



Application Note: JN-AN-1003

JN51xx Boot Loader Operation

This Application Note describes the functionality of the boot loaders for the NXP JN517x, JN516x, JN514x and JN5139 wireless microcontrollers, covering the following topics:

- Operation of the boot loader after a cold start
 - Operation of the boot loader after a warm start
 - The Flash image application header
 - The serial interface used to communicate with the boot loader
-

Introduction

The JN514x and JN5139 devices have a ROM-based boot loader, while the JN517x and JN516x boot loader is stored in internal Flash memory. The boot loader is executed by the CPU following a reset or on waking from sleep or deep sleep mode, when power is applied.

- The JN514x/JN5139 boot loader is designed to operate with external serial memory (normally Flash memory) connected to the SPI interface - it loads the user application from this memory into RAM and starts program execution
- The JN517x/JN516x boot loader executes user applications stored in internal Flash memory and can also load applications from external serial (SPI) memory

JN514x/JN5139 Boot Loader Operation

When started, the JN514x/JN5139 boot loader performs one of the following operations:

- Starts execution of the application, already present in RAM
- Copies an application from the external Flash memory device (starting at Flash memory location 0) to RAM and starts application execution
- Enters programming mode

The application must contain two entry points from which execution may begin - a wake-up point and a reset point. The locations of these points are stored in the header information contained within Flash memory.

If the application preserves the contents of the internal RAM when the device goes to sleep via the **vAHI_Sleep()** function, passing either `E_AHI_SLEEP_OSCON_RAMON` or `E_AHI_SLEEP_OSCOFF_RAMON`, then the boot loader will restart the application from the wake-up entry point rather than the reset entry point.

If the application is not present in RAM, the boot loader will attempt to load the application from the external Flash memory. If the boot loader finds a valid application at address 0 in Flash memory, this application will be copied into RAM.

Alternatively, the boot loader may enter programming mode if the SPI MISO line is held low when the device is reset, or if no valid application image is found in the external Flash memory.

The boot loader operations for the JN5148, JN5142 and JN5139 devices are illustrated below. **The JN5148-001, JN5148-J01 and JN5148-Z01 variants have different boot loader operations** - the JN5148-J01 and JN5148-Z01 variants have the same boot loader operation as the JN5142 device. JN5142 variants (JN5142-001 and JN5142-J01) have the same boot loader operation and are referred to as JN5142 below.

JN5148-001 Operation

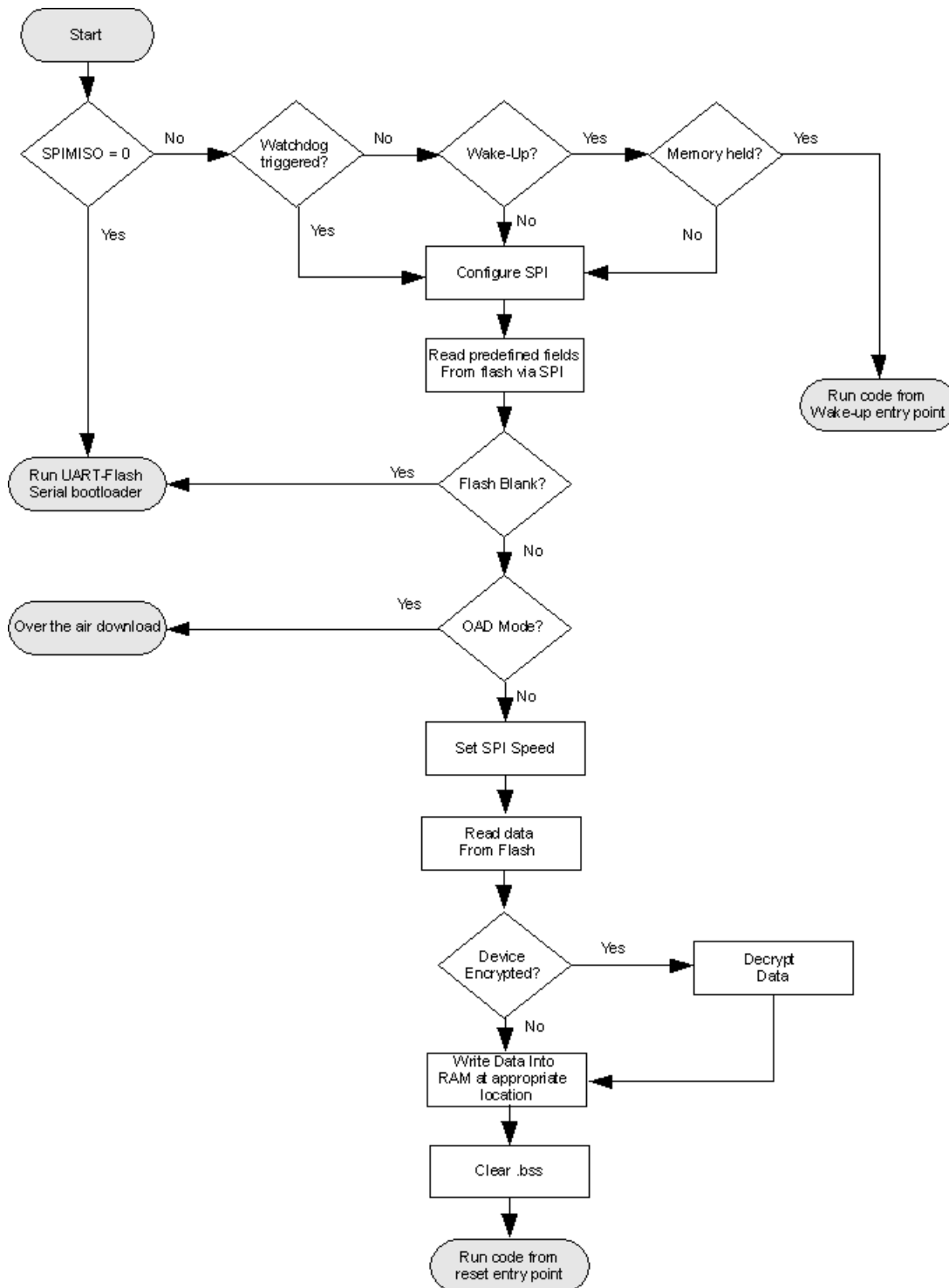


Figure 1: JN5148-001 Boot Loader Operation

JN5142 and JN5148-J01/Z01 Operation

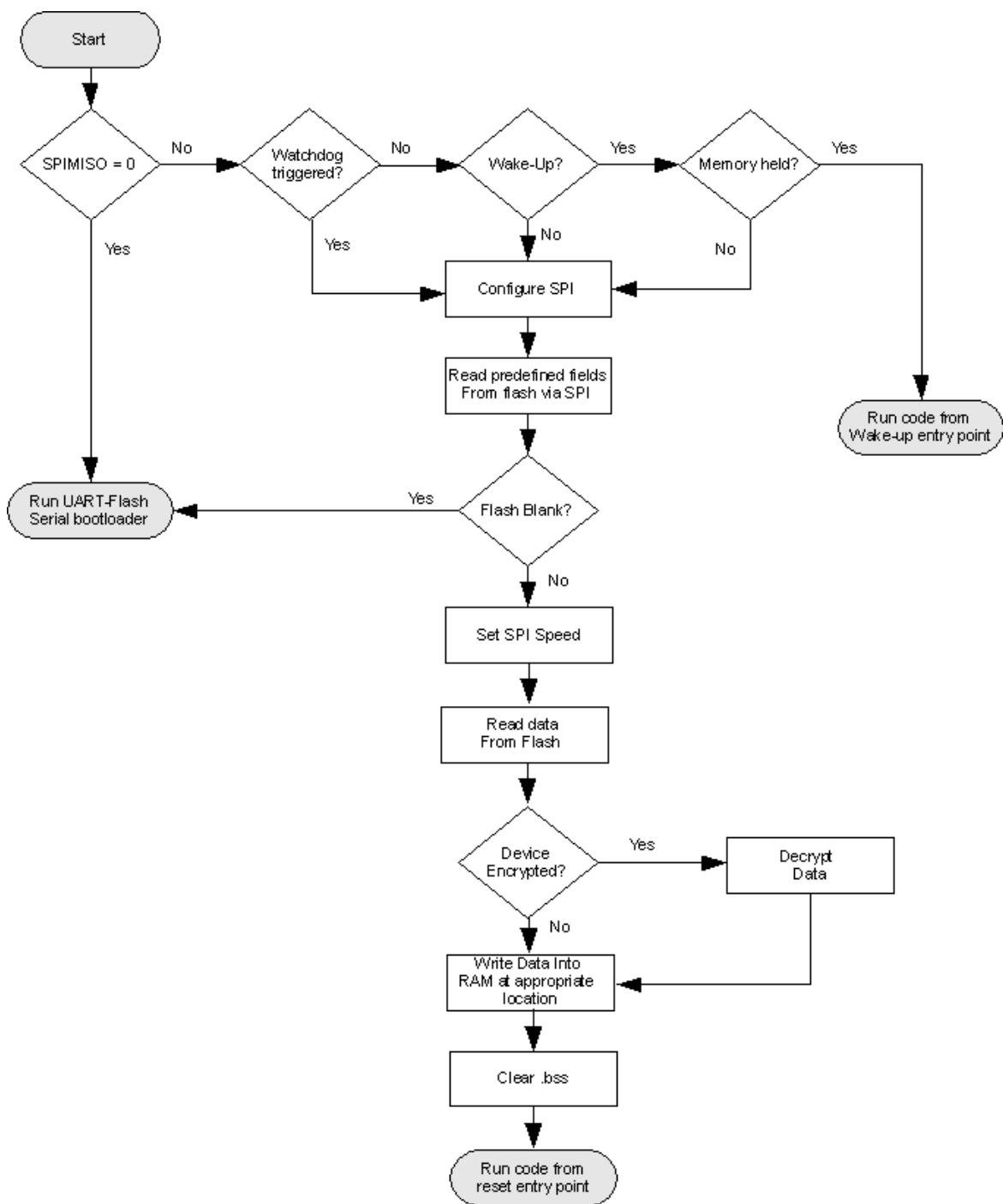


Figure 2: JN5142 and JN5148-J01/Z01 Boot Loader Operation

JN5139 Operation

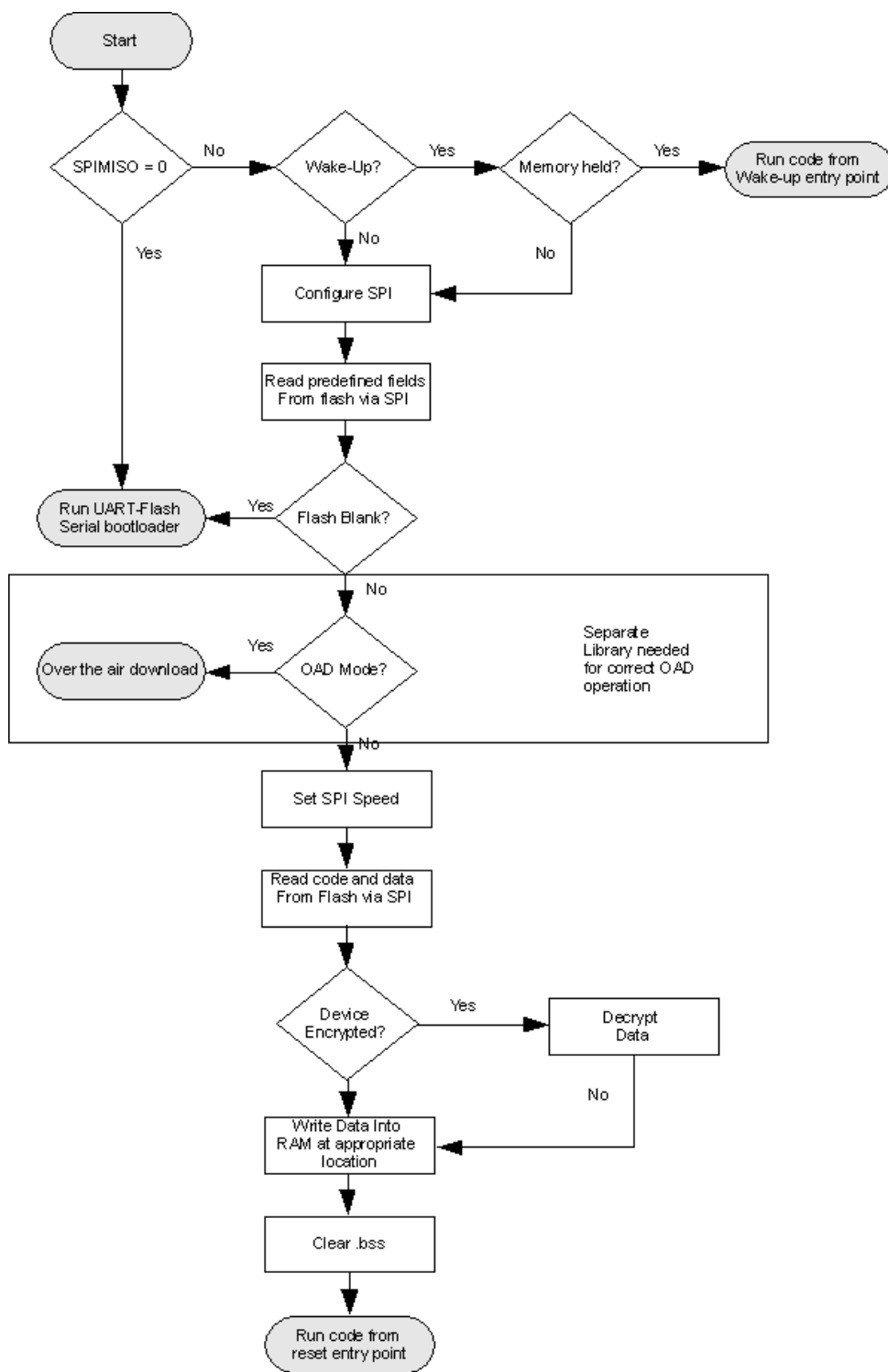


Figure 3: JN5139 Boot Loader Operation

JN516x/7x Boot Loader Operation

The JN516/7x boot loader performs one of the following operations:

- Starts execution from internal Flash memory
- Copies application from external Flash memory into internal Flash memory and starts application execution
- Enters programming mode

The application must contain two entry points from which execution may begin - a wake-up point and a reset point. The locations of these points are stored in the header information contained within the internal Flash memory. If the application puts the device to sleep using the **vAHI_Sleep()** function, on wake-up the boot loader will restart the application code from the wake-up entry point.

If no application is present in the internal Flash memory, the boot loader will search an optional external SPI Flash device for a valid application binary. If a binary is found, it will be loaded into internal Flash memory and executed from the reset entry point.

Alternatively, the boot loader may enter programming mode if the SPI MISO line is held low when the device is reset, or if no valid application image is found in the internal or external Flash memory.

JN516x/7x Operation

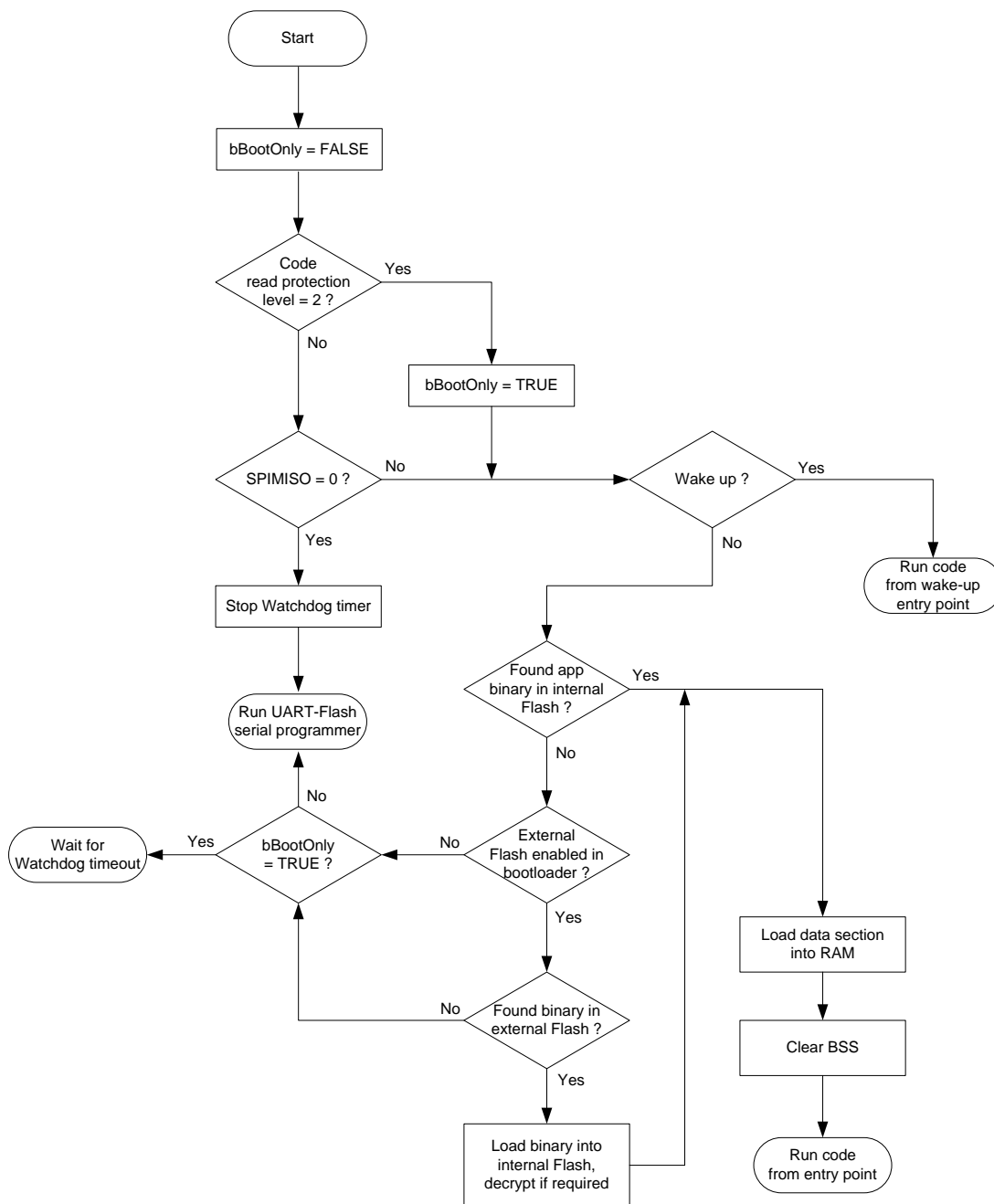


Figure 4: JN516x/7x Boot Loader Operation

JN516x/7x Image Selection

Following a reset, the JN516x/7x boot loader searches the internal Flash memory for a valid image. At the start of each sector, it checks the Boot Image Record (BIR), defined in Table 4, to determine whether it conforms to the following:

- Magic Number for JN516x: 0x123456781122334455667788
 Magic Number for JN517x: 0x123456782233445566778899
 (JN517x can also the Magic Number from the JN516x)
- Status is valid (any value other than 0x00 or 0xFF)
- Application ID matches the search value. By default this is 0x0000, but it is possible to specify a different value - this is to allow a second-stage boot loader (which would have an Application ID of 0) to choose a specific application and then use the boot loader to select and run it.

As soon as the boot loader finds the first valid BIR, it will remap the Flash memory so that the image is at the base address of the application area of this memory, and then runs that image. It does not check all valid images and then choose the best one - so if there are two valid images in Flash memory, it will always choose the one that starts at the lowest sector number.

If the boot loader does not find a valid BIR at the start of a sector, it moves to the next sector and tries again. This process repeats until it all sectors have been searched, at which point the boot loader checks the external Flash memory in the same way (if external Flash download is not disabled). If a valid image is found in external Flash memory, it is copied to internal Flash memory (with decryption performed, if required). However, if the search of external Flash memory fails to find a valid BIR, the boot loader enters programming mode or, if that is disabled, goes into a while (1) loop.

JN516x/7x Invalidating Images

To disable an image, the BIR must be invalidated by overwriting some of the data in it. One method to achieve this is to program the Status field byte to 0x00, which will mark the image as invalid. This can be accomplished using the Flash Memory functions detailed in the *JN516x Integrated Peripherals API User Guide (JN-UG-3087)* or *JN517x Integrated Peripherals API User Guide (JN-UG-3118)*.

JN516x/7x Flash Remapping

The JN516x and JN517x contain the ability to remap sectors of the internal Flash memory so that they appear at different locations within the memory map. The boot loader uses this functionality to ensure that, regardless of where an image is physically located, it appears to start at the beginning of the address space allocated to the internal Flash memory.

The Flash remapping uses two registers. These registers store the logical remapping of the physical sectors and, by default, are:

| Register Name | Default Value | Comment |
|----------------------|---------------|---|
| REG_SYS_FLASH_REMAP | 0x76543210 | Map physical sectors 0-7 to logical sectors 0-7 |
| REG_SYS_FLASH_REMAP2 | 0xFEDCBA98 | Map physical sectors 8-15 to logical sectors 8-15 |

The number of physical sectors for each device is shown below:

| Device | Physical Sectors | Comment |
|--------|------------------|--|
| JN5161 | 2 | Only REG_SYS_FLASH_REMAP register is available |
| JN5164 | 5 | Only REG_SYS_FLASH_REMAP register is available |
| JN5168 | 8 | Only REG_SYS_FLASH_REMAP register is available |
| JN5169 | 16 | REG_SYS_FLASH_REMAP2 register is also used |
| JN5179 | 16 | REG_SYS_FLASH_REMAP2 register is also used |

It should be noted that the Flash Memory functions (of the Integrated Peripherals API) always use logical (possibly remapped) addresses, rather than physical addresses.

Because of this Flash remapping, there is no need to modify the address to which the application is linked, since it will always start from 0x80000, and so the default linker file settings in the SDK remain valid. For example, in the case of JN5169, from the running application's viewpoint, it is always located in sectors 0 to 7 and the spare area is always sectors 8 to 15 (assuming the application requires 8 sectors of storage, i.e. it is between 224 and 256 Kbytes in size).

The boot loader only remaps the sectors containing application image data. If the application is less than 224 Kbytes in size (i.e. requires 7 sectors or less for storage), only sectors 0 to 6 are remapped. As a result, when the device is running a 7-sector image in the second image 'space' (i.e. in physical sectors 8-14), the boot loader will only remap those sectors, so the Flash mapping will be as illustrated below.

| Logical Sector | Physical Sector | Comment |
|----------------|-----------------|-----------------|
| 0 | 8 | Swapped with 0 |
| 1 | 9 | Swapped with 1 |
| 2 | 10 | Swapped with 2 |
| 3 | 11 | Swapped with 3 |
| 4 | 12 | Swapped with 4 |
| 5 | 13 | Swapped with 5 |
| 6 | 14 | Swapped with 6 |
| 7 | 7 | Not swapped |
| 8 | 0 | Swapped with 8 |
| 9 | 1 | Swapped with 9 |
| 10 | 2 | Swapped with 10 |
| 11 | 3 | Swapped with 11 |
| 12 | 4 | Swapped with 12 |
| 13 | 5 | Swapped with 13 |
| 14 | 6 | Swapped with 14 |
| 15 | 15 | Not swapped |

This implementation allows one or more sectors to be left unswapped at the end of the memory space, so that they can be used for permanent data storage etc.

When implementing an in-service application upgrade system (e.g. ZigBee "Over-The-Air" (OTA)), an issue may occur if the new image that has been downloaded requires more sectors for storage than the current image being executed. The boot loader only remaps the sectors that it needs to. Therefore, if your application is only 6 sectors long, only sectors 0 to 5 will be remapped. As a result, when the device is running a 6-sector image in the second

image 'space' (i.e. in physical sectors 8-13), the boot loader will only remap those sectors, so the Flash mapping will be as illustrated below.

| Logical Sector | Physical Sector | Comment |
|----------------|-----------------|-----------------|
| 0 | 8 | Swapped with 0 |
| 1 | 9 | Swapped with 1 |
| 2 | 10 | Swapped with 2 |
| 3 | 11 | Swapped with 3 |
| 4 | 12 | Swapped with 4 |
| 5 | 13 | Swapped with 5 |
| 6 | 6 | Not swapped |
| 7 | 7 | Not swapped |
| 8 | 0 | Swapped with 8 |
| 9 | 1 | Swapped with 9 |
| 10 | 2 | Swapped with 10 |
| 11 | 3 | Swapped with 11 |
| 12 | 4 | Swapped with 12 |
| 13 | 5 | Swapped with 13 |
| 14 | 14 | Not swapped |
| 15 | 15 | Not swapped |

However, if the upgrade system does not account for the discontinuity in the physical sector mapping and the new image is larger than the old one (enough to require an additional sector of storage), it will simply place the new image at logical sectors 8-14, which in this case is not a continuous block of physical sectors. Then, when the boot loader remaps to it, it will assume it is in physical sectors 0-6, when in fact, sector 6 is undefined.

One solution is to manually remap the Flash sectors during initialisation. If the application sees that the current image does not start at sector 0, it forces the mapping to this:

| Logical Sector | Physical Sector | Comment |
|----------------|-----------------|---------------------|
| 0 | 8 | Already swapped |
| 1 | 9 | Already swapped |
| 2 | 10 | Already swapped |
| 3 | 11 | Already swapped |
| 4 | 12 | Already swapped |
| 5 | 13 | Already swapped |
| 6 | 14 | Now swapped with 6 |
| 7 | 7 | Not swapped |
| 8 | 0 | Already swapped |
| 9 | 1 | Already swapped |
| 10 | 2 | Already swapped |
| 11 | 3 | Already swapped |
| 12 | 4 | Already swapped |
| 13 | 5 | Already swapped |
| 14 | 6 | Now swapped with 14 |
| 15 | 7 | Not swapped |

The code required to implement this remapping is as follows:

```
u8CurrentImageSector = u32REG_SysRead(REG_SYS_FLASH_REMAP) & 0xf;

/* If u8CurrentImageSector is 0, there has been no remapping, so no need to
do anything. Otherwise, ensure remap is set as 3 blocks of sequential
sectors. */

if (u8CurrentImageSector > 0)
{
    /* Remapping will not affect the current running image,
    which was already running in a continuous block at the base
    application Flash address */
    vREG_SysWrite(REG_SYS_FLASH_REMAP, 0x7EDCBA98);
    vREG_SysWrite(REG_SYS_FLASH_REMAP2, 0xf6543210);
}
```

The header file **PeripheralRegs_JN5169.h** (in **Components/HardwareApi/Include**) will need to be added to the application when using this example code on the JN5169 device.



Note: To achieve backward compatibility with the JN513x/JN514x devices, the Flash Memory functions assume that the Flash address space starts at 0, and it adds 0x80000 internally.

JN516x/7x Code Read Protection (CRP)

Code Read Protection is a mechanism that allows the user to enable different levels of security so that access to the JN516x/7x boot loader via the serial interface can be restricted. The CRP level can be set using the JN51xx Production Flash Programmer (JN-SW-4107).

CRP Level 0 – No Protection (Default)

All commands provided via the boot loader serial interface can be used.

CRP Level 1 - Flash read protection

The contents of internal Flash memory cannot be read via the boot loader serial interface. However, the internal Flash memory can still be erased and written - see Table 8 for a complete list of all serial commands supported in this mode.

CRP Level 2

In this mode, all access to the boot loader serial interface is disabled. If this level of security is used then the JTAG interface should also be disabled using the JN51xx Production Flash Programmer (JN-SW-4107).



Note: Any CRP change becomes effective only after the device has gone through a power cycle.

JN516x/7x Encrypting Images in External Flash Memory

It may be necessary to store an application image in external serial Flash memory. This can be the case when the application image is greater than half the size of memory available, meaning that only one image can be stored within the internal Flash memory. During an

in-service upgrade of such an application, the new image must be stored in external serial Flash memory. Once the download is complete and the image has been validated, the boot loader will copy it to internal Flash memory. Data is copied from external to internal Flash memory starting with the image header and then proceeding with the image body. However, the magic number present in the BIR is not written until the entire image has been successfully programmed into internal Flash. This prevents the boot loader from attempting to execute an image that was not completely copied due to issues such as a power interruption.

It is possible to encrypt the image located within external Flash memory and have the boot loader decrypt it as it is copied to internal Flash memory. This encryption and decryption is performed using the AES algorithm in conjunction with a 128-bit key.

The key to be used by the boot loader to decrypt the image must be programmed into the device using the JN51xx Production Flash Programmer (JN-SW-4107). In addition, CRP level 2 and the External Flash Decrypt options must be configured.

The image that will be stored in external flash memory can be encrypted using the JET tool that is described in the *JET User Guide (JN-UG-3081)*.



Note: The use of external serial Flash memory by the boot loader can be disabled using the JN51xx Production Flash Programmer (JN-SW-4107).

JN516x/7x Index Sector

The JN516x/7x devices contain a quantity of One Time Programmable (OTP) memory as part of the embedded Flash (also referred to as the Index Sector). This can be used to securely hold such things as a user 64-bit MAC address, a 128-bit AES security key and general user data. The index sector is organised as sixteen 128-bit words.


| Word | Address | Usage |
|------|------------|------------------------|
| 0 | 0x01001500 | Internal use |
| 1 | 0x01001510 | Customer Settings |
| 2 | 0x01001520 | Trim Values |
| 3 | 0x01001530 | Reserved |
| 4 | 0x01001540 | Customer Field 0 |
| 5 | 0x01001550 | Customer Field 1 |
| 6 | 0x01001560 | Customer Field 2 |
| 7 | 0x01001570 | Customer MAC Address |
| 8 | 0x01001580 | Device MAC Address |
| 9 | 0x01001590 | Internal use |
| 10 | 0x010015A0 | Internal use |
| 11 | 0x010015B0 | Internal use |
| 12 | 0x010015C0 | External Flash AES Key |
| 13 | 0x010015D0 | Reserved |
| 14 | 0x010015E0 | DC-DC converter |
| 15 | 0x010015F0 | Reserved |

Table 1: JN516x/7x Index Sector Organisation

The contents of each word are described in the following sections.


Customer Settings

| Bit | Field | Description | Default |
|--------|------------|--|---------|
| 127:96 | - | Repeat of bits 31:0 | - |
| 95:64 | - | Repeat of bits 31:0 | - |
| 63:32 | - | Repeat of bits 31:0 | - |
| 31:8 | Reserved | - | - |
| 7 | FLASH_LOAD | If 0 then loading firmware from external Flash is disabled | 0x1 |
| 6 | FLASH_ENC | If 0 then the contents of external Flash are decrypted (using AES key contained in word 12) as they are loaded into internal Flash memory | 0x1 |
| 5:4 | CRP_LEVEL | Code Read Protection (CRP) level: 00 = CRP level 2 (no programming allowed via UART) 01 = CRP level 1 (Flash read protection) 10 = CRP level 2 (no programming allowed via UART) 11 = CRP level 0 (no protection) | 0x3 |
| 3:1 | VBO_THR | Customer alterable Voltage Brown-Out (VBO) threshold setting. The value programmed automatically controls the VBO threshold settings to allow the user to alter the default starting voltage of the device: 000 = 2.7V 001 = 3.0V 010 = 2.3V 011 = 2.4V 100 = 2.1V 101 = 2.2V 110 = 1.95V 111 = 2.0V | 0x7 |
| 0 | CPU_JTAG | If 0 then CPU JTAG access is disabled | 0x1 |

 **Note:** This word must only be written once. For example, you cannot write part of the word and then alter some further bits later. The programmer will automatically repeat the values across bits 127:96, 95:64, 3:32 and 31:0.

Trim Values

| Bit | Field | Description | Default |
|--------|------------|---|---------|
| 127:96 | - | Repeat of bits 31:0 | - |
| 95:64 | - | Repeat of bits 31:0 | - |
| 63:32 | - | Repeat of bits 31:0 | - |
| 31:4 | Reserved | - | - |
| 3:0 | HSRCO_TRIM | If not 0xF (indicating trimming code not determined), this is the trimming code determined during device test to get the high-speed RC oscillator close to 32MHz. | 0xF |


 **Note:** This word is read only.

Customer Fields

| Bit | Field | Description | Default |
|-------|-------|---------------------------|---------|
| 0:127 | CF_0 | 128 bits for customer use | All 1s |


| Bit | Field | Description | Default |
|-------|-------|---------------------------|---------|
| 0:127 | CF_1 | 128 bits for customer use | All 1s |

| Bit | Field | Description | Default |
|-------|-------|---------------------------|---------|
| 0:127 | CF_2 | 128 bits for customer use | All 1s |

 **Note:** Each word must only be written once. For example, you cannot write part of the word and then alter some further bits later.


Customer MAC Address

| Bit | Field | Description | Default |
|--------|--------|--|---------|
| 127:64 | CF_3 | 64 bits for customer use | All 1s |
| 63:32 | CMAC_L | Lower 32 bits of customer programmed MAC address | All 1s |
| 31:0 | CMAC_H | Upper 32 bits of customer programmed MAC address | All 1s |

 **Note:** This word must only be written once. For example, you cannot write part of the word and then alter some further bits later. Once an address has been written to this word, it will be displayed by the Flash programmer used to access the JN516x/7x device.

Device MAC Address

| Bit | Field | Description | Default |
|--------|----------|--|---------|
| 127:64 | Reserved | - | - |
| 63:32 | MAC_L | Lower 32 bits of MAC programmed during manufacturing | - |
| 31:0 | MAC_H | Upper 32 bits of MAC programmed during manufacturing | - |

 **Note:** This word is read only.

External Flash AES Key

| Bit | Field | Description | Default |
|-------|---------|--|---------|
| 0:127 | AES_KEY | AES key used by boot loader to decrypt contents of external Flash as they are copied into internal Flash (when FLASH_ENC field in Customer Settings word is set to 0). | All 1s |



Note: This word must only be written once. For example, you cannot write part of the word and then alter some further bits later.

Read/Writing Index Sector Words

All words within the Index Sector can be read by a user's application firmware simply by reading the memory address given in Table 1.

Most words can be read by the JN51xx Production Flash Programmer (JN-SW-4107). Writing to words within the Index sector can also be performed using this Flash Programmer.

Flash Header

Any user application programmed into the Flash memory device must include a header that contains information about the program and the device in which it is stored. The format of this header is detailed below for each JN517x/JN516x/JN514x/JN5139 microcontroller.

JN5148-001 Flash Header

| Bytes | Word | Contents |
|------------------|------|--|
| 0x0000 to 0x0003 | 0 | Configuration bytes: 0xAABBCCDD AA - Reserved (always set to 0) BB - Reserved (AA repeat, always 0) CC - Flash access configuration byte (see below for details) DD - Flash access configuration byte (repeated) |
| 0x0004 to 0x0007 | 1 | The start address in RAM where the application data needs to be copied to (word-aligned). The data copied to this address will start from Flash address 0x30 |
| 0x0008 to 0x000B | 2 | Length of data to copy to RAM from Flash address 0x30 (includes sections: .mac, .rodata, .data, .text, .bss) |
| 0x000C to 0x000F | 3 | Chip ROM version for which the binary was compiled |
| 0x0010 to 0x0013 | 4 | Must be 0 |
| 0x0014 to 0x0017 | 5 | Start address for .bss segment in RAM (word-aligned) |
| 0x0018 to 0x001B | 6 | Length of .bss segment in RAM |
| 0x001C to 0x001F | 7 | Wake-up entry point (word-aligned) (address of AppWarmStart()) |
| 0x0020 to 0x0023 | 8 | Reset entry point (word-aligned) (address of AppColdStart()) |
| 0x0024 to 0x0027 | 9 | OAD (Over-Air Download) channel scan |
| 0x0028 to 0x0029 | 10 | OAD channel scan bitmap |
| 0x002A to 0x002B | | OAD application/server identification |
| 0x002C to 0x002D | 11 | Unused |
| 0x002E | | Revert and valid flags for OAD |
| 0x002F | | Invalid flags for OAD |
| 0x0030 to End | | Application data The first 32 bytes contain an 8-byte MAC address and 24 bytes for ZigBee use |

Table 2: JN5148-001 Flash Header

The Flash header is defined at compile/link time by the chip-specific linker file located in the SDK directory at **Chip\<Chip Name>\Build**.

Flash Access Configuration Byte

If the first Flash access configuration byte is 0xFF, this indicates a blank Flash memory. Otherwise, the Flash access configuration byte is interpreted as follows:

| Bit | Contents |
|-----|--|
| 7:6 | Always set to 3 |
| 5 | Address field supported: 0 for 16-bit address 1 for 24-bit address |
| 4:0 | SPI clock divider value to be used. This will be used to set the SPI divider setting in the SPI master, padded with a 0 as the MSB of the 6-bit divider field (0x1F is not a valid setting for this field) |

The first four bytes will be read from Flash memory with a SPI clock speed of 1 MHz. The boot loader will then use the SPI clock divider from the Flash access configuration byte to set the SPI clock for the remaining reads.

JN5142 and JN5148-J01/Z01 Flash Header

| Bytes | Word | Contents |
|----------------------|-------|---|
| 0x0000 to 0x000F | 0 - 3 | 16-byte Boot Image Record |
| 0x0010 to 0x0017 | 4 - 5 | 64-bit MAC address |
| 0x0018 to 0x0027 | 6 - 9 | Encryption Initialisation Vector (ignored if unencrypted) |
| 0x0028 to 0x0029 | 10 | 16-bit load address for .text segment in RAM (word aligned) |
| 0x002A to 0x002B | 10 | 16-bit length of .text segment, in 32-bit words |
| 0x002C to 0x002D | 11 | 16-bit load address for .bss segment in RAM (word aligned) |
| 0x002E to 0x002F | 11 | 16-bit length of .bss segment in RAM, in 32-bit words |
| 0x0030 to 0x0033 | 12 | 32-bit wake-up entry point (word aligned) – warm start |
| 0x0034 to 0x0037 | 13 | 32-bit reset entry point (word aligned) – cold start |
| 0x0038 to (MemA – 1) | 14 - | .text segment |
| MemA to (MemB-1) | | .data segment |
| MemB | | Overlay segment |

Table 3: JN5142 and JN5148-J01/Z01 Flash Header

MemA = (Length of .text segment x 4) + 0x0038

MemB = (Length of .data segment x 4) + MemA

Note: A built binary file contains an extra 4 bytes (for the version number) at the beginning of the file which are stripped out when the file is stored in Flash memory.

The Flash header is defined at compile/link time by the chip-specific linker file located in the SDK directory at **Chip\<Chip Name>\Build**.

JN51xx Boot Image Record (BIR)

The 16-byte Boot Image Record, contained in words 0-3 of the Flash header, is structured as follows:

| Name | Size | Comment |
|----------------|----------|---|
| Magic Number | 12 bytes | Pattern to denote start sector of boot image |
| Configuration | 1 byte | Used to carry configuration information to set up the SPI to suit the Flash device, and size of the image contained in Flash memory Bits 7:4 – Image size in 32-Kbyte blocks: 0000: 32K 0001: 64K 0010: 96K 0011: 128K 0100: 160K 0101: 192K 0110: 224K 0111: 256K 1000: 288K 1001: 320K 1010: 352K 1011: 384K 1100: 416K 1101: 448K 1110: 480K 1111: 512K Bit 3 - Delay Read to next SPI clock edge (DRE) hardware option to enhance reliability of reading data from slower devices: 0 = Disabled 1 = Enabled Bits 2:0 - SPI clock rate: 000: 16 MHz 001: 8 MHz 010: 4 MHz 011: 2 MHz 100: 1 MHz 101: 0.5 MHz 110, 111 reserved |
| Status | 1 byte | 0xFF – empty (erased) 0x00 – invalid 0x01 – valid 0x02 to 0xFE – reserved |
| Application ID | 2 bytes | |

Table 4: Boot Image Record (BIR) for JN5142, JN5148-J01/Z01 and JN516x/7x

The boot loader searches for the first image that is valid and that has an Application ID in **BIR>Configuration>Application ID** which matches the search (default search is 0). If none is found, it enters boot programming mode for communication over the UART. All BIRs are aligned to the minimum image-size boundary of 32 Kbytes. They will be further apart if the images are larger than 32 Kbytes. In such cases, the **BIR>Configuration>Image size** field allows the boot loader to find the next image, assuming that the BIR is deemed valid (note that a BIR is deemed valid if the magic number is correct). The **BIR>Status** field refers to the status of the rest of the image after the BIR.

JN5139 Flash Header

The JN5139 Flash header is as described for the JN5148 device in Table 2 except for the reserved fields at Flash addresses 0x00 and 0x01, which should always be 7 (instead of 0). The information provided for JN5148 on the 'Flash access configuration byte' is also applicable to the JN5139 device.

The Flash header is defined at compile/link time by the chip-specific linker file located in the SDK directory at **Chip\.**

JN516x Flash Header

| Bytes | Word | Contents |
|----------------------|-------|---|
| 0x00 - 0x0F | 0 - 3 | 16-byte Boot Image Record |
| 0x10 - 0x1D | 4 - 7 | Encryption Initialisation Vector (ignored if unencrypted) |
| 0x1E - 0x1F | 7 | 16-bit Software Configuration Options |
| 0x20 - 0x23 | 8 | 32-bit Length of Binary Image in bytes |
| 0x24 - 0x27 | 9 | 32-bit .data section Flash start address |
| 0x28 - 0x29 | 10 | 16-bit .data section load address in RAM (word aligned) |
| 0x2A - 0x2B | 10 | 16-bit .data section length in 32-bit words |
| 0x2C - 0x2D | 11 | 16 bit .bss section start address in RAM (word aligned) |
| 0x2E - 0x2F | 11 | 16-bit .bss section length in 32-bit words |
| 0x30 - 0x33 | 12 | 32-bit wake-up entry point (word aligned) – warm start |
| 0x34 - 0x37 | 13 | 32-bit reset entry point (word aligned) – cold start |
| 0x0038 to (MemA - 1) | 14 - | .text segment |
| MemA | | .data segment |

Table 5: JN516x Flash Header

MemA = (Length of .text segment x 4) + 0x0038

MemB = (Length of .data segment x 4) + MemA



Note: A built binary file contains an extra 4 bytes (for the version number) at the beginning of the file which are stripped out when the file is stored in Flash memory.

The Flash header is defined at compile/link time by the chip-specific linker file located in the SDK directory at **Chip\. The 16-byte Boot Image Record (BIR) is the same as described in Table 3.**

JN517x Flash Header

| Bytes | Word | Contents |
|---------------------|-------|---|
| 0x00 - 0x03 | 0 | Version Number |
| 0x04 - 0x13 | 1 – 4 | 16-byte Boot Image Record |
| 0x14 – 0x21 | 5 – 8 | Encryption Initialisation Vector (ignored if unencrypted) |
| 0x22 – 0x23 | 8 | 16-bit Software Configuration Options |
| 0x24 – 0x27 | 9 | 32-bit Length of Binary Image in bytes |
| 0x28 – 0x2B | 10 | 32-bit .data section Flash start address |
| 0x2C – 0x2D | 11 | 16-bit .data section load address in RAM (word aligned) |
| 0x2E – 0x2F | 11 | 16-bit .data section length in 32-bit words |
| 0x30 – 0x31 | 12 | 16 bit .bss section start address in RAM (word aligned) |
| 0x32 – 0x33 | 12 | 16-bit .bss section length in 32-bit words |
| 0x34 – 0x37 | 13 | 32-bit wake-up entry point (word aligned) – warm start |
| 0x38 - 0x3B | 14 | 32-bit reset entry point (word aligned) – cold start |
| 0x003C to (MemA –1) | 15 - | .text segment |
| MemA | | .data segment |

Table 6: JN517x Flash Header

MemA = (Length of .text segment x 4) + 0x003C

MemB = (Length of .data segment x 4) + MemA

Software Configuration Options

Before the boot loader jumps to the application entry point, the JTAG debugger is configured as detailed in the options below.

JN516x

| Bit | Contents |
|------|--|
| 2-15 | Unused |
| 1 | 0 – JTAG Debug on DIO4-7 1 – JTAG Debug on DIO12-15 |
| 0 | Debug Enable (1 Enable, 0 Disable) |

JN517x

| Bit | Contents |
|-----|--------------------------------------|
| 8 | Debug Enable 0 – Disable, 1 - Enable |
| 7 | 0 – TDI on DIO9, 1 – TDI on DI015 |
| 6 | 0 – TDO on DIO9, 1 – TDO on DI015 |
| 5 | 0 – TMS on DIO11, 1 – TMS on DI018 |
| 4 | 0 – TCK on DIO17, 1 – TCK on DIO6 |
| 3 | TDO Enable |
| 2 | TDO Enable |
| 1 | TMS Enable |
| 0 | TCK Enable |

External (Flash) Memory

The external serial memory must be connected to the SPI port using the SPISEL0 select line.

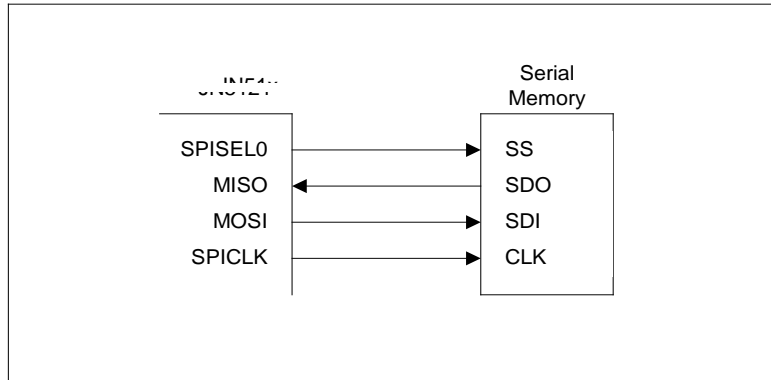


Figure 5: Connecting External Serial Memory

JN516x/7x Supported Devices

The JN516x/7x series support the following devices:

- SST 25VF0101 (128 Kbytes)
- ATMEL AT25F512 (64 Kbytes)
- ST M25P10-A (128 Kbytes)
- ST M25P40 (512 Kbytes)

JN5148 Supported Devices

The JN5148 supports the following devices:

- SST 25VF0101 (128 Kbytes)
- ATMEL AT25F512 (64 Kbytes)
- ST M25P10-A (128 Kbytes)
- ST M25P40 (512 Kbytes)

JN5142 Supported Devices

The JN5142 supports all the above devices (as for JN5148) plus the following devices:

- ST M25P05-A (64 Kbytes)
- ST M25P20-A (256 Kbytes)

JN5139 Supported Devices

The JN5139 supports the following devices:

- SST 25VF0101 (128 Kbytes)
- ATMEL AT25F512 (64 Kbytes)
- ST M25P10-A (128 Kbytes)

Memory Maps

JN514x/JN5139 Memory Maps

The JN5148 device has 128 Kbytes of RAM, the JN5142 device has 32 Kbytes of RAM, and the JN5139 device has 96 Kbytes of RAM.

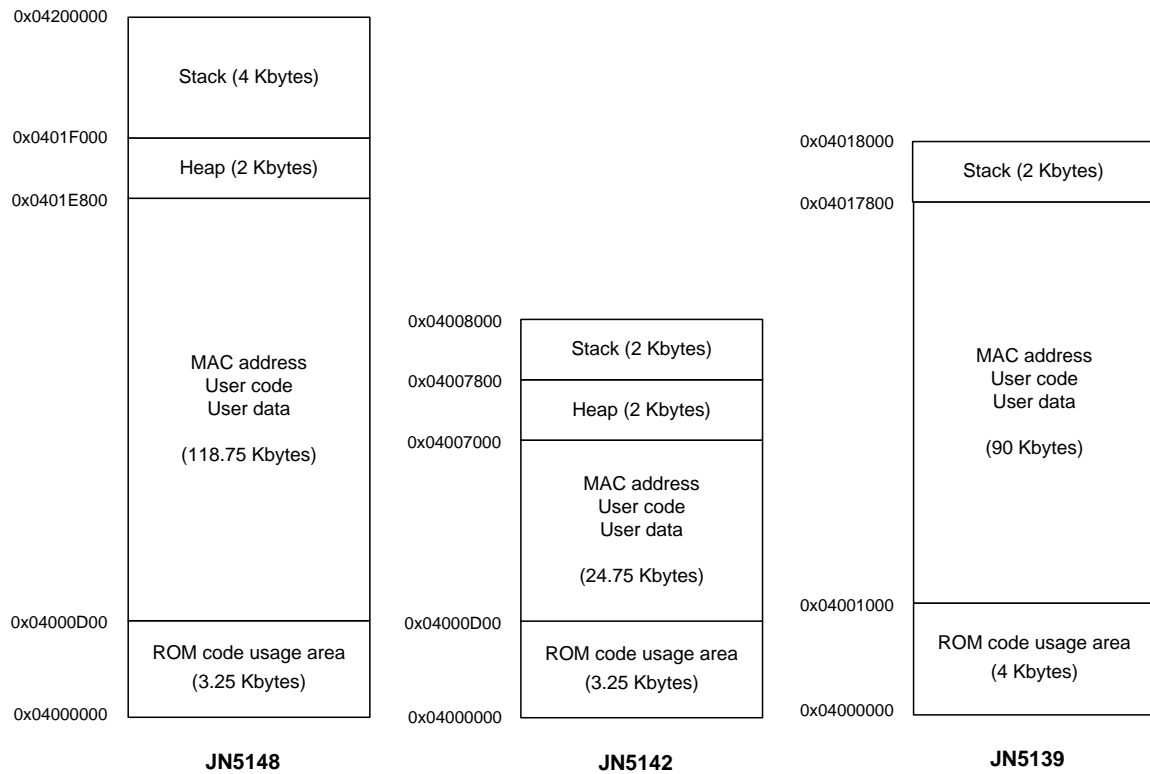


Figure 6: JN514x/JN5139 Memory Maps

JN516x/7x Memory Maps

On the JN516x and JN517x devices, user application code is stored and executed in the internal Flash memory, while user data is stored in RAM. The internal Flash memory is memory-mapped and can be read using normal memory read accesses. However, writes to the internal Flash memory must be done using functions of the JN516x and JN517x Integrated Peripherals API.

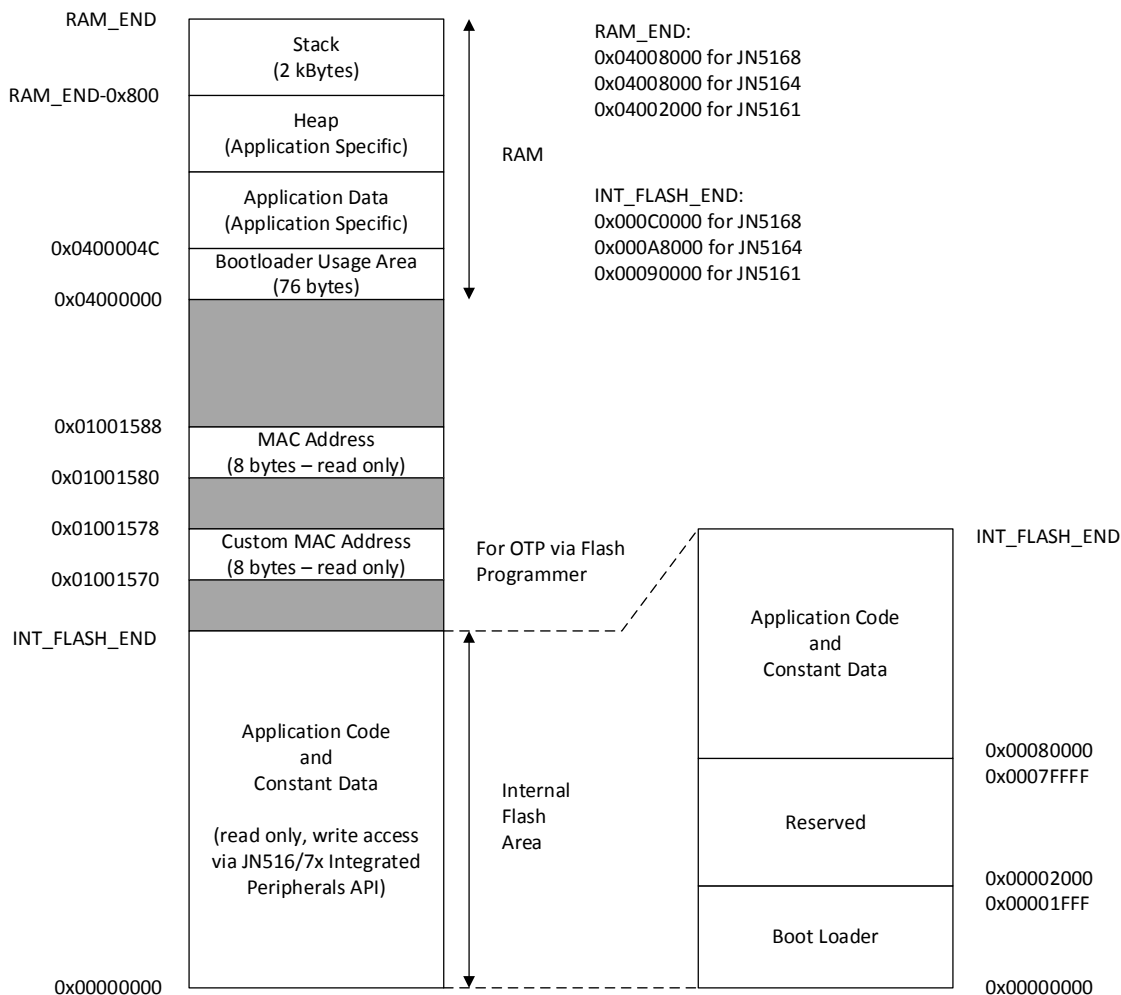


Figure 7: JN516x Memory Maps

JN51xx Boot Loader Operation

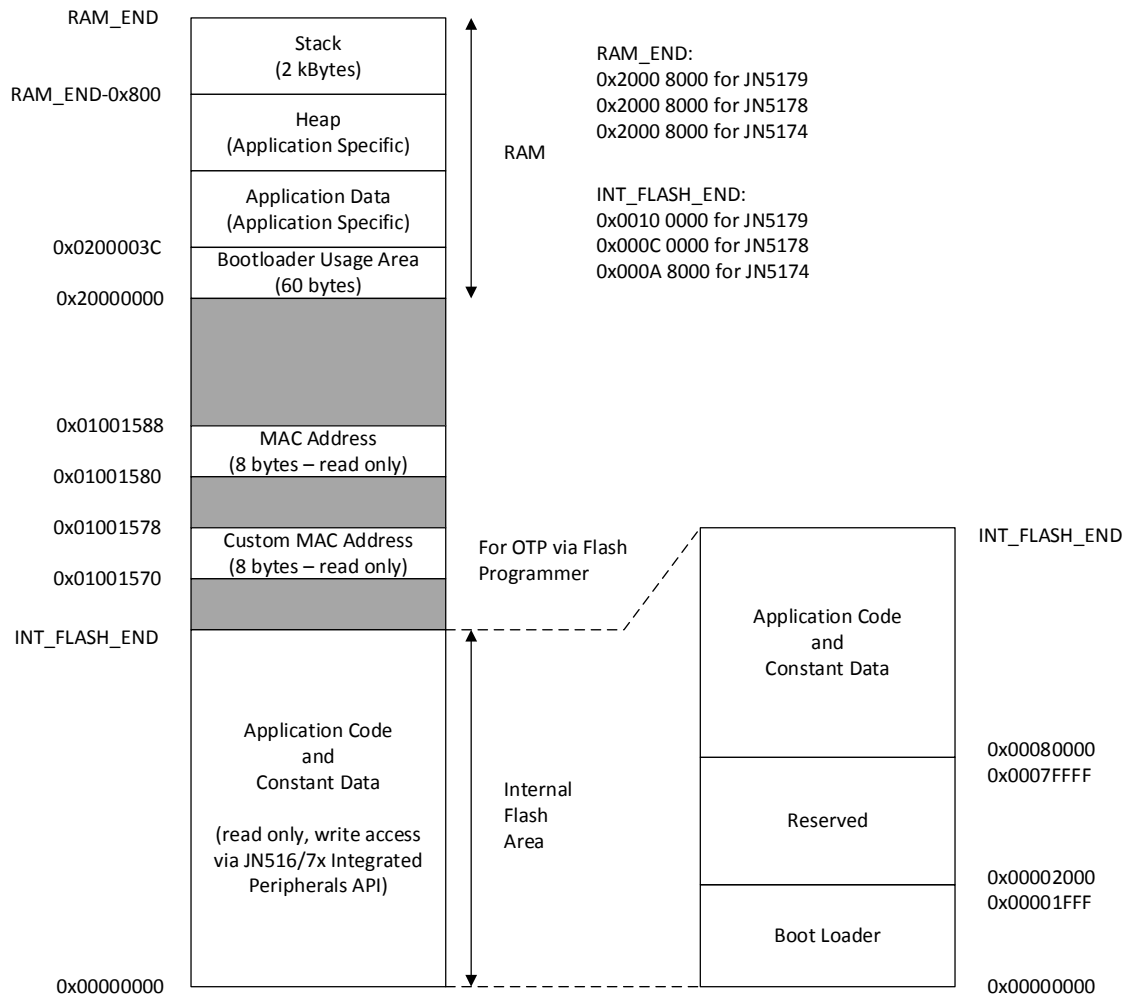


Figure 8 JN517x Memory Map



Note: The MAC address is read-only. A custom MAC address can be programmed which supersedes the production MAC address. However, this is a 'one time programmable' process, which cannot be reversed. The custom MAC address can be programmed via the JN51xx Flash programmer.

Serial Protocol

The protocol defines a message structure and set of messages that allow a client to communicate with the JN514x/JN5139 chip, and hence perform operations on the external Flash memory. Serial communication between the PC and boot loader is implemented as 8,N,1,38K4 from the JN514x/JN5139 UART 0.

The JN514x/JN5139 device will respond to the serial protocol when it has entered programming mode, either due to a reset when the MISO line is asserted or if the Flash chip is deemed to be blank.

Message Structure

A message contains four fields and is structured as follows:

| Field | 1 | 2 | 3 | 4 |
|-------|--------|--------------|--------------|----------|
| Bytes | 1 | 1 | (x) | 1 |
| Data | Length | Message Type | Message Data | Checksum |

Table 7: Message Structure

Length The number of bytes in the rest of the message, including the checksum. Note that the maximum total message size is 255.

Message Type Defined below.

Message Data Variable length and dependent on message type. May be zero-length (i.e. not present)

Note: For Flash/RAM Read/Write requests, a maximum of 128 bytes should be read or written.

Checksum Calculated over the preceding part of the message, including the Length field, by implementing a logical Exclusive-OR operation on the previous bytes.

Byte Transfer Timeout

A timeout is applied to the reception of each individual byte of a message. This timeout depends on the CPU clock frequency. When using the 16-MHz clock, in your download program you are advised not to allow more than 5 seconds between bytes transmitted.

Message Types

Message types are as follows (N/A means that an optional field is not present):

| Message Type | Meaning | Originator | Supported Devices | # Parmes | Message Parameters | Response Message Type | Enabled in CRP1 |
|--------------|------------------------------------|-------------------|--|----------|--|-----------------------|-----------------|
| 0x00 to 0x06 | Reserved | N/A | N/A | N/A | N/A | N/A | No |
| 0x07 | Flash erase request (All Sectors) | PC | JN5148 JN5142 JN5139 JN516x JN517x | 0 | N/A | 0x08 | Yes |
| 0x08 | Flash erase response | JN514x/ JN5139 | JN5148 JN5142 JN5139 JN516x JN517x | 1 | #1: Status (1 byte): 0x00: Erase successful 0xFF: Parameter error 0xF7: Auth error | N/A | Yes |
| 0x09 | Flash program request | PC | JN5148 JN5142 JN5139 JN516x JN517x | 2 | #1: Flash address (4 bytes), LSB first #2: Up to 128 bytes of data Note: Data should not span page write boundary as defined by the Flash specification. | 0x0A | Yes |
| 0x0A | Flash program response | JN514x/ JN5139 | JN5148 JN5142 JN5139 JN516x JN517x | 1 | #1: Status (1 byte): 0x00: Program successful 0xFF: Readback verify failed 0xF7: Auth error | N/A | Yes |
| 0x0B | Flash read request | PC | JN5148 JN5142 JN5139 JN516x JN517x | 2 | #1: Flash address (4 bytes), LSB first #2: Length (2 bytes), LSB first (Length should not be greater than 128 bytes) | 0x0C | No |
| 0x0C | Flash read response | JN514x/ JN5139 | JN5148 JN5142 JN5139 JN516x JN517x | 2 | #1: Status (1 byte): 0x00 (success) 0xF7 (Auth error) #2: Bytes read from Flash | N/A | No |
| 0x0D | Sector erase request | PC | JN5148 JN5142 JN5139 JN516x JN517x | 1 | #1: Sector to erase (1 Byte) Note: Ensure the write protection is disabled by writing 0 to the status register using message 0x0F. | 0x0E | Yes |
| 0x0E | Sector erase response | JN514x/ JN5139 | JN5148 JN5142 JN5139 JN516x JN517x | 1 | #1: Status (1 byte): 0x00: Erase successful 0xFF: Erase error 0xF7: Auth error | N/A | Yes |
| 0x0F | Write SR (status register) request | PC | JN5148 JN5142 JN5139 JN516x JN517x | 1 | #1: SR Value (1 byte) | 0x10 | Yes |

JN51xx Boot Loader Operation

| Message Type | Meaning | Originator | Supported Devices | # Params | Message Parameters | Response Message Type | Enabled in CRP1 |
|--------------|----------------------------|--------------------------------------|--|----------|---|-----------------------|-----------------|
| 0x10 | Write SR response | JN514x/ JN5139 | JN5148 JN5142 JN5139 JN516x JN517x | 1 | #1: Status (1 byte): 0x00: Write successful 0xFF: Write error 0xF7: Auth error | N/A | Yes |
| 0x14 | Set Reset request | PC | JN5148 JN5142 JN5139 JN516x JN517x | 0 | N/A | 0x15 | Yes |
| 0x15 | Set Reset request response | JN5139 JN514x JN516x JN517x | JN5148 JN5142 JN5139 JN516x JN517x | 1 | #1: Status (1 byte): 0x00: success | N/A | Yes |
| 0x1D | RAM write request | PC | JN5148 JN5142 JN5139 JN516x JN517x | 2 | #1: RAM address (4 bytes), LSB first #2: Bytes to write into RAM (Up to 128 bytes) | 0x1E | No |
| 0x1E | RAM write response | JN514x/ JN5139 | JN5148 JN5142 JN5139 JN516x JN517x | 1 | #1: Status (1 byte): 0x00 (success) 0xF7 (Auth error) | N/A | No |
| 0x1F | RAM read request | PC | JN5148 JN5142 JN5139 JN516x JN517x | 2 | #1: RAM address (4 bytes), LSB first #2: No. of bytes to read (2 bytes), LSB first (should not be greater than 128 bytes) | 0x20 | Yes |
| 0x20 | RAM read response | JN514x/ JN5139 | JN5148 JN5142 JN5139 JN516x | 2 | #1: Status (1 byte): 0x00 (success) 0xF7 (Auth error) #2: x bytes read from RAM | N/A | Yes |
| 0x21 | Run request | PC | JN5148 JN5139 JN516x JN517x | 1 | #1: Address to jump to (4 bytes), LSB first | 0x22 | No |
| 0x22 | Run response | JN514x/ JN5139 | JN5148 JN5139 JN516x JN517x | 1 | #1: Status (1 byte): 0x00 (success) 0xF7 (Auth error) | N/A | No |
| 0x25 | Read Flash ID request | PC | JN5148 JN5142 JN5139 JN516x JN517x | 0 | N/A | 0x26 | Yes |
| 0x26 | Read Flash ID response | JN514x/ JN5139 | JN5148 JN5142 JN5139 JN516x JN517x | 3 | #1: Status (1 byte): 0x00 (success) 0xF7 (Auth error) #2: Flash manufacturer ID (1 byte) #3: Flash device ID (1 byte) | N/A | Yes |
| 0x27 | Change Baud Rate Request | PC | JN5148 JN5142 JN5139 JN516x JN517x | 1 | #1: Serial Clock Divisor 1 = 1000000 2 = 500000 9 = 115200 (approx.) 26 = 38400 (approx.) | N/A | Yes |

JN51xx Boot Loader Operation

| Message Type | Meaning | Originator | Supported Devices | # Parms | Message Parameters | Response Message Type | Enabled in CRP1 |
|--------------|----------------------------|-------------------|--|---------|---|-----------------------|-----------------|
| 0x28 | Change Baud Rate Response | JN514x/ JN5139 | JN5148 JN5142 JN5139 JN516x JN517x | 1 | #1: Status (1 byte): 0x00 (success) | 0x27 | Yes |
| 0x2C | Select Flash type | PC | JN5148 JN5142 JN5139 JN516x JN517x | 2 | #1 : Flash type (1 byte) (see table below for mapping) #2: Custom programming jump address (4 bytes), LSB first (set to 0000), | 0x2D | Yes |
| 0x2D | Select Flash type response | JN514x/ JN5139 | JN5148 JN5142 JN5139 JN516x JN517x | 1 | #1 byte: Status | N/A | Yes |
| 0x32 | Get Chip ID Request | PC | JN5148 JN5142 JN516x JN517x | 0 | N/A Note: The Chip ID for the JN5139 and JN5121 can be obtained by a RAM read on address 0x100000FC | N/A | Yes |
| 0x33 | Get Chip ID Response | JN514x | JN5148 JN5142 JN516x JN517x | 2 | #1: Status (1 byte): 0x00 (success) #2: Chip ID (4 bytes) (MSB first): Note: The Chip ID for the JN5139 can be obtained by a RAM read of 4 bytes from address 0x100000FC The Chip IDs for the current range of microcontrollers are: JN5148: 0x10404686 JN5142: 0x00005686 JN5139: 0x10002000 The above IDs are subject to change with further chip revisions. | 0x32 | Yes |

Table 8: Message Types

Mapping Flash IDs to Flash Device Types

The JN517x, JN516x, JN5148, JN5142 and JN5139 devices support multiple external Flash device types. It is the responsibility of the PC application to instruct the boot loader which Flash device to use. To determine the Flash device type connected, issue a Read Flash ID request (0x25). The device will respond with the Manufacturer ID and Device ID shown in Table 9. To select a Flash device type, issue a Select Flash Type request (0x2C), with the appropriate Select Flash Type in Table 6.

| Manufacturer ID | Device ID | Select Flash Type | Type |
|-----------------|-----------|-------------------|------------------------------------|
| 0x05 | 0x05 | 4 | ST M25P05-A |
| 0x10 | 0x10 | 0 | ST M25P10-A |
| 0x11 | 0x11 | 5 | ST M25P20-A |
| 0x12 | 0x12 | 3 | ST M25P40 |
| 0xBF | 0x49 | 1 | SST 25VF010A |
| 0x1F | 0x60 | 2 | Atmel 25F512 |
| 0xCC | 0xEE | 8 | Internal Flash (JN516x/7x only) |

Table 9: Flash IDs

Response Status Code

| Status Type | ID |
|------------------|------|
| OK | 0 |
| Not supported | 0xFF |
| Write fail | 0xFE |
| Invalid response | 0xFD |
| CRC error | 0xFC |
| Assert fail | 0xFB |
| User interrupt | 0xFA |
| Read fail | 0xF9 |
| TST error | 0xF8 |
| Auth error | 0xF7 |
| No response | 0xF6 |

Table 10: Response Status Codes

Binary Version

Binary Version on JN514x and JN5139

When a binary file is built, the ROM version for which it was built is stored in the binary file at offset 0x0C for JN5148 and JN5139, and at offset 0x00 for JN5142. This is configured in the linker configuration file stored in the chip build directory, i.e. one of:

- **Chip\JN5148\Build\AppBuild_JN5148.Id**
- **Chip\JN5142\Build\AppBuild_JN5142.Id**
- **Chip\JN513x\Build\AppBuild_JN5139.Id**

To verify that the binary file is correct for the target device, the boot loader should perform a RAM read of 4 bytes from address 0x00000004. If the two versions do not match then the binary file has been built for the wrong target.

Binary Version on JN516x/7x

The device version is stored in the application binary at offset 0x00. This is configured in the linker configuration file stored in the chip build directory:

Chip\<<Chip>\AppBuild.Id

The version number is 4 bytes long and is configured as shown in the table below.

| Byte | Contents | JN5179 | JN5168 | JN5164 | JN5161 |
|-------|--|--------|--------|--------|--------|
| 0 | Flash size in increments of 32K 0x00 – 32K 0x01 – 64K 0x02 – 96K ... 0x07 - 256K 0x0F – 512K | 0x0F | 0x07 | 0x04 | 0x01 |
| 1 | RAM size in increments of 8K 0x00 – 8K 0x01 – 16K 0x02 – 24K 0x03 – 32K | 0x03 | 0x03 | 0x03 | 0x00 |
| 2 - 3 | Chip Type JN516x = 0x0008 JN5179 = 0x000A | 0x000A | 0x0008 | 0x0008 | 0x0008 |

Table 11: JN516x/7x Version

On the JN516x family the 4 bytes of version information are stripped from the binary when it is programmed into Flash memory (the NXP Flash programming tools do this automatically). For the JN517x family, the version information is retained (see Table 6: JN517x Flash Header).

Example Programming Sequence

The PC application must dynamically determine which Flash device is connected to the wireless microcontroller. This is achieved by issuing the FL_READ_ID_REQ message. The response will give the Manufacturer ID and Device ID of the Flash device attached.

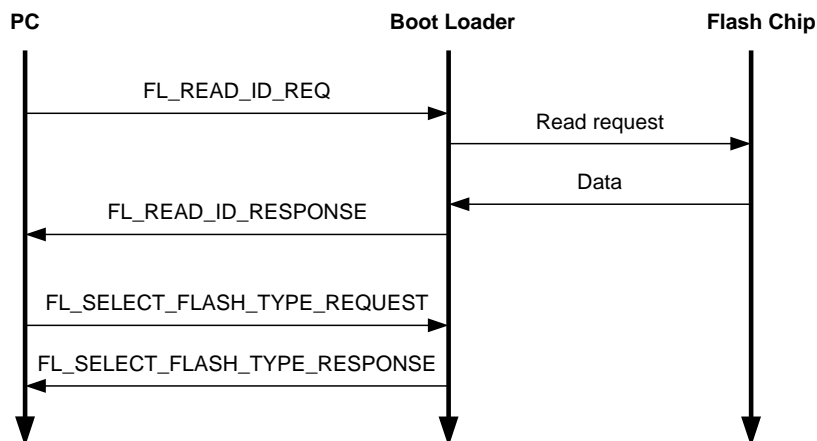


Figure 9: Flash Device Type Selection Sequence

Once the Flash device type is known, the type can be set within the boot loader by sending a FL_SELECT_FLASH_TYPE_REQUEST.

After the Flash type has been selected, the programming sequence is the same for all JN51xx devices, as illustrated below.

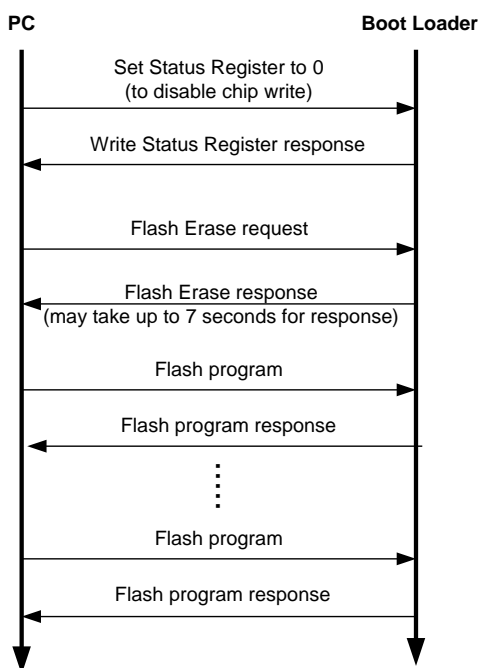


Figure 10: Flash Programming Sequence

Revision History

| Version | Notes |
|---------|---|
| 1.0 | Initial Release |
| 1.1 | Corrected Flowchart |
| 1.2 | Added Boot Loader Timing Parameters |
| 1.3 | Re-templated |
| 1.4 | Corrected Flowchart |
| 1.5 | <ul style="list-style-type: none"> - New format - Corrected JN5121 details - Added information for the JN5139 and JN5148 devices - Merged with JN-AN-1007 "Boot Loader Serial Protocol" |
| 1.6 | Internal version with JN5142 device added |
| 1.7 | Differences between JN5148 variants described and JN5121 device removed |
| 1.8 | Added reference to JN5142-J01 |
| 1.9 | Added information for JN516x series |
| 1.10 | Added more detailed information about in-service application upgrades on JN516x devices |
| 1.11 | Added section detailing contents of JN516x Index Sector and information about safe copying from external to internal Flash memory |
| 1.12 | Added text in various sections to include JN517x family |
| 1.13 | Updated Flash remapping description to include details for the JN5169 and JN5179 devices |

Important Notice

Limited warranty and liability — Information in this document is believed to be accurate and reliable. However, NXP Semiconductors does not give any representations or warranties, expressed or implied, as to the accuracy or completeness of such information and shall have no liability for the consequences of use of such information. NXP Semiconductors takes no responsibility for the content in this document if provided by an information source outside of NXP Semiconductors.

In no event shall NXP Semiconductors be liable for any indirect, incidental, punitive, special or consequential damages (including - without limitation - lost profits, lost savings, business interruption, costs related to the removal or replacement of any products or rework charges) whether or not such damages are based on tort (including negligence), warranty, breach of contract or any other legal theory.

Notwithstanding any damages that customer might incur for any reason whatsoever, NXP Semiconductors' aggregate and cumulative liability towards customer for the products described herein shall be limited in accordance with the *Terms and conditions of commercial sale* of NXP Semiconductors.

Right to make changes — NXP Semiconductors reserves the right to make changes to information published in this document, including without limitation specifications and product descriptions, at any time and without notice. This document supersedes and replaces all information supplied prior to the publication hereof.

Suitability for use — NXP Semiconductors products are not designed, authorized or warranted to be suitable for use in life support, life-critical or safety-critical systems or equipment, nor in applications where failure or malfunction of an NXP Semiconductors product can reasonably be expected to result in personal injury, death or severe property or environmental damage. NXP Semiconductors and its suppliers accept no liability for inclusion and/or use of NXP Semiconductors products in such equipment or applications and therefore such inclusion and/or use is at the customer's own risk.

Applications — Applications that are described herein for any of these products are for illustrative purposes only. NXP Semiconductors makes no representation or warranty that such applications will be suitable for the specified use without further testing or modification.

Customers are responsible for the design and operation of their applications and products using NXP Semiconductors products, and NXP Semiconductors accepts no liability for any assistance with applications or customer product design. It is customer's sole responsibility to determine whether the NXP Semiconductors product is suitable and fit for the customer's applications and products planned, as well as for the planned application and use of customer's third party customer(s). Customers should provide appropriate design and operating safeguards to minimize the risks associated with their applications and products.

NXP Semiconductors does not accept any liability related to any default, damage, costs or problem which is based on any weakness or default in the customer's applications or products, or the application or use by customer's third party customer(s). Customer is responsible for doing all necessary testing for the customer's applications and products using NXP Semiconductors products in order to avoid a default of the applications and the products or of the application or use by customer's third party customer(s). NXP does not accept any liability in this respect.

Export control — This document as well as the item(s) described herein may be subject to export control regulations. Export might require a prior authorization from competent authorities.

All trademarks are the property of their respective owners.

NXP Semiconductors

For the contact details of your local NXP office or distributor, refer to:

www.nxp.com