AN2990
Rev. 0, 7/2005

Using the MSC8126 Turbo Coprocessor (TCOP) Driver

By Tina Redheendran

Decoding turbo code blocks heavily taxes calculation resources. To lighten this burden, the MSC8126 turbo coprocessor (TCOP) decodes high data rate input streams while requiring a minimum CPU-time overhead, thus freeing the SC140 cores for other processing tasks. The TCOP offers superior performance and flexibility, a simple interface, and support for the 3GPP and CDMA2000 standards. This application note describes the Freescale TCOP driver software for use in programming and operating the TCOP.

CONTENTS

1 TCOP Overview....1
2 Directory Structure 2
3 TCOP Driver Files 4
4 Running the TCOP Driver 5
5 Modifying the TCOP Driver 7
5.1 TCOP Parameter Settings 7
5.2 TCOP Interrupt Handler 9

1 TCOP Overview

The TCOP is a shared resource, and the four SC140 cores of the MSC8126 DSP must service it one at a time. The TCOP uses the upper 220 KB portion of the M2 memory (address 0x01040000–0x01076FFF). When the TCOP is turned off, the local bus can access its memory and prepare data. When the TCOP is turned on, it treats its memory as private, and the local bus cannot access it. The SC140 cores can access the TCOP memory at any time through the MQBus, but if both the SC140 core and the TCOP access the same 32 KB memory group, the core access stalls because the TCOP access has higher priority.

Before it is enabled, the TCOP must be provided with control parameters and the data to decode. Control parameters are written into the control registers, TCR1 and TCR2. These parameters are fairly stable and, in most applications, must be written only once. Parameters include block size, puncturing scheme (coding rate), number of iterations, and whether to interrupt after a block is decoded. The TCOP registers can be written only when the TCOP is disabled, but they can be read at any time.

Additionally, 8-word aligned pointers must be provided to the base address of the input data blocks in the TCOP area of the M2 memory. The data must be in the correct format and order. For all rates, there must be pointers to the interleave table, for the lambda input data and for Yp1 and Yp2 input data. For rate 1/4 and 1/5, there must also be pointers to Yp1a and Yp2a input data. The SC140 core or the host must also provide the interleave table in the same memory area. The host has full programmability in configuring this table, which is very important for the interleave table since it is not yet stable and tends to vary between versions of the same standard.

When the TCOP is enabled, it reads the data block from M2 memory, decodes it, and returns a decoded block in the same location, with no SC140 core involvement throughout the decoding process (the SC140 cores do require a very small amount of post-processing). The TCOP can run a number of iterations and then interrupt the SC140 core(s). Alternatively, the TCOP can be programmed to stop on a quality criterion. If all the soft lambda results exceed a programmable threshold, the decoding stops, and the host is interrupted.

For details on Turbo decoding theory and TCOP programming, operation, and data formats, refer to the TCOP chapter of the *MSC8126 Reference Manual* and also to the Freescale application note entitled *Using the Turbo Decode Coprocessor (TCOP) of the MSC8126 DSP Device* (AN2785). Both documents are available at the web site listed on the back cover of this document.

2 Directory Structure

The TCOP driver runs on the MSC8126ADS and runs only on one core of the MSC8126 (Core 0). To run the TCOP driver, you need a MSC8126ADS board with the appropriate connection cable and power supply, as well as a properly installed version of CodeWarrior™ for StarCore™. This application note assumes familiarity with CodeWarrior and with the MSC8126 processor, including the TCOP.

The TCOP driver files are contained in a zip file named `tcop_driver.zip`. **Figure 1** shows the directory structure for the driver files after they are extracted from the zip file. There are two main directories, one for each of the CodeWarrior (CW) projects included in the TCOP driver.

- The `host` directory contains the CW project for the MSC8103 host: `host.mcp`. This simple project supplies the reset information for the MSC8126 and flashes LEDs 7, 8, 12, and 13 on the MSC8126ADS in an alternating pattern to verify that the driver in running.
- The `tcop_driver` directory contains the CW project for Core 0 of the MSC8126: `tcop_driver.mcp`. This project programs and operates the TCOP. This directory also contains the `main.c` program with the code to set up the TCOP parameters and call the TCOP driver code and the MSC8126 register header file, `MSC812x_RegMemMap.h`. Within the `tcop_driver` directory are two more directories that contain the code for the TCOP driver.
  - `test_files` contains the data for verifying the operation of the TCOP driver. This directory holds the input data and the reference output data for the two test vectors included with the driver.
  - `tcop` contains the TCOP driver functions that are described in detail in the next section.

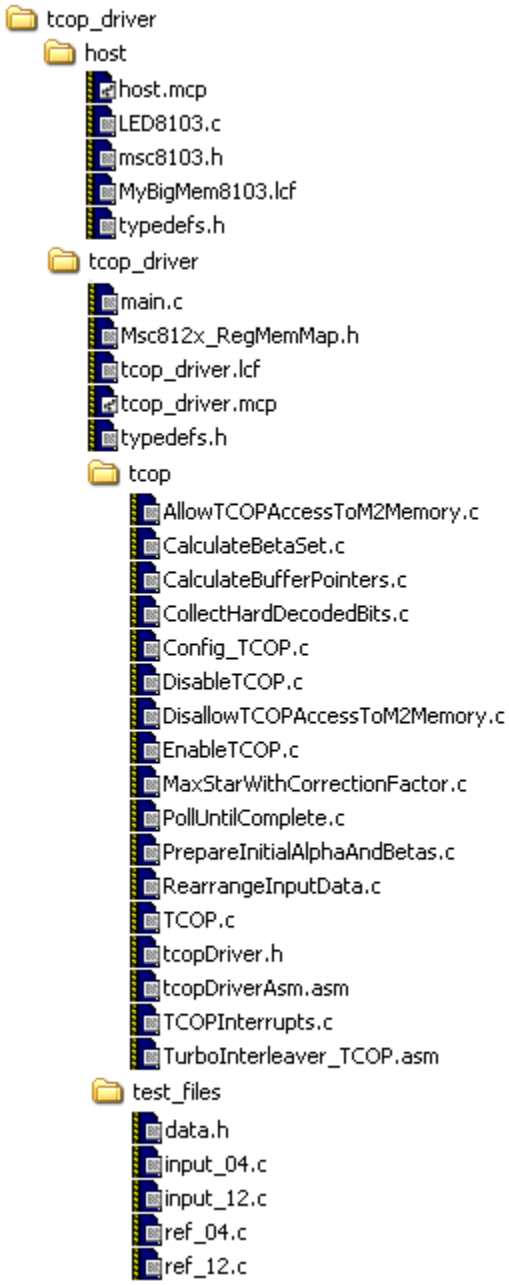

Figure 1. TCOP Driver File Structure

3 TCOP Driver Files

Most TCOP driver files can program and operate the TCOP as they are. You do not need to change most of the files to change the TCOP processing mode. Files that require change are described in detail in **Section 5**, *Modifying the TCOP Driver,* on page 7. The TCOP driver files are listed in **Table 1**.

Table 1. TCOP Driver Files

File	Description
TCOP.c	The main driver code to perform turbo decoding with the TCOP, this file sets up the pointers to the TCOP buffers and registers and calls most of the other TCOP driver functions that set up and operate the TCOP and collect the output data.
tcopDriver.h	The header file containing all the function prototypes, constant definitions, and type definitions.
CalculateBufferPointers.c	Calculates the base addresses to all the buffers the TCOP needs to perform the decoding: • Interleaver table • Systematic and parity input symbol buffers • Alpha and beta buffers for internal use • Hard decision lambda buffer for internal use and for output • Optional soft output results buffer
RearrangeInputData.c	This function rearranges the input data as the TCOP requires and treats the input data as one array with the systematic data stream and two parity data streams interleaved. This function separates the systematic and parity data into three separate buffers. The coding rate is 1/3 and only the Yp1 and Yp2 buffers are filled. You must rewrite this function if you are using coding rate 1/2, 1/4, or 1/5.
PrepareInitialAlphaAndBetas.c	This function calculates the initial betas for each MAP decoder and sets the initial alphas to zero. It calls CalculateBetaSet.c to determine the initial value for the betas.
CalculateBetaSet.c	This function calculates the initial betas for an indicated MAP decoder using the equations referenced in the TCOP section of the *MSC8126 Reference Manual*.
MaxStarWithCorrectionFactor.c	This function calculates the MaxStar function, which is an approximation of a log add calculation by a maximum function with a correction factor. This function can also calculate a simple max function (with no correction factor) by setting the correction factor to negative infinity and is used when the TCOP approximation is set to MAX-Log MAP (not MaxStar).
Config_TCOP.c	This function configures the TCOP by setting the TCOP registers (TCR1, TCR2, CFR, and SCR) and the base address registers (YSBA, YP1BA, YP2BA, IOBA, LBA, BIBA, and optionally SRBA, YP1ABA, and YP2ABA).
AllowTCOPAccessToM2 Memory.c	This function sets the bit in the TCOP GSCR, which disconnects the 220 KB shared memory from the local bus and connects it to the TCOP. This function should be called after the host has filled the data for the TCOP in the shared memory.
EnableTCOP.c	This function enables the TCOP by setting the TCOP enable bit (TEN) in the TCR1. All configuration registers should be set before this function is called.
DisableTCOP.c	This function cancels the TCOP decoding by clearing the TCOP enable bit (TEN) in the TCR1. Typically, the TCOP automatically disables itself when the stop condition is met, so you do not usually call this function.
DisallowTCOPAccessToM2 Memory.c	This function clears the bit in the TCOP GSCR that disconnects the 220 KB shared memory from the TCOP and connects it to the local bus. Call the function before the host starts to read the data processed by the TCOP.
CollectHardDecodedBits.c	This function collects the hard output results from the lambda output buffer by extracting the least significant bit of each of its elements.

Table 1. TCOP Driver Files (Continued)

File	Description
TCOPInterrupts.c	This file contains the functions to enable and disable the TCOP interrupts and the interrupt initialization function that ties the TCOP interrupt to the SC140 core LIC and PIC. It includes the TCOP interrupt handler that runs when the TCOP meets its stop condition and finishes processing data.
PollUntilComplete.c	This function polls the TSR[END] flag until it is set, indicating that the TCOP has met its stop condition and finished processing data. This function is used only if interrupts are not used to determine when TCOP processing finishes.
tcopDriverAsm.asm	This assembly routine at the TCOP interrupt vector jumps to the interrupt handler when the TCOP interrupt is triggered.
TurboInterleaver_TCOP.asm	An assembly routine that generates the turbo interleaver table for a given block length. This routine is called from the main.c function.

4 Running the TCOP Driver

The TCOP driver is very easy to run, and it automatically verifies that the data is correctly decoded. Use the following steps to run the TCOP driver:

1. Set up the MSC8126ADS board and ensure that CodeWarrior is installed and running.
2. Open the `host.mcp` file. A window similar to that shown in **Figure 2** appears, except that the `tcop_driver` project is not yet open.
3. Click on the Debug icon at the top of the `host.mcp` project window (see **Figure 2**). CodeWarrior should automatically build this project (if the build settings in the IDE preferences are set to **ALWAYS BUILD BEFORE RUNNING**), open the `tcop_driver` project, build the `tcop_driver` project, and open the debug windows for both projects.
4. Under the Multi-Core Debug menu, choose **RUN ALL** to run both projects. LEDs 7, 8, 12, and 13 on the MSC8126ADS flash and a standard I/O window opens. **Figure 3** shows what the text in the STDIO window looks like when the TCOP driver runs correctly.

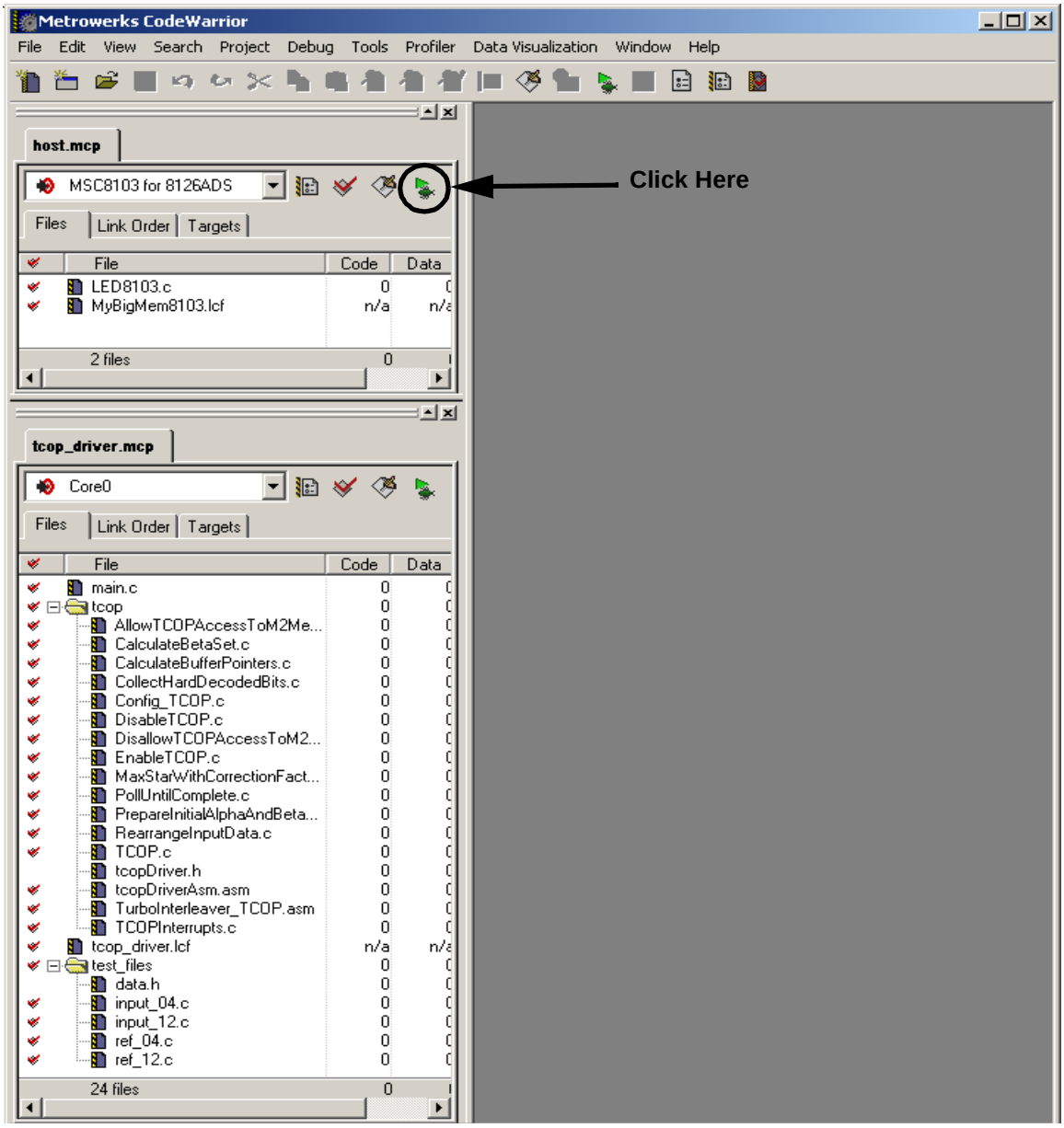

Figure 2. TCOP Driver CodeWarrior Projects

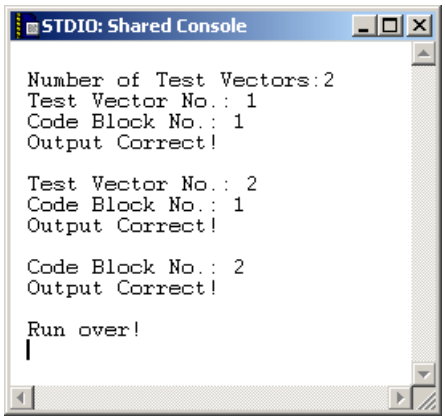

Figure 3. TCOP Driver Output

5 Modifying the TCOP Driver

Two sections of code can be modified to change the operation of the TCOP driver from its original state: the parameter settings to program the TCOP and the interrupt handler routine.

5.1 TCOP Parameter Settings

The code to set up the TCOP parameters resides in the `main.c` function and is shown in **Example 1**. This code sets a number of variables that are written to the TCOP control registers: TCR1, TCR2, CFR, and SCR. The code sets up the parameters listed in **Table 2**, all of which are variable values.

Table 2. TCOP Driver Files

File	Description
usiNumDecodeIterations	Defines the number of turbo iterations to run when the stop condition is set to normal mode. Each turbo decoding iteration contains two MAP decoders, and the TCOP is programmed with the number of MAP decoders. Thus, the TCOP is programmed with twice the number of turbo decoding iterations or two times `usiNumDecodeIterations`. The TCOP driver sets the number of iterations to 4, but it can be changed to any value up to 15.
usiInverseRate	The inverse of the coding rate. For example, the TCOP driver sets this variable to 3 to select a coding rate of 1/3. The TCOP can decode at rates of 1/2, 1/3, 1/4, or 1/5, so the inverse coding rate can be set to 2, 3, 4, or 5.
MaxApproximation	Defines the type of approximation used by the MAP decoder: MAX-Log-MAP, which approximates the log function with the max function, or MAX*, which adds a correction term. The TCOP driver sets this variable to MAX_STAR_LOG_MAP, but it can be changed to MAX_LOG_MAP.
siCorrectionFactor	The correction factor for the MAX* approximation. If the MAX* approximation (using the MAX-Log-MAP approximation) is not used, this variable should be set to the constant MINUS_INFTY_LI. When the MAX* approximation is used, the correction factor value must match the format of the other inputs, as described in the TCOP section of the *MSC8126 Reference Manual*.
Puncturing	Defines which parity bits are to be computed by which MAP decoder. Three examples of the puncturing variable are given in the TCOP driver, which correspond to the CDMA2000 rate 1/2, 1/3, and 1/4 standards: INVERT_RATE_2, INVERT_RATE_3, and INVERT_RATE_4. The TCOP driver uses INVERT_RATE_3, but this variable can be changed to any appropriate puncturing function. For details on the puncturing function, refer to the TCOP section of the *MSC8126 Reference Manual*.
InterruptEnable	Determines whether the interrupt triggered by the TCOP End of Operation flag is enabled. This variable can be set to INTERRUPT_DISABLED or INTERRUPT_ENABLED. The TCOP driver enables the TCOP interrupt by setting this variable to INTERRUPT_ENABLED. If the interrupt is disabled by setting this variable to INTERRUPT_DISABLED, the TCOP End of Operation bit must be polled to determine when TCOP operation completes.
SoftOutput	Determines whether the TCOP generates only hard decision results or soft ones as well. When this variable is set to DUMP_SOFT_DECISION, the SRBA is initialized and allocated a correct amount of memory. The TCOP driver does not generate the soft decision results and sets this variable to DONT_DUMP_SOFT_DECISION.
FirstMapDecoder	Specifies whether the TCOP starts with MAP1 without an interleaver or MAP2 with an interleaver. The TCOP driver sets this variable to MAP1_FIRST. Setting this variable to MAP2_FIRST is useful for stopping the TCOP in the middle of an iteration and then continuing. However, the LBA area must be initialized to Ys, which the TCOP driver does not do.

Table 2. TCOP Driver Files (Continued)

File	Description
StopCondition	Determines when the TCOP stops processing. The TCOP driver sets the stop condition to normal mode by setting this variable to STOP_NORMAL_MODE. It then executes the number of iterations set by the `usiNumDecodeIterations` parameter. This variable can also be set to STOP_POWER_SAVING_MODE. In the power saving mode, the TCOP stops if `usiNumDecodeIterations` number of iterations completes, but it may stop beforehand if both the other stop criteria are met: • A minimum number of iterations have already completed. • The threshold stop condition is met; that is, if there is a given number (or less) of soft lambdas smaller in absolute value than a given threshold.
usiMinInterations	Defines the minimum number of turbo iterations to run when the stop condition is set to power savings. Each turbo decoding iteration contains two MAP decoders, and the TCOP is programmed with the number of MAP decoders. Thus, the TCOP is programmed with two times `usiMinInterations`. The TCOP driver does not use this parameter when the stop condition is set to normal mode. However, when the stop condition is changed to power savings, the minimum number of iterations can be changed to any value up to 15.
stSCR_REG.LTC	Defines the number of soft lambdas allowed for meeting the stop condition (less than or equal to the threshold value). The TCOP driver does not use this parameter when the stop condition is set to normal mode. However, when the stop condition is changed to power savings, the minimum number of iterations can be changed to any value up to 256.
stSCR_REG.LTV	Defines the stop condition absolute threshold value. The TCOP driver does not use this parameter when the stop condition is set to normal mode. However, when the stop condition is changed to power savings, the threshold value must match the format of the other inputs, as described in the TCOP chapter of the *MSC8126 Reference Manual*.
uliCodeBlkLen	Defines the size of the data block to be processed. The block size is the number of original symbols, excluding the tail bits and the parity symbols. That number is also equal to the number of words in the interleaved offset table. The TCOP driver processes two data sets with different values for uliCodeBlkLen: 1312 for the first data set and 3856 for the second data set (not shown in **Example 1**). This variable can be changed to any value up to 32,767.

Example 1. TCOP Parameter Settings Code in main.c

```
/* Setup TCOP Parameters */
usiNumDecodeIterations = 4;          /* =MaxMap/2 */
usiInverseRate = 3;                  /* 2, 3, 4, or 5 */
MaxApproximation = MAX_STAR_LOG_MAP; /* MAX_STAR_LOG_MAP or MAX_LOG_MAP */
iCorrectionFactor = 0x30;            /* Use when MaxApprox = MAX_STAR_LOG_MAP */
/*iCorrectionFactor = (signed int) MINUS_INFTY_LI;*//* Use when MaxApprox = MAX_LOG_MAP */
Puncturing = INVERT_RATE_3;          /* INVERT_RATE_3 or INVERT_RATE_2 or INVERT_RATE_4 */
InterruptEnable = INTERRUPT_ENABLED; /* INTERRUPT_DISABLED or INTERRUPT_ENABLED */
SoftOutput = DONT_DUMP_SOFT_DECISION;/* DONT_DUMP_SOFT_DECISION or DUMP_SOFT_DECISION */
FirstMapDecoder = MAP1_FIRST;        /* MAP1_FIRST or MAP2_FIRST */
StopCondition = STOP_NORMAL_MODE;    /* STOP_NORMAL_MODE or STOP_POWER_SAVING_MODE */
usiMinInterations = 0;               /* =MinMap/2 Only used when Stop Condition=PowerSavings */
stSCR_REG.LTC = 0;                   /* Only used when Stop Condition = PowerSavings */
stSCR_REG.LTV = 0;                   /* Only used when Stop Condition = PowerSavings */

.
.
.

                uliCodeBlkLen    =   1312;
```

5.2 TCOP Interrupt Handler

The TCOP interrupt handler routine resides in the `TCOPInterrupts.c` file and is shown in **Example 2**. This code simply sets a flag to signal that the TCOP has finished processing data, disables the TCOP interrupt, and clears any pending interrupt requests. This routine can be modified to complete any necessary processing after the TCOP is finished, such as triggering a DMA session to move the output data off the MSC8126 device or starting another TCOP session.

Example 2. TCOP Interrupt Handler Code in TCOPInterrupts.c

```
void TCOPINT_Handler()
{
        tcop_done = 0x1;
        DISABLE_TCOP_INTERRUPTS_LIC();
        qbus_place_holder->LICAISR = (UWord32) 0x04000000;        /* Clear pending request */
        qbus_place_holder->IPRA = (UWord32) 0x00000040;
}
```

NOTES:

NOTES:

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.: AN2990
Rev. 0
7/2005