Application Note

AN2367/D
Rev. 0, 10/2002

Using the Multiphase Motor Commutation TPU Function (COMM) with the MPC500 Family

Ken Terry
TECD

This TPU Programming Note is intended to provide simple C interface routines to the multiphase motor commutation TPU function (COMM). [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 Commutation (COMM) TPU function uses multiple TPU channels to produce the drive enable signals necessary for commutating brushless motors. Motor types supported include three- and four-phase brushless d.c. and three-phase switched reluctance motors. The COMM function is used in conjunction with another TPU function to provide a choice of sensored (Hall effect or optical) or sensorless (from an encoder) commutation. The signals produced by COMM are externally gated with a PWM, also generated by the TPU, to drive the motor. The COMM function has been optimized for flexibility and this may allow it to meet the requirements of other multi-signal applications.

2 Detailed Description

The COMM TPU function produces the commutation drive signals for a variety of brushless motor types. The function uses up to eight adjacent TPU channels (one master channel and from one to seven slave channels) to generate the commutation signals. For example, six channels can be used for a three-phase brushless d.c. motor, or four channels can be used for a four-phase motor. All TPU service activity takes place on the master channel - the slaves are used only as synchronized output pins. When a commutation state change occurs, the pin state change on each output channel is synchronized using an output match based on the TCR1 counter.

COMM has been designed to work with several TPU input functions to provide a choice of sensored or sensorless commutation. To support these input functions, COMM has two basic operating modes selected by the host sequence bits on the master channel:

[1] The information in this Programming Note is based on TPUPN09. It is intended to compliment the information found in that Programming Note.

2.1 Sensored Mode

Sensored Mode is designed to work with the Hall effect decode (HALLD) TPU function. HALLD is an input function that decodes either two or three Hall effect or optical sensors, and a CPU-supplied direction input, into a state number. HALLD writes this state number into the parameter RAM of the COMM function master channel and then issues a link to that channel. On receipt of the link, COMM obtains the output pin configuration that corresponds to the new commutation state from a table in parameter RAM, and subsequently outputs the new commutation state. The user has complete control over the commutation sequence via the programmable state table in the COMM function and a CPU force feature which allows the CPU to force a particular commutation state. Figure 1 shows a typical Hall effect set-up and control waveforms.

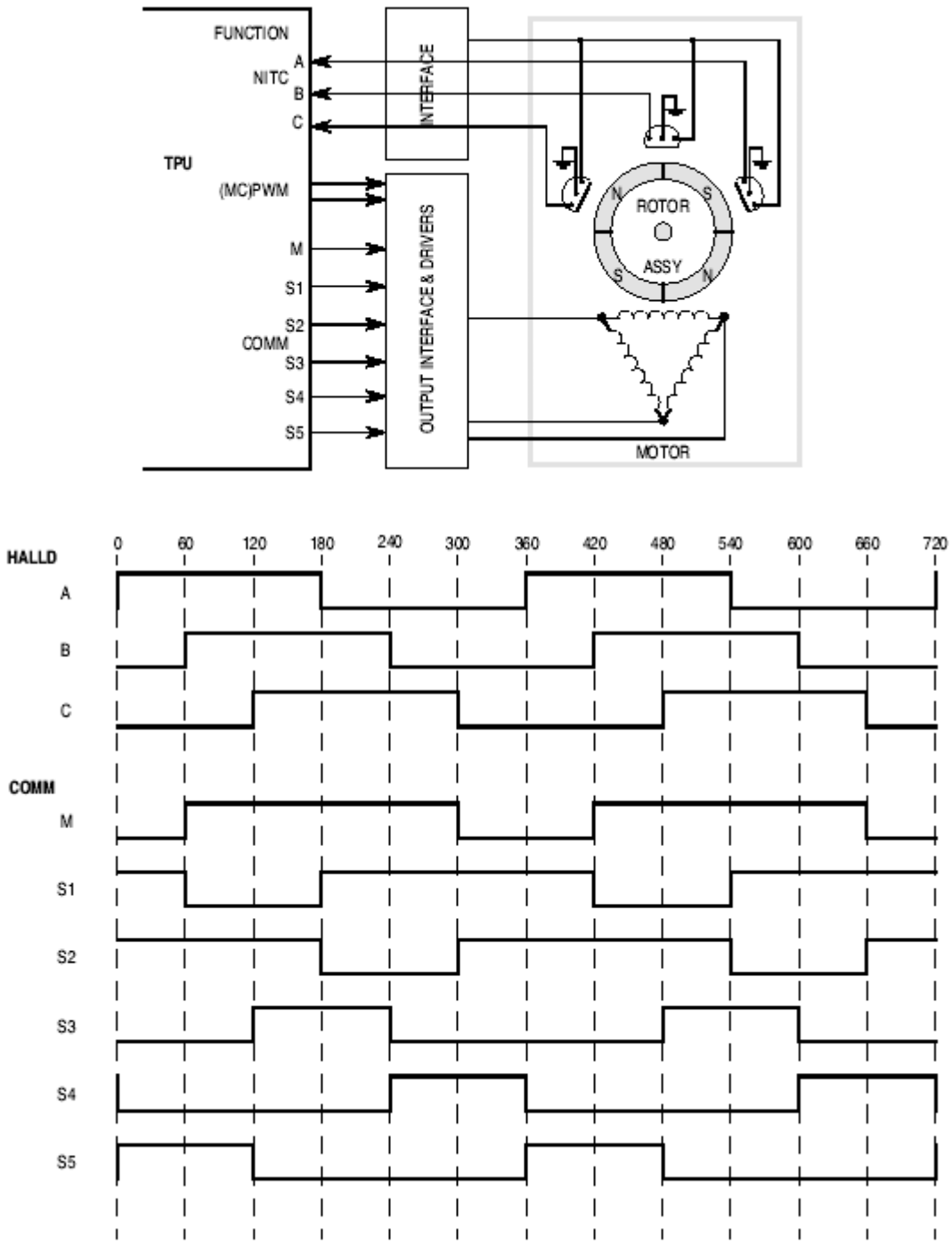

Figure 1. Typical Hall Effect Setup and Waveforms

2.2 Sensorless Mode

Sensorless Mode is designed to support sensorless commutation from a high resolution encoder on the shaft of the motor. To keep the COMM function as flexible as possible, the sensorless mode of operation has been

implemented as a programmable state machine. The basis for the production of the commutation signals is a counter representing the angular position of the motor. This counter is the position count output of a TPU input function such as Quadrature Decode (QDEC) or Fast Quadrature Decode (FQD), which is derives angular position from the shaft encoder. Figure 2 shows typical shaft encoder set-up and control waveforms.

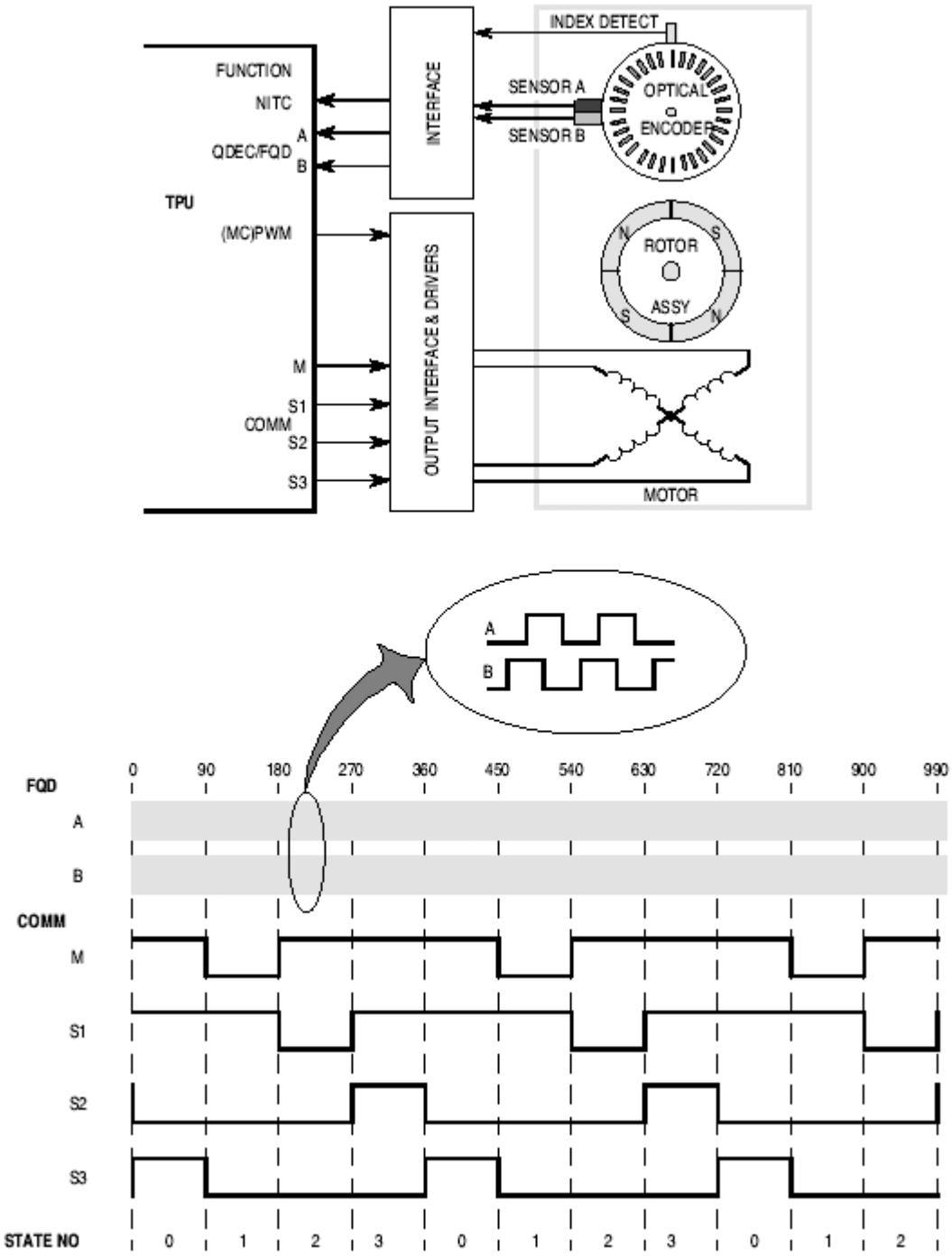

Figure 2. Typical Sensorless Set-up and Waveforms

The COMM function directly accesses this counter, located anywhere in parameter RAM, and performs tests on it to determine the required state of the commutation output pins. This process is carried out without CPU intervention. The COMM function maintains upper and lower angular limits to the current state in parameter RAM. On each service of the COMM function, the position counter is compared with these limits. If the position count has passed either limit, the state number is updated and a new state parameter is obtained from a circular table in parameter RAM. The state parameter contains the pin configuration and length of the new state in position counts. New upper and lower angular limits are calculated using the length of the new state and then stored for use in subsequent state tests. The new pin configuration is subsequently output on the COMM channels.

As an additional feature of sensorless operation, a CPU supplied angular offset is added to the position count before the limit tests. This parameter, which can be updated at any time, allows the CPU to advance or retard previously programmed state switching angles. The offset parameter can be used to start the motor in a particular direction, to partially compensate for TPU service latencies, to maintain torque at high motor speeds, and to force braking on the motor.

The number of states in the commutation sequence, the length in angular position counts of each state, the number of channels used for commutation, and the pin states for each state are all independently programmable by the user. These capabilities make the function suitable for a wide variety of commutation schemes.

On the original TPU, the layout of the parameter RAM places some restrictions on the maximum table size, dependent on which channel is selected as the master channel. These restrictions are described in TPUPN09. The TPU3 on the MPC555, with eight parameters per channel does impose the same restrictions and the maximum table size does not depend on which channel is selected as the master.

Up to eight TPU channels can be used as COMM signal outputs (including the master channel), and each state has a length which is individually programmable over an 8 bit range in position counts. In applications using an encoder with a very high resolution, an 8-bit range for the state length in position counts may not be sufficient. COMM allows the use of multiple state table entries programmed to have the same output pin configuration, to effectively lengthen the state.

In sensorless operation, the COMM function has been designed to update the commutation signals on a periodic basis, and there are two "sub-modes" of operation. These are Match Update and Link Update. Only Match Update is supported by the C interface routines.

In this mode, a user programmable periodic match on the master COMM channel is used to invoke updating of the commutation signals. Assuming an accurate position encoder, the accuracy of commutation is dependent on this periodic update rate. A faster update rate results in more accurate commutation, but a higher TPU overhead. This mode is for use with the QDEC or FQD TPU input functions.

2.3 All Modes

The host CPU can force any commutation state, to put the motor stator field into a known configuration, at any time. The CPU can also interrogate the COMM function at any time to determine which state is currently active. Since the state table is programmed during initialization, the user can decide which state number corresponds to which output pin configuration. This flexibility allows a variety of commutation schemes to be implemented.

To drive the motor, the outputs of the COMM function TPU channels will be used to gate a PWM generated on another TPU channel onto the motor phase drivers.

The TPU is flexible enough to support an application with Hall effect or optical sensors for commutation and an encoder on the motor shaft for deriving speed and direction information. This may be beneficial in environments where encoder information can not be relied upon to be completely accurate.

2.4 COMM Routines

This programming note describes a number of routines which can be used to provide a simple and easy to use interface. There are eight routines in total, and these are contained in two files – tpu_comm.h and tpu_comm.c. The routines are defined in tpu_comm.c and the function prototypes are contained in tpu_comm.h. The file tpu_comm.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_comm_init_sensored (struct TPU3_tag *tpu, UINT8 channel, INT16 no_of_pins, INT16 update_period, INT16 comm_states[], INT16 no_of_states)
- void tpu_comm_force_state (struct TPU3_tag *tpu, UINT8 channel, INT16 state_no)
- void tpu_comm_init_sensorless_match (struct TPU3_tag *tpu, UINT8 channel, INT16 no_of_pins, INT16 counter_addr, INT16 update_period, INT16 comm_states[], INT16 no_of_states)
- void tpu_comm_write_upper (struct TPU3_tag *tpu, UINT8 channel, INT16 upper)
- void tpu_comm_write_lower (struct TPU3_tag *tpu, UINT8 channel, INT16 lower)
- void tpu_comm_write_offset (struct TPU3_tag *tpu, UINT8 channel, INT16 offset)
- INT16 tpu_comm_get_state_no (struct TPU3_tag *tpu, UINT8 channel)
- void tpu_comm_start_update (struct TPU3_tag *tpu, UINT8 channel)

2.4.1 void tpu_comm_init_sensored

This function is used to set up and initialize the selected TPU channels for the COMM function in Sensored (HALLD) mode. In this mode the COMM function works in conjunction with the HALLD function, which must also be set up and enabled on the TPU.

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

The function has six 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 the master channel for the COMM function. The number of pins required for commutation determines the number of channels used by COMM. The additional required channels are adjacent to, and follow, the master channel in the TPU.

- no_of pins - This parameter determines the number of channels to be used to generate the commutation signals (including the master channel). The maximum number of channels that can be used is eight. To set up the COMM function for three-phase commutation, using six channels, no_of_pins should be 0x06.
- update_period – This parameter is used to schedule a match on the COMM channels. This ensures that, each time a link is received by the master channel, the new pin states occur simultaneously on multiple channels. update_period should be set to a value which allows the pins to change state just after completion of service on the master channel. Table 1 shows the state timing for the Commutation Function.
- comm_states[] – This is an array of values containing the PIN CONFIG values for each commutation state. The lower eight bits of each array element make up the PIN CONFIG field. This is right justified - if six pins are used for commutation, the lower six bits of the field are used to hold the pin values. A logic one in the PIN CONFIG field will result in the corresponding pin being driven low, and a logic zero will result in the pin being driven high.
- no_of_states – This parameter indicates the number of states used in the commutation sequence. Note that, although the NO_OF_STATES parameter in the TPU parameter RAM is not used by the COMM function in sensored mode, no_of_states is required to indicate the size of the comm_states[] array passed to the function.

2.4.2 void tpu_comm_force_state

This function causes the selected output state to be driven on the TPU pins. The PIN CONFIG field within the selected comm_states array element determines the levels driven onto the COMM channel pins. 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 the master channel for the COMM function.
- state_no – This parameter determines which comm_states array element provides the PIN CONFIG field to drive the pin levels.

2.4.3 void tpu_comm_init_sensorless_match

This function is used to set up and initialize the selected TPU channels for the COMM function in Sensorless (FQD or QDEC) mode. In this mode the COMM function works in conjunction with the FQD or QDEC functions, which must also be set up and enabled on the TPU.

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

The function has seven 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 the master channel for the COMM function. The number of pins required for commutation determines the number of channels used by COMM. The additional required channels are adjacent to, and follow, the master channel in the TPU.
- no_of pins - This parameter determines the number of channels to be used to generate the commutation signals (including the master channel). The maximum number of channels that can be used is eight. To set up the COMM function for three-phase commutation, using six channels, no_of_pins should be 0x06.
- counter_addr – This parameter is used to indicate the address in parameter RAM of the angular position counter that is used as the basis of the state tests in sensorless mode. For example, if COMM is used in conjunction with FQD, and channel 4 is used as the master channel for FQD, then counter_addr would be set to 0x0042 to select POSITION_COUNT in FQD's parameter RAM as the angular position counter.
- update_period – In sensorless mode this parameter determines the frequency of update (in TCR1 counts) of the commutation signals. It is used to schedule a periodic match on the master channel.
- comm_states[] – This is an array of values containing the LENGTH and PIN CONFIG values for each commutation state. LENGTH comprises the upper eight bits of an array element and defines the length (in position counter increments) of each state. PIN CONFIG comprises the lower eight bits of an array element. This is right justified. If six pins are used for commutation, the lower six bits of the field are used to hold the pin values. A logic one in the PIN CONFIG field will result in the pin being driven low, and a logic zero will result in the pin being driven high.
- no_of_states – This parameter indicates the number of states used in the commutation sequence.

2.4.4 void tpu_comm_write_upper)

This function sets the upper angular boundary of the current commutation state. After the CPU has completed forcing states during the motor start-up sequence, upper should be written to an appropriate value based on the position count value and the current state length. After this has been done the TPU will automatically update the upper angular boundary value each time a state transition occurs. 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 the master channel for the COMM function.
- upper – This value contains the value of the upper angular boundary of the current commutation state (in position counts).

2.4.5 void tpu_comm_write_lower

This function sets the lower angular boundary of the current commutation state. After the CPU has completed forcing states during the motor start-up sequence, lower should be written to an appropriate value based on the position count value and the current state length. After this has been done the TPU will automatically update the lower angular boundary value each time a state transition occurs. 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 the master channel for the COMM function.

- lower – This value contains the value of the lower angular boundary of the current commutation state (in position counts).

2.4.6 void tpu_comm_write_offset

This function sets the offset, which is used by the TPU to advance or retard all state switching angles on the fly. 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 the master channel for the COMM function.
- offset – This value contains the value of the offset in position counts. This is added to the position counter value to provide a new value that is used in the position limit tests.

2.4.7 INT16 tpu_comm_get_state_no

This function returns a value state_no, which represents the current commutation state. The function 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 the master channel for the COMM function.

2.4.8 void tpu_comm_start_update

This function starts the sensorless match update sequence. It should be called once the initial commutation state has been set and the upper, lower and offset parameters written. The function 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 the master channel for the COMM function.

2.5 Function Configuration

The steps necessary for the CPU to configure COMM depend on the mode of operation selected. The following describes the required sequence of steps:

Sensored Mode

1. 1. Call *tpu_comm_init_sensored* to select and set up the required TPU channels for the COMM function, and initialize the comm_states array and update_period.
2. 2. Call *tpu_comm_force_state* to set the COMM channels to the required initial commutation-state.
3. 3. Call *tpu_enable* (defined in mpc500.c and supplied with the standard header files) to set the TPU master COMM channel to the required priority.
4. 4. Wait for the appropriate bits in the TPU HSRR0 register to clear. This can be checked by calling *tpu_get_hsr* (defined in mpc500.c and supplied with the standard header files).
5. 5. Initialize the HALLD function.
6. 6. The HALLD function now supplies link requests along with the required STATE_NO value when a commutation update is required.

Sensorless Match Update Mode

1. Call *tpu_comm_init_sensorless_match* to select the required TPU channels for the COMM function and initialize *counter_addr*, the *comm_states* array, *update_period* and *no_of_pins*.
2. Initialize the QDEC or FQD function.
3. Call *tpu_comm_force_state* to set the COMM channels to the required initial commutation-state.
4. Call *tpu_enable* (defined in mpc500.c and supplied with the standard header files) to set the TPU master COMM channel to the required priority.
5. Wait for the appropriate bits in the TPU HSRR0 register to clear. This can be checked by calling *tpu_get_hsr* (defined in mpc500.c and supplied with the standard header files).
6. Fetch the position count value from the FQD or QDEC channels and calculate the initial upper and lower parameter values.
7. Call *tpu_comm_upper*, *tpu_comm_lower* and *tpu_comm_offset* to initialise upper, lower and offset.
8. Call *tpu_comm_start_update*. The COMM function now runs in sensorless mode, accessing the QDEC or FQD channels as required.

3 Use and Performance of the COMM Function

Like all TPU functions, the performance limit of the COMM function in a given application is dependent on the service time (latency) of other active TPU channels. This is due to the way the scheduler operates. Worst case latency in any TPU application can be closely estimated. To analyze the performance of an application that appears to approach the limits of the TPU, use the guidelines given in the TPU reference manual, the information from the state time table below and the figures from the state timing tables of the other active TPU functions.

Since COMM will always be used in conjunction with at least two other TPU functions - an input function and a PWM function - the overall proposed system must be analyzed to assess the possible performance.

Section 7 of TPUPN09 provides detailed notes on performance and use of the COMM function.

Table 1. Commutation Function—State Timing

State Number & Name	Max. CPU Clock Cycles	RAM accesses by TPU
S1 FORCE_COMM	24 + (14 * NO_OF_PINS)	5
S2 UPDATE_COMM	64 + (14 * NO_OF_PINS)	16
S3 MATCH_UPDATE_COMM	62 + (14 * NO_OF_PINS)	16

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

3.1 COMM Function Example

The following example shows how the COMM function can be configured to work with the HALLD function (in sensored mode).

Table 2 shows a possible state table for a three-phase motor using sensored mode and HALLD.

Table 2. Example State Table for a 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 Note “Using the Hall Effect Decode TPU Function (HALLD)” for more information on the HALLD interface routines.

3.1.1 Program

```
/*************************************************************************/
/* FILE NAME: 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                              */
```

COMM Function Example

```
/*                                                                          */
/* 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 */

/*        initialise 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);
```

COMM Function Example

```
/*      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
          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

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