MOTOROLA
SEMICONDUCTOR
PROGRAMMING NOTE

Order this document
by TPUPN03/D

# Frequency Measurement TPU Function (FQM)

**by Kevin Anderson**

## 1 Functional Overview

The FQM function counts the number of pulses that are presented to a channel pin within a user specified time window. The pulse count is available to the user as a 16-bit number. Either rising or falling edges can be used as the beginning of a pulse. There are two basic modes of operation. In single shot mode, pulses are accumulated for a single window time. In continuous mode, pulses are automatically accumulated in repetitive windows.

## 2 Detailed Description

The time window is specified as a number of TCR clock ticks. The maximum value is $8000. Either TCR may be used as the time base. The function is initialized by specifying the time window and issuing a host service request. The channel is then initialized by the microcode and waits for the first selected edge. The time window begins with the first selected edge following initialization. Thus the function can be pre-initialized to wait for pulses to appear on the pin.

The term “selected edge” is used to describe an edge, either rising or falling, that has been specified as the beginning of a pulse. Since the first selected edge to appear after initialization starts the first window, the second selected edge is counted as the end of the first pulse. Thus pulses are counted as they complete. The number of completed pulses is accumulated until the time window expires. At that time the accumulated number is written to parameter RAM and an interrupt service request is generated. Partial pulses are not counted. If pulse completion coincides with the end of window time, it is counted as a complete pulse.

In single-shot mode, the function idles when a pulse is complete. New single-shot accumulation times can be initiated by issuing a host service request. Window accumulation time can also be changed prior to issuing the request.

In continuous mode, a new time accumulation window begins coincident with the end of the previous window, and pulse counting restarts at zero. If a pulse begins in a window and continues into the next window, it is counted as the first pulse in the new window when the first selected edge in the new window is detected. Thus, no pulse is lost, even when it straddles accumulation windows.

At the end of each time accumulation period, the newly-accumulated value is written to parameter RAM and an interrupt service request is generated. The accumulated value is available to the CPU until the end of the current window. When the current window ends, the accumulated value is overwritten. CPU interrupt latency must be less than window time in order to guarantee that no accumulation values are missed.

In continuous mode, the user can modify the length of the time accumulation window at any time by changing the window value in parameter RAM. The new window length takes effect at the end of the current window time. It is not necessary to disable the function or issue a host service request in order to make the change.

The FQM function has been optimized for fast execution and small size. The function does not attempt to distinguish false edges due to noisy inputs. Any pulse long enough to pass through the pin synchronizer and digital filter is counted. The pin synchronizer and digital filter reject all pulses narrower than two CPU system clocks and pass all pulses wider than four CPU system clocks. A companion function to FQM, called pulse accumulate in a programmable window (PAPW) offers noise rejection of longer pulses at the expense of slightly longer execution time and increased code size. PAPW can be used in the same manner as FQM.

## 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. FQM function code size is:

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

## 4 FQM Function Parameters

This section provides detailed descriptions of FQM function parameters stored in channel parameter RAM. **Figure 1** shows TPU parameter RAM address mapping. **Figure 2** shows the parameter RAM assignment used by the FQM 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 1 TPU Channel Parameter RAM CPU Address Map**

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $YFFFW0 | NOT USED | | | | | | | | | | | | | | | |
| $YFFFW2 | NOT USED | | | | | | | | | | | | | | | |
| $YFFFW4 | CHANNEL_CONTROL | | | | | | | | | | | | | | | |
| $YFFFW6 | WINDOW_SIZE | | | | | | | | | | | | | | | |
| $YFFFW8 | PULSE_COUNT | | | | | | | | | | | | | | | |
| $YFFFWA | IN_WINDOW_ACCUMLATION | | | | | | | | | | | | | | | |

Y = Channel number

Parameter Write Access

Written by CPU

Written by TPU

**CHANNEL_CONTROL Bit Encoding**

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $YFFFW4 | NOT USED | | | | | | | TBS | | | | PAC | | | PSC | |

**CHANNEL_CONTROL Options**

| ACTION | TBS | | | | PAC | | | PSC | |
|---|---|---|---|---|---|---|---|---|---|
| — | | | | | | | | 1 | 1 |
| Detect Rising Edge | | | | | 0 | 0 | 1 | | |
| Detect Falling Edge | | | | | 0 | 1 | 0 | | |
| Capture TCR1, Match TCR1 | 0 | 0 | 0 | 0 | | | | | |
| Capture TCR2, Match TCR2 | 0 | 0 | 1 | 1 | | | | | |

**Figure 2 FQM Function Parameter RAM Assignment**

### 4.1 CHANNEL CONTROL

This parameter is used by the function to initialize the channel. It must be written by the CPU prior to issuing a host service request and assigning priority to the channel. The only legal values for this parameter are shown in **Figure 2**. Any other values cause indeterminate operation. Bits 9–15 are not used and are ignored.

### 4.2 PSC

These bits are used by the initialization state to configure the channel. Since this is an input function these bits must always be set to the NIL value (11).

### 4.3 PAC

These bits are used during function initialization to configure the transition detector. Although the detector itself can be set to detect rising edges, falling edges, both rising and falling edges, or to not detect any edge, FQM deals only with rising or falling edges, and thus recognizes only the PAC values that correspond to these two cases. The edge type selected must also be the same as the edge specified by host sequence bit 1. If the two selections are not the same, the function does not perform correctly.

### 4.4 TBS

These bits are used during initialization to select the timebase for the function. Either TCR1 or TCR2 can be selected.

### 4.5 WINDOW_SIZE

This parameter specifies the duration of the accumulation window. It is written by the CPU. Duration is specified in TCR clock ticks. The maximum value is $8000. Minimum window time is based on the service latency of the function, which varies according to the type and number of functions active at any one time. This parameter must be written prior to issuing a host service request and assigning priority to the channel. Once the channel is enabled, window size can be changed at any time while the channel is running. The new value takes effect when current window time expires.

### 4.6 PULSE_COUNT

This is the 16-bit result register for the function. It contains the number of pulses detected during the previous window. Since the register is written by the TPU at the end of each window time, in continuous mode the CPU has only one window time in which to read the stored value. In single shot mode, the value remains in the register until the next window accumulation is scheduled and that window time ends.

### 4.7 IN_WINDOW_ACCUMULATION

FQD uses this 16-bit location to store a running pulse accumulation value during each window time. The value is reset to zero at the beginning of each window. IN_WINDOW_ACCUMULATION is a scratchpad value. CPU reads of the register do not affect function operation, but CPU writes can corrupt an accumulation in progress.

## 5 Host Interface to FQM Function

This section provides information concerning the TPU host interface to the FQM function. Figure 3 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 3 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] — FQM 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 | Operating Mode |
|---|---|
| 00 | Begin with Falling Edge - Single-Shot Mode |
| 01 | Begin with Falling Edge - Continuous Mode |
| 10 | Begin with Rising Edge - Single-shot Mode |
| 11 | Begin with Rising Edge - Continuous 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 | Initialization |
|---|---|
| 00 | No Host Service (Reset Condition) |
| 01 | Undefined |
| 10 | Initialize |
| 11 | Undefined |

**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 | 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 |
|---|---|
| 0 | Channel interrupt not asserted |
| 1 | Channel interrupt asserted |

## 6 Configuration of FQM Function

The CPU configures the FQM function as follows:

1. The appropriate channel priority bits are cleared, disabling the channel.
2. The FQM function number is written to the channel function select bits.
3. CHANNEL_CONTROL and WINDOW_SIZE are written to channel parameter RAM.
4. The host sequence bits are written, selecting the desired action edge and mode of operation.
5. An HSR is issued to initialize the function.
6. The channel priority bits are written to enable the function and assign channel priority.
7. The TPU executes the initialization state.

After initialization, the TPU waits for the first selected edge to begin the time window. At the expiration of the time window, the accumulated value is written to the PULSE_COUNT parameter and an interrupt service request is made. In single-shot mode, the function then goes to an idle state. In continuous mode, the function immediately begins pulse accumulation in a new window. In continuous mode an interrupt service request is made at the completion of every window time.

Once single-shot mode has completed, another single-shot sequence can be scheduled by issuing an initialization HSR. Mode and window time parameters can be modified before writing to the HSR register. If an initialization HSR is made prior to the expiration of a current time window, that accumulation is aborted and a new accumulation begins.

During continuous mode operation, the window time parameter can be modified without re-initialization. The new time period takes effect as soon as the current window time expires. If an initialization HSR is made prior to the expiration of a current time window, the current accumulation is aborted and a new accumulation begins. To change mode during continuous mode operation, first disable the channel, then write the appropriate parameter registers and host sequence bits, issue an HSR, and then write the priority bits. This procedure prevents indeterminate results due to modification of sequence bits while the function is running.

# 7 Performance and Use of FQM Function

## 7.1 Performance

Since microcode must execute whenever an edge is detected, there is a minimum pulse period which guarantees that all pulses will be counted. When a single FQM function is in use and no other TPU channels are active, the absolute minimum pulse width is 22 CPU cycles plus a TST time.

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 and the information in the FQM state timing table below.

**Table 1 Frequency Measurement Function — State Timing**

| State Number and Name | Max CPU Clock Cycles | RAM Accesses by TPU |
|---|---|---|
| S1 INIT_FQM | 6 | 1 |
| S2 FIRST_EDGE_FQM | 8 | 2 |
| S3 COUNT_EM_FQM | | |
| Match only | 16 | 4 |
| Transition only | 12 | 2 |
| Match and Transition | 22 | 4 |

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

## 7.2 Changing Mode

The host sequence bits are used to select the FQM function operating mode. Change host sequence bit values only when the function is stopped or disabled (channel priority bits =%00). Disabling the channel before changing mode avoids conditions that cause indeterminate operation.

## 8 Frequency Measurement Examples

The FQM function counts pulses within a user-specified time window. The following examples show the capabilities of the function. Each example includes a description of the example, a diagram of initial parameter RAM content, initial control bit settings, and a diagram showing the relationship between the window and the pulses. Assume a pulse period of 75 TCR1 clock ticks. This is not required to set up the function but is used to illustrate what happens to partial pulses and edges concurrent with the end of window time. Unless otherwise noted, all examples use TPU channel 0.

### 8.1 Example A

#### 8.1.1 Description

Single-shot mode. Count the number of pulses beginning with a falling edge using TCR1. Accumulate for 500 ($1F4) TCR1 clock ticks and stop. Store the accumulated value in location $YFFF08.

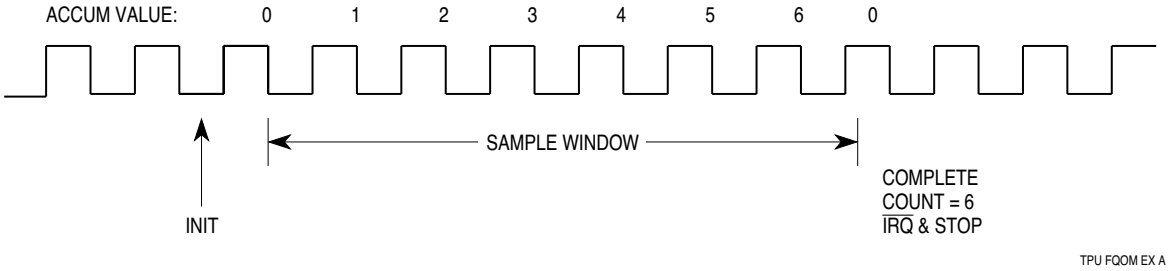
#### 8.1.2 Initialization

Load parameter RAM as shown. Write HSQ =%00, then issue HSR =%10 to initialize. Enable channel interrupt, select the function in the channel function select register, and set the channel priority bits to start the function.

**Table 2 FQM Channel Parameter RAM**

| | 15 | | | | | | | 8 | | | | | | | | 0 | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $YFFF00 | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | |
| $YFFF02 | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | |
| $YFFF04 | X | X | X | X | X | X | X | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | CH_CNTL |
| $YFFF06 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 0 | WIND_SZ |
| $YFFF08 | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | PLS_CNT |
| $YFFF0A | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | ACCUM |

#### 8.1.3 Timing Diagram



## 8.2 Example B

### 8.2.1 Description

Single-shot mode. Count the number of pulses beginning with a rising edge using TCR2. Accumulate for 300 ($12C) TCR2 clock ticks and stop. Store the accumulated value in location $YFFF08. Note that in this example the end of a pulse is coincident with the end of a window. If this edge passes through the digital filter before the window expires it is counted. However, if the edge has occurred but the digital filter has not qualified it, the pulse is not counted.

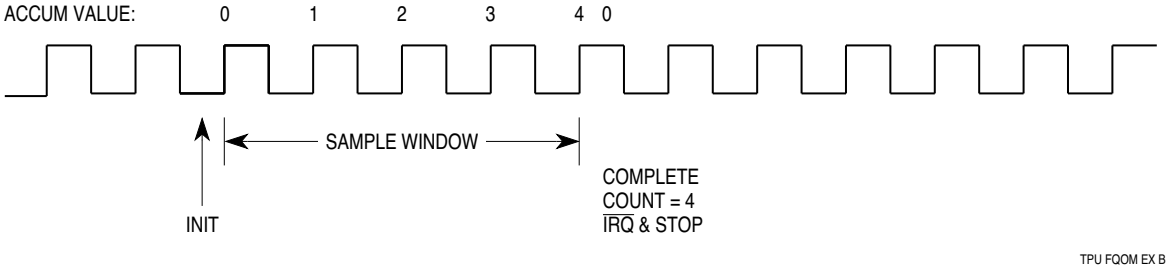
### 8.2.2 Initialization

Load parameter RAM as shown. Write HSQ =%10, then issue HSR =%10 to initialize. Enable channel interrupt, select the function in the channel function select register, and set the channel priority bits to start the function.

**Table 3 FQM Channel Parameter RAM**

| | 15 | | | | | | | 8 | | | | | | | | 0 | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $YFFF00 | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | |
| $YFFF02 | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | |
| $YFFF04 | X | X | X | X | X | X | X | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | CH_CNTL |
| $YFFF06 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | WIND_SZ |
| $YFFF08 | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | PLS_CNT |
| $YFFF0A | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | ACCUM |

### 8.2.3 Timing Diagram



### 8.3 Example C

#### 8.3.1 Description

Continuous mode. Count the number of pulses beginning with a falling edge using TCR2. Accumulate for 350 ($15E) TCR2 clock ticks. At the end of each window time, store the accumulated value in location $YFFF08.

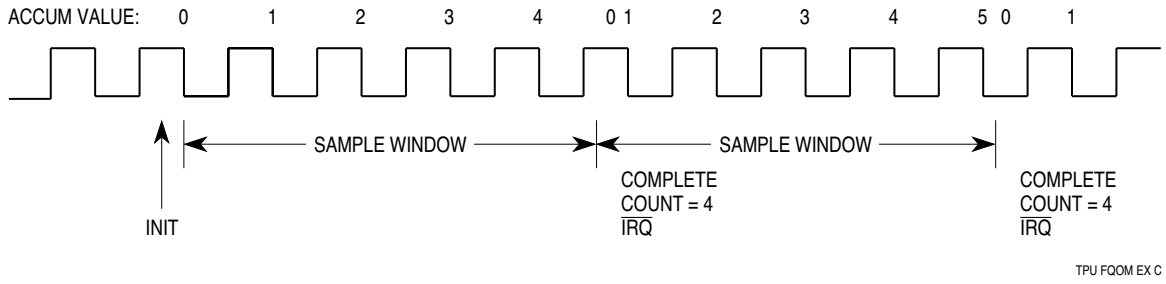
#### 8.3.2 Initialization

Load parameter RAM as shown. Write HSQ = %01, then issue HSR = %10 to initialize. Enable channel interrupt, select the function in the channel function select register, and set the channel priority bits to start the function.

**Table 4 FQM Channel Parameter RAM**

| | 15 | | | | | | | 8 | | | | | | | | 0 | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $YFFF00 | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | |
| $YFFF02 | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | |
| $YFFF04 | X | X | X | X | X | X | X | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 1 | CH_CNTL |
| $YFFF06 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | WIND_SZ |
| $YFFF08 | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | PLS_CNT |
| $YFFF0A | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | ACCUM |

#### 8.3.3 Timing Diagram



## 8.4 Example D

### 8.4.1 Description

Continuous mode. Count the number of pulses beginning with a rising edge using TCR1. Accumulate for 300 ($12C) TCR1 clock ticks and stop. Store the accumulated value in location $YFFF08. Note that in this example the end of a pulse is coincident with the end of a window. If this edge passes through the digital filter before the window expires it is counted. However, if the edge has occurred but the digital filter has not qualified it, the pulse will be counted in the next window time.

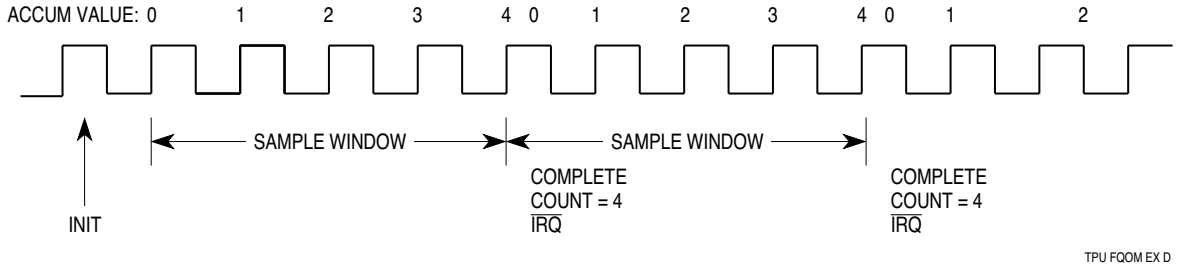
### 8.4.2 Initialization

Load parameter RAM as shown. Write HSQ = %11, then issue HSR = %10 to initialize. Enable channel interrupt, select the function in the channel function select register, and set the channel priority bits to start the function.

**Table 5 FQM Channel Parameter RAM**

| | 15 | | | | | | | 8 | | | | | | | | 0 | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $YFFF00 | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | |
| $YFFF02 | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | |
| $YFFF04 | X | X | X | X | X | X | X | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | CH_CNTL |
| $YFFF06 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | WIND_SZ |
| $YFFF08 | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | PLS_CNT |
| $YFFF0A | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | ACCUM |

### 8.4.3 Timing Diagram



# 9 FQM Algorithm

The FQM function consists of three states, which are described below. In the state descriptions, the PULSE_COUNT parameter contains the number of pulses counted during the last window time. The IN_WINDOW_ACCUMULATION parameter contains the number of pulses counted so far in the current window. The channel is configured to respond to either rising or falling edges. Figure 4 is a state flow diagram of the FQM function.

## 9.1 STATE1 — INIT_FQM

This state, entered as a result of HSR 10, performs initialization of the FQM function.

The pin is configured as an input per the PSC field in the CHANNEL_CONTROL parameter
Either rising or falling edges are selected per the PAC field in CHANNEL_CONTOL
TCR1 or TCR2 is selected per the TBS field in CHANNEL_CONTROL
Channel service is enabled
Flag0 is set to indicate that the next edge will be the first one since initialization
All latches are cleared
The state ends.

## 9.2 STATE2 — FIRST_EDGE_FQM

This state is entered as a result of a transition on the pin and Flag0=1.

MRL is tested (has an unexpected match occurred?)
If MRL = 1 (a match has occurred)
The match latch is cleared
The state ends
If MRL = 0 (the state was entered with a transition)
IN_WINDOW_ACCUMULATION is cleared
WINDOW_SIZE is added to the value in the capture register and written to match register
Flag0 is cleared to indicate that the first valid edge has been found
The transition latch is cleared
The state ends.

### 9.3 STATE3 — COUNT_EM_FQM

This state is entered on either a match or a transition on the pin. Since a match indicates the end of window time, all current pulses must be counted. Since transitions block the assertion of matches, the order of servicing determines whether a pulse is counted in the current window. If a match has occurred, a pending transition must have happened after the match, and must be counted in the next window. The algorithm always checks for transitions after matches so that transitions occurring close to matches can be counted without waiting for another TST. This allows for shorter pulse periods in heavily loaded applications.

```
MRL is tested (has a match occurred?)
    If MRL = 1 (a match has occurred)
        IN_WINDOW_ACCUMULATION is copied to PULSE_COUNT
        WINDOW_SIZE is added to the value in match register to create new end of window time
        The new end of window time is written to the match register
        The match latch is cleared
        The TPU requests interrupt service from the CPU.
        HSQ0 is tested
            If HSQ0 = 0 (this is single-shot mode)
                Further channel service requests are disabled
                Zero is written to IN_WINDOW_ACCUMULATION
                The state ends.
            If HSQ0 = 1 (this is continuous mode)
                TDL is tested (has a transition occurred?)
                    If TDL = 0 (no transition has occurred)
                        Zero is written to IN_WINDOW_ACCUMULATION
                        The state ends.
                    If TDL = 1(a transition has occurred)
                        IN_WINDOW_ACCUMULATION is incremented
                        The transition latch is cleared
                        The state ends.
    If MRL = 0 (a match has not occurred)
        TDL is tested (has a transition occurred?)
            If TDL = 0 (no transition has occurred)
                Zero is written to IN_WINDOW_ACCUMULATION
                The state ends.
            If TDL = 1 (a transition has occurred)
                IN_WINDOW_ACCUMULATION is incremented
                The transition latch is cleared
                The state ends.
```

**KEY:**

| HSR | M/TSR | LSR | PIN | FLAG0 | FLAG1 |
|---|---|---|---|---|---|
| XX | X | X | X | X | X |

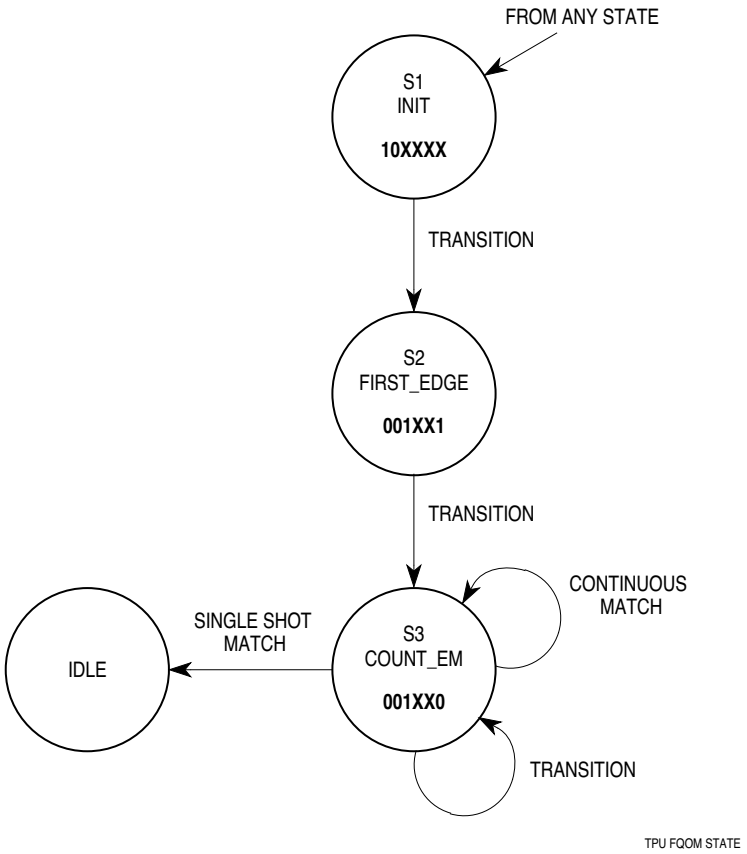



**Figure 4 FQM Function State Flow Diagram**

**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
**Mfax™:** RMFAX0@email.sps.mot.com - TOUCHTONE (602) 244-6609, U.S. and Canada Only 1-800-774-1848
**INTERNET:** http://Design-NET.com
**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