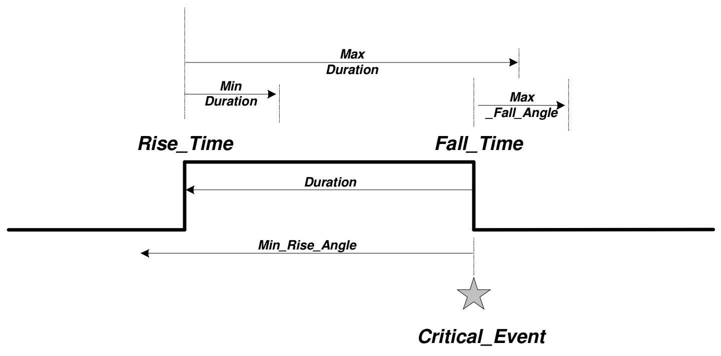
eTPU Primitive Implementation:

Multi-Pulse Time-Angle Combination for the eTPU

MPTACe



MCD – 5412

Revision 5.0

May 29, 2009

DELPHI

CONFIDENTIAL

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

TABLE OF CONTENTS

Multi-Pulse Time-Angle Combination Algorithm

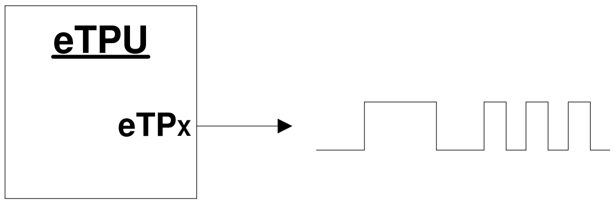
1. Introduction

This algorithm may be used to generate an output pulse, or a series of output pulses on any eTPU channel. The timing of the initial output pulse is based on the pulse width (or duration) and either the end of the pulse (called End – Duration Mode, or ED) or the start of the pulse (called Start - Duration Mode, or SD). The host may specify the falling edge (in ED mode) or rising edge (in SD mode) as either an angle or as a time.

A configurable number of trailing pulses are also available where the high time and low times are specified.

Either End - Duration Mode (ED) or Start - Duration Mode (SD) can be performed on any channel, and both can be used on the same application.

2. System Overview



2.1 Inputs

There are no inputs required for the operation of the MPTACe algorithm, unless the ***Critical_Event*** is specified as an angle by setting the function mode flag ***MPTACe_In_Angle_Mode***. In this case, the MPTACe primitive will use the global pointer ***pEPPE_Engine_Data*** (see Application Implementation document) to read engine position data from the EPPE primitive in order to perform a time-to-angle conversion.

2.2 Outputs

There is one output produced by the operation of this algorithm. When the output is high (1) it is defined as being "on", and when the output is low (0) it is defined as being "off".

2.3 End – Duration (ED)

In End_Duration mode, the host provides the falling edge event in either a time or an angle. The rise time is specified as an offset from the falling edge. **Figure 1** illustrates the pulse in both time and angle modes along with the relationship of the variables associated with each mode. Details are available in section 4.1.1.

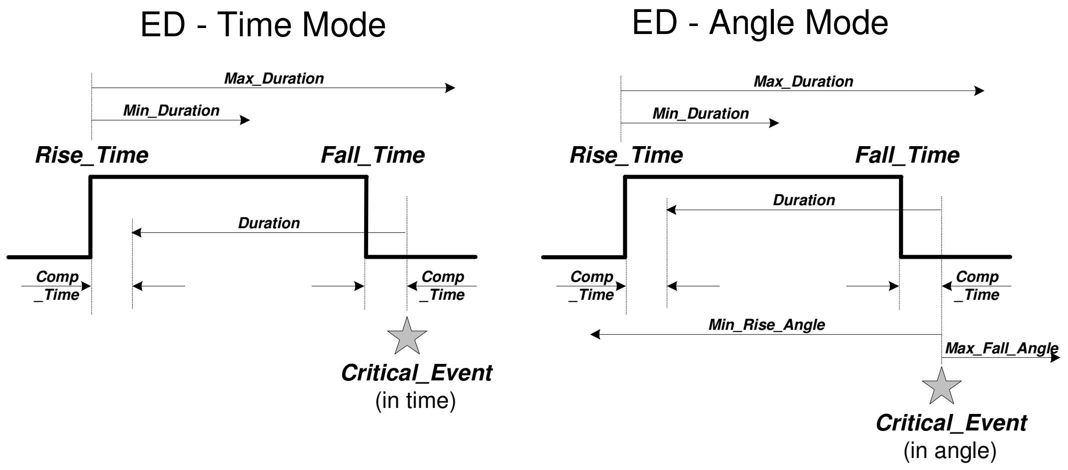


Figure 1: Initial MPTACe Pulse in ED Mode

2.4 Start – Duration (SD)

In SD mode, the host specifies the rising edge in either a time or an angle. The high time of the pulse is specified by an offset from the rising edge. **Figure 2** illustrates the pulse in both time and angle modes along with the relationship of the variables associated with each mode. Details are available in section 4.2.

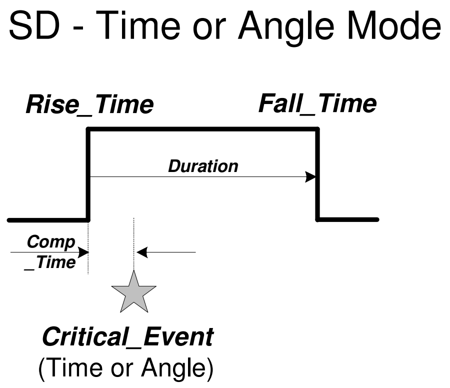
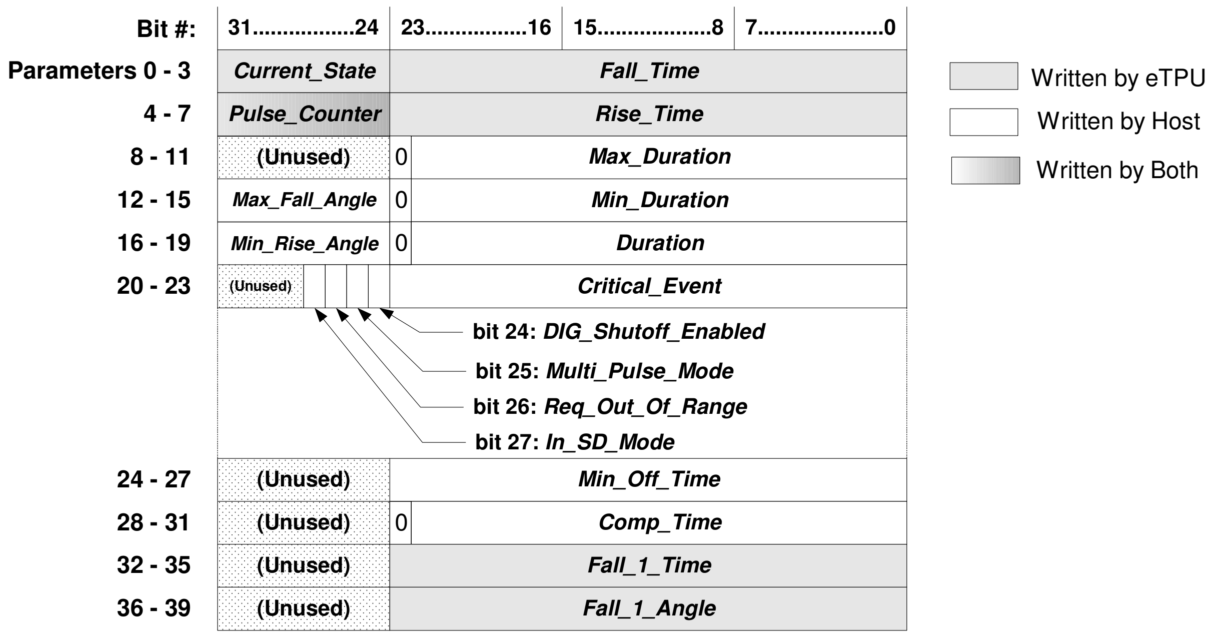


Figure 2: Initial MPTACe Pulse in SD Mode

3. Host CPU Interface

This section describes how the MPTACe function communicates with its host CPU. The parameters required to execute the algorithm are listed and described, and the initialization necessary for operation is detailed.

3.1 MPTACe Parameter RAM Map



Bit #:	31................24	23................16	15..................8	7....................0
Parameters 0 - 3	*Current_State*	*Fall_Time*		
4 - 7	*Pulse_Counter*	*Rise_Time*		
8 - 11	(Unused)	0 *Max_Duration*		
12 - 15	*Max_Fall_Angle*	0 *Min_Duration*		
16 - 19	*Min_Rise_Angle*	0 *Duration*		
20 - 23	(Unused)	*Critical_Event*		
24 - 27	(Unused)	*Min_Off_Time*		
28 - 31	(Unused)	0 *Comp_Time*		
32 - 35	(Unused)	*Fall_1_Time*		
36 - 39	(Unused)	*Fall_1_Angle*		

bit 24: *DIG_Shutoff_Enabled*
bit 25: *Multi_Pulse_Mode*
bit 26: *Req_Out_Of_Range*
bit 27: *In_SD_Mode*

Written by eTPU
Written by Host
Written by Both

Host Service Requests :

HSR 7:	Shutdown
HSR 6:	Request
HSR 5:	Initialization
HSR 4:	Update
HSR 3:	Abort_Request
HSR 2:	(Unused)
HSR 1:	(Unused)

Function Mode Bits:

FM0 : ***MPTACe_Use_TCR2***
0 = TCR1 timebase
1 = TCR2 timebase

FM1: ***MPTACe_In_Angle_Mode***
0 = False, in time mode
1 = True, in angle mode

Entry Point Flags:
Flag0: (Unused)

Interrupts:
Falling edge of final pulse processed

Channel Mode (for angle mode):
Rising Edge and wakeup, 1st Pulse
sm_st - **S**ingle **M**atch, **S**ingle **T**ransition
Falling Edge and wakeups, 1st Pulse
em_b_st - **E**ither **M**atch, **B**locking, **S**ingle **T**ransition
During Extra Pulses
m2_o_st - Service on **2**nd **M**atch, **O**rdered, **S**ingle **T**ransition
Final Time Off
sm_st - **S**ingle **M**atch, **S**ingle **T**ransition

Figure 3: Parameter RAM for MPTACe Primitive

3.2 Global RAM for MPTACe primitive

The following global RAM parameters may be accessed by the MPTACe primitive. See the Application Implementation document for the microcode set for the actual memory locations of these parameters.

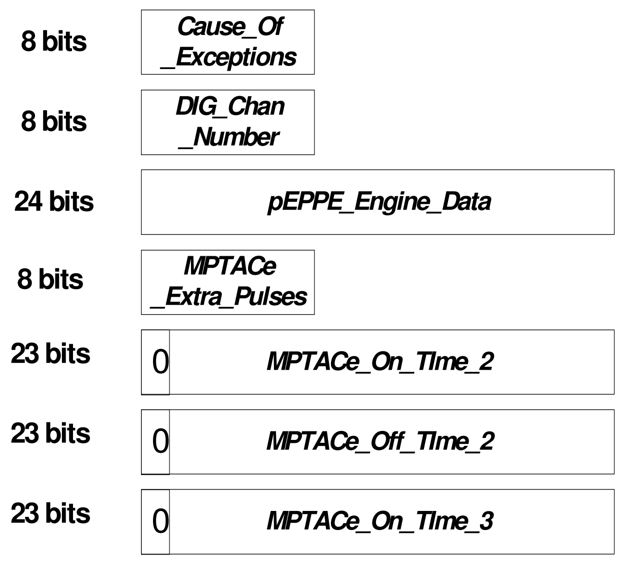


Figure 4: Global Variables Used by MPTACe

3.3 Parameter RAM Definition

Parameter	Size	Range	Units	CPU Access	eTPU Access
Local RAM Definition					
Fall_Time	24 bits	0x0 - 0xFFFFFF	TCRx Counts	R	R/W
Pulse_Counter	8 bits	0x0 - 0xFF	Counts	R/W	R/W
Rise_Time	24 bit	0x0 - 0xFFFFFF	TCRx Counts	R	R/W
Current_State	8 bits	0x0 - 0x07	Enumeration	R	R/W
Max_Duration	23 bits	0x0 - 0x7FFFFF	TCRx Counts	W	R
Min_Duration	23 bits	0x0 - 0x7FFFFF	TCRx Counts	W	R
Max_Fall_Angle	8 bits	0x0 - 0xFF	Integer Angle (# of teeth)	W	R
Duration	23 bits	0x0 - 0x7FFFFF	TCRx Counts	W	R
Min_Rise_Angle	8 bits	0x0 - 0xFF	Integer Angle	W	R

Parameter	Size	Range	Units	CPU Access	eTPU Access
			(# of teeth)		
Critical_Event	24 bits	0x0 - 0xFFFFFF	TCRx Counts or (Angle / 256)	W	R
In_SD_Mode	1 bit	0,1	Flag	W	R
Req_Out_Of_Range	1 bit	0,1	Flag	W	R
Multi_Pulse_Mode	1 bit	0,1	Flag	W	R
DIG_Shutoff_Enabled	1 bit	0,1	Flag	W	R
Min_Off_Time	24 bits	0x0 - 0xFFFFFF	TCRx Counts	W	R
Comp_Time	23 bits	0x0 - 0x7FFFFF	TCRx Counts	W	R
Fall_1_Time	24 bits	0x0 - 0xFFFFFF	TCRx Counts	R	R/W
Fall_1_Angle	24 bits	0x0 - 0xFFFFFF	(Angle Mode only): Angle / 256	R	R/W
Global RAM Definition					
DIG_Chan_Num	5 bits	0x0 - 0x1F	eTPU Channel #	W	R
pEPPE_Engine_Data	24 bits	0x0 - 0xFFFFFF	eTPU address	W	R
MPTACe_Extra_Pulses	8 bits	0x0 - 0xFF	Counts	W	R
MPTACe_On_Time_2	23 bits	0x0 - 0x7FFFFF	TCRx Counts	W	R
MPTACe_Off_Time_2	23 bits	0x0 - 0x7FFFFF	TCRx Counts	W	R
MPTACe_On_Time_3	23 bits	0x0 - 0x7FFFFF	TCRx Counts	W	R

Table 1 - MPTACe RAM Parameters and Access Restrictions

3.3.1 Parameters written by the Host CPU

3.3.1.1 Function Mode Bits

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

Use_TCR2:

FM0 – Set (1) if the time base is to be TCR2; cleared (0) if the time base is to be TCR1.

MPTACe_In_Angle_Mode:

FM1 – Set (1) if ***Critical_Event*** is specified as an angle; cleared (0) if ***Critical_Event*** is specified as a time.

3.3.1.2 Channel Relative Parameters Written by the Host CPU

Max_Duration: This is the maximum amount of time for the initial output pulse to be high. The eTPU will not allow the initial output pulse to remain high if ***Max_Duration*** has been met. This parameter is 23 bits long and has the same resolution as the channel, specified by ***MTPACe_Use_TCR2***. **In SD, this parameter is not used.**

Both ***Min_Duration*** and ***Max_Duration*** are used for placing the limits on the falling edge of the initial output pulse. These parameters are **not** used to calculate the placement of the rising edge.

Min_Duration: This is the minimum amount of time for the initial output pulse to be high. The eTPU will not allow the initial output pulse to fall until ***Min_Duration*** has been met. This parameter is 23 bits long and uses the time base specified by ***MPTACe_Use_TCR2***. **In SD mode, this parameter is not used.**

Max_Fall_Angle: In ED mode, this is the maximum offset in integer angles (or teeth) from ***Critical_Event*** that the initial output pulse is allowed to fall. This parameter is used when the falling edge needs to be extended until ***Min_Duration*** is reached, however, it cannot be extended more than ***Max_Fall_Angle*** from ***Critical_Event***. This parameter is 8 bits long and consists of an integer tooth count. This parameter is only significant when in ED mode, using angles **(*MPTACe_In_Angle_Mode* = 0).**

Duration: This is the amount of time for the initial output pulse to be high and is used to place the rising edge in ED mode and the falling edge in SD mode. This parameter is 23 bits long and has the same resolution as the channel.

Min_Rise_Angle: In ED mode, this is the minimum offset in integer angles (or teeth) prior to ***Critical_Event*** that the initial output pulse is allowed to rise. This parameter is 8 bits long and does not encompass fractional teeth. This parameter is only significant when in ED mode, using angles **(*MPTACe_In_Angle_Mode* = 0).**

Critical_Event: In ED, ***Critical_Event*** specifies the location of the falling edge of the first pulse in the series. In SD, ***Critical_Event*** specifies the location of the rising edge of the first pulse in the series. ***Critical_Event*** can be either a time or an angle, indicated by ***MPTACe_In_Angle_Mode***. When in angle mode, this variable is divided into two sections. The upper (16-bit) section, *Integer,* represents the pulse count where the falling edge of the initial output pulse should occur. The lower (8-bit) section, *Fraction,* is the portion of the next period where the falling edge of the initial output pulse should occur.

In_SD_Mode: A flag written by the host to indicate start-duration mode is requested. See **Section 4.2** for details on this mode.

Req_In_Future: A flag set by the host to indicated that the **Request HSR** sent when in time mode (MPTACe_In_Angle_Mode = 0) is for a pulse more than tcrx + 0x7FFFFF in the future. The eTPU does not set up the pulse, but waits for an **Update HSR**.

Multi_Pulse_Mode: If only one pulse is desired, set ***Multi-Pulse Mode*** = 0; if multi-pulse mode is desired, set ***Multi_Pulse_Mode*** = 1.

DIG_Shutoff_Enabled: If ***DIG_Shutoff_Enabled*** = 1, the MPTACe algorithm will, on the falling edge of the initial pulse, switch to the direct injection fuel channel selected by the global parameter ***DIG_Chan_Num*** and shut off any fuel pulse which may be in progress. If this feature is not desired, set ***DIG_Shutoff_Enabled*** = 0.

Min_Off_Time: In single-pulse mode, this is the minimum low (off) time after the end of the pulse; in multi-pulse mode, this is the desired low (off) time after the initial output pulse, as well as the minimum low (off) time after the end of the last pulse in the series.

Comp_Time: This is the amount of time the MPTACe signal needs to compensate for due to the filtering on the engine position input pin. Both the rising and falling edges are shifted by this amount of time, such that the duration is preserved.

3.3.1.3 Global Parameters Written by the Host CPU

DIG_Chan_Num: (global) This is the number of the eTPU channel on which it is desired to shut off fuel whenever the first falling edge of an MPTACe pulse series occurs (see ***DIG_Shutoff_Enabled***). ***DIG_Chan_Num*** will be ignored if ***DIG_Shutoff_Enabled*** = 0. This parameter is 5 bits long.

pEPPE_Engine_Data: (global) A pointer to the pulse count, critical edge time and period of the EPPE input signal used to convert ***Duration*** from a time to an angle.

MPTACe_Extra_Pulses: (global) The desired number of output pulses in a multi-pulse series to occur after the initial pulse has been delivered (eg, if a series of 5 pulses are desired, set ***MPTACe_Extra_Pulses*** = 4). This parameter is 8 bits long.

MPTACe_On_Time_2: (global) This 23 bit variable is the desired high (on) time for the second output pulse in a multi-pulse series.

MPTACe_Off_Time_2: (global) This 23 bit variable is the desired low (off) time after the second output pulse in a multi-pulse series (and all subsequent pulses in the series, if any, excluding the very last pulse in the series).

MPTACe_On_Time_3: (global) This 23 bit variable is the desired high (on) time for the third output pulse in a multi-pulse series (and all subsequent pulses in the series, if any).

3.3.2 Parameters written by both the Host and the eTPU

Pulse_Counter: Used only in multi-pulse mode (i.e., ***Multi_Pulse_Mode*** = 1). Counts the number of additional pulses to be delivered after the initial output pulse. ***Pulse_Counter*** is initialized to 0 at the falling edge of the final pulse in the series, and then increments by 1 at the end of each falling edge during the next multi-pulse sequence until the value of the global ***MPTACe_Extra_Pulses*** is reached (or passed). This parameter is 8 bits long. If the host desires to complete the initial pulse in a pulse stream, but abort the multi-pulses, it can set ***Pulse_Counter*** to ***MPTACe_Extra_Pulses***.

3.3.3 Parameters written by the eTPU

3.3.3.1 Channel Flags

No channel flags are used by the MPTACe primitive.

3.3.3.2 Channel Parameters Written by the eTPU

Fall_Time: Time of the falling edge of the initial MPTACe pulse in a series. If multiple pulses are selected, ***Fall_Time*** is also updated with the time of the falling edge of each remaining pulse in the series. ***Fall_Time*** is recorded in the time base selected by ***MPTACe_Use_TCR2***.

Rise_Time: Time of the rising edge of the initial MPTACe pulse in a series. In contrast to ***Fall_Time***, if multiple pulses are selected, ***Rise_Time*** is **not** updated with the time of the rising edge of each remaining pulse in the series. ***Rise_Time*** is recorded in the time base selected by ***MPTACe_Use_TCR2***.

Current_State: Variable used to track the various stages of the pulse series. Details on these states are included in section 4. The valid states are:

(0) WAITING_FOR_REQUEST, (1)RISING_WAKEUP_PENDING, (2)RISING_EDGE_PENDING, (3)FALLING_WAKEUPS_PENDING, (4)FALLING_EDGE_PENDING, (5)MIN_OFF_IN_PROGRESS, (6)MIN_OFF_W_REQUEST and (7)MULTI_PULSE_IN_PROGRESS.

Fall_1_Time: Time of the falling edge of the initial MPTACe pulse in a series. In contrast to ***Fall_Time***, if multiple pulses are selected, ***Fall_1_Time*** is **not** updated with the time of the falling edge of the remaining pulses in the series. ***Fall_1_Time*** is recorded in the time base selected by ***MPTACe_Use_TCR2***.

Fall_1_Angle: When ***MPTACe_In_Angle_Mode*** = 1, ***Fall_1_Angle*** will contain the angle of the falling edge of the initial MPTACe pulse in a series. In contrast to ***Fall_Time***, if multiple pulses are selected, ***Fall_1_Angle*** is **not** updated with the angle of the falling edge of the remaining pulses in the series. ***Fall_1_Angle*** is divided into two sections. The upper (16-bit) section, *Integer*, represents the pulse count where the falling edge of the initial output pulse occurred. The lower (8-bit) section, *Fraction,* is the portion of the next period where the falling edge of the initial output pulse occurred.

When ***MPTACe_In_Angle_Mode*** = 0, ***Fall_1_Angle*** will **not** contain valid angle information.

3.3.4 Other RAM Parameters

In **Angle Mode**, the eTPU will use one other RAM parameters: ***Period*** from the Engine Position Processing for eTPU Primitive (EPPE). This parameter is pointed to by the global parameter ***pEPPE_Engine_Data.*** Please refer to the Application Implementation document for more information on these parameters.

3.4 MPTACe Initialization

The CPU should initialize the MPTACe function as follows:

1. Clear parameter RAM.

2. Write the desired time base for the MPTACe channel to ***MPTACe_Use_TCR2*** (FM0) in the FM (channel function mode) field in the ETPUCxSCR register. The following definitions are provided:

   #define MPTACE_TIMEBASE_TCR1 (0)

   #define MPTACE_TIMEBASE_TCR2 (1)

3. Select whether the value in ***Critical_Event*** is in time or angle units by writing to ***MPTACe_In_Angle_Mode*** (FM1) in the FM (channel function mode) field in the ETPUCxSCR register. The following definitions are provided:

   #define MPTACE_TIME_MODE (0)

   #define MPTACE_ANGLE_MODE (1)

4. Enable interrupts from MPTACe to the host in one of two ways:

   i. If any interrupts from MPTACe to the host are desired, write %1 to the CIE bit for the MPTACe channel in the ETPUCIER register, otherwise write %0 to disable interrupts. Note the state of this bit is reflected in the ETPUCxCR register.

   OR

   ii. Interrupts can also be enabled from MPTACe to the host by writing %1 to the CIE bit in the ETPUCxCR register, otherwise write %0 to disable interrupts. Note the state of this bit is reflected in the ETPUCIER register.

5. ETPU Channel Configuration Register (ETPUCxCR):

   a. Select the standard entry table condition for the MPTACe primitive by writing a 0 to the ETCS field. This value is passed from the microcode set as:

      #define MPTACE_ENTRYTABLE_TYPE (0)

   b. Write the MPTACe function number to the Channel Function Select (CFS) field. The following definition is provided:

      #define MPTACE _FUNCTION_NUM (x)

      where 'x' value depends on the microcode set.

   c. Write the MPTACe channel's parameter base address to the CPBA field.

6. If multi-pulse mode is desired, write appropriate values to the global parameters ***MPTACe_On_Time_2***, ***MPTACe_Off_Time_2***, ***MPTACe_On_Time_3***, and ***MPTACe_Extra_Pulses***.

7. If the DIG fuel shut-off feature is to be used (see **Section 5.2**), write the number of the desired fuel channel to ***DIG_Chan_Num*** and set ***DIG_Shutoff_Enabled* = *1.*** Otherwise set ***DIG_Shutoff_Enabled*** = 0.

8. Issue an **Initialize** Host Service Request by writing (%101) to the HSR field in the ETPUCxHSRR register. The following definition is provided.

   #define MPTACE _HSR_INIT (5)

9. Enable the channel by assigning a priority to the CPR field of the ETPUCxCR register. Wait for the HSR bits to be cleared before issuing another HSR.

10. **End-Duration Mode**:

    When operating in ED, write the values of ***Critical_Event***, ***Min_Duration***, ***Max_Duration***, ***Duration***, ***Min_Off_Time*** and ***Comp_Time***. Also set ***In_SD_Mode*** = 0.

    If operating in angle mode, also write values to ***Min_Rise_Angle*** and ***Max_Fall_Angle.***

    - OR -

    Start-Duration Mode:

    When operating in SD, write the values of ***Critical_Event***, ***Min_Off_Time***, ***Comp_Time***, and ***Duration***. Also set ***In_SD_Mode*** = 1 to enable SD mode.

11. The host CPU can now issue a **Request** Host Service Request by writing (%110) to the HSR field in the ETPUCxHSRR register. The following definition is provided.

    #define MPTACE_HSR_REQUEST (6)

3.5 Request HSR

A **Request HSR** is used to initiate a pulse stream. When in time mode, the flag ***Req_Out_Of_Range*** is provided so the host software can request a pulse even though it is more than the range of the time bases in the future. An **Update HSR** would then need to be sent once the edge is within the range of the time base.

If a **Request HSR** is sent during the ***Min_Off_Time*** after the initial pulse (if ***Multi_Pulse_Mode*** = 0) or after the final pulse of the pulse stream (***Multi_Pulse_Mode*** = 1) then it is queued until the ***Min_Off_Time*** is complete. Once complete, the new pulse is instigated.

If a **Request HSR** is sent while a pulse stream is in progress, it is treated as an **Update HSR** (see **Section 3.6**).

3.6 Update HSR

An **Update HSR** can be sent at any time during the MPTACe pulse (or initial pulse when ***Multi_Pulse_Mode*** = 1), whenever new ***Duration, Min_Duration, Max_Duration, Comp_Time, Min_Rise_Angle*** or ***Max_Fall_Angle*** information is available. When the HSR is received, the eTPU re-calculates any pending matches.

When Multi-pulse mode is disabled (***Multi_Pulse_Mode*** = 0), an **Update HSR** is ignored once the initial pulse is complete and the signal is being held low for ***Min_Off_Time***.

Even though the **Update HSR** is ignored during the subsequent pulses when in multi-pulse mode (***Multi_Pulse_Mode*** = 1), the parameters specifying the widths of the pulses (***MPTACe_On_Time_2***, ***MPTACe_Off_Time_2*** and ***MPTACe_On_Time_3***) can be updated at any point in the pulse stream. The new value is used at the next rising edge of the following pulse.

Details on the steps taken when processing an **Update HSR** are included in section 4.1.2.

3.7 Abort HSR

If the host desires to abort a pulse or a pulse stream in progress, an abort HSR may be issued. The eTPU forces the pin low, transitions ***Current_State*** to WAITING_FOR_REQUEST, clears all flags and cancels any pending matches.

If the host desires to allow the initial pulse to complete, but abort the pulses specified by the global ***MPTACe_Extra_Pulses***, the host may set ***Pulse_Counter*** to ***MPTACe_Extra_Pulses***. The MPTACe primitive completes the pulse in progress and aborts any remaining pulses.

4. MPTACe Operation

4.1 End - Duration (ED)

4.1.1 General Operation - ED

The following three figures illustrate a typical end-duration pulse in either time or angle mode, along with how the signal can be limited by ***Min_Duration*** or ***Max_Duration.*** Next, two figures are provided to illustrate how additional limits are implemented on the pulse when in angle mode (only) by using ***Min_Rise_Angle*** and ***Max_Fall_Angle***. For simplicity, ***Comp_Time*** is assumed to be 0 for this section. Please refer to section 4.3 for details on how ***Comp_Time*** affects the MPTACe initial pulse.

The first figure illustrates the typical case along with the relationship of the parameters.

CASE 1: Typical

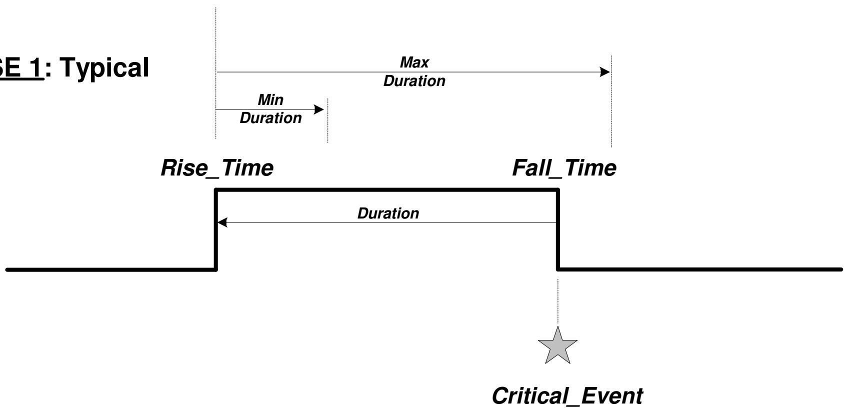


Figure 5: Typical MPTACe Output Signal

The second case can occur when acceleration after the rising edge is set up causes ***Critical_Event*** to occur earlier than expected. The actual high time is extended until the on time reaches the value of ***Min_Duration***.

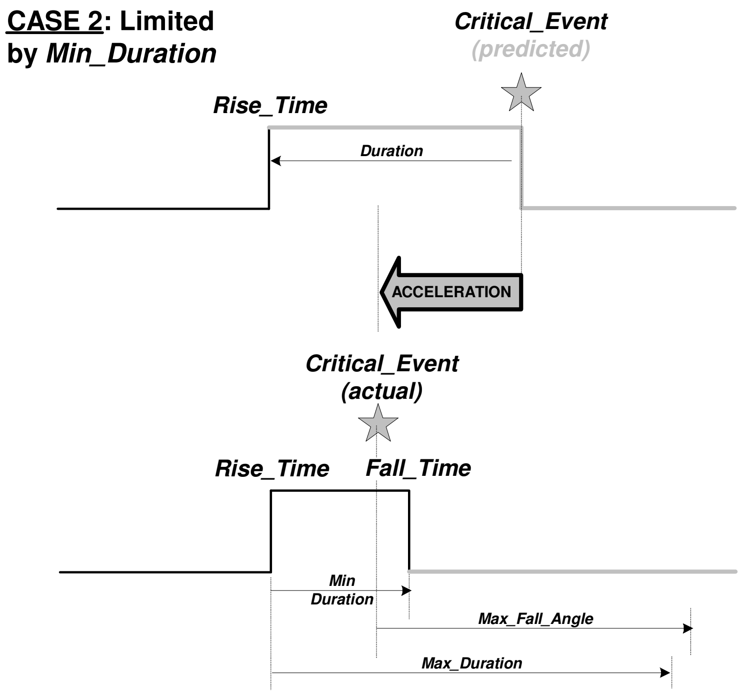


Figure 6: MPTACe Initial Pulse Limited by *Min_Duration*

Case 3 is similar to case 2, except this time a deceleration occurred after the rising edge was set up, causing ***Critical_Event*** to occur later than expected. In this case, the high time is limited by the parameter ***Max_Duration***.

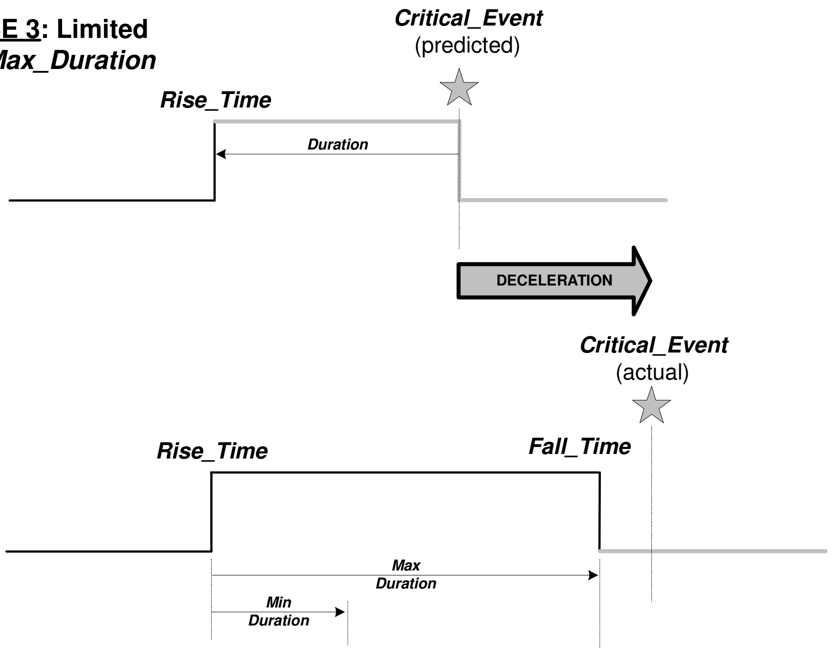


Figure 7: MPTACe Initial Pulse Limited by *Max_Duration*

Case 4 illustrates an additional limitation available when operating in angle mode. See **Figure 8**. In this case the projected rising edge is limited by ***Min_Rise_Angle***. This case is implemented to protect against the possibility of a noise pulse on the engine position input causing the Period of this signal to be extremely small. When MPTACe converts ***Duration*** from time to angle using this erroneously small period, the result would be a very large duration (angle). ***Min_Rise_Angle*** is a gross protection mechanism to prevent the calculated duration from causing the edge to rise too early.

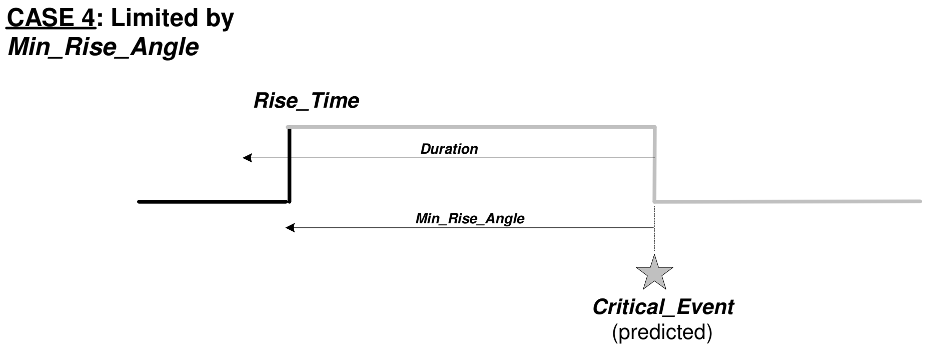


Figure 8: MPTACe Initial Pulse Limited by *Min_Rise_Angle*

The final case is similar to case 2 where the falling edge is extended until ***Min_Duration*** is met. However in this example, this new falling edge time is extended by ***Min_Duration***, but then limited by the parameter ***Max_Fall_Angle***. This limit is useful at times when the host requests massive changes to the duration of the MPTACe pulse, pushing the falling edge too far into the future.

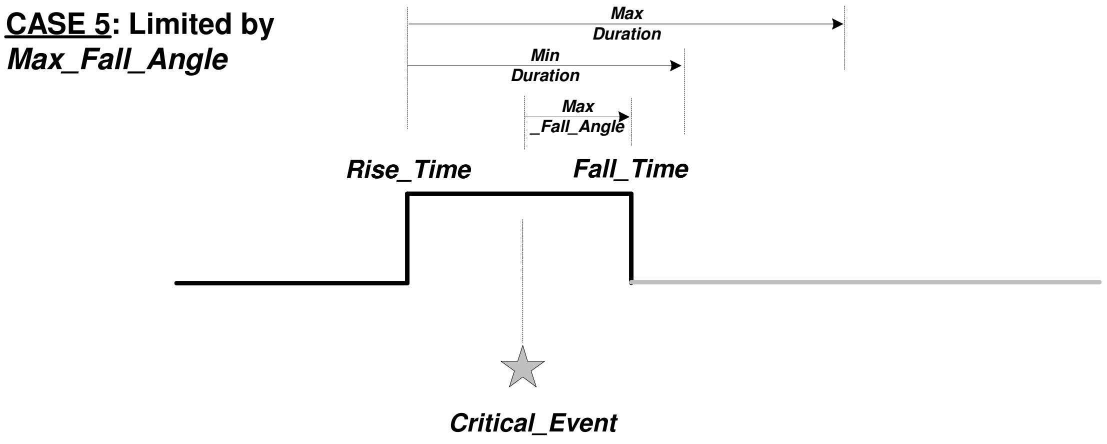


Figure 9: MPTACe Initial Pulse Limited by *Max_Fall_Angle*

4.1.2 Detailed Operation - ED in Angle Mode

Figure 10 lists the states the MPTACe algorithm transitions through while generating the initial MPTACe pulse when in End-Duration (angle) mode.

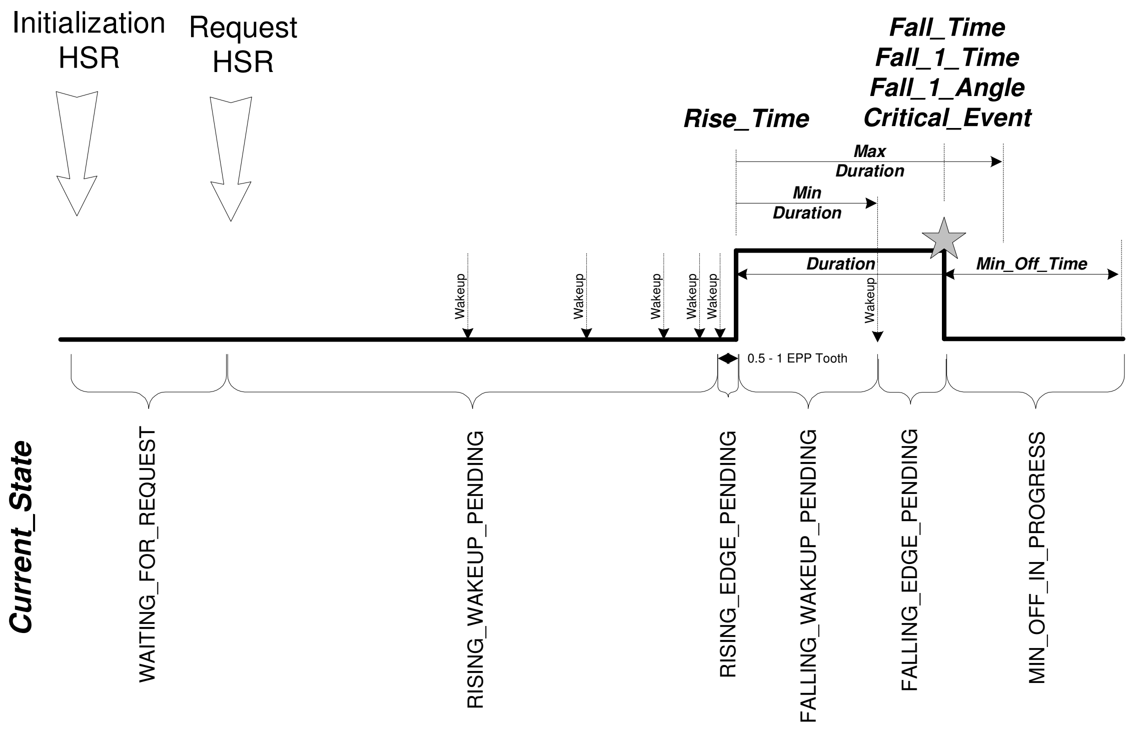


Figure 10: States of the Initial Pulse in Angle Mode

Figure 11 and **Figure 12** contain state transition diagrams for the ED pulse in angle and time mode, respectively. These diagrams explain what events cause the MTPACe algorithm to transition from one state to another.

Following are the details of the steps taken in each state when in angle mode.

WAITING_FOR_REQUEST:

No actions are performed in this state. The algorithm is simply waiting for either the first **Request HSR** since initialization completed or the MPTCe pulse has completed and the algorithm is waiting for a **Request HSR** for the next pulse.

Update HSRs are ignored in this state.

RISING_WAKE-UP_PENDING:

When this state is entered the eTPU performs the following steps.

1) Converts ***Duration*** from time to an angle and limits it ***Min_Rise_Angle***.

2) Calculates the desired wake-up angle as half of the distance to the projected rise angle.

3) If the wake-up angle is more than ½ a tooth in the future (or the rising edge is less than 1 tooth in the future), the rising edge wake-up match is set up.

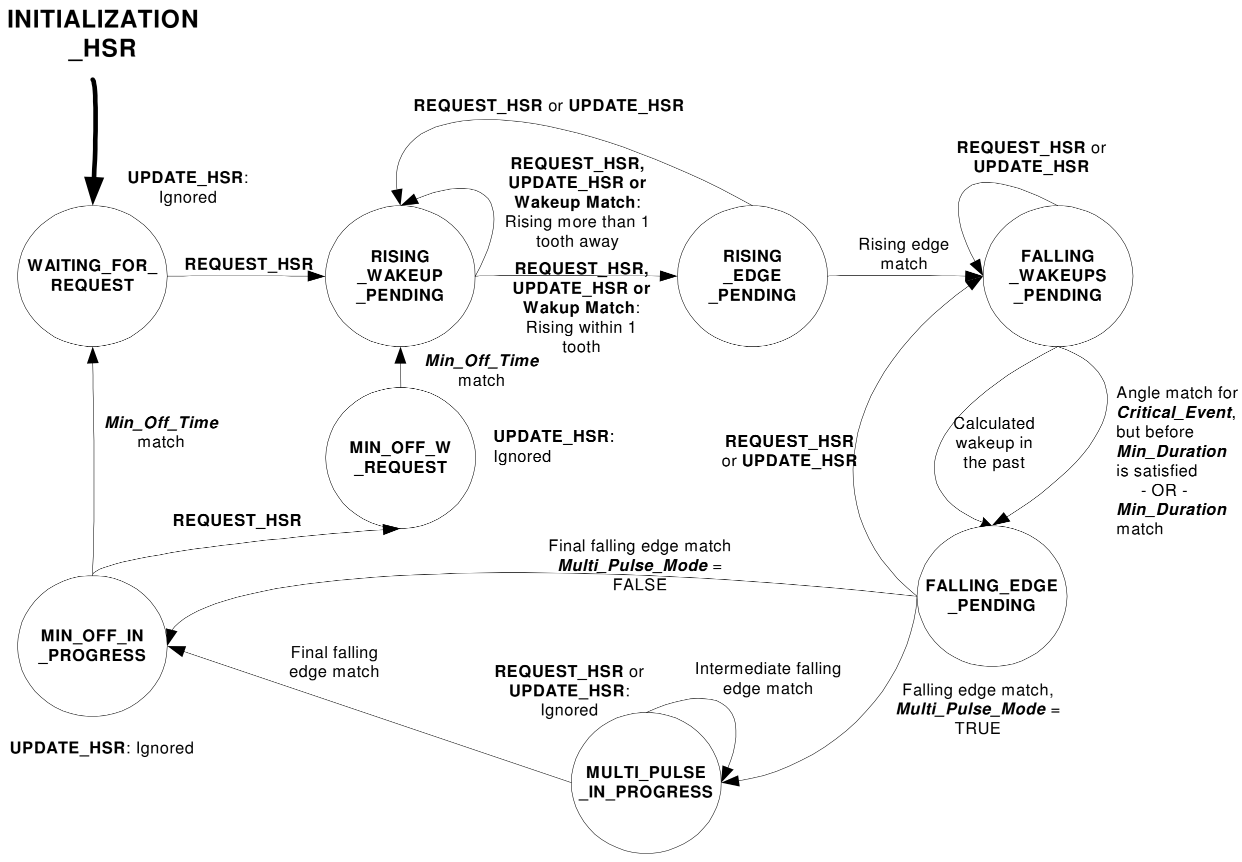
Steps 1 – 3 are repeated for each rising wake-up match until the eTPU is within 1 tooth of the projected rising edge and step 4 is performed.

4) Once the wake-up angle is within ½ a tooth of the actual rise angle (or the projected rising edge is less than 1 tooth in the future), the algorithm transitions to RISING_EDGE_PENDING where the rising edge is set up.

RISING_EDGE_PENDING:

When the state RISING_EDGE_PENDING is entered, the rising edge match is set up.

If an **Update HSR** is received while in this state, the state is changed back to RISING_WAKEUP_PENDING where the rising edge is re-calculated.



INITIALIZATION _HSR
UPDATE_HSR: Ignored
WAITING_FOR_ REQUEST
REQUEST_HSR
RISING _WAKEUP _PENDING
REQUEST_HSR or UPDATE_HSR
REQUEST_HSR, UPDATE_HSR or Wakeup Match: Rising more than 1 tooth away
REQUEST_HSR, UPDATE_HSR or Wakup Match: Rising within 1 tooth
RISING _EDGE _PENDING
Rising edge match
REQUEST_HSR or UPDATE_HSR
FALLING _WAKEUPS _PENDING
Min_Off_Time match
MIN_OFF_W _REQUEST
UPDATE_HSR: Ignored
Min_Off_Time match
REQUEST_HSR or UPDATE_HSR
Calculated wakeup in the past
Angle match for Critical_Event, but before Min_Duration is satisfied - OR - Min_Duration match
REQUEST_HSR
Final falling edge match Multi_Pulse_Mode = FALSE
FALLING_EDGE _PENDING
MIN_OFF_IN _PROGRESS
Final falling edge match
REQUEST_HSR or UPDATE_HSR: Ignored
Intermediate falling edge match
Falling edge match, Multi_Pulse_Mode = TRUE
UPDATE_HSR: Ignored
MULTI_PULSE _IN_PROGRESS

Figure 11: States for an MPTACe Pulse in ED, Angle Mode

FALLING_WAKE-UP_PENDING:

When this state is entered, ***Rise_Time*** is recorded based on the time base selected in MPTACe_Use_TCR2.

Two wake-up matches are then set up, neither of which affects the state of the pin. The first is a time based match set up for ***Min_Duration*** from the rising edge. The second is an angle based match set up for ***Critical_Event*** (– ***Comp_Time*** converted to angle).

If either of the above wake-up matches is in the past, ***Current_State*** is immediately transitioned to FALLING_EDGE_PENDING to set up the falling edge matches.

FALLING_EDGE_PENDING:

When the FALLING_EDGE_PENDING state is entered two matches are set up, both of which cause the MPTACe pin to fall. The occurrence of either match blocks the other from occurring. The matches that are set up, depends on whether the ***Min_Duration*** based match or the ***Critical_Event*** wake-up match occurred.

If the match based on ***Min_Duration*** has occurred, then the falling edge is set up with an angle match for ***Critical_Event*** (- ***Comp_Time*** converted to angle) and a time match to limit the falling edge by ***Max_Duration***.

If the match based on ***Critical_Event*** has occurred then the falling edge needs to be extended to fulfill the ***Min_Duration*** requirement, but not past ***Max_Fall_Angle***. A falling edge match based on time is set up for ***Min_Duration*** from the rising edge along with falling edge match based on angle for ***Max_Fall_Angle*** from the ***Critical_Event*** (- ***Comp_Time*** converted to angle).

If an **Update HSR** (or **Request HSR**) is received when in this state, the algorithm returns to the FALLING_WAKEUP_PENDING state where the falling edge (and falling wake-up matches) are re-calculated.

MULTI_PULSE_IN_PROGRESS:

When the MULTI_PULSE_IN_PROGRESS state is entered, ***Fall_Time***, ***Fall_1_Time*** and ***Fall_1_Angle*** are all recorded. Next, two matches are set up to form the next pulse in the pulse stream.

If this is the second pulse in the series, the first match is the rising edge match set up to guarantee the MPTACe signal is held low between events for at least

Min_Off_Time. The second match sets up the falling edge so the signal is held high for ***MPTACe_On_Time_2***.

If this is the third or more pulse in the multi-pulse series, the first match sets up the rising edge match so the MPTACe signal is held low for ***MPTACe_Off_Time_2***. The next match sets up the falling edge so the signal is held high for ***MPTACe_On_Time_3***. (See **Figure 14**.)

An **Update HSR** and/or **Request HSR** are ignored while in this state.

<u>MIN_OFF_IN_PROGRESS</u>:

When the MIN_OFF_IN_PROGRESS state is entered, the ***Fall_Time*** is recorded and an interrupt is sent to the host. The final match is then set up to guarantee the MPTACe signal is held low between events for at least ***Min_Off_Time***.

An **Update HSR** is ignored while ***Min_Off_Time*** is being processed.

If a **Request HSR** is received while in the MIN_OFF_IN_PROGRESS state, the request is queued by transitioning ***Current_State*** to MIN_OFF_W_REQUEST.

<u>MIN_OFF_W_REQUEST</u>:

No steps are performed when this state is entered.

Update HSRs and **Request HSRs** are ignored.

4.1.3 Detailed Operation: ED in Time Mode

If the host wishes to provide ***Critical_Event*** as a time instead of an angle, set ***MPTACe_In_Angle_Mode*** to False (0). The steps executed by the MTPACe primitive when in time mode are similar to those in angle mode except the wake-up states are not required. A state transition diagram is provided in **Figure 12**. Following are the details of the steps executed in each phase.

<u>WAITING_FOR_REQUEST</u>:

Follows the same steps as angle mode, except if the ***Request_Out_Of_Range*** flag is set, then the **Request HSR** completes without setting up the edge. An Update HSR must be issued with the flag ***Request_Out_Of_Range*** cleared to start the pulse.

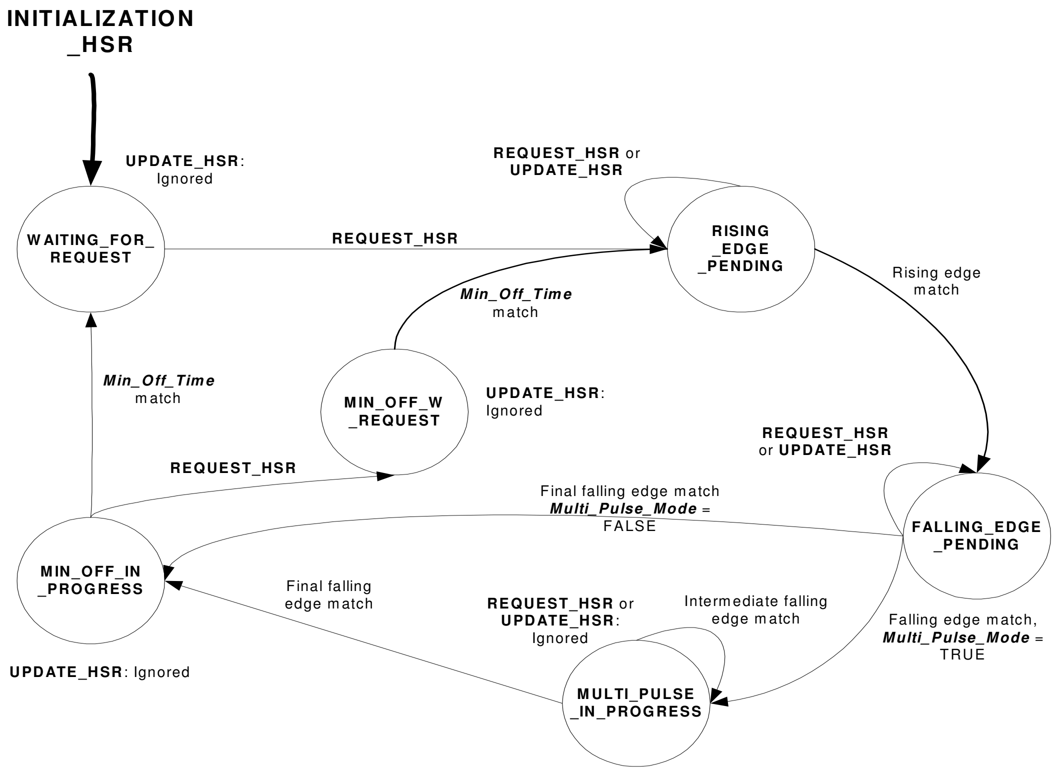


Figure 12: States of the Initial ED Pulse in Time Mode

and the Initial SD Pulse in either Time or Angle Mode

RISING_EDGE_PENDING:

When this state is entered, the rising edge is set up to occur at time **Duration** offset before the ***Critical_Event***, and shifted earlier in time by the value in ***Comp_Time***.

If an **Update HSR** is issued after the signal has risen, and the ***Request_Out_Of_Range*** flag is set, then the pulse is held high for the time specified in ***Max_Duration***.

FALLING_EDGE_PENDING:

The following steps are executed when this state is entered.

When this state is entered, the ***Rise_Time*** is recorded based on the time base selected in MPTACe_Use_TCR2. Next the requested falling edge time is calculated to be the ***Critical_Event*** (- ***Comp_Time***). This time is limited to the rising edge time plus ***Min_Duration*** and also the rising edge time plus ***Max_Duration*** and the falling edge match is set up.

The remaining states (MULTI_PULSE_IN_PROGRESS, MIN_OFF_IN_PROGRESS and MIN_OFF_W_REQUEST) are processed the same in time mode as in angle mode.

4.2 Start - Duration (SD)

4.2.1 General Operation: (SD)

Figure 2 illustrates an MPTACe initial pulse when in Start-Duration mode, for either time or angle modes.

4.2.2 Detailed Operation: (SD)

The state transition diagram in **Figure 12** illustrates not only the states for an End-Duration initial pulse in time mode, but also that for a Start-Duration pulse in either time or angle modes. Following are the steps that are taken when each state is entered when in SD mode.

WAITING_FOR_REQUEST:

Follows the same steps as angle mode.

RISING_EDGE_PENDING:

When this state is entered, the rising edge is set up to occur at time or angle specified by ***Critical_Event***, and depending on the function mode flag ***MPTACe_In_Angle_Mode***. The rising edge match is shifted earlier in time by the value in ***Comp_Time***. If ***MPTACe_In_Angle_Mode*** is set, then ***Comp_Time*** is converted to an angle before subtracting from ***Critical_Event***.

FALLING_EDGE_PENDING:

The following steps are executed when this state is entered.

When this state is entered, the ***Rise_Time*** is recorded based on the time base selected in ***MPTACe_Use_TCR2***. Next the falling edge match is set up to fall at time ***Rise_Time + Duration***.

The remaining states (MULTI_PULSE_IN_PROGRESS, MIN_OFF_IN_PROGRESS and MIN_OFF_W_REQUEST) are processed the same in any of the modes.

4.3 Compensation Time

The variable ***Comp_Time*** is used to compensate for the delay imposed by the filters on the engine position input pin. The placement of the MPTACe pulse (or pulse stream) is shifted by the amount of time in ***Comp_Time.*** **Figure 13** illustrates a typical MPTACe pulse shifted by ***Comp_Time***.

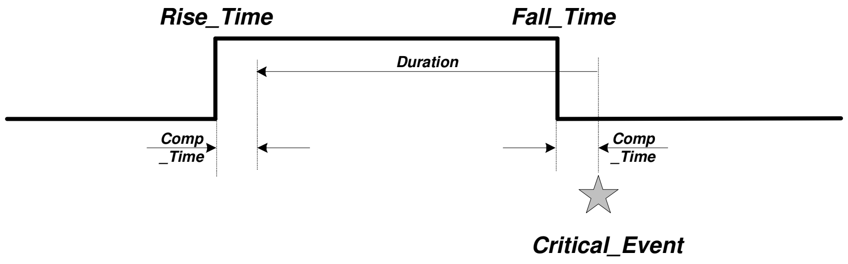


Figure 13: Shift of MPTACe Pulse by *Comp_Time*

4.4 Multi-Pulse Operation

Please refer to **Figure 9** for the following discussion. **This entire section deals exclusively with multi-pulse operation (i.e., *Multi-Pulse Mode* = 1).**

The MPTACe primitive may be configured to generate a series of pulses by setting ***Multi_Pulse_Mode*** flag to a one (1). **The distinctions between ED and SD only apply to the initial pulse in the series**. Once the initial pulse has finished, all additional pulses will be delivered entirely according to the values contained in the multi-pulse variables (***Min_Off_Time***, ***On_Time_2***, ***Off_Time_2***, ***TAC_On_Time_3***, and ***Extra_Pulses***).

If ***Extra_TAC_Pulses*** is set equal to 0, multi-pulse mode will behave exactly the same as single-pulse mode. The following description assumes that ***Extra_Pulses*** is non-zero.

At the falling edge of the initial pulse, the eTPU checks to see if multi-pulse mode is selected (***Multi_Pulse_Mode*** = 1). If so, it sets up two matches where only the completion of the second match is the eTPU triggered for service. The first is set for the signal to rise at time ***Rise_Time*** + ***Min_Off_Time***. The second is for the falling edge, and set up up for ***On_Time_2*** past the rising edge. ***Current_State*** is then transitioned to MULTI_PULSE_IN_PROGRESS.

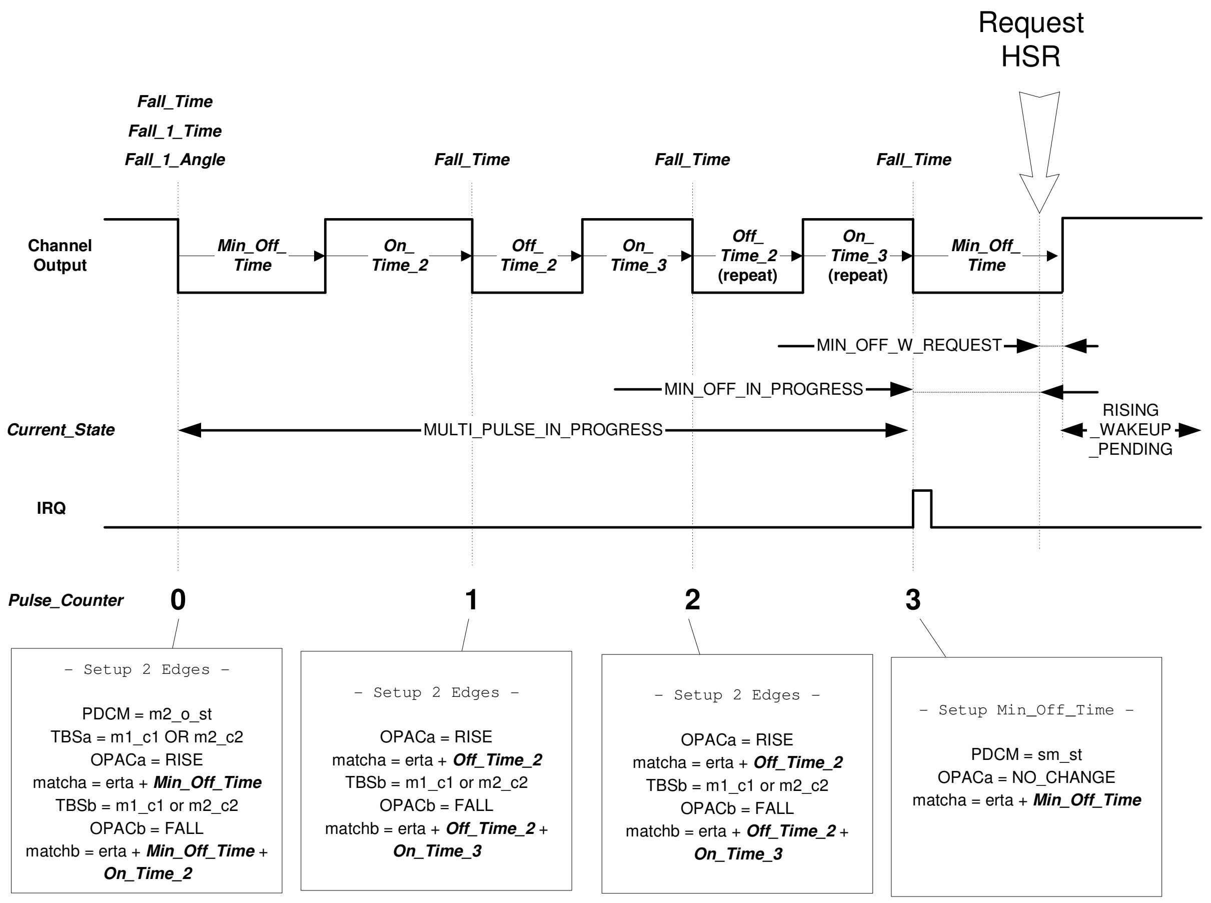


Figure 14: Multi-Pulse Timing (*Extra_Pulses* = 3)

Subsequent pulses are set up on the next falling edge, such that the signal is low for ***Off_Time_2*** and on for ***On_Time_3***.

Once the number of pulses requested as specified by ***Extra_Pulses*** have been issued, one more final match is set up for ***Min_Off_Time*** in the future and ***Current_State*** is transitioned to MIN_OFF_IN_PROGRESS.

As with single pulse mode, once the MIN_OFF_IN_PROGRESS match occurs ***Current_State*** transitions to WAITING_FOR_REQUEST and becomes idle until a **Request HSR** is issued.

As discussed in section 3.5, the next pulse (or pulse stream) can be requested anytime after the final falling edge. If a **Request HSR** is received after the final falling edge while the signal is held low for ***Min_Off_Time***, it is queued by transitioning ***Current_State*** to MIN_OFF_W_REQUEST. Once ***Min_Off_Time*** has been met the next pulse is instigated.

5. Miscellaneous

5.1 Time Mode Limitations

If the CPU is providing ***Critical_Event*** as a time (***MPTACe_In_Angle_Mode*** = 0) any desired future edge time must be between *TCRx* and *(TCRx + 0x7FFFFF)*; otherwise, the edge time will be interpreted as being in the past by the eTPU, and the desired action will occur immediately. If the edge is beyond this range, set the flag ***Req_Out_Of_Range*** when the **Request HSR** is set and the eTPU waits to set up the pulse until an **Update HSR** is received.

If an **Update HSR** or a **Request HSR** is issued after the edge has risen, and ***Req_Out_Of_Range*** flag is set, then the signal is held high until ***Max_Duration*** is met.

5.2 Angle Mode Limitations

When using End-Duration and angle mode, if the requested falling edge determined by ***Critical_Event – Comp_Time*** (converted to angle) occurs at the same time as ***Rise_Time*** + ***Min_Duration***, then **the falling edge is delayed until the thread completes processing and is re-scheduled. This delay is between 30 – 50 usec.**

5.3 Request in the Past

If a request for an end-duration pulse, either time or angle mode, is issued where both the rising and falling edges are in the past, a pulse is still generated whose high-time is equal to ***Min_Duration***.

5.4 Fuel Shut-Off for Direct-Injection Applications

If the MPTACe algorithm is used to generate EST pulses for a Direct-Injection application, it may be set up by the user to switch to a desired DIG fuel channel at the falling (spark) edge, and shut off any fuel pulse that may be in progress. (When ***Multi_Pulse_Mode*** is set, this action occurs at the falling edge of the first pulse in a series.) MPTACe may also manipulate certain flags that are used by the Direct-Injection fuel algorithm.

NOTE: This version of MPTACe switches to the specified DIG fuel channel, but does not perform any actions. The actions need to be defined once the Direct-Injection fuel algorithm is available for the eTPU.

To activate this feature, the user should write the number of the desired fuel channel to ***DIG_Chan_Num*** during initialization, and set the ***DIG_Shutoff_Enabled*** flag.

5.5 Global Exception

Under certain error conditions, the MPTACe primitive issues a global exception to the host and writes its channel number to the global variable ***Cause_Of_Exception***. Below are the conditions that cause MPTACe to generate a global exception:

1.) An HSR is issued to the MPTACe primitive that has not been defined.

2.) A link has been issued to the MPTACe primitive (links are not supported by MPTACe).

3.) During the processing of a match, ***Current_State*** is determined to contain an invalid state. The pulse is aborted along with saving the channel number and generating the global exception.

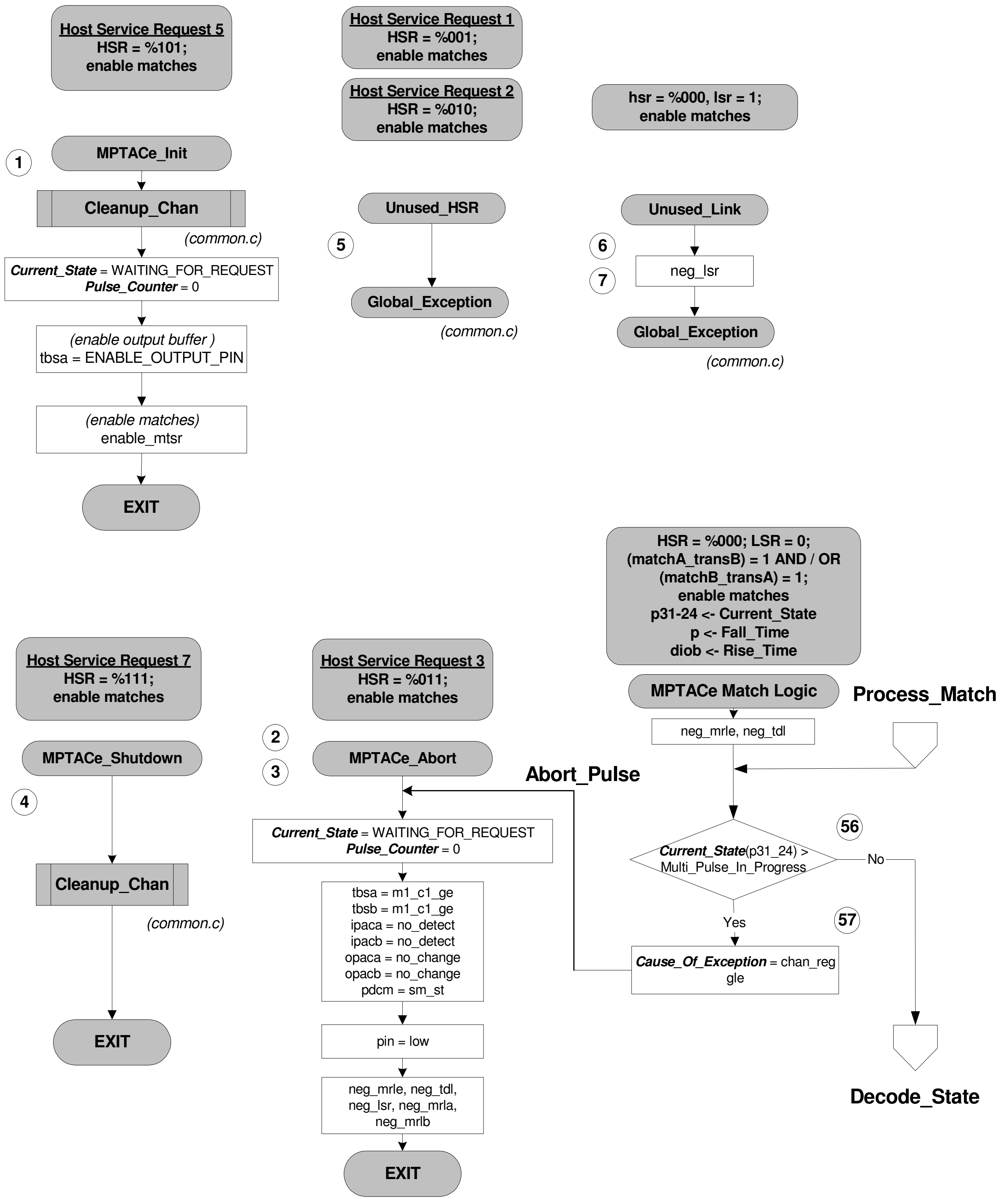
6. Primitive Timing

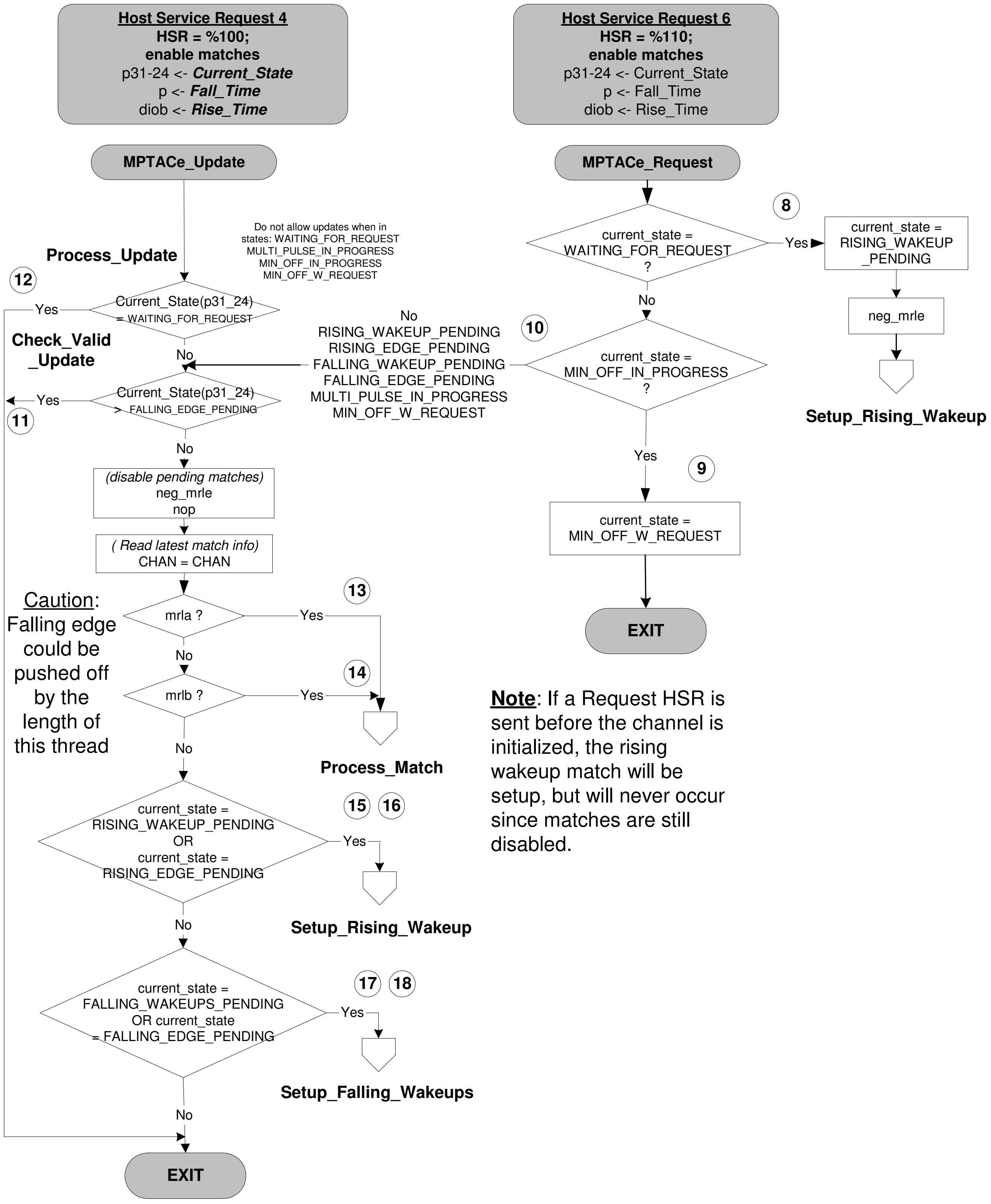
MPTACe Primitive Phase	Worse Case µ cycles	RAM Accesses (in Worse Case Path)
Initialize HSR	**11**	**2**
Abort_Request HSR	**4**	**2**
Shutdown HSR	**8**	**0**
Request HSR	**95** (Request during rising wake-up)	**8**
Update HSR	**92** (Update during rising wake-up)	**7**
Match Processing (angle mode)		
Rising Wake-up	**88**	**7**
Rising Edge	**57**	**7**
Falling Wake-ups	**48**	**6**
Falling Edge	**35**	**11**
Min_Period w/ Request	**89**	**7**

Size of Parameter RAM	= **10 words** (32 bits)
Size of Local RAM	= **0 words** (32 bits)
Size of Code	= **225 words** (32 bits)

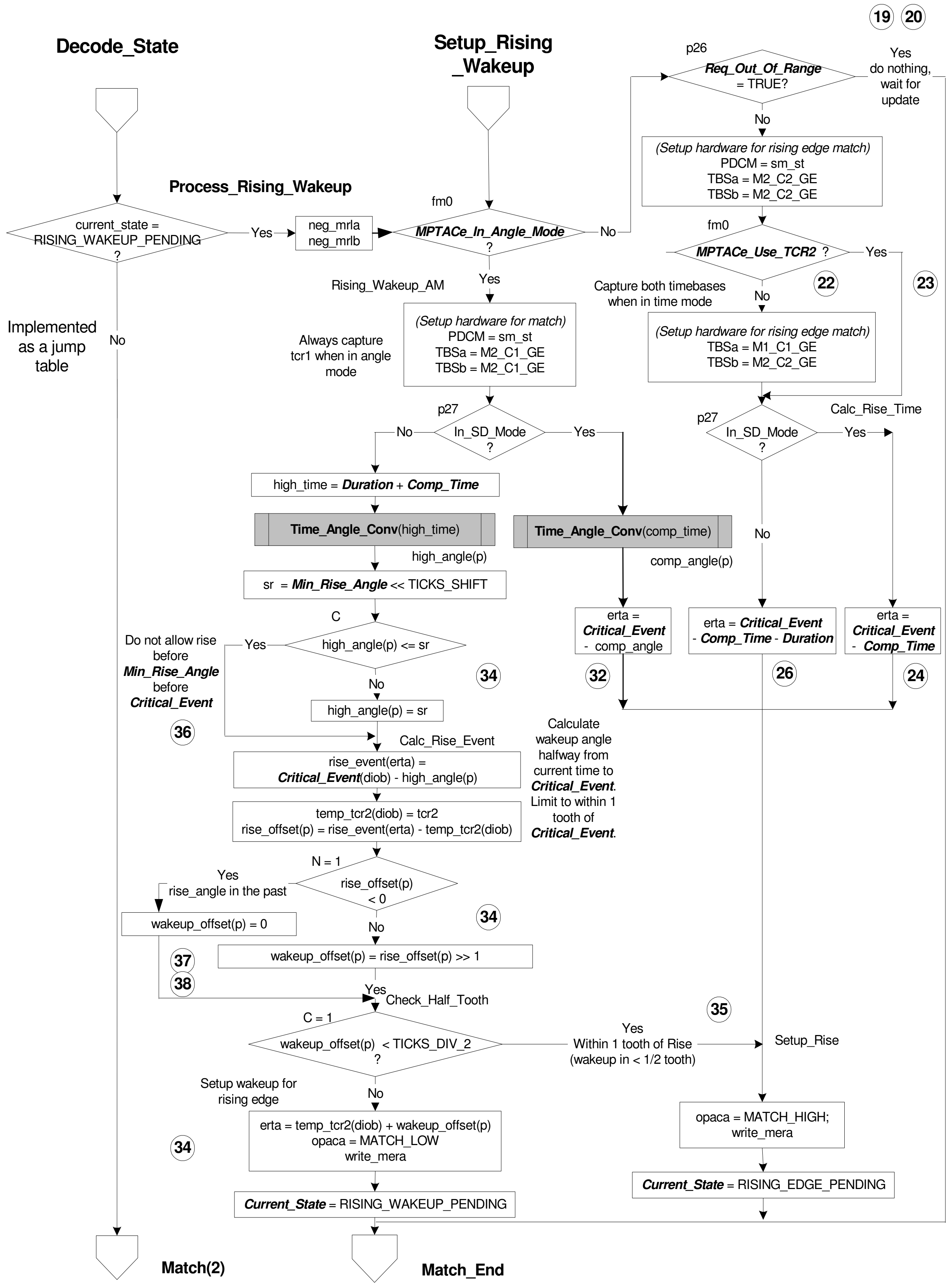
7. Flowcharts and Phase Diagram

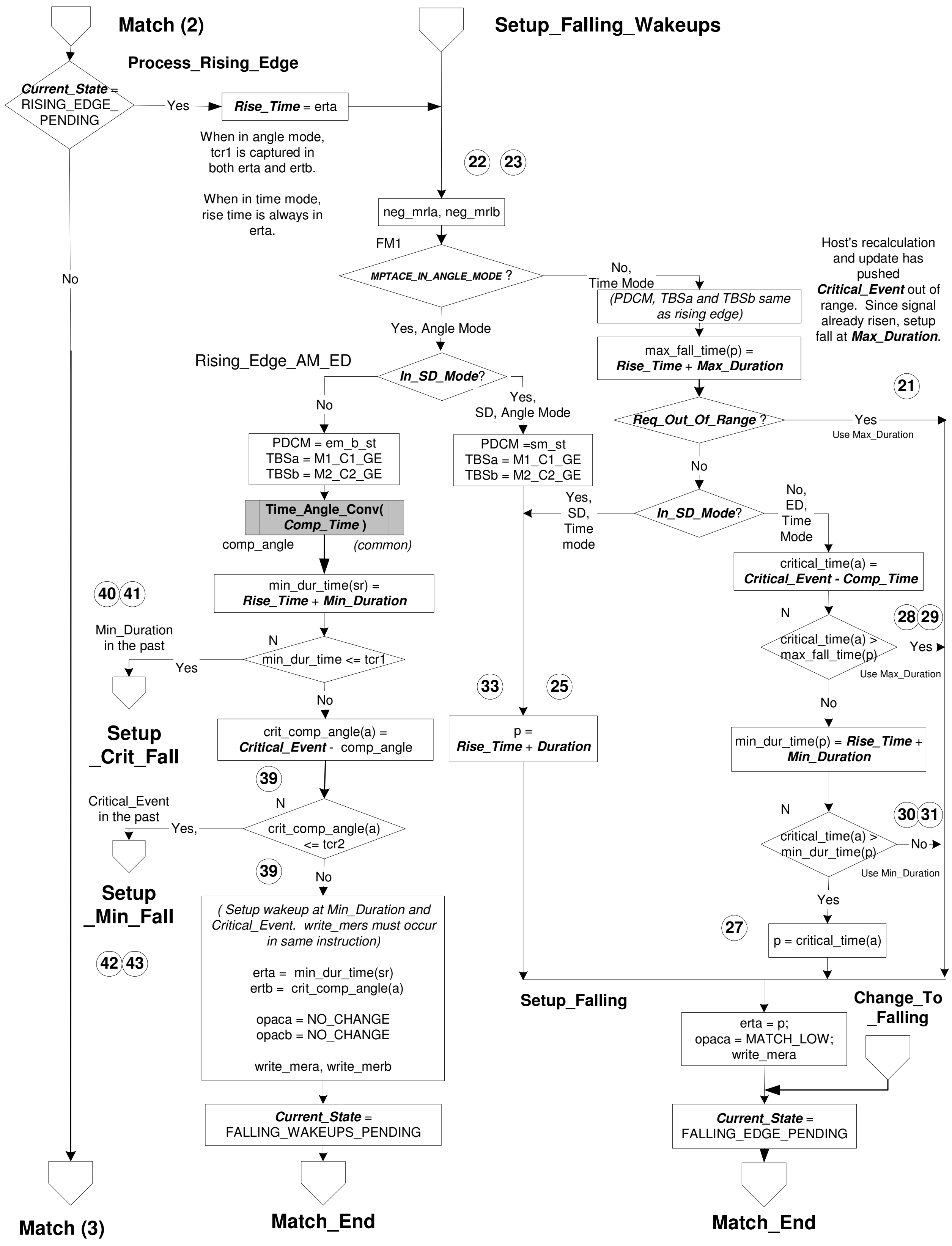
X Indicates the test case associated with the flow chart. Details on the test case are included in the software test plan document entitled "STP_5412.doc" available in CM Synergy in the kok_pt1 database, under ComplexIO/eTPU/Primitives/Spark/MPTACe/Verification/VerificationTestPlan.



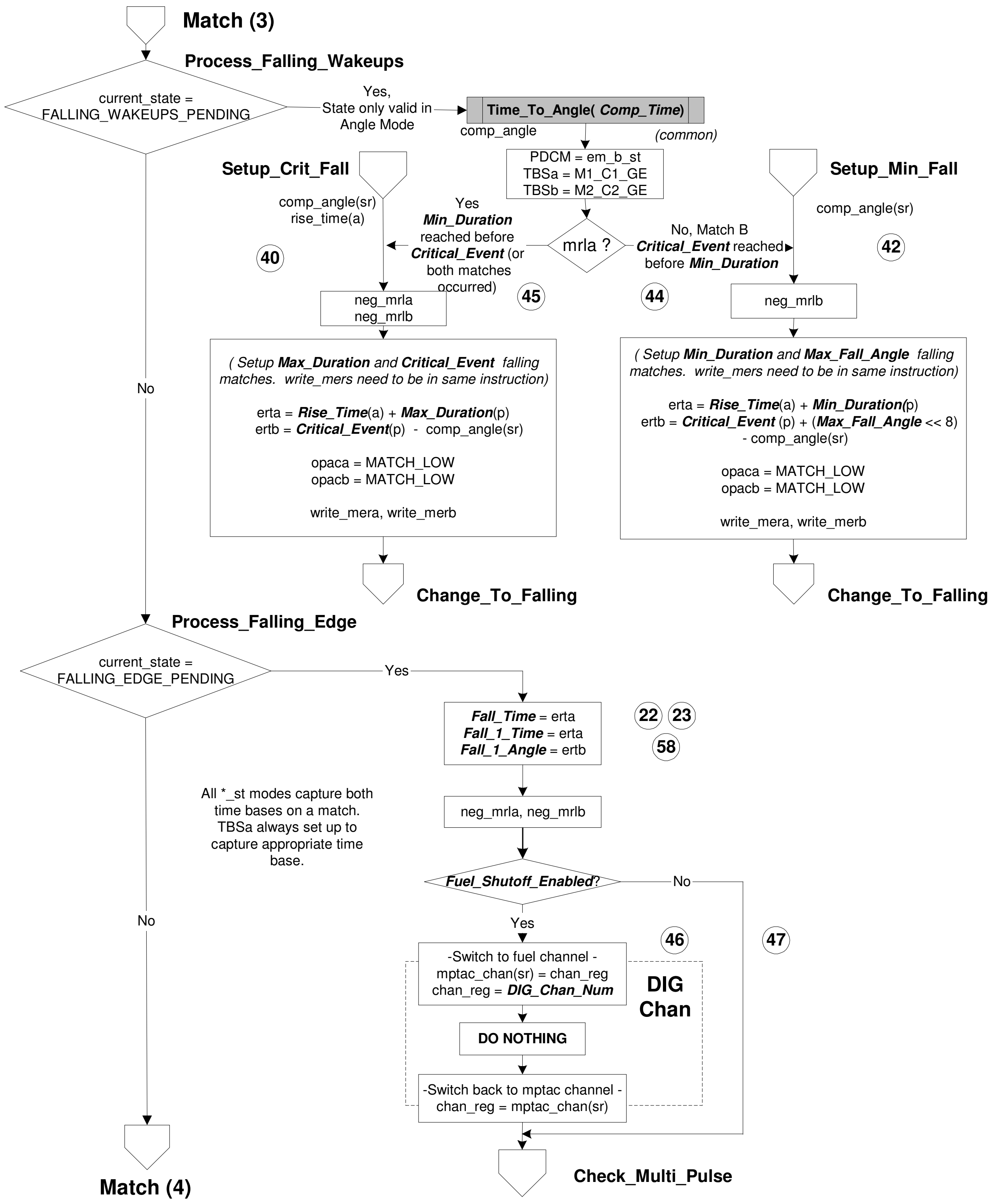

Host Service Request 4
HSR = %100;
enable matches
p31-24 <- Current_State
p <- Fall_Time
diob <- Rise_Time
MPTACe_Update
Process_Update
Do not allow updates when in states: WAITING_FOR_REQUEST MULTI_PULSE_IN_PROGRESS MIN_OFF_IN_PROGRESS MIN_OFF_W_REQUEST
12
Current_State(p31_24) = WAITING_FOR_REQUEST
Yes
No
Check_Valid _Update
Current_State(p31_24) > FALLING_EDGE_PENDING
Yes
11
No
(disable pending matches) neg_mrle nop
(Read latest match info) CHAN = CHAN
Caution: Falling edge could be pushed off by the length of this thread
mrla ?
Yes
13
No
mrlb ?
Yes
14
No
Process_Match
current_state = RISING_WAKEUP_PENDING OR current_state = RISING_EDGE_PENDING
Yes
15
16
No
Setup_Rising_Wakeup
current_state = FALLING_WAKEUPS_PENDING OR current_state = FALLING_EDGE_PENDING
Yes
17
18
Setup_Falling_Wakeups
No
EXIT
Host Service Request 6
HSR = %110;
enable matches
p31-24 <- Current_State
p <- Fall_Time
diob <- Rise_Time
MPTACe_Request
current_state = WAITING_FOR_REQUEST ?
Yes
8
current_state = RISING_WAKEUP _PENDING
neg_mrle
Setup_Rising_Wakeup
No
10
No RISING_WAKEUP_PENDING RISING_EDGE_PENDING FALLING_WAKEUP_PENDING FALLING_EDGE_PENDING MULTI_PULSE_IN_PROGRESS MIN_OFF_W_REQUEST
current_state = MIN_OFF_IN_PROGRESS ?
Yes
9
current_state = MIN_OFF_W_REQUEST
EXIT

Note: If a Request HSR is sent before the channel is initialized, the rising wakeup match will be setup, but will never occur since matches are still disabled.

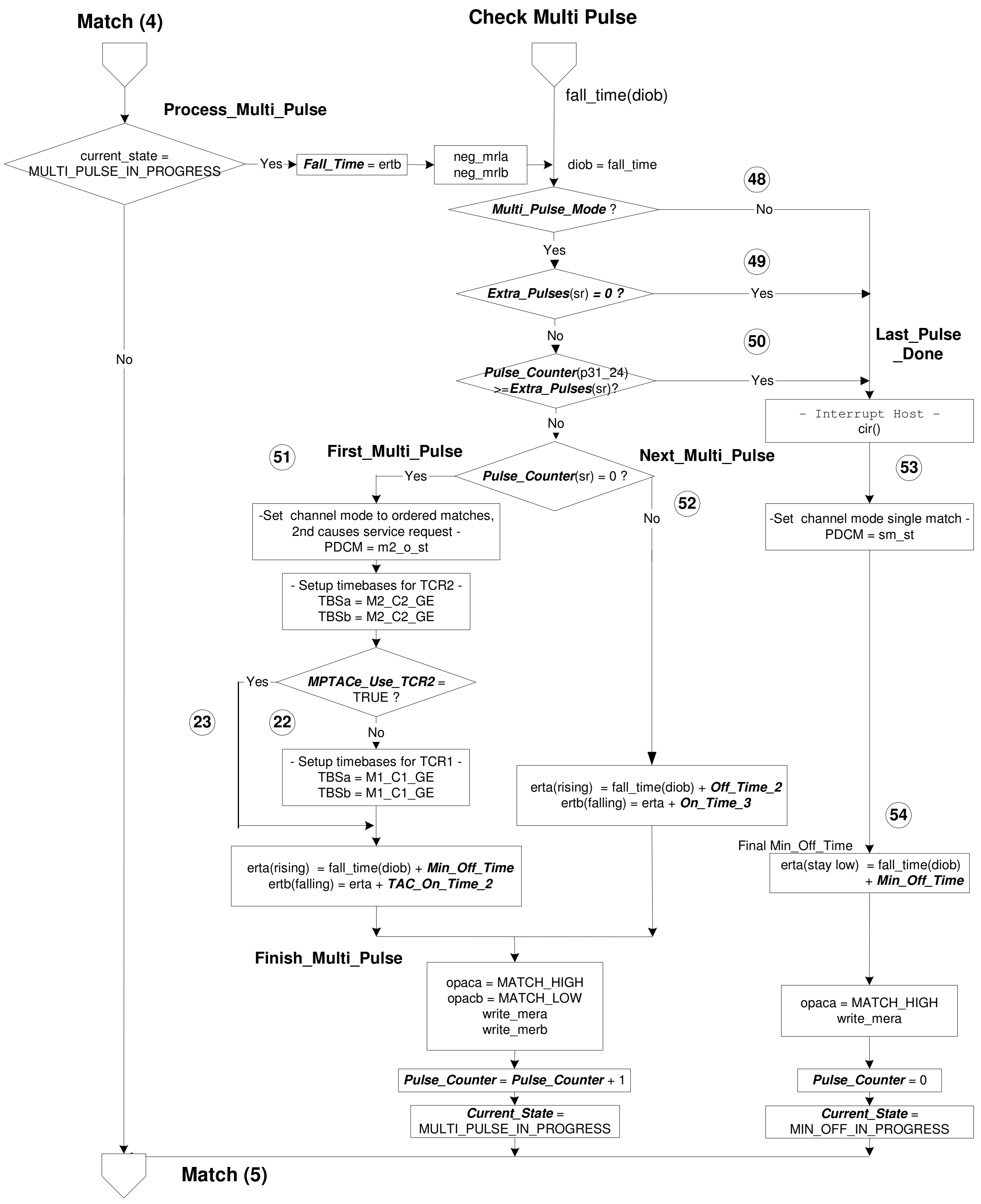

19
20
Decode_State
Setup_Rising _Wakeup
p26
Req_Out_Of_Range = TRUE?
Yes do nothing, wait for update
No
(Setup hardware for rising edge match) PDCM = sm_st TBSa = M2_C2_GE TBSb = M2_C2_GE
Process_Rising_Wakeup
fm0
current_state = RISING_WAKEUP_PENDING ?
Yes
neg_mrla neg_mrlb
MPTACe_In_Angle_Mode ?
No
fm0
MPTACe_Use_TCR2 ?
Yes
22
23
Rising_Wakeup_AM
Yes
Capture both timebases when in time mode
No
Implemented as a jump table
No
Always capture tcr1 when in angle mode
(Setup hardware for match) PDCM = sm_st TBSa = M2_C1_GE TBSb = M2_C1_GE
(Setup hardware for rising edge match) TBSa = M1_C1_GE TBSb = M2_C2_GE
p27
In_SD_Mode ?
No
Yes
p27
In_SD_Mode ?
Calc_Rise_Time
Yes
high_time = Duration + Comp_Time
Time_Angle_Conv(high_time)
Time_Angle_Conv(comp_time)
No
high_angle(p)
comp_angle(p)
sr = Min_Rise_Angle << TICKS_SHIFT
C
erta = Critical_Event - comp_angle
erta = Critical_Event - Comp_Time - Duration
erta = Critical_Event - Comp_Time
Do not allow rise before Min_Rise_Angle before Critical_Event
Yes
high_angle(p) <= sr
No
34
32
26
24
high_angle(p) = sr
36
Calc_Rise_Event
Calculate wakeup angle halfway from current time to Critical_Event. Limit to within 1 tooth of Critical_Event.
rise_event(erta) = Critical_Event(diob) - high_angle(p)
temp_tcr2(diob) = tcr2 rise_offset(p) = rise_event(erta) - temp_tcr2(diob)
N = 1
Yes rise_angle in the past
rise_offset(p) < 0
wakeup_offset(p) = 0
34
No
37
38
wakeup_offset(p) = rise_offset(p) >> 1
Yes
Check_Half_Tooth
35
C = 1
wakeup_offset(p) < TICKS_DIV_2 ?
Yes Within 1 tooth of Rise (wakeup in < 1/2 tooth)
Setup_Rise
Setup wakeup for rising edge
No
erta = temp_tcr2(diob) + wakeup_offset(p) opaca = MATCH_LOW write_mera
opaca = MATCH_HIGH; write_mera
34
Current_State = RISING_EDGE_PENDING
Current_State = RISING_WAKEUP_PENDING
Match(2)
Match_End



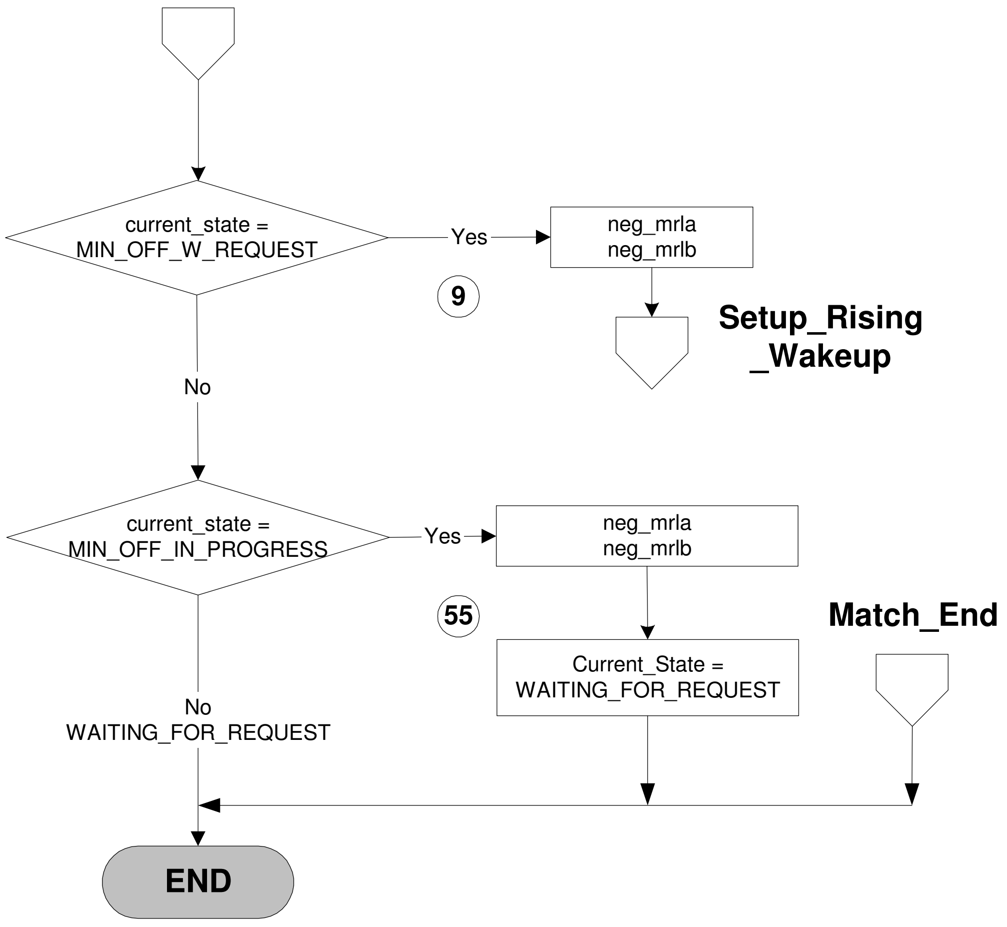
Match (2)
Process_Rising_Edge
Current_State = RISING_EDGE_PENDING
Yes
Rise_Time = erta
When in angle mode, tcr1 is captured in both erta and ertb.
When in time mode, rise time is always in erta.
No
Match (3)
Setup_Falling_Wakeups
22
23
neg_mrla, neg_mrlb
FM1
MPTACE_IN_ANGLE_MODE ?
No, Time Mode
(PDCM, TBSa and TBSb same as rising edge)
max_fall_time(p) = Rise_Time + Max_Duration
Host's recalculation and update has pushed Critical_Event out of range. Since signal already risen, setup fall at Max_Duration.
21
Req_Out_Of_Range ?
Yes
Use Max_Duration
No
In_SD_Mode?
Yes, SD, Time mode
No, ED, Time Mode
critical_time(a) = Critical_Event - Comp_Time
N
28
29
critical_time(a) > max_fall_time(p)
Yes
Use Max_Duration
No
min_dur_time(p) = Rise_Time + Min_Duration
N
30
31
critical_time(a) > min_dur_time(p)
No
Use Min_Duration
Yes
27
p = critical_time(a)
Yes, Angle Mode
Rising_Edge_AM_ED
In_SD_Mode?
No
Yes, SD, Angle Mode
PDCM = em_b_st
TBSa = M1_C1_GE
TBSb = M2_C2_GE
PDCM =sm_st
TBSa = M1_C1_GE
TBSb = M2_C2_GE
Time_Angle_Conv(Comp_Time)
comp_angle
(common)
40
41
min_dur_time(sr) = Rise_Time + Min_Duration
Min_Duration in the past
Yes
N
min_dur_time <= tcr1
No
Setup _Crit_Fall
33
25
p = Rise_Time + Duration
crit_comp_angle(a) = Critical_Event - comp_angle
39
Critical_Event in the past
Yes,
N
crit_comp_angle(a) <= tcr2
39
No
Setup _Min_Fall
42
43
(Setup wakeup at Min_Duration and Critical_Event. write_mers must occur in same instruction)
erta = min_dur_time(sr)
ertb = crit_comp_angle(a)
opaca = NO_CHANGE
opacb = NO_CHANGE
write_mera, write_merb
Current_State = FALLING_WAKEUPS_PENDING
Match_End
Setup_Falling
erta = p;
opaca = MATCH_LOW;
write_mera
Change_To _Falling
Current_State = FALLING_EDGE_PENDING
Match_End



Match (3)
Process_Falling_Wakeups
current_state = FALLING_WAKEUPS_PENDING
Yes, State only valid in Angle Mode
Time_To_Angle(Comp_Time)
comp_angle
(common)
PDCM = em_b_st
TBSa = M1_C1_GE
TBSb = M2_C2_GE
mrla ?
Yes Min_Duration reached before Critical_Event (or both matches occurred)
No, Match B Critical_Event reached before Min_Duration
Setup_Crit_Fall
comp_angle(sr)
rise_time(a)
40
45
44
42
Setup_Min_Fall
comp_angle(sr)
neg_mrla
neg_mrlb
neg_mrlb
(Setup Max_Duration and Critical_Event falling matches. write_mers need to be in same instruction)
erta = Rise_Time(a) + Max_Duration(p)
ertb = Critical_Event(p) - comp_angle(sr)
opaca = MATCH_LOW
opacb = MATCH_LOW
write_mera, write_merb
(Setup Min_Duration and Max_Fall_Angle falling matches. write_mers need to be in same instruction)
erta = Rise_Time(a) + Min_Duration(p)
ertb = Critical_Event (p) + (Max_Fall_Angle << 8) - comp_angle(sr)
opaca = MATCH_LOW
opacb = MATCH_LOW
write_mera, write_merb
Change_To_Falling
Change_To_Falling
No
Process_Falling_Edge
current_state = FALLING_EDGE_PENDING
Yes
Fall_Time = erta
Fall_1_Time = erta
Fall_1_Angle = ertb
22
23
58
All *_st modes capture both time bases on a match. TBSa always set up to capture appropriate time base.
neg_mrla, neg_mrlb
Fuel_Shutoff_Enabled?
No
Yes
46
47
No
-Switch to fuel channel -
mptac_chan(sr) = chan_reg
chan_reg = DIG_Chan_Num
DIG Chan
DO NOTHING
-Switch back to mptac channel -
chan_reg = mptac_chan(sr)
Match (4)
Check_Multi_Pulse


Match (4)
Check Multi Pulse
fall_time(diob)
Process_Multi_Pulse
current_state = MULTI_PULSE_IN_PROGRESS
Yes
Fall_Time = ertb
neg_mrla
neg_mrlb
diob = fall_time
48
Multi_Pulse_Mode ?
No
Yes
49
Extra_Pulses(sr) = 0 ?
Yes
No
50
Last_Pulse _Done
No
Pulse_Counter(p31_24) >=Extra_Pulses(sr)?
Yes
- Interrupt Host -
cir()
No
51
First_Multi_Pulse
Next_Multi_Pulse
Yes
Pulse_Counter(sr) = 0 ?
53
52
No
-Set channel mode to ordered matches, 2nd causes service request -
PDCM = m2_o_st
-Set channel mode single match -
PDCM = sm_st
- Setup timebases for TCR2 -
TBSa = M2_C2_GE
TBSb = M2_C2_GE
Yes
MPTACe_Use_TCR2 = TRUE ?
23
22
No
- Setup timebases for TCR1 -
TBSa = M1_C1_GE
TBSb = M1_C1_GE
erta(rising) = fall_time(diob) + Off_Time_2
ertb(falling) = erta + On_Time_3
54
Final Min_Off_Time
erta(stay low) = fall_time(diob) + Min_Off_Time
erta(rising) = fall_time(diob) + Min_Off_Time
ertb(falling) = erta + TAC_On_Time_2
Finish_Multi_Pulse
opaca = MATCH_HIGH
opacb = MATCH_LOW
write_mera
write_merb
opaca = MATCH_HIGH
write_mera
Pulse_Counter = Pulse_Counter + 1
Pulse_Counter = 0
Current_State = MULTI_PULSE_IN_PROGRESS
Current_State = MIN_OFF_IN_PROGRESS
Match (5)

Match (5)



current_state = MIN_OFF_W_REQUEST
Yes
neg_mrla
neg_mrlb
9
Setup_Rising _Wakeup
No
current_state = MIN_OFF_IN_PROGRESS
Yes
neg_mrla
neg_mrlb
55
Match_End
Current_State = WAITING_FOR_REQUEST
No
WAITING_FOR_REQUEST
END

8. Revision Log

Each microcode document is assigned a revision number. The numbers are assigned according to the following scheme (used for all documents after 10/16/92):

Rev ***x.y***

x identifies the microcode, where

1 - represents the original release of microcode
2 - represents the first release of changed microcode
3 - represents the second change to microcode
etc.

y identifies the document, where

0 - represents the original release of documentation for this microcode
1 - represents the first document change for the same microcode
2 - represents the second change to the document for the same microcode
etc.

8.1 Revision History

Revision	Date	Record	Author
1.0	02/25/05	Initial release.	Mary Hedges
2.0 (SCR 4110)	11/11/05	Passed FM1 definitions to host.	Warren Donley
3.0 (SCR 4605)	02/24/06	Replaced **Common_Unused_Entry** with **Global_Exception**.	Warren Donley
4.0 (SCR 4907)	08/23/06	Stored ***Fall_1_Time*** and ***Fall_1_Angle*** on falling edge of initial output pulse in a series.	Warren Donley
5.0 (SCR 125)	05/29/09	- Fixed bug by loading rise time into "erta" during an Update HSR. - Fixed potential problem by not using "a" register after calling **Time_Angle_Conv**.	Warren Donley