Application Note

AN2368/D
Rev. 0, 10/2002

# Using the Hall Effect Decode (HALLD) TPU Function with the MPC500 Family

Ken Terry
TECD

This TPU Programming Note is intended to provide simple C interface routines to the Hall effect decode function (HALLD). [1] The routines are targeted for the MPC500 family of devices, but they should be easy to use with any device that has a TPU.

## 1 Functional Overview

The Hall effect decode function is a TPU input function that uses two or three channels to decode the signals from hall effect sensors into a state number. It has been primarily designed for use with the COMM TPU function in brushless motor applications but can also be applied to other applications requiring the decoding of multiple digital inputs.

## 2 Detailed Description

The HALLD function uses two or three adjacent TPU channels configured as inputs. The choice of two or three-channel mode is made during initialization. The primary purpose of this function is to decode the digital signals derived from hall effect sensors in a brushless motor, along with a direction input from the CPU, into a state number that is passed to the commutation output TPU function (COMM) via a link request. The state number therefore represents the current angular position of the rotor. In response to the link, the COMM function outputs the commutation signals corresponding to this state in order to turn the motor in the required direction. A PWM function is also required, the output of which is gated by the COMM signals onto the motor phases. Figure 1 shows such an application with the associated signals.

[1]The information in this Programming Note is based on TPUPN10. It is intended to compliment the information found in that Programming Note.

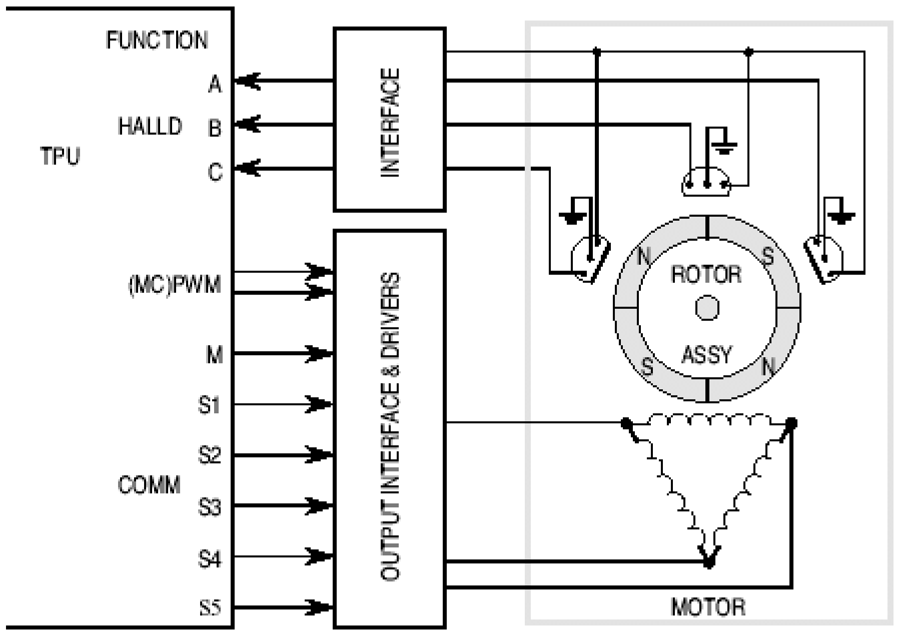

**Figure 1. Hall Effect Sensor Application**

The direction input from the CPU is supplied to the function via the parameter RAM of the HALLD channel A (the one with the lowest channel number). The function effectively performs a 3 to 8 or 4 to 16 decode with the CPU direction input being the third or fourth input depending on the mode. In two channel mode, the output of the function is a state number from 0 to 7 and in three channel mode the output is a state number from 0 to 15. Table 1 and Table 2 show the state numbers produced by HALLD for the various input conditions in the two modes - channel A is the HALLD channel with the lowest channel number, channel B the next lowest and channel C the highest (3 channel mode only).

**Table 1. HALLD Decoding Result in 2-Channel Mode**

| Channel B | Channel A | DIRECTION | Decoded STATE_NO |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 2 |
| 1 | 0 | 0 | 4 |
| 1 | 1 | 0 | 6 |
| 0 | 0 | 1 | 1 |
| 0 | 1 | 1 | 3 |
| 1 | 0 | 1 | 5 |
| 1 | 1 | 1 | 7 |

Table 2. HALLD Decoding Result in 3 Channel Mode

| Channel C | Channel B | Channel A | DIRECTION | Decoded STATE_NO |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 2 |
| 0 | 1 | 0 | 0 | 4 |
| 0 | 1 | 1 | 0 | 6 |
| 1 | 0 | 0 | 0 | 8 |
| 1 | 0 | 1 | 0 | 10 |
| 1 | 1 | 0 | 0 | 12 |
| 1 | 1 | 1 | 0 | 14 |
| 0 | 0 | 0 | 1 | 1 |
| 0 | 0 | 1 | 1 | 3 |
| 0 | 1 | 0 | 1 | 5 |
| 0 | 1 | 1 | 1 | 7 |
| 1 | 0 | 0 | 1 | 9 |
| 1 | 0 | 1 | 1 | 11 |
| 1 | 1 | 0 | 1 | 13 |
| 1 | 1 | 1 | 1 | 15 |

The state number output of the HALLD function can be stored in any location in TPU PRAM via a user programmed pointer. When used with the COMM function the pointer is programmed to store the state number in the COMM master channel PRAM. Note that the HALLD function performs a straight decode and does not reject the invalid sensor states found in some motor applications - these must be handled by correct set-up of the COMM function state table- see 'Notes on Use and Performance of HALLD' and COMM function documentation for more details.

Other uses of the HALLD function are possible, such as reducing the I/O requirements of the main CPU. For example, many applications offer the user a series of switches to set up options. Normally these switches require one I/O line each so that 8 options use 8 input pins on an I/O port (or three switches plus some CPU overhead). In many applications I/O is at a premium and the whole TPU may not be fully utilized for timing tasks. In these cases, with the HALLD function, the number of switches can be reduced to 3 and placed on spare TPU pins. The TPU will decode the required option number, which the CPU can read at any time. In this mode the HALLD function would be programmed to store the state number in a spare PRAM location of one of its own channels and therefore link to itself. A pinstate parameter for each HALLD channel is also maintained by the TPU and this can be read directly by the CPU to determine the level of that channel pin after the last transition on that channel.

## 2.1 HALLD Routines

This Programming Note describes a number of routines which can be used to provide a simple and easy to use interface. There are seven routines in total, and these are contained in two files – tpu_halld.h and tpu_halld.c. The routines are defined in tpu_halld.c and the function prototypes are contained in tpu_halld.h. The file tpu_halld.h needs to be included in any file that uses the routines.

The routines are written in C and examples of their use are provided. The examples make use of the standard header files provided for the MPC555 and the MPC500 utility routines.

The following routines are described in detail in the sections below:

- void tpu_halld_init(struct TPU3_tag *tpu, UINT8 channel, INT16 direction, INT16 state_no_address, UINT8 mode);
- void tpu_halld_enable(struct TPU3_tag *tpu, UINT8 channel, UINT8 priority)
- void tpu_halld_disable(struct TPU3_tag *tpu, UINT8 channel
- void tpu_halld_set_state_no_address(struct TPU3_tag *tpu, UINT8 channel, INT16 state_no_address)
- void tpu_halld_set_direction(struct TPU3_tag *tpu, UINT8 channel, INT16 direction)
- INT16 tpu_halld_get_state_no(struct TPU3_tag *tpu, UINT8 channel)
- INT16 tpu_halld_get_pinstate(struct TPU3_tag *tpu, UINT8 channel)

### 2.1.1 void tpu_halld_init

This function is used to set up and initialize the selected channels for the HALLD function. In order to avoid any unpredictable operation of the TPU, the function disables the selected channels before configuring them for the HALLD function. Clearing the appropriate bits in the CPRO registers disables TPU channels. If any function state is executing at the time a channel is disabled, it will continue until it is completed. It is therefore advisable that, if the channels are used for any other TPU function prior to their configuration for HALLD, an appropriate delay is implemented before calling the tpu_halld_init function.

The function has five parameters:

- *tpu – This is a pointer to the TPU3 module to be used. It is of type TPU3_tag which is defined in the standard header file m_tpu3.h
- channel – This is the channel selected as channel A for the HALLD function. HALLD uses either two or three channels depending on the mode selected. These channels are adjacent and follow channel A.
- direction – This is a 16 bit parameter used by the HALLD function. In a brushless motor application, direction allows the direction of rotation to be specified as it decodes the lsb of the decoded state number. The parameter has two legal values, 0x0000 and 0x0001, defined as HALLD_DIRECTION_0 and HALLD_DIRECTION_1 in halld.h. The translation of these values into motor direction is dependent on the programming of the state table in the COMM function. The parameter is used only by channel A.
- state_no_address – This defines which parameter RAM address is used to store the decoded STATE_NO. The channel associated with the address also receives a link request each time STATE_NO is written by HALLD. Although state_no_address is a 16 bit integer value, only the lower 8 bits are used. These represent the lower eight bits of address within the parameter RAM where STATE_NO is written. For example the value 0x0032 selects parameter 1 of channel 3.
- mode – This selects either two channel or three channel mode of operation. The values HALLD_TWO_CHANNEL_MODE and HALLD_THREE_CHANNEL_MODE are defined in halld.h. The value of mode is stored in the parameter RAM (parameter 2) of the channel assigned as channel A. It is important that this location is not overwritten by the CPU or by any other TPU function.

### 2.1.2 void tpu_halld_enable

This function is used to enable the channels assigned to the HALLD function and is called after tpu_halld_init to enable HALLD operation.

The function has three parameters.

- *tpu - This is a pointer to the TPU3 module to be used. It is of type TPU3_tag, which is defined in m_tpu3.h
- channel – This is the channel selected as channel A for the HALLD function. The parameter should always be Channel A for both two and three channel mode.
- priority – This parameter selects the priority to be assigned to the channels. There are three priority levels. These are defined as TPU_PRIORITY_HIGH, TPU_PRIORITY_MIDDLE and TPU_PRIORITY_LOW in the header file mpc500_util.h.

### 2.1.3 void tpu_halld_disable

This function is used to disable the channels assigned to the HALLD function. It has two parameters.

- *tpu - This is a pointer to the TPU3 module to be used. It is of type TPU3_tag, which is defined in m_tpu3.h
- channel – This is the channel selected as channel A for the HALLD function. The parameter should always be Channel A for both two and three channel mode.

### 2.1.4 void tpu_halld_set_state_no_address

This function is used to set the state_no_address parameter. This defines the location in the TPU parameter RAM where STATE_NO is written. The function has three parameters.

- *tpu - This is a pointer to the TPU3 module to be used. It is of type TPU3_tag which is defined in m_tpu3.h
- channel – This is the channel selected as channel A for the HALLD function. The parameter should always be Channel A for both two and three channel mode.
- state_no_address – This defines which parameter RAM address is used to store the decoded STATE_NO. The channel associated with the address also receives a link request each time STATE_NO is written by HALLD. Although state_no_address is a 16 bit integer value, only the lower 8 bits are used. These represent the lower eight bits of address within the parameter RAM where STATE_NO is written. For example the value 0x0032 selects parameter 1 of channel 3.

### 2.1.5 void tpu_halld_set_direction

This function sets the direction parameter. It has three parameters.

- *tpu – This is a pointer to the TPU3 module to use. It is of type TPU3_tag which is defined in m_tpu3.h
- channel – This is the channel selected as channel A for the HALLD function.

- direction – This is a 16 bit parameter used by the HALLD function. In brushless motor control applications it allows the CPU to specify the direction of rotation dependant on the state table within the COMM function which is used in conjunction with HALLD. The parameter has two legal values, 0x0000 and 0x0001, defined as HALLD_DIRECTION_0 and HALLD_DIRECTION_1 in halld.h

### 2.1.6 INT16 tpu_halld_get_state_no

This function returns the decoded STATE_NO value. It has two parameters.

- *tpu – This is a pointer to the TPU3 module to use. It is of type TPU3_tag which is defined in m_tpu3.h
- channel – This is the channel selected as channel A for the HALLD function.

### 2.1.7 INT16 tpu_halld_get_pinstate

This function returns the PINSATE value for the selected HALLD channel. It has two parameters.

- *tpu – This is a pointer to the TPU3 module to use. It is of type TPU3_tag, which is defined in m_tpu3.h
- channel – This is can be any one of the channels selected for the HALLD function.

# 3 Performance and Use of the HALLD Function

## 3.1 Speed

Like all TPU functions, the performance limit of the HALLD function in a given application is dependent on the service time (latency) of other active TPU channels. This is due to the operational nature of the scheduler. When HALLD is the only function running on the TPU, it can decode state changes at approximately 540KHz in two channel mode and 470 kHz in three channel mode with a CPU bus speed of 40 MHz.

When more TPU channels are active, this performance will be degraded. The scheduler however, assures that the worst case latencies in any TPU application can be closely estimated. It is therefore recommended that the guidelines given in the TPU reference manual are used, along with the figures given in the HALLD state timing table, to perform an analysis on any proposed TPU application that appears to approach the performance limits of the TPU.

**Table 3. Hall Effect Decode Function—State Timing**

| State Number & Name | Max. CPU Clock Cycles | RAM accesses by TPU |
|---|---|---|
| S1 INIT_2CH_HALLD | 60 | 8 |
| S2 INIT_3CH_HALLD | 74 | 10 |
| S3 TRANS_HALLD<br>i) 2 Channel Mode<br>ii) 3 Channel Mode | <br>56<br>70 | <br>8<br>10 |

NOTE: Execution times do not include the time slot transition time (TST= 10 or 14 CPU clocks)

## 3.2 Using HALLD with the COMM TPU Function

HALLD has been primarily designed for use with COMM in a brushless motor application. Observing the following points will greatly improve the performance of the combination.

During initialization of the HALLD function, a link request to the COMM function will be generated when each channel is initialized. This means that in a 3-channel case, COMM would receive 3 links. In addition STATE_NO is only valid after all the channels have been initialized so that the first two links are associated with an invalid STATE_NO. To avoid COMM responding to these links, it is best to initialize HALLD first and then initialize COMM after all HALLD HSRs have been cleared. In this way COMM will use a valid STATE_NO from initialization onwards.

In some motor applications, not all sensor states are valid. For example in a 3 phase brushless motor the three hall effect sensors produce only six valid states, but the other two can occur momentarily due to noise. Since HALLD performs a straight binary decode of the sensor inputs and does not reject invalid states, these erroneous states can be passed on to the COMM function. The COMM function allows for this eventuality via the programmable state table. The entries in the state table corresponding to the invalid states should be configured to take the appropriate action (usually to turn off the phase drivers). Refer to the Programming Note “Multiphase Motor Commutation TPU Function (COMM)” (TPUPN09/D) for more detail.

## 3.3 Using HALLD for General Purpose Input Pins

As previously mentioned, HALLD can be used as a type of input port. In this configuration STATE_NO_ADDR is programmed to store STATE_NO into one of the spare parameter RAM locations of one of the HALLD channels. This means that a HALLD channel will receive the link request, but link requests are ignored by the function. There are two ways of using HALLD for general-purpose inputs. The first, where the inputs are used in encoded form, has already been described. HALLD can be used more simply to read the state of the pins directly, using the PINSTATE parameter and ignoring the value of STATE_NO. After initialization the CPU can read the PINSTATE parameter of the channels at any time (using function tpu_get_pinstate) and obtain the level of the channel pin after the last edge was serviced.

## 3.4 Example 1

The following example shows how to set up and configure the HALLD function using the interface routines. Once the HALLD function has been configured the program runs a small loop of code which fetches the decoded state_no value.

### 3.4.1 Program

```
/***************************************************************************/
/* FILE NAME: halld_ex1.c                     COPYRIGHT (c)  2002 */
/* VERSION: 1.0                                                        */
/*                                                                     */
/* DESCRIPTION: This program is a simple example of using the HALLD      */
/*              interface routine to configure the HALLD function for    */
/*              three-channel mode                                      */
/*=====================================================================*/
/* COMPILER: Diab Data        VERSION: 4.3f                             */
/*                                                                          */
/* HISTORY                                                                  */
/* REV      AUTHOR      DATE       DESCRIPTION OF CHANGE                    */
/* ---   -----------  ---------    ---------------------                    */
/* 1.1   K Terry       25/7/02     Demo program for HALLD setup             */
/*                                                                          */
/****************************************************************************/
#include "mpc555.h"
#include "tpu_halld.h"
#include "mpc500.c"        /* Configuration routines for MPC555 EVB */
#include "mpc500_util.h"   /* Utility routines for using MPC500 devices */


struct TPU3_tag *tpua = &TPU_A;   /* pointer for TPU routines */

void delay (void)
{
          int delay;
          for (delay = 0; delay < 0x100; delay++);
}


void main ()
{

          int state = 0;
          INT16 state_no;

          setup_mpc500(40);       /*Setup device and programm PLL to 40MHz*/


          /* configure TPU channels 3, 4 and 5 to run the HALLD function in
          three-channel mode, state no is stored in the low order eight bits
          of parameter 3 in channel 5 */

tpu_halld_init(tpua, 3, HALLD_DIRECTION_0, 0x0055,
 HALLD_THREE_CHANNEL_MODE);
       tpu_halld_enable(tpua, 3, TPU_PRIORITY_MIDDLE);
```

```
while(1)
{
          state_no = tpu_halld_get_state_no(tpua, 3);
          delay();
}
```

## 3.5 Example 2

The following example shows how the HALLD function can be configured to work with the COMM function (in sensored mode). Table 4 shows a possible state table for a three-phase motor using sensored mode and HALLD.

**Table 4. Example State Table for 3-Phase Motor Using in Sensored Mode and HALLD**

| HALL C | HALL B | HALL A | DIRECTION | STATE_NO | COMM State Table |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | XXXXXXXX XX101100 |
| 0 | 0 | 0 | 1 | 1 | XXXXXXXX XX011010 |
| 0 | 0 | 1 | 0 | 2 | XXXXXXXX XX110100 |
| 0 | 0 | 1 | 1 | 3 | XXXXXXXX XX011001 |
| 0 | 1 | 0 | 0 | 4 | XXXXXXXX XXNNNNNN |
| 0 | 1 | 0 | 1 | 5 | XXXXXXXX XXNNNNNN |
| 0 | 1 | 1 | 0 | 6 | XXXXXXXX XX110010 |
| 0 | 1 | 1 | 1 | 7 | XXXXXXXX XX101001 |
| 1 | 0 | 0 | 0 | 8 | XXXXXXXX XX101001 |
| 1 | 0 | 0 | 1 | 9 | XXXXXXXX XX110010 |
| 1 | 0 | 1 | 0 | 10 | XXXXXXXX XXNNNNNN |
| 1 | 0 | 1 | 1 | 11 | XXXXXXXX XXNNNNNN |
| 1 | 1 | 0 | 0 | 12 | XXXXXXXX XX011001 |
| 1 | 1 | 0 | 1 | 13 | XXXXXXXX XX110100 |
| 1 | 1 | 1 | 0 | 14 | XXXXXXXX XX011010 |
| 1 | 1 | 1 | 1 | 15 | XXXXXXXX XX101100 |

The program configures the HALLD Function to operate in three-channel mode with TPU channel 3 assigned as Channel A.

The COMM Function is set to use 6 pins to drive the commutation states. Sixteen commutation states are defined. TPU channel 8 is assigned as the master channel.

The program includes a small piece of demonstration code which uses the QADC_A PORTQA[0:2] pins to drive state values. If these pins are connected to the HALLD TPU channels, the HALLD function will decode the states and provide a state number to the COMM function, which will in turn drive the appropriate commutation state values on the TPU COMM channel pins.

The user should refer to the Programming Notes “Multiphase Motor Commutation TPU Function (COMM)” and “Using the Multiphase Motor Commutation TPU Function (COMM)” for information on the COMM interface routines.

## 3.6 Program

```
/**************************************************************************/
/* FILE NAME: halld_comm_ex.c               COPYRIGHT (c)  2002     */
/* VERSION: 1.0                                                           */
/*                                                                        */
/* DESCRIPTION: This program demonstrates the use of the COMM and HALLD     */
/*              TPU functions. It sets up HALLD for three-channel           */
/*              operation and configures COMM to use a 16 entry commutation */
/*              state table running in sensored HALLD mode                  */
/*========================================================================*/
/* COMPILER: Diab Data        VERSION: 4.3f                               */
/*                                                                        */
/* HISTORY                                                                */
/* REV      AUTHOR      DATE       DESCRIPTION OF CHANGE                  */
/* ---   -----------  ---------    ---------------------                  */
/* 1.0   K Terry       30/8/02      Demo. Program for TPU COMM/HALLD        */
/*                                  Function                              */
/**************************************************************************/
#include "mpc555.h"
#include "tpu_halld.h"
#include "tpu_comm.h"
#include "mpc500.c"        /* Configuration routines for MPC555 EVB */
#include "mpc500_util.h"   /* Utility routines for using MPC500 devices */


void main ()
{
          struct    TPU3_tag *tpua = &TPU_A;   /* pointer for TPU routines */
          struct    halld_func *halld1;

          INT16 state = 0;
          INT16     state_no;
          INT16     tpu_comm_states[16] = {0x002c, 0x001a, 0x0034, 0x0031,
                                               0x0000, 0x0000, 0x0032, 0x0029,
                                               0x0029, 0x0032, 0x0000, 0x0000,
                                               0x0019, 0x0034, 0x001a, 0x002c
```

```
                                        };

          setup_mpc500(40);       /* Setup device and programm PLL to 40MHz */

/*        initialize tpu comm function for tpu_a,

          master channel is 8
          no_of_pins = 6

          update_period = 37 = 0x0025 - for sensored mode
          UPDATE_PERIOD(min)(CPU clocks) = 64 + 14 * NO_OF_PINS
          UPDATE_PERIOD(min)(TCR1 clocks) = (64 + 14 * NO_OF_PINS) / 4
          - assumes DIV4 clock (TPUMCR3[EPSCKE] = 0, TPUMCR[PSCK] = 0
          and TPUMCR[TCR1P] = 00)
*/
        tpu_comm_init_sensored(tpua, 8, 6, 0x0025, tpu_comm_states, 16);

/*      force STATE_0 for COMM function on tpua, master channel 8 */
        tpu_comm_force_state(tpua, 8, 0);


/*        enable COMM function master channel (8) for middle priority */
          tpu_enable(tpua, 8, TPU_PRIORITY_MIDDLE);

/*        wait for end of service (force_state) */

          while(tpu_get_hsr(tpua, 8)!=0);


/*        set up HALLD function for 3 channel mode using channels 3, 4 and 5
          initial value of DIRECTION = 0, state_no_address is set to 0x0083
          to place the decoded STATE_NO value from HALLD into the required COMM
          parameter RAM location for STATE_NO
*/

        tpu_halld_init(tpua, 3, HALLD_DIRECTION_0, 0x0083,
        HALLD_THREE_CHANNEL_MODE);

        tpu_halld_enable(tpua, 3, TPU_PRIORITY_MIDDLE);
```

```
/* The following loop of code is intended to demonstrate the operation of the
   HALLD and COMM functions operating together. It will drive state values
   onto the QADC_A PORTQA[0:2] pins. If these are connected to the HALLD
   input channels(A, B and C], the HALLD function decodes the state values and
   supplies the decoded state number to the COMM function which in turn drives
   the appropriate commutation state value onto the TPU COMM channels */



          QADC_A.PORTQA.R = 0;
          QADC_A.DDRQA.R = 0xFF00;

          while(1)
          {
                    tpu_halld_set_direction(tpua, 3, HALLD_DIRECTION_0);
                    for (state = 0; state < 8; state++)
                    {
                              QADC_A.PORTQA.R = state;

                              /* wait for completion of COMM function service */
                              while (!((tpua->CISR.R) & (0x0001 << 8)));

                              tpua->CISR.R = 0;
                              state_no = tpu_comm_get_state_no(tpua, 8);
                    }
                    tpu_halld_set_direction(tpua, 3, HALLD_DIRECTION_1);

                    for (state = 0; state < 8; state++)
                    {
                              QADC_A.PORTQA.R = state;

                              /* wait for completion of COMM function service */
                              while (!((tpua->CISR.R) & (0x0001 << 8)));

                              tpua->CISR.R = 0;
                              state_no = tpu_comm_get_state_no(tpua, 8);
                    }
          }
}
```

THIS PAGE INTENTIONALLY LEFT BLANK

THIS PAGE INTENTIONALLY LEFT BLANK

THIS PAGE INTENTIONALLY LEFT BLANK

***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 Muenchen, 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
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

freescale™ semiconductor