AN13695

Dynamic Voltage Scaling using PVT Sensor on i.MX RT500

Rev. 0 — 30 August 2022 Application note

Document information

Information	Content
Keywords	Dynamic Voltage Scaling, PVT Sensor, i.MX RT500
Abstract	This document describes how the PVT Sensor works, and how to use it in an application to extend battery life. This application note is part of NXP's application software pack.

1 Introduction

i.MX RT500 is an excellent highly integrated MCU that is ideal for many low-power applications like wearables. This MCU has many features to reduce and optimize power consumption in order to extend battery life. The RT500 also supports Dynamic Voltage Scaling (DVS) to optimize the active power with the operating frequencies. This document covers an integrated feature in the RT500, when used properly, can potentially reduce power further and lower the voltage below minimum specifications in the data sheet. This feature monitors the Process, Voltage, and Temperature (PVT) variation of each unique device at runtime. Using this PVT Sensor can reduce power by up to 30 % on the main supply rail for the digital logic, called VDDCORE. The PVT Sensor is designed to help reduce voltage and power in active modes, including the RT500 Sleep mode. The PVT Sensor does not improve power consumption in the static low-power modes and is not used in those modes.

This document describes how the PVT Sensor works, and how to use it in an application to extend battery life. This application note is part of NXP's application software pack, which includes these additional references for the PVT Sensor:

- Example project using PVT Sensor - dynamically adjusts the VDDCORE voltage at runtime, within an application using FreeRTOS. The driver for the PVT Sensor is provided as a library to easily integrate into any application for the i.MX RT500.
- Getting Started Lab Guide – the document for the example project that details: using the example, the software in the example, and the steps to integrate the PVT Sensor library in an application
- Getting Started Video – demonstrates the example and steps in the Lab Guide

1.1 Glossary

- ADC - Analog to Digital Converter
- API – Application Programming Interface
- DVS - Dynamic Voltage Scaling
- FRO – Free-Running Oscillator
- FRO_DIV4 – The Divide-by-4 output of the FRO
- LPOSC – Low-Power Oscillator
- LVD – Low-Voltage Detect
- MCU – Micro-Controller Unit
- OTP – One-Time Programmable
- POR – Power-On Reset
- PVT – Process, Voltage, Temperature
- SDK – Software Development Kit

2 Process, Voltage, and Temperature (PVT) variation

The power consumption and the performance of a semiconductor can change during its operation when the temperature or voltage varies. Also, there are differences in comparing one specific device to another due to the variation in the manufacturing process. To optimize power consumption and performance of a device like i.MX RT500, it is important to understand how the silicon is affected by these variations.

2.1 Silicon process variation

Modern semiconductors are of such small geometries today that subtle differences in the manufacturing process can lead to noticeable differences in the key properties of each device. Even on the same wafer, each die can have different properties. One property that varies due to the process is the speed of the transistors. In this document, the variation in the speed of each device's transistors is one-dimensional. The speed of both NFETs and PFETs on a specific device can be at a Slow-Slow (SS) limit, at the Fast-Fast (FF) limit, or somewhere in between. When we plot the distribution of this speed for every device manufactured, we have a Gaussian distribution, see Figure 1. The median of these speeds is Typical-Typical (TT). The testing procedure has limits for what speeds are acceptable for production. Outliers that are slower than the SS limit or faster than the FF limit will be discarded.

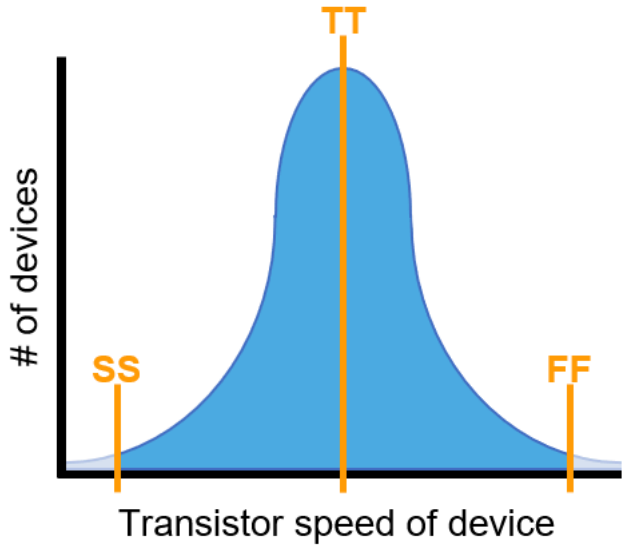

Figure 1. Gaussian distribution of transistor speeds of individual devices

There are tradeoffs comparing devices with different speeds depending on the process variation: SS devices have lower static leakage current, but FF devices have faster performance when active. At the same clock frequency, FF devices can be powered with a lower voltage than SS devices and still achieve the same performance. Since the SS transistors are slower, they must be turned on harder using higher supply voltage to keep up with the FF devices.

2.2 Temperature variation

Silicon performance and power consumption are also affected by the temperature. i.MX RT500 has an operation range of ambient temperature from -20 to 70C, with typical temperature of 25C. Comparing the performance of a specific device, if the temperature is reduced to -20C, the transistors are slower compared to 25C. That means that to achieve the same clocking frequency at 25C, as a device cools to -20C, the supply voltage must be increased to drive those transistors harder. And as that device warms back up to 25C, the voltage can be reduced while achieving the same clock frequency.

2.3 Minimum voltage required

To minimize power consumption in an application that must be clocked at a specific frequency, the supply voltage would be set at the minimum voltage required for that frequency. Increasing the voltage above that minimum wastes battery life.

The data sheet for the device includes specifications for the minimum voltage. However, these limits are set to guarantee operation of all devices with a wide range of PVT variations. Consider this example from the i.MX RT500 data sheet. Table 1 comes from the "General Operating Conditions" table in the data sheet, refer to the data sheet for the latest specifications. This document focuses on clocking from the FRO at 192 MHz. The table below specifies that when the main_clk is 192 MHz, the supply rail VDDCORE requires a minimum 0.9 V to operate.

Table 1. "General Operating Conditions" VDDCORE specifications from RT500 data sheet

Name	Conditions	Minimum (V)
VDDCORE	Retention mode	0.58
	Active Mode (M33/DSP/GPU Max Frequency = 60 MHz, FBB)	0.7
	Active Mode (M33/DSP/GPU Max Frequency = 100 MHz, FBB)	0.8
	Active Mode (M33/DSP/GPU Max Frequency = 192 MHz, FBB)	0.9
	Active Mode (M33/DSP/GPU Max Frequency = 230 MHz, FBB)	1.0
	Active Mode (M33/DSP/GPU Max Frequency = 275 MHz, FBB)	1.1

The 0.9 V requirement is based on worst-case conditions: an SS device running at a temperature of -20C. Since most devices manufactured are faster than SS, most devices can clock at 192 MHz with a voltage less than 0.9 V. Since most devices do not spend much time operating at -20C and are closer to room temperature, that enables operating at some voltage less than 0.9 V. It is rare that a device requires 0.9 V to operate at 192 MHz.

To be able to operate at the ideal minimum voltage for a given frequency, each unique device must determine that voltage at runtime, while accounting for temperature changes. The integrated PVT Sensor is the solution to this challenge.

3 Operating with PVT Sensor

The PVT Sensor is a peripheral in the i.MX RT500 that monitors the timing margin of the `main_clk` in the VDDCORE power domain at runtime. It triggers an interrupt when this timing margin is not met, which indicates VDDCORE must be increased to meet the required timing. Figure 2 shows a closed loop system using an external PMIC to adjust the voltage to VDDCORE. When conditions change and the PVT Sensor indicates that VDDCORE must be increased, the application requests the PMIC to increase this voltage through I2C.

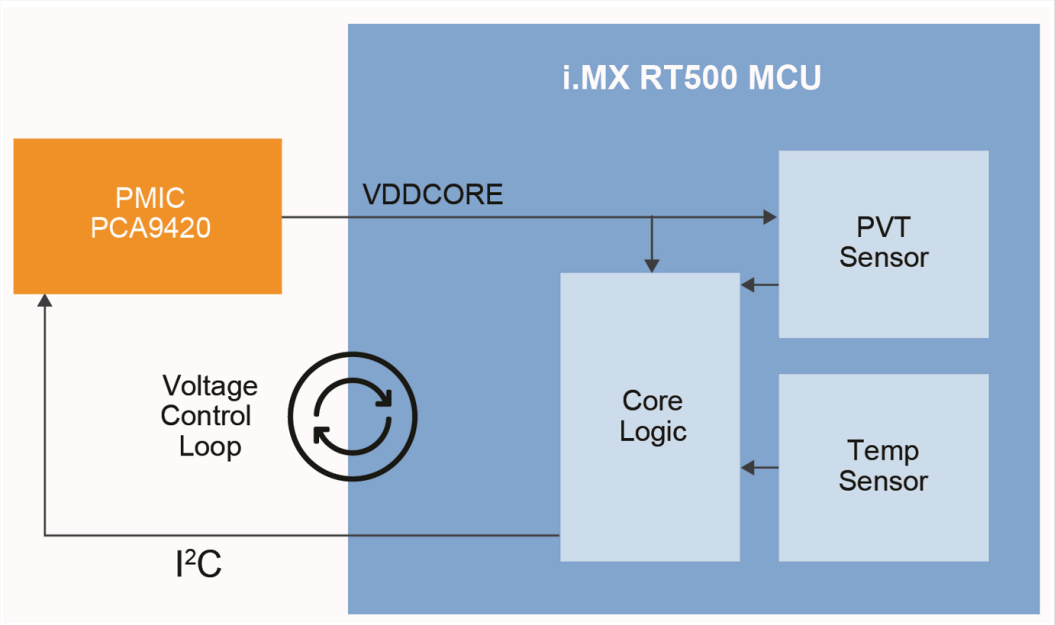

Figure 2. Closed-loop voltage control with PTV sensor

Using this control loop with the PVT Sensor enables the application to operate with VDDCORE voltage below the 0.9 V minimum limit in the data sheet. The PVT Sensor helps find the ideal minimum voltage each device can operate at, and adjusts the voltage at runtime if the temperature changes.

3.1 Monitors timing margin

The PVT Sensor monitors the timing margin of `main_clk` using a delay line, see Figure 3. `Main_clk` is divided by 2 and generates the Launch CLK. The Launch CLK is compared to a delayed version of itself. This delay is programmable and must be set to a value that correlates to the speed of that unique device (as in SS/TT/FF). The timing comparison is done through a flip-flop that uses Launch CLK to clock the delayed version of itself. If the added delay is too long, the flip-flop output is high, indicating a timing issue and triggering an alarm.

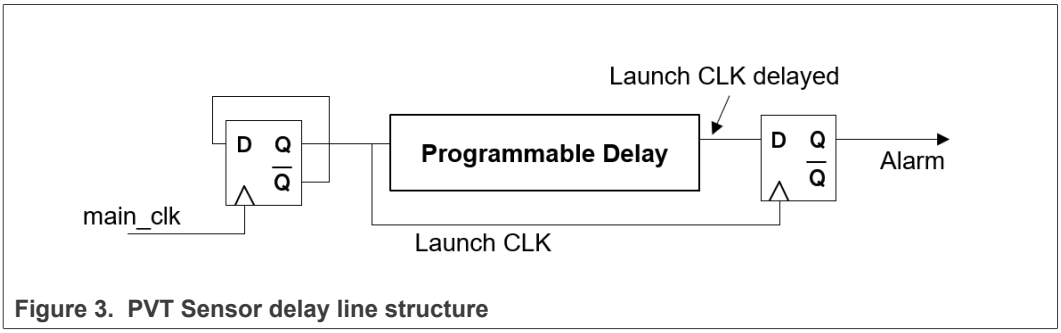

Figure 3. PVT Sensor delay line structure

For the PVT Sensor to properly monitor the `main_clk` timing, the delay line must be programmed with a delay that matches the worst-case timing path of the logic clocked by `main_clk`. See section 4 Ideal PVT Delay Setting. Once programmed with this delay, the PVT Sensor emulates this worst-case timing path in the VDDCORE logic. This delay tracks the actual timing required by this unique device across temperature and voltage variations. As the voltage decreases or as the temperature gets colder, the

delayed Launch CLK has a longer delay. When the alarm signal triggers, the application increases the VDDCORE voltage, which increases the performance of the logic, which decreases the delay on Launch CLK.

Figure 4 shows the timing of the PVT Delay Line. The status of delayed Launch CLK is sampled on every the rising edge of Launch CLK. When operating above the minimum voltage needed for that device and current temperature, the delay is short and the delayed Launch CLK is low at rising edge of Launch CLK, so the alarm output remains low. However, when the VDDCORE voltage is too low for current conditions, the delay is long enough that the delayed Launch CLK is still high at the rising edge of Launch CLK. The alarm output is high, triggering an interrupt to indicate that higher VDDCORE voltage is needed to reduce the timing delay.

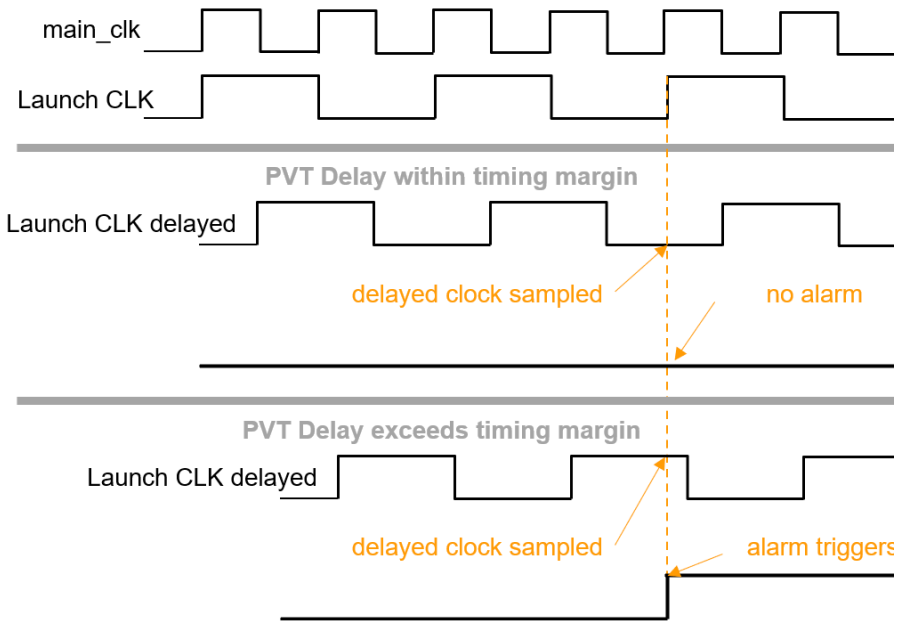

Figure 4. PVT Delay line timing with alarm trigger

3.2 Reducing supply voltage at runtime

When using the PVT Sensor and reducing VDDCORE, the PVT alarm indicates when higher voltage is required. But the PVT Sensor does not give any indication of when VDDCORE can be further reduced. Instead, the application tests a lower voltage periodically. If that reduced voltage is too low, the PVT immediately triggers the alarm, and the application increases the voltage back to the level it was. But if no alarm is triggered, VDDCORE can remain at that reduced voltage.

Figure 5 shows an example of VDDCORE dynamically changing as temperature changes. This graph starts where VDDCORE in green has already been reduced to 850 mV by using the PVT Sensor and is operating at 25C. The PMIC PCA9420 is used, which has voltage steps of 25 mV. While the temperature drops, the PVT Delay Line gets slower, and at some point the PVT alarm triggers. The application requests PMIC to increase 1 step to 875 mV. The temperature continues to drop, so another PVT alarm triggers, and VDDCORE is increased to 900 mV. The application is continually testing if a reduced voltage is acceptable. Typically, reducing the voltage 1 step immediately triggers the PVT interrupt, and the voltage is increased 1 step back to the previous level.

As the temperature warms, at some point the reduced voltage test does not trigger an alarm, and the PVT Sensor allows VDDCORE to reduce 1 step to 875 mV. As the temperature continues to warm, this example eventually drops down to 800 mV. And as the temperature then cools back to 25C, the PVT Sensor triggers its alarm when voltage must be increased, the VDDCORE returns to 850 mV.

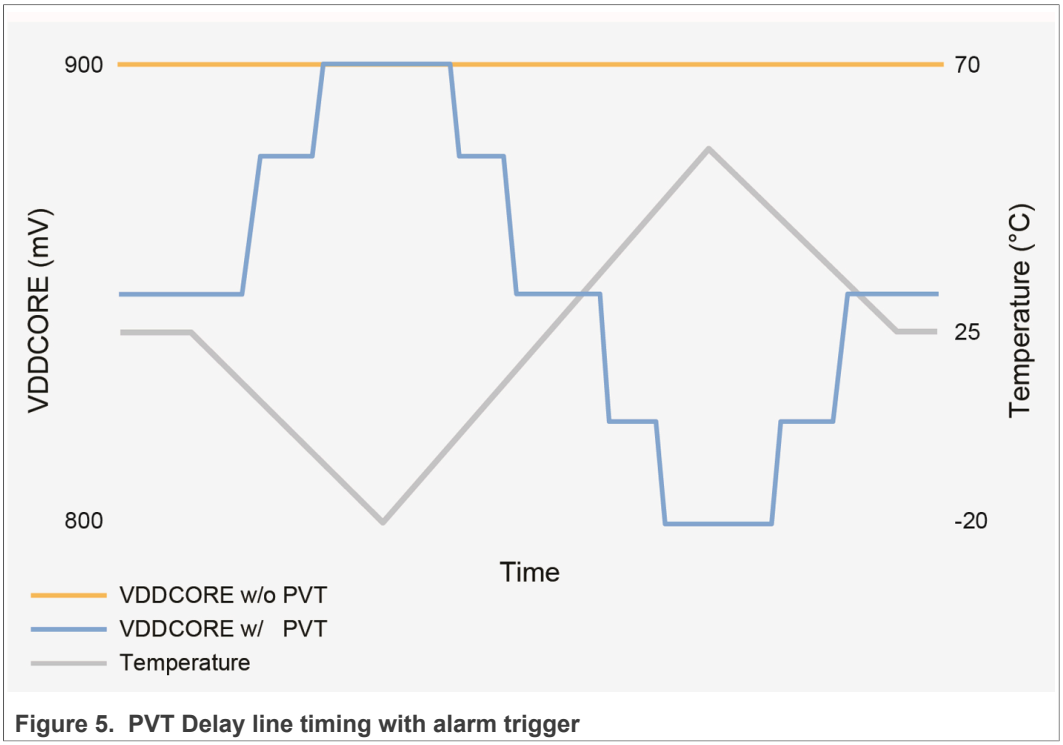

Figure 5. PVT Delay line timing with alarm trigger

When the application reduces VDDCORE using the PVT Sensor, it can test for a lower voltage using a loop. This loop reduces the voltage 1 step, waits for a time delay, and then repeats. See Figure 6 for a flowchart. When the PVT Sensor detects the voltage is too low, it interrupts the application, which raises the voltage 1 step.

Section 5.2 discusses minimum voltage requirements for the PVT Sensor. If the PVT Sensor has already reduced the voltage to the minimum allowed by the application, then the application can stop testing for a lower VDDCORE, until after VDDCORE is increased again.

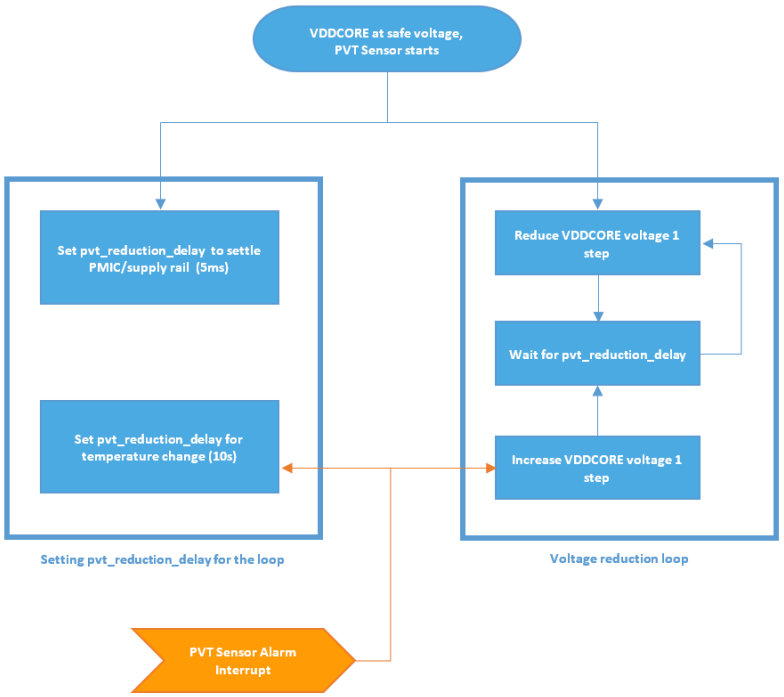

Figure 6. Voltage reduction flowchart with PVT Sensor

The application can use two different time delays for this voltage reduction loop. After initializing the PVT Sensor, the goal is to quickly find the minimum acceptable voltage in the current conditions. This time delay can be short, and must give the PMIC and supply voltage rail enough time to settle at the new voltage. NXP's testing with the PCA9420 PMIC found once the PMIC changes the voltage 1 step, the supply rail settles very quickly. But the application needed more time to control the PMIC using I2C and send the new register settings. The time measured for this PMIC communication was about 2 ms from calling the API to update the PMIC, to when the PMIC changed VDDCORE voltage. Therefore, with some added headroom, the example in this application software pack uses a 5 ms delay for the PMIC during the initial voltage reduction.

For applications using the PVT Sensor, the initial voltage reduction ends when the PVT Sensor alarm triggers. The application then increases the voltage 1 step, and has now found the minimum acceptable voltage in the present conditions. The next stage for the application is to check if the temperature has changed enough that a lower voltage is now acceptable. In this stage, a longer delay is used since temperature changes much slower compared to the PMIC settle time. Based on analyzing the thermal mass of the system, and testing applications with different delay lengths, NXP found that a 10 s delay is a good option for most applications. This application example uses a 10 s delay. Other applications can change this delay to optimize. Shorter delays mean that the app spends more time adjusting the PMIC and testing the voltage; the downside of longer delays is that the app may execute at a higher power consumption longer than necessary.

4 Ideal PVT Delay setting

As discussed earlier in Section 3.1, the key to tuning the PVT sensor is to program the delay line with a value that correlates with the Process variation (speed) of each unique device. NXP has tested and characterized many devices that range from the SS limit to FF. This testing was done to find the ideal PVT Delay value to use on each device.

NXP tested with firmware written to exercise many of the critical timing paths in the SoC and tested at various temperatures to find the minimum VDDCORE voltage required under all of these test conditions. This characterization was done using FRO trimmed at 192 MHz, sourcing `main_clk` at 192 MHz, no PLLs, or other clocks turned on at higher frequencies. Then, for all these devices and conditions, PVT delay values were found to correlate with the minimum voltage of each device. This feature has found reliable PVT delay values that optimize the VDDCORE voltage but have enough margin to avoid faults at runtime.. This margin allows the application to use the PVT alarm as an early indication that the voltage is getting too low before a failure occurs. The application then has time to increase the voltage before conditions change further, but the application must do so immediately using the PVT Sensor interrupt.

To better enable using the PVT Sensor, NXP has started programming the ideal PVT delay value in the OTP fuses. NXP recommends using the value stored in these fuses, as they were tested and confirmed with each device during the manufacturing test. However, older devices already manufactured do not have these OTP fuses. For those devices, another option is using the integrated Ring Oscillator.

4.1 PVT Delay stored in OTP fuses

Starting with a specific manufacturing date code, NXP programs each i.MX RT500 device during manufacturing with the ideal PVT Delay value to use. NXP recommends reading this value using the PVT software library and API `PVT_ReadDelayFromOTP()`. Then program the PVT Sensor with this delay value using `PVT_SetDelay()`.

This PVT Delay value is stored at OTP fuse word 0x1D1. Valid ranges of this value are 1 to 63. `PVT_ReadDelayFromOTP()` reports if this value in OTP is not in the valid range. The application can use this OTP location to confirm if the device has a PVT Delay value programmed, or if it is an older device with this location unprogrammed. If `PVT_ReadDelayFromOTP()` returns 1, or if the application reads this OTP location and finds the value is 0 or greater than 63, the application should use the Ring Oscillator to calibrate the PVT delay, as described in the next section.

Also be aware of the requirements before reading the fuses in OTP. The data sheet specifies a minimum VDDCORE voltage for reading the OTP, and the application may need to increase VDDCORE first. Refer to the data sheet for the latest requirements. The OTP driver must be initialized before reading the fuses. `PVT_ReadDelayFromOTP()` will do it for the application if the `otp_initialized` argument is false. The MCUXpresso SDK includes an OTP driver example that demonstrates initializing the OTP and reading the fuses.

4.2 Calibrating PVT Delay using Ring Oscillator

For older devices that do not store the PVT Delay value in OTP, the application can use the integrated Ring Oscillator to calibrate the PVT Delay. This oscillator clocks a counter at a frequency that relates to the Process variation of the unique device. SS devices have a slower frequency, and FF devices have higher. The PVT software library includes some APIs which measure the frequency of this Ring Oscillator, and return the PVT Delay to use.

The Ring Oscillator method is not as accurate as using the value programmed in OTP fuses. But NXP provides this option as an alternative for older devices. The application only needs to measure the Ring Oscillator once during the lifetime of the product. This step is typically done during manufacturing of the end product, and the application can store the PVT delay value in non-volatile memory. Once the PVT Delay value is stored, it

is similar to the OTP fuse method, and the application can read this value when using the PVT Sensor. The Ring Oscillator does not need to be used again.

The frequency of the Ring Oscillator is sensitive to temperature and VDDCORE voltage. Measuring its frequency in the wrong conditions can lead to a PVT Delay value that could cause issues or not be optimal. Here are some important requirements when measuring the Ring Oscillator:

- Set VDDCORE at 0.9 V and allow the power supply rail to settle before calling the Ring Oscillator APIs.
- Only call the Ring Oscillator APIs when the ambient temperature is room temperature at 25C. Remember, this step is done at manufacturing in a controlled environment.

Here is the sequence for calling the Ring Oscillator APIs from the PVT Sensor library:

1. `PVT_EnableRingOsc();`
2. Wait 500 ms
3. `PVT_ReadDelayFromRingOsc(&delay, 500);`
4. `PVT_DisableRingOsc();`

The timing of starting the counter with `PVT_EnableRingOsc()` and stopping with `PVT_ReadDelayFromRingOsc()` is important for properly measuring the Ring Oscillator frequency. The example application uses a 500 ms delay between these APIs. The amount of time the Ring Oscillator was enabled is passed as an argument in `PVT_ReadDelayFromRingOsc()`, and must match the actual time delay. Therefore, interrupts or other tasks should not be enabled or interfere with this timing while the Ring Oscillator is enabled.

4.3 Optimizing PVT Delay with temperature

The programmable delay in the PVT Sensor does not perfectly correlate in temperature with the ideal VDDCORE. The PVT Delay value retrieved from either method is a safe value that works for the whole temperature range, including the worst-case conditions at -20C. The PVT Sensor works well using this value. But the power can be further optimized at warmer temperatures. When the temperature is 25 °C or higher, the PVT Delay can be reduced by 2. It has the potential to reduce the VDDCORE voltage further.

Updating the PVT Delay at runtime requires the application to monitor the temperature. To do it, use the built-in temperature sensor with the ADC. The application can then use this sequence below to update the PVT delay value based on temperature.

1. `PVT_Stop()`
2. `PVT_SetDelay(delay)`
3. `PVT_ClearAlertCount()`
4. `PVT_Start()`

Some applications may find monitoring the temperature and changing the PVT Delay is not worth doing. Those applications can use a fixed PVT Delay, like the example with the associated software pack.

5 Using the PVT Sensor in application

Here are additional guidance and considerations for the application leveraging the PVT Sensor.

5.1 Integrating PVT Sensor library in application

To ease the integration with an application, the PVT-related code is provided as a library. This application software pack has another Getting Started document that covers the example application, and the steps to include the PVT library with the application project. The APIs for the library are documented in the pvt.h source file.

5.2 Requirements for using the PVT Sensor

Proper usage of the PVT Sensor allows an application to operate the i.MX RT500 at a VDDCORE voltage below the data sheet minimum specification. It can lead to issues or faults in the application if the following requirements are not met.

The minimum VDDCORE required is not always limited by the frequency of the `main_clk`. The data sheet also has restrictions on VDDCORE for specific peripherals, features, or use-cases of the MCU. The PVT Sensor was designed to monitor the timing margin of the timing paths of the logic in the VDDCORE domain. But it was not designed to monitor VDDCORE for these other data sheet restrictions. The list below includes some additional limitations to consider, and provides some details from the data sheet. However, always refer to the data sheet for the full details, latest specifications, and other requirements not listed here. If there are multiple VDDCORE minimum requirements in the data sheet that apply, the highest minimum must be used, regardless of what the PVT Sensor reports:

- FRO has minimum VDDCORE requirements based on the trim frequency and dividers used
- FlexSPI and SDHC have requirements based on the clock frequencies to these peripherals or their external memories
- OTP fuses, reading OTP fuses requires a high VDDCORE voltage. It is important when reading the PVT Delay value stored in OTP
- MIPI_DSI_VDD11 is a power supply pin that requires at least 0.85 V when using the MIPI DSI for a display. Technically, this pin is a separate power supply than VDDCORE. But board designs frequently short MIPI_DSI_VDD11 to VDDCORE, including the RT500 EVK.

For the FRO requirements on VDDCORE, the example application sets lower limits used with the PVT Sensor to comply with the data sheet specifications. This example clocks `main_clk` from the FRO. The macro `MIN_VOLTAGE` is set to 0.8 V when the FRO is trimmed at 192 MHz.

Clock frequencies are very important to consider when using the PVT Sensor. The characterization and testing (discussed in Section 4 Ideal PVT Delay Setting) were performed with FRO trimmed at 192 MHz, and clocking `main_clk` at 192 MHz. Other frequencies for `main_clk` are currently not supported with the PVT Sensor. For clocks other than `main_clk`, higher frequencies are not allowed. It includes the PLLs, which tend to generate some internal clock frequencies well above 192 MHz. Even when dividers are used on the PLL outputs, the internal frequencies require higher VDDCORE voltages to operate. Therefore, the PLLs should be disabled when using the PVT Sensor.

The application can also place maximum VDDCORE voltage limits used with the PVT Sensor. Consider a use-case like this example application: `main_clk` is 192 MHz from the FRO. The application is not using other features that require higher VDDCORE voltage, the data sheet's "General Operating conditions" minimum specification of 0.9 V applies for 192 MHz. It is the starting voltage for the app before it enables the PVT Sensor. Once enabled, the app tries to reduce the voltage. While using the PVT Sensor,

if the voltage increases to 0.9 V and the PVT triggers another alarm, the voltage does not need to be increased. The app can keep VDDCORE at 0.9 V because the data sheet confirms this voltage works for all devices and across the full temperature range.

The VDDCORE voltage requirements in the data sheet and this document are for the voltage at the VDDCORE pins of the MCU. If setting the PMIC to these same minimum VDDCORE limits, there is potential for the voltage at the VDDCORE pins to be under the requirement, which could lead to issues. For example, if the minimum requirement for VDDCORE is 800 mV, and the PMIC is set to supply 800 mV, the voltage at the VDDCORE pins could be less than 800 mV. When maximizing battery life and optimizing VDDCORE with the PVT Sensor, the application is accepting more risk by reducing the margin in the system. This can lead to the voltage margin for VDDCORE operation down in the millivolts level. Therefore it is important to consider the power supply system and PCB design when driving VDDCORE. The PMIC has a voltage accuracy tolerance for the output. And likely the PMIC output voltage will vary with temperature. Typically a switching regulator is driving VDDCORE, so that also introduces ripple on the supply rail. And VDDCORE voltage could also be lowered in the system before the VDDCORE pins, like IR drop or due to transients. These impacts and others to the VDDCORE voltage should be considered during the hardware design and for the PMIC voltage setting at runtime, to ensure that the VDDCORE voltage requirements are met.

There are also requirements for interrupts in an application using the PVT Sensor. When a higher VDDCORE voltage is required, the PVT Sensor triggers an alarm interrupt. But it is the application that must request the higher voltage from the PMIC. Therefore, best practices should be used to manage interrupts. The PVT Sensor IRQ should be serviced quickly without much latency, and it should not be blocked for long. The example application is using an RTOS, and the PVT Sensor ISR only notifies the `pvt_task`, which is what sends the request to the PMIC. If tasks or other application codes are used to respond to the PVT Sensor interrupt, then that code has the same requirements as the ISR, and must be allowed to execute quickly, to raise the VDDCORE voltage.

One final requirement already discussed is the settling time for the VDDCORE power rail. Any time the application is using the PVT Sensor and reduces the VDDCORE voltage, it must wait long enough for the PMIC to change the voltage, and the supply rail to settle to the new voltage. The exact delay can depend on the board design, like the PMIC/ regulator driving VDDCORE and the capacitance on this supply rail. If the app drops the voltage too quickly before it has settled, the PVT Sensor may not be able to detect a timing issue before the voltage is dropped again, which can lead to a fault occurring.

5.3 PVT Sensor considerations with low-power modes

Using the low-power modes with the PVT Sensor can add further complications to the application. For different power mode options in the RT500, Sleep mode does not really have an impact when using the PVT Sensor. In Sleep mode, the clock to the core is gated, but `main_clk` remains clocked, and the PVT Sensor can continue to operate in Sleep mode, and can wake the MCU with its interrupt. Also, waking from (Full) Deep Powerdown Mode wakes through the reset process, which likely means the application needs to start at a safe VDDCORE voltage and restart the PVT Sensor.

The power mode considered here with the PVT Sensor is Deep Sleep mode. Consider a use-case similar to this example application, clocking `main_clk` at 192 MHz. The MCU is in active mode using the PVT Sensor. VDDCORE originally started at 0.9 V, and started dropping with the PVT Sensor. The temperature is high, and this enables VDDCORE to drop to 0.8 V. Now the application enters the power mode Deep Sleep, which also disables clocks and powers-down the FRO. The logic is in a static state, so

the PVT Sensor is no longer active. The MCU is in this state for some time, and during this state the temperature cools. The worst-case example is when the temperature drops to minimum -20C. Now the device wakes up, and at these conditions the device may need VDDCORE up to 0.9 V before it can clock properly at 192 MHz. If the application wakes back up using 0.8 V and attempts to clock at 192 MHz, likely the logic will not operate correctly, and a fault will occur. The PVT Sensor would trigger the alarm immediately after the clock reaches a high frequency with a VDDCORE voltage that is too low. However, since the logic may not be operating correctly, the firmware may not execute properly, and the app likely is not able to request higher voltage from the PMIC.

The PVT Sensor can still be used with low-power modes, but situations like the above example should be avoided. The safest option is to wake back to a safe VDDCORE voltage. Revisit the example above, except this time when the MCU wakes, the hardware automatically raises the VDDCORE voltage to 0.9 V. This can be done before the application resumes execution, and before the FRO ungates the clock at 192 MHz. The app is able to execute at 192 MHz, and it can re-enable the PVT Sensor, and reduce the voltage again to an optimized level.

The `PMIC_MODE` pins on the i.MX RT500 enable the method above to wake up at a pre-configured safe voltage. There are two `PMIC_MODE` pins, which allow the MCU to request up to 4 different power configurations in hardware. The `PMIC_MODE` 0b00 is a special mode because the MCU will switch to this mode after POR or when waking from (Full) Deep Powerdown modes. The other 3 `PMIC_MODE` options are arbitrary, and can be used as desired by the application. The PMIC can be pre-configured to respond to these mode pins with the desired power configuration. When waking from Deep Sleep mode, the `PMIC_MODE` pins revert to the state in the `PDRUNCFG0[PMIC_MODE]` bits. The i.MX RT500 Reference Manual has more details on using the `PMIC_MODE` pins with the low-power modes. But Table 2 is an example of using the PMIC_MODE pins for an application that uses both Deep Sleep mode and the PVT Sensor. Some examples of voltages for VDDCORE are used here, but always refer to the data sheet for the required voltages:

Table 2. Example configuration of PMIC_MODE pins for Deep Sleep with PVT Sensor

PMIC_MODE	Power Mode	VDDCORE (V)	Description
0b00	Active, Boot	1.13	Default mode after POR. See the data sheet for VDDCORE requirements during boot
0b01	Active	0.9 / dynamic	VDDCORE starts at 0.9 V in this mode and is dynamically adjusted with the PVT Sensor.
0b10	Deep Sleep	0.6	Minimum retention voltage

Given the PMIC_MODE states above, here is an example sequence to wake back to a safe VDDCORE voltage:

1. MCU boots with `PMIC_MODE` of 0b00
2. Application configures PMIC, including VDDCORE voltages 0.9 V for `PMIC_MODE` 0b01, and 0.6 V for `PMIC_MODE` 0b10
3. Application changes `PDRUNCFG0[PMIC_MODE]` bitfield to 0b01. This changes `PMIC_MODE` pins, and PMIC adjusts VDDCORE to 0.9 V
4. Application enables PVT Sensor, and dynamically changes VDDCORE voltage setting for `PMIC_MODE` 0b01.
5. Before entering Deep Sleep, application configures PMIC for safe voltage 0.9 V using same `PMIC_MODE` 0b01

6. Application enters Deep Sleep, MCU changes PMIC_MODE pins to 0b10, which triggers PMIC to reduce voltage to 0.6 V
7. With wake-up event, PMIC_MODE pins revert to previous mode 0b01, triggers PMIC to raise VDDCORE to safe voltage 0.9 V
8. Application can restart PVT Sensor to reduce VDDCORE voltage

The downside of the safe voltage method above is that the application must first raise VDDCORE to a safe voltage level before entering Deep Sleep. This process is not the most energy efficient. Some applications may be able to wake and resume execution at the same reduced voltage, but this will depend on the use-case of each application. When the MCU is in Deep Sleep mode, temperature change is the main condition that could cause an issue with VDDCORE voltage during wake. Applications where ambient temperature does not change much, may be able to wake to the previous voltage found with the PVT Sensor. Some applications could have large temperature changes, but wake fast enough relative to the slower temperature change. In those cases, the temperature may not have a chance to change much during the short Deep Sleep duration. The example project uses this option: it enters Deep Sleep and drops VDDCORE to 0.6 V for 5 seconds, and then wakes to the previous reduced voltage. If an application chooses to wake with a reduced voltage, these limitations and conditions must be understood and considered. It may be better to wake to the safe voltage, and restart the PVT Sensor.

There is another option for applications waking from Deep Sleep with the i.MX RT500. When waking from Deep Sleep, there are only two clock options to source main_clk for the initial wake-up process: FRO_DIV4 or the LPOSC. Typically the application wakes clocked from FRO_DIV4, so main_clk is 48 MHz when the FRO is trimmed at 192 MHz. When main_clk is at a low frequency like this, VDDCORE is allowed to be at a low voltage, potentially down to 0.7 V. Therefore, the application has an opportunity to execute and check conditions first, before switching the clock source to FRO_DIV1 and clocking at 192 MHz. The app could measure the temperature with the integrated temp sensor, and compare it with a stored temperature from before sleeping. Or a better option may be to use a hardware timer, and measure how long it was in Deep Sleep. Either way, if the temperature change or sleep duration was large enough, the app can increase VDDCORE to a safe voltage first, before switching to the fast clock frequency. If the app determines there was not much change from conditions before entering Deep Sleep, it can resume at the reduced voltage found by the PVT Sensor before sleeping.

NXP's MCUXpresso SDK provides a driver for changing the power mode of the MCU, like entering Deep Sleep. If the application changes the voltage/frequency after waking using this SDK driver, be aware of how this driver works. This SDK driver is fsl_power, and includes the API POWER_EnterDeepSleep(). This API will ensure when the MCU wakes, it is clocked from FRO_DIV4 or LPOSC. But before returning to the application, it restores the clock source and frequency used when calling POWER_EnterDeepSleep(). If the app wants to wake and initially execute at a reduced frequency, the app should switch to that clock source/frequency first before calling POWER_EnterDeepSleep(). For example, this is the general sequence for main_clk at 192 MHz, and then waking at 48 MHz:

1. Application in Active mode, main_clk already clocked at 192 MHz from FRO_DIV1, PVT Sensor has already enabled a reduced voltage on VDDCORE
2. Change main_clk source to FRO_DIV4, frequency drops to 48 MHz
3. Call POWER_EnterDeepSleep(), MCU enters Deep Sleep mode
4. MCU wakes from wake event, execution resumes in POWER_EnterDeepSleep() from FRO_DIV4

5. `POWER_EnterDeepSleep()` returns to application
6. Application checks temperature change and/or duration in Deep Sleep.
7. If necessary, application increases VDDCORE voltage to support the higher frequency
8. Application changes `main_clk` to `FRO_DIV1`, back to 192 MHz

The best method to use when waking from Deep Sleep with the PVT Sensor is specific to the application. It depends on conditions like the temperature, how long the MCU is in Deep Sleep, the power supply system, the efficiency of changing supply voltages, and the wake-up latency requirements. If in doubt, it is best to use the safest method, and raise VDDCORE to a safe voltage level during wake-up.

5.4 Configuring Low-Voltage Detect (LVD)

The i.MX RT500 integrates an LVD that monitors the VDDCORE supply rail. The low-voltage detection threshold is programmable using `PMC_LVDCORECTRL[LVDCORELVL]` bits. There are some important considerations when using the LVD with DVS and with the PVT Sensor.

The LVD provides two primary functions:

- LVD can trigger when VDDCORE falls below the threshold. This can interrupt the application as an early warning, or it can reset the MCU
- On initial power-up and on wake-up from deep sleep, the LVD determines when VDDCORE is high enough to allow the MCU to operate.

The LVD minimum falling threshold is 720 mV, and the application can increment in 15 mV steps using LVDCORELVL. The rising threshold is 20 mV above the falling threshold for hysteresis. However, there are tolerances for the threshold across PVT variation. This means that each LVDCORELVL setting has ranges for each threshold, shown in Table 3.

Table 3. i.MX RT500 LVD Threshold ranges

LVDCORELVL	Falling Level (mV)		Rising Level (mV)	
	Min	Max	Min	Max
0	698	742	717	763
1	713	757	732	778
2	727	773	747	793
3	742	788	762	808
4	757	803	776	824
5	771	819	791	839
6	786	834	806	854
7	801	849	820	870
8	815	865	835	885
9	830	880	850	900
10	845	895	864	916
11	859	911	879	931
12	874	926	894	946
13	889	941	908	962

Table 3. i.MX RT500 LVD Threshold ranges*...continued*

LVDCORELVL	Falling Level (mV)		Rising Level (mV)	
	Min	Max	Min	Max
14	903	957	923	977
15	918	972	938	992

The LVD's interrupt and reset are optional. In active mode, the application could disable these and ignore the LVD. But if the application uses the LVD for its monitoring feature while dynamically changing the VDDCORE voltage, the LVD threshold must be configured. One option is to consider the minimum VDDCORE voltage allowed by the application for all devices. 800 mV is a good example since that is the minimum required for the FRO at 192 MHz. Then the application can set LVDCORELVL to the falling threshold range that is below the minimum voltage at the pins of VDDCORE. In this example, LVDCORELVL=3 would be the highest option, since option 4 could trigger the LVD when VDDCORE is at the valid level of 800 mV. Another option for applications is to adjust the LVD threshold when changing VDDCORE, to keep them more tightly aligned.

If using Deep Sleep mode and reducing VDDCORE, the LVD has additional considerations. First, if the application lowers VDDCORE below the LVD falling threshold (the minimum is 742 mV), then the LVD reset and interrupt must be disabled. The LVD rising threshold must also be set above the minimum VDDCORE voltage needed at wake-up. And this LVD value must be set before entering Deep Sleep. The example of waking to a safe voltage discussed in Section 5.3 used 900 mV minimum. To guarantee the voltage at the VDDCORE pins is at least 900 mV before the wake-up process starts, LVDCORELVL=13 can be used. But notice that the high limit with option 13 is 962 mV. This requires the PMIC to drive VDDCORE to at least 962 mV to ensure the MCU wakes. After waking, the application can reduce LVDCORELVL, and lower VDDCORE.

Another example discussed in Section 5.3 was waking at 48 MHz with FRO_DIV4. In that case, 700 mV could be the minimum VDDCORE required to wake, and the minimum range LVDCORELVL=0 will work. Again, it means that the PMIC should increase VDDCORE to at least 763 mV to ensure waking.

5.5 Troubleshooting

Using the PVT Sensor can greatly benefit the power consumption. But it can create some challenges when diagnosing issues in the application. As covered in this document, there are several requirements and limitations to consider when reducing VDDCORE with the PVT Sensor. When issues occur related to the PVT Sensor, it is usually because VDDCORE voltage is too low for the current conditions and clock frequencies. Common symptoms when the logic cannot keep up with the clock frequency include the app appearing to lockup or be unresponsive, erratic unpredictable faults that are hard to replicate consistently, problems only occurring on a small percentage of devices (because they are SS or FF devices), or problems only occurring at certain temperatures and issues that happen while waking from Deep Sleep.

If issues like these occur and the PVT Sensor is used in the application, the first step to diagnose is to disable the PVT Sensor. Keep VDDCORE at a fixed safe voltage. Since it is during debugging, the maximum 1.155 V can be used. If the issue goes away, then it is likely related to VDDCORE going too low at some point. From there, the application can be debugged to track down at what point VDDCORE is too low.

If using the PVT Sensor with low-power modes, try testing by disabling entry into the low-power modes to see if the issue goes away. If so, the issue is likely related to the entry/exit of a low-power mode. If the application is not already waking to a safe voltage level, try testing that method to rule out VDDCORE during wake-up.

The LVD could also be related to issues with using DVS or Deep Sleep mode. If unexpected resets are occurring, they may be triggered by the LVD. While debugging, try disabling the LVD reset to see if it impacts the issue. Potentially the LVD falling threshold is set too high relative to the VDDCORE supplied, and occasionally it triggers unwanted resets. However, consider that if the LVD is triggering, perhaps the issue is that VDDCORE is dropping lower than expected.

If issues are related to using Deep Sleep mode, the LVD may also be involved. For debugging, try disabling the LVD reset and interrupt before entering Deep Sleep mode. The issue could be also related to VDDCORE and LVD during wake-up. To help rule out issues, increase LVDCORELVL to a high setting that ensures VDDCORE is at a high and safe voltage before the MCU starts the wake-up process. Then ensure that the PMIC is increasing VDDCORE above the upper limit of the LVD rising threshold to trigger wake-up.

Another diagnostic method that can be helpful is using some output pins to help capture the state of the MCU during the application. The CLKOUT signal can be driven to a pin, and drive main_clk (or a divided down main_clk) or other clocks out to be measured. Also a GPIO can be used to track entry/exit of low-power modes. Then CLKOUT, wake GPIO, PMIC_MODE pins, and VDDCORE voltage can be monitored with test tools. It may help identify when VDDCORE has issues during wake or with changing clock frequencies. When measuring VDDCORE voltage, try to measure as close to the RT500 VDDCORE pins as possible, like on the bypass capacitors. Measuring the output at the PMIC may not expose how the VDDCORE pins are supplied.

Conclusion

The PVT Sensor integrated in the i.MX RT500 is a great option to increase the battery life in products. But the operating requirements detailed in this document are important for applications to avoid issues and maximize the benefits. NXP has tested a large volume of RT500 devices using the PVT Sensor, following these guidelines and requirements. The power consumption on the VDDCORE supply rail decreased 30 % on some devices. This is clocking at 192 MHz, and compared to the same device with VDDCORE at 0.9 V. So the battery life benefits can be substantial. For those applications where battery life matters most, explore the PVT Sensor and see how it can help in your application.

6 Revision history

Revision history

Revision number	Date	Substantive changes
0	30 August 2022	Initial release

7 Legal information

7.1 Definitions

Draft — A draft status on a document indicates that the content is still under internal review and subject to formal approval, which may result in modifications or additions. NXP Semiconductors does not give any representations or warranties as to the accuracy or completeness of information included in a draft version of a document and shall have no liability for the consequences of use of such information.

7.2 Disclaimers

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.

Terms and conditions of commercial sale — NXP Semiconductors products are sold subject to the general terms and conditions of commercial sale, as published at http://www.nxp.com/profile/terms, unless otherwise agreed in a valid written individual agreement. In case an individual agreement is concluded only the terms and conditions of the respective agreement shall apply. NXP Semiconductors hereby expressly objects to applying the customer's general terms and conditions with regard to the purchase of NXP Semiconductors products by customer.

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.

Suitability for use in non-automotive qualified products — Unless this data sheet expressly states that this specific NXP Semiconductors product is automotive qualified, the product is not suitable for automotive use. It is neither qualified nor tested in accordance with automotive testing or application requirements. NXP Semiconductors accepts no liability for inclusion and/or use of non-automotive qualified products in automotive equipment or applications.

In the event that customer uses the product for design-in and use in automotive applications to automotive specifications and standards, customer (a) shall use the product without NXP Semiconductors' warranty of the product for such automotive applications, use and specifications, and (b) whenever customer uses the product for automotive applications beyond NXP Semiconductors' specifications such use shall be solely at customer's own risk, and (c) customer fully indemnifies NXP Semiconductors for any liability, damages or failed product claims resulting from customer design and use of the product for automotive applications beyond NXP Semiconductors' standard warranty and NXP Semiconductors' product specifications.

Translations — A non-English (translated) version of a document, including the legal information in that document, is for reference only. The English version shall prevail in case of any discrepancy between the translated and English versions.

Security — Customer understands that all NXP products may be subject to unidentified vulnerabilities or may support established security standards or specifications with known limitations. Customer is responsible for the design and operation of its applications and products throughout their lifecycles to reduce the effect of these vulnerabilities on customer's applications and products. Customer's responsibility also extends to other open and/or proprietary technologies supported by NXP products for use in customer's applications. NXP accepts no liability for any vulnerability. Customer should regularly check security updates from NXP and follow up appropriately.

Customer shall select products with security features that best meet rules, regulations, and standards of the intended application and make the ultimate design decisions regarding its products and is solely responsible for compliance with all legal, regulatory, and security related requirements concerning its products, regardless of any information or support that may be provided by NXP.

NXP has a Product Security Incident Response Team (PSIRT) (reachable at PSIRT@nxp.com) that manages the investigation, reporting, and solution release to security vulnerabilities of NXP products.

7.3 Trademarks

Notice: All referenced brands, product names, service names, and trademarks are the property of their respective owners.

NXP — wordmark and logo are trademarks of NXP B.V.

AMBA, Arm, Arm7, Arm7TDMI, Arm9, Arm11, Artisan, big.LITTLE, Cordio, CoreLink, CoreSight, Cortex, DesignStart, DynamIQ, Jazelle, Keil, Mali, Mbed, Mbed Enabled, NEON, POP, RealView, SecurCore, Socrates, Thumb, TrustZone, ULINK, ULINK2, ULINK-ME, ULINK-PLUS, ULINKpro, µVision, Versatile — are trademarks or registered trademarks of Arm Limited (or its subsidiaries) in the US and/or elsewhere. The related technology may be protected by any or all of patents, copyrights, designs and trade secrets. All rights reserved.

Airfast — is a trademark of NXP B.V.

Contents

For more information, please visit: http://www.nxp.com

Date of release: 30 August 2022
Document identifier: AN13695