AN2144
Rev. 3, 12/2004

Programming the MSC8101 Periodic Interrupt Timer (PIT)

By Joe Liccese

This document describes the steps in configuring the Baud-Rate Generator (BRG) registers, Parallel I/O Port registers, Periodic Interrupt Timer (PIT) registers, SIU-CPM Interrupt Controller (SIC) registers, and the Programmable Interrupt Controller (PIC) registers to generate PIT interrupts. The two system interface unit (SIU) timers, PIT and time counter (TMCNT), use the same clock source, TIMERSCLK, which can be derived from several sources. This application note describes the set-up necessary for use of a BRG as the clock source for the timer clock for use by the PIT. This document also describes the set-up necessary to enable and handle PIT interrupts. **Figure 1** illustrates the integration of the MSC8101 clock synthesizer, BRG, timer clock, and PIT signals and registers used to clock the PIT timer.

CONTENTS

1 Periodic Interrupt Timer Set-up

The subsections that follow walk you through the main sections of the PIT clocking diagram shown in **Figure 1** and describe the set up procedure for each.

1.1 Baud-Rate Generator

The communications processor module (CPM) contains eight independent, identical baud-rate generators for use with the fast communications controller (FCC), serial communication controller (SCC), and serial management controller (SMC). The clocks produced by the BRGs are sent to the bank-of-clocks selection logic, where they are routed to the controllers. The output of a BRG can also be routed to a pin for external use (see **Figure 2**). The clock source for a BRG can be BRGCLK or an external clock. The BRGCLK is an internal signal generated in the MSC8101 clock synthesizer specifically for the BRGs. BRGCLK is two times the CPM clock; so if the CPM clock, based on the MODCLK settings, runs at 50 MHz, then BRGCLK runs at 100 MHz. This section describes how to set up the BRG and parallel I/O port registers to use the BRGCLK source that produces a 32.768 KHz BRG1 clock for use as the source for the timer clock. The MSC8101 clock synthesizer module is not discussed. However, keep in mind that the MODCLK configuration uses a 50 MHz crystal to produce a BRGCLK of 50 MHz to the BRG. The MODCLK setting chosen is mode 57.[1]

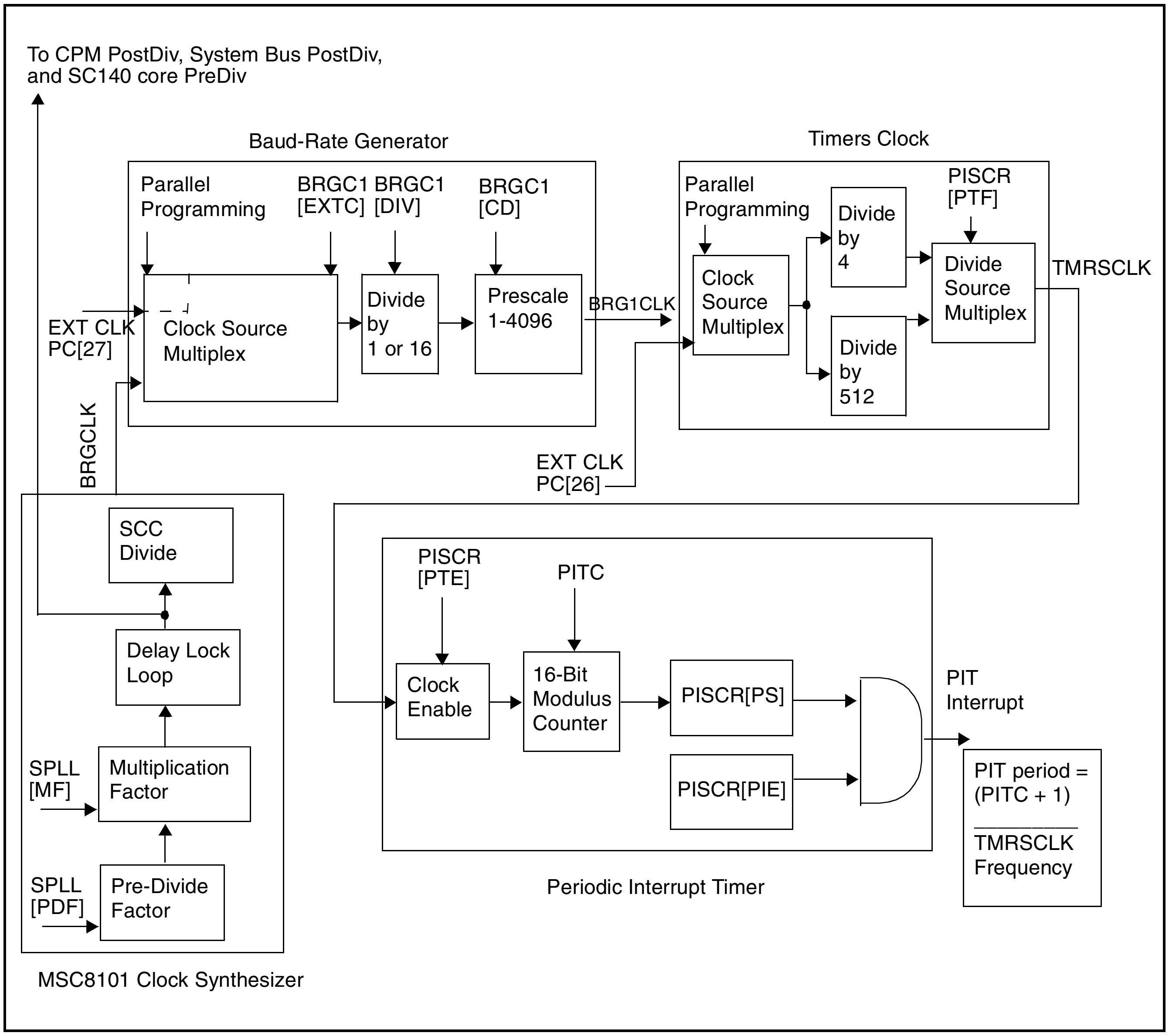

Figure 1. PIT Clocking Diagram

1. For information on MSC8101 clock mode selection, consult the Freescale Semiconductor application notes listed in **Section 4**, *References*, on page 7.

With an on-board crystal of 50 MHz, setting MODCLK to mode 57 and the System Clock Control Register (SCCR) division factor for the BRG bits (30–31) to 01 (divide by 16) produces a 50 MHz BRGCLK to the BRG. See **Table 1**.

Table 1. MODCLK Set to Mode 57 With a 50 MHz Crystal

Component/Register Bits	Frequency
Crystal	50 MHz
SPLL[DF] (divide by 2)	25 MHz
SPLL[MF] (multiply by 10)	250 MHz
SCCR[DF] (divide by 5)	50 MHz

The only BRG that can be used as a clock source to the timer clocks is BRG1. To configure BRGCLK as the source to BRG1, we set the BRGC1[16–17]:ETXC bits to 00. Next, we determine the value for the Clock Divider, BRGC1[19–30]:CD bits. The clock divider presets an internal 12-bit counter that decrements at the DIV16 output rate. When the counter reaches zero, it is reloaded with CD. A CD value of 0xFFF produces the minimum clock rate for BRG (divide by 4096) while a CD value of 0x000 produces the maximum rate (divide by 1). Since ours is a 32.768 KHz BRG1 clock, with a BRGCLK of 50 MHz the CD bits are set to 0x5F5, effectively dividing the BRGCLK by 1526 (0x5F5+1), resulting in a BRG1 clock of approximately 32.765 KHz.

The BRGC1[31]:DIV16 bit selects a divide-by-1 or a divide-by-16 prescaler before reaching the clock divider. For our demonstration, this bit is set to 0 (divide by 1), since we have already achieved our desired BRG1 clock rate.

To enable the BRG count, we set the BRGC1[15]:EN bit. BRGC1 now begins generating a 32.723 KHz clock. The value actually loaded into BRGC1 is 0x00010BEA. This completes the set-up of BRG1, the timer clock source clock.

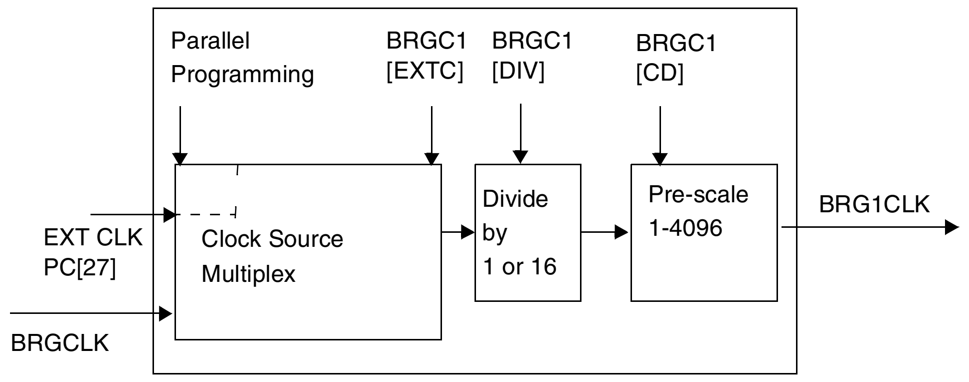

Figure 2. Baud-Rate Generator

1.2 Parallel I/O Ports

The CPM supports four general-purpose I/O (GPIO) ports, A–D. Each pin in the I/O ports is configured as a GPIO signal or as a dedicated peripheral interface signal. Port C is unique in that eight of its pins can generate interrupts to the interrupt controller. Whether a pin operates as a GPIO pin or as a dedicated peripheral depends on the settings in the Port Pin Assignment Register (PPARx). Each pin is independently configured as a GPIO pin if the corresponding PPARx bit is cleared or as a dedicated peripheral if it is set. A pin is configured as input if the corresponding control bit in the Port Data Direction Register (PDIRx) is cleared or as an output if it is set. All PPARx and PDIRx bits are cleared on system reset.

Data written to the Port Data Register (PDATx) is stored in an output latch. If a port pin is configured as an output, the output latch is gated onto the port pin. When PDATx is read, the port pin itself is read. The Port Open-Drain Register (PODRx) determines whether the corresponding pin is actively driven as an output or as an open-drain driver. Clearing a PODRx bit configures the pin as an actively driven output, and setting a PODR bit configures the pin is an open-drain driver.

The Port Special Options register (PSORx) bits are effective only if the corresponding PPARx bits = 1 (dedicated peripheral). With its corresponding PPARx bit set, a PSOR bit cleared selects the dedicated peripheral option 1. If the PPARx bit is set, then the dedicated peripheral option 2 is selected.[1] The PIT source example configures the PPAR, PSOR, PDIR, and PODR registers as follows:

Table 1-1.

PSORC = 0x00000000	**All Port C pins option 1**
PDIRC = 0x00000005	Configures PC[31] and PC[29] as outputs and all other Port C pins as inputs
PPARC = 0x00000001	Configures PC[31] as a dedicated peripheral (BRG1)
PODR = 0x00000000	Dedicated output pins actively driven as an output

The Port C I/O pins [31] and [29] are used for debug purposes only. PortC[31] monitors the BRG1 clock signal, and the PORTC[29] output is toggled within the PIT interrupt service routine (ISR) to monitor the PIT period.

1.3 Timer Clock

The Timer Clock module generates the TIMERSCLK signal for use by the PIT and TMCNT SIU timers. The source for TIMERSCLK can be derived from several external sources or BRG1. A TIMERSCLK source is selected by setting up the Parallel I/O Port C registers and the Periodic Interrupt Status and Control Register (PISCR).[2] In the application discussed here, the desired TIMERSCLK is 8.192 KHz. To produce this desired interval using a 32.723 KHz source, we set the PISCR Periodic Interrupt Frequency (PTF) bit (14) to 1. This results in a divide by 4 producing an approximately 8.192 KHz TIMERSCLK signal.

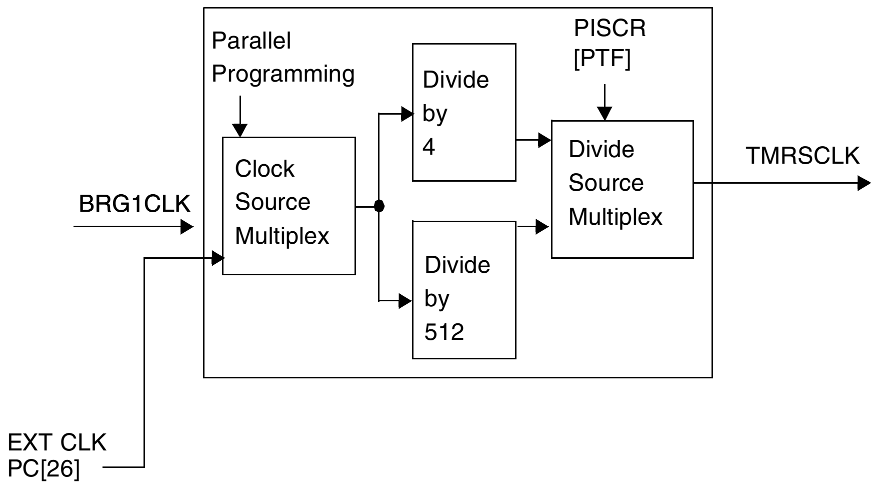

Figure 3. Timer Clock

1. Refer to Tables 41-1 through 41-4 in the *MSC8101 Reference Manual* for proper configuration of port pins.
2. For parallel I/O port C configuration, refer to the section on Parallel I/O ports in Chapter 41 of the *MSC8101 Reference Manual.*

1.4 Periodic Interrupt Timer

The periodic interrupt timer (PIT) consists of a 16-bit counter clocked by TIMERSCLK. The counter decrements to zero when it is loaded with a value from the PIT Count register (PITC). After the counter reaches zero, PISCR[PS] is set and an interrupt is generated, to the SIC if PISCR[PIE] is set. At the next input clock edge, the value in the PITC is again loaded into the counter and the process repeats. When a new value is loaded into the PITC, the PIT is updated, the divider is reset, and the counter begins counting.

Setting PISCR[PS] creates a pending interrupt that remains pending until PISCR[PS] is cleared. Any write to the PITC stops the current countdown, and the count resumes with the new PITC value. If PISCR[PTE] = 0, the PIT cannot count and retains the old count value.

The Periodic Interrupt Timer Count Register (PITC) contains the 16 bits to be loaded into the modulus counter.Since the TIMERSCLK we are generating is 8.192 KHz, to generate a PIT period of 10ms we write 0x00510000 to the PITC register. Using the PITC Period formula shown in **Figure 4**, we get the following values:

- pit period = (pitc + 1) / TIMERSCLK
- pit period = (81+1)/8192
- pit period = 0.010 (every 10ms PIT generates an interrupt)

The PITR, is a read-only register that shows the current count down value for the PIT. This counter is not affected by reads.

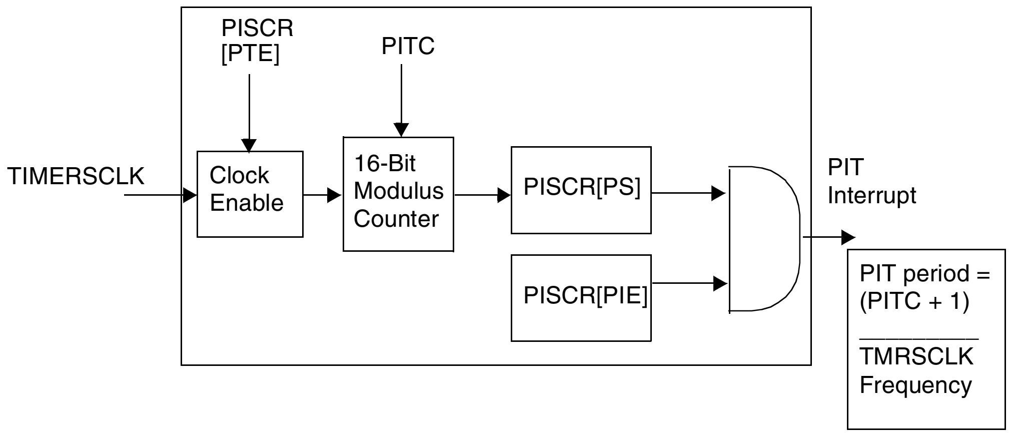

Figure 4. Periodic Interrupt Timer

2 Handling PIT Interrupts

The MSC8101 interrupt scheme consists of three different interrupt controllers:

- Programmable interrupt controller (PIC), which operates in the SC140 core
- SIU-CPM interrupt controller (SIC), which generates interrupt requests to the PIC
- External SIU-CPM interrupt controller (SIC_EXT), which generates interrupt requests to an external host CPU.

The SIC receives interrupts from internal sources such as the PIT or the TMCNT, from the CPM and from external sources such as port C parallel I/O pins or interrupt requests (IRQs). The SIC generates interrupt requests to the PIC to be handled by the SC140 core.

The PIC receives interrupts from DSP peripherals, DMA, external IRQ[2-3], EONCE and the SIC. When the PIC detects an interrupt request (IR) on one or more of its inputs, it arbitrates each IR according to its priority level and location and generates the following:

- An IR signal to the SC140 core, indicating that an IR input has requested interrupt service from the SC140 Core.
- An RIPL[2-0] signal indicating the priority of the IR
- An entry in the predefined vector address bus (VAB), determined by the location of the IR

Programming the System Interface Unit (SIU) mask registers, SIMR_H and SIMR_L, provides a means of masking interrupt requests to the core. Each SIMR bit corresponds to an interrupt source. To enable an interrupt; write a one to the corresponding SIMR bit. When a masked interrupt source has a pending interrupt request, the corresponding SIU Interrupt Pending Register (SIPNR) bit is set, even though the interrupt is not generated to the core. Pending unmasked interrupts are presented to the core in order of priority. The interrupt vector that allows the core to locate the Interrupt Service Routine (ISR) is made available to the core by reading SIU Interrupt Vector Register (SIVEC). The interrupt controller passes an interrupt vector corresponding to the highest priority, unmasked, pending interrupt. The SIU registers fall into three categories:

- Interrupt controller registers - control configuration, prioritization, and masking of interrupts and include registers for determining the interrupt sources.
- System configuration and protection registers - configures the SIU, defines the base address for the internal memory map, configures the watchdog timer, specifies bus characteristics, as well as general functionality of the 60x and local buses; such as arbitration, error status, and control.
- Periodic Interrupt registers - configure and provide status for periodic interrupts.

The PIC is a peripheral module to serve all the IRs and NMIs received from MSC8101 peripherals and I/O pins. The PIC is memory mapped to the SC140 Core and is accessed via the SC140 Core QBus.

The peripheral bus (Qbus) interface provides the control and status registers and address decoding generation. The PIC serves a total of 24 IRs and 8 NMIs. Each IR can be configured as edge-triggered or level-triggered and can be assigned a priority in the range 0 through 7, where priority 0 masks the interrupt. On system reset all IRs are masked and configured as level-triggered. In addition the Core Status Register (SR) IPL bits are set effectively disabling interrupts.

2.1 Interrupt Programming Model

The SIU and PIC interrupt programming model, as it applies toward SIC and PIT interrupts, consists of the following steps:

1. Set the interrupt table base address in the Vector Base Address (VBA) register.
2. Clear any previous PIT interrupt (sipnr_h |= 0x00000002).
3. Enable PIT interrupt in the SIC (simr_h |= 0x00000002).
4. Clear any previous SIC interrupt in IRQ Pending Register B (IPRB = 0x00000000).
5. Configure Edge/Level-Triggered IRQ Priority Register E (ELIRE) to enable $\overline{\text{IRQ16}}$ (SIC interrupt) (ELIRE = (ELIRE & 0xFFF0) | 0x0003 - IRQ16 level-triggered, $\overline{\text{IRQ16}}$ priority level 3).
6. Configure the SC140 core Status Register (SR) to permit level 3 interrupts (sr_h &= 0xFFBF).
7. Locate the SIC interrupt handler (ISR) at the appropriate VBA offset (VBA+0x0C00).
8. Enable interrupts (**ei**).

2.2 Interrupt Handler Model

The programming model for handling a PIT interrupt consists of the following steps:

1. Clear the Periodic Interrupt Status bit, PISCR[PS] (8). This bit is set when PISR count = 0. Writing a one to PISCR[PS] clears this bit. (piscr |= 0x0080).
2. Clear the PIT interrupt bit in the SIU Interrupt Pending Register (sipnr_h |= 0x00000002).
3. Clear the SIC interrupt bit in the PIC Interrupt Pending Register B (iprb |= 0x0001).
4. Execute PIT handler. For the PIT example we simply toggle PC[29] so we may monitor the PIT period. (pdatc ^ 0x0004).

3 Building the PIT Example

This section describes the steps necessary to build, run, and verify the example PIT interrupt on an MSC8101ADS board. To build the example, download the associated PIT interrupt software (AN2144SW) and extract the files into a project folder. Double click on the file entitled: `test.mcp` to launch the Metrowerks® CodeWarrior® for StarCore™ tool suite and load the PIT interrupt project into it. The `test.mcp` project file is a Metrowerks CodeWarrior v2.02 project file.

The PIT example source is designed to produce a 10 ms PIT interval when it runs on an MSC8101ADS board with a 50 MHz crystal and a MODCLK setting of mode 57. If your MSC8101ADS board is configured differently, you must follow the PIT application note steps to modify the source for your particular set-up.

After the PIT interrupt project is loaded into CodeWarrior and any necessary modifications are made during MSC8101ADS set-up, select the build icon from the CodeWarrior project tool bar to build the PIT interrupt example executable.

After the build successfully completes, click on the debug icon on the CodeWarrior project tool bar, to download and run the PIT interrupt example. You can monitor the BRG clock and the PIT interval by connecting an oscilloscope to GPIO PORTC[31] (P2–D1) and PORTC[29} (P2–D3) pins, respectively. The BRG clock should indicate 50 MHz, and the PIT interval should indicate a 10 ms interval. Since you are toggling GPIO pin PORTC[29] within the PIT ISR, the actual scope value should indicate 2x the desired interval (20 ms).

4 References

The following application notes and software packages can be found on the web site listed on the back cover of this document.

[1] AN2306/D, *Clock Mode Selection for MSC8101 and MSC8103 Mask Set 1K87M*

[2] AN2306SW, Excel spreadsheet to accompany application note AN2306/D.

[3] AN2288/D, *Clock Mode Selection for MSC8101 Mask Set 2K42A*.

[4] AN2288SW, Excel spreadsheet to accompany application note AN2288/D.

[5] AN2144SW, software to accompany application note AN2144 (this application note).

How to Reach Us:

Home Page:
www.freescale.com

E-mail:
support@freescale.com

USA/Europe or Locations not listed:
Freescale Semiconductor
Technical Information Center, CH370
1300 N. Alma School Road
Chandler, Arizona 85224
+1-800-521-6274 or +1-480-768-2130
support@freescale.com

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

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

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 Order No.: AN2144
Rev. 3
12/2004