Freescale Semiconductor
Application Note

Document Number: AN4383
Rev. 0, 03/2012

PXS30 and PXS20 Safety Features

by: David Connelly and Curt Hillier
Austin, Texas

1 Introduction

The PXS30 and PXS20 are dual-core devices with built-in hardware features that allow them to achieve high safety integrity levels and reliability. These features include redundant logic, internal self-test, fault control handling, and redundancy checkers. Key components around the main processor core are redundant and referred to as the Sphere of Replication (SoR). These modules include the flash controller, interrupt controller, SRAM controller, DMA, crossbar, system timers, error correction status module, memory protection unit, peripheral bridge, software watchdog timer, semaphores, and the core itself. All the outputs from the SoR (addresses, data, and control signals) are checked by the redundancy control checker unit (RCCU) on every clock cycle, which can detect critical lockstep failure. Peripheral blocks outside the SoR used for data transfers may use a cyclic redundancy checker (CRC) that executes at-speed checksum calculations for data transfers. Error Correction (ECC) is featured on the RAM and flash memories, which will correct single bit

Contents

errors and detect double bit errors. Additionally, at every reset, Logical Built-In Self-Test (LBIST) is executed on the digital logic and Memory Built-In Self-Test (MBIST) is executed on internal memory. Any internal faults found during reset will be communicated to the internal Fault Collection and Control Unit (FCCU) for error handling. These devices are targeted to be included in a system that meets ISO26262 ASILD and IEC61508 SIL3 integrity levels.

2 Core logic and Sphere of Replication (SoR)

There are two run configurations available for dual-core architecture on the PXS30 and PXS20:

- Decoupled Parallel mode (DPM)—targets higher performance since each core operates independently. This is the default mode for the PXS30.
- Lock Step mode (LSM)—targets higher safety levels since both cores are in sync on every instruction cycle and critical faults around the SoR can be detected and handled properly. This is the default mode for the PXS20.

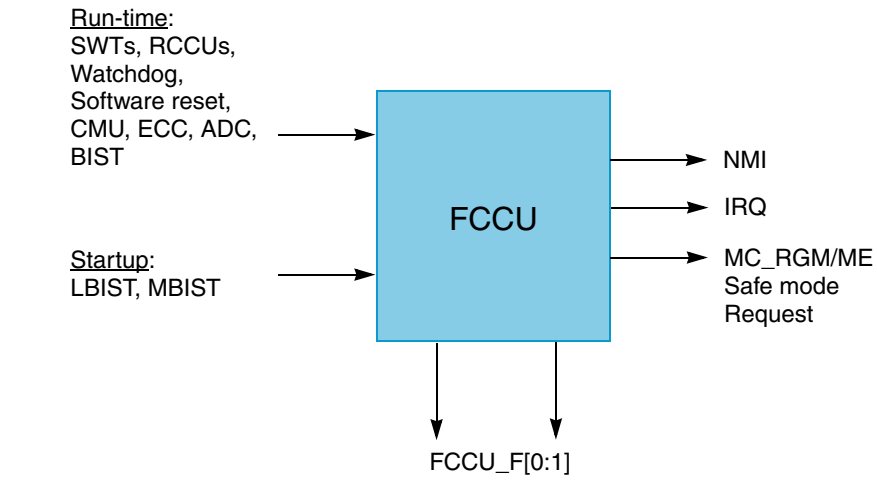
This document focuses on the safety features in LSM.

Fault detection for address, data, and control signals coming from the SoR is handled by the Redundancy Control Checker Unit (RCCU). The signals are checked on every clock cycle in LSM for safety critical applications, and the SoR is seen as one channel by the application. Faults are triggered if any difference in the output of the two cores is detected. The fault triggers a reaction which prevents propagation of the fault, and puts the microcontroller into a fail-safe mode. The RCCU itself is duplicated and covered by LBIST to guarantee proper functionality. The design is an application independent safety concept.

Non-replicated peripherals outside of the SoR that are used in safety applications and are not self-tested are recommended to be read or written twice by application software.

3 Fault detection using BIST, STCU, and FCCU

When the device is reset, the System Status and Control Module (SSCM) reads the boot configuration from shadow flash, and then passes control to the Self-Test Control Unit (STCU) for self testing. Both the MBIST and LBIST are executed before the MCU releases RESET. Any detected physical defects in the logic or memories are routed to the FCCU and errors are reported to the Fault Controller if necessary. MBIST, LBIST, and PBIST are not configurable to execute at any point outside of the startup sequence. The FCCU also handles all run-time faults from the various modules, as detailed in Figure 1.



Legend

SWT: Software Watchdog Timer	BIST: Built-In Self-Test
RCCU: Redundancy Control Checker Unit	NMI: Nonmaskable Interrupt
Watchdog: Core Watchdog	IRQ: Interrupt
CMU: Clock Monitor Unit	MC_RGM: Magic Carpet Reset Generation Module
ECC: Error Correction	ME: Mode Entry Module
ADC: Analog-to-Digital Converter	FCCU_F[0:1]: Fault Output Pins

Figure 1. Fault Collection and Control Unit signals

3.1 Built-In Self-Test coverage

The memory blocks outside of flash memory that are covered with MBIST include SRAM, cache, DMA, FlexCAN TX/RX buffers, FlexRay memory, ROM and Data Table, and Boot Assist Module (BAM). The flash memory has its own Built-In Self Test feature, which is executed by the user application code during runtime (not at reset). Here, it is referred to as the *array integrity check*. Please refer to the *PXS30 Microcontroller Reference Manual* for details on how to run array integrity check. Finally, the LBIST covers all the digital logic gates in the periphery surrounding the SoR. LBIST covers a total of three partitions: one partition each for Safety Lake 0 and Safety Lake 1, and a third partition for the remainder of the LBIST-covered IP modules. Each of the three partitions has its own BIST controller, in part to fulfill safety requirements and also to help prevent exceeding power limits. For a view of what is covered and not covered on the device with LBIST, refer to Figure 2.

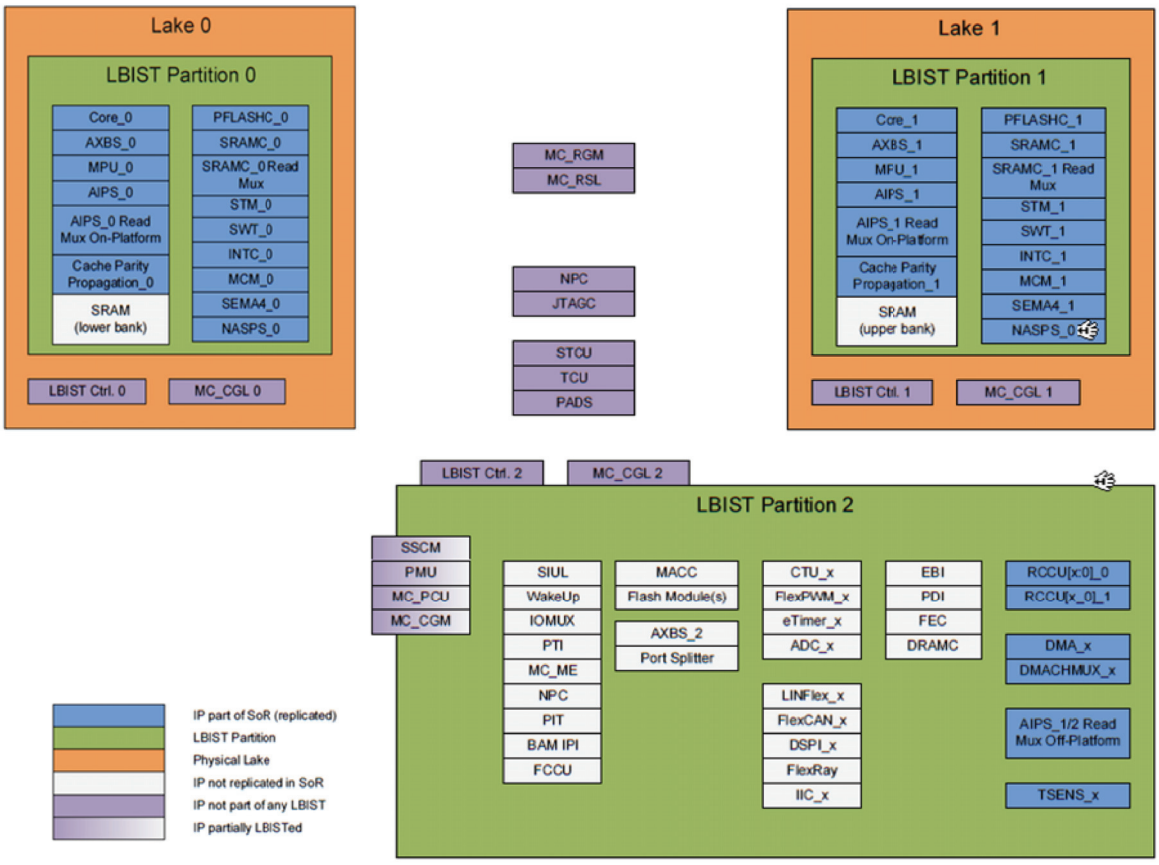


Figure 2. PXS30 replicated periphery, lake assignments, and logic BIST partitions

3.2 Reset

There are five phases (PHASE0–PHASE3, IDLE) during the reset sequence based on the type of reset requested. A power-on reset (POR) will start at PHASE0, which includes internal regulator and internal RC oscillator power-up. Other resets, such as “destructive,” “long” (that is, “functional”), or “short,” can be triggered based on the reset source. For example, assertion of the $\overline{\text{RESET}}$ signal externally (that is, pushing the reset button on the evaluation board) will trigger a “long” (or “functional”) reset starting at PHASE1, while a short reset stemming from a detected error may begin at PHASE3. Short resets are useful in cases where resetting the flash memory module is not required.

The STCU will activate on a long external or destructive reset, which includes power-on reset, low voltage detect, and mode entry destructive reset. It should be noted that the STCU can also be bypassed at reset depending on the reset configuration data located in the shadow flash memory block. For an example of STCU reset data configuration, refer to AN4389, *PXS30 Self Test Control Unit (STCU) Reset Configuration Data in Shadow Flash*. An overview of the reset flow can be seen in Figure 3.

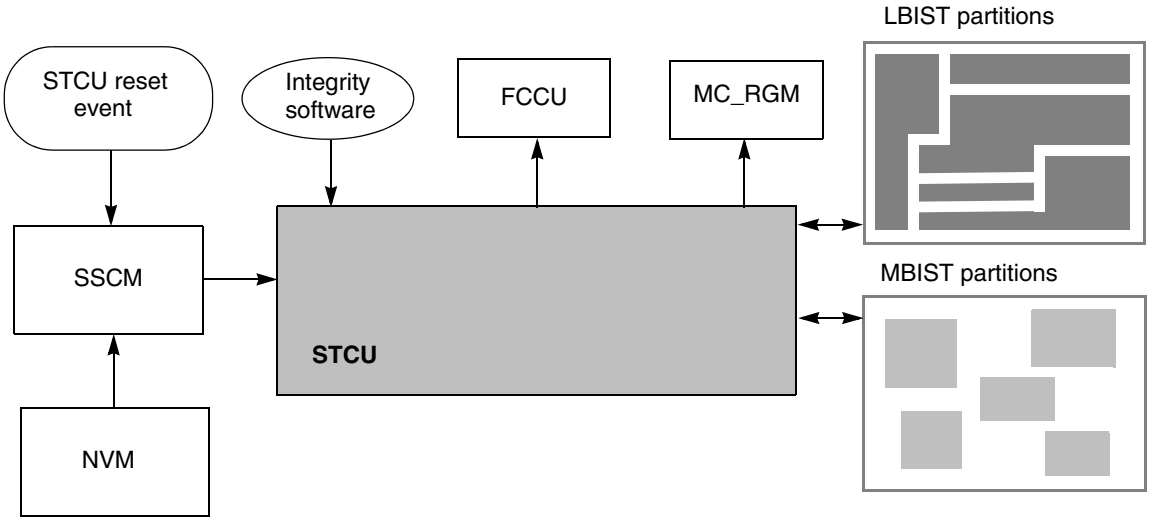


Figure 3. Detailed reset flow

1. After an STCU reset event, the SSCM detects that the device self-test has not been run yet.
2. The SSCM reads the self-test parameters from the flash nonvolatile memory (NVM).
3. The SSCM loads the self-test parameters into the STCU and passes control over to the STCU.
4. The STCU manages MBISTs and updates its internal status.
5. The STCU manages LBISTs and updates its internal status.
6. If faults are detected, the STCU reports the test failures to the FCCU.
7. The STCU signals the MC_RGM that the tests are complete, and the boot sequence proceeds to the next phase which passes control to the user applications software. However, if a fault is detected the STCU keeps the device in reset until the STCU Reset Event is applied.

NOTE

After self test has been run and no fault is detected, the SSCM will pass control to the CPUs where startup application software should check the STCU status information and LBIST MISR values to ensure that no errors were reported. This action can prevent possible faults within the STCU itself.

4 Application self-test features and error correction

ADC—The Analog-to-Digital Converter has the ability to run self-test (ABIST) while the device is in normal operating mode to verify that conversions are working correctly. The ABIST is configured and runs from internal memory-mapped registers; faults are routed to the FCCU for error handling. The user has the option to run an internal test on the ADC module within the safety process window (~10 ms) if it is required to meet the safety requirements of the system. There are values stored in the test flash block that represent the upper and lower threshold of conversions on internal channels. These values must be copied to STAWxR registers and referenced to check the results of the self-test. Different algorithms contained within the self-test features cover the internal bandgap, reference, and supply voltages, as well as the

internal analog components of the ADC (for example, resistive and capacitive Digital-to-Analog Converters). While the ADC is running in a user application, an internal watchdog can be implemented in order to detect whether a converted value is outside of a guarded range and then, optionally, whether to generate an interrupt to the CPU. Three internal ADC channels are dedicated to monitoring main PMU power (+5V or +3.3V), VDD_LV_COR supply (+1.2V), and the on-board temperature sensor.

Flash—A feature called array integrity self check (AIC) can be executed with user application code to verify the flash contents and ensure the surrounding digital logic is not faulty. AIC can be executed at any time by the user application code for safety critical applications. Additionally, flash blocks also have error correction and detection circuitry (ECC) which protects against single-bit errors and flags double-bit errors. The ECC circuitry itself can be tested for functionality using internal ECC logic testing. This feature will validate that correct data is not accidently modified and that single-bit errors are correctly modified.

SRAM—Internal SRAM has built-in ECC error detection that can be enabled through the Error Correction Status Module (ECSM). Outside of the MBIST, which runs at reset, no other internal self-test features exist for the SRAM memory block.

Clocks—Clock monitors have the ability to detect and mitigate clock disturbances. Three Clock Monitor Units (CMU) are available; they monitor the system clock, motor control clock, and the FlexRay clock. Each monitor has configuration registers that can be set to detect upper and lower frequency limits on each respective clock source. The default state of the monitors is disabled out of reset, as a valid operating frequency needs to be set up prior to enabling the CMU.

Power—There are low voltage detect (LVD) monitors for the 1.2 V core and the 3.3 V flash, I/O, and ADC supplies. In addition, there is a high voltage detect (HVD) monitor for the 1.2 V core power. Comparators are used to ensure that the voltages are operating in the correct range. The voltage level at which the LVD or HVD error is triggered is configured through trim bits in the Power Management Unit (PMU). LVD/HVD detection is configurable and can be disabled if necessary.

Cyclic Redundancy Checker (CRC) Unit—The CRC offloads the CPU with checksum computations that are used to generate an output signature for data transfers. It can be used for memory-to-memory, peripheral-to-memory, and memory-to-peripheral transfers using the CPU or the DMA. The CRC does not have any external signals. It can also be used to validate important internal data such as configuration registers for certain modules.

Enhanced Motor Control Timer (eTimer)—Some of the output channel pairs of the eTimer module can be configured in a redundant manner, which allows output error checking. If there is a mismatch on the output of the two channels, an error flag will be triggered and cause an interrupt to occur. If a mismatch occurs, the two redundant output channels are put in their inactive state. An optional watchdog function can also be enabled.

Fault Control and Collection Unit (FCCU)—The FCCU is the central point in the system that collects faults and controls the device reaction without CPU intervention. There are two configurable bidirectional FCCU pins that typically show a “non-faulty” state but that will change their signature once a fault is detected. The FCCU has its own redundancy checker units that are completely separate from the RCCU logic of the SoR. The internal redundancy control of the FCCU is used to guarantee that signals sent to the interrupt controller and the reset generation module are valid. The FCCU has configuration fields for each critical and noncritical fault. The configuration allows the microcontroller to be put into a fail-safe state

that includes reset, shut down, or signaling on the error out signals. Each one of these faults can be injected (simulated) in the system using either the critical or noncritical fake fault registers. This allows a specific fail mode and reaction to be tested and debugged on the system level. All critical faults are software-recoverable by default and most will trigger a long function reset. These can be configured to be hardware-recoverable, short reset, or no reset action taken. Noncritical faults are also software-recoverable by default and will trigger a long functional reset or have no reaction, depending on the configuration. For example:

```
/*******************************************************/
/* FCCU needs to be in CONFIG state to change settings */
/*******************************************************/
FCCU.CFG_TO.R = 0x00000006; /* CONFIG STATE Time Out: 4096 micro sec */
FCCU.CTRLK.R = FCCU_CTRLK_OP1;/* Key for the operation OP1 */
FCCU.CTRL.R = FCCU_CTRL_OPR1; /* Set the FCCU into the CONFIG state [OP1] */
while(FCCU.CTRL.B.OPS != FCCU_CTRL_OPS3){;}       /* wait for the completion of the operation */

/*******************************************************/
/* Configure FCCU for Critical Faults */
/*******************************************************/
FCCU.CFG.R = 0x00000E38;/*fast switching, dual rail for fccu[0:1] output pins */
FCCU.CF_CFG0.R = 0xFFFFFFFF;/* software recoverable */
FCCU.CF_CFG1.R = 0x000000FF;
FCCU.CFS_CFG[0].R = 0xAAAAAAAA;/* long functional reset */
FCCU.CFS_CFG[1].R = 0xAAAA800A;/* long or no reset depending on fault */
FCCU.CFS_CFG[2].R = 0x0000082A;/* long functional reset */

/*******************************************************/
/* Configure FCCU for Non-Critical Faults */
/*******************************************************/
FCCU.NCF_CFG.R = 0xFFFFFFFF;/* software recoverable */
FCCU.NCFS_CFG[0].R = 0x0000AAAA;/* long functional or no reset reaction */
FCCU.NCFS_CFG[1].R = 0x00000000;/* no reset reaction */
FCCU.NCFE.R = 0x06D01CFF;/* enable or disable non-critical faults */
FCCU.NCFTOE.R = 0x06381FFF;/* timeout enabled or disabled for non-critical faults */
FCCU.NCF_TO.R = 0x0000FFFF;/* non critical fault timeout */

/*******************************************************/
/* FCCU back to NORMAL state (out of CONFIG state) */
/*******************************************************/
FCCU.CTRLK.R = FCCU_CTRLK_OP2;  /* Key for the operation OP2 */
FCCU.CTRL.R = FCCU_CTRL_OPR2; /* Set the FCCU into the NORMAL state [OP2] */
while(FCCU.CTRL.B.OPS != FCCU_CTRL_OPS3){;}/* wait for the completion of the operation */

/* Set Fault Here */
FCCU.CFF.R = fault_id;/* eg: fault_id = 0 for source 0 RCCU[0] critical fault */
/* OR */
FCCU.NCFF.R = fault_id;/* eg: fault_id = 0 for core watchdog non-critical fault */
/*******************************************************************************/
/* Here, if rest is configured, code execution will begin at the code entry point */
/*******************************************************************************/



/*******************************/
```

```
/* Flow to handle a fault */
/*******************************/

if (RGM.FES.B.F_FCCU_SAFE || RGM.FES.B.F_FCCU_HARD)

{
        /* Flag first - this is optional */
        if (RGM.FES.B.F_FCCU_SAFE) safe_mode = TRUE;
        if (RGM.FES.B.F_FCCU_HARD) hard_fault = TRUE;

}


if (RGM.FES.B.F_FCCU_SAFE == 0x1)  //functional event status safe mode
{
        if (ME.GS.B.S_CURRENTMODE == SAFE_MODE)//global status
        {
                RGM.FES.B.F_FCCU_SAFE = 0x1;  // Clear Safe Request
                SetMode(DRUN_MODE);        // Attempt to enter DRUN mode
        }
        /* Enable peripherals */
        ModeInit();
}

ME.IMTS.B.S_SEA = 1;/* clear invalid mode transition */
clear_CF();
clear_NCF();
RGM.FES.R = 0xFFFF;/* clear functional event status register */
RGM.DES.R = 0xFFFF;/* clear destructive event status register */


/*****************************************************************
 *****************************************************************
        Functions for reference below
/*****************************************************************
 *****************************************************************


/*****************************************************************
* FUNCTION : clear_NCF
* DESCRIPTION : clear critical fault status register
* INPUTS : None
* OUTPUTS : None
*****************************************************************/
void clear_NCF()
{
        int b;

        FCCU.NCFK.R = FCCU_NCFK_KEY;
        FCCU.NCFS0.R = 0xFFFFFFFF;
        while(FCCU.CTRL.B.OPS != 0x3){}  /* wait for the completion of the operation */
        b = FCCU.NCFS0.R;
}


/*****************************************************************
* FUNCTION : clear_CF
* DESCRIPTION : clear critical fault status register
```

```
* INPUTS : None
* OUTPUTS : None
*************************************************************/
void clear_CF()
{
        int i, a[i];

        for(i=0; i<2; i++)
        {
                FCCU.CFK.R = FCCU_CFK_KEY;
                FCCU.CFS[i].R = 0xFFFFFFFF;
                while(FCCU.CTRL.B.OPS != 0x3){} /* wait for the completion of the
                                                   operation */
                a[i]=FCCU.CFS[i].R;
        }
   }
}

/*************************************************************
* FUNCTION : ModeInit
* DESCRIPTION : Initialize the Mode Entry Module
* INPUTS : None
* OUTPUTS : None
*************************************************************/
void ModeInit(void)
{
        uint32_t i;

        /* Enable of all operation modes */
        //ME.MER.R = 0x000005FF;

        /* Setting RUN Configuration Register ME_RUN_PC */
        for(i = 0; i <= 7; i++)
        {
                ME.RUNPC[i].R=0x000000FE; /* Peripheral ON in every mode */
        }

        SetMode(DRUN_MODE);
}

/*************************************************************
* FUNCTION : SET_MODE
* DESCRIPTION : Sets cpu run mode and waits for completion
* INPUTS : mode - required run mode
* OUTPUTS : None
*************************************************************/
void SetMode(uint8_t mode)
{
        uint32_t retry = 0, timeout=0x2FF;

        do
        {
                ME.MCTL.R = (mode << 28 | 0x00005AF0u);         /* Mode & Key */
                ME.MCTL.R = (mode << 28 | 0x0000A50Fu);         /* Mode & Key */
                while((ME.GS.B.S_MTRANS == 1) && (timeout--)){} /* Wait for mode entry to
complete */
                retry++;
```

```
        } while ((ME.GS.B.S_CURRENTMODE != mode) && (retry < 1000)); /* Check DRUN mode has
been entered */
}
```

5 Conclusion

In addition to the safety elements built into the device, there are recommended external measures to be taken to ensure a high safety integrity level of the system. External power supply monitors can be implemented, in addition to external watchdog timers and error out monitors for fault conditions. The $\overline{\text{RESET}}$ signal is a bi-directional pin on the PXS30 that can be used to drive the reset on external circuitry. In summary, the dual-core redundant hardware, built-in self-test, clock/voltage monitoring, and error correction of the PXS devices allow high safety integrity levels be achieved in the target system.

THIS PAGE IS INTENTIONALLY BLANK

How to Reach Us:

Home Page:
www.freescale.com

Web Support:
http://www.freescale.com/support

USA/Europe or Locations Not Listed:
Freescale Semiconductor, Inc.
Technical Information Center, EL516
2100 East Elliot Road
Tempe, Arizona 85284
+1-800-521-6274 or +1-480-768-2130
www.freescale.com/support

Europe, Middle East, and Africa:
Freescale Halbleiter Deutschland GmbH
Technical Information Center
Schatzbogen 7
81829 Muenchen, Germany
+44 1296 380 456 (English)
+46 8 52200080 (English)
+49 89 92103 559 (German)
+33 1 69 35 48 48 (French)
www.freescale.com/support

Japan:
Freescale Semiconductor Japan Ltd.
Headquarters
ARCO Tower 15F
1-8-1, Shimo-Meguro, Meguro-ku,
Tokyo 153-0064
Japan
0120 191014 or +81 3 5437 9125
support.japan@freescale.com

Asia/Pacific:
Freescale Semiconductor China Ltd.
Exchange Building 23F
No. 118 Jianguo Road
Chaoyang District
Beijing 100022
China
+86 10 5879 8000
support.asia@freescale.com

For Literature Requests Only:
Freescale Semiconductor Literature Distribution Center
1-800-441-2447 or 303-675-2140
Fax: 303-675-2150
LDCForFreescaleSemiconductor@hibbertgroup.com

Document Number: AN4383
Rev. 0
03/2012