

# AN12591

## PMSM Field Oriented Control with LCD Display and Control Based on MIMXRT1010

Rev. 1 — November 2019

Application Note

### 1 Introduction

This application note describes the implementation of the sensorless Field-Oriented Control software for 3-phase Permanent Magnet Synchronous Motors (PMSM). It also includes LCD display and speed control on the MIMXRT1010 Evaluation Kit (EVK) based on the NXP i.MX RT1010 processor.

The i.MX RT1010 offers a new entry-point into the i.MX RT crossover processor series by providing the lowest-cost LQFP package option, combined with the high performance and ease of use known throughout the entire i.MX RT series.

This scheme provides a low-cost solution for motor control applications with LCD control and display, such as dishwashers, coffee machines, air purifiers.

This application note mainly introduces hardware implementation, including a detailed peripheral setup and driver description. The principle of sensorless PMSM Field-Oriented Control are described in *Sensorless PMSM Field-Oriented Control* (document DRM148).

### 2 Development platform

The FRDM-MC-LVPMSM development platform has the power supply input voltage of 24-48 VDC with the reverse polarity protection circuitry. The auxiliary power supply of 5.5 VDC is created to supply the FRDM MCU boards. The output current is up to 5 A RMS. The inverter itself is realized by the 3-phase bridge inverter (6-MOSFETs) and the 3-phase MOSFET gate driver. The analog quantities (such as the 3-phase motor currents, DC-bus voltage, and DC-bus current) are sensed on this board. There is also an interface for speed and position sensors (Encoder, Hall).

A TFT display (2.8" diagonal) with capacitive touchscreen is connected to RT1010-evk through Arduino connectors. This shield uses SPI for the display and the capacitive touchscreen controller uses I2C for communication with MCU. The LCD can be used to set motor control commands and display some motor control related parameters.

The block diagram of a complete NXP Freedom motor control development kit with LCD display and control is shown in [Figure 1](#).

### Contents

<b>1 Introduction</b> .....	<b>1</b>
<b>2 Development platform</b> .....	<b>1</b>
<b>3 MCU features and peripheral settings</b> .....	<b>3</b>
3.1 Hardware timing and synchronization.....	3
3.2 Peripheral settings.....	4
3.3 Motor control CPU load.....	6
<b>4 Tuning and controlling the application</b> .....	<b>7</b>
4.1 Hardware requirement & rework steps.....	7
4.2 Application control using LCD GUI.....	11
4.3 Application control using FreeMaster.....	13
<b>5 References</b> .....	<b>15</b>
<b>6 Revision history</b> .....	<b>15</b>







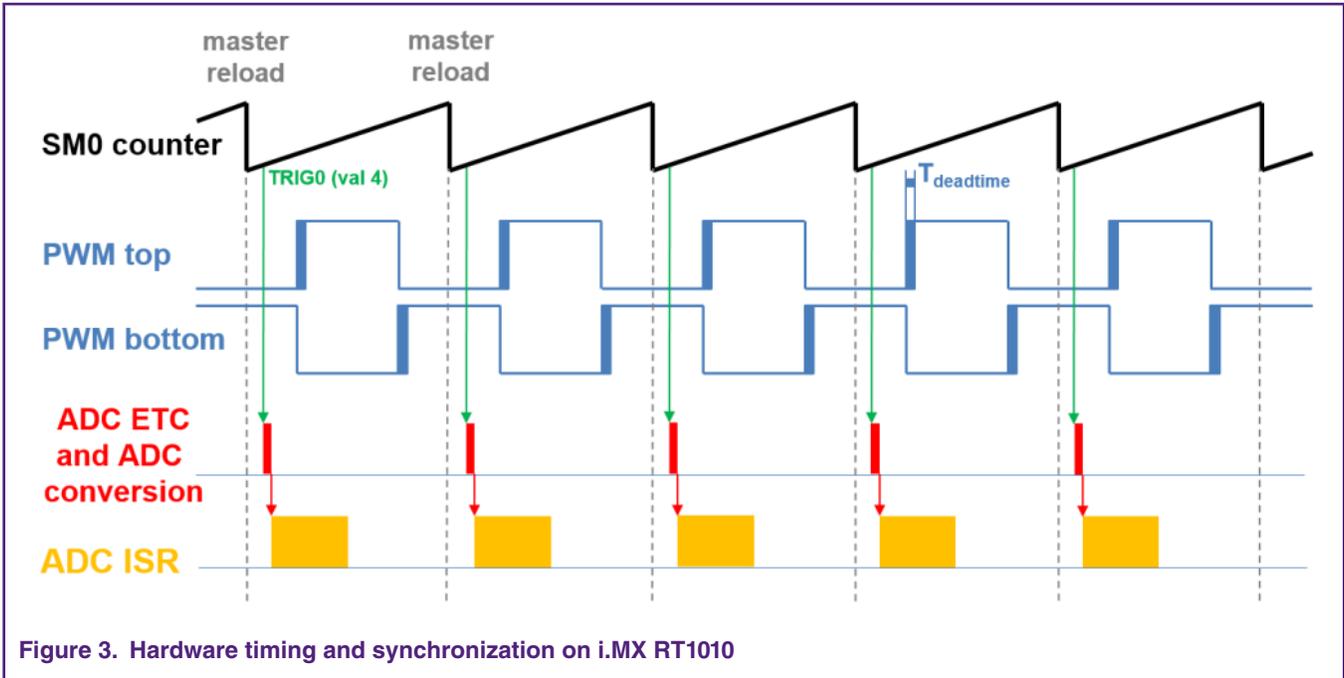
Figure 2. Actual Demo Hardware

### 3 MCU features and peripheral settings

The i.MX RT1010 is a new processor family featuring NXP's advanced implementation of the Arm Cortex-M7 Core which operates at speeds of up to 500 MHz in this chip. It provides high CPU performance and best real-time response. The i.MX RT1010 provides various memory interfaces, including NOR FLASH, Quad SPI, HyperBus, and a wide range of other interfaces for connecting peripherals, such as WLAN, Bluetooth™, GPS. Same as other i.MX processors, i.MX RT1010 also has rich audio features, including S/PDIF and I2S audio interface.

#### 3.1 Hardware timing and synchronization

Correct and precise timing is crucial for motor control applications. Therefore, the motor control-dedicated peripherals take care of the timing and synchronization on the hardware layer. In addition, it is possible to set the PWM frequency as a multiple of the ADC interrupt (ADC ISR) frequency where the FOC algorithm is calculated. In this case, the PWM frequency is equal to the FOC frequency. The timing diagram is shown in [Figure 3](#).



**Figure 3. Hardware timing and synchronization on i.MX RT1010**

- The top signal shows the eFlexPWM counter (SM0 counter). The dead time is emphasized at the PWM top and PWM bottom signals. The SM0 submodule generates the master reload at every opportunity.
- The SM0 generates trigger 0 (when the counter counts to a value equal to the TRIG4 value) for the ADC\_ETC (ADC External Trigger Control) with a delay of approximately  $T_{deadtime}/2$ . This delay ensures correct current sampling at the duty cycles close to 100 %.
- ADC\_ETC starts the ADC conversion.
- When the ADC conversion is completed, the ADC ISR (ADC interrupt) is entered. The FOC calculation is done in this interrupt.

### 3.2 Peripheral settings

This section describes the peripherals used for the motor control. On i.MX RT1010, there are three submodules from the enhanced FlexPWM (eFlexPWM) used for 6-channel PWM generation and one 12-bit ADC for the phase currents and DC-bus voltage measurement. The eFlexPWM and ADC are synchronized via submodule 0 from the eFlexPWM. The following settings are in the *mcdrv\_evkbimxrt1010.c* and *board.c* files and in their header files.

#### 3.2.1 Clock Control Module (CCM)

The CCM generates and controls the clocks of various modules in the design and manages the low-power modes. This module uses the available clock sources to generate the clock roots. The clock sources used in the motor control application are:

- PLL6, also called System PLL, with a frequency of 500 MHz.
- PLL3, also called USB1 PLL, with a frequency of 480 MHz.

The Arm clock core works at a frequency of 500 MHz and the clock source is PLL6. For this setting, these registers are set: CBCMR[PRE\_PERIPH\_CLK\_SEL], CBCDR[PERIPH\_CLK\_SEL], and CBCDR[AHB\_PODF] in *clock\_config.c*. The ADC, XBAR, and PWM are clocked from the IPG\_CLK\_ROOT output which has a frequency of 125 MHz. The CBCDR[IPG\_PODF] register must be set for this setting. The IPG\_CLK\_ROOT is sourced from the CORE\_CLK\_ROOT. The LPUART is sourced from the PLL3 at a frequency of 480 MHz divided by 6.

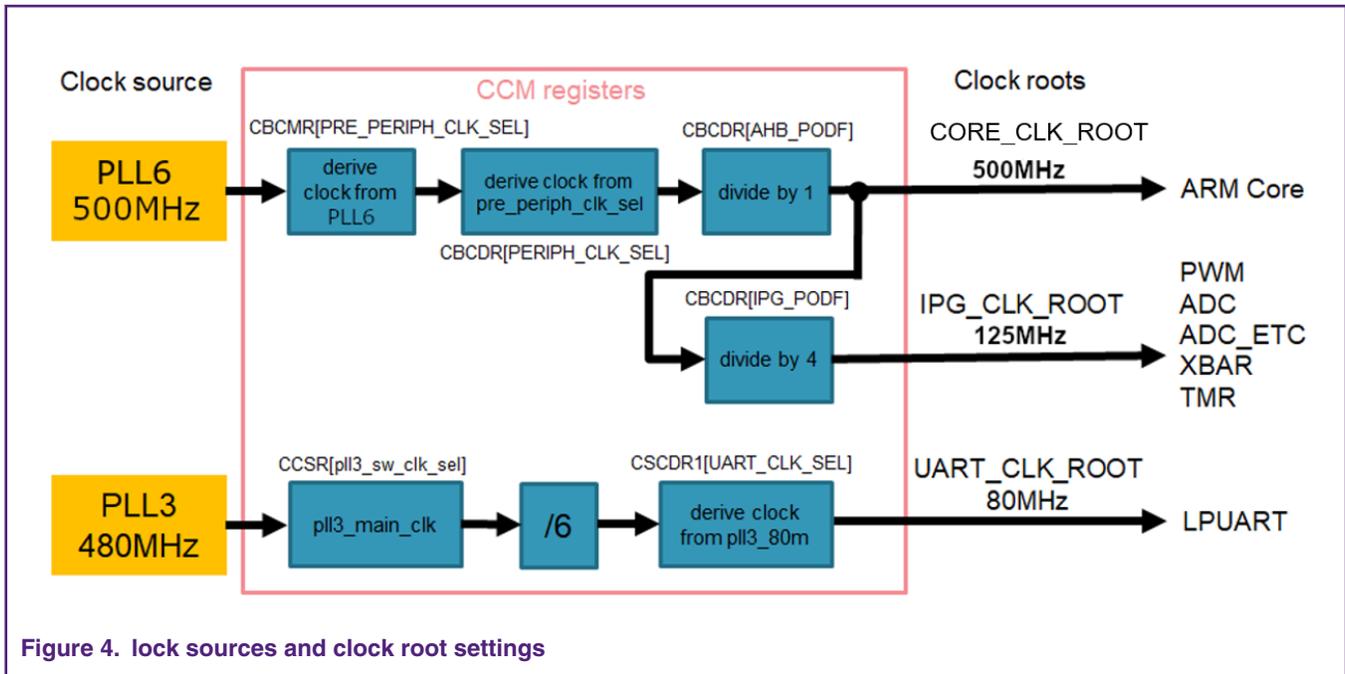


Figure 4. Clock sources and clock root settings

### 3.2.2 PWM generation—PWM1

- The eFlexPWM is clocked from the 125 MHz IPG\_CLK\_ROOT.
- Six channels from three submodules are used for the 3-phase PWM generation. Submodule 0 generates the master reload at event every  $n$ th opportunity, depending on the user-defined macro `M1_FOC_FREQ_VS_PWM_FREQ`.
- Submodules 1 and 3 get their clocks from submodule 0.
- The counters at submodules 1 and 3 are synchronized with the master reload signal from submodule 0 (submodule 2 is not used).
- Submodule 0 is used for synchronization with ADC\_ETC. The submodule generates the output trigger after the PWM reload, when the counter counts to VAL4.
- Fault mode is enabled for channels A and B at submodules 0, 1, and 3 with automatic fault clearing (the PWM outputs are re-enabled at the first PWM reload after the fault input returns to zero).
- The PWM period (frequency) is determined by how long it takes the counter to count from INIT to VAL1. By default, `INIT = -MODULO/2 = -6250` and `VAL1 = MODULO/2 - 1 = 6249`. The eFlexPWM clock is 125 MHz so it takes 0.0001 s (10 kHz).
- Dead time insertion is enabled. The dead time length is defined by the user in the `M1_PWM_DEADTIME` macro. ADC External Trigger Control—ADC\_ETC

### 3.2.3 ADC External Trigger Control—ADC\_ETC

- The ADC\_ETC module enables multiple users to share the ADC modules in the Time Division Multiplexing (TDM) way. The external triggers can be brought from the Cross BAR (XBAR) or other sources. The ADC scan is started via ADC\_ETC.
- ADC have its own trigger chains.
- The trigger chain length is set to 3. The back-to-back ADC trigger mode is enabled.

### 3.2.4 Analog sensing—ADC1

- ADC1 is used for the MC analog sensing of the currents and DC-bus voltage.
- The clock frequency for ADC1 is 62.5 MHz. It is taken from IPG\_CLK\_ROOT and divided by 2.

- The ADCs operate as 10-bit with the single-ended conversion and hardware trigger selected. The ADCs are triggered from ADC\_ETC by the trigger generated by eFlexPWM.
- The conversion complete interrupt is enabled and serves the FOC fast loop algorithm generated after the last scan is completed by ADC1.

### 3.2.5 Peripheral interconnection —XBARA1

The crossbar is used to interconnect the trigger from the PWM to the ADC\_ETC in this application.

- The FLEXPWM1\_PWM1\_OUT\_TRIG0\_1 output trigger (generated by submodule 0) is connected to ADC\_ETC\_XBAR0\_TRIG0.

### 3.2.6 LCD display—SPI

SPI(Serial Peripheral Interface) is used for the LCD communication between the MCU board and the LCD display.

- The baud rate is set to 500000 bit/s.
- The frame rate is set to 8 bits per frame.
- CPOL is set to Clock Polarity Active High.
- CPHA is set to Clock Phase First Edge.
- The direction is set to MSB First.
- The other settings are set to default.

### 3.2.7 LCD touch control—I2C

I2C is used for the LCD communication between the MCU board and the LCD touch part.

- The baud rate is set to 100000 bit/s.
- The other settings are set to default.

### 3.2.8 FreeMASTER communication—LPUART0

LPUART0 (Low-Power Universal Asynchronous Receiver and Transmitter) is used for the FreeMASTER communication between the MCU board and the PC.

- The baud rate is set to 115200 bit/s.
- The receiver and transmitter are both enabled.

## 3.3 Motor control CPU load

The CPU load is measured using the SysTick timer. The CPU load is dependent on the fast loop (FOC calculation) and slow loop (speed loop) frequencies. In this case, it applies to the fast loop frequency of 10 kHz and the slow loop frequency of 1 kHz. In this application, the slow loop is obtained by dividing the frequency of fast loop, so there is no independent slow loop. So the total CPU load equals to the fast loop load and is calculated using the following equations.

$$CPU_{fast} = cycles_{fast} \frac{f_{fast}}{f_{cpu}} 100 \left[ \% \right]$$

Where the used characters are:

$$CPU_{fast}$$

--the CPU load taken by the fast loop (ADC ISR).

$$cycles_{fast}$$

--the number of cycles consumed by the fast loop.

--the frequency of the fast loop calculation (10 kHz).

$$f_{cpu}$$

--the CPU frequency.

Using the equations above the motor control CPU load is 5.884%.

## 4 Tuning and controlling the application

The reference design can be controlled by the GUI based on LCD or FreeMaster with PC.

This section provides information about the tools and recommended procedures for controlling the sensorless PMSM Field-Oriented Control (FOC) application. Includes the setup steps of hardware, software environment and demonstration steps.

### 4.1 Hardware requirement & rework steps

#### 4.1.1 Hardware requirement

- RT1010 EVK
- [FRDM-MC-LVPMSM](#)
- [FRDM-MC-LVMTR](#)
- [LCD shield](#)

Connector with long pin for Arduino like [Figure 5](#):

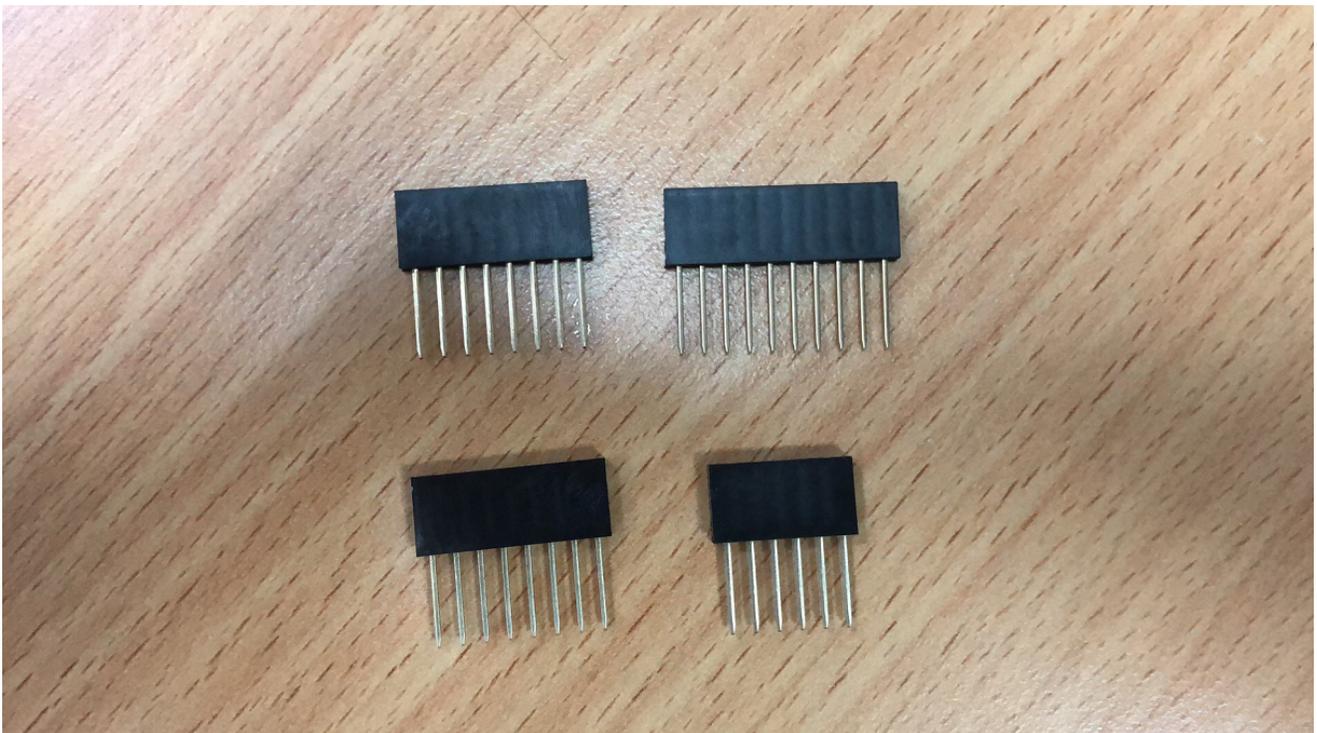


Figure 5. Connector for Arduino

#### 4.1.2 Hardware rework steps

- Welding 0 ohm resistor to R1849,R1850,R1852,R1853,R1854.

- Welding 0 ohm resistor R795,R796,R797,R798.
- DNP resistor R10.
- Left pad of R793 connect to Right pad of R799 [Figure 6](#)
- Left pad of R794 connect to Right pad of R800 [Figure 6](#)
- pad of R1870 connect to Left pad of R10 [Figure 7](#)

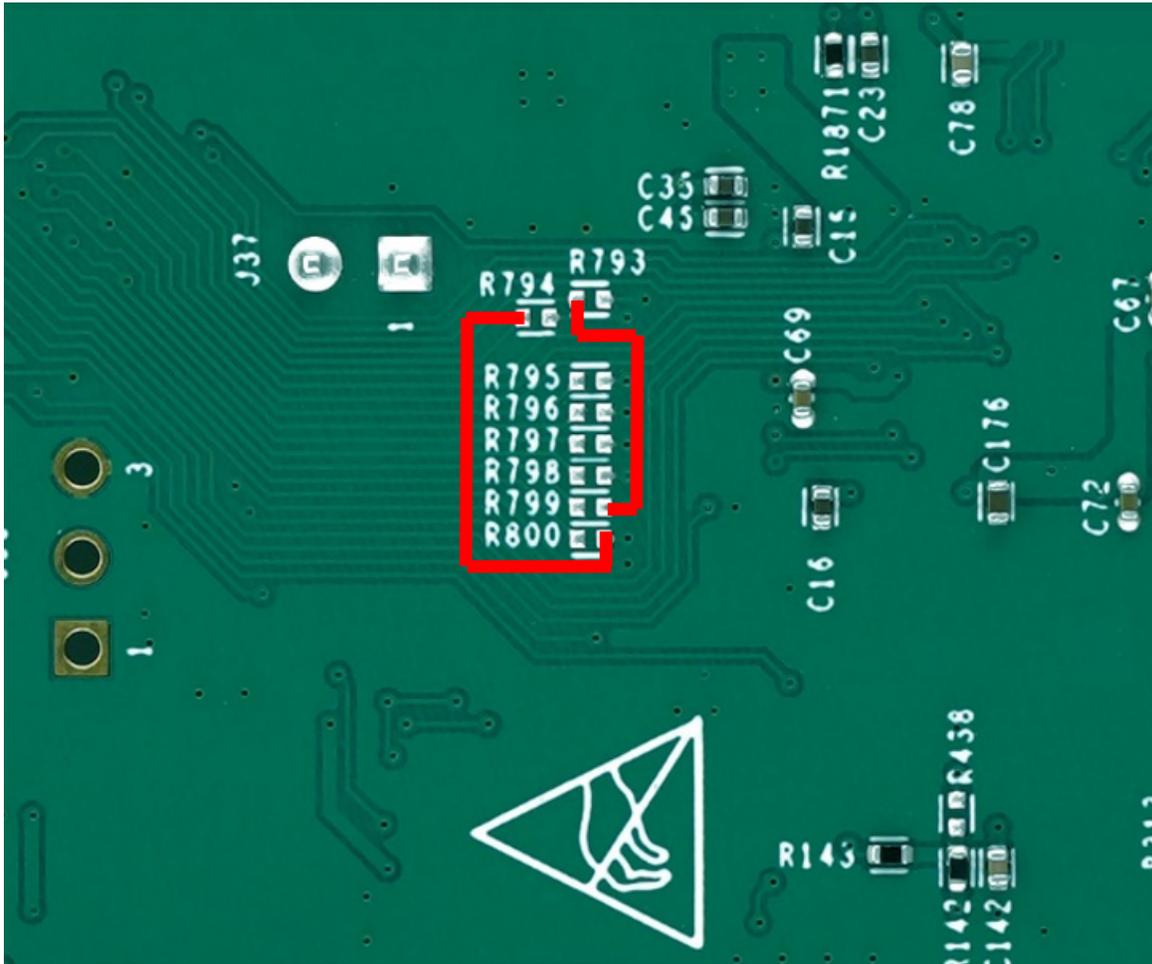


Figure 6. Left pad of R793 connect to Right pad of R799 and Left pad of R794 connect to Right pad of R800

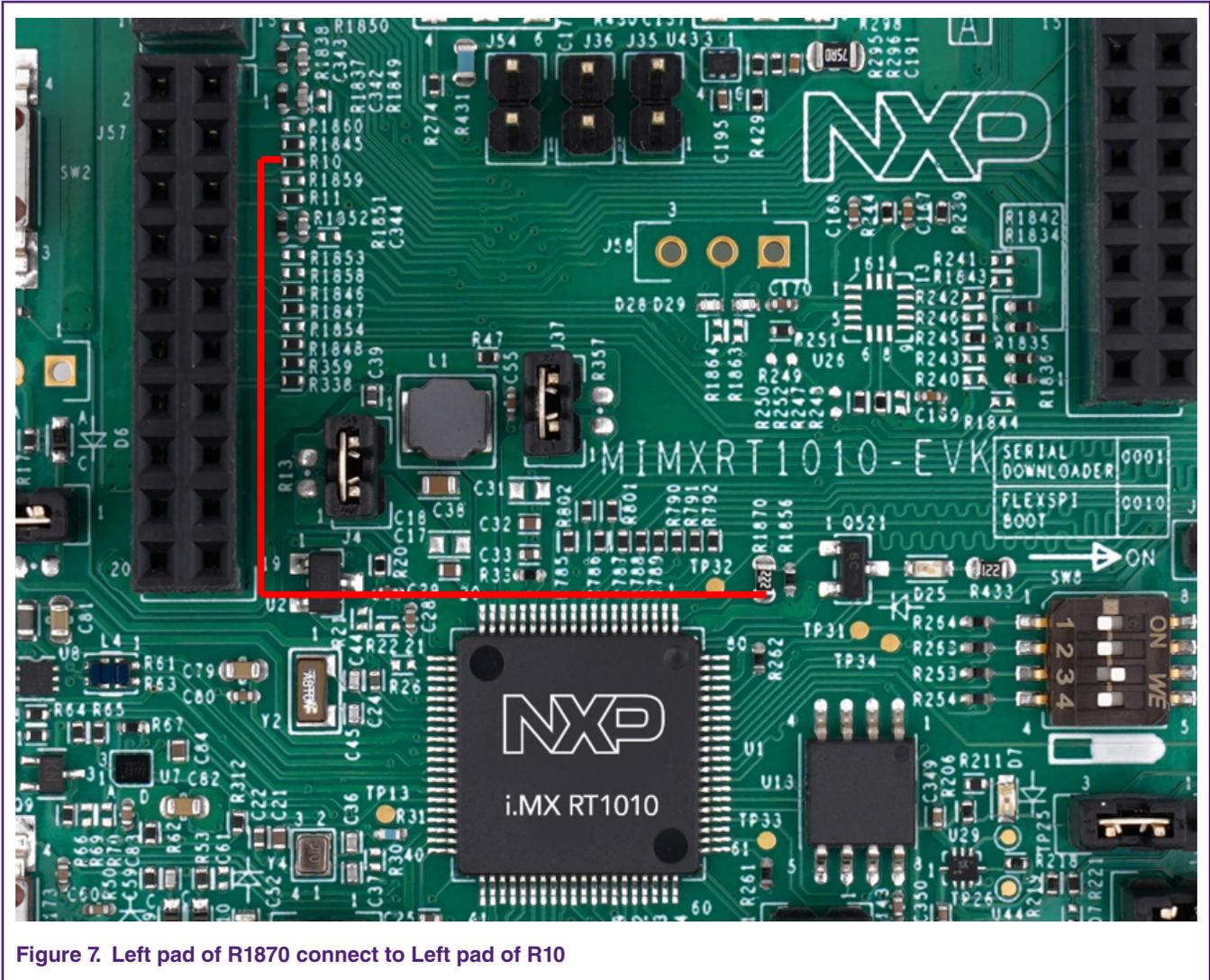


Figure 7. Left pad of R1870 connect to Left pad of R10

- Connect the FRDM-MC-LVPMSM to RT1010 EVK ,LCD with the connector plug in the FRDM-MC-LVPMSM board like following [Figure 8](#)

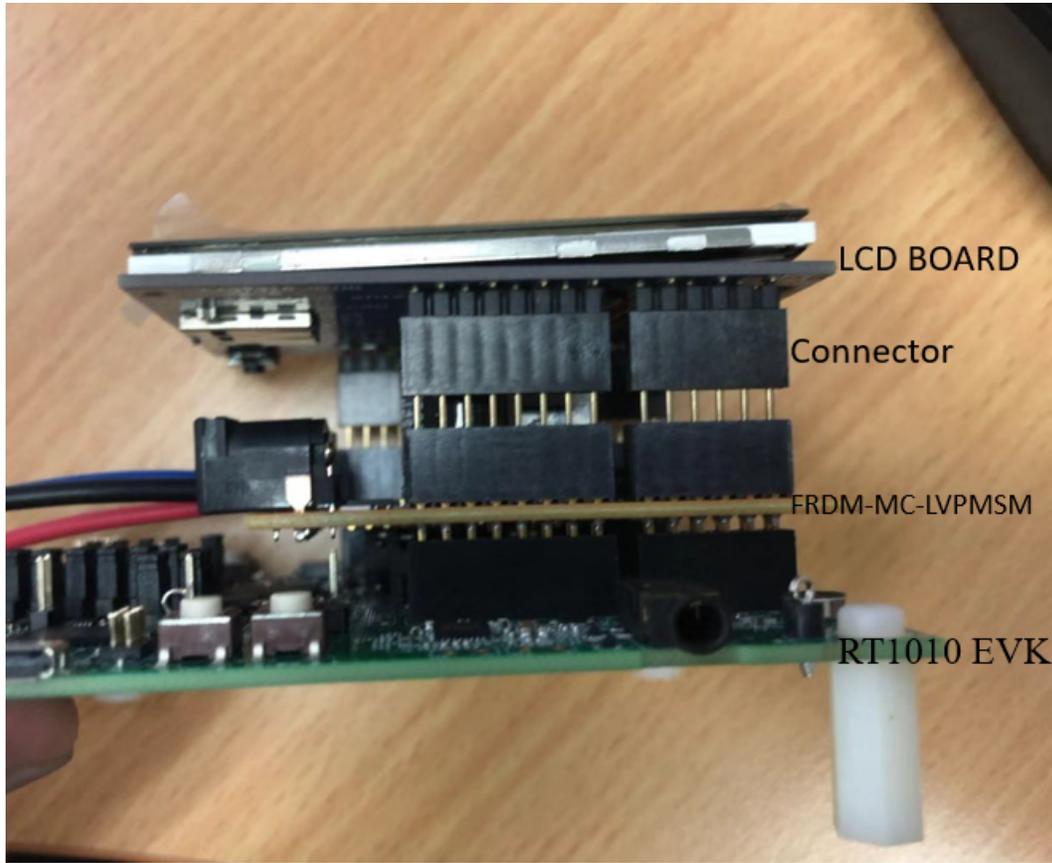


Figure 8. Board connection

- Remove the connector of J1 and then power on the board with 24 V adapter.

**NOTE**

Make sure when using 24 V power, no jumper is connected on J1



## 4.2 Application control using LCD GUI

### 4.2.1 Software environment setup

- Open the project with IAR(8.32 version or above).
- Choose “flexspi\_nor\_debug” branch.
- Make and download the firmware to chip.

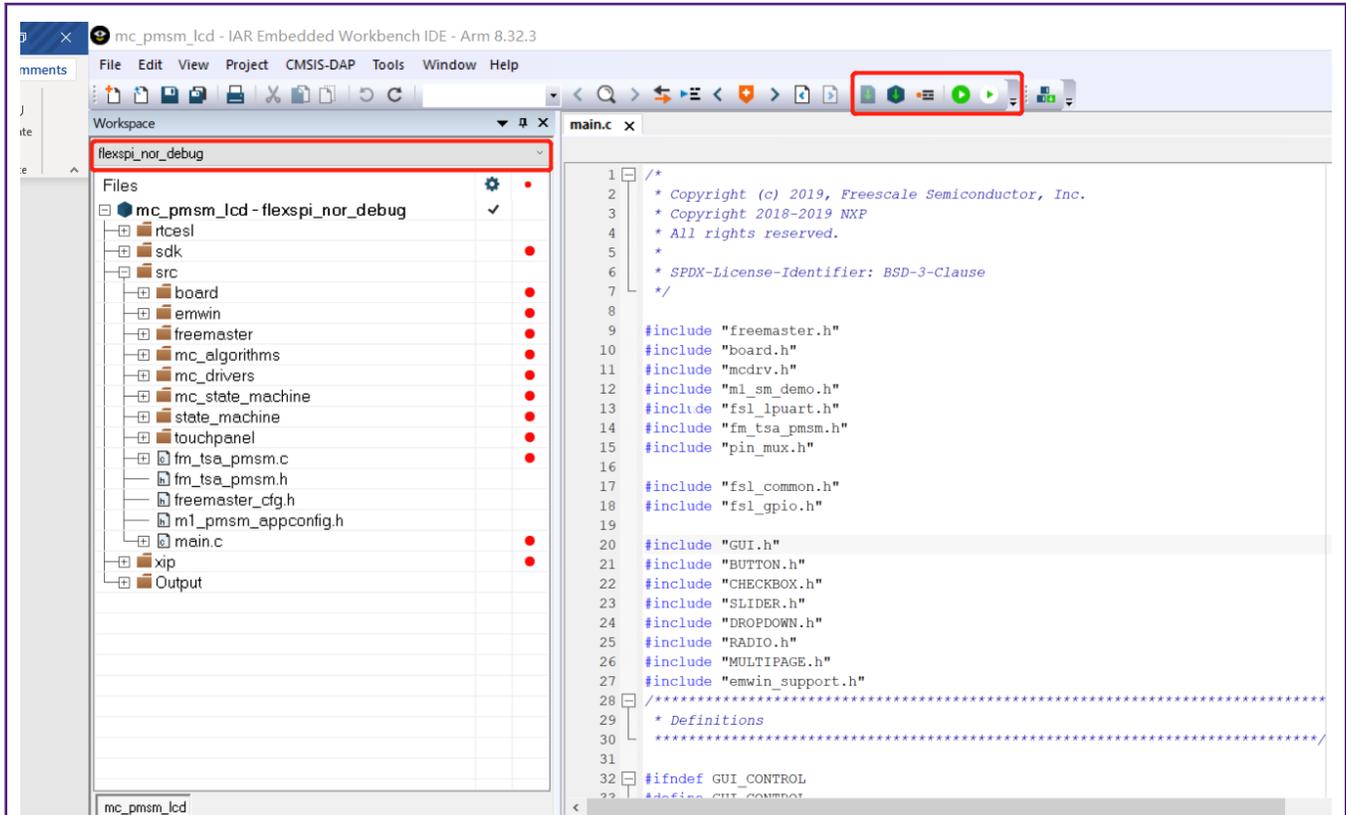


Figure 10. Software environment

### 4.2.2 Demonstration steps

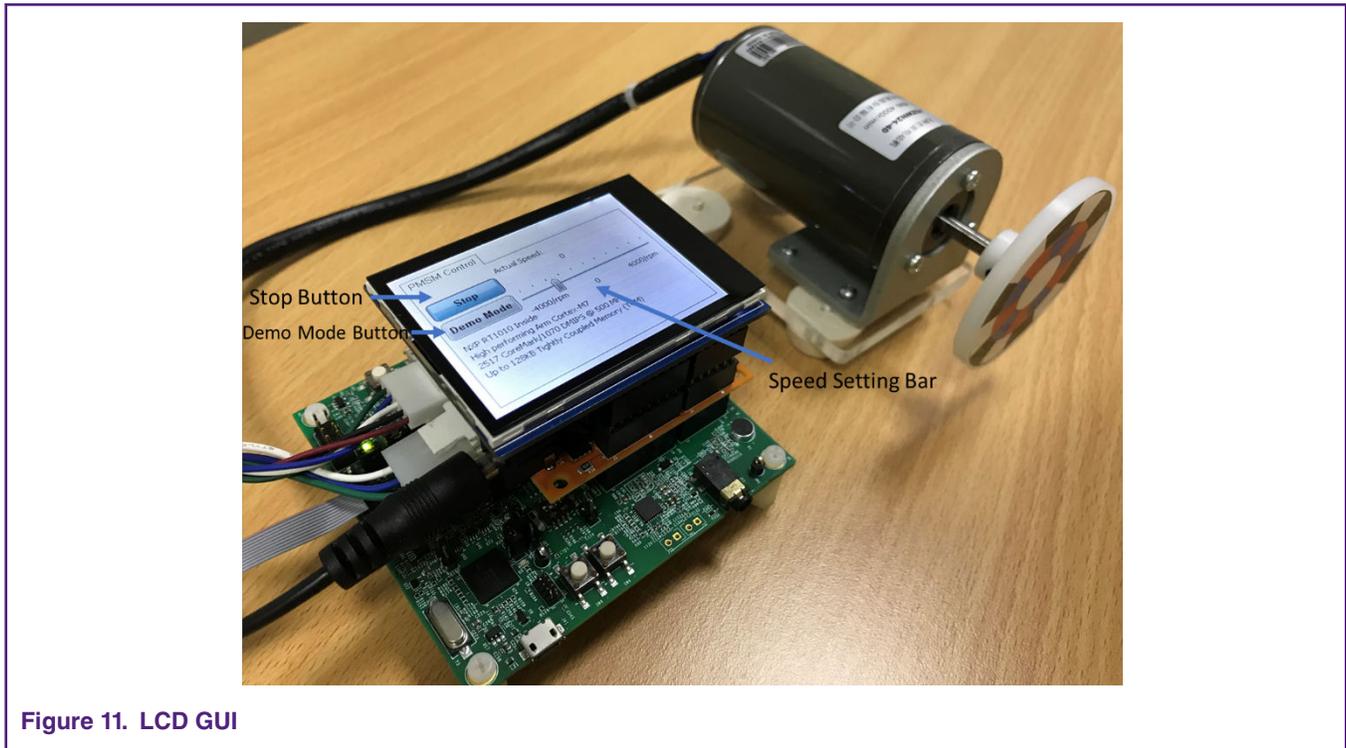


Figure 11. LCD GUI

- Remove the connector of J1 and then power on the board with 24 V adapter.
- To run the demo in demo mode, Click the 'Demo Mode' icon: In this mode, PMSM motor automatically changes different speeds in forward and reversal mode. Actual speed is displayed by speed bar or text box above it.
- Slide the speed setting bar can also change the speed command axis to set the motor speed manually.
- To stop the demo Click the 'Stop' icon.

## 4.3 Application control using FreeMaster

### 4.3.1 Software environment setup

- Open the project with IAR(8.32 version or above)
- Choose "flexspi\_nor\_debug" branch.
- Mask predefined definitions "GUI\_CONTROL" in main.c file to disable GUI related initialization and enable FreeMaster control.
- Make and download the firmware to chip.
- Open the FreeMaster project(src\common\motor\_control\freemaster\pmsm\_demo.pmp )with FreeMaster (2.5 version or above)

### 4.3.2 FreeMaster demonstration steps

- Configure related RS232 port and set 115200 baud rate.
- Click "GO" button to start the communication.
- [Figure 12](#) shows the FreeMaster interface, Set "Application Switch" to "ON" by value set or click "RUN" button.
- Set speed command by direct change the value of "Speed Required" or click "DemoMode" button to enable the demo in demo mode: In this mode PMSM motor will automatically change different speeds in forward and reversal mode.

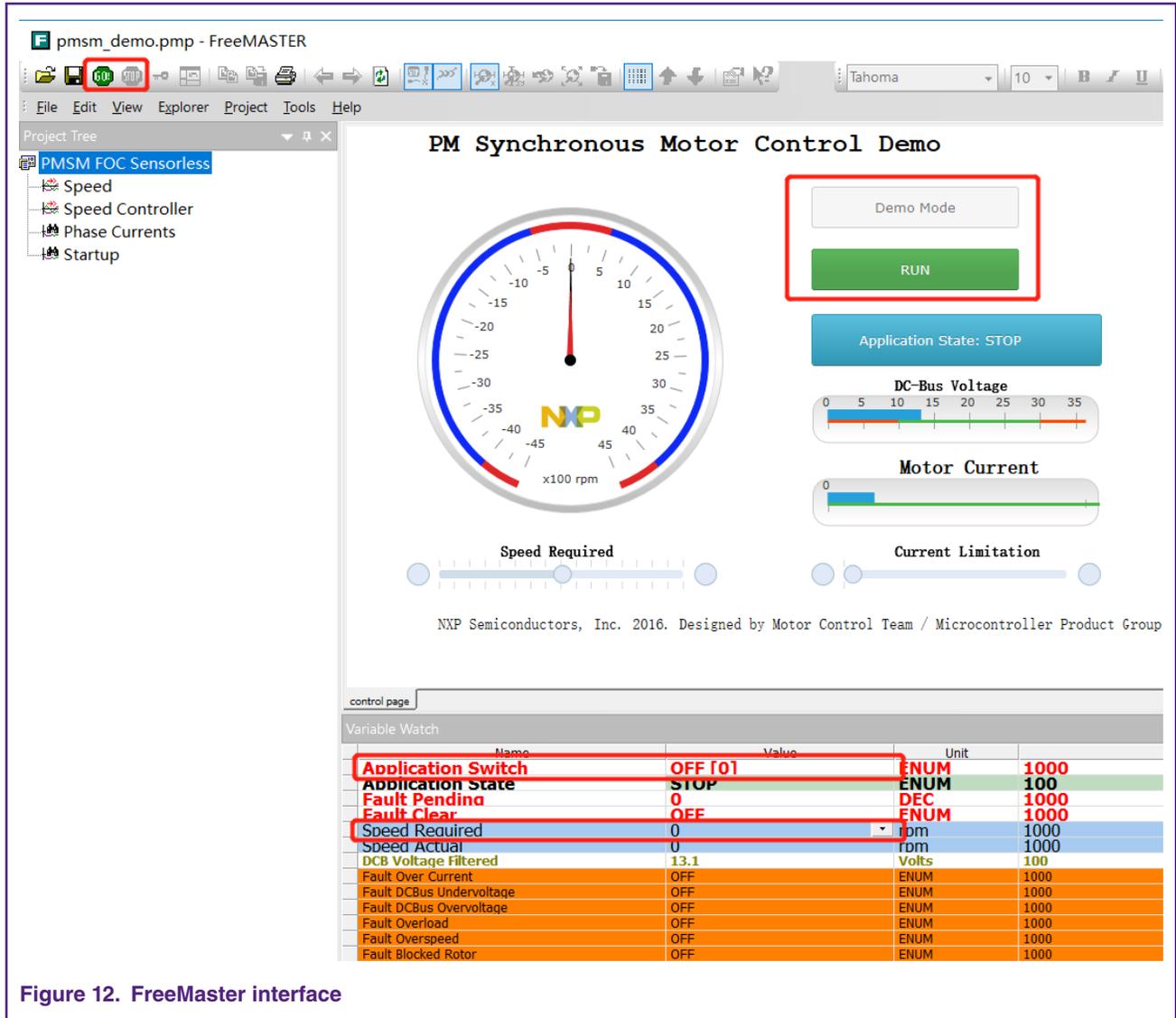


Figure 12. FreeMaster interface

- Customer can add the variables to be observed to oscilloscope or watch window to monitor motor status (Figure 13) .



Figure 13. FreeMASTER oscilloscope interface

## 5 References

These references are available on [www.nxp.com](http://www.nxp.com):

1. *Sensorless PMSM Field-Oriented Control* (document [DRM148](#)).
3. *Sensorless PMSM Field-Oriented Control on Kinetis KV* (document [AN5237](#)).
4. *PMSM Field-Oriented Control on MIMXRT1050 EVK* (document [AN12169](#)).
5. *i.MX RT1010 Processor Reference Manual* (document [IMXRT1010RM](#)).

## 6 Revision history

Rev. Number	Date	Substantive Change(s)
0	September 2019	Initial release.
1	November 2019	Added section 3.3 and 4.3, updated figure 10

## How To Reach Us

### Home Page:

[nxp.com](http://nxp.com)

### Web Support:

[nxp.com/support](http://nxp.com/support)

Information in this document is provided solely to enable system and software implementers to use NXP products. There are no express or implied copyright licenses granted hereunder to design or fabricate any integrated circuits based on the information in this document. NXP reserves the right to make changes without further notice to any products herein.

NXP makes no warranty, representation, or guarantee regarding the suitability of its products for any particular purpose, nor does NXP assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. "Typical" parameters that may be provided in NXP data sheets and/or specifications can and do vary in different applications, and actual performance may vary over time. All operating parameters, including "typicals," must be validated for each customer application by customer's technical experts. NXP does not convey any license under its patent rights nor the rights of others. NXP sells products pursuant to standard terms and conditions of sale, which can be found at the following address: [nxp.com/SalesTermsandConditions](http://nxp.com/SalesTermsandConditions).

While NXP has implemented advanced security features, all products may be subject to unidentified vulnerabilities. Customers are responsible for the design and operation of their applications and products to reduce the effect of these vulnerabilities on customer's applications and products, and NXP accepts no liability for any vulnerability that is discovered. Customers should implement appropriate design and operating safeguards to minimize the risks associated with their applications and products.

NXP, the NXP logo, NXP SECURE CONNECTIONS FOR A SMARTER WORLD, COOLFLUX, EMBRACE, GREENCHIP, HITAG, I2C BUS, ICODE, JCOP, LIFE VIBES, MIFARE, MIFARE CLASSIC, MIFARE DESFire, MIFARE PLUS, MIFARE FLEX, MANTIS, MIFARE ULTRALIGHT, MIFARE4MOBILE, MIGLO, NTAG, ROADLINK, SMARTLX, SMARTMX, STARPLUG, TOPFET, TRENCHMOS, UCODE, Freescale, the Freescale logo, Altivec, C-5, CodeTEST, CodeWarrior, ColdFire, ColdFire+, C-Ware, the Energy Efficient Solutions logo, Kinetis, Layerscape, MagniV, mobileGT, PEG, PowerQUICC, Processor Expert, QorIQ, QorIQ Qonverge, Ready Play, SafeAssure, the SafeAssure logo, StarCore, Symphony, VortiQa, Vybrid, Airfast, BeeKit, BeeStack, CoreNet, Flexis, MXC, Platform in a Package, QUICC Engine, SMARTMOS, Tower, TurboLink, UMEMS, EdgeScale, EdgeLock, eIQ, and Immersive3D are trademarks of NXP B.V. All other product or service names are the property of their respective owners. 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,  $\mu$ Vision, 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. Oracle and Java are registered trademarks of Oracle and/or its affiliates. The Power Architecture and Power.org word marks and the Power and Power.org logos and related marks are trademarks and service marks licensed by Power.org.

© NXP B.V. 2019.

All rights reserved.

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

For sales office addresses, please send an email to: [salesaddresses@nxp.com](mailto:salesaddresses@nxp.com)

Date of release: November 2019

Document identifier: AN12591

