Application Note

FAQs: NOR Flash

This application note addresses some frequently asked questions about Renesas NOR flash devices.

Contents

1. Should decoupling capacitors be used with the flash drive, and, if so, how?

To minimize voltage fluctuations and provide sufficient localized energy to the chip, place one or more decoupling capacitors between the V_{CC} and GND pins as close as possible to the flash. The recommended value of the decoupling capacitors is 1 µF, with an optional 100 nF capacitor in parallel

Certain high-performance devices might require a larger decoupling capacitor; for example: 4.7 µF with an optional 100 nF capacitor in parallel.

To improve performance, use low ESR ceramic capacitors.

For more information, see [1].

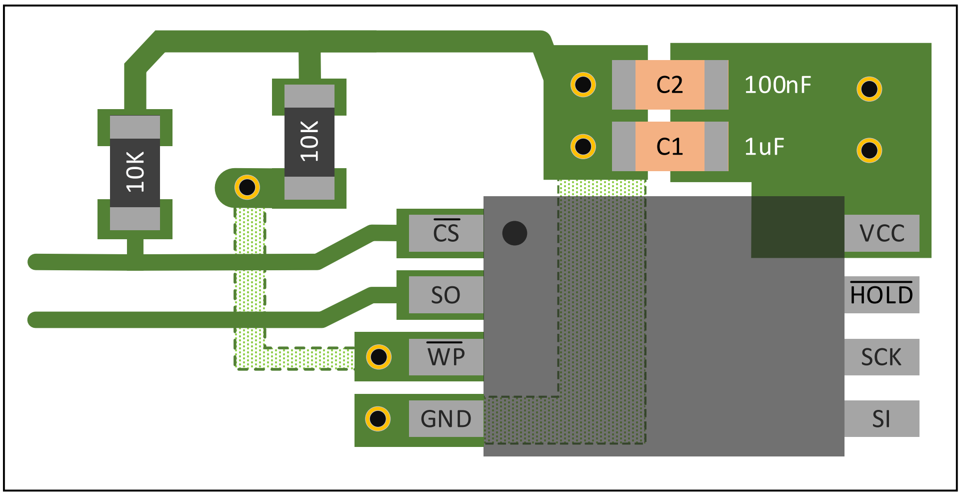
2. Which of the flash device pins should have pull-up resistors?

- $\overline{CS}$ (Chip Select): This is a low-speed, active-low signal that is asserted by the host when sending a command to the flash. A pull-up resistor on $\overline{CS}$ ensures that the voltage on $\overline{CS}$ tracks V_{CC} during power-up. Also, this way the device wakes up in inactive state ($\overline{CS}$ high).
- $\overline{WP}/IO_2$: this pin can be configured either as $\overline{WP}$, a low-speed write-protection signal, or as IO_2 a high speed I/O signal used when quad-SPI communications is enabled.
- $\overline{HOLD}$ (IO_3): this pin can be configured either as $\overline{HOLD}$ a low-speed signal used to pause communication, or as IO_3 a high speed I/O signal used when quad-SPI communications is enabled

Typically, $\overline{\text{WP}}$ and $\overline{\text{HOLD}}$ signals have internal pull-ups, but for certain devices the datasheet may recommend also using external pull-ups on these signals.

An additional advantage of adding pull-up resistors is that it makes access to the signals easier during debugging. The pads are physically present on the PCB, and the pull-ups can be installed or left unpopulated, depending on the situation.

For dual-purpose signals ($\overline{\text{WP}}$/IO$_2$ and $\overline{\text{HOLD}}$/IO$_3$), when an external pull-up resistor is used, place the pad of the pull-up resistor connected to the high-speed I/O signal close to the signal trace. This minimizes stub length, resulting in reduced signal reflections.



Example of placing and routing decoupling capacitors and pull-up resistors (minimizing stub length)

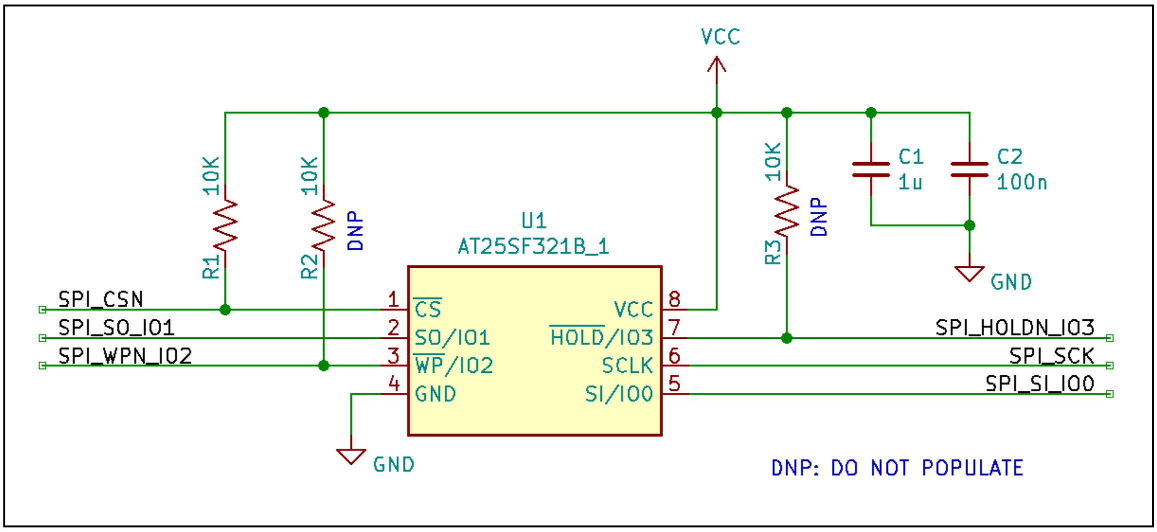
For more information, see [1].

3. What are the most important signal routing tips?

Place the device as close as possible to the MCU. Also, ensure the high-speed signal traces are as short as possible. Ideally, all high-speed signals (possibly all signals) are referenced to an adjacent solid ground plane, with no gaps in the ground plane.

Equalize trance lengths as much as possible for the high-speed I/O signals, especially at high frequencies (100 MHz and above).

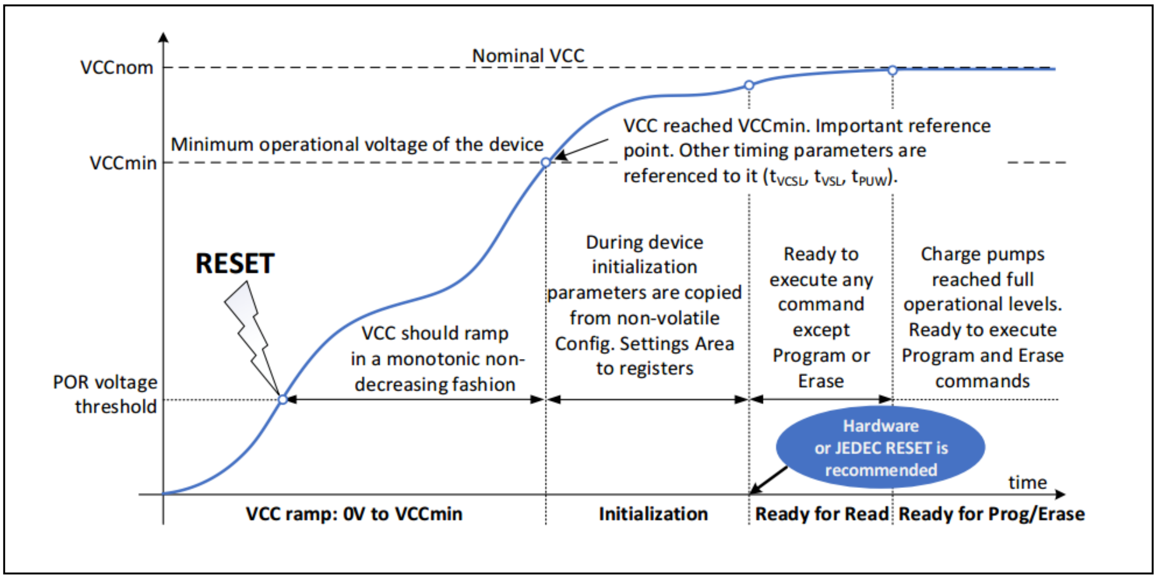
4. Can you provide schematic examples?



Schematic for SPI Flash (8-pin SOIC)

5. Power-up/-down tips

To ensure that the device is properly initialized during power-up, the power supply must rise monotonically, non-decreasingly. The V_{CC} waveform must not have a dip or a prolonged steady value during the rising edge. It must reach the minimum operational voltage for the device within the period specified in the datasheet.



V_{CC} Ramp

For more information, see [2].

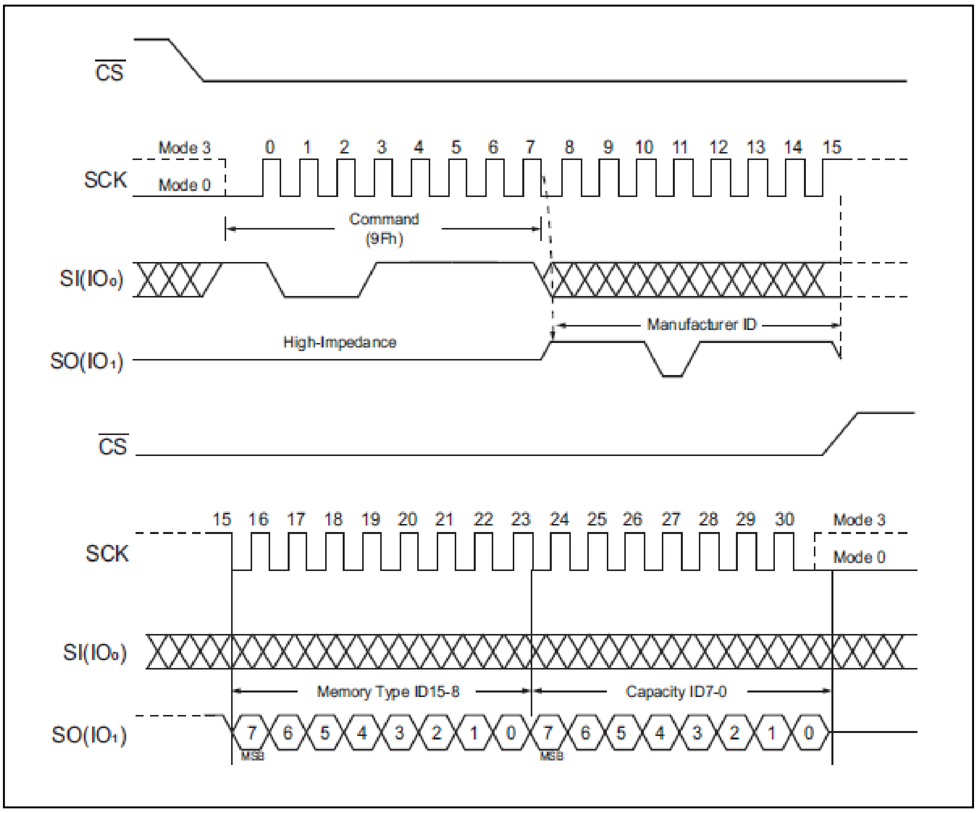
6. During system bring-up what are the basic checklist items associated with the flash device?

First, verify that the flash device is properly installed, with correct orientation, and that the component values match the schematic.

Power up the system with a DMM, or better, verify voltages with an oscilloscope. Ensure that the flash power supply and signal voltages are within the supported voltage range of the product.

Using a low-frequency SPI clock, start sending basic commands from the host. Using an oscilloscope or a logic analyzer, monitor the $\overline{CS}$, SCK, SI, and SO signals during communication, and ensure they comply with the datasheet. Start with the “read manufacturer/device ID” command (opcode 9Fh, see Figure below). Ensure a correct response is received from the flash device. If so, this means the SPI communication is working in both directions.

Increase the SPI clock speed up to the frequency you intend to use in the system, and re-test. For high-speed signals such as clock and I/O signals, verify signal integrity.



Opcode 9Fh, Read Manufacturer/Device ID

7. How do I get a software driver for my flash device?

It is difficult to provide a generic software driver since a driver implementation depends on the host MCU/MPU and, specifically, the architecture of the flash controller within the host. Every host controller has a different architecture, and the driver heavily depends on that.

Renesas can provide example drivers upon demand for specific hosts.

If you already have a flash driver for your current system and want to switch to a Renesas flash product, Renesas can review your driver and describe the modifications to apply (usually minor changes). Alternatively, Renesas can explain the differences between your currently used flash product and the Renesas flash product to help you modify the driver on your own.

8. What is a correct erase/program sequence?

1. Send a write-enable command to the flash (opcode 06h).
2. Send a block/chip erase or a page-program command.
3. Read status register 1, and check bit 0 (READY/BUSY bit). Do not resume normal operation until this bit is clear.

Note that bit 7 in DataFlash products is the READY/BUSY bit and its polarity is opposite that of other product families

Erase and program operations are relatively long. After sending the erase/program command, the erase/program operation is carried out in the background. During that period, most flash products block most other operations (such as read). Reading the status register for checking the READY/BUSY bit is allowed. Many products allow suspending an erase/program operation in case the system must read from the memory array. The erase/program operation can later be resumed.

For more information, see [3]

9. What tools are available to program an image into the device (in-system or stand-alone)?

During development, a common way is to build a flash loader plug-in, which is invoked by the tool chain or IDE when user requests to download the image into flash memory. We provide flash loader code examples in `https://github.com/dialog-semiconductor/Flash-Universal-Flashloader`. There are also debug/programming probes such as J-Link from Segger (`https://www.segger.com/products/debug-probes/j-link/`).

For other programming tools, including industrial, see `https://www.dialog-semiconductor.com/products/memory/programmer-support`.

10. How to switch from single SPI to quad-SPI?

Set the quad-enable (QE) bit (status register 2 bit 1). This changes the functions of the $\overline{\text{WP}}$ (write-protect) and $\overline{\text{HOLD}}$ pins, and they become I/O pins IO_2 and IO_3, respectively. Pins SI and SO keep their I/O functionality but are renamed IO_0 and IO_1 in the context of quad-SPI. At this point, it is possible to send commands with quad elements. For example, read commands with 1-4-4 format (opcode on a single line, address on four parallel lines, and data on four parallel lines).

11. What are dummy cycles? How are they used?

Dummy cycles are wait cycles that are required for some read commands because of to latency. This means that after sending the opcode and address parts of the read command, the host must wait a few cycles until data is returned from the flash. For each read command, the datasheet specifies if and how many dummy cycles are required. Some read commands have a configurable number of dummy cycles. For these commands, the host can use more dummy cycles with higher SPI clock frequencies and fewer dummy cycles with lower SPI clock frequencies.

12. How can I protect a flash device?

There are multiple methods for protecting on-chip resources from accidental or intentional modifications.

Any command that intends to modify the memory array or one of the status registers must be preceded by a Write-Enable command. This method is designed to prevent accidental writes.

The memory array can be protected by one of two methods, depending on the specific product. These methods enable protection of parts, or of the memory array, or the entire memory array. The protection configuration is done by writing to status registers or by using dedicated protection commands.

The status registers can be protected by a hardware mechanism: the write-protect ($\overline{\text{WP}}$) pin. When the status registers are protected, the flash configuration is protected from modification. This includes the protection status of the memory array.

For more information, see [4]

Notes

[1] AN105-A1: PCB Design and Layout Considerations

[2] AN501: Power-Up/Down Considerations in NOR Flash

[3] AN500A: NOR Flash Memory Erase Operation

[4] AN115: Flash Protection Schemes and Methods in Renesas NOR Flash Products

13. Revision History

Revision	Date	Description
A0	Jun 14, 2022	Initial release.