Freescale Semiconductor

PROGRAMMING NOTE

Order this document
by TPUPN10/D

Hall Effect Decode TPU Function (HALLD)

by Jeff Wright

1 Functional Overview

The Hall effect decode function is a TPU input function that uses two or three channels to decode signals from Hall effect sensors into a state number. The function is designed primarily for use with the COMM TPU function in brushless motor applications, but it can also be used in 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** and **Figure 2** show a typical application and associated signals.

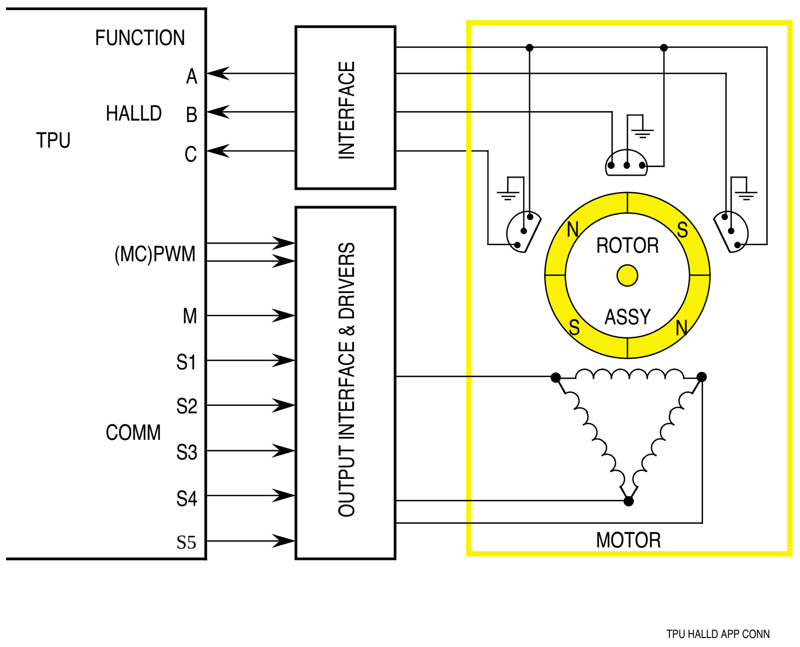


Figure 1 Hall Effect Sensor Application

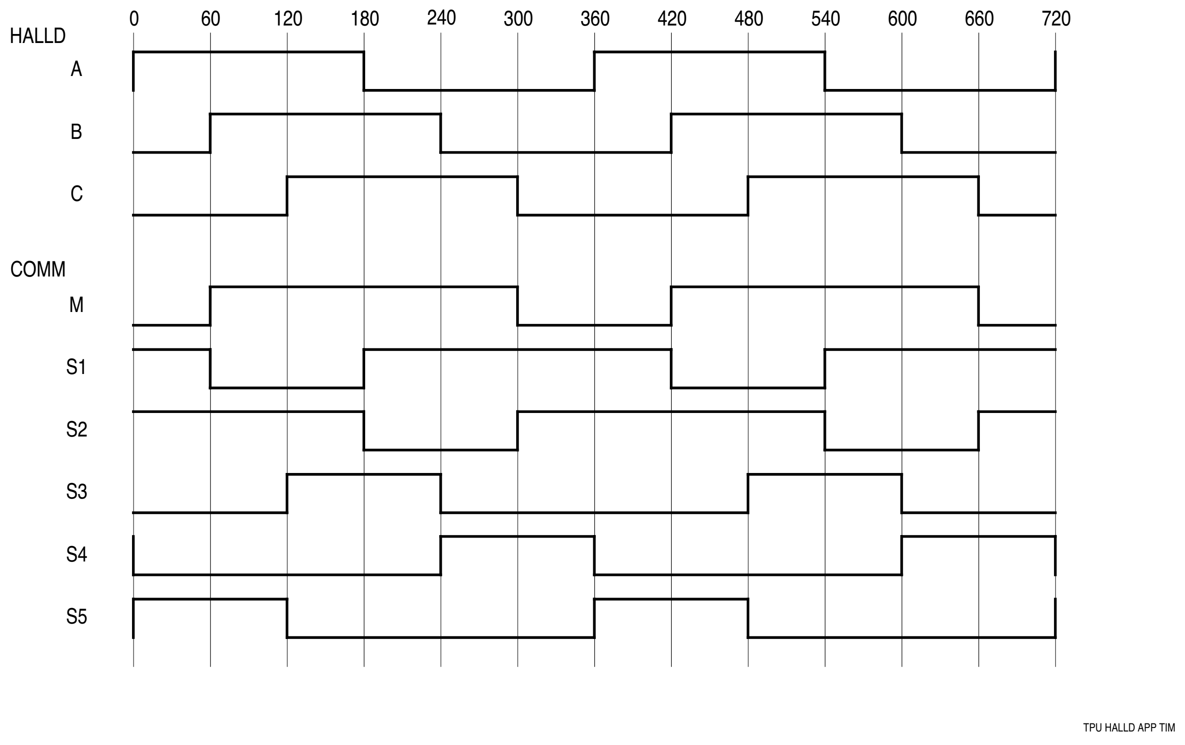


Figure 2 Typical Three Phase Hall Effect Waveforms

The direction input from the CPU is supplied to the function via the parameter RAM of HALLD channel A (the channel with the lowest channel number). The function effectively performs a 3-to-8 or 4-to-16 decode — the CPU direction input is the third or fourth input, depending on the operating mode. In two channel mode, the output of the function is a state number in the range 0 to 7, and in three channel mode the output is a state number in the range 0 to 15. The following tables show 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 has the next lowest number, and channel C has the highest number (three channel mode only).

Table 1 HALLD Decoding In Two 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 In Three 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 parameter RAM 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 parameter RAM. Note that the HALLD function performs a straight decode and does not reject the invalid sensor states found in some motor applications — these states must be handled by correct setup of the COMM function state table. See **7 Performance and Use of Function** in this note and in *Commutation Output TPU Function (COMM)*, Freescale document number TPUPN09/D, for more details.

Other uses of the HALLD function, such as reduction of main CPU I/O requirements, are possible. For example, many applications offer the user a series of switches that are used to set up options. Normally these switches require one I/O line each, so that eight options use eight 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, using the HALLD function can reduce the number of switches to three, and the lines can be connected to spare TPU pins. The TPU can decode the required option number, and the CPU can read it at any time. When used in this way, the HALLD function is programmed to store the state number in a spare parameter RAM location that belongs to one of the function channels, and therefore links to itself. A pin state parameter for each HALLD channel is also maintained by the TPU — this parameter can be read directly by the CPU to determine the level of the channel pin after the last transition on that channel.

3 Function Code Size

Total TPU function code size determines what combination of functions can fit into a given ROM or emulation memory microcode space. HALLD function code size is:

22 µ instructions + 8 entries = **30 long words**

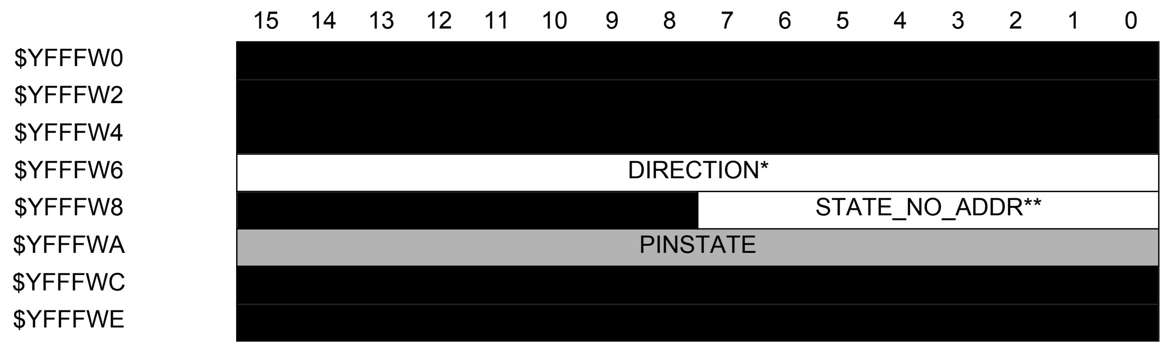
4 Function Parameters

This section provides detailed descriptions of function parameters stored in channel parameter RAM. **Figure 3** shows TPU parameter RAM address mapping. **Figure 4** shows the parameter RAM assignment used by the function. In the diagrams, Y = M111, where M is the value of the module mapping bit (MM) in the system integration module configuration register (Y = $7 or $F).

Channel Number	Base Address	Parameter Address 0	1	2	3	4	5	6	7
0	$YFFF##	00	02	04	06	08	0A	—	—
1	$YFFF##	10	12	14	16	18	1A	—	—
2	$YFFF##	20	22	24	26	28	2A	—	—
3	$YFFF##	30	32	34	36	38	3A	—	—
4	$YFFF##	40	42	44	46	48	4A	—	—
5	$YFFF##	50	52	54	56	58	5A	—	—
6	$YFFF##	60	62	64	66	68	6A	—	—
7	$YFFF##	70	72	74	76	78	7A	—	—
8	$YFFF##	80	82	84	86	88	8A	—	—
9	$YFFF##	90	92	94	96	98	9A	—	—
10	$YFFF##	A0	A2	A4	A6	A8	AA	—	—
11	$YFFF##	B0	B2	B4	B6	B8	BA	—	—
12	$YFFF##	C0	C2	C4	C6	C8	CA	—	—
13	$YFFF##	D0	D2	D4	D6	D8	DA	—	—
14	$YFFF##	E0	E2	E4	E6	E8	EA	EC	EE
15	$YFFF##	F0	F2	F4	F6	F8	FA	FC	FE

— = Not Implemented (reads as $00)

Figure 3 TPU Channel Parameter RAM CPU Address Map



W = Channel number
*Channel A only.
**1 channel only; channel B in 2 channel mode, channel C in 3 channel mode.

Parameter Write Access

⬜	Written by CPU
▫	Written by TPU
▨	Written by CPU and TPU
⬛	Unused parameters

Figure 4 HALLD Function Parameter RAM Assignment

4.1 DIRECTION

This 16-bit parameter is written by the CPU and read by the TPU. In a brushless motor application, DIRECTION allows the CPU to specify the desired direction of rotation as it determines the LSB of the decoded state number. DIRECTION has only two legal values, $0000 and $0001. The translation of these values into motor direction is dependent upon the programming of the state table in the COMM function. The parameter DIRECTION exists only on channel A, the lowest numbered HALLD channel.

4.2 PINSTATE

These 16-bit parameters (one in each channel) are maintained by the TPU and contain a value representing the level of the channel pin when the last edge was serviced. The value $8000 is used to represent a pin high level, and $0000 to represent a pin low level. When a transition on one of the HALLD channels is serviced, only the PINSTATE parameter of that channel is updated. The CPU should not write the PINSTATE parameters while HALLD is running or an erroneous state decode may occur.

4.3 STATE_NO_ADDRE

This parameter, which resides in the upper HALLD channel (B in two channel mode or C in three channel mode) specifies the parameter RAM address of the location that receives the decoded STATE_NO. The channel associated with that address also receives a link request each time STATE_NO is written by HALLD. For example, if STATE_NO_ADDR = $0032 then STATE_NO is written to parameter 1 (second parameter) of channel 3 and a link to channel 3 is generated.

4.4 STATE_NO

This is the output of the HALLD function. STATE_NO is not shown in the parameter diagrams because its location is determined by the contents of STATE_NO_ADDR — in many applications, STATE_NO will reside in a non-HALLD channel. STATE_NO has a range of $0000 to $000F, and all 16 bits are written to the destination parameter RAM address specified by STATE_NO_ADDR.

5 Host Interface to Function

This section provides information concerning the TPU host interface to the function. **Figure 5** is a TPU address map. Detailed TPU register diagrams follow the figure. In the diagrams, Y = M111, where M is the value of the module mapping bit (MM) in the system integration module configuration register (Y = $7 or $F).

Address	15 8	7 0
$YFFE00	TPU MODULE CONFIGURATION REGISTER (TPUMCR)	
$YFFE02	TEST CONFIGURATION REGISTER (TCR)	
$YFFE04	DEVELOPMENT SUPPORT CONTROL REGISTER (DSCR)	
$YFFE06	DEVELOPMENT SUPPORT STATUS REGISTER (DSSR)	
$YFFE08	TPU INTERRUPT CONFIGURATION REGISTER (TICR)	
$YFFE0A	CHANNEL INTERRUPT ENABLE REGISTER (CIER)	
$YFFE0C	CHANNEL FUNCTION SELECTION REGISTER 0 (CFSR0)	
$YFFE0E	CHANNEL FUNCTION SELECTION REGISTER 1 (CFSR1)	
$YFFE10	CHANNEL FUNCTION SELECTION REGISTER 2 (CFSR2)	
$YFFE12	CHANNEL FUNCTION SELECTION REGISTER 3 (CFSR3)	
$YFFE14	HOST SEQUENCE REGISTER 0 (HSQR0)	
$YFFE16	HOST SEQUENCE REGISTER 1 (HSQR1)	
$YFFE18	HOST SERVICE REQUEST REGISTER 0 (HSRR0)	
$YFFE1A	HOST SERVICE REQUEST REGISTER 1 (HSRR1)	
$YFFE1C	CHANNEL PRIORITY REGISTER 0 (CPR0)	
$YFFE1E	CHANNEL PRIORITY REGISTER 1 (CPR1)	
$YFFE20	CHANNEL INTERRUPT STATUS REGISTER (CISR)	
$YFFE22	LINK REGISTER (LR)	
$YFFE24	SERVICE GRANT LATCH REGISTER (SGLR)	
$YFFE26	DECODED CHANNEL NUMBER REGISTER (DCNR)	

Figure 5 TPU Address Map

CIER — Channel Interrupt Enable Register **$YFFE0A**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CH 15	CH 14	CH 13	CH 12	CH 11	CH 10	CH 9	CH 8	CH 7	CH 6	CH 5	CH 4	CH 3	CH 2	CH 1	CH 0

CH	Interrupt Enable
0	Channel interrupts disabled
1	Channel interrupts enabled

CFSR[0:3] — Channel Function Select Registers **$YFFE0C – $YFFE12**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CFS (CH 15, 11, 7, 3)				CFS (CH 14, 10, 6, 2)				CFS (CH 13, 9, 5, 1)				CFS (CH 12, 8, 4, 0)			

CFS[4:0] — Function Number (Assigned during microcode assembly)

HSQR[0:1] — Host Sequence Registers **$YFFE14 – $YFFE16**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CH 15, 7		CH 14, 6		CH 13, 5		CH 12, 4		CH 11, 3		CH 10, 2		CH 9, 1		CH 8, 0	

CH[15:0]	Action Taken
00	Channel A
01	Channel B
10	Channel B
11	Channel C (3 channel mode only)

HSRR[0:1] — Host Service Request Registers **$YFFE18 – $YFFE1A**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CH 15, 7		CH 14, 6		CH 13, 5		CH 12, 4		CH 11, 3		CH 10, 2		CH 9, 1		CH 8, 0	

CH[15:0]	Initialization
00	No Host Service (Reset Condition)
01	Not Used
10	Initialize — 2 channel mode
11	Initialize —3 channel mode

CPR[1:0] — Channel Priority Registers **$YFFE1C – $YFFE1E**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CH 15, 7		CH 14, 6		CH 13, 5		CH 12, 4		CH 11, 3		CH 10, 2		CH 9, 1		CH 8, 0	

CH[15:0]	Channel Priority
00	Disabled
01	Low
10	Middle
11	High

CISR — Channel Interrupt Status Register **$YFFE20**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CH 15	CH 14	CH 13	CH 12	CH 11	CH 10	CH 9	CH 8	CH 7	CH 6	CH 5	CH 4	CH 3	CH 2	CH 1	CH 0

CH	Interrupt Status
x	Not used

6 Function Configuration

In many applications HALLD is used in conjunction with another function, such as COMM, that uses the decoded state number. In these cases, the initialization of the system as a whole must be considered. For an example see **7.6 Using HALLD with the COMM TPU Function**. The following section details the basic configuration of the HALLD function. The CPU configures the HALLD function as follows.

1. Disables the channels by clearing the two channel priority bits on each of the HALLD channels.
2. Selects the HALLD function on two or three channels by writing the HALLD function number to the appropriate function select bits.
3. Initializes DIRECTION in channel A (lowest numbered HALLD channel) parameter RAM.
4. Initializes STATE_NO_ADDR in channel B (two channel mode) or channel C (three channel mode) parameter RAM.
5. Identifies channels by writing the appropriate host sequence bits. The following values are used:
   Channel A = %00
   Channel B = %01 or %10
   Channel C = %11
6. Issues an HSR %10 to both channels for two channel operating mode or issues an HSR %11 to three channels for three channel operating mode.
7. Enables servicing by assigning H, M or L priority to the appropriate channels. All HALLD channels are normally assigned the same priority, to ease system performance calculations, but this is not essential to ensure correct operation.

The TPU then executes the initialization state on each channel. If all HSRs are issued at the same time, and all channels have the same priority, then the channels will be serviced in order, starting with channel A. As each channel is serviced, the HSR bits of that channel are cleared. Only after the last channel has been serviced can STATE_NO be considered valid. Thereafter, the TPU responds to transitions on any HALLD channel, then decodes and updates the state number. The CPU can force an update to STATE_NO at any time by issuing another HSR %10 (two channel mode) or another HSR %11 (three channel mode) to ONE of the channels.

7 Performance and Use of Function

7.1 Performance

Like all TPU functions, the performance limit of the HALLD function in a given application is dependent upon 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 230 kHz in two channel mode and 198 kHz in three channel mode with a CPU bus speed of 16.78 MHz.

When more TPU channels are active, this performance is reduced. However, the scheduler assures that the worst case latencies in any TPU application can be closely estimated. To perform an analysis of any proposed TPU application that appears to approach the performance limits of the TPU, it is recommended that the guidelines given in the TPU reference manual be used along with the figures given in the HALLD state timing table. Additionally, the documentation for the commutation (COMM) TPU function includes a worked example on calculating system performance in a brushless motor application that uses the HALLD, COMM, and MCPWM PWM functions.

Table 3 HALLD State Timing

State Number and Name	Max. CPU Clock Cycles	RAM accesses by TPU
S1 INIT_2CH_HALLD	60	8
S2 INIT_3CH_HALLD	74	10
S3 TRANS_HALLD		
2 Channel Mode	56	8
3 Channel Mode	70	10

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

7.2 Noise Immunity

Features in the hardware of the TPU and the microcode of the HALLD function protect the decoded state number from a permanent incorrect value caused by noise on one of the input channels. However, a transient incorrect STATE_NO is possible.

All TPU input channels incorporate a digital filter which blocks pulses of less than two CPU clocks duration and passes pulses of greater than four CPU clock duration. A pulse with duration of three clocks may or may not pass through the filter. If a noise pulse is greater than four CPU clocks in duration, it causes a service request on the relevant HALLD channel. There are three possible cases to consider.

7.3 Case A

A noise pulse is less than two CPU clocks in duration (See **Figure 6**). The filter blocks the pulse, and the function takes no action.

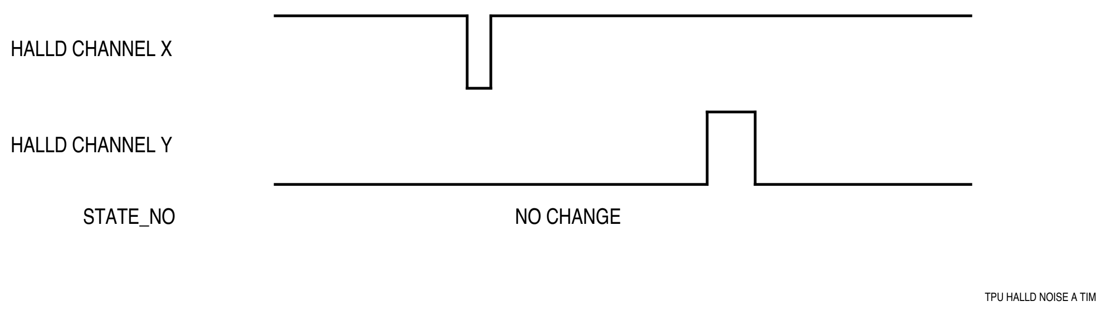


Figure 6 Noise Immunity Case A:
Positive Or Negative Pulses Two CPU Clocks Or Less In Duration.

7.4 Case B

A noise pulse is long enough to pass through the digital filter, but by the time the channel is serviced the pin has returned to its original state (See **Figure 7**). The function decodes the same value for STATE_NO as it did on the last valid transition, but a link request to the destination channel is still generated.

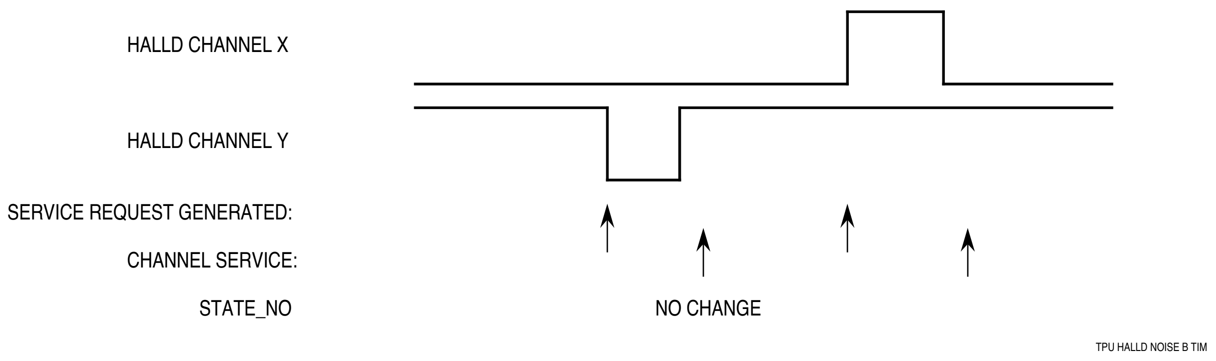


Figure 7 Noise Immunity Case B:
Positive Or Negative Pulses Four CPU Clocks Or Greater In Duration
But Less Than TPU Service Latency At The Time Of The Pulse.

7.5 Case C

A noise pulse is long enough to be present when the channel is serviced (See **Figure 8**). An incorrect STATE_NO is decoded and written to parameter RAM, and the destination channel receives a link request. However, microcode operation guarantees that the channel will also recognize the other edge of the noise pulse, and the second edge causes the channel to be serviced again. This time, the correct STATE_NO is decoded, and another link request to the destination channel is generated. The incorrect STATE_NO is present only for the duration of the noise pulse plus service latency for the second edge.

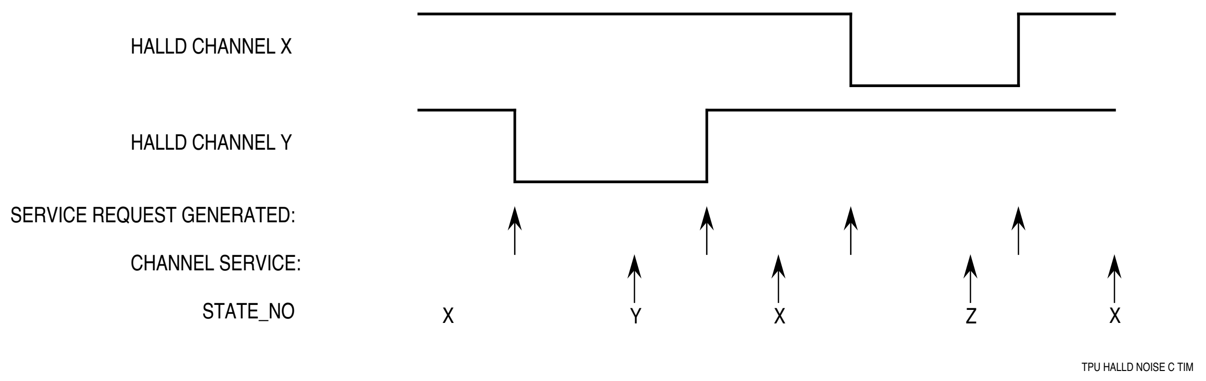


Figure 8 Noise Immunity Case C:
Positive Or Negative Pulses Four CPU Clocks Or Greater In Duration
And Greater Than TPU Service Latency At The Time Of The Pulse

7.6 Using HALLD with the COMM TPU Function

HALLD has been primarily designed for use with the COMM function 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 is generated when each channel is initialized. This means that in a three channel case, COMM would receive three links. In addition, STATE_NO is only valid after all the channels have been initialized, so the first two links are associated with an invalid STATE_NO. To avoid COMM responding to these links, initialize HALLD first, then initialize COMM after all HALLD HSRs have been cleared. In this way, COMM uses a valid STATE_NO from initialization onward.

In some motor applications, certain sensor states are invalid. For example, in a three phase brushless motor, the three Hall effect sensors produce only six valid states, but the other two states 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 programmable state table in the COMM function allows the designer to deal with this eventuality. The entries in the state table that correspond to invalid states should be configured to take the appropriate action (usually to turn off the phase drivers). Refer to*Commutation Output TPU Function (COMM)* (TPUPN09/D) for more detail.

7.7 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 a spare parameter RAM location of one of the HALLD channels. This means that a HALLD channel receives 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 an encoded form, has already been described. HALLD can be used more simply, as two or three normal digital inputs, ignoring the value of STATE_NO. After initialization, the CPU can read the PINSTATE parameter of the function channels at any time, to get the level of the channel pin after the last edge was serviced.

8 Function Examples

The following examples show configuration of the Hall effect decode function for both two and three channel modes. Each example includes a description of the example and diagrams of the initial parameter RAM content and the initial control bit settings.

8.1 Example A

8.1.1 Description

Configure channels 1 and 2 to run HALLD in two channel mode. The STATE_NO output should be written to parameter 2 of channel 10. Initial DIRECTION should be $0000.

8.1.2 Initialization

Disable channels 1 and 2 by clearing priority bits. Select HALLD function by programming the function select register of each channel. Configure parameter RAM of each channel as shown below. Write HSQ = %00 to channel 1 (A) and HSQ = %01 to channel 2 (B). Issue HSR = %10 to both channels to initialize and start the decode. Write the priority bits of both channels to a non-zero value to enable service.

Table 4 Channel 1 (A) Parameter RAM

	15							8								0	
$YFFF10	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	
$YFFF12	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	
$YFFF14	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	
$YFFF16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	DIRECTION
$YFFF18	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	
$YFFF1A	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	PINSTATE

DIRECTION = $0000

Table 5 Channel 2 (B) Parameter RAM

	15							8								0	
$YFFF20	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	
$YFFF22	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	
$YFFF24	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	
$YFFF26	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	
$YFFF28	X	X	X	X	X	X	X	X	1	0	1	0	0	1	0	0	STATE_NO_ADDR
$YFFF2A	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	PINSTATE

STATE_NO_ADDR = $A4

The function now runs, and decodes a new STATE_NO whenever a valid transition occurs on channel 1 or 2. After initialization is complete, the PINSTATE parameters represent the level of the corresponding channel after the last transition on that channel. The CPU can change DIRECTION at any time, but for it to have an immediate effect on STATE_NO, an HSR %10 must be issued to one of the channels. Each time a new STATE_NO is written, a link is generated to channel 10.

8.2 Example B

8.2.1 Description

Configure channels 3, 4 and 5 to run HALLD in three channel mode. The STATE_NO output should be written to parameter 1 of a master COMM channel which has been chosen as channel 13. Initial DIRECTION should be $0001.

8.2.2 Initialization

Disable channels 3, 4 and 5 by clearing priority bits. Select HALLD function by programming the function select register of each channel. Configure parameter RAM of each channel as shown below. Write HSQ = %00 to channel 3 (A), HSQ = %01 to channel 4 (B) and HSQ = %11 to channel 5 (C). Issue HSR = %11 to all HALLD channels to initialize and start the decode. Write the priority bits of the three channels to a non-zero value to enable service.

Table 6 Channel 3 (A) Parameter RAM

	15							8								0	
$YFFF30	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	
$YFFF32	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	
$YFFF34	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	
$YFFF36	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	DIRECTION
$YFFF38	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	
$YFFF3A	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	PINSTATE

DIRECTION = $0001

Table 7 Channel 4 (B) Parameter RAM

	15							8								0	
$YFFF40	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	
$YFFF42	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	
$YFFF44	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	
$YFFF46	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	
$YFFF48	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	
$YFFF4A	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	PINSTATE

Table 8 Channel 5 (C) Parameter RAM

	15							8								0	
$YFFF50	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	
$YFFF52	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	
$YFFF54	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	
$YFFF56	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	
$YFFF58	X	X	X	X	X	X	X	X	1	1	0	1	0	0	1	0	STATE_NO_ADDR
$YFFF5A	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	PINSTATE

STATE_NO_ADDR = $D2

The function now runs, and decodes a new STATE_NO whenever a valid transition occurs on one of the HALLD channels. After initialization is complete, the PINSTATE parameters represent the level of the corresponding channel after the last transition on that channel. The CPU can change DIRECTION at any time, but for it to have an immediate effect on STATE_NO, an HSR %11 must be issued to one of the channels. Each time a new STATE_NO is written, a link is generated to the master COMM channel.

9 Function Algorithm

The following description is provided as a guide only, to aid understanding of the function. The exact sequence of operations in microcode may be different from that shown, in order to optimize speed and code size. TPU microcode source listings for all functions in the TPU function library can be downloaded from the Freescale Freeware bulletin board. Refer to *Using the TPU Function Library and TPU Emulation Mode* (TPUPN00/D) for detailed instructions regarding downloading and compiling microcode.

The HALLD function consists of three states, which operate as described below. For clarity, reference is made to internal channel flag0. This is an internal TPU control bit that is not available to the user.

9.1 STATE1: INIT_2CH_HALLD

This state is entered as a result of a host service request type %10. It initializes the channel into two channel mode and performs a STATE_NO decode.

```
The channel is configured as an input and to detect either transition type
Transition service requests are enabled
Internal channel flag0 is cleared to indicate the two channel mode of operation
The current pin state is read and the latches cleared to enable capture of any future edges
If the pin is low
    $0000 is stored in PINSTATE
Else
    $8000 is stored in PINSTATE
STATE_NO is formed as follows
    STATE_NO = DIRECTION + {2 ∗ channel A PINSTATE[MSB]} + {4 ∗ channel B PINSTATE[MSB]}
STATE_NO is stored in the PRAM location addressed by STATE_NO_ADDR
A link request is generated to the destination channel of the state number
The state ends
```

9.2 STATE2: INIT_3CH_HALLD

This state is entered as a result of a host service request type %11. It initializes the channel into three channel mode and performs a STATE_NO decode.

```
The channel is configured as an input and to detect either transition type
Transition service requests are enabled
Internal channel flag0 is set to indicate the three channel mode of operation
The current pin state is read and the latches cleared to enable capture of any future edges
If the pin is low
    $0000 is stored in PINSTATE
Else
    $8000 is stored in PINSTATE
STATE_NO is formed as follows
    STATE_NO = DIRECTION + {2 ∗ channel A PINSTATE[MSB]} + {4 ∗ channel B PINSTATE[MSB]}
            + {8 ∗ channel C PINSTATE[MSB]}
STATE_NO is stored in the PRAM location addressed by STATE_NO_ADDR
A link request is generated to the destination channel of the state number
The state ends
```

9.3 STATE3: TRANS_HALLD

This state is entered as a result of a transition on one of the HALLD channels. It performs a STATE_NO decode.

```
The current pin state is read and the latches cleared to enable capture of any future edges
If the pin is low
    $0000 is stored in PINSTATE
```

```
Else
    $8000 is stored in PINSTATE
If internal channel flag0 = 0,
    STATE_NO is formed as follows
        STATE_NO = DIRECTION + {2 ∗ channel A PINSTATE[MSB]}
            + {4 ∗ channel B PINSTATE[MSB]}
Else
    STATE_NO is formed as follows:
        STATE_NO = DIRECTION + {2 ∗ channel A PINSTATE[MSB]}
            + {4 ∗ channel B PINSTATE[MSB]} + {8 ∗ channel C PINSTATE[MSB]
STATE_NO is stored in the PRAM location addressed by STATE_NO_ADDR
A link request is generated to the destination channel of the state number
The state ends
```

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 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