

# Using the Programmable Gain Amplifier in the S12ZVLA

by: Jesus Sanchez

## Contents

1	Introduction.....	1
2	Programmable gain amplifier.....	1
3	Example software.....	6
4	Conclusion.....	9

## 1 Introduction

This application note describes the Programmable Gain Amplifier (PGA) in the S12ZVLA128 microcontroller that is a member of the S12 MagniV 16-bit microcontroller devices. The further sections describe in detail how to use the PGA module in a typical application. These and other features make this device ideal for LIN node applications such as sensors, switch panels, or small actuators.

This application note provides a simple PGA driver which will allow the user to enable measurements on any of the S12ZVLA128 MagniV devices. It also provides a detailed explanation of the configuration of the registers.

## 2 Programmable gain amplifier

The PGA offers multiple internal gain options, its gain can be programmed from 10x, 20x, 40x and 80x. There are two PGA inputs: AN2 and AN3. The reference voltage should be selectable internally from the DAC outputs (DACI),  $VDDA/2$ , or from the device pinout, at pin AN1. The output of the second amplification stage (PGA\_OUT) must be routed to internal ADC channel 7.

The following are the features of PGA:



## Programmable gain amplifier

- Amplification of analog input signal with selectable gain of 10x, 20x, 40x, 80x
- Typical current consumption 1 mA
- Offset compensation
- Internal VDDA / 2 reference voltage generation or external signal as reference voltage (see top level connections)
- Amplifier output connected to ADC

The following are the applications/uses of PGA:

- Amplification
- Differential to Single-Ended Conversion
- Differential-Input, Differential-Output Signal
- Industrial and Automotive Signal Conditioning
- Sensor Interface and Signal Processing
- General Purpose Data Acquisition

## 2.1 Input/ Output connections

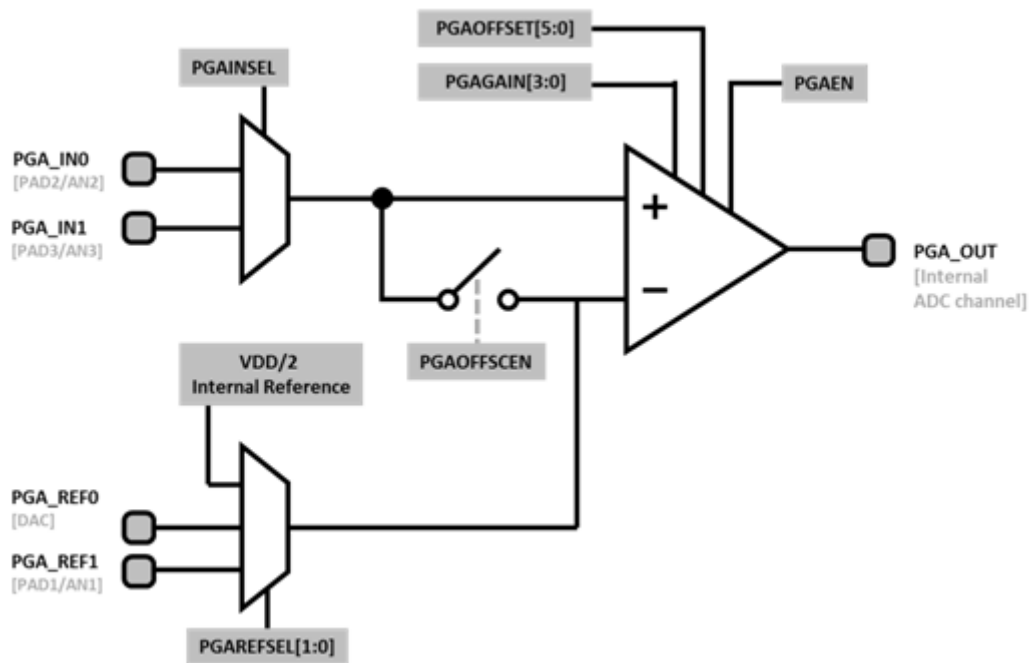


Figure 1. PGA block diagram

Table 1. Input/ Output PGA signals

SIGNAL	DESCRIPTION
PGA_IN0	This analog pin is used as amplifier plus input voltage if the associated control register bit is set.
PGA_IN1	This analog pin is used as amplifier plus input voltage if the associated control register bit is set.
PGA_REF0	This analog pin is used as reference voltage and amplifier minus input voltage if the associated control register bit is set.
PGA_REF1	This analog pin is used as reference voltage and amplifier minus input voltage if the associated control register bit is set.

Table continues on the next page...

**Table 1. Input/ Output PGA signals (continued)**

SIGNAL	DESCRIPTION
PGA_OUT	This analog pin provides the analog amplifier output voltage of the PGA as a function of the gain, offset and the reference voltage.

## 2.2 Initialization

Follow the details given below to initialize PGA.

- Configure CPMU voltage regulator to 5 V.
- Set up the system clock.
- Enable the ADC Module.
- Initialize the offset compensation routine for PGA.
- Configure PGA to the circuit application.
- Wait a settling time.
- Read [PGA\_OUT] value.

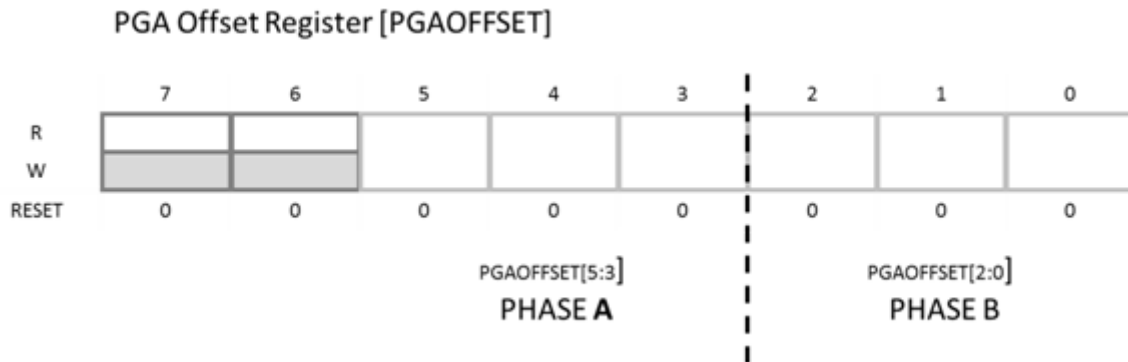
**NOTE**

The PGA will be operated from the analog 5 V power domain VDDA.

### 2.2.1 Offset compensation

The offset compensation is used to compensate the input offset in the amplifier. The following procedure is recommended.

- The PGA\_EN bit and the PGAOFFSCEN bit must be set to enable the PGA and to connect the PGAIN internally to the reference voltage.
- Select the reference generation from the internal VDDA/2 buffer. The gain must be set to 80x.
- The offset compensation is done in two steps as shown in the [Figure 2](#).



**Figure 2. PGA Offset register (PGAOFFSET)**

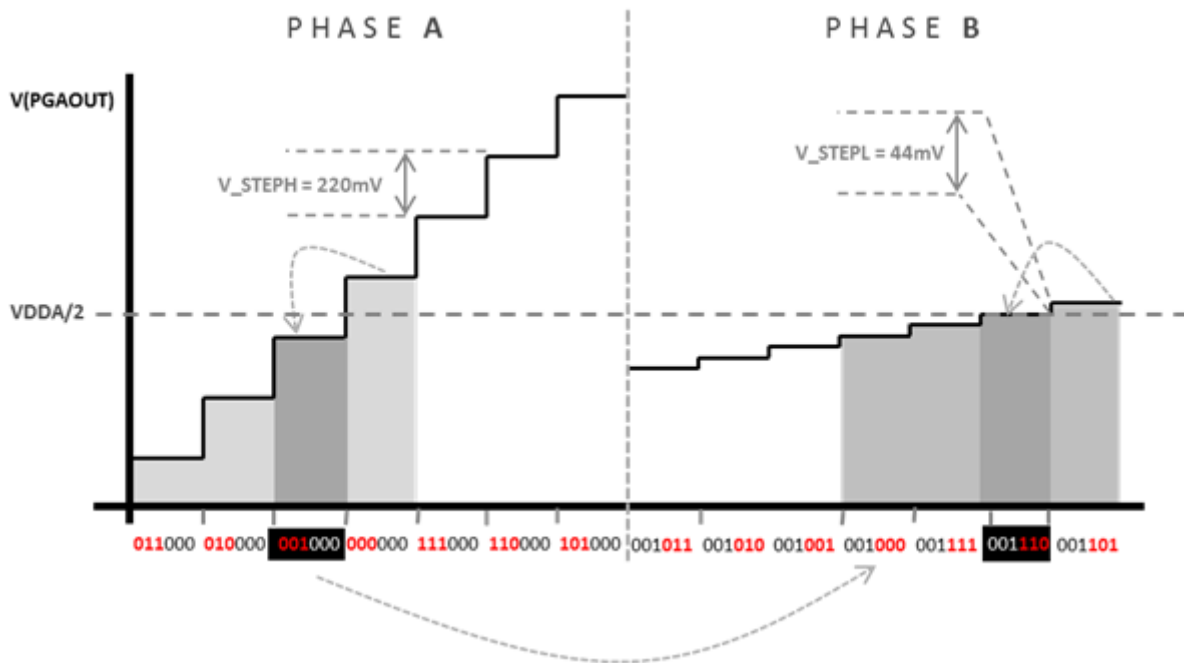
1. Phase A: With PGAOFFSET[2:0] = 0x000, step through the offset compensation values using the 3 MSBs of the PGAOFFSET[5:3] = {0x011, 0x010, 0x001, 0x000, 0x111, 0x110, 0x101} and measure the PGA\_OUT value with the ADC. Select as optimal offset compensation value for the higher three bits the PGAOFFSET[5:3] which is closest to the expected ADC reading for VDDA/2.

## Programmable gain amplifier

- Phase B: With the optimal PGAOFFSET[5:3] value, and setting step through the offset compensation values PGAOFFSET[2:0] = {0x011, 0x010, 0x001, 0x000, 0x111, 0x110, 0x101} and measure the PGA\_OUT value with the ADC. Select as optimal offset compensation value for the lower three bits the PGAOFFSET[2:0] which is closest to the expected ADC reading of  $V_{DDA}/2$ .

**Table 2. Offset compensation steps**

Phase A		Phase B	
PGAOFFSET[5:3]	$\Delta V_{OUT}$	PGAOFFSET[2:0]	$\Delta V_{OUT}$
011	$-3 \cdot V_{STEP\_H}$	011	$-3 \cdot V_{STEP\_H}$
010	$-2 \cdot V_{STEP\_H}$	010	$-2 \cdot V_{STEP\_H}$
001	$-1 \cdot V_{STEP\_H}$	001	$-1 \cdot V_{STEP\_H}$
000	0	000	0
111	$+1 \cdot V_{STEP\_H}$	111	$+1 \cdot V_{STEP\_H}$
110	$+2 \cdot V_{STEP\_H}$	110	$+2 \cdot V_{STEP\_H}$
101	$+3 \cdot V_{STEP\_H}$	101	$+3 \cdot V_{STEP\_H}$
si100	0	100	0



**Figure 3. Offset compensation timing diagram**

## 2.3 Using the PGA for differential voltage measurement

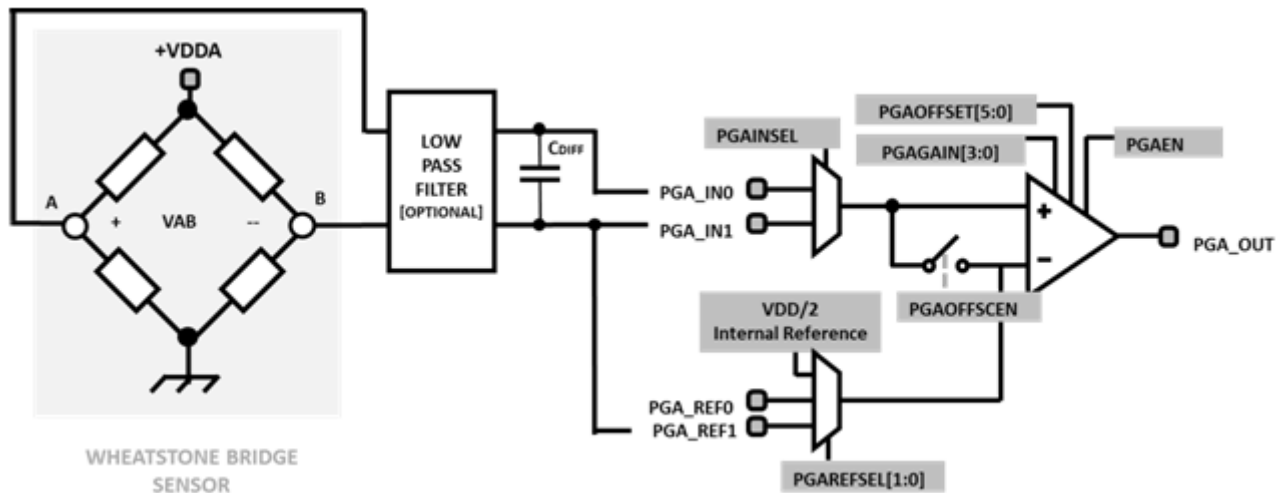
For sensor applications it is often required to measure a small differential voltage  $V_{diff}$ . The PGA is not capable of amplifying a differential voltage, but an algorithm to calculate the differential voltage can be implemented.

The PGA contains two input pins PGA\_IN0 and PGA\_IN1 which can be multiplexed by the ADC command list (see section 1.9.6.1 of the reference manual) or manually using the register PGAINSEL. By subtracting the ADC readings of the two pins the amplified differential voltage can be calculated.

$$V_{DIFF} = \frac{[(PGAOUT\_READ(PGA\_IN0) - (PGAOUT\_READ(PGA\_IN1))]}{PGA\_GAIN}$$

For this algorithm two requirements must be met:

1. The minimum time for the input signal multiplexing is given by PGA to ADC settling time  $t_{pGA\_settling}$ . The rate of signal change within  $t_{pGA\_settling}$  must be small.
2. The common mode input voltage range of the differential input signals must be limited that for a given gain A PGA a reference voltage  $V_{ref}$  can be selected so that both amplified signals do not saturate.



**Figure 4. Application circuit with a Wheatstone bridge sensor**

In cases where noise is relatively high, it's usually add LPB filter the output of a bridge sensor. This reduces wide band noise and can help to reject EMI/RFI. In the figure X, it is shown is a single-pole differential low-pass filter configuration. The resistors in the low-pass band filter should likewise be very small, no more than 200 ohms. If they are too large, increased noise and gain errors will result. The differential capacitor  $[C_{DIFF}]$ , size depends on the desired data rate and settling time. It can be anywhere from 100 pF and 1 uF.

### PGA circuit application

Using an equivalent Wheatstone Bridge in order to apply a differential-voltage (from 10 mV to 300 mV) to the PGA interface, it is verified the accuracy and stability of the gains pre-defined in the module.

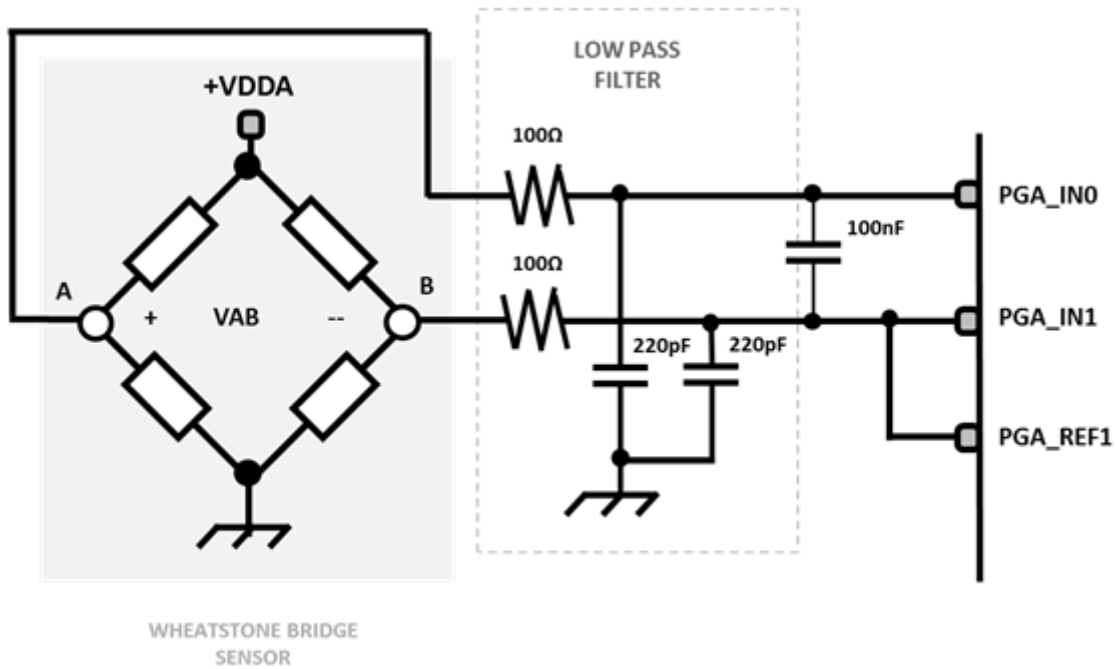


Figure 5. PGA circuit application.

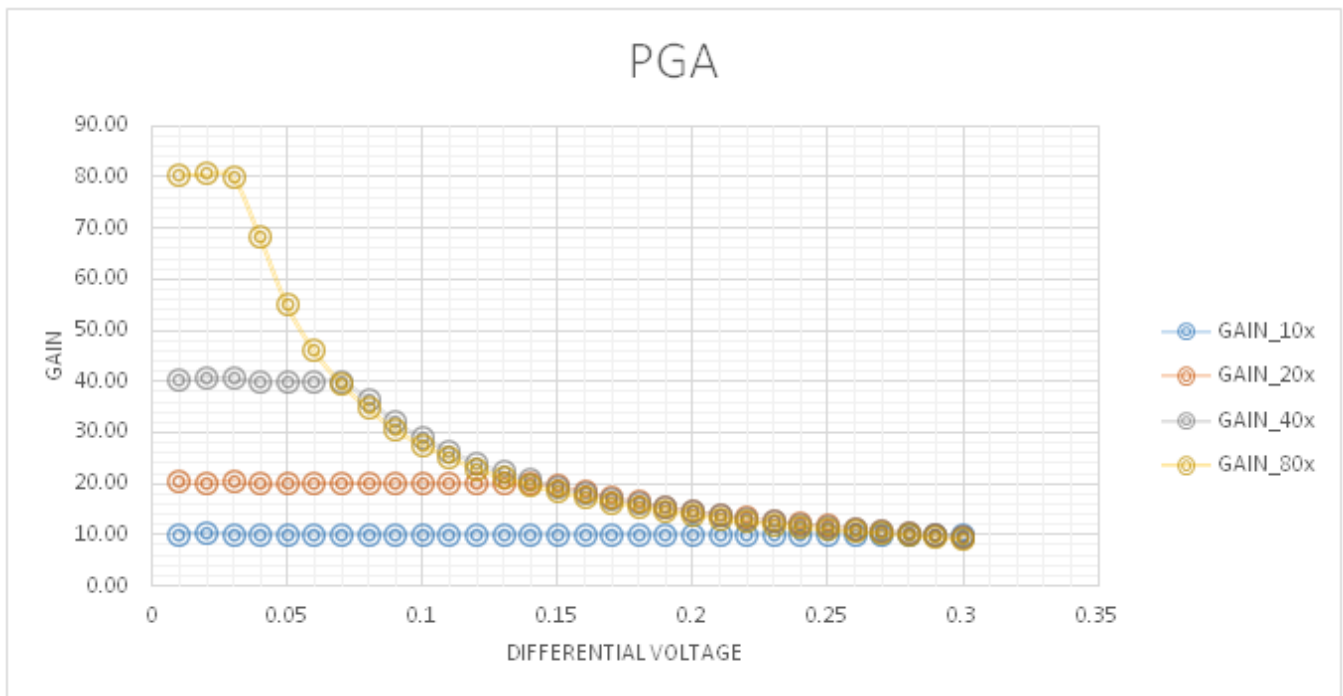


Figure 6. PGA Gain vs Differential voltage input

### 3 Example software

The tables in the sections below describes all functions and variables used for the PGA circuit application.

## 3.1 Global Variables

VARIABLE	DESCRIPTION
tPGA_setting	Indicates the settling time after the PGA channel configuration
n_samples	Indicates the samples number of the PGA reads in order to obtain the average

## 3.2 Functions

FUNCTION	DESCRIPTION
Clock_init();	Initializes REFCLK for PLL
ADC_init();	Initializes ADC configuration
ADC_read();	Read ADC channel
PGAOFFSET_COMP_ROUTINE();	Initializes the offset compensation routine.
PGAOUT_READ();	Obtain the PGAOUT value of a single input PGAIN0 or PGAIN1 and REF1 input as reference.
PGAOUT_READDIFF();	Obtain the PGAOUT value of a differential measurement between PGAIN0 and PGAIN1 inputs and REF1 input as reference.
delayx(tPGA_settling);	Delay time after the PGA channel configuration

- **void Clock\_init(unsigned char bus\_clk);**

<b>Name Function</b>	Clock_init();		
<b>Description</b>	Initializes REFCLK for PLL		
<b>Parameters</b>	bus_clk	<b>Name</b>	<b>Frequency(MHz)</b>
		CLK_8_MHZ	8
		CLK_12_5_MHZ	12.5
		CLK_24_MHZ	24
<b>Returns</b>	None		

- **void ADC\_init(void);**

<b>Name Function</b>	ADC_init();
<b>Description</b>	Initializes ADC configuration.
<b>Parameters</b>	None
<b>Returns</b>	None

- **unsigned int ADC\_read(unsigned int channel);**

**Example software**

<b>Name Function</b>	ADC_read();
<b>Description</b>	Read ADC channel
<b>Parameters</b>	For the PGA_OUT (internal ADC channel), the parameter of this function is indicated as follows: ADC_read (INT_AN (7));
<b>Returns</b>	Returns the reads of the (internal) ADC channel,

- unsigned int PGAOFFSET\_COMP\_ROUTINE(void);

<b>Name Function</b>	PGAOFFSET_COMP_ROUTINE();
<b>Description</b>	Initializes the offset compensation routine.
<b>Parameters</b>	None
<b>Returns</b>	Returns the offset compensation value.

- unsigned int PGAOUT\_READ(unsigned int pgainput,unsigned pgainx);

<b>Name Function</b>	PGAOUT_READ();		
<b>Description</b>	Obtain the PGAOUT value of a single input PGAIN0 or PGAIN1 and REF1 input as reference.		
<b>Parameters</b>	<b>pgainput</b>	<b>Name</b>	<b>Input</b>
		PGA_INPUT0	PGA_IN0
		PGA_INPUT1	PGA_IN1
	<b>pgainx</b>	<b>Name</b>	<b>Gain</b>
		GAIN_10x	10
		GAIN_20x	20
		GAIN_40x	40
		GAIN_80x	80
<b>Returns</b>			

- unsigned int PGAOUT\_READDIFF(unsigned int pgareference,unsigned pgainx);

<b>Name Function</b>	PGAOUT_READDIFF();		
<b>Description</b>	Obtain the average of the PGAOUT value. The differential measurement is determinates as PGAIN0 minus PGAIN1 inputs and REF1 input as reference.		
<b>Parameters</b>	<b>pgareference</b>	<b>Name</b>	<b>Reference</b>
		PGA_VREF_halfVDDA	VDDA/2
		PGA_VREF_0	PGA_REF0
		PGA_VREF_1	PGA_REF1
	<b>pgainx</b>	<b>Name</b>	<b>Gain</b>
		GAIN_10x	10
		GAIN_20x	20
		GAIN_40x	40
	GAIN_80x	80	
<b>Returns</b>			



- `void delayx(long int time);`

<b>Name Function</b>	<code>delayx(tPGA_settling);</code>
<b>Description</b>	Delay time after the PGA channel configuration.
<b>Parameters</b>	
<b>Returns</b>	None

## 4 Conclusion

This application note describes several potential uses for the on-chip PGA module on the S12ZVLA microcontrollers, also it highlights the importance of having very accurate resistors in the input filter if is need. For precise differential measurements, a 1 % resistor may be not enough. We also saw that a good precision of the GAIN that is useful for limiting the offset on the output voltage. The concepts can be expanded to meet other use cases as well. The CodeWarrior software project for these examples is available on [www.nxp.com](http://www.nxp.com).

**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).

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, and UMEMS are trademarks of NXP B.V. All other product or service names are the property of their respective owners. ARM, AMBA, ARM Powered, Artisan, Cortex, Jazelle, Keil, SecurCore, Thumb, TrustZone, and  $\mu$ Vision are registered trademarks of ARM Limited (or its subsidiaries) in the EU and/or elsewhere. ARM7, ARM9, ARM11, big.LITTLE, CoreLink, CoreSight, DesignStart, Mali, mbed, NEON, POP, Sensinode, Socrates, ULINK and Versatile are trademarks of ARM Limited (or its subsidiaries) in the EU and/or elsewhere. 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.

© 2016 NXP B.V.

Document Number AN5326  
Revision 0, 09/2016

