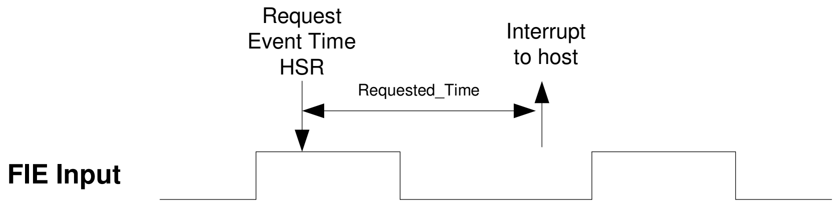
eTPU Function Implementation:

Frequency Input

(FIE)



MCD – 5401

Revision 5.0

May 8, 2006

DELPHI

CONFIDENTIAL

Microcode Technology
Powertrain Electronics Software Technology
M/S CT-40D, Delco Electronics, Kokomo, Indiana 46904-9005

Table of Contents

eTPU Function:
Frequency Input

1. Introduction

This document describes the Frequency Input with Event Time algorithm as implemented in the eTPU. It was designed to be used with any input signal for which edge count and frequency information are desired.

The frequency input portion of this primitive can record the time and accumulated counts for either rising or falling edges or both.

This primitive also supports an event time option. An interrupt is generated to the host when a desired event time match is achieved. The actual time of the event is recorded. The event time can be updated while in progress.

2. System Overview

2.1 Inputs

One frequency input is required for the operation of this function (see **Figure 1**).

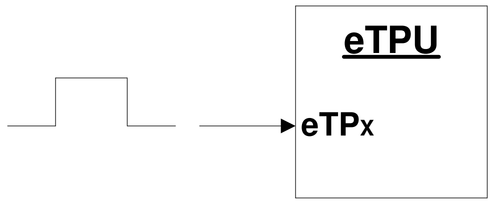


Figure 1 – Hardware Configuration

2.2 Outputs

There are no outputs produced by the operation of this function.

3. Memory Map

The memory map for the FIE function is shown in **Figure 2**.

Bit #:	**31................24**	**23................16**	**15..................8**	**7....................0**
	Flags	***Edge_Count***		
	bit 31: ***ET_Pending***			
	(not used)	***Edge_Time***		
	(not used)	***Requested_Time***		
	(not used)	***Actual_Event_Time***		

HSR7 (%111):	**Shutdown**
HSR6 (%110):	**Initialize (detect rising edges)**
HSR5 (%101):	**Initialize (detect falling edges)**
HSR4 (%100):	**Initialize (detect either edges)**
HSR3 (%011):	**Request Event Time**
HSR2 (%010):	**Update Event Time**
HSR1 (%001):	**(not used)**

FM0:	***FI_Use_TCR2***	**FLAG0:**	**(not used)**
	0 = False (use TCR1 as timebase)		**0 =**
	1 = True (use TCR2 as timebase)		**1 =**
FM1:	***ET_Use_TCR2***	**FLAG1:**	**(not used)**
	0 = False (use TCR1 as timebase)		**0 =**
	1 = True (use TCR2 as timebase)		**1 =**

*** Note Flags are write only**

Figure 2 – Memory Map

3.1 RAM Definition

The parameter RAM definitions are summarized in **Table 1**.

Parameter	**Range**	**Units**	**CPU Access**	**eTPU Access**
Parameter RAM				
ET_Pending	0 – 1	Flag	-	R/W
Edge_Count	0 – 0xFFFFFF	Counts	R	R/W
Edge_Time	0 – 0xFFFFFF	TCRx Units	R	R/W
Requested_Time	0 – 0xFFFFFF	TCRx Units	R/W	R
Actual_Event_Time	0 – 0xFFFFFF	TCRx Units	R	R/W
***FI_Use_TCR2* (FM0)**	0 – 1	Flag	R/W	R
***ET_Use_TCR2* (FM1)**	0 – 1	Flag	R/W	R

Table 1 - RAM Definitions

3.1.1 CPU Write / eTPU Read

The following parameters are written by the CPU and read by the eTPU:

Requested_Time	The time in TCRx counts of a requested event. The host is notified with an interrupt when the actual event occurs.

3.1.1.1 Function Mode Bits

The two Function Mode (FM) bits are located in the ETPUCxSCR register. They are written by the CPU and read by the eTPU:

***FI_Use_TCR2* (FM0)**	Cleared (0) if the time base for the frequency input processing is to be TCR1. Set (1) if the time base is to be TCR2.
***ET_Use_TCR2* (FM1)**	Cleared (0) if the time base for the event time is to be TCR1. Set (1) if the time base is to be TCR2.

3.1.2 eTPU Write / CPU Read

The following parameters are written by the eTPU and read by the CPU:

Edge_Count	Accumulated count of the critical edges. Either rising, falling or both are counted depending on the initialization HSR used.
Edge_Time	Time of the last detected edge in TCRx counts.

NOTE: *Edge_Count* and *Edge_Time* are always updated coherently by the eTPU.

Actual_Event_Time	The actual time of the match event specified by ***Requested_Time***. The host is interrupted at this time.

3.1.2.1 eTPU Write / eTPU Read

ET_Pending	Flag used by eTPU indicating a requested match event has been setup.

3.1.2.2 Channel Flags

The Channel Flags are not used in this primitive.

3.2 Initialization

The CPU should initialize the FIE function as follows:

1. Write the encoded value for the FIE primitive to the channel's field within the correct channel function select register (CFSx). See the Application Implementation document.
2. Set/clear the ***FI_Use_TCR2*** and ***ET_Use_TCR2*** bits appropriately.
3. Issue an **Initialize** Host Service Request based on the desired critical edge(s).
    a. (%110) – Detect rising edges
    b. (%101) – Detect falling edges

    c. (%100) – Detect either edges

4. Enable the FIE channel by assigning to it a non-zero priority (CPR > %00).

When an **Initialize** HSR is issued, the eTPU will respond by configuring the FIE channel to detect the edges specified by the initialization HSR.

4. FIE Operation

4.1 Frequency Input Operation

The FIE channel hardware is initialized to capture both timebases when a critical edge occurs (as specified with the initialization HSR). When the edge occurs, the eTPU:

1. Reads the event time in the timebase specified by ***FI_Use_TCR2*** flag (FM0) and writes to ***Edge_Time***.
2. Increments ***Edge_Count***.
3. Issue a DMA.
4. Prepares hardware to receive next edge.

Edge_Time and ***Edge_Count*** are written coherently by the eTPU.

4.2 Event Time Match Operation

4.2.1 Requesting an Event Time Match

Event time match operation can run concurrently with the frequency input function. To initiate an event time match:

1. Select the timebase for the match event by setting the ***ET_Use_TCR2*** flag (FM1) to 0 for TCR1 timebase and 1 for TCR2.
2. Write the time of the event into ***Requested_Time*** in the selected timebase.
3. Issue a ***Request_Event_Time*** HSR (%011).

The eTPU responds to a ***Request_Event_Time*** HSR by simply setting up the event with the selected timebase. Once the event occurs, the eTPU:

1. Writes time of the match to ***Actual_Event_Time***.

2. Interrupts the host.

NOTE: If a critical edge occurs on the frequency input after the time of the match event but before the match is serviced, ***Actual_Event_Time*** will contain the time of the edge. The difference is at most the time from the match to the time to service the channel.

If a ***Request_Event_Time*** HSR is issued while a match is pending, the request is treated as an ***Update_Event_Time*** HSR.

4.2.2 Updating an Event Time Match

To update event time match information:

1. Write updated information to ***Requested_Time***.
2. Issue a ***Update_Event_Time*** HSR (%010).

The eTPU responds to the HSR by simply updating the match information. If the original event has already been processed, the ***Update_Event_Time*** HSR is ignored. If the original event match has occurred but not processed, the new information is cancelled when the original match is processed

.

4.3 Global Exception

Under certain error conditions, the FIE function will issue a global exception to the host and write its channel number to the global variable ***Cause_Of_Exception***. Below are the conditions that cause FIE to generate a global exception:

1. An HSR is issued to the FIE function that has not been defined.
2. A link to the FIE function occurs.

4.4 Shutting Down the FIE Function

This function can be disabled by issuing a **Shutdown** Host Service Request (%111), waiting for the HSR bits to clear, and setting the priority level to disabled (%00).

4.5 Switching from FIE to Another eTPU Function

To switch from the FIE function to another eTPU function, the CPU should shut down the FIE function (see **Section 4.4**) and then follow the directions for initializing the new function.

4.6 Function Latency

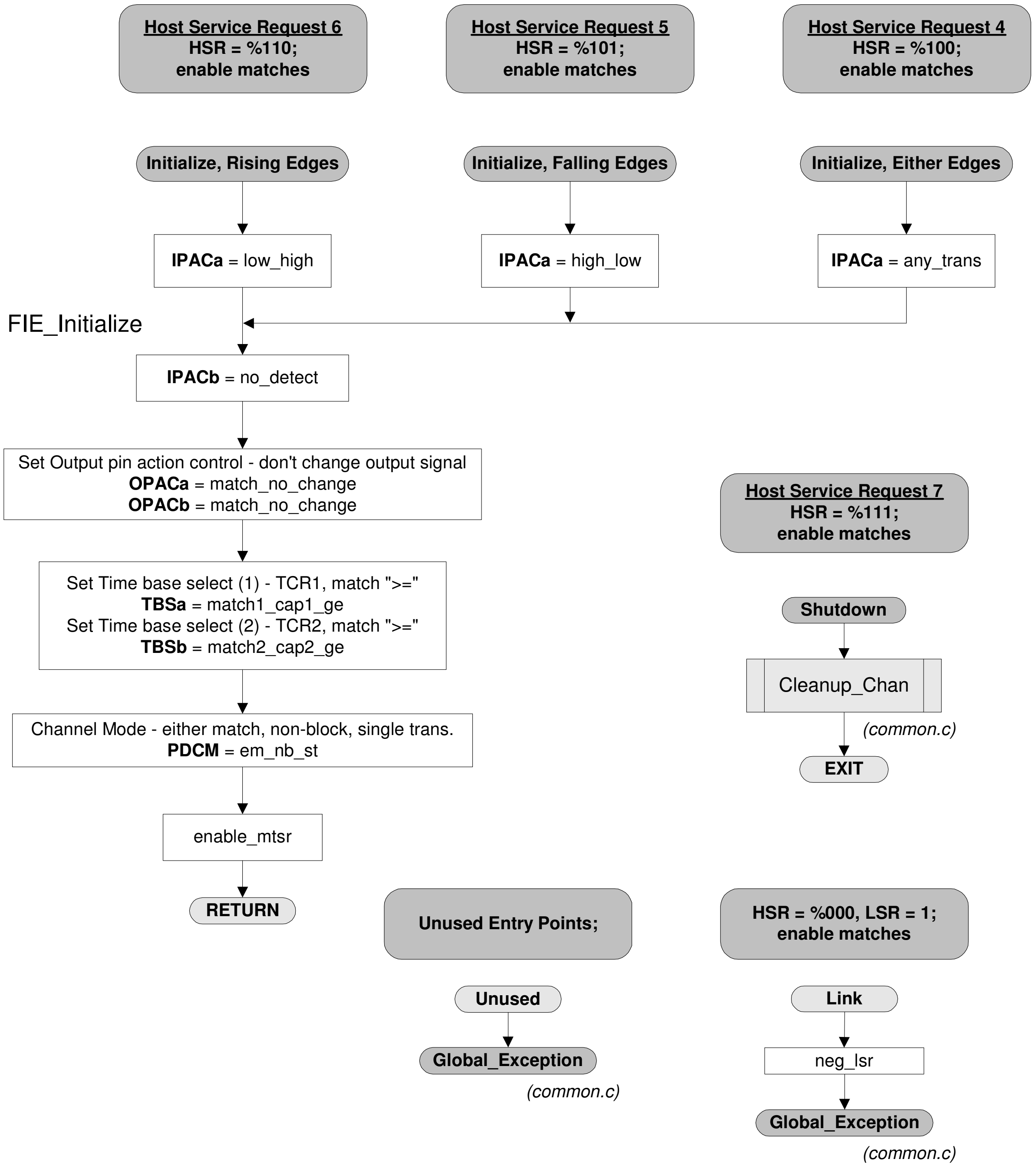
The worst-case execution times for the various threads of the PWMIE function are shown below (see the Application Implementation document to convert microcycles into real time based on crystal frequency). Note that these values do not take into account latencies due to other functions in the application, priorities, etc.

FIE Thread	Worst-Case µcycles *	RAM Accesses
Shutdown HSR (%111)	**8**	**0**
Initialize (detect rising edges) HSR (%110)	**4**	**0**
Initialize (detect falling edges) HSR (%101)	**4**	**0**
Initialize (detect either edges) HSR (%100)	**3**	**0**
Request Event Time HSR (%011)	**6**	**2**
Update Event Time HSR (%010)	**4**	**1**
FIE_Handle_M2_T1	**12**	**5**
FIE_Handle_M1	**14**	**5**
Link	**5**	**1**

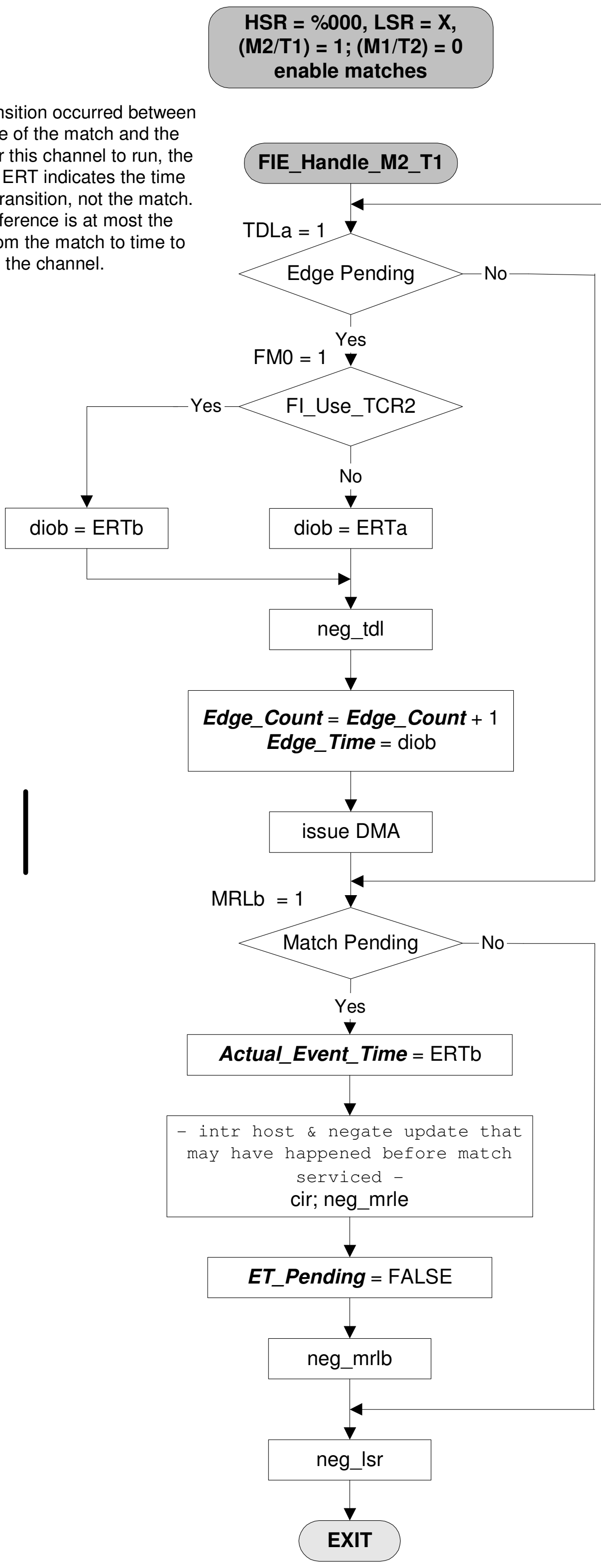
* These numbers do not consider potential collisions while accessing RAM.

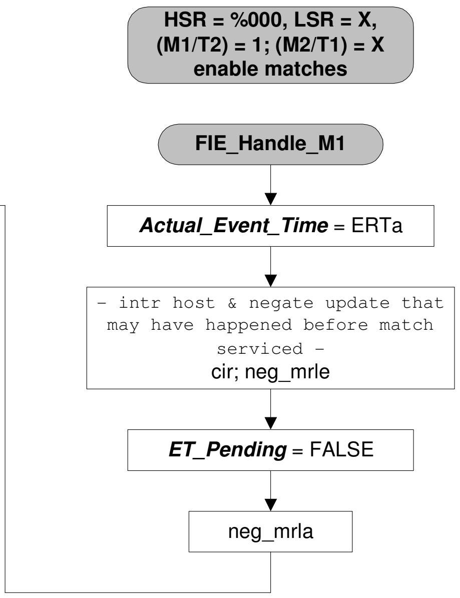
FIE primitive code size = 45 microcode instructions

5. Flowcharts

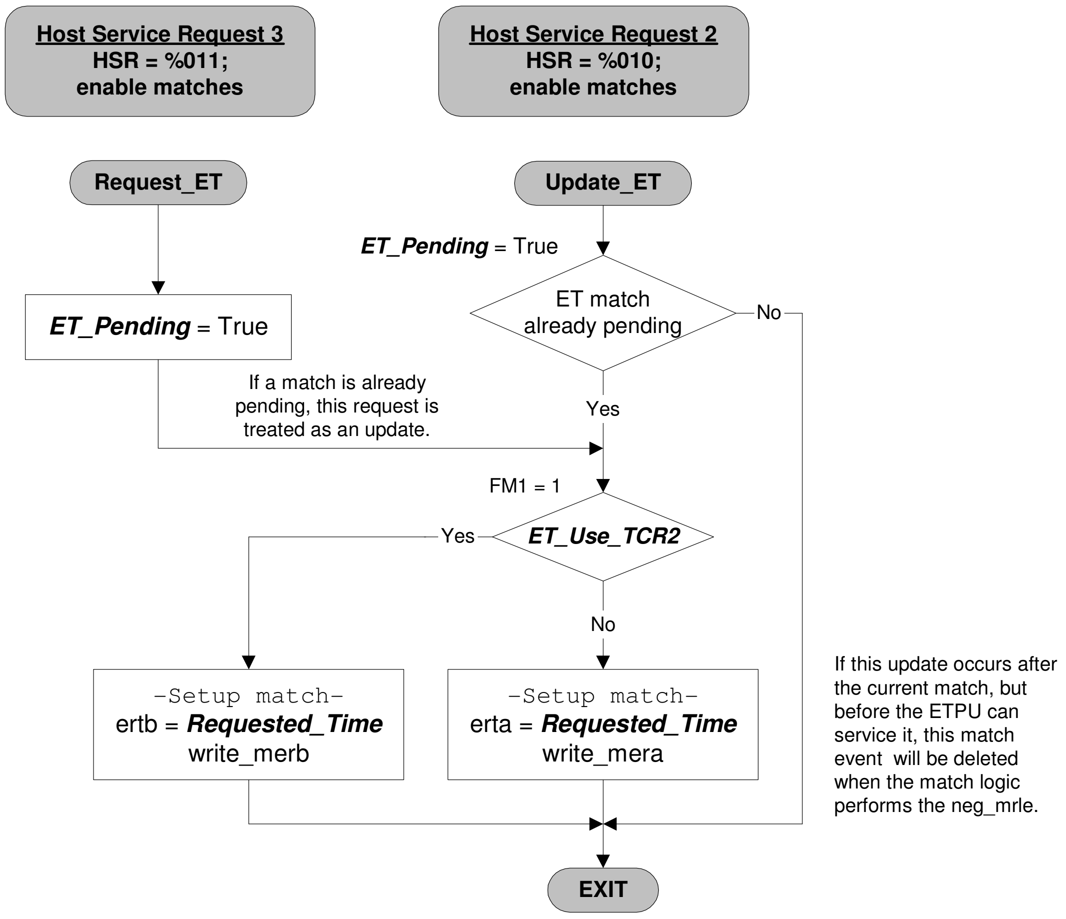


If a transition occurred between the time of the match and the time for this channel to run, the time in ERT indicates the time of the transition, not the match. The difference is at most the time from the match to time to service the channel.





M2/T1 may also be active with this entry table member. Check for pending transition (1)


Host Service Request 3
HSR = %011;
enable matches
Host Service Request 2
HSR = %010;
enable matches
Request_ET
Update_ET
ET_Pending = True
ET_Pending = True
ET match
already pending
No
Yes
If a match is already pending, this request is treated as an update.
FM1 = 1
ET_Use_TCR2
Yes
No
-Setup match-
ertb = Requested_Time
write_merb
-Setup match-
erta = Requested_Time
write_mera
If this update occurs after the current match, but before the ETPU can service it, this match event will be deleted when the match logic performs the neg_mrle.
EXIT

6. Revision Log

6.1 Document Revision Numbering

Each microcode document is assigned a revision number. The numbers are assigned according to the following scheme (used for all documents after May, 2003):

Rev x.y

x identifies the microcode, where

1 represents the <u>original</u> release of <u>microcode</u>
2 represents the first release of <u>changed microcode</u>
3 represents the second <u>change</u> to <u>microcode</u> etc.

y identifies the document, where

0 represents the <u>original</u> release of <u>documentation</u> for this microcode
1 represents the first <u>document change</u> for the <u>same microcode</u>
2 represents the second <u>change</u> to the <u>document</u> for the <u>same microcode</u>

6.2 Revision History

Revision	Date	Record	Author
1.0	09-11-03	Initial release of documentation.	Mary Hedges
2.0	05-21-04	Upgrade FIE primitive	William Ditty
3.0 (SCR 4109)	11-11-05	Corrected names associated with FM1 bit and removed include of common.h.	Warren Donley
4.0 (SCR 4520)	01-13-06	Converted from C to assembly language and changed operating mode to **em_nb_st**.	Warren Donley
5.0 (SCR 4718	05/08/06	Add DMA feature to input capture function.	Doug Tackitt