Light Control Module Reference
Design based on Kinetis EA MCU and
Quad High-side Switch
(MC10XS3425)
## Contents

<table>
<thead>
<tr>
<th>Section number</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Chapter 1</td>
<td></td>
</tr>
<tr>
<td>1.1</td>
<td>Introduction</td>
<td>5</td>
</tr>
<tr>
<td>1.1</td>
<td>Introduction</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>Chapter 2</td>
<td></td>
</tr>
<tr>
<td>2.1</td>
<td>Kinetis EA</td>
<td>7</td>
</tr>
<tr>
<td>2.1</td>
<td>Kinetis EA</td>
<td>7</td>
</tr>
<tr>
<td></td>
<td>Chapter 3</td>
<td></td>
</tr>
<tr>
<td>3.1</td>
<td>System concept</td>
<td>9</td>
</tr>
<tr>
<td>3.2</td>
<td>Board diagram</td>
<td>11</td>
</tr>
<tr>
<td>3.3</td>
<td>Quad high-side switch</td>
<td>11</td>
</tr>
<tr>
<td>3.3.1</td>
<td>Introduction</td>
<td>11</td>
</tr>
<tr>
<td>3.3.2</td>
<td>MC10XS3425 device features</td>
<td>12</td>
</tr>
<tr>
<td>3.3.3</td>
<td>SPI protocol description</td>
<td>12</td>
</tr>
<tr>
<td>3.3.4</td>
<td>Logic, commands and registers</td>
<td>13</td>
</tr>
<tr>
<td>3.3.4.1</td>
<td>Serial input communication</td>
<td>13</td>
</tr>
<tr>
<td>3.3.4.2</td>
<td>Serial Output Communication (Device Status Return Data)</td>
<td>14</td>
</tr>
<tr>
<td></td>
<td>Chapter 4</td>
<td></td>
</tr>
<tr>
<td>4.1</td>
<td>Software description</td>
<td>17</td>
</tr>
<tr>
<td>4.2</td>
<td>Functions description</td>
<td>18</td>
</tr>
<tr>
<td>4.3</td>
<td>Software logic flow</td>
<td>25</td>
</tr>
<tr>
<td></td>
<td>Chapter 5</td>
<td></td>
</tr>
<tr>
<td>5.1</td>
<td>FreeMASTER tool</td>
<td>27</td>
</tr>
<tr>
<td>5.2</td>
<td>FreeMASTER graphical user interface</td>
<td>27</td>
</tr>
</tbody>
</table>
Chapter 1
Introduction

1.1 Introduction

This document is used to introduce the technical implementation for Light Control Module Reference Design (LCM).

The LCM system uses Kinetis EA series Microcontroller (MCU) to control and diagnose up to four automotive lamps, using a quad high-side driver eswitch. Control signals are mainly from switches; and a potentiometer to control the brightness bulbs. The LCM system also includes a user graphical interface using FreeMASTER. This user interface controls the light intensity of the lamps connected to the light output connector, turns the lamps on/off/toggles them, and it is also used to check faults in the system.
Chapter 2
Kinetis EA Series

2.1 Kinetis EA

The Kinetis EA Series MCUs for Automotive is a highly scalable portfolio of 32-bit ARM® Cortex®-M0 based processors and MCUs aimed for the automotive markets. The family is optimized for cost-sensitive applications offering low pin-count option with very low power consumption. With 2.7-5.5 V supply and focus on exceptional EMC/ESD robustness, Kinetis EA series devices are well suited to a wide range of applications ranging from body applications to generic sensor nodes. In automotive body application, the Kinetis EA Series MCUs for Automotive is a great option for entry level body controller or gateway module, window/roof/sun-roof controller, immobilizer or seat/mirror controller just to mention a few.

Many modules are available on KEA128 including ARM Cortex-M0+ core based processors, system module such as SIM/PMC/WDOG/AIPS/BME/MCM, memories such as up to 128 KB flash memory and up to 8 KB SRAM, Clocks, ADC, Timers such as RTC/PIT/FTM, communication modules such as UART/SPI/I2C/CAN and HMI module.

LCM system uses SPI to communicate with E-switch, the ADC module to sample the signal from the potentiometer, and the KBI module to read button states. It also includes LIN and CAN transceivers for Automotive Network communications.
Chapter 3
Hardware Design

3.1 System concept
The LCM system is designed to control up to 4 lamps using the KEA128 and a quad high-side driver eswitch (MC10XS3425). The system board has the following features:

- ARM Cortex-M0 based Kinetis KEA128 and automotive microcontroller in 64 LQFP package.
- MC10XS3425EK Quad High-side Switch.
- Output Connector to connect up to 4 Lamps.
- MC33662 LIN transceiver.
- MC33901 CAN transceiver.
- 4 User LEDs
- 4 Potentiometers
- 4 User buttons
- SWD connector interface for debugging and communication with FreeMASTER.
Figure 3-1. System block diagram
3.2 Board diagram

![Board diagram](image)

Figure 3-2. Board diagram

3.3 Quad high-side switch

3.3.1 Introduction

The 10XS3425 is one in a family of devices designed for low-voltage automotive lighting applications. Its four low RDS(on) MOSFETs (dual 10 mΩ/dual 25 mΩ) can control four separate 55 W / 28 W bulbs, and/or Xenon modules, and/or LEDs.

Programming, control, and diagnostics are accomplished using a 16-bit SPI interface. Its output with selectable slew-rate improves electromagnetic compatibility (EMC) behavior. Additionally, each output has its own parallel input or SPI control for pulse-width modulation (PWM) control if desired. The 10XS3425 allows the user to program via the SPI the fault current trip levels and duration of acceptable lamp inrush.
The four channels can be controlled individually by external/internal clock-signals or by direct inputs. Using the internal clock allows fully autonomous device operation. Programmable output voltage slew rates (individually programmable) helps improve EMC performance. To avoid shutting off the device upon inrush current, while still being able to closely track the load current, a dynamic overcurrent threshold profile is featured. Switching current of each channel can be sensed with a programmable sensing ratio. Whenever communication with the external microcontroller is lost, the device enters a fail-safe operation mode, but remains operational, controllable, and protected.

### 3.3.2 MC10XS3425 device features

- Four protected 10 mΩ and 25 mΩ high side switches
- Operating voltage range of 6.0 to 20 V with sleep current < 5.0 μA, extended mode from 4.0 to 28 V
- 8.0 MHz 16-bit 3.3 V and 5.0 V SPI control and status reporting with daisy chain capability
- PWM module using external clock or calibratable internal oscillator with programmable outputs delay management
- Smart overcurrent shutdown, severe short-circuit, over-temperature protections with time limited auto-retry, and fail-safe mode in case of MCU damage
- Output OFF or ON open-load detection compliant to bulbs or LEDs and short-to-battery detection. Analog current feedback with selectable ratio and board temperature feedback.

Freescale analog ICs are manufactured using the SMARTMOS process, a combinational BiCMOS manufacturing flow that integrates precision analog, power functions and dense CMOS logic together on a single cost-effective die.

### 3.3.3 SPI protocol description

The SPI interface has a full duplex, three-wire synchronous data transfer with four I/O lines associated with it: Serial Input (SI), Serial Output (SO), Serial Clock (SCLK), and Chip Select (CSB).

The SI/SO pins of the MC10XS3425 follow a first-in first-out (D15 to D0) protocol, with both input and output words transferring the most significant bit (MSB) first. All inputs are compatible with 5.0 or 3.3 V CMOS logic levels.
3.3.4 Logic, commands and registers

3.3.4.1 Serial input communication

To communicate to the MC 10XS3425 device it is necessary to send a 16-bit command using SPI. A message is transmitted by the KEA128 starting with the MSB D15 and ending with the LSB, D0. MC 10XS3425 commands are listed in the table below.

Table 3-1. Serial input commands

<table>
<thead>
<tr>
<th>SI Register</th>
<th>SI Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>D15</td>
<td>D14</td>
</tr>
<tr>
<td>STATR_s</td>
<td>WIN</td>
</tr>
<tr>
<td>PWM_s</td>
<td>WIN</td>
</tr>
<tr>
<td>CONFR0_s</td>
<td>WIN</td>
</tr>
<tr>
<td>CONFR1_s</td>
<td>WIN</td>
</tr>
<tr>
<td>OCR_s</td>
<td>WIN</td>
</tr>
</tbody>
</table>

Table continues on the next page...
Table 3-1. Serial input commands (continued)

<table>
<thead>
<tr>
<th>SI Register</th>
<th>SI Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>GCR WIN D</td>
<td>0 0 1 0 1 0 VDD_FAIL_en PWM CLOCK_s el TEM P_en CSN S_en CSN S1 CSN S0 X OV_d is</td>
</tr>
<tr>
<td>CALR WIN D</td>
<td>0 0 1 1 0 1 0 1 0 1 1</td>
</tr>
<tr>
<td>Register state after RST=0 or V_VDD or V_SUPPLY(P OR) Condition</td>
<td>0 0 0 X X 0 0 0 0 0 0 0 0 0</td>
</tr>
</tbody>
</table>

1. x=Don't care

The first column is the name of registers, and registers ending with a 's' means there are four of them each representing one channel. The table below shows bit assignment of the command.

Table 3-2. Serial input command bit assignment

<table>
<thead>
<tr>
<th>Bit significance</th>
<th>SI msg bit</th>
<th>Message bit description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MSB</td>
<td>D15</td>
<td>Watchdog in: toggles to satisfy Watchdog requirements.</td>
</tr>
<tr>
<td></td>
<td>D14:D13</td>
<td>Register address bits used in some cases for input selection.</td>
</tr>
<tr>
<td></td>
<td>D12:D10</td>
<td>Register address bits.</td>
</tr>
<tr>
<td></td>
<td>D9</td>
<td>Not used (set logic [0])</td>
</tr>
<tr>
<td>LSB</td>
<td>D8:D0</td>
<td>Used to configure the inputs, outputs, and the device protection feature and SO status content.</td>
</tr>
</tbody>
</table>

3.3.4.2 Serial Output Communication (Device Status Return Data)

SO data will represent information ranging from fault status to register contents, user selected by writing to the STATR bits OD4, OD3, OD2, OD1, and OD0. The value of the previous bits SOA4 and SOA3 will determine which output the SO information applies to for the registers, which are output specific, that is, Fault, PWMR, CONFR0, CONFR1, and OCR registers.
Note that the returned status data is based on the previous STATR_S command, so if a specific register needs to be read, it is needed to send two STATR_S commands. The first command is to tell the MC 10XS3425 which register is requested, and the second command is to retrieve the requested data. The returned data format is shown in the following table.

**Table 3-3. Serial out status data**

<table>
<thead>
<tr>
<th>Previous STATR</th>
<th>SO Returned Data</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>SO A4 SO A3 SO A2 SO A1 SO A0</td>
</tr>
<tr>
<td>STA TR_s</td>
<td>A1 A0 0 0 0 WDI N SO A4 SO A3 SO A2 SO A1 SO A0 NM POR UV OV OL ON <em>s OL OFF <em>s OS</em> s OT</em> s SC_ s OC_ s</td>
</tr>
<tr>
<td>PWM R_s</td>
<td>A1 A0 0 0 1 WDI N SO A4 SO A3 SO A2 SO A1 SO A0 NM 28 W_ s ON <em>s PW M6</em> s PW M5_ s PW M4_ s PW M3_ s PW M2_ s PW M1_ s PW M0_ s</td>
</tr>
<tr>
<td>CON FR0_ s</td>
<td>A1 A0 0 1 0 WDI N SO A4 SO A3 SO A2 SO A1 SO A0 NM X X X X DIR _dis <em>s Retr y_u nlimited <em>s PW M1</em> s PW M0</em> s</td>
</tr>
<tr>
<td>CON FR1_ s</td>
<td>A1 A0 0 1 1 WDI N SO A4 SO A3 SO A2 SO A1 SO A0 NM X X X X DIR _dis <em>s Retr y_u nlimited <em>s PW M1</em> s PW M0</em> s</td>
</tr>
<tr>
<td>OCR _ s</td>
<td>A1 A0 1 0 0 WDI N SO A4 SO A3 SO A2 SO A1 SO A0 NM Xen on_ s BC1_ s BC O_s OC 0_ s OC Hl_ s OC L01_ s OC L00_ s OC <em>mode</em> s OC disp</td>
</tr>
<tr>
<td>GCR</td>
<td>0 0 1 0 1 WDI N SO A4 SO A3 SO A2 SO A1 SO A0 NM VD D FAIL on_ s PW M_e n CL OC K_s el TE MP_en CS NS_ en CS NS1 CS NS0 X X X X X X X X X X X X X X X X X X CL OC K_f ail CAL _fail OT _W</td>
</tr>
<tr>
<td>DIAG R0</td>
<td>0 0 1 1 1 WDI N SO A4 SO A3 SO A2 SO A1 SO A0 NM X X X X X X X X X X X X X X X X X X CLI OC K_f ail CAL _fail OT _W</td>
</tr>
<tr>
<td>DIAG R1</td>
<td>0 1 1 1 1 WDI N SO A4 SO A3 SO A2 SO A1 SO A0 NM X X X X IN3 IN2 IN1 IN0 WD _en</td>
</tr>
<tr>
<td>DIAG R2</td>
<td>1 0 1 1 1 WDI N SO A4 SO A3 SO A2 SO A1 SO A0 NM X X X X X X X X X X 0 0 0 0 0 0 0 0 0</td>
</tr>
</tbody>
</table>
The 10XS3425 enters in Normal mode after start-up if following sequence is provided:

- VPWR and VDD power supplies must be above their under voltage thresholds,
- generate wake-up event (wake-up = 1) from 0 to 1 on RSTB. The device switches to Normal mode with the SPI register content reset (as defined in Table 3-1 and Table 3-3). All features of the 10XS3425 will be available after 50 µs (typical), and all SPI registers are set to default values (set to logic [0]).
- Toggle WD bit from 0 to 1. And, in case the PWM module is used (PWM_en bit is set to logic [1]) with an external reference clock.
- Apply the PWM clock on the IN0 input pin after a maximum of 200 µs (min. 50 µs).

If the correct start-up sequence is not provided, the PWM function is not guaranteed.
Chapter 4
Software

4.1 Software description

This software demonstrates the functionality of the KEA128LEDLIGHTING board.

The software communicates the KEA128 with the MC10XS3425 eswitch via SPI, to control the duty cycle of the eswitch outputs, and to receive faults that occur in the system. SPI0 is configured in master mode, with automatic chip select, CPHA =1 and CPOL =0. SPI0 is set to work on KEA128 pins PTE0, PTE1, PTE2, and PTE3.

Faults detected by eswitch will be reported to the KEA128 via SPI. The KEA128 shows the faults in the system via the FreeMASTER interface.

In case of an under voltage fault, the system will turn off. In the next time, when the system is on, the LED on PTB5 will turn on, indicating that an under voltage fault has occurred and the outputs will be off. To clear the under voltage fault to return the system to its normal operation, press any SW2-5 or click in "Clear Under Voltage Fault" button in FreeMASTER.

NOTE

When an under voltage fault occurred, communication with the board and FreeMASTER will be lost. To reestablish communication, stop the FreeMASTER communication and start it again once the board has recovered from the under voltage condition.

After the under voltage fault is removed, the LED on PTB5 will turn off and the system will return to its normal operation.

Potentiometers are connected to ADC channels 12-15 to update the PWM duty cycle of the eswitch outputs. Each potentiometer modifies one duty cycle of the eswitch output. ADC is configured with 8-bit resolution and uses software trigger to read the voltage from the potentiometers.
Based on the ADC value read by each potentiometer the duty cycle value will be send to the eswitch via an SPI command, and the duty cycle will be updated. The duty cycle of the outputs can be also changed with the sliders in the FreeMASTER interface.

Board switches, SW2-5, are connected to KBI 1 channels 16-19 of the KEA128. Each switch controls one output state. KBI is configured to enable channels 16-19 and to generate an interrupt on the falling edge.

Every time that one of these switches is pressed an interrupt will be generated. When a switch is pressed the corresponding output will change their state to toggle, turn ON or turn OFF the lamp. This output state can be also modified using the "Output Control" button in the FreeMASTER interface.

The RTC is configured to generate an interrupt every 500 ms. This interrupt determines the period at which the outputs will toggle in case they are in the toggle state. Also an LED on PTB2 will toggle at this period to indicate that the KEA128 is working.

### 4.2 Functions description

Functions inside the software are divided by Kinetis EA peripherals, MC10XS3425 driver library, and lighting application functions.

The Kinetis EA peripherals functions are in the following files: ADC.c, CLK.c, GPIO.c, KBI.c, RTC.c, and SPI.c. These files contain functions to initialize and use each peripheral. Below is a description of the peripheral functions.

**ADC**

**ADC_Init(uint8_t mode)**. Initialize ADC module. ADC is configured in continuous mode operation, trigger by software and bus clock selected for ADC.

Received parameter: ADC desired resolution, 8-,10-,12-bit mode.

Return value: none.

**uint16_t ADC_Read(uint8_t channel)**. Reads ADC selected channel.

Received parameter: ADC channel to read.

Return value: ADC conversion value.

**Enable_ADC_CH(channel)**. This macro enables desired ADC channel.

Received parameter: ADC channel to enable.

Return value: none.
CLK.

**Clk_Init().** Initialize the clocks to run at 20 MHz from the 8 MHz external XTAL

Received parameter: none

Return value: none

**GPIO**

**GPIO_Init().** Initialize pin PTB2-5 as outputs.

Received parameter: none

Return value: none

**KBI**

**KBI_Init().** Initialize KBI 1 module. Enables KBI channels 16, 17, 18, and 19. Polarity set to falling edge

Received parameter: none.

Return value: none.

**KBI_SetCallback(pt2FuncU8 ptr).** Set a callback function to execute on a falling edge of an enabled KBI pin

Received parameter: pointer to function with an UINT8 argument.

Return value: none

**KBI_IRQHandler().** KBI interrupt routine, calls the user callback.

Received parameter: none.

Return value: none.

**RTC.**

**RTC_Init().** Initialize RTC counter to generate an interrupt every 500 ms.

Received parameter: none.

Return value: none.

**RTC_SetCallback (pt2FuncU8 ptr).** Set the callback function to call on successful matches from any channel

Received parameter: pointer to function with an UINT8 argument.

Return value: none.
RTC_IRQHandler(). RTC interrupt routine, calls the user callback.

Received parameter: none.

Return value: none.

SPI.

SPI_Init(). Configures SPI0 on pins PTE0, PTE1, PTE2, and PTE3. SPI is configured in master mode, with automatic chip select, CPHA =1 and CPOL =0.

Received parameter: none

Return value: none

uint8_t SPI_Read(). Reads SPI data once the buffer is full.

Received parameter: none

Return value: SPI received data.

SPI_Write(uint8_t data). Send SPI data once buffer is empty.

Received parameter: data to be send via SPI

Return value: none.

The MC10XS3425 library is located in the eswitch.c file. This field contains the functions related to initialize, reset, disable PWM, enable PWM, send commands, read registers status, turn on and turn off switch outputs. Below is a description of the functions included in the switch file.

eswitch.

reset_es(). Reset eswitch device.

Received parameter: ECMD structure.

Return value: none.

eswitch_init(). Initialize eswitch. Calls SPI_Init(), to set up an SPI communication of 9600 baud rate. Calls pwm_enable() to enable PWM in eswitch.

Received parameter: ECMD structure.

Return value: none.

send_cmd(ECMD _ecmd). Send command to eswitch.

Received parameter: ECMD structure.

Return value: none.
ECMD read_status(ECMD _ecmd). Read eswitch status.
Received parameter: ECMD structure.
Return value: ECMD structure.

pwm_enable(). Enable PWM in eswitch.
Received parameter: none.
Return value: none.

pwm_disable(). Disable PWM in eswitch
Received parameter: none.
Return value: none.

light_on(uint8_t ch). Turn on eswitch channel.
Received parameter: channel to turn on as a parameter. Ch value must be between 0-3.
Return value: none.

light_off(uint8_t ch). Turn off eswitch channel.
Received parameter: channel to turn on as a parameter. Ch value must be between 0-3.
Return value: none.

light_pwm_on(uint8_t ch, uint8_t lvl). Turn on using PWM in eswitch channel.
Received parameter: receives eswthic channel to turn on, duty cycle of the PWM as parameters. Ch value must be between 0-3. Lvl value must be between 0-128.
Return value: None

ECMD read_statr(uint8_t ch). Read status register from switch.
Received parameter: eswitch channel to read status register
Return value: ECMD structure with status register value

ECMD read_pwmr(uint8_t ch). Read PWM register from eswitch
Received parameter: eswitch channel to read PWM register
Return value: ECMD structure with PWM register value

ECMD read_gcr(void). Read global configuration register from eswitch.
Received parameter: none.
Return value: ECMD structure with global configuration register value.
ECMD read_cnfr0(uint8_t ch). Read configuration register 0 from eswitch.
Received parameter: eswitch channel to read configuration register 0.
Return value: ECMD structure with configuration register 0 value.

ECMD read_cnfr1(uint8_t ch). Read configuration register 1 from eswitch.
Received parameter: eswitch channel to read configuration register 1.
Return value: ECMD structure with configuration register 1 value.

ECMD read_ocr(uint8_t ch). Read overcurrent register from switch.
Received parameter: eswitch channel to overcurrent register.
Return value: ECMD structure with overcurrent register value.

ECMD read_diagr0(uint8_t ch). Read diagr0 data.
Received parameter: eswitch channel to diagr0.
Return value: ECMD structure with diagr0 value.

ECMD read_diagr1(uint8_t ch). Read diagr1 data.
Received parameter: eswitch channel to diagr1.
Return value: ECMD structure with diagr1 value.

ECMD read_diagr2(uint8_t ch). Read diagr2 data.
Received parameter: eswitch channel to diagr2.
Return value: ECMD structure with diagr2 value.

Functions related to the system application are located in the Lighting_Applications.c file. This file contains the functions that check for faults in the outputs, check for under voltage fault, update PWM output duty cycle based on potentiometers or FreeMASTER interface, and output states based on board switches and FreeMASTER interface. Below is a description of the functions included in the Lighting_Applications file.

Lighting_Applications

Read_Status_Ch0(void). Read the status register to check faults on Ch0
Received parameter: none.
Return value: none.

Read_Status_Ch1(void). Read the status register to check faults on Ch1
Received parameter: none.
Return value: none.

**Read_Status_Ch2(void).** Read the status register to check faults on Ch2
Received parameter: none.
Return value: none.

**Read_Status_Ch3(void).** Read the status register to check faults on Ch3
Received parameter: none.
Return value: none.

**Check_Under_Voltage_Fault(void).** Checks if an under voltage fault is generated
Received parameter: none.
Return value: none.

**Update_Duty_Cycles_Pot(void).** Updates the duty cycle array based on the potentiometers
Received parameter: none.
Return value: none.

**Update_Ch0_State(void).** Updates state of CH0 based on button state, and update PWM duty_cycle of CH0
Received parameter: none.
Return value: none.

**Update_Ch1_State(void).** Updates state of CH1 based on button state, and update PWM duty_cycle of CH1
Received parameter: none.
Return value: none.

**Update_Ch2_State(void).** Updates state of CH2 based on button state, and update PWM duty_cycle of CH2
Received parameter: none.
Return value: none.

**Update_Ch3_State(void).** Updates state of CH3 based on button state, and update PWM duty_cycle of CH3
Received parameter: none.
Return value: none.
4.3 Software logic flow

Start

System init
Init eswitch, GPIO KBI, RTC, ADC

Under Voltage fault occurred?

No

Yes

Under Voltage fault cleared?

No

Yes

Read output status to check for faults

Read switch state

FreeMASTER PWM control ON?

Yes

Update output PWM duty cycle by FreeMASTER slider

No

Update output PWM duty cycle by potentiometer

Send command to eswitch

End
Chapter 5
Application Control

5.1 FreeMASTER tool

The FreeMASTER run-time debugging tool is used to control the application and monitor application variables during run-time. The document KEA128LEDLIGHTRD Quick start Guide, available at freescale.com, contains information on how to set up the FreeMASTER application in order to control the reference design application.

5.2 FreeMASTER graphical user interface

The following figure shows and explains how to use the FreeMASTER user interface to control the reference design application.
Figure 5-1. FreeMASTER user interface
Information in this document is provided solely to enable system and software implementers to use Freescale 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. Freescale reserves the right to make changes without further notice to any products herein.

Freescale makes no warranty, representation, or guarantee regarding the suitability of its products for any particular purpose, nor does Freescale 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 Freescale 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. Freescale does not convey any license under its patent rights nor the rights of others. Freescale sells products pursuant to standard terms and conditions of sale, which can be found at the following address: freescale.com/SalesTermsandConditions.

Freescale, the Freescale logo, and Kinetis are trademarks of Freescale Semiconductor, Inc., Reg. U.S. Pat. & Tm. Off. ARM and Cortex are registered trademarks of ARM Limited (or its subsidiaries) in the EU and/or elsewhere. All other product or service names are the property of their respective owners.

© 2015 Freescale Semiconductor, Inc.