MOTOROLA
SEMICONDUCTOR
PROGRAMMING NOTE

Order this document
by TPUPN02/D

Fast Quadrature Decode TPU Function (FQD)

by Jeff Wright

1 Functional Overview

The fast quadrature decode function is a TPU input function that uses two channels to decode a pair of out-of-phase signals in order to increment or decrement a (position) counter. It is particularly useful for decoding position and direction information from a slotted encoder in motion control systems, thus replacing expensive external solutions. **Figure 1** shows a typical application.

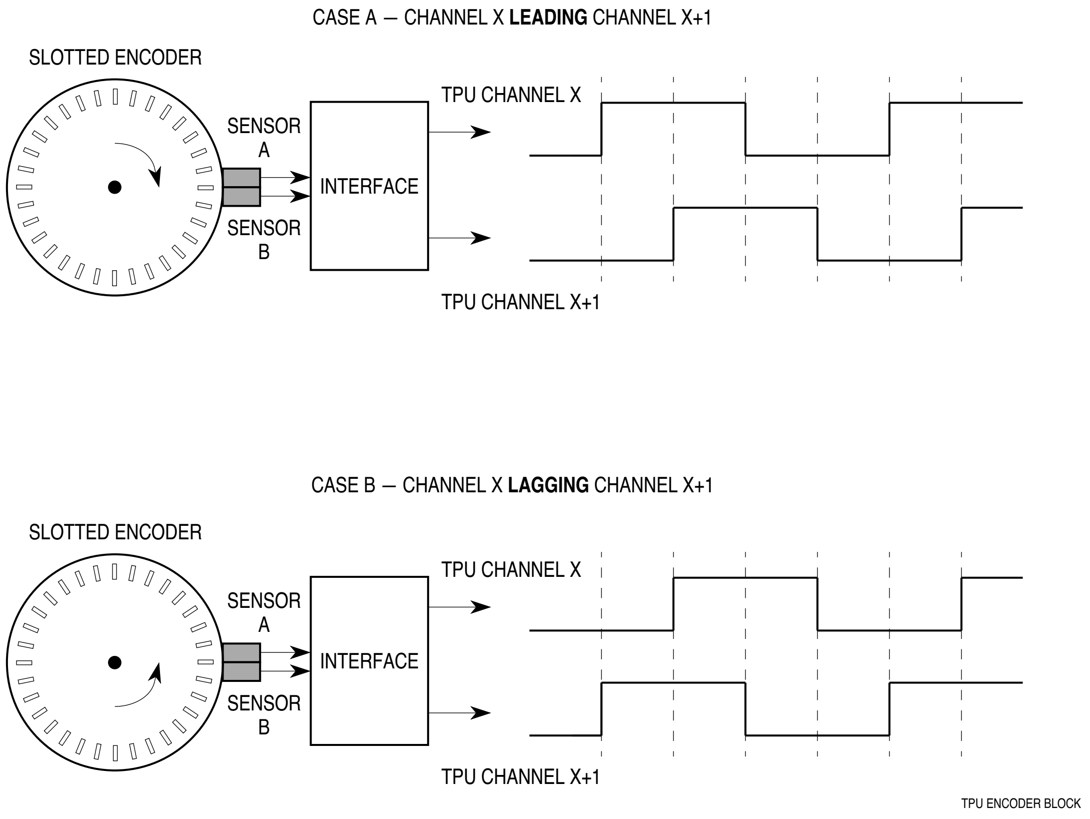


Figure 1 Typical FQD Application

2 Detailed Description

The FQD function uses a pair of adjacent TPU channels to decode quadrature signals into a 16-bit counter in parameter RAM (PRAM). The counter is updated when a valid transition is detected on either one of the two inputs — full ‘4x’ resolution is derived from the encoder signals. The counter is incremented or decremented depending on the lead/lag relationship of the two signals at the time of servicing the transition. The user can read or write the counter at any time. The counter is free running, overflowing to $0000 or underflowing to $FFFF depending on direction.

In systems where the counter may overflow or underflow, the user must ensure that the CPU reads the counter periodically. Maximum period is $8000 counts at maximum signal frequency. Two's complement arithmetic can then be used by the CPU to maintain position and direction information.

When initialized, the FQD function is configured so that the first edge on either channel results in a counter update.

Since the two FQD channels, which must always be adjacent, operate differently, this note uses the convention of referring to the channel with the lower channel number as the primary channel. The other channel is referred to as the secondary channel.

The FQD function differs from the QDEC function in having both normal and fast modes of operation. In operation, the CPU dynamically switches the FQD function between modes depending on the current encoder speed.

2.1 Normal Mode

In normal mode, both quadrature signals are decoded by the TPU and the counter is updated by one for each valid transition on either channel (see **Figure 2**). The counter is incremented or decremented depending on the lead/lag relationship of the two signals at the time of transition service. See **9 Fast Quadrature Decode Algorithm** for a definition of the lead/lag test.

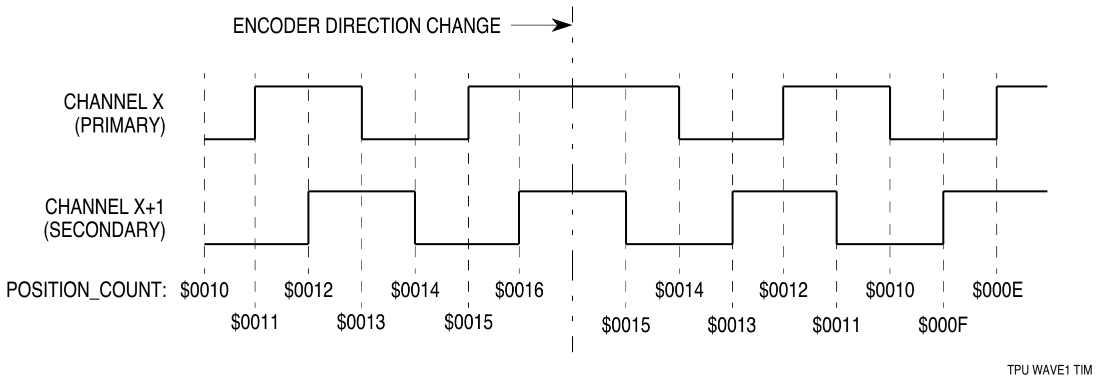


Figure 2 Normal Mode Operation

2.2 Fast Mode

In fast mode, only the primary channel is serviced. The counter is updated by four for each rising transition (see **Figure 3**). All falling transitions are ignored. In fast mode, the TPU can reliably decode at more than quadruple the maximum count rate of normal mode. No direction decoding is done in fast mode — the counter is updated in the same direction as when the last transition was serviced in normal mode.

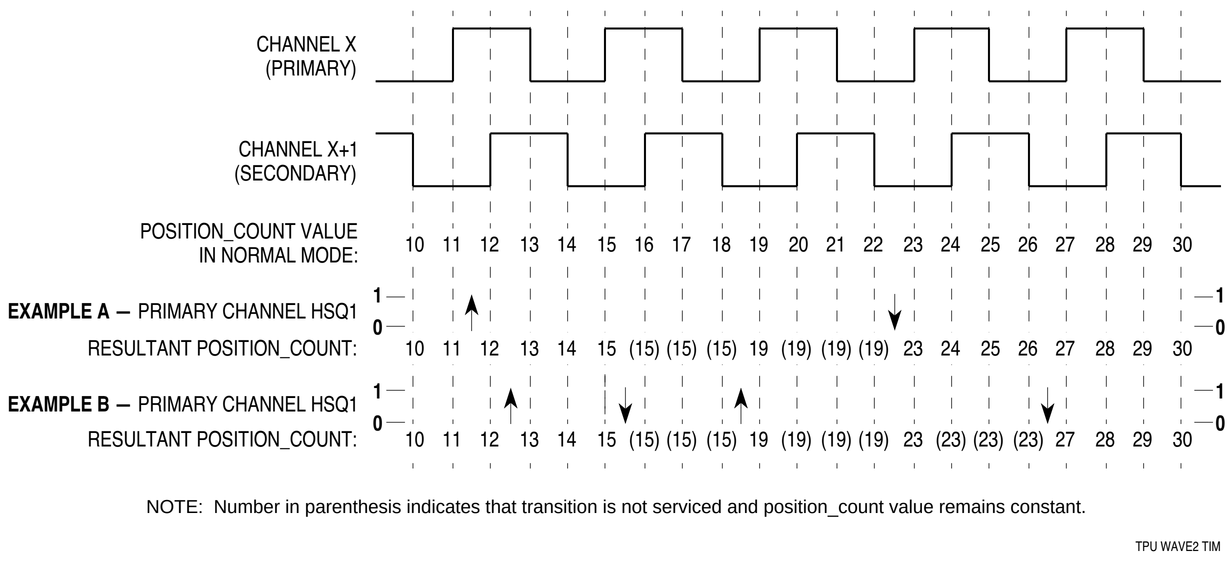


Figure 3 Fast Mode Operation

FQD mode of operation can be changed at any time by the CPU via a host sequence bit on the primary channel. Any requested change in operating mode takes effect when the next rising transition on the primary channel is serviced.

No counts are lost when switching in and out of fast mode, although there is a four LSB uncertainty in the counter while in fast mode, due to the ‘by 4’ update.

If application performance requires that fast mode be used, the CPU should start FQD in normal mode and switch to fast mode when the derived speed (from periodic reads of the position counter) is above a certain threshold. The function should run in fast mode until the speed falls below threshold, when the CPU should switch back to normal mode. The speed threshold at which to switch modes is determined by overall TPU system activity and must be evaluated for each application.

2.3 Time Stamp

In normal mode, the FQD function provides a time stamp referenced to TCR1 for every valid signal edge. The host CPU can also request a current TCR1 value. These two features allow position and speed interpolation by the host CPU between quadrature edges at very slow count rates.

2.4 Discrete Input/Transition Counter

A single channel programmed to run FQD can be used as a digital input pin with a transition counter.

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. FQD function code size is:

38 μ instructions + 8 entries = **46 long words**

4 Function Parameters

This section provides detailed descriptions of function parameters stored in channel parameter RAM. **Figure 4** shows TPU parameter RAM address mapping. 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	**Base**	**Parameter Address**							
Number	**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 4 TPU Channel Parameter RAM CPU Address Map

	15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
$YFFFW0	░ EDGE_TIME
$YFFFW2	▒ POSITION_COUNT
$YFFFW4	□ TCR1_VALUE
$YFFFW6	□ CHAN_PINSTATE
$YFFFW8	□ CORR_PINSTATE_ADDR
$YFFFWA	□ EDGE_TIME_LSB_ADDR
$YFFFWC	■
$YFFFWE	■

W = Channel number

Primary Channel RAM Assignment

	15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
$YFFF(W+1)0	■
$YFFF(W+1)2	■
$YFFF(W+1)4	░ TCR1_VALUE
$YFFF(W+1)6	░ CHAN_PINSTATE
$YFFF(W+1)8	□ CORR_PINSTATE_ADDR
$YFFF(W+1)A	□ EDGE_TIME_LSB_ADDR
$YFFF(W+1)C	■
$YFFF(W+1)E	■

W = Primary Channel number

Secondary Channel RAM Assignment

Parameter Write Access

□	Written by CPU
░	Written by TPU
▒	Written by CPU and TPU
■	Unused parameters

Figure 5 FQD Function Parameter RAM Assignment

4.1 TCR1_VALUE

This 16 bit parameter is updated by the TPU to contain the latest value of the TCR1 internal counter. TCR1_VALUE is updated on two occasions:

During initialization of the function.

During the service of a TCR1 read host service request issued by the host CPU.

This parameter can be used along with EDGE_TIME to perform position and speed interpolation at slow count rates. TCR1_VALUE can reside in the parameter RAM of either or both FQD channels, but the parameter is only updated for the channel that receives the host service request from the CPU. Since the TPU must respond to the HSR before copying the TCR1 value to PRAM, the value obtained does not exactly correspond to the TCR1 value at the time the HSR is issued. The difference depends on the latency of the TPU and the prescaler value of TCR1. See **7 Performance and Use of Function** for details.

4.2 CHAN_PINSTATE

These 16-bit parameters (one for each channel) are maintained by the TPU. Each parameter contains a value that represents the logic level of the channel pin when the last valid transition was serviced. The value $8000 is used to represent a pin high level, and $0000 to represent a pin low level. When an edge is serviced, the new pin state is compared with the last pin state stored in CHAN_PINSTATE — if the states are the same, then a valid transition has not occurred (noise) and the counter is not updated.

The CHAN_PINSTATE parameters are also used to determine the phase (lead/lag) relationship between the two FQD channels so that POSITION_COUNT is updated in the correct direction. To perform this lead/lag test, the channel compares its new pin state with the CHAN_PINSTATE parameter of the other FQD channel (obtained via CORR_PINSTATE_ADDR) and from the relationship takes the appropriate action. See **9 Fast Quadrature Decode Algorithm** for an explanation of the lead/lag tests.

The CPU must not write CHAN_PINSTATE parameters while FQD is running, or an erroneous update of POSITION_COUNT can occur.

4.3 CORR_PINSTATE_ADDR

These parameters (one for each channel) are initialized by the CPU to contain the address in PRAM of the CHAN_PINSTATE parameter of the corresponding FQD channel. They are used to obtain the CHAN_PINSTATE parameter of the corresponding channel for the lead/lag test. For example, if channels 0 and 1 are being used for FQD, the CORR_PINSTATE_ADDR of channel 0 should be $16 and CORR_PINSTATE_ADDR of channel 1 should be $06. These parameters are written once prior to initialization and must not be changed while FQD is running.

4.4 EDGE_TIME_LSB_ADDR

These parameters (one for each channel) are initialized by the CPU to contain the address in PRAM of the LSB (odd address) of the EDGE_TIME parameter. The EDGE_TIME_LSB_ADDR parameters of both FQD channels must point to the same PRAM location for the FQD function to operate correctly. This parameter is used to access both EDGE_TIME and POSITION_COUNT parameters. For example, if channels 0 and 1 are being used for FQD and EDGE_TIME and POSITION_COUNT are chosen to reside in channel 1, then the EDGE_TIME_LSB_ADDR of both channels 0 and 1 must be programmed to $11. These parameters are written once prior to initialization and must not be changed while FQD is running.

4.5 EDGE_TIME

This 16-bit parameter, which resides in the parameter RAM of only one FQD channel, is updated by the TPU when a valid signal transition is serviced on either channel in normal mode only. It contains the TCR1 value that was captured in hardware at the time of the signal transition — it thus provides a time stamp for the host CPU.

EDGE_TIME can reside in the PRAM of either FQD channel, but must be in the same PRAM as POSITION_COUNT, because both parameters are referenced by the EDGE_TIME_LSB_ADDR address pointer. See **7 Performance and Use of Function** for more details.

4.6 POSITION_COUNT

This 16-bit counter is the primary output of the FQD function. POSITION_COUNT resides in the parameter RAM of only one FQD channel. POSITION_COUNT can be read or written at any time by the CPU. Normally, POSITION_COUNT is initialized by the CPU, then left to run as a free running counter.

POSITION_COUNT can reside in the PRAM of either FQD channel, but must be in the same PRAM as EDGE_TIME, because both parameters are referenced by the EDGE_TIME_LSB_ADDR address pointer.

4.7 HSQ0

Host sequence bit 0 is written by the CPU. HSQ0 is used by the TPU to determine whether the channel being serviced is the primary or secondary channel. HSQ0 of the primary channel must be cleared and HSQ0 of the secondary channel must be set. The primary channel is always the channel with the lower channel number of the pair — if channels 3 and 4 are to run FQD, then channel 3 HSQ0 must equal 0 and channel 4 HSQ0 must equal one.

4.8 HSQ1

Host sequence bit 1 is written by the CPU. HSQ1 of the primary channel is used to select normal or fast mode of operation. HSQ1 of the secondary channel is not used. If primary channel HSQ1 = 0, then normal mode is selected. If primary channel HSQ1 = 1, then fast mode is selected. Although the CPU can write HSQ1 at any time to change modes, the function should always be started in normal mode.

5 Host Interface to Function

This section provides information concerning the TPU host interface to the FQD function. **Figure 6** is a TPU address map. Detailed TPU register diagrams follow the figure. In **Figure 6** and in the register 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 6 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
X	Not used by this function

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
XXXX	FQD 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]	Operating Mode
00	Primary Channel – Normal Mode
01	Secondary Channel – Normal Mode
10	Primary Channel – Fast Mode
11	Secondary Channel – Fast Mode

HSRR[1:0] — 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]	Action
00	No Host Service (Reset Condition)
01	Not Used
10	Read TCR1
11	Initialize

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 by this function

6 Function Configuration

The CPU configures the FQD function as follows.

1. Disables the channels by clearing the two channel priority bits on each of the FQD channels (not necessary from reset).
2. Selects the FQD function on both channels by writing the FQD function number to their function select bits.
3. Initializes CORR_PINSTATE_ADDR and EDGE_TIME_LSB_ADDR in parameter RAM of both channels.
4. Initializes POSITION_COUNT to the desired start value.
5. Selects one channel as the primary channel and the other as the secondary channel via HSQ0.
6. Selects normal mode of operation by ensuring that HSQ1 of the primary channel is cleared.
7. Issues an HSR type%11 to each channel to initialize the function.
8. Enables servicing by assigning H, M, or L priority to the channel priority bits. Both FQD channels must be assigned the same priority to ensure correct operation.

The TPU then executes the initialization state and starts decoding the two input signals.

NOTE

The CHAN_PINSTATE parameters must not be read by the CPU until after the TPU has negated the HSR bits during initialization.

Fast/Normal mode switching is controlled by the CPU via host sequence bit 1 of the primary channel.

7 Performance and Use of Function

7.1 Performance

Like all TPU functions, the performance limit of the FQD 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 a pair of FQD channels are being used in normal mode and no other TPU channels are active, the minimum time between count edges on the two channels is 50 CPU clock cycles. This is equivalent to a count rate of approximately 330 kcounts per second with a system clock speed of 16.78 MHz, or a count rate of approximately 420 kcounts per second with a system clock speed of 20.97 MHz. In fast mode, the minimum time between rising edges on the primary channel is 30 CPU clock cycles. Since the counter is updated by four on each primary rising edge, this is equivalent to a count rate of approximately 2.2 Mcounts per second with a system clock speed of 16.78 MHz, or a count rate of approximately 2.8 Mcounts per second with a system clock speed of 20.97 MHz.

Table 1 Fast Quadrature Decode Function — State Timing

State Number & Name	Max CPU Clock Cycles	RAM Accesses by TPU
S1 INIT_FQD	12	3
S2 READ_TCR1_FQD	2	1
S3 EDGE_NORM_FQD		
Remain in Normal Mode	36	8
Switch to Fast Mode	40	8
S4 EDGE_FAST_FQD		
Remain in Fast Mode	16	4
Switch to Normal Mode	26	5

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

When more TPU channels are active, performance is lessened — if two sets of encoder signals are decoded using four channels, then the maximum count rate in each mode (with a 16.78 kHz bus) is limited to approximately 165 kcounts and 1.1 Mcounts respectively. Use of other functions, such as PWM, also lessens performance.

Since the scheduler assures that worst-case latencies in any TPU application can be closely estimated, it is recommended that the guidelines in the *TPU Reference Manual* (TPURM/AD) be used with the figures given in the fast quadrature decode state timing table to perform an analysis of any proposed application that appears to approach the performance limits of the TPU. If the FQD function fails to meet the system performance requirements, then the down/up counter (DUC) TPU function should be evaluated as an alternative.

7.2 Accuracy

Since the TPU takes time to respond to an input transition, there is always a 1 LSB uncertainty in a CPU read of POSITION_COUNT in normal mode while the input signals are active. In fast mode, the uncertainty increases to 4 LSB due to the 'by 4' update of POSITION_COUNT.

These uncertainties only apply while an external system such as a motor is active. After the system has been brought to a stop with FQD in normal mode, and the last transition has been serviced, POSITION_COUNT is accurate.

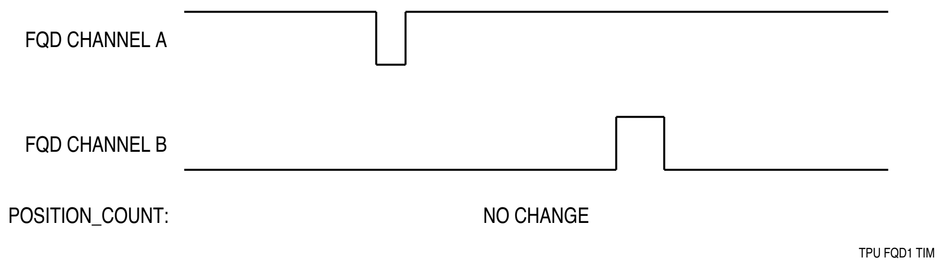
7.3 Noise Immunity

To a large extent, TPU hardware and the FQD function microcode protect the counter from erroneous updates due to noise. All TPU input channels incorporate a digital filter which rejects pulses of less than 2 CPU clocks and guarantees to pass pulses of greater than 4 CPU clocks. In addition, when servicing a transition in normal mode, the FQD function always checks the new pin state against the pin state from the last service, and if they are equal then no action is taken. This protects against a noise pulse that is long enough to get through the digital filter, but not long enough to last from the actual transition time to the time that the TPU services the channel. In fast mode, where only rising edges are serviced, no microcode noise immunity is provided.

Despite these precautions, there may be situations where noise on both channels simultaneously causes erroneous updates of the counter. Under these conditions, it is recommended that additional external protection, such as Schmitt trigger buffers or an additional filter stage, be added.

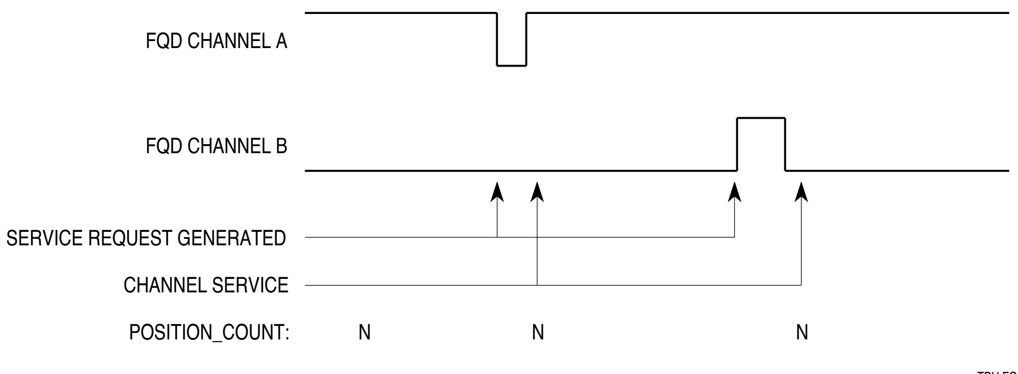
The following examples are intended to illustrate the extent of the noise immunity inherent in the TPU itself and the FQD function.

CASE A: Short positive or negative pulses 2 CPU clocks or less in duration.



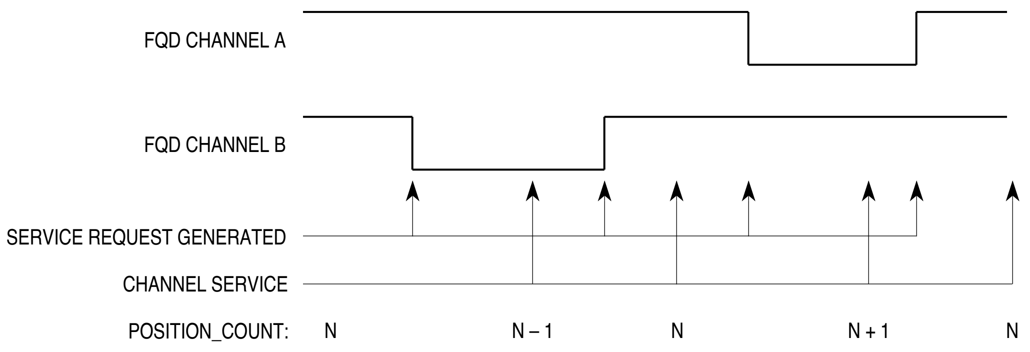
Result: Rejected by hardware filter on TPU input pins —no service requests.

CASE B: Positive or negative pulses 4 CPU clocks or greater in duration, but less than TPU service latency at the time of the pulse.



Result: One service request per pulse — rejected in software by pin state history test.

CASE C: Positive or negative pulses 4 CPU clocks or greater in duration and greater than TPU service latency at the time of the pulse.



Result: Two service requests per pulse. Both edges are serviced and counted resulting in a net error of zero on POSITION_COUNT.

Note that pulses of three CPU clocks in length may pass through the input filter — they can either be case A or case B. Noise rejection cannot be guaranteed when case C noise exists simultaneously on both channels. Only case A noise rejection is provided when running in fast mode.

7.4 Using FQD with Three-Signal Encoders

Many shaft encoders supply two quadrature signals plus an index signal that generates a pulse once per revolution. This pulse usually has a fixed relationship to other system parameters and is used for alignment during startup.

Three-signal encoders can be decoded when FQD is used in conjunction with the TPU function called new input transition counter (NITC). FQD decodes the quadrature signals and the index pulse is fed to the NITC channel. NITC allows any location in parameter RAM to be captured on a specified edge and the value presented to the CPU. In this case, NITC would be configured to capture the POSITION_COUNT parameter of FQD. The NITC channel should be run on a lower channel number than the FQD primary channel, and assigned the same priority as the FQD channel.

7.5 Using the Time Stamp Feature

The time stamp feature has been provided in normal mode to allow the CPU to perform speed and position interpolation at very slow encoder speeds. At low speeds, the number of transitions counted between CPU reads of POSITION_COUNT is too small to provide reliable information. Due to the following restrictions, the time stamp feature should be used with care.

7.6 EDGE_TIME and POSITION_COUNT Coherency

The TPU cannot coherently update both the EDGE_TIME and POSITION_COUNT parameters during the service of a transition. A CPU read of these two parameters may return values that do not correlate, such as a new EDGE_TIME with an 'old' POSITION_COUNT value (EDGE_TIME is updated first).

This problem can be handled by performing multiple CPU reads of the two parameters with a delay between the reads. The delay must be greater than or equal to the worst-case time between the TPU writing EDGE_TIME and POSITION_COUNT of 14 CPU clocks. For example, the following CPU action could be used.

```
GET_PARAM:  Read EDGE_TIME & POSITION_COUNT
LOOP:         Store EDGE_TIME in TEMP1 & POSITION_COUNT in TEMP2
              Delay 14 CPU clocks
              Read EDGE_TIME & POSITION_COUNT
              If EDGE_TIME ≠ TEMP1 or POSITION_COUNT ≠ TEMP2 then
                  goto LOOP
              Endif
VALID:        TEMP1 and TEMP2 are coherent and valid.
```

7.7 TCR1 Timebase Read

To actually perform interpolation, the CPU must obtain valid EDGE_TIME and POSITION_COUNT parameters as described above, then read the TCR1 timebase at fixed intervals to calculate a new position.

Since the CPU must issue an HSR to obtain the latest TCR1_VALUE, and since that HSR is subject to normal TPU scheduling, there is an uncertainty in the returned TCR1_VALUE that is dependent upon both TPU latency at the time of issuing the HSR and upon the selected prescaler value for TCR1.

In the best case (TPU idle at time of HSR issue) there is a delay equivalent to 16 CPU clocks between the time the CPU writes the HSR bits and the time the TPU writes TCR1_VALUE and clears the HSR bits.

7.8 Using FQD as a Discrete Input/Transition Counter

A single TPU channel programmed to run FQD can be used as a discrete input pin and transition counter. To be used in this way, the FQD function must be in normal mode, with the channel programmed as a primary channel. The EDGE_TIME_LSB_ADDR parameter must point to the LSB of the channel's own parameter 0 and the CORR_PINSTATE_ADDR parameter must point to the channel's own CHAN_PINSTATE parameter. An HSR %11 should be issued to initialize the function.

When the FQD function is configured as described, CHAN_PINSTATE is updated as each transition is serviced, and contains a value representing the latest pin level ($8000 = high, $0000 = low). POSITION_COUNT holds the number of transitions on the pin (positive and negative).

An immediate update of CHAN_PINSTATE can be invoked at any time by issuing an HSR %11. The CPU should not interpret the value of CHAN_PINSTATE until the TPU has completed the host service request and the HSR bits have been negated.

8 Fast Quadrature Decode Examples

The following examples show configuration of the fast quadrature decode function for both quadrature decode and for operation as an input pin with transition counter. Each example includes a description of the example, a diagram of the initial parameter RAM content, and the initial control bit settings.

8.1 Example A

Configure channels 1 and 2 to run FQD. The initial position should be $1000.

Disable channels 1 and 2 by clearing priority bits (CPR1[3:2] and CPR1[5:4]). Select FQD function by programming the function select register of each channel. Configure parameter RAM of each channel as shown below. Write HSQR1[3:2] = %00 (channel 1 primary) and HSQR1[5:4] = %01 (channel 2 secondary). Write HSRR1[3:2] and HSRR1[5:4] = %11 to initialize both channels and start quadrature decode. Write the priority bits of both channels to the same non-zero value.

Table 2 Channel 1 (Primary) Parameter RAM

	15							8								0	
$YFFF10	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	EDGE_TIME
$YFFF12	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	POSITION_COUNT
$YFFF14	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	TCR1_VALUE
$YFFF16	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	CHAN_PINSTATE
$YFFF18	0	0	0	0	0	0	0	0	0	0	1	0	0	1	1	0	$26
$YFFF1A	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	$11

POSITION_COUNT = $1000

Table 3 Channel 2 (Secondary) 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	TCR1_VALUE
$YFFF26	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	CHAN_PINSTATE
$YFFF28	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1	0	$16
$YFFF2A	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	$11

The function now runs, decodes transitions on channel 1 or 2, and increments or decrements POSITION_COUNT accordingly. The CPU can read or write POSITION_COUNT at any time. If fast mode operation is required, it is controlled via HSQR1[1:0] (channel 1 is the primary channel).

8.2 Example B

8.2.1 Description

Configure channel 12 to act as an input pin with a transition counter. Initialize the counter to zero.

8.2.2 Initialization

Disable channel 12 by clearing priority bits (CPR0[9:8]). Select FQD function by programming the function select register of channel 12. Configure channel 12 parameter RAM as shown below. Write HSQR0[9:8] = %00 (primary channel). Write HSRR0[9:8] = %11 to channel 12 to initialize, read pin level and start counting transitions. Write channel 12 priority bits to a non-zero value.

Table 4 Channel 12 Parameter RAM

	15							8								0	
$YFFFC0	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	EDGE_TIME
$YFFFC2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	TRANS_COUNT
$YFFFC4	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	TCR1_VALUE
$YFFFC6	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	CHAN_PINSTATE
$YFFFC8	0	0	0	0	0	0	0	0	1	1	0	0	0	1	1	0	$C6
$YFFFCA	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	1	$C1

TRANS_COUNT (POSITION_COUNT) = $0000

The function now runs, detects transitions on channel 12, and increments the transition counter (POSITION_COUNT) accordingly. On completion of the initialization HSR and any subsequent edge service, the parameter CHAN_PINSTATE contains the latest level of the channel pin ($8000 for high, $0000 for low). The CPU can read or write the transition counter at any time.

Note that the TCR1 read HSR can still be used when FQD is operating in this mode.

9 Fast Quadrature Decode 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 to optimize speed and code size. TPU microcode source listings for all functions in the TPU function library can be downloaded from the Motorola Freeware bulletin board. Refer to *Using the TPU Function Library and TPU Emulation Mode* (TPUPN00/D) for detailed instructions on downloading and compiling microcode.

The fast quadrature decode function consists of four states, which operate as described below. For clarity, reference is made to internal channel flags 0 and 1 in the following description. These are internal TPU control bits that are not available to the user.

9.1 STATE1 — INIT_FQD

This state is entered as a result of a host service request type %11.

```
The channel is configured as an input with TCR1 as a timebase.
The pin is configured to detect any transition.
Transition service requests are enabled.
The current pin state is read.
    If the pin is low
        $0000 is stored in CHAN_PINSTATE
    Else
        $8000 is stored in CHAN_PINSTATE
    Endif.
Flag 0 is negated to force normal mode of operation.
The current value of TCR1 is read and stored in TCR1_VALUE.
The state ends.
```

9.2 STATE2 — READ_TCR1_FQD

This state is entered as a result of a host service request type %10.

```
The current value of TCR1 is read and stored in TCR1_VALUE.
The state ends.
```

9.3 STATE3 — EDGE_NORM_FQD

This state is entered as a result of a transition on an FQD channel pin while internal channel flag 0 is negated.

The channel pin state is read.

The transition latch cleared to enable detection of further edges.

```
If the new pin state = CHAN_PINSTATE
    The state ends (noise).
Endif.
If the pin is low
    $0000 is stored in CHAN_PINSTATE
Else
    $8000 is stored in CHAN_PINSTATE
Endif.
```

The TCR1 value captured at the time of the edge is stored in EDGE_TIME.

Using this new pin state along with the pin state of the other FQD channel and host sequence bit 0 (primary or secondary channel), POSITION_COUNT is incremented or decremented by 1 according to the lead/lag tests explained below.

```
If POSITION_COUNT is incremented,
        assert internal channel flag 1
    Else
        negate internal channel flag 1
    Endif.
```

When service resulted from a rising edge on the primary channel and host sequence bit 1 is asserted then fast mode is entered as follows.

```
    The secondary channel is disabled.
    The primary channel is configured to detect rising edges only.
    Internal channel flag 0 is asserted.
The state ends.
```

9.4 STATE4 — EDGE_FAST_FQD

This state is entered as a result of a rising transition on an FQD primary channel while internal channel flag 0 is asserted.

```
The transition latch cleared to enable detection of further edges.
    If internal channel flag 1 is asserted
        POSITION_COUNT is incremented by 4
    Else
        POSITION_COUNT is decremented by 4
    Endif.
```

When host sequence bit 1 is negated then normal mode is entered as follows.

```
The secondary channel is re-enabled and its CHAN_PINSTATE parameter corrected.
The primary channel is set to detect any edge.
Internal channel flag 0 is negated.
```

The state ends.

9.5 Explanation of Lead/Lag Test

The lead/lag test is performed to determine the phase relationship of the two FQD signals and hence whether to increment or decrement the parameter POSITION_COUNT. The CHAN_PINSTATE parameters of the two channels are added together and the resulting N bit is used along with the edge type and channel type (primary or secondary — host sequence bit 0) to result in the following operation:

Table 5 Lead/Lag Test Results

Serviced Transition	Test Description
Primary Rising	If last secondary transition was falling, then primary channel is leading secondary channel and POSITION_COUNT is incremented. If last secondary transition was rising, then primary channel is lagging secondary channel and POSITION_COUNT is decremented.
Primary Falling	If last secondary transition was rising, then primary channel is leading secondary channel and POSITION_COUNT is incremented. If last secondary transition was falling, then primary channel is lagging secondary channel and POSITION_COUNT is decremented.
Secondary Rising	If last primary transition was rising, then primary channel is leading secondary channel and POSITION_COUNT is incremented. If last primary transition was falling, then primary channel is lagging secondary channel and POSITION_COUNT is decremented.
Secondary Falling	If last primary transition was falling, then primary channel is leading secondary channel and POSITION_COUNT is incremented. If last primary transition was rising, then primary channel is lagging secondary channel and POSITION_COUNT is decremented.

NOTES

How to reach us:
USA/EUROPE/Locations Not Listed: Motorola Literature Distribution;
P.O. Box 5405, Denver Colorado 80217. 1-800-441-2447, (303) 675-2140

JAPAN: Nippon Motorola Ltd.; Tatsumi-SPD-JLDC,
6F Seibu-Butsuryu-Center, 3-14-2 Tatsumi Koto-Ku, Tokyo 135, Japan. 81-3-3521-8315
ASIA PACIFIC: Motorola Semiconductors H.K. Ltd.; 8B Tai Ping Industrial Park,
51 Ting Kok Road, Tai Po, N.T., Hong Kong. 852-26629298

M

MOTOROLA