM7 parts.

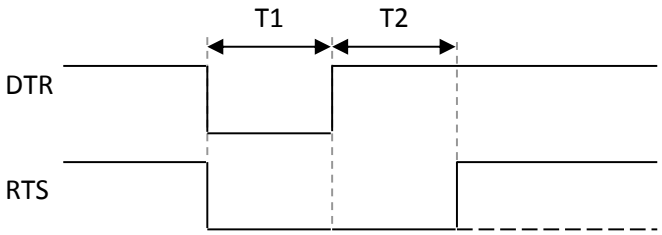


DTR controls the Reset signal and RTS controls the ISP entry signal. The idea is that when DTR is asserted the microcontroller is held in reset. When RTS is asserted the ISP entry pin is placed into the state that starts the on-chip bootloader when exiting reset.

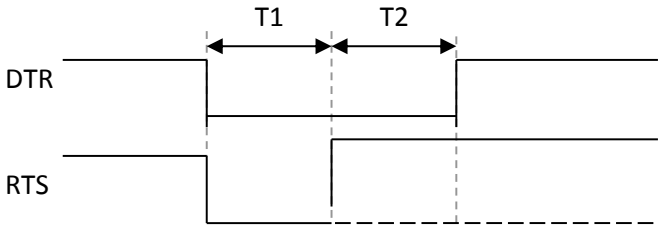
When the COM Port is not in use or the serial cable is not connected, the RS232 signals are pulled low by the transceiver. This results in the TTL signals being high. Therefore when the TTL DTR and RTS signals are high, P0.14 and RST must be in a state that allows the device to reset normally and execute code.

The waveforms generated are as follows. Note that the signal levels are TTL (it is assumed DTR and RTS have already been passed through an RS232 transceiver).

Start of ISP operation (starting the Bootloader):



End of ISP operation (executing firmware):



Note that when a PC COM port is closed both DTR and RTS are de-asserted by the operating system.

Other options for use of the COM port signals are available. Keeping DTR and RTS asserted during ISP operations allows hardware to detect ISP operations.

Note that when selecting to keep RTS asserted while the COM Port is open, or selecting to assert DTR and RTS while the COM Port is open, and high speed communications is selected, there will be pulses on the RTS (and DTR) signals just before the ISP operation. This is because the high speed communications feature needs to reconfigure the COM port several times before the ISP operation is performed. For each reconfigure Windows de-asserts the DTR and RTS signals. Flash Magic then immediately reasserts the DTR and RTS signals resulting in a pulse.

If the default COM port behavior is selected then the DTR/RTS signals will behave normally for a COM port. For example they will be asserted when the COM port is opened.

An option is provided to completely disable all DTR/RTS use. This is required for any hardware or specialty USB to RS232 bridges that cannot support or function correctly when either DTR or RTS are asserted.

3.2 Communications

Options in this group affect how Flash Magic communicates.

A delay can be introduced after opening the communications channel to the target hardware.

If communications with a part fails to work it is recommended to try adjusting the line feed settings. Flash Magic automatically attempts to detect the line feed settings required by the part but sometimes this is not successful and options are available to explicitly set how line feeds are used.

Enabling the high speed communications option will cause Flash Magic to attempt to negotiate a higher baud rate after an initial connection has been established.

When communicating with the device Flash Magic can send and receive data at the same time to achieve the fastest data rate. This type of transmission is called full-duplex. Turning the half-duplex option on will cause Flash Magic to only transmit one byte at a time, waiting for the byte to be echoed from the Bootloader before transmitting the next byte. While this will slow the data rate down it allows ISP to be performed via half-duplex serial buses, such as RS-485 and RS-485 derivatives such as J1708.

Note however that you must design your hardware such that the PC and the Bootloader do not receive the bytes they transmit otherwise each will be confused.

3.3 Timings

In this section the timeouts Flash Magic should use when performing ISP operations are specified. Normally default settings are used, however if you wish to change the timeouts then check the option to use my timeouts and fill in the values in the boxes.

Flash Magic uses two timeouts, regular and long. Each timeout is specified in seconds. The regular timeout is used for most ISP operations. The long timeout is used for erasing and performing blank checks.

The default settings are four seconds for the regular timeout and 60 seconds for the long timeout. It is recommended to use the default settings.

If you are using a USB to COM port converter then you may find that increasing the timeouts will resolve communication problems that are sometimes present with those converters.

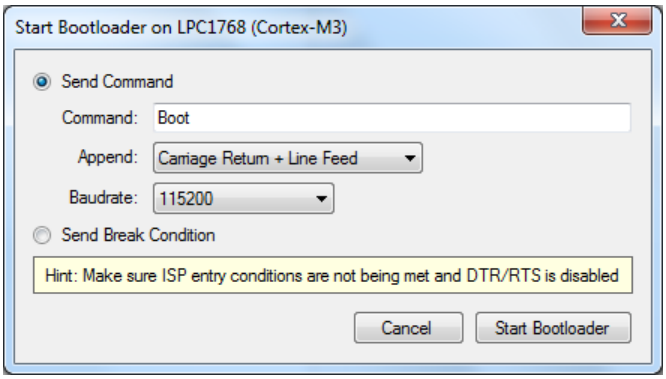
3.4 Misc

If a device is encountered with an incorrect signature, it is possible to turn of the signature checking by checking the Disable signature checking option. Please report any incorrect signatures to support@esacademy.com. Please include the signature read from the device along with all markings on the top of the device itself.

4. Starting Bootloaders

If a custom bootloader has been developed then it may be necessary to start it by sending a command, e.g. if a command line interface has been implemented. Flash Magic includes the option to send a command before starting ISP operations.

Access the Start Bootloader dialog by going to the ISP menu and choosing Start Bootloader…



There are two options available: an ASCII command or a break condition. For the ASCII command enter it and choose the baud rate and any control characters to be appended.

Special characters may be inserted into the command to provide additional functionality. They consist of a special character followed by two hexadecimal characters and have the following meanings:

%HH - transmit the character HH, where HH is the hexadecimal value of the character
$HH - delay for HH x 100ms, where HH is a hexadecimal value
&AA - command. The function of the command depends upon the hexadecimal value AA and currently can be one of the following:

0 - flush RX buffer
1 - echo on
2 - echo off
3 - send break condition
4 - don't wait for a response

By default at the start of a command the echo is on.
Regardless of whether the echo is on or off, when the end of a command is reached the device must return a '.' to indicate the command was successfully received. If &04 is included anywhere in the command then the device does not need to return a response.

Examples:

%0D%02	Transmits a carriage return followed by an STX
%80	Transmits 80H
$14	Delays for 20 x 100ms = 2 seconds
$0A	Delays for 10 x 100ms = 1 second

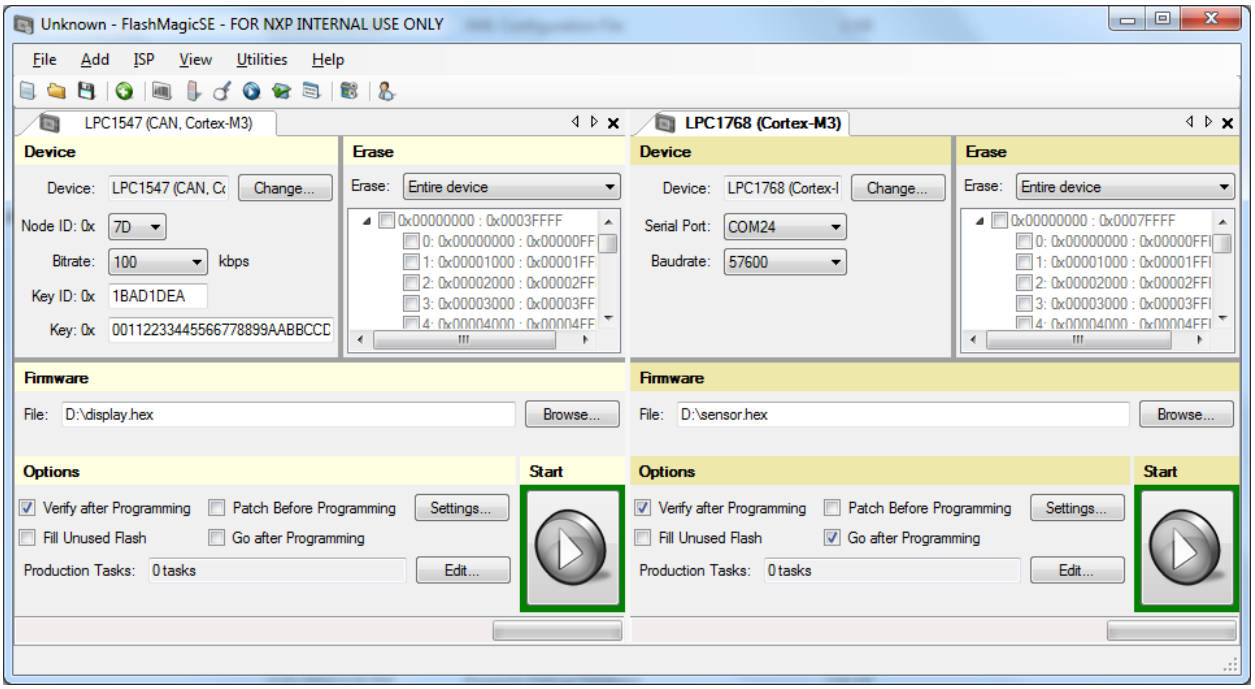
&00	Flushes the RX buffer
A&02B&01C	Transmits ‘A’, waits for ‘A’ to be echoed, transmits ‘B’ then transmits ‘C’ and waits for ‘C’ to be echoed.
%0D$14&00A	Transmits a carriage return, waits for 2 seconds, flushes the RX buffer and transmits ‘A’.

5. Multiple Programmers (Production System Only)

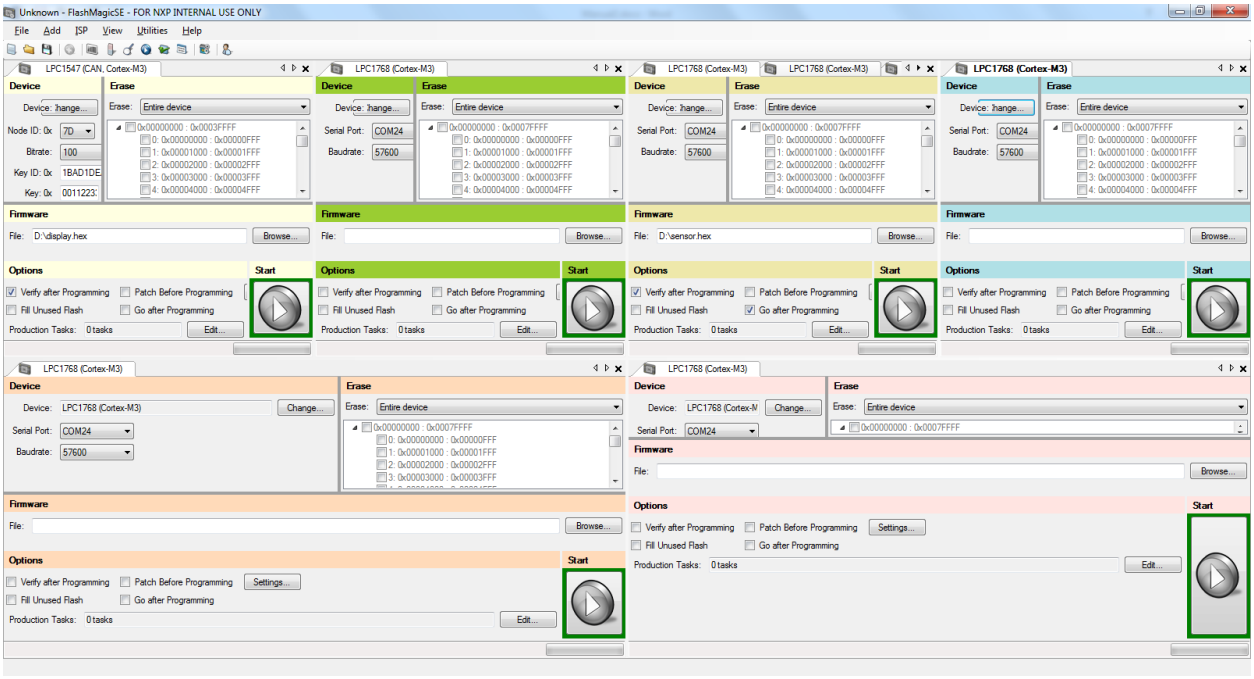
In the Production System version Flash Magic supports multiple “programmers” that can be used in parallel for gang programming.

A “programmer” is the user interface comprising the device, erase, firmware, options and start sections.

To add a new programmer choose Programmer… from the Add menu. The new programmer will open under a new tab. The tab can be dragged and docked to various parts of the user interface.



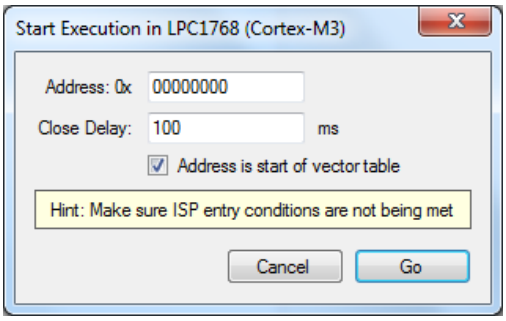
Up to 10 programmers can be shown at once.



This is useful not only for gang programming but for projects where there are multiple microcontrollers on a single circuit board. One programmer can be added for each microcontroller, programming different firmware.

6. Firmware Execution

To execute firmware on the target microcontroller choose Go… from the ISP menu.



Enter the address to start execution from. Click on Go to start execution.

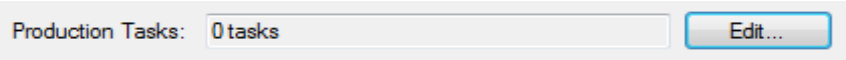
Optionally enter a delay in milliseconds to cause Flash Magic to pause between sending the Go command and closing the communications channel. This is useful when executing from RAM with an LPC2xxx device and closing the COM port may reset the device, due to DTR and RTS being used.

If the code to be executed starts with an interrupt vector table then check the option. Flash Magic will extract the stack pointer and execution address and use them, if supported by the device.

7. Production Tasks (Production System Only)

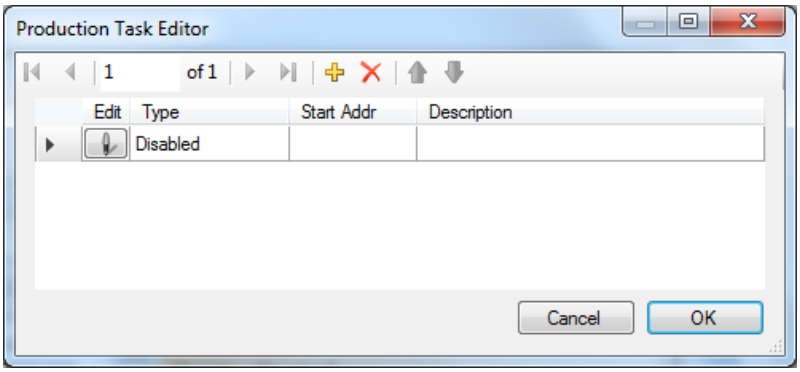
Production tasks are a set of tasks that are performed just before programming takes place. They can be used to add additional data to be programmed, either from binary files, hex files or from arbitrary executables.

The user interface for production tasks can be found in the options section of the main window.

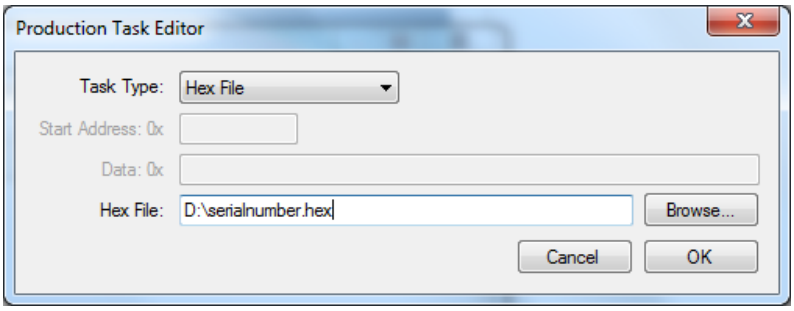


To add, modify or delete tasks click on the Edit… button.

Click on the yellow + button to add a new task.



Click on the edit button to modify a task.



Choose a task type, enter the settings and click on OK.

To delete a task select it in the list and click on the red x button.

The order that tasks are executed can be modified. Select a task and click on the up or down arrow buttons to move the task up or down in the list. The task at the top of the list will be executed first.

The following sections describe the types of tasks that are supported.

7.1 Disabled

This task does nothing.

7.2 Hex File

The hex file is loaded in and merged with the firmware file chosen in the main window. Any duplicate memory locations are overwritten with the contents of the task hex file.

7.3 Patch

A piece of hard-coded binary data is merged with the firmware file chosen in the main window, overwriting any values.

Enter the address for the start of the data and the data to use. Data is entered as a hexadecimal value and is stored in little-endian format.

7.4 Binary File

The contents of the binary file are loaded in and merged with the firmware file chosen in the main window, starting at the specified memory address. Any duplicate memory locations are overwritten with the contents of the binary file.

7.5 Hex Generator

Choose the location of an executable (EXE file) that outputs to standard output Intel hex records. This allows pieces of data such as serial numbers to be generated on the fly or fetched from a database.

8. Command Line Interface

Flash Magic includes a command line version. The file name depends on the version of Flash Magic that you have. The free version is called fm.exe for example.

The command line interface to Flash Magic allows the features of Flash Magic to be accessed from the command line/DOS. This provides a powerful and flexible way of integrating Flash Magic into:

- your project's build process
- the Integrated Development Environment you use for developing applications.

If you use a Batch file to build your project then a call to Flash Magic can be added onto the end of the batch file allowing one-step build and program of your application.

Most modern IDEs allow users to add custom menu entries that run any command you desire. Therefore it is simple to add menu entries to run Flash Magic and program the latest Hex file.

It is possible to configure Flash Magic to output the result of all operations to an ASCII text file. The output is described in this manual enabling programs to be written to parse the text file and provide automated testing of devices or gang-programming of devices.

8.1 Overview

The command line accepts a sequence of directives telling it what to do. There are two types of directive:

- Configuration
- Task

When called the command line interface will look at all the configuration directives used and apply them. Then it will execute the tasks in the order they are specified on the command line. This allows complex sequences to be constructed performing multiple steps in one go.

When the command line is used it must have at least one task specified otherwise it won't do anything.

A directive has the following format:

-<single character name>
--<long name>

i.e. a single hyphen specifies a single character name and a double hyphen specifies a long name. Many directives have both a single character name and a long name and you can choose which to use.

Values can be passed to a directive as follows:

```
--device LPC54608
--device=”LPC54608”
```

Wrap the values in double quotes if the value contains spaces.

Some directives accept more than one value. In that case separate the values with commas. Don’t add spaces. For example:

```
--timeouts 60,120
--timeouts=”60,120”
```

The special directive -h or --help shows a list of all supported directives. For example:

```
fm.exe -h
```

8.2 Examples

Here are some example command lines. For full details of what the directives do, what directives are available and how they can be used use the -h directive.

Read the signature using the serial port (UART):

```
fm.exe --device LPC54608 --serialport COM5 --baudrate 57600 --readsignature
```

Erase sectors three and four using SWD:

```
fm.exe --device LPC54608 --daplink --erasesectors 3,4
```

Erase sector six **then** erase page 12 **then** read the unique serial number using SWD:

```
fm.exe --device LPC54608 --daplink --erasesectors 6 --erasepages 12
      --readserialnumber
```

Using the serial port erase the entire device **then** program **then** verify, using custom timeout settings:

```
fm.exe --device LPC1768 --serialport COM6 --baudrate 38400 –erasedevice
      --program="D:\firmware.hex" --verify="D:\firmware.hex" --timeouts 4000,60000
```

Erase the device **then** program a hex file **then** program another hex file, using SWD:

```
fm.exe --device LPC54608 --daplink --erasedevice --program=”D:\firmware.hex”
      --program=”D:\serialnumber.hex”
```

9. Ethernet Bootloader

This chapter provides an overview of the Ethernet bootloader for LPC17xx/LPC2xxx devices, including a quick start guide.

The Ethernet bootloader has been developed by NXP and is bundled with Flash Magic. To access the files go to Start Menu -> All Programs -> Flash Magic -> Ethernet Bootloader. The bootloader will run on any LPC17xx/LPC2xxx device with an on-chip Ethernet controller.

The bootloader is provided in the form of a pre-built hex file and Keil Realview project. Also included is a sample blinky application and an application note from NXP. For full details of the bootloader please refer to the application note.

Note that Vista/7 users will need to run Flash Magic as an administrator in order to use the Ethernet bootloader. This is achieved by right-clicking on the application and choosing "Run As Administrator" from the menu. Administrator rights are required for the IP address assignment.

9.1 Quick Start For Keil MCB1700/MCB2300

The following are steps to quickly get started using the bootloader with the Keil MCB1700 and MCB2300 boards. For other boards please see the later sections of this chapter on changes that might be needed.

Initially the MCB1700/MCB2300 board should be set up so that the device can be programmed using the UART (COM0 connector). This is required in order to program the Ethernet bootloader into the device.

Select to erase all flash and program the pre-built ethernet_bootloader.hex. This file can be found in the Bootloader/Obj folder inside the Ethernet Bootloader folder, accessible from the Start Menu. The bootloader occupies the first two flash blocks of the device.

Once programmed disconnect the serial cable and connect an Ethernet cable.

If you are using a LPC175x then set jumper E/U to connect positions 2 and 3 and set jumper E/C to connect positions 2 and 3. This connects pin P2.8 to MDC and pin P2.9 to MDIO for the software-based MIIM layer. If you wish to use a different evaluation board with different GPIO pins for MDC and MDIO then you will need to edit the init_emac function and mdio.c. Note that this is not required for the LPC176x devices as they use a hardware-based MIIM layer.

Reset the board to execute the bootloader. LED P2.6 (MCB1700) or P2.7 (MCB2300) should start flashing to indicate the bootloader is executing.

In Flash Magic select the same device but with the "Ethernet" suffix, for example "LPC2388 Ethernet". This will display the IP and MAC address options in Step 1 of the main window.

The MAC address must match the value hard-coded into the bootloader. Set the MAC address to the default value of "0c-1d-12-e0-1f-10". Note that MAC addresses are a set of six two-digit hexadecimal values separated by hyphens.

Initially the bootloader does not have an IP address, so it must be assigned one. This is achieved by entering the desired IP address into Flash Magic. IP addresses are four decimal values separated by periods. You must pick an address that is on the same subnet as your PC and is not currently used by any other device on the network. For example if your PC has the IP address "192.168.0.10" then you might enter "192.168.0.11". Note that only the last value in the address is changed. This indicates the address is on the same subnet.

Select "Erase Blocks Used By Hex File", and select the Blinky.hex example application hex file. The hex file can be found in the User Code Sample Blinky\Flash folder inside the Ethernet Bootloader folder, accessible from the Start Menu. Program the hex file into the device.

Reset the device to execute the Blinky example application.

To get back to the bootloader use the UART bootloader to erase block 2 then reset the device. This method is used only for illustrative purposes for this quick start guide. See the NXP application note for details of other options, such as using CRP and a dedicated bootloader entry pin.

9.2 IP Address

The IP address is only assigned to the bootloader while Flash Magic is performing programming operations. Once programming operations have finished the IP address is no longer assigned and a different one may be chosen.

To find the current IP address of your PC go to Start Menu -> Run and enter "cmd" into the box. Click on "OK". At the command prompt in the window that opens enter "ipconfig" and press Enter. The IP address is shown on the line starting with "IP Address".

9.3 Bootloader Configuration

The default settings for the bootloader will likely need to be changed. The settings are contained in sbl_config.h, and Keil's uVision provides an easy to use wizard for changing the settings. Open the project in uVision, open the sbl_config.h file and click on the "Configuration Wizard" tab at the bottom of the editor window.

The method of activating the Ethernet bootloader can be controlled using the Code Read Protection setting and/or the Update Entry Pin. The Update Entry Pin must be different to the one used for the UART bootloader. Additionally the Ethernet bootloader always executes if no user code is found in block 2 of the device.

See the NXP application note for full details of the bootloader configuration.

9.4 Protocol Timeout

Embedded Systems Academy has added an additional configuration option that is not described in the application note. The option is found in sbl_config.h and it is called "Timeout", under the "Protocol Options" heading.

When the bootloader first executes it waits for handshaking commands. Once the handshaking has completed the bootloader will only respond to packets from the same PC that performed the handshaking.

If no packets are received for the length of the timeout period then the bootloader will go back to waiting for handshaking commands. This ensures that if communications are disrupted or become confused for some reason the bootloader will automatically return to a known state.

The value is specified in milliseconds and has a maximum value of 1000000ms (16.67 minutes).

9.5 UDP Tx Delay

It may be the case that for lower CPU frequencies the bootloader is not able to process the UDP packets containing data quickly enough. Flash Magic by default transmits the packets as quickly as possible.

If there are communication problems try increasing the UDP Tx Delay in the main window. This adds a post-transmit delay to every UDP packet.

10. External Flash

Some NXP microcontrollers support external memory devices, such as the LPC1800, LPC4070/80 and LPC4300 families. Flash Magic supports programming the external memory devices using customizable bootloaders.

The bootloaders are in the form of hex files stored in the “Bootloaders” subfolder in the Flash Magic installation. Copying a hex file to this folder will cause the file to be scanned when Flash Magic is started to determine if it is a valid bootloader. If it is then it appears in the device database in the External Flash group and can be selected for use.

There are two primary ways to use these bootloaders:

1. The microcontroller must be configured to boot from a UART and then reset. It will wait for a header and image to be provided. When the Start button is clicked in Flash Magic it will automatically download the bootloader and then use it to perform the external flash operations that have been chosen, in a single step.

Note: configuring the microcontroller to boot from a UART is not the same as invoking an internal NXP bootloader. See the user manual from NXP for details of how to configure the microcontroller to boot from a UART.

This method is called “BootROM” and is used by the LPC1800 and LPC4300 families.

2. The internal UART-based bootloader is started. When the Start button is clicked in Flash Magic the external flash bootloader is downloaded and executed in a single step. It is then used to perform the external flash operations that have been choosen.

This method is used by the LPC4070/80 family.

Flash Magic is supplied with some bootloader hex files and example source code that can be used as a basis for developing your own bootloaders.

10.1 Bootloader Protocol

This section describes the protocol used for communication between Flash Magic and a custom bootloader.

Communications Method

The bootloader communicates with Flash Magic using a UART configured to a fixed baud rate. It must be the same UART that the microcontroller was configured to boot from.

Initialization

When the bootloader starts executing it should transmit the string “ESAFLASHOS” followed by a carriage return and line feed (CR-LF). This should only be sent after initialization has been completed, for example configuring I/O and reading descriptions of the attached external memory.

If the initialization should fail then the string “INITERR” followed by CR-LF should be transmitted.

Protocol Overview

The bootloader waits for 16-byte command packets. If no data is received for one second then it should go back to waiting for a new 16-byte command packet.

The contents of the command packet determine what is transmitted after the packet, if anything.

All multi-byte variables are in little-endian format.

If the bootloader receives a single '?' instead of a command packet then it should respond with the string “ESAFLASHOS” followed by CR-LF and go back to waiting for a command packet.

Erase All

Command packet:

Byte	Description
0	0x01
1 – 15	Reserved. Set to zero.

Responses:

- “OK” + CR-LF
- “ERROR” + CR-LF

Erase Block

Command packet:

Byte	Description
0	0x02
1 – 4	32-bit address of block
5 – 15	Reserved. Set to zero.

Responses:

- “OK” + CR-LF
- “ERROR” + CR-LF

Program Data

Command packet:

Byte	Description
0	0x00
1 – 2	16-bit value containing number of data blocks to be programmed
3 – 6	32-bit value containing checksum of data
7 – 10	32-bit value containing start address of data to be programmed
11 – 15	Reserved. Set to zero.

After the command packet has been sent the binary data must be sent in blocks. After each block has been sent the bootloader returns a response:

- "PROGOK" + CR-LF
- "PROGERR" + CR-LF

When the last block is transmitted the bootloader will respond with PROGOK or PROGERR and then verify the download and programming was successful for the entire data. It will respond with:

- "CHECKSUM" + CR-LF = checksum mismatch
- "VERIFY" + CR-LF = data was not correctly programmed
- "OK" + CR-LF

10.2 Source Code

Example source code for a bootloader can be found in the "Bootloaders\External Memory" subfolder in the Flash Magic installation.

The folder structure is as follows:

- Core
  - Contains the bootloader protocol implementation and descriptor declarations
- <memory device name>
  - <microcontroller>-<hardware>
    - Contains microcontroller-specific source files and the Keil uVision project file

Descriptor

A valid bootloader contains a block of data that describes the bootloader. Flash Magic reads this block to determine how the bootloader can be used. The descriptor block is defined in descriptor.h and must be declared somewhere in the bootloader source code. The descriptor can be located anywhere in the hex file. Multi-byte values must be in little-endian format.

The descriptor contains the start and end RAM addresses that correspond to the memory where the bootloader should be loaded to. This must match the boot method of the microcontroller and for the LPC1800 and LPC4300 it is 0x1000000 upwards. The linker must place the bootloader at this RAM address. For the LPC4070/80 it is 0x10000300 upwards. The first 0x300 bytes are used by the on-chip bootloader.

The descriptor also contains the range of memory that can be programmed and the programming block size, which must match the programming size of the external memory.

Other fields in the descriptor define the baud rate to use along with a name and description of the bootloader, the sector sizes and the value of an erased byte.

Timing

The bootloader module exposes a global variable called gBootloaderTick. This must be incremented every millisecond. The example bootloader does this using the SystemTick interrupt handler.

UART API

The bootloader expects an API to be available for access to the UART for communications. It must contain functions for initialization, sending and receiving. The receive function must be non-blocking, i.e. if no data is waiting to be read then return immediately.

Memory API

The bootloader expects an API to be available for access to the memory being programmed. This API is very similar to the one used by Keil Flash Modules (FLM files) and conversion from an FLM should be straightforward. The API contains functions for initialization, erasing, programming and reading.

Skeleton Main

The following code shows a skeleton main function for using the bootloader module.

```
int main(void) {
  if (!Bootloader_Init(12)) while(1);

  while(1) {
    Bootloader_Process();
  }
}
```

An initialization function is called passing the CPU frequency and then a process function is called in the background loop.

11. Single Wire Debug (SWD)

Flash Magic supports programming microcontrollers over SWD. There are two options:

- NXP DAPLINK/CMSIS-DAP Debug Interface
- EmSA SWD Bridge Interface

Boards from NXP are already programmed with the DAPLINK/CMSIS-DAP debug interface. Simply connect to your PC and Flash Magic will recognize the board automatically and show it in the interface drop-down list along with a unique serial number allowing multiple boards to be distinguished.

The EmSA SWD Bridges are firmware files included with Flash Magic that can be programmed onto Link2 (using DFU mode) and LPC11U35 compatible debug interfaces.

11.1 EmSA SWD over Link2 Bridge

This bridge uses the USB interface on a LPC1800 or LPC4300 to communicate with Flash Magic and SWD to communicate with the target device (LPCxpresso Link2 circuit design). A standalone LPC-Link2 board can also be used.

The LPC1800/LPC4300 must be configured to boot into USB DFU mode. The NXP LPC USB driver must be installed. The option to install this is part of the Flash Magic installer, and is optional. If you are unsure if it is installed re-run the Flash Magic installer and select the option to install it.

Flash Magic will automatically detect the hardware, program in the bridge code and start using it. Only one bridge can be connected to a PC at once. The bridge appears as a USB HID device.

To use this bridge choose it from the interface drop-down list in section 1 of the main window.

The LPC1800/LPC4300 must use a 12MHz crystal. If an LED is connected to pin P1[1] then it will flash at 500Hz when the bridge code is executing.

The following SWD connections are used:

LPC1800/LPC4300 Pin	Function	Notes
P1[5]	SWDIO Transmit Enable	When high SWDIO is driven by LPC1800/LPC4300. When low SWDIO is driven by target device.
P1[6]	SWDIO	SWD Signal
P1[17]	SWCLK	SWD Signal
P2[5]	/RESET	Connects to reset of target device
P2[6]	/RESET Enable	When high LPC1800/LPC4300 drives reset of target device using /RESET. When low target device reset is pulled high or under control of a reset button.

11.2 EmSA SWD over LPC11U35 Bridge

This bridge uses the USB interface on a LPC11U30 to communicate with Flash Magic and SWD to communicate with the target device.

Before using this interface the LPC11U30 must be programmed with the bridge code. The easiest way to do that is using the mass-storage USB mode. The binary file can be found in the Interfaces subfolder in the Flash Magic installation.

Once programmed and running Flash Magic will automatically detect the hardware and start using it. Only one bridge can be connected to a PC at once. The bridge appears as a USB HID device.

To use this bridge choose it from the interface drop-down list in section 1 of the main window.

The LPC11U20/30 must use a 12MHz crystal. If an LED is connected to pin P0_20 then it will flash at 500Hz when the bridge code is executing.

The following SWD connections are used:

LPC11U20/30 Pin	Function	Notes
P0_8	SWDIO	SWD Signal
P0_7	SWCLK	SWD Signal
P0_2	/RESET	Connects to reset of target device