

*Application Note*

AN2530/D  
Rev. 0, 5/2003

Standard Space Vector  
Modulation with Dead-Time  
Correction TPU Function Set  
(svmStdDt)

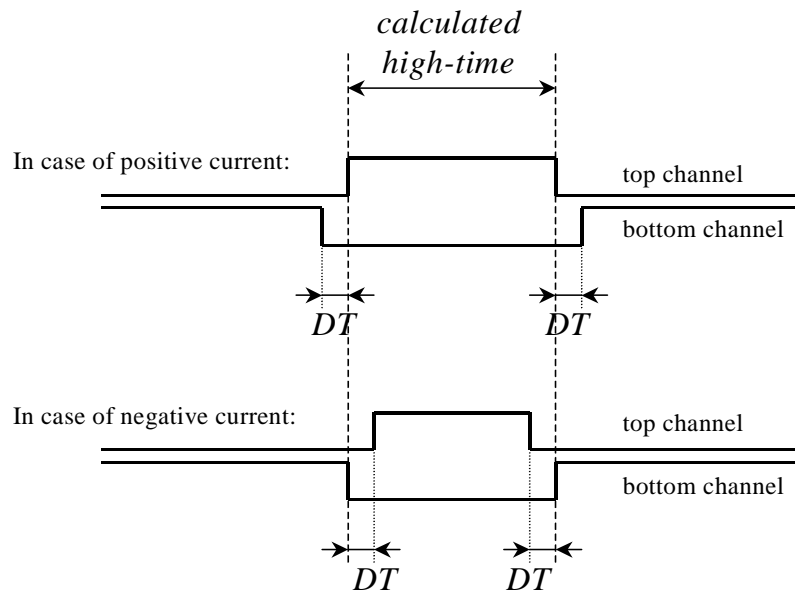
By Milan Brejl, Ph.D.

Freescale Semiconductor, Inc.

**Functional Overview**

The Standard Space Vector Modulation with Dead-Time Correction TPU function set (svmStdDt) extends the functionality of the Standard Space Vector Modulation TPU function set (svmStd) by the dead-time correction technique. Apart from this the functionality is the same in all aspects.

The dead-time correction technique requires knowledge of the instantaneous direction of phase currents. In the case of positive phase current the top channel high-time is equal to the calculated high-time and the bottom channel has to control the dead-time. In the case of negative phase current the bottom channel low-time is equal to the calculated high-time and the top channel has to control the dead-time. See [Figure 1](#).

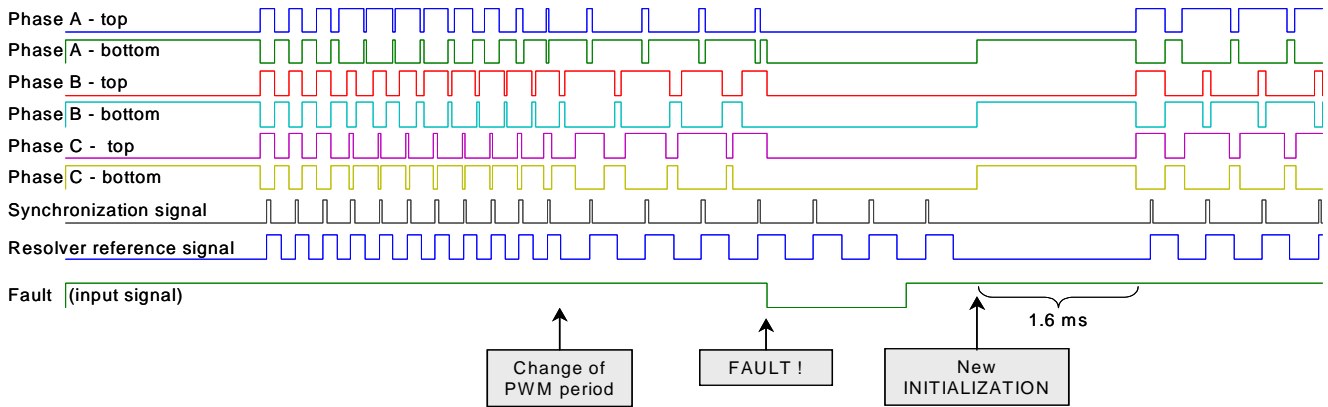


**Figure 1. Dead-Time Correction Technique**

The function set consists of 5 TPU functions:

- Standard Space Vector Modulation with Dead-Time Correction – Top (svmStdDt\_top)
- Standard Space Vector Modulation with Dead-Time Correction – Bottom (svmStdDt\_bottom)
- Synchronization signal for Standard Space Vector Modulation with Dead-Time Correction (svmStdDt\_sync)
- Resolver Reference Signal for Standard Space Vector Modulation with Dead-Time Correction (svmStdDt\_res)
- Fault Input for Standard Space Vector Modulation with Dead-Time Correction (svmStdDt\_fault)

The svmStdDt\_top and svmStdDt\_bottom TPU functions work together to generate a 6-channel 3-phase center-aligned PWM signal with dead-time between the top and bottom channels. The Synchronization Signal for svmStdDt function can be used to generate one or more adjustable signals for a wide range of uses, that are synchronized to the PWM, and track changes in the PWM period. The Resolver Reference Signal for svmStdDt function can be used to generate one or more 50% duty-cycle adjustable signals that are also synchronized to the PWM. The Fault Input for svmStdDt function is a TPU input function that sets all PWM outputs low when the input signal goes low. See [Figure 2](#).



**Figure 2. Signals generated by svmStdDt TPU function set**

**Function Set Configuration**

None of the TPU functions in the Standard Space Vector Modulation with Dead-Time Correction TPU function set can be used separately. The svmStdDt\_top and svmStdDt\_bottom functions have to be used together. The svmStdDt\_top is used on 3 channels, the svmStdDt\_bottom on a further 3 channels, and within each phase, the function svmStdDt\_top has to be assigned on a lower TPU channel than the function svmStdDt\_bottom. This is illustrated in the examples in [Table 2](#) and [Table 3](#). One or more channels running Synchronization Signal for svmStdDt as well as Resolver Reference Signals for svmStdDt functions can be added to the svmStdDt\_top and svmStdDt\_bottom functions. They can run with different settings on each channel. The function Fault Input for svmStdDt can also be added to the svmStdDt\_top and svmStdDt\_bottom functions. It is recommended to use it on channel 15, and to set the hardware option that disables all TPU output pins when the channel 15 input signal is low (DTPU bit = 1). This ensures that the hardware reacts quickly to a pin fault state. Note that it is not only the PWM channels, but all TPU output channels, including the synchronization signals, that are disabled in this configuration.

[Table 1](#) shows the configuration options and restrictions.

**Table 1. svmStdDt TPU function set configuration options and restrictions**

TPU function	Optional/Mandatory	How many channels	Assignable channels
svmStdDt_top	mandatory	3	any 3 channels, within each phase a lower TPU channel than the same phase svmStdDt_bottom
svmStdDt_bottom	mandatory	3	any 3 channels, within each phase a higher TPU channel than the same phase svmStdDt_top
svmStdDt_sync	optional	1 or more	any channels
svmStdDt_res	optional	1 or more	any channels
svmStdDt_fault	optional	1	any, recommended is 15 and DTPU bit set

[Table 2](#) and [Table 3](#) show two examples of configuration.

**Table 2. Example of configuration**

Channel	TPU function	Priority
0	svmStdDt_top	high

**Table 2. Example of configuration**

Channel	TPU function	Priority
1	svmStdDt_bottom	high
2	svmStdDt_top	high
3	svmStdDt_bottom	high
4	svmStdDt_top	high
5	svmStdDt_bottom	high
10	svmStdDt_sync	low
15	svmStdDt_fault	high

**Table 3. Example of configuration**

Channel	TPU function	Priority
0	svmStdDt_top	high
1	svmStdDt_top	high
2	svmStdDt_top	high
3	svmStdDt_bottom	high
4	svmStdDt_bottom	high
5	svmStdDt_bottom	high
10	svmStdDt_sync	low
11	svmStdDt_res	low
15	svmStdDt_fault	high

**Table 4** shows the TPU function code sizes.

**Table 4. TPU function code sizes**

TPU function	Code size
svmStdDt_top	26 $\mu$ instructions + 8 entries = 34 long words
svmStdDt_bottom	196 $\mu$ instructions + 8 entries = 204 long words
svmStdDt_sync	26 $\mu$ instructions + 8 entries = 34 long words
svmStdDt_res	38 $\mu$ instructions + 8 entries = 46 long words
svmStdDt_fault	9 $\mu$ instructions + 8 entries = 17 long words

**Configuration Order**

The CPU configures the TPU as follows.

1. Disables the channels by clearing the two channel priority bits on each channel used (not necessary after reset).
2. Selects the channel functions on all used channels by writing the function numbers to the channel function select bits.
3. Initializes function parameters. The parameters *T*, *prescaler*, *DT*, *MPW*, *SQRT3* and *sync\_presc\_addr* must be set before initialization. If an

svmStdDt\_sync channel or an svmStdDt\_res channel is used, then also its parameters must be set before initialization.

4. Issues an HSR (Host Service Request) type %10 to one of the svmStdDt\_bottom channels to initialize all PWM channels. Issues an HSR type %10 to the svmStdDt\_sync channels, svmStdDt\_res channels and svmStdDt\_fault channel, if used.
5. Enables servicing by assigning high, middle or low priority to the channel priority bits. All PWM channels must be assigned the same priority to ensure correct operation. The CPU must ensure that the svmStdDt\_sync or svmStdDt\_res channels are initialized after the initialization of PWM channels:
  - assign a priority to the PWM channels to enable their initialization
  - if a Synchronization Signal or a Resolver Reference Signal channel is used, wait until the HSR bits are cleared to indicate that initialization of the PWM channels has completed and
  - assign a priority to the svmStdDt\_sync or svmStdDt\_res channels to enable their initialization

**NOTE:** A CPU routine that configures the TPU can be generated automatically using the MPC500\_Quick\_Start Graphical Configuration Tool.

---

## Detailed Function Description

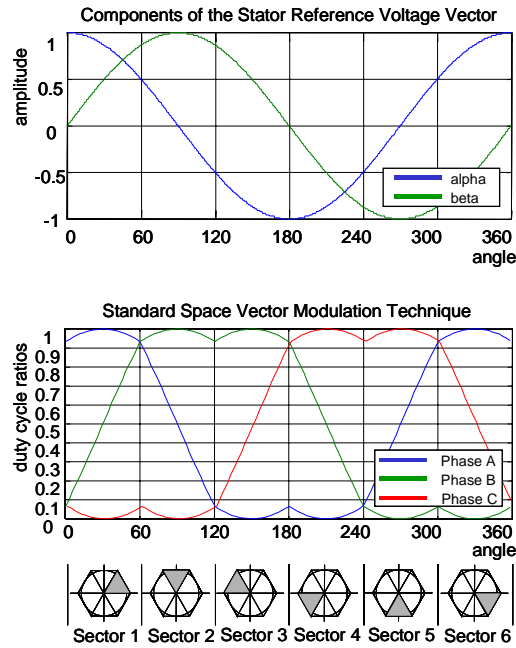
### Standard Space Vector Modulation with Dead-Time Correction – Top (svmStdDt\_top) and Standard Space Vector Modulation with Dead-Time Correction – Bottom (svmStdDt\_bottom)

The svmStdDt\_top and svmStdDt\_bottom TPU functions work together to generate a 6-channel, 3-phase PWM signal, with dead-time between the top and bottom channels. In order to charge the bootstrap transistors, the PWM signals start to run 1.6ms after their initialization (at 20MHz TCR1 clock). The functions generate signals corresponding to Reference Voltage Vector Amplitude of 0 (50% duty-cycle) until the first reloaded values are processed.

The CPU controls the PWM output by setting the TPU parameters. The Stator Reference Voltage Vector components  $u_a$  and  $u_b$  have to be adjusted during run time. The PWM period  $T$  and the *prescaler* – the number of PWM periods per reload of new values – are also read at each reload, so these parameters can be changed during run time. Conversely, dead-time ( $DT$ ) and minimum pulse width ( $MPW$ ) are not supposed to be changed during run time. The phase currents *currentA*, *currentB* and *currentC* are read by the TPU asynchronously to the PWM parameters reload. They are read in the last part of edge-time calculation to reflect the latest state of the phase currents. The CPU notifies the TPU that the new reload values are prepared by setting the LD\_OK parameter. The TPU notifies the CPU that the reload values have been read and new values can be written by clearing the LD\_OK parameter.

The TPU writes the parameter Sector, which indicates the current Stator Reference Voltage Vector position in sector 1 to 6.

The following figures show the input Stator Reference Voltage Vector components  $u_{\hat{a}}$  and  $u_{\hat{\beta}}$ , corresponding sectors and output PWM signal duty cycle ratios:



**Figure 3. Standard Space Vector Modulation with Dead-Time Correction Technique**

The following equations describe how the Space Vector Modulation PWM signal high-times  $ht_A$ ,  $ht_B$ ,  $ht_C$  and transition times  $t_{low-high}$  and  $t_{high-low}$  of each channel are calculated:

$$U_{\beta} = T \cdot u_{\beta}$$

$$U_{\alpha} = T \cdot u_{\alpha}$$

$$X = U_{\beta}$$

$$Y = \frac{U_{\beta} + U_{\alpha} \sqrt{3}}{2}$$

$$Z = \frac{U_{\beta} - U_{\alpha} \sqrt{3}}{2}$$

	Y < 0			Y >= 0		
	Z < 0	Z >= 0		Z < 0		Z >= 0
		X <= 0	X > 0	X <= 0	X > 0	
<b>Sector:</b>	<b>V.</b>	<b>IV.</b>	<b>III.</b>	<b>VI.</b>	<b>I.</b>	<b>II.</b>

**Phase A:**

Positive current  
top channel

$$t_{\text{low-high}} = \text{center\_time} - \frac{ht_A}{2}$$

$$t_{\text{high-low}} = \text{center\_time} + \frac{ht_A}{2}$$

bottom channel

$$t_{\text{high-low}} = \text{center\_time} - \frac{ht_A}{2} - DT$$

$$t_{\text{low-high}} = \text{center\_time} + \frac{ht_A}{2} + DT$$

Negative current  
top channel

$$t_{\text{low-high}} = \text{center\_time} - \frac{ht_A}{2} + DT$$

$$t_{\text{high-low}} = \text{center\_time} + \frac{ht_A}{2} - DT$$





bottom channel

$$t_{\text{high-low}} = \text{center\_time} - \frac{ht_A}{2}$$






$$t_{\text{low-high}} = \text{center\_time} + \frac{ht_A}{2}$$

Phase B and Phase C similarly with  $ht_B$  and  $ht_C$  substituted to  $ht_A$ .

*Host Interface*

	Written By CPU		Written by both CPU and TPU
	Written By TPU		Not Used

**Table 5. svmStdDt\_top Control Bits**

Name	Options
<div style="display: flex; justify-content: space-around; width: 100px;"> <span>3</span><span>2</span><span>1</span><span>0</span> </div>  Channel Function Select	svmStdDt_top function number (Assigned during assembly the DPTRAM code from library TPU functions)
<div style="display: flex; justify-content: space-around; width: 100px;"> <span>1</span><span>0</span> </div>  Channel Priority	00 – Channel Disabled 01 – Low Priority 10 – Middle Priority 11 – High Priority
<div style="display: flex; justify-content: space-around; width: 100px;"> <span>1</span><span>0</span> </div>  Host Service Bits (HSR)	00 – No Host Service Request 01 – Not used 10 – Not used 11 – Not used
<div style="display: flex; justify-content: space-around; width: 100px;"> <span>1</span><span>0</span> </div>  Host Sequence Bits (HSQ)	xx – Not used
<div style="display: flex; justify-content: space-around; width: 100px;"> <span>0</span> </div>  Channel Interrupt Enable	x – Not used



**Table 5. svmStdDt\_top Control Bits**

Name	Options
<div style="text-align: center;">0</div> Channel Interrupt Status	x – Not used

**Table 6. svmStdDt\_bottom Control Bits**

Name	Options
<div style="text-align: center;">3 2 1 0</div> Channel Function Select	svmStdDt_bottom function number (Assigned during assembly the DPTRAM code from library TPU functions)
<div style="text-align: center;">1 0</div> Channel Priority	00 – Channel Disabled 01 – Low Priority 10 – Middle Priority 11 – High Priority
<div style="text-align: center;">1 0</div> Host Service Bits (HSR)	00 – No Host Service Request 01 – Not used 10 – Initialization 11 – Stop
<div style="text-align: center;">1 0</div> Host Sequence Bits (HSQ)	xx – Not used
<div style="text-align: center;">0</div> Channel Interrupt Enable	0 – Channel Interrupt Disabled 1 – Channel Interrupt Enabled
<div style="text-align: center;">0</div> Channel Interrupt Status	0 – Interrupt Not Asserted 1 – Interrupt Asserted

TPU function *svmStdDt\_bottom* generates an interrupt when the current values of *Ualfa*, *Ubeta*, *T* and *prescaler* have been read by the TPU and indicates to the CPU that it can write new variables. The CPU program can either wait for this interrupt to occur, or poll the *LD\_OK* bit to check it has cleared. The interrupt is generated at each reload by one of the bottom channels. The top channels do not generate any interrupts.

**Table 7. svmStdDt\_top and svmStdDt\_bottom Parameter RAM**

Channel	Parameter	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Phase A top channel	0	htA																
	1	HLtime_AT																
	2	bottom_chan_A																
	3	currentA																
	4	LD_OK																
	5	Sector																
	6																	
	7	fault_pinstat																
Phase A bottom channel	0	LHtime_AB																
	1	HLtime_AB																
	2	UA																
	3	UB																
	4	Ualfa																
	5	Ubeta																
	6	UA3																
	7																	
Phase B top channel	0	htB																
	1	HLtime_BT																
	2	bottom_chan_B																
	3	currentB																
	4	SQRT3																
	5	sync_presc_addr																
	6																	
	7																	
Phase B bottom channel	0	LHtime_BB																
	1	HLtime_BB																
	2	T_copy																
	3	dec																
	4	T																
	5	prescaler																
	6																	
	7																	
Phase C top channel	0	htC																
	1	HLtime_CT																
	2	bottom_chan_C																
	3	currentC																
	4	prsc_copy																
	5	center_time																
	6																	
	7																	

**Table 7. svmStdDt\_top and svmStdDt\_bottom Parameter RAM**

Channel	Parameter	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Phase C bottom channel	0	LHtime_CB															
	1	HLtime_CB															
	2	min_ht															
	3	max_ht															
	4	DT															
	5	MPW															
	6																
	7																

**Table 8. svmStdDt\_top and svmStdDt\_bottom parameter description**

Parameter	Format	Description
Parameters written by CPU		
Ualfa, Ubeta	16-bit fractional	Stator Reference Voltage Vector components
currentA	0 or 1	0 ... positive current on phase A 1 ... negative current on phaseA
currentB	0 or 1	0 ... positive current on phase B 1 ... negative current on phaseB
currentC	0 or 1	0 ... positive current on phase C 1 ... negative current on phaseC
T	16-bit unsigned integer	PWM period in number of TCR1 TPU cycles
prescaler	16-bit unsigned integer	The number of PWM periods per reload of new values
DT	16-bit unsigned integer	Dead-time in number of TCR1 TPU cycles
MPW	16-bit unsigned integer	Minimum pulse width in number of TCR1 TPU cycles. See <a href="#">Performance</a> for details.
SQRT3	16-bit fractional	$\sqrt{3}/2 = 0.866 = \$6EDA$ constant
sync_presc_addr	8-bit unsigned integer	address of synchronization channel <i>prescaler</i> parameter: $\$X4$ , where X is synchronization channel number. \$0 if no synchronization channel is used.
Parameters written by both TPU and CPU		
LD_OK	1-bit	0 ... CPU can update variables 1 ... TPU can read variables CPU sets 1, TPU sets 0
Parameters written by TPU		

**Table 8. svmStdDt\_top and svmStdDt\_bottom parameter description**

Parameter	Format	Description
Sector	16-bit unsigned integer	The position of Stator Reference Voltage Vector in a sector. The Sector can be 1, 2, 3, 4, 5 or 6
fault_pinstate	0 or 1	If fault channel is used, state of fault pin: 0 ... low 1 ... high
Other parameters are just for TPU function inner use.		

*Performance*
**Table 9. svmStdDt\_top State Statistics**

State	Max IMB Clock Cycles	RAM Accesses by TPU
HL	2	1
LH_C5	36	10

**Table 10. svmStdDt\_bottom State Statistics**

State	Max IMB Clock Cycles	RAM Accesses by TPU
INIT	108	32
STOP	38	0
LH	2	1
HL	6	1
LH_RLD	44	16
C1	48	3
C2	48	4
C3	50	3
C4	48	8

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

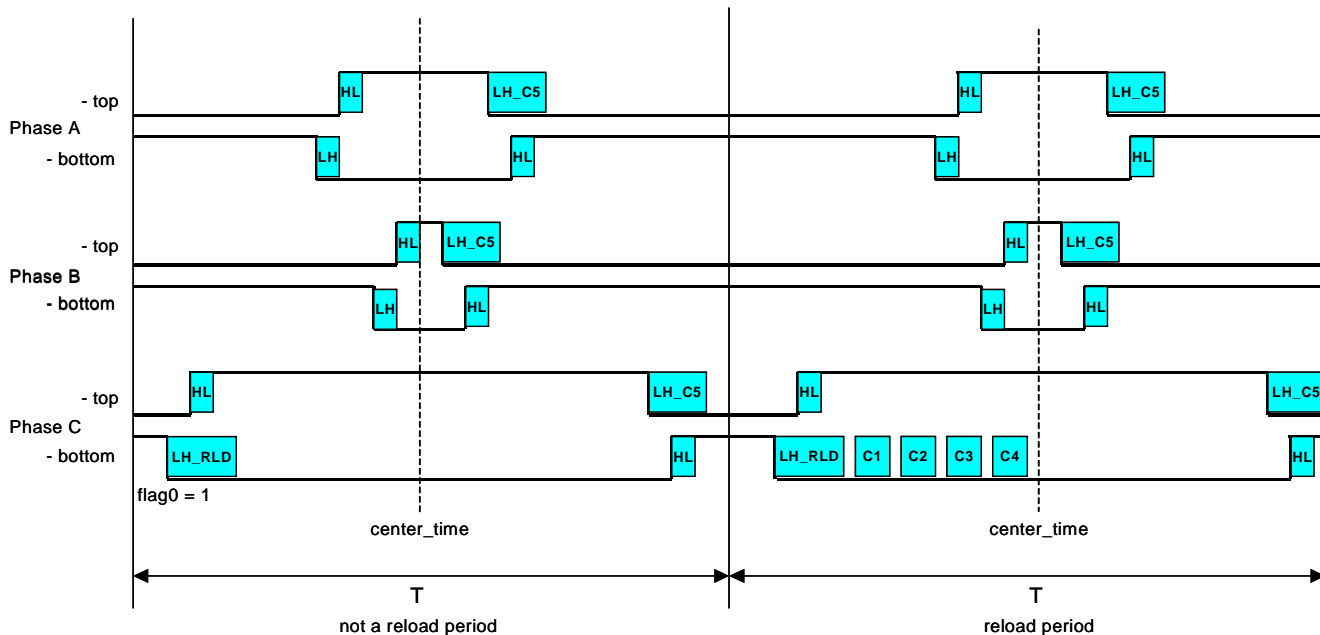


Figure 4. `svmStdDt_top` and `svmStdDt_bottom` timing

**NOTE:** The bottom channel with longest momentary low-time is marked by a `flag0` and runs the `LH_RLD` and `C1`, `C2`, `C3`, `C4` states.

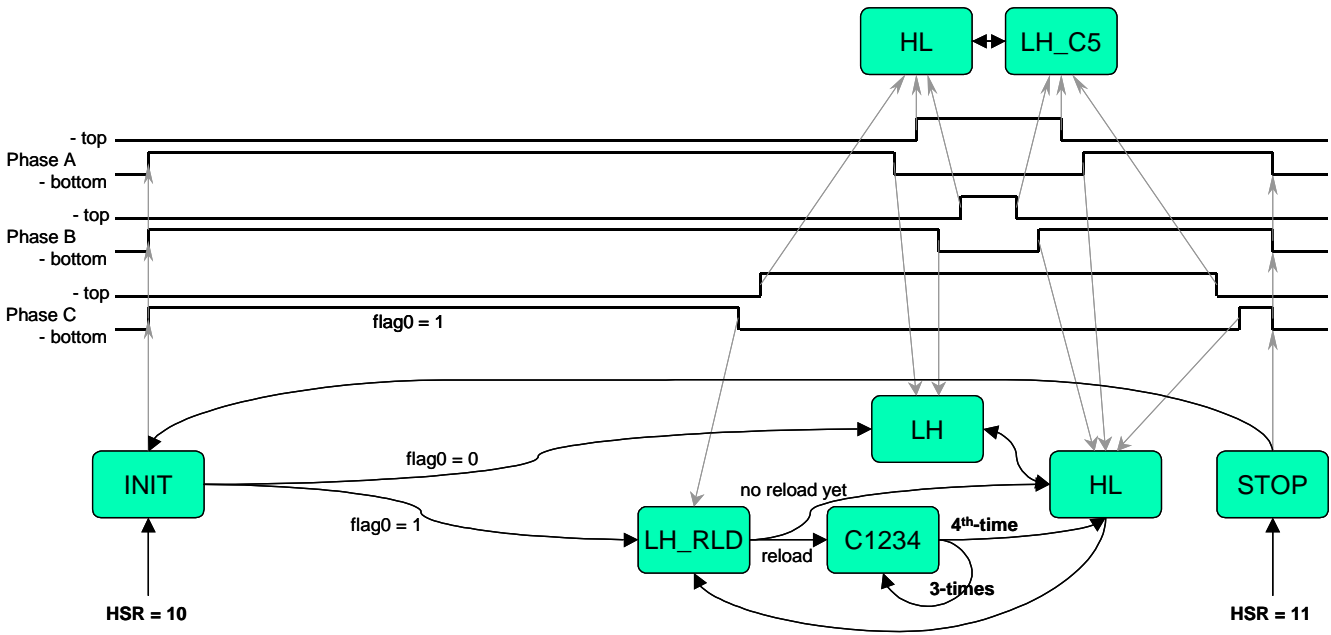


Figure 5. svmStdDt\_top and svmStdDt\_bottom state diagram

*Minimum Pulse Width*

The TPU cannot generate PWM signals with duty cycle ratios very close to 0% or 100%. The minimum pulse width that the TPU can be guaranteed to correctly generate is determined by the TPU function itself and by the activity on the other channels. When the TPU function is requested to generate a narrower pulse a collision can occur. To prevent this, the parameter *MPW* (minimum pulse width) is introduced. The TPU functions *svmStdDt\_top* and *svmStdDt\_bottom* limit the narrowest generated pulse widths to *MPW*. The CPU program should check, and limit, the maximum amplitude of the Stator Reference Voltage Vector before decomposition to  $u_{\hat{a}}$ ,  $u_{\hat{b}}$  components. The maximum amplitude of the Stator Reference Voltage Vector should be less than

$$1 - \frac{2(MPW + 2DT)}{T}$$

If this is not the case, the TPU function will start to limit the minimum pulse widths to *MPW* to prevent a collision, and the duty cycle ratio traces will be deformed as shown on [Figure 6](#).

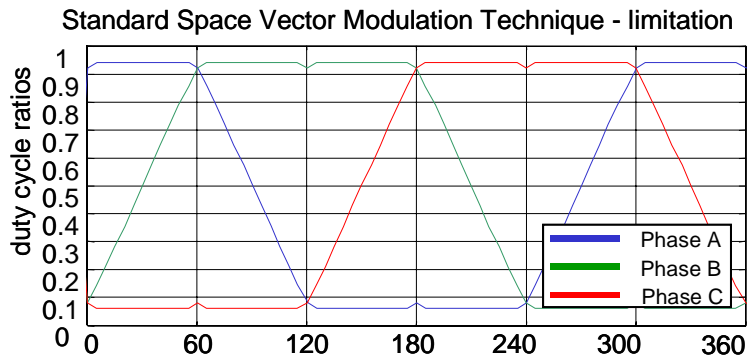


Figure 6. Effect of limitation

The *MPW* is written by the CPU. The *MPW* depends on the whole TPU unit configuration, especially the lengths of the longest states of other functions, and their priorities, running on the same TPU. The *MPW* has to be correctly calculated at the time the whole TPU unit is configured.

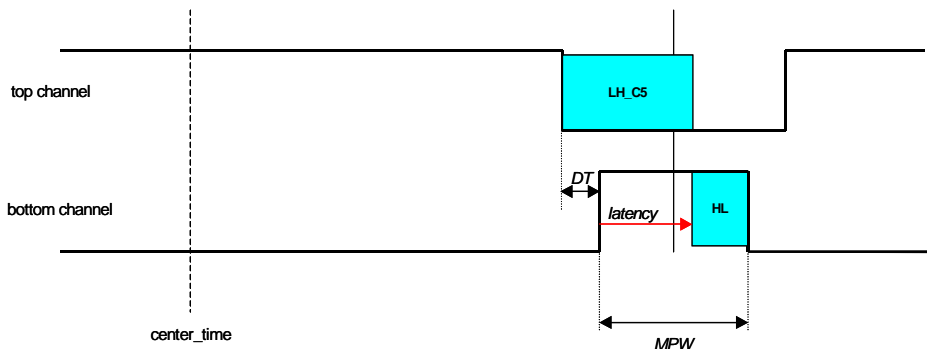


Figure 7. Timing of the worst case

When *svmStdDt\_top* and *svmStdDt\_bottom* are running alone on one TPU, the minimum pulse width can be calculated according to **Figure 7**. This illustrates the worst case timing. The bottom channel low to high transition runs the HL state that sets the following high to low transition. The HL state lasts 6 IMB clock cycles (see **Table 10**). Each state is preceded by the Time Slot Transition (TST), which takes 10 IMB clock cycles. So the time necessary to set the next transition on the bottom channel is 16 IMB clock cycles. In addition, there is a latency between the low to high transition and the start of the HL state. The top channel state LH\_C5, which is serviced at the time, causes the latency. The LH\_C5 state lasts 36 IMB clock cycles (see **Table 9**). Its time slot transition is

10 IMB clock cycles. The service starts immediately after the top channel high to low transition, which occurs at a period of  $DT$  before the bottom channel low to high transition (see **Figure 7**), so that the latency is 36 IMB clock cycles + 10 IMB clock cycles –  $DT$ . The  $svmStdDt$  functions are designed so that no other  $svmStdDt$  state can request service at this time. The  $MPW$ , in the case when only  $svmStdDt$  functions are running on one TPU, is then

$$\begin{aligned} & \text{latency} + 16 \text{ IMB clock cycles} = \\ & = 36 \text{ IMB clock cycles} + 10 \text{ IMB clock cycles} - DT + 16 \text{ IMB clock cycles} = \\ & = 62 \text{ IMB clock cycles} - DT \end{aligned}$$

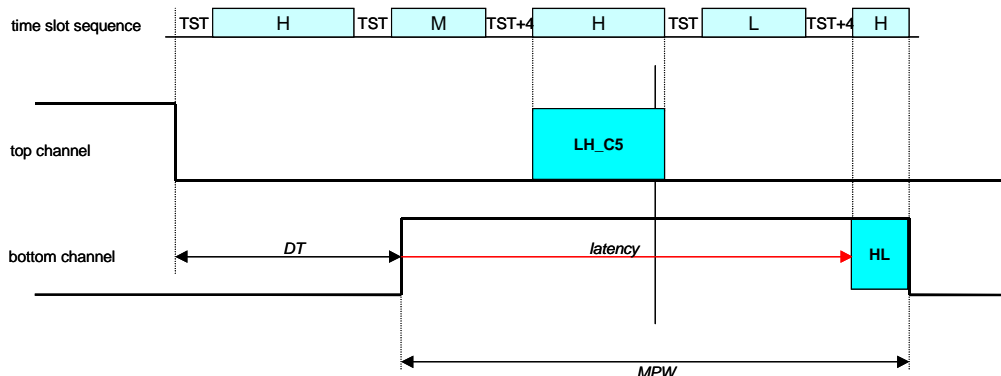
and is a minimum at least 16 IMB clock cycles (when latency = 0).

Note that the  $MPW$ , as well as the  $DT$ , are not entered into the parameter RAM in IMB clock cycles, but in TCR1 clock cycles. It is recommended for the  $svmStdDt$  function to configure the TCR1 clock to its maximum speed, which is the IMB clock divided by 2. In this case the  $MPW = 31 - DT$ , with a minimum value of 8.

When other functions are running together, on the same TPU, with the  $svmStdDt$  functions, the latency could be lengthened. To maintain sufficiently high performance of  $svmStdDt$ , it is recommended that the following rules are followed when configuring the TPU:

- assign  $svmStdDt$  PWM channels high priority
- assign  $svmStdDt$  PWM functions on low channel numbers so that no other function with high priority is assigned a channel with a lower number

In this instance, one of the two worst case timing cases can happen. These are illustrated in **Figure 8** and **Figure 9**. Which case occurs depends on the  $DT$ .



**Figure 8. Worst case timing – case one**



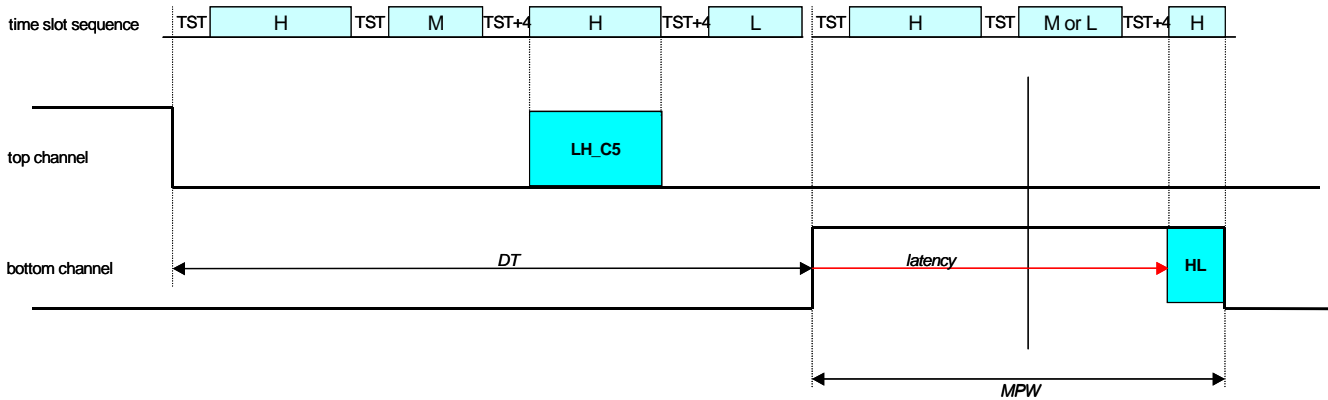


Figure 9. Worst case timing – case two

The time slot sequences at the top of both figures shows when a state of a high (H), middle (M) or low (L) priority is serviced in the worst case. To calculate the *MPW* follow these steps:

- Get the lengths of the longest states.
  - It is necessary to know the lengths of the longest states within all functions of each priority group. The initialization states are not considered – only the running states. Let's denote *H* as the time period of the longest state within all functions running on high priority (Do not consider *svmStdDt* functions). Let's denote *M* as the time period of the longest state within all functions running on middle priority and *L* as the time period of the longest state within all functions running on low priority.
- Decide which timing case can occur.
  - The first case can occur when the *DT* (in IMB clock cycles) is less than  $TST + H + TST + M + TST+4 + LH\_C5 + TST+4 + L$  (see [Figure 9](#)) that is  $4 * TST + 8 + H + M + L + LH\_C5$  that is  $84 + H + M + L$  IMB clock cycles.
 

if *DT* (in IMB clock cycles) <  $84 + H + M + L$   
then – case one  
else – case two
- Calculate *MPW* based on case one or case two.
  - In **case one** the *MPW* is (according to [Figure 8](#))  $TST + H + TST + M + TST+4 + LH\_C5 + TST + L + TST+4 + HL - DT$  that is  $100 + H + M + L - DT$  IMB clock cycles.
 

$MPW$  (in IMB clock cycles) =  $100 + H + M + L - DT$

- In **case two** the MPW is (according to **Figure 9**)  
 $TST + H + TST + \max(M,L) + TST+4 + HL$   
 that is  $40 + H + \max(M,L)$  IMB clock cycles.

$$MPW \text{ (in IMB clock cycles)} = 40 + H + \max(M,L)$$

- Convert *MPW* in IMB clock cycles to *MPW* in TCR1 clock cycles based on TCR1 prescaler settings.

When there are no channels of middle or low priority, simply leave out all the *H* or *L* and the following TST or TST+4 from the formulas.

When the recommended configuration rules are not adhered to, the timing of the worst case is much more complicated. It requires some familiarity with the details of the TPU priority scheme. In this case, the Worst-Case Latency (WCL), which is automatically calculated by the MPC500\_Quick\_Start Graphical Configuration Tool, can serve as a good approximation. This is always longer than the real-case is. Let the WCL be calculated after the configuration of the TPU channels and then find the longest WCL value within all the *svmStdDt* PWM channels. Convert the number, from IMB clock cycles to TCR1 clock cycles, to get the *MPW*.

**Synchronization  
 signal for Standard  
 Space Vector  
 Modulation with  
 Dead-Time  
 Correction  
 (*svmStdDt\_sync*)**

The *svmStdDt\_sync* TPU function uses information obtained from *svmStdDt* PWM functions, the actual PWM center times and the PWM periods. This allows a signal to be generated, which tracks the changes in the PWM period and is always synchronized with the PWM. The synchronization signal is a positive pulse generated repeatedly after the *prescaler* or *presc\_copy* PWM periods (see next paragraph). The low to high transition of the pulse can be adjusted by a parameter, either negative or positive, to go a number of TCR1 TPU cycles before or after the PWM period center time. The pulse width *pw* is another synchronization signal parameter.

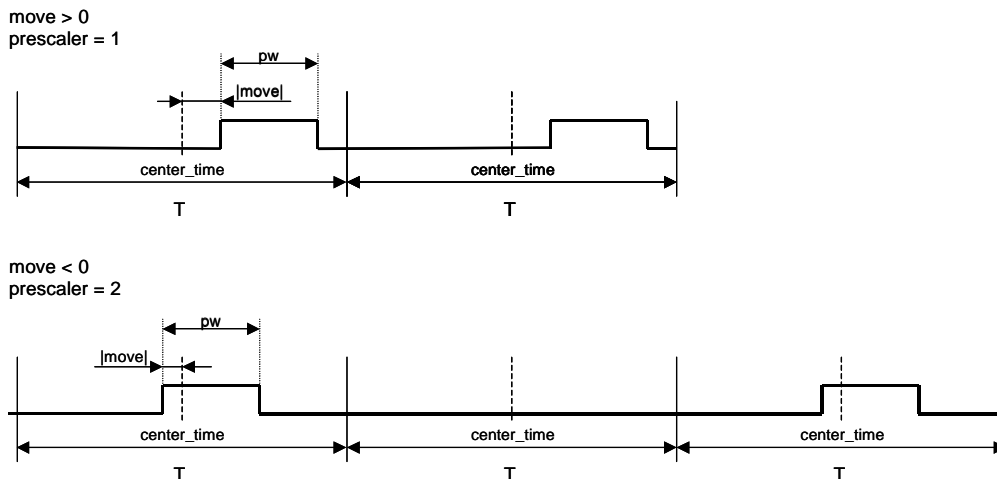


Figure 10. Synchronization signal adjustment examples

*Synchronized Change of PWM Prescaler And Synchronization Signal Prescaler*

The `svmStdDt_sync` TPU function actually uses the `presc_copy` parameter instead of the `prescaler` parameter. The `prescaler` parameter holds the prescaler value that is copied to the `presc_copy` by the `svmStdDt_bottom` function at the time the PWM parameters are reloaded. This ensures that new prescaler values for the PWM signals, as well as the synchronization signal, are applied at the same time. Write the synchronization signal `prescaler` parameter address to the `sync_presc_addr` parameter to enable this mechanism. Write 0 to disable it, and remember to set the synchronization signal `presc_copy` parameter instead of the `prescaler` parameter in this case.

Host Interface

- Written By CPU
- Written by both CPU and TPU
- Written By TPU
- Not Used

Table 11. `svmStdDt_sync` Control Bits

Name	Options
<div style="display: flex; justify-content: space-around; width: 100px;"> <span>3</span><span>2</span><span>1</span><span>0</span> </div> <div style="display: flex; align-items: center; margin-top: 5px;"> <div style="border: 1px solid black; width: 15px; height: 15px; margin-right: 5px;"></div> <div style="border: 1px solid black; width: 15px; height: 15px; margin-right: 5px;"></div> <div style="border: 1px solid black; width: 15px; height: 15px; margin-right: 5px;"></div> <div style="border: 1px solid black; width: 15px; height: 15px;"></div> </div>	<p><code>svmStdDt_sync</code> function number (Assigned during assembly the DPTRAM code from library TPU functions)</p>
<div style="display: flex; justify-content: space-around; width: 40px;"> <span>1</span><span>0</span> </div> <div style="display: flex; align-items: center; margin-top: 5px;"> <div style="border: 1px solid black; width: 15px; height: 15px; margin-right: 5px;"></div> <div style="border: 1px solid black; width: 15px; height: 15px;"></div> </div>	<p>00 – Channel Disabled 01 – Low Priority 10 – Middle Priority 11 – High Priority</p>

**Table 11. svmStdDt\_sync Control Bits**

Name	Options
<div style="display: flex; align-items: center;"> <div style="margin-right: 10px;"> <div style="display: flex; justify-content: space-around; width: 20px;"> <span>1</span><span>0</span> </div> <div style="border: 1px solid black; width: 15px; height: 15px; margin: 2px;"></div> <div style="border: 1px solid black; width: 15px; height: 15px; margin: 2px;"></div> </div> <div>Host Service Bits (HSR)</div> </div>	00 – No Host Service Request 01 – Not used 10 – Initialization 11 – Not used
<div style="display: flex; align-items: center;"> <div style="margin-right: 10px;"> <div style="display: flex; justify-content: space-around; width: 20px;"> <span>1</span><span>0</span> </div> <div style="background-color: green; width: 15px; height: 15px; margin: 2px;"></div> <div style="background-color: green; width: 15px; height: 15px; margin: 2px;"></div> </div> <div>Host Sequence Bits (HSQ)</div> </div>	xx – Not used
<div style="display: flex; align-items: center;"> <div style="margin-right: 10px;"> <div style="display: flex; justify-content: space-around; width: 20px;"> <span>0</span> </div> <div style="border: 1px solid black; width: 15px; height: 15px; margin: 2px;"></div> </div> <div>Channel Interrupt Enable</div> </div>	0 – Channel Interrupt Disabled 1 – Channel Interrupt Enabled
<div style="display: flex; align-items: center;"> <div style="margin-right: 10px;"> <div style="display: flex; justify-content: space-around; width: 20px;"> <span>0</span> </div> <div style="background-color: blue; width: 15px; height: 15px; margin: 2px;"></div> </div> <div>Channel Interrupt Status</div> </div>	0 – Interrupt Not Asserted 1 – Interrupt Asserted

TPU function svmStdDt\_sync generates an interrupt after each low to high transition.

**Table 12. svmStdDt\_sync Parameter RAM**

Channel	Parameter	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Synchronization channel	0	move															
	1	pw															
	2	prescaler															
	3	presc_copy															
	4	time															
	5	dec															
	6	T_copy															
	7																

**Table 13. svmStdDt\_sync parameter description**

Parameter	Format	Description
Parameters written by CPU		
move	16-bit signed integer	The number of TCR1 TPU cycles to forego (negative) or come after (positive) the PWM period center time
pw	16-bit unsigned integer	Synchronization pulse width in number of TCR1 TPU cycles.

Table 13. svmStdDt\_sync parameter description

Parameter	Format	Description
prescaler	16-bit unsigned integer	The number of PWM periods per synchronization pulse – use in case of synchronized prescalers change
presc_copy	16-bit unsigned integer	The number of PWM periods per synchronization pulse – use in case of asynchronized prescalers change
Parameters written by TPU		
Other parameters are just for TPU function inner use.		

Performance

There is one limitation. The absolute value of parameter *move* has to be less than a quarter of the PWM period *T*.

$$|move| < \frac{T}{4}$$

Table 14. svmStdDt\_sync State Statistics

State	Max IMB Clock Cycles	RAM Accesses by TPU
INIT	12	5
S1	12	6
S2	8	3
S3	16	7

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

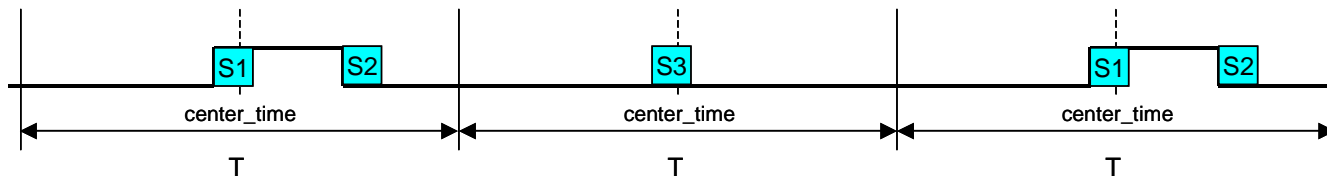


Figure 11. svmStdDt\_sync timing

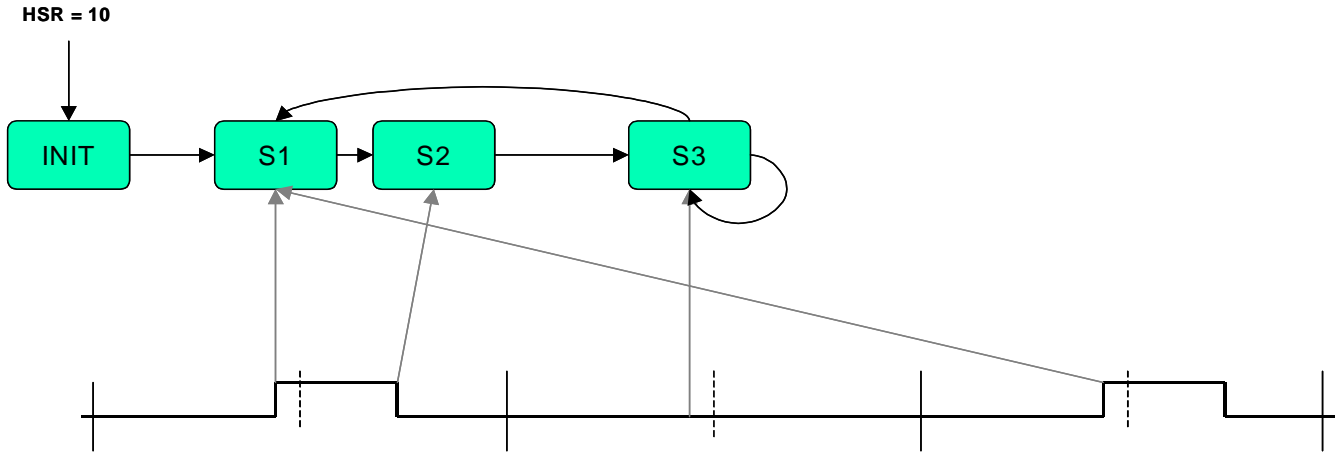
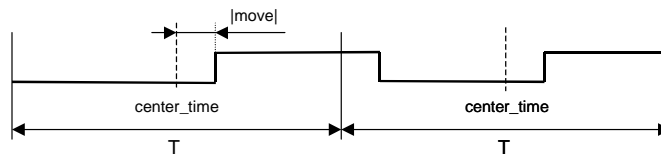


Figure 12. svmStdDt\_sync state diagram

**Resolver Reference Signal for Standard Space Vector Modulation with Dead-Time Correction (svmStdDt\_res)**

The svmStdDt\_res TPU function uses information read from the svmStdDt PWM functions, the actual PWM center times and the PWM periods. This allows a signal to be generated, which tracks the changes of the PWM period and is always synchronized with the PWM. The resolver reference signal is a 50% duty-cycle signal with a period equal to *prescaler* or synchronization channel *presc\_copy* PWM periods (see next paragraph). The low to high transition of the pulse can be adjusted by a parameter, either negative or positive, to go a number of TCR1 TPU cycles before or after the PWM period center time.

move > 0  
prescaler = 1



move < 0  
prescaler = 2

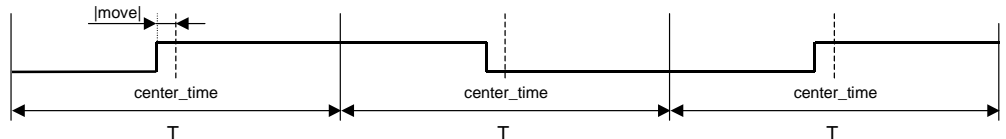






Figure 13. Resolver reference signal adjustment examples

*Synchronized Change of PWM Prescaler And Resolver Reference Signals Prescaler*

The svmStdDt\_res TPU function can inherit the Synchronization Signal prescaler that is synchronously changed with the PWM prescaler. Write the synchronization signals *presc\_copy* parameter address to the *presc\_addr* parameter to enable this mechanism. Write 0 to disable it, and in this case set the *prescaler* parameter to directly specify prescaler value.

*Host Interface*

 Written By CPU	 Written by both CPU and TPU
 Written By TPU	 Not Used

**Table 15. svmStdDt\_res Control Bits**







Name	Options
<div style="display: flex; justify-content: space-around; width: 100px;"> <span>3</span><span>2</span><span>1</span><span>0</span> </div>  Channel Function Select	svmStdDt_res function number (Assigned during assembly the DPTRAM code from library TPU functions)
<div style="display: flex; justify-content: space-around; width: 40px;"> <span>1</span><span>0</span> </div>  Channel Priority	00 – Channel Disabled 01 – Low Priority 10 – Middle Priority 11 – High Priority
<div style="display: flex; justify-content: space-around; width: 40px;"> <span>1</span><span>0</span> </div>  Host Service Bits (HSR)	00 – No Host Service Request 01 – Not used 10 – Initialization 11 – Not used
<div style="display: flex; justify-content: space-around; width: 40px;"> <span>1</span><span>0</span> </div>  Host Sequence Bits (HSQ)	xx – Not used
<div style="display: flex; justify-content: space-around; width: 40px;"> <span>0</span> </div>  Channel Interrupt Enable	x – Not used
<div style="display: flex; justify-content: space-around; width: 40px;"> <span>0</span> </div>  Channel Interrupt Status	x – Not used

Table 16. svmStdDt\_res Parameter RAM

Channel	Parameter	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Resolver	0	move																
	1																	
	2	presc_addr																
	3	prescaler																
	4	time																
	5	dec																
	6	T_copy																
	7																	

Table 17. svmStdDt\_res parameter description

Parameter	Format	Description
Parameters written by CPU		
move	16-bit signed integer	The number of TCR1 TPU cycles to forego (negative) or come after (positive) the PWM period center time
presc_addr	16-bit unsigned integer	\$00X6, where X is a number of Synchronization Signal channel, to inherit Sync. channel prescaler or \$0000 to enable direct specification of prescaler value in prescaler parameter
prescaler	1, 2, 4, 6, 8, 10, 12, 14, ...	The number of PWM periods per synchronization pulse – use when apresc_addr = 0
Parameters written by TPU		
Other parameters are just for TPU function inner use.		

Performance

There is one limitation. The absolute value of parameter *move* has to be less than a quarter of the PWM period *T*.

$$|move| < \frac{T}{4}$$

Table 18. svmStdDt\_res State Statistics

State	Max IMB Clock Cycles	RAM Accesses by TPU
INIT	12	5



Table 18. svmStdDt\_res State Statistics

State	Max IMB Clock Cycles	RAM Accesses by TPU
S1	26	9
S3	18	7

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

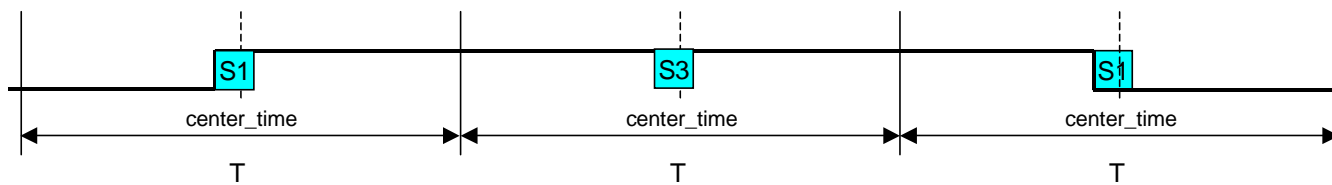


Figure 14. svmStdDt\_res timing

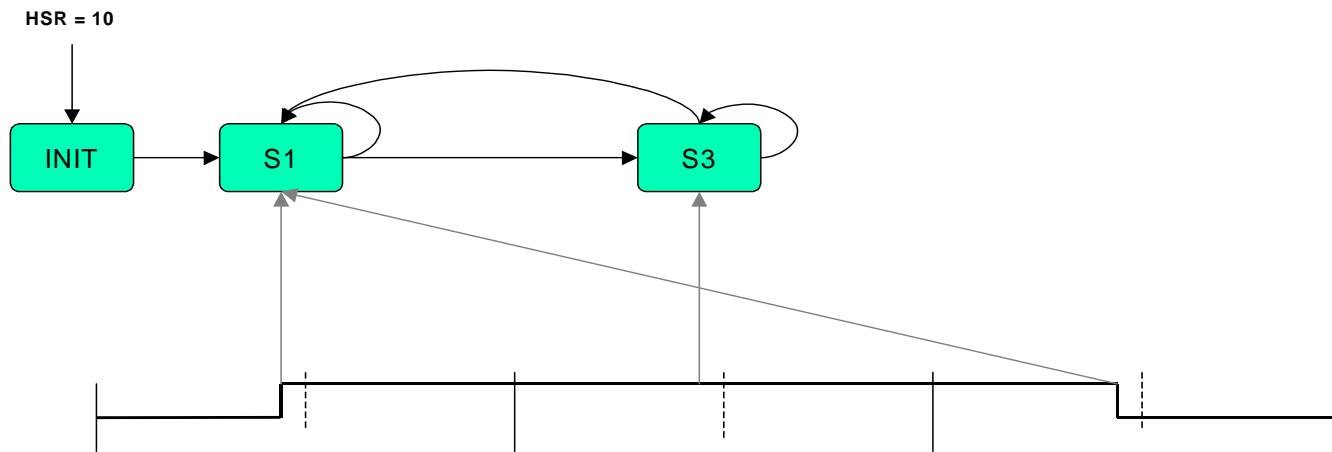


Figure 15. svmStdDt\_res state diagram

**Fault Input for Standard Space Vector Modulation with Dead-Time Correction (svmStdDt\_fault)**

The svmStdDt\_fault is an input TPU function that monitors the pin, and if a high to low transition occurs, immediately sets all PWM channels low and cancels all further transitions on them. The PWM channels, as well as the synchronization and resolver reference signal channels (if used), have to be initialized again to start them running.

The function returns the actual pinstate as a value of 0 (low) or 1 (high) in the parameter *fault\_pinstate*. The parameter is placed on the Phase A – top channel to keep the fault channel parameter space free.

Host Interface

<input type="checkbox"/>	Written By CPU	<input type="checkbox"/>	Written by both CPU and TPU
<input type="checkbox"/>	Written By TPU	<input type="checkbox"/>	Not Used

**Table 19. svmStdDt\_fault Control Bits**

Name	Options
<div style="display: flex; justify-content: space-around; width: 100px;"> <span>3</span><span>2</span><span>1</span><span>0</span> </div> <div style="display: flex; align-items: center;"> <div style="border: 1px solid black; width: 15px; height: 15px; margin-right: 5px;"></div> <div style="border: 1px solid black; width: 15px; height: 15px; margin-right: 5px;"></div> <div style="border: 1px solid black; width: 15px; height: 15px; margin-right: 5px;"></div> <div style="border: 1px solid black; width: 15px; height: 15px;"></div> </div> Channel Function Select	svmStdDt_fault function number (Assigned during assembly the DPTRAM code from library TPU functions)
<div style="display: flex; justify-content: space-around; width: 100px;"> <span>1</span><span>0</span> </div> <div style="display: flex; align-items: center;"> <div style="border: 1px solid black; width: 15px; height: 15px; margin-right: 5px;"></div> <div style="border: 1px solid black; width: 15px; height: 15px;"></div> </div> Channel Priority	00 – Channel Disabled 01 – Low Priority 10 – Middle Priority 11 – High Priority
<div style="display: flex; justify-content: space-around; width: 100px;"> <span>1</span><span>0</span> </div> <div style="display: flex; align-items: center;"> <div style="border: 1px solid black; width: 15px; height: 15px; margin-right: 5px;"></div> <div style="border: 1px solid black; width: 15px; height: 15px;"></div> </div> Host Service Bits (HSR)	00 – No Host Service Request 01 – Not used 10 – Initialization 11 – Not used
<div style="display: flex; justify-content: space-around; width: 100px;"> <span>1</span><span>0</span> </div> <div style="display: flex; align-items: center;"> <div style="background-color: green; width: 15px; height: 15px; margin-right: 5px;"></div> <div style="background-color: green; width: 15px; height: 15px;"></div> </div> Host Sequence Bits (HSQ)	xx – Not used
<div style="display: flex; justify-content: space-around; width: 100px;"> <span>0</span> </div> <div style="display: flex; align-items: center;"> <div style="border: 1px solid black; width: 15px; height: 15px; margin-right: 5px;"></div> </div> Channel Interrupt Enable	0 – Channel Interrupt Disabled 1 – Channel Interrupt Enabled
<div style="display: flex; justify-content: space-around; width: 100px;"> <span>0</span> </div> <div style="display: flex; align-items: center;"> <div style="background-color: blue; width: 15px; height: 15px; margin-right: 5px;"></div> </div> Channel Interrupt Status	0 – Interrupt Not Asserted 1 – Interrupt Asserted

TPU function svmStdDt\_fault generates an interrupt when a high to low transition appears.

**Table 20. svmStdDt\_fault Parameter RAM**

Channel	Parameter	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Fault input	0																
	1																
	2																
	3																
	4																
	5																
	6																
	7																

**Table 21. svmStdDt\_fault parameter description**

Parameter	Format	Description
Parameters written by TPU		
fault_pinstate	0 or 1	State of fault pin: 0 ... low 1 ... high

Performance

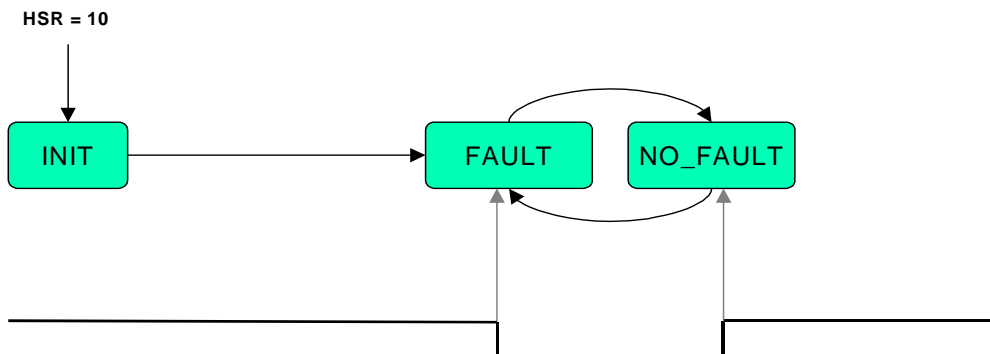
**Table 22. svmStdDt\_fault State Statistics**

State	Max IMB Clock Cycles	RAM Accesses by TPU
INIT	8	2
FAULT	44	1
NO_FAULT	4	1

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



**Figure 16. svmStdDt\_fault timing**



**Figure 17. svmStdDt\_fault state diagram**

## How to Reach Us:

### Home Page:

[www.freescale.com](http://www.freescale.com)

### E-mail:

[support@freescale.com](mailto:support@freescale.com)

### USA/Europe or Locations Not Listed:

Freescale Semiconductor  
 Technical Information Center, CH370  
 1300 N. Alma School Road  
 Chandler, Arizona 85224  
 +1-800-521-6274 or +1-480-768-2130  
[support@freescale.com](mailto:support@freescale.com)

### Europe, Middle East, and Africa:

Freescale Halbleiter Deutschland GmbH  
 Technical Information Center  
 Schatzbogen 7  
 81829 Muenchen, Germany  
 +44 1296 380 456 (English)  
 +46 8 52200080 (English)  
 +49 89 92103 559 (German)  
 +33 1 69 35 48 48 (French)  
[support@freescale.com](mailto:support@freescale.com)

### Japan:

Freescale Semiconductor Japan Ltd.  
 Headquarters  
 ARCO Tower 15F  
 1-8-1, Shimo-Meguro, Meguro-ku,  
 Tokyo 153-0064  
 Japan  
 0120 191014 or +81 3 5437 9125  
[support.japan@freescale.com](mailto:support.japan@freescale.com)

### Asia/Pacific:

Freescale Semiconductor Hong Kong Ltd.  
 Technical Information Center  
 2 Dai King Street  
 Tai Po Industrial Estate  
 Tai Po, N.T., Hong Kong  
 +800 2666 8080  
[support.asia@freescale.com](mailto:support.asia@freescale.com)

### For Literature Requests Only:

Freescale Semiconductor Literature Distribution Center  
 P.O. Box 5405  
 Denver, Colorado 80217  
 1-800-441-2447 or 303-675-2140  
 Fax: 303-675-2150  
[LDCForFreescaleSemiconductor@hibbertgroup.com](mailto:LDCForFreescaleSemiconductor@hibbertgroup.com)

Information in this document is provided solely to enable system and software implementers to use Freescale Semiconductor products. There are no express or implied copyright licenses granted hereunder to design or fabricate any integrated circuits or integrated circuits based on the information in this document. Freescale Semiconductor reserves the right to make changes without further notice to any products herein. Freescale Semiconductor makes no warranty, representation or guarantee regarding the suitability of its products for any particular purpose, nor does Freescale Semiconductor assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. "Typical" parameters which may be provided in Freescale Semiconductor data sheets and/or specifications can and do vary in different applications and actual performance may vary over time. All operating parameters, including "Typicals" must be validated for each customer application by customer's technical experts. Freescale Semiconductor does not convey any license under its patent rights nor the rights of others. Freescale Semiconductor products are not designed, intended, or authorized for use as components in systems intended for surgical implant into the body, or other applications intended to support or sustain life, or for any other application in which the failure of the Freescale Semiconductor product could create a situation where personal injury or death may occur. Should Buyer purchase or use Freescale Semiconductor products for any such unintended or unauthorized application, Buyer shall indemnify and hold Freescale Semiconductor and its officers, employees, subsidiaries, affiliates, and distributors harmless against all claims, costs, damages, and expenses, and reasonable attorney fees arising out of, directly or indirectly, any claim of personal injury or death associated with such unintended or unauthorized use, even if such claim alleges that Freescale Semiconductor was negligent regarding the design or manufacture of the part.

