Freescale Semiconductor
Application Note

Document Number: AN3556
Rev. 0, 1/2008

Time of Day Technique Using API Periodic Wakeup in Full Stop Mode for the S12X Family

by: Philippe Simon and Jean-Francois Thery

1 Introduction

Low power consumption is an important consideration in a wide range of applications. This is a critical parameter in automotive applications to prevent high battery discharge rates and conserve the battery charge between vehicle starts (recharge cycles).

Car manufacturers expect that a car can still start after six weeks in sleep mode and keep a very good accuracy on the daily digital clock. The S12X family of microcontrollers provides a range of features to enable users to reach this goal. This document describes the technique to implement time of day algorithms in full stop mode using the autonomous periodic interrupt module.

Contents

freescale semiconductor

2 Low Power Modes on the S12X Family

Numerous application notes have already been written on the topic. Among them are AN3289 Low Power Techniques for the S12X Family and AN2461 Low Power Management using HCS12 and SBC Devices. These application notes discuss the different modes available to optimize S12(X) current consumption.

The algorithm described is intended to be used in a system that is permanently powered up.

Periodic activity is requested to update the daily clock, check external events occurrence, and perform some processing if activity is required.

Car manufacturer targets for automotive body and dashboard applications lead to use S12X full stop mode allowing reaching below 200 µA average power consumption.

3 Scenario

The S12X achieves the lowest power consumption while running in full stop mode and keeping the API (autonomous periodic interrupt) running on its own RC oscillator (APICLK bit = 0). Periodic wakeup sequence P_{API} are generated to check IO/ADC ports activity and to increment the time of day counter.

The time to execute these software tasks is named t_{API}.

P_{API} can be set up using the eleven bits of the VREGAPIRH / VREGAPIRL registers and the following formula: Period = (VREGAPIR[11:0] +1) * 0.2 ms

Then, the reference period is set up with steps of 0.2 ms, and a better settlement can be performed by trimming the API with the API trimming register (VREGAPITR).

With the trimmed API internal clock is available with ±10% precision.

12	c	Trimmed API Internal clock $\Delta f/f_{nominal}$	df_{API}	– 10%	—	+10%	—

Figure 1. Trimmed Internal Clock

This is not enough to reach the TOD constraints. In fact for most car makers maximum deviation allowed is 0.5 s per day (worst case 1 s per day) at room temperature, which is more in the range of 5 ppm.

(One day represents 3600*24 = 86400 s, then 0.5 s / day = 0.5/86400 ~ 5.6 ppm).

So, the primary idea is to use a more precise clock source as a reference for the TOD main counter. The external crystal appears to be the best candidate to perform this task, assuming that crystal accuracy and software offset compensation allows reaching the 5.6 ppm described above.

Knowing the external crystal frequency, we use a general-purpose timer to measure precisely the API timing deviation. This measurement is performed periodically while the MCU is in run mode, clocked by external crystal and with no PLL involved to reduce power consumption. We can then measure elapsed time between API periods.

The time to execute these tasks of calibration is named t_{CAL}.

This calibration is performed periodically with a period called P_{CAL}.

4 Power Budget

With the notation mentioned in Section 3, “Scenario,” global current consumption is the summation of three currents:

- S12X running in full stop mode with fast wakeup and API generating periodic wakeup based on internal RC oscillator
- S12X running in self clock mode every P_{API} during t_{API}, updating TOD and checking peripheral activities.
- S12X running in run mode on external crystal with no PLL, every P_{CAL} during t_{CAL}, updating TOD and recalibrating the API period.

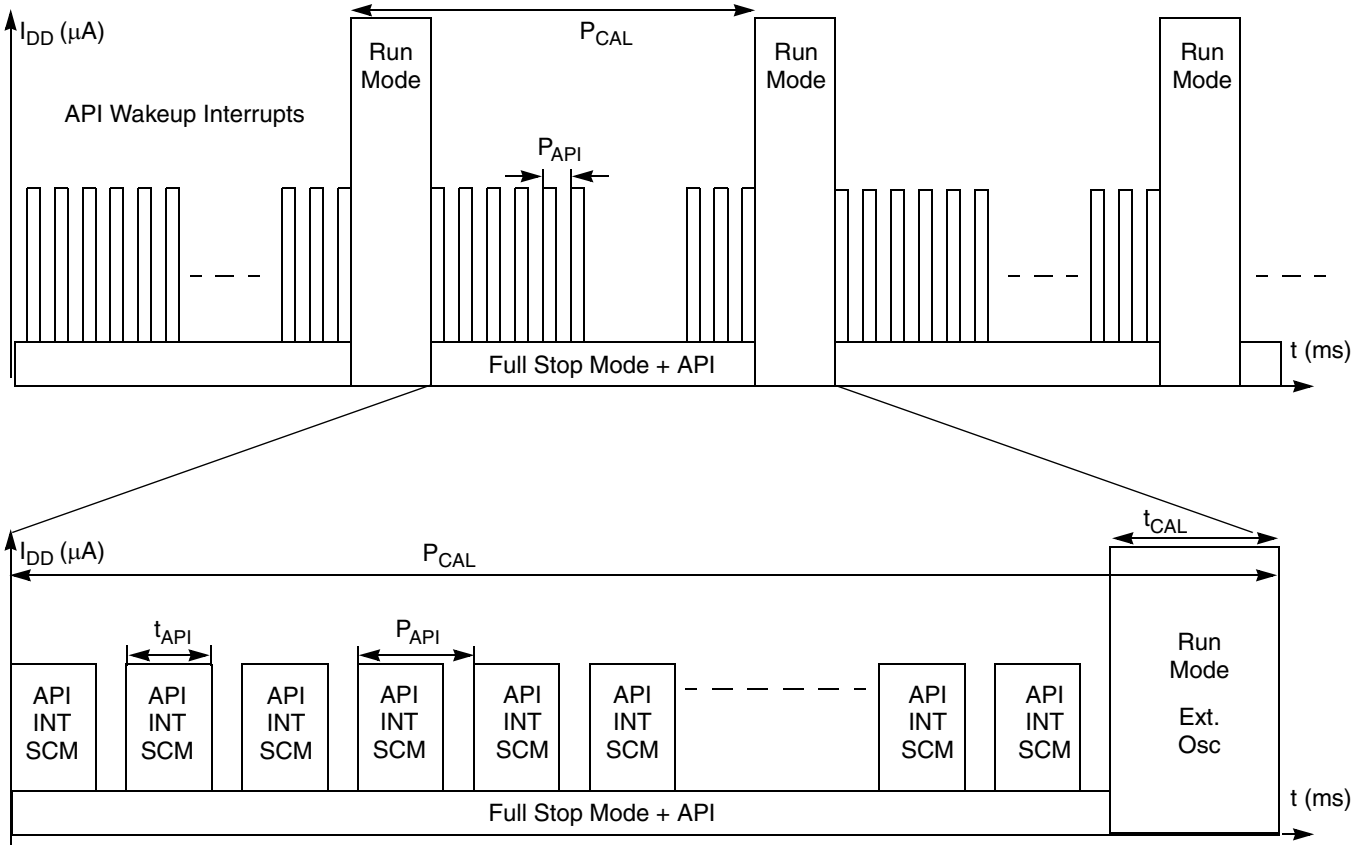

Figure 2.

As an example that has been measured by experiment with following values:

External crystal frequency: 4 MHz

t_{API} = 100 µs, P_{API}= 13 ms, t_{CAL} = 180 ms, P_{CAL} = 10 s

then

- Idd Full Stop = 60 µA (1)

1. given values have been measured at room temperature on a couple of parts as rough estimation but definitively not max value.

- Idd API Int = 4 mA [(1)] x T_{API}/ P_{API} = 31 µA
- Idd Run mode = 6 mA [(1)] x t_{CAL}/P_{CAL}/ = 108 µA (internal bus frequency is 2 MHz, no PLL)

Global total budget in time is around 199 µA

5 Detailed Algorithms

5.1 Choice of Recalibration Length — t_{CAL}

MCU is woken up periodically to be run on the external crystal. PWMCNT01 is set up as a 16-bit free running counter (concatenation of two 8-bit counters), clocked by the bus clock with the precision of the external crystal. This reference will allow precisely measuring the elapsed time between API interrupts.

To reach a precision of 5.6 ppm (see Section 3, “Scenario”), counter needs to count long enough to measure several API interrupts.

With a bus clock at 1 MHz (1 µs per tick), 5.6 ppm will be achieved by resolving the following equation:

Precision = Bus Clock Tick/ t_{CAL}

With a precision of 5.6 ppm ($5.6/10^6$), then t_{CAL} is equal to 10^6/5.6 µs ~ 180 ms.

In other words, 5.6 ppm precision is equivalent to a precision of 1 µs tick out of 180.000 µs time laps (t_{CAL}).

We can then easily deduce the number of API interrupt that would be requested.

To reuse figures given in Section 4, “Power Budget” (P_{API} = 13 ms), then 14 P_{API} will be needed to perform the calibration t_{CAL}. This parameter is named IncrementPeriod.

NOTE:

This formula demonstrates that tCAL is closely linked with the external crystal frequency. Then, with constant precision, higher frequency crystals will lead to a smaller tCAL value and then a reduced current consumption (see Section 4, “Power Budget”)

The PWMCNT01 16 bit counter will overflow at 0xffff and restart at zero. In our example, counting 180.000µs will lead the counter to overflow twice. This needs to be managed by the software algorithm.

5.2 Time of Day Counter

Time of recalibration (t_{CAL}) measures precisely the time elapsed between n times API. This measurement is updated and stored after each calibration in a variable named TimeBase.

TimeBase is used as the reference to increment the time of day counter. This means that after every IncrementPeriod — the TOD counter will be incremented by TimeBase value.

Note that the variation of TimeBase value must be small enough between two consecutive recalibration periods to guarantee a continuum towards the global precision. You may adapt the algorithm to add a dynamic calculation of P_{CAL}, depending on external temperature that may change P_{API} precision.

Nevertheless, external temperature variation is generally considered as a smooth transition phenomena and application dependent. Then P_{API} may vary from a second to tens of seconds.

The TOD counter is incremented modulo 1 second, meaning that if counter is greater or equal to 1 second, counter is re initialized with the rest of the subtraction.

5.3 Main Loop

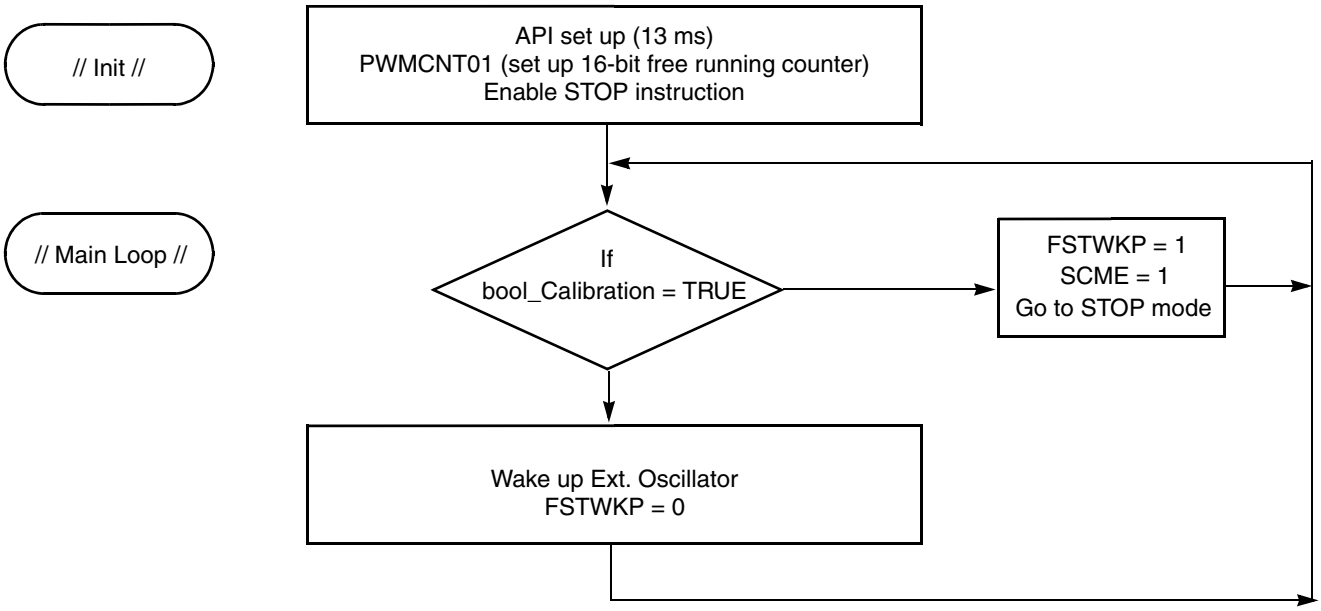

Figure 3. Main Loop

NOTE

The API setup routine must initialize the API timeout period (VREGAPIRH/L) and also the autonomous periodic interrupt trimming register (VREGAPITR). This trimming value is silicon dependent and must be initialized after a power on reset. Register setup can be performed by software using an external crystal as a reference and measuring, with the PWMCNT01 counter. The closest API timeout period reached with a dedicated trim value. (The dichotomy method may be used here.)

6 TimeBase Management

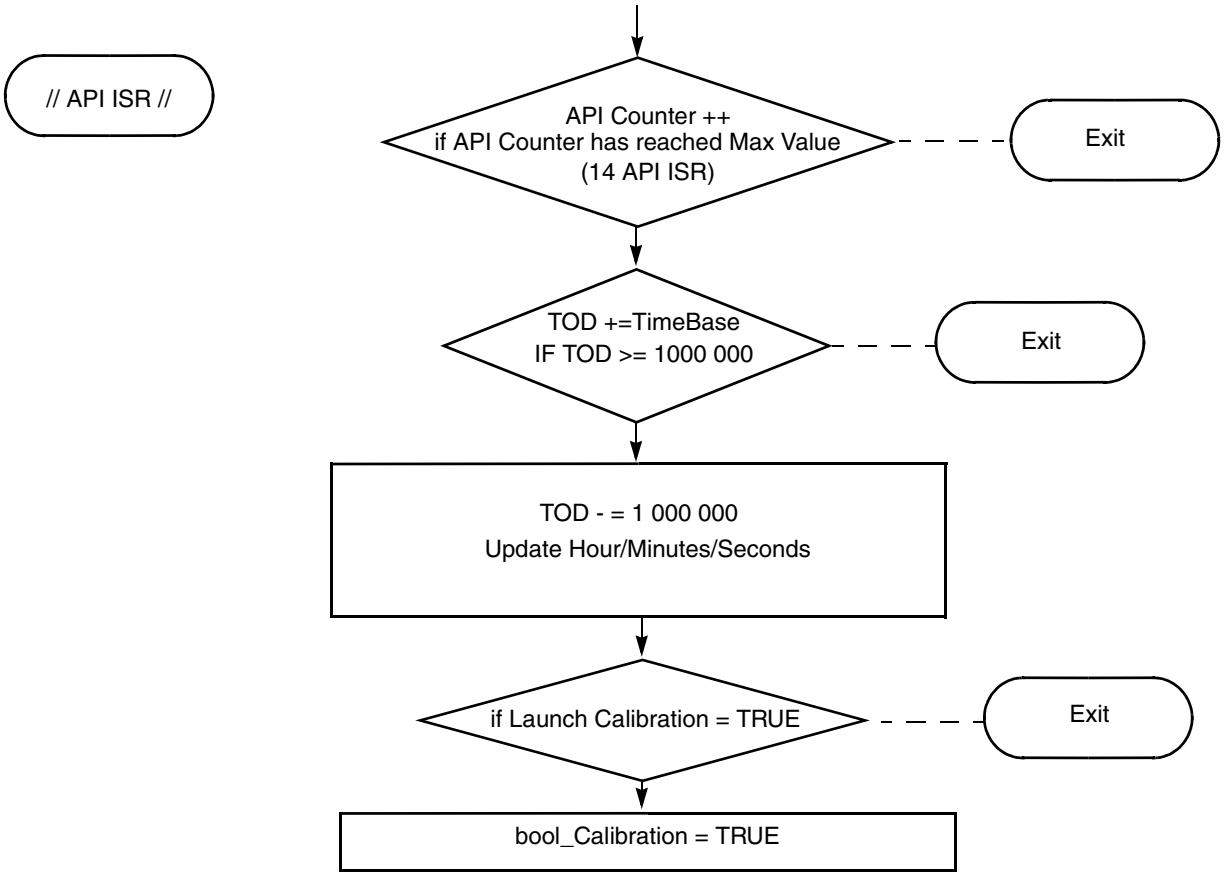

Figure 4. TimeBase Management

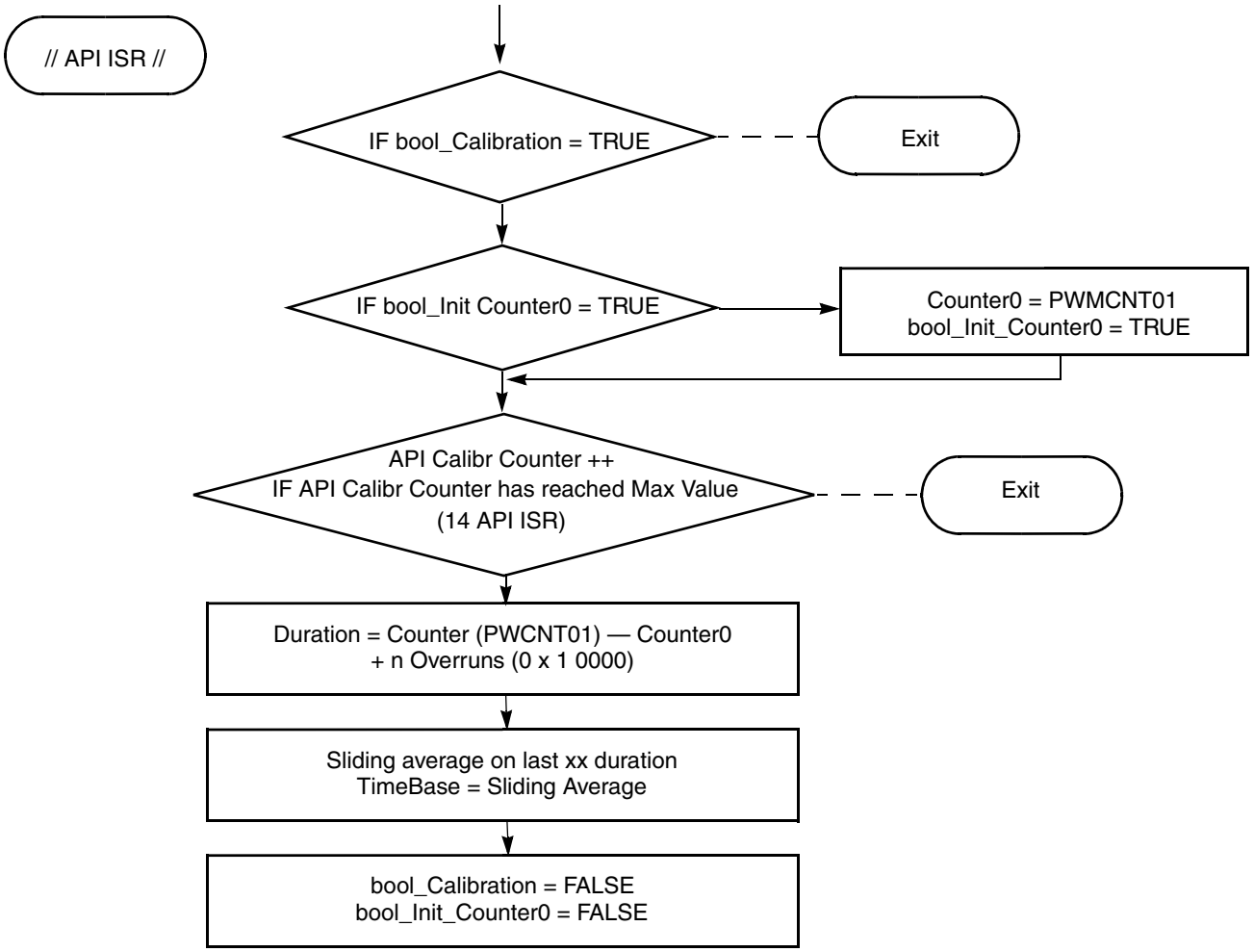

Figure 5. calibration and TimeBase Value Update

To filter the variation of the successive TimeBase values, the algorithm operates a sliding average calculation on the previous recorded values, offering a better immunity to potential noisy values. Assuming that N is the size of the buffer that contains historical values of TimeBase the sliding average is calculated as follows:

$$\text{Sliding Average} = \frac{\sum_{i=0}^{N-1} \text{TimeBaseBuffer}[i]}{N}$$

7 Real Time Consideration

The measurement of the time length between APIs interrupt service routines (ISR) during recalibration is performed inside the API ISR. The value of the counter PWMCNT01 is stored at the beginning and at the end of the calibration phase. Calculating the difference between the last and first values gives the instantaneous TimeBase value.

Special care needs to be taken while capturing the PWMCNT01 counter value. It has to be done at the first instruction of the ISR to avoid any delay or latency induced by code execution and to be deterministic (assuming the time to launch the ISR and reach this first instruction is a constant in time).

Moreover, the calibration timing need to be very precise and API ISR will have to get the highest priority in the execution flow of the software, while no other interrupt should disturb timing calculation.

Then, while running in STOP mode with periodical wake up, the API ISR must be the only one to be served during calibration phase.

In full run mode, while the application is running full speed after engaging the PLL, the recalibration appears to be useless because the MCU is running full time on external crystal clock, guaranteeing PWMCNT01 counter to be clocked with crystal precision. Then, the API ISR has to check the status of the PLL LOCK bit (CRGFLG register) to detect in which mode (full run or stop mode) the application is running.

Then each API ISR immediately needs to take PWMCNT01 values without recalibration periodic phases. Software will have to manage transition phases (between stop mode / recalibration phases and full run modes).

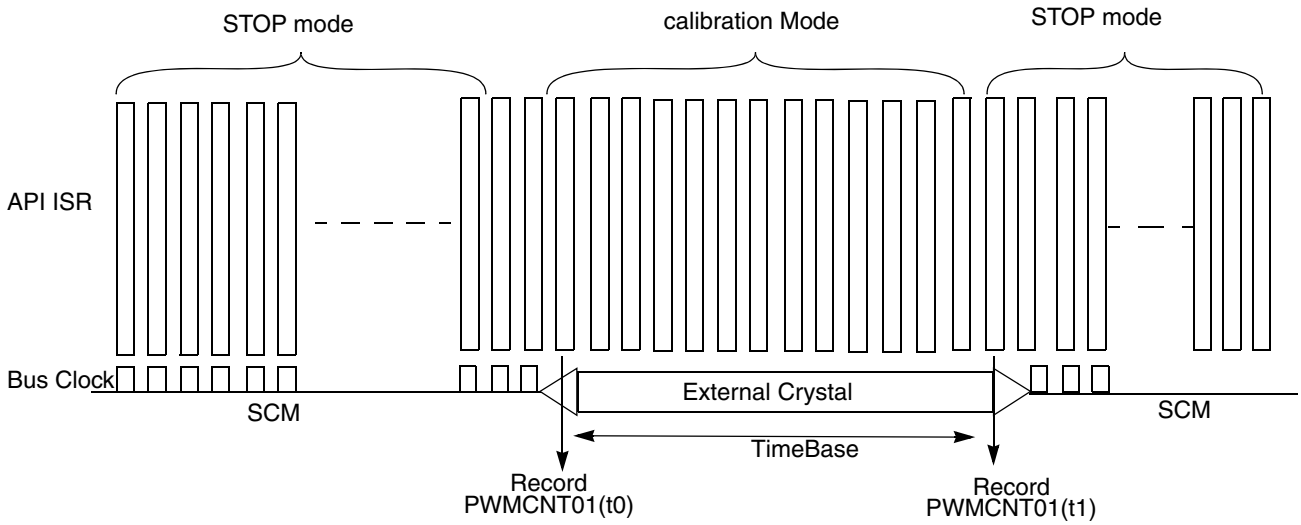

Figure 6. Calibration Mode

8 Summary

Thanks to its dedicated independent internal API RC oscillator, S12X has very low power consumption in stop mode using periodic API wakeup. The above algorithm has demonstrated that it is possible to add the time of day feature (hours/minutes/seconds) on this MCU with very good accuracy. This takes place without compromising the high power budget expectations induced by embedded automotive applications.

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 Hong Kong Ltd.
Technical Information Center
2 Dai King Street
Tai Po Industrial Estate
Tai Po, N.T., Hong Kong
+800 2666 8080
support.asia@freescale.com

For Literature Requests Only:
Freescale Semiconductor Literature Distribution Center
P.O. Box 5405
Denver, Colorado 80217
1-800-441-2447 or 303-675-2140
Fax: 303-675-2150
LDCForFreescaleSemiconductor@hibbertgroup.com

Document Number: AN3556
Rev. 0
1/2008

freescale™
semiconductor