Application Note

AN2528/D
Rev. 0, 5/2003

Standard Space Vector Modulation TPU Function Set (svmStd)

By Milan Brejl, Ph.D.

Functional Overview

Standard Space Vector Modulation (svmStd) is a technique that is used to implement a straightforward method of switching motor windings in applications such as AC induction motor control and PMSM motor control. The function set consists of 5 TPU functions:

- Standard Space Vector Modulation – Top (svmStd_top)
- Standard Space Vector Modulation – Bottom (svmStd_bottom)
- Synchronization Signal for Standard Space Vector Modulation (svmStd_sync)
- Resolver Reference Signal for Standard Space Vector Modulation (svmStd_res)
- Fault Input for Standard Space Vector Modulation (svmStd_fault)

The svmStd_top and svmStd_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 the svmStd function

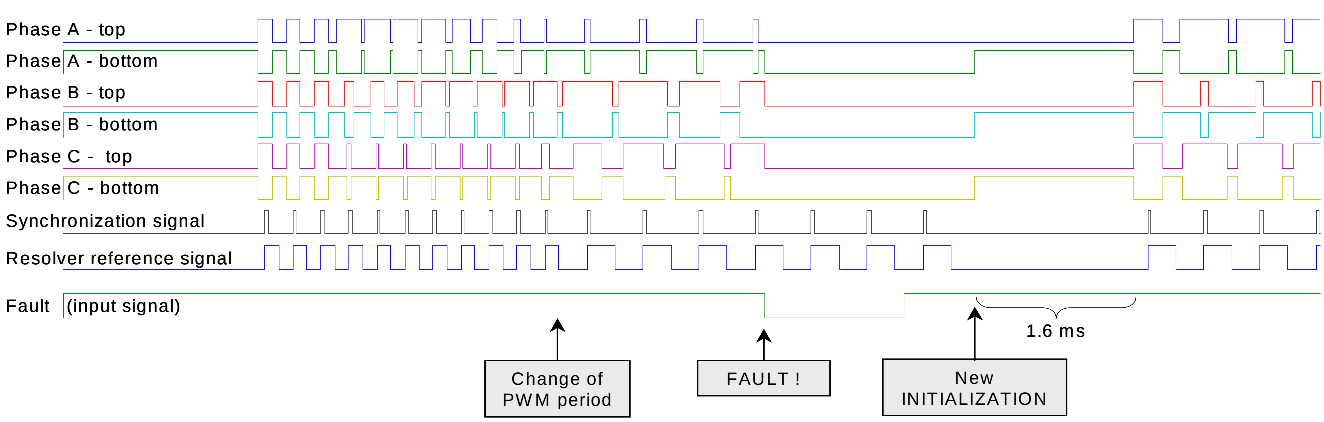


Figure 1. Signals generated by svmStd TPU function set

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 the svmStd 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 the svmStd function is a TPU input function that sets all PWM outputs low when the input signal goes low. See **Figure 1**.

Function Set Configuration

None of the TPU functions in the Standard Space Vector Modulation TPU function set can be used separately. The svmStd_top and svmStd_bottom functions have to be used together. The svmStd_top is used on 3 channels, the svmStd_bottom on a further 3 channels, and within each phase, the function svmStd_top has to be assigned on a lower TPU channel than the function svmStd_bottom. This is illustrated in the examples in **Table 2** and **Table 3**. One or more channels running a Synchronization Signal for svmStd as well as Resolver Reference Signals for svmStd functions can be added to the svmStd_top and svmStd_bottom functions. They can run with different settings on each channel. The function Fault Input for svmStd can also be added to the svmStd_top and svmStd_bottom functions. It is recommended to use it on channel 15, and to select 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. svmStd TPU function set configuration options and restrictions

TPU function	Optional/ Mandatory	How many channels	Assignable channels
svmStd_top	mandatory	3	any 3 channels, within each phase a lower TPU channel than the same phase svmStd_bottom
svmStd_bottom	mandatory	3	any 3 channels, within each phase a higher TPU channel than the same phase svmStd_top
svmStd_sync	optional	1 or more	any channels
svmStd_res	optional	1 or more	any channels
svmStd_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	svmStd_top	high
1	svmStd_bottom	high
2	svmStd_top	high
3	svmStd_bottom	high
4	svmStd_top	high
5	svmStd_bottom	high
10	svmStd_sync	low
15	svmStd_fault	high

Table 3. Example of configuration

Channel	TPU function	Priority
0	svmStd_top	high
1	svmStd_top	high
2	svmStd_top	high
3	svmStd_bottom	high
4	svmStd_bottom	high
5	svmStd_bottom	high
10	svmStd_sync	low
11	svmStd_res	low
15	svmStd_fault	high

Table 4 shows the TPU function code sizes.

Table 4. TPU function code sizes

TPU function	Code size
svmStd_top	16 µ instructions + 8 entries = 24 long words
svmStd_bottom	197 µ instructions + 8 entries = 205 long words
svmStd_sync	26 µ instructions + 8 entries = 34 long words
svmStd_res	38 µ instructions + 8 entries = 46 long words
svmStd_fault	9 µ 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 svmStd_sync channel or an svmStd_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 svmStd_bottom channels to initialize all PWM channels. Issues an HSR type %10 to the svmStd_sync channels, svmStd_res channels and svmStd_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 svmStd_sync or svmStd_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 svmStd_sync or svmStd_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 – Top (svmStd_top) and Standard Space Vector Modulation – Bottom (svmStd_bottom)

The svmStd_top and svmStd_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 reload values are processed.

The CPU controls the PWM output by setting the TPU parameters. The Stator Reference Voltage Vector components $u_{á}$ and $u_{â}$ 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 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 $u_{á}$ and $u_{â}$, corresponding sectors and output PWM signal duty cycle ratios:

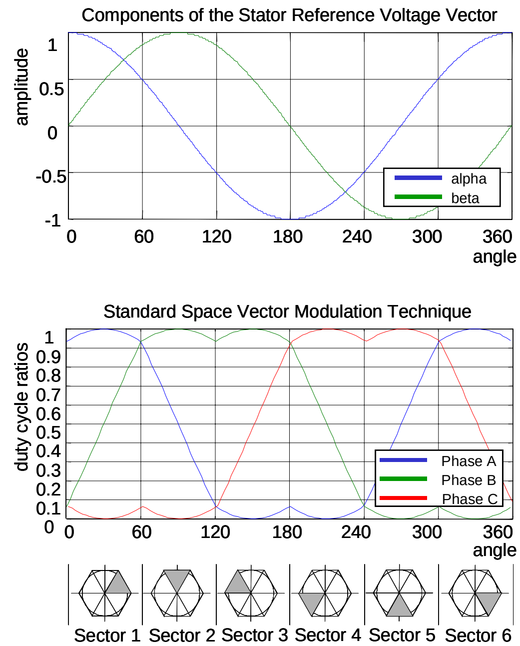


Figure 2. Standard Space Vector Modulation Technique

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

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

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

$$\mathrm{X} = U_\beta$$

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

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

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

Sector I., IV.:

$$ht_A = \frac{T+X-Z}{2}$$

$$ht_B = \frac{T+X+Z}{2} = t_A + Z$$

$$ht_C = \frac{T-X+Z}{2} = t_B - X$$

Sector II., V.:

$$ht_A = \frac{T+Y-Z}{2}$$

$$ht_B = \frac{T+Y+Z}{2} = t_A + Z$$

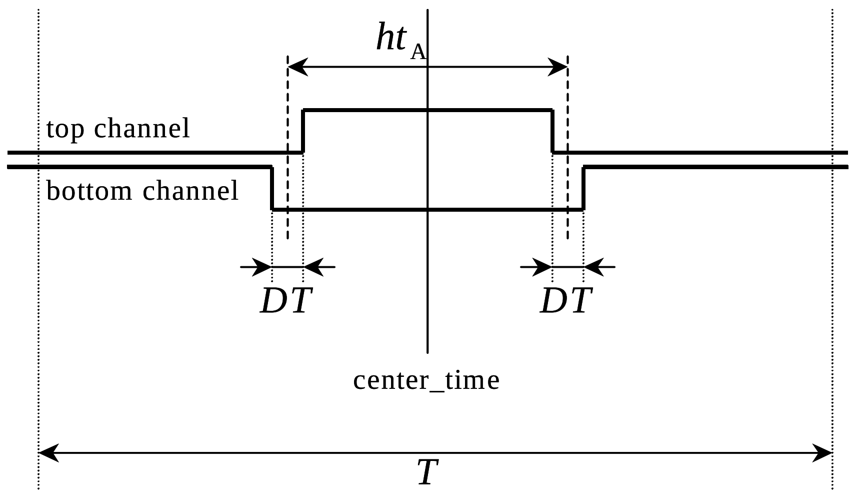
$$ht_C = \frac{T-Y-Z}{2} = t_A - Y$$

Sector III., VI.:

$$ht_A = \frac{T-X+Y}{2}$$

$$ht_B = \frac{T+X-Y}{2} = t_C + X$$

$$ht_C = \frac{T-X-Y}{2} = t_A - Y$$



Phase A:

– top channel

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

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

– bottom channel

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

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

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. svmStd_top Control Bits

Name		Options
3 2 1 0	Channel Function Select	svmStd_top function number (Assigned during assembly the DPTRAM code from library TPU functions)
1 0	Channel Priority	00 – Channel Disabled 01 – Low Priority 10 – Middle Priority 11 – High Priority
1 0	Host Service Bits (HSR)	00 – No Host Service Request 01 – Not used 10 – Not used 11 – Not used
1 0	Host Sequence Bits (HSQ)	xx – Not used
0	Channel Interrupt Enable	x – Not used
0	Channel Interrupt Status	x – Not used

Table 6. svmStd_bottom Control Bits

Name		Options
3 2 1 0	Channel Function Select	svmStd_bottom function number (Assigned during assembly the DPTRAM code from library TPU functions)
1 0	Channel Priority	00 – Channel Disabled 01 – Low Priority 10 – Middle Priority 11 – High Priority
1 0	Host Service Bits (HSR)	00 – No Host Service Request 01 – Not used 10 – Initialization 11 – Stop

Table 6. svmStd_bottom Control Bits

Name		Options
1 0	Host Sequence Bits (HSQ)	xx – Not used
0	Channel Interrupt Enable	0 – Channel Interrupt Disabled 1 – Channel Interrupt Enabled
0	Channel Interrupt Status	0 – Interrupt Not Asserted 1 – Interrupt Asserted

TPU function svmStd_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. svmStd_top and svmStd_bottom Parameter RAM

Channel	Parameter	15–0
Phase A top channel	0	htA
	1	HLtime_AT
	2	bottom_chan_A
	3	center_time
	4	LD_OK
	5	Sector
	6	
	7	fault_pinstate
Phase A bottom channel	0	LHtime_AB
	1	HLtime_AB
	2	UA
	3	UB
	4	Ualfa
	5	Ubeta
	6	
	7	

Table 7. svmStd_top and svmStd_bottom Parameter RAM

<table>
<tr><th>Channel</th><th>Parameter</th><th>15</th><th>14</th><th>13</th><th>12</th><th>11</th><th>10</th><th>9</th><th>8</th><th>7</th><th>6</th><th>5</th><th>4</th><th>3</th><th>2</th><th>1</th><th>0</th></tr>
<tr><td rowspan="8">Phase B top channel</td><td>0</td><td colspan="16">htB</td></tr>
<tr><td>1</td><td colspan="16">HLtime_BT</td></tr>
<tr><td>2</td><td colspan="16">bottom_chan_B</td></tr>
<tr><td>3</td><td colspan="16">UA3</td></tr>
<tr><td>4</td><td colspan="16">SQRT3</td></tr>
<tr><td>5</td><td colspan="16">sync_presc_addr</td></tr>
<tr><td>6</td><td colspan="16"></td></tr>
<tr><td>7</td><td colspan="16"></td></tr>
<tr><td rowspan="8">Phase B bottom channel</td><td>0</td><td colspan="16">LHtime_BB</td></tr>
<tr><td>1</td><td colspan="16">HLtime_BB</td></tr>
<tr><td>2</td><td colspan="16">T_copy</td></tr>
<tr><td>3</td><td colspan="16">dec</td></tr>
<tr><td>4</td><td colspan="16">T</td></tr>
<tr><td>5</td><td colspan="16">prescaler</td></tr>
<tr><td>6</td><td colspan="16"></td></tr>
<tr><td>7</td><td colspan="16"></td></tr>
<tr><td rowspan="8">Phase C top channel</td><td>0</td><td colspan="16">htC</td></tr>
<tr><td>1</td><td colspan="16">HLtime_CT</td></tr>
<tr><td>2</td><td colspan="16">bottom_chan_C</td></tr>
<tr><td>3</td><td colspan="16">prsc_copy</td></tr>
<tr><td>4</td><td colspan="16"></td></tr>
<tr><td>5</td><td colspan="16"></td></tr>
<tr><td>6</td><td colspan="16"></td></tr>
<tr><td>7</td><td colspan="16"></td></tr>
<tr><td rowspan="8">Phase C bottom channel</td><td>0</td><td colspan="16">LHtime_CB</td></tr>
<tr><td>1</td><td colspan="16">HLtime_CB</td></tr>
<tr><td>2</td><td colspan="16">min_ht</td></tr>
<tr><td>3</td><td colspan="16">max_ht</td></tr>
<tr><td>4</td><td colspan="16">DT</td></tr>
<tr><td>5</td><td colspan="16">MPW</td></tr>
<tr><td>6</td><td colspan="16"></td></tr>
<tr><td>7</td><td colspan="16"></td></tr>
</table>

Table 8. svmStd_top and svmStd_bottom parameter description

Parameter	Format	Description
Parameters written by CPU		
Ualfa, Ubeta	16-bit fractional	Stator Reference Voltage Vector components
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

Table 8. svmStd_top and svmStd_bottom parameter description

Parameter	Format	Description
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 **Performance** for details.
SQRT3	16-bit fractional	sqrt(3)/2 = 0.866 = $6EDA constant
sync_presc_addr	8-bit unsigned integer	address of synchronization channel *prescaler* 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		
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. svmStd_top State Statistics

State	Max IMB Clock Cycles	RAM Accesses by TPU
HL	2	1
LH_C5	28	10

Table 10. svmStd_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

Table 10. svmStd_bottom State Statistics

State	Max IMB Clock Cycles	RAM Accesses by TPU
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)*

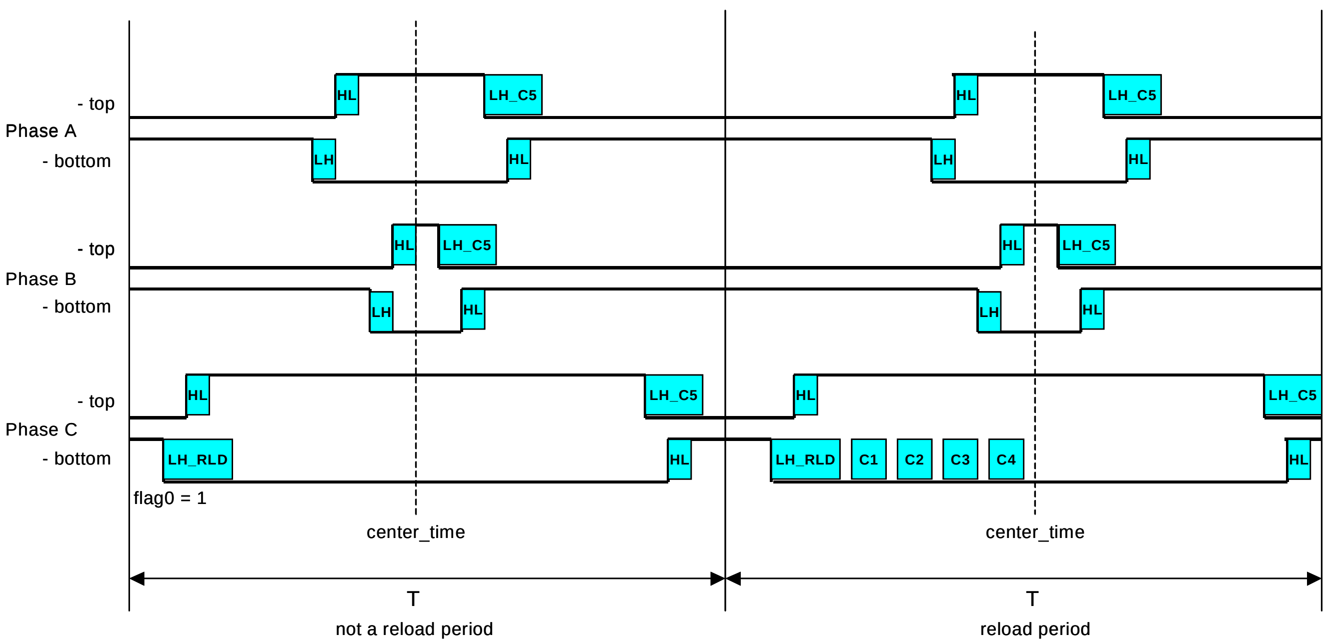


Figure 3. svmStd_top and svmStd_bottom timing

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.*

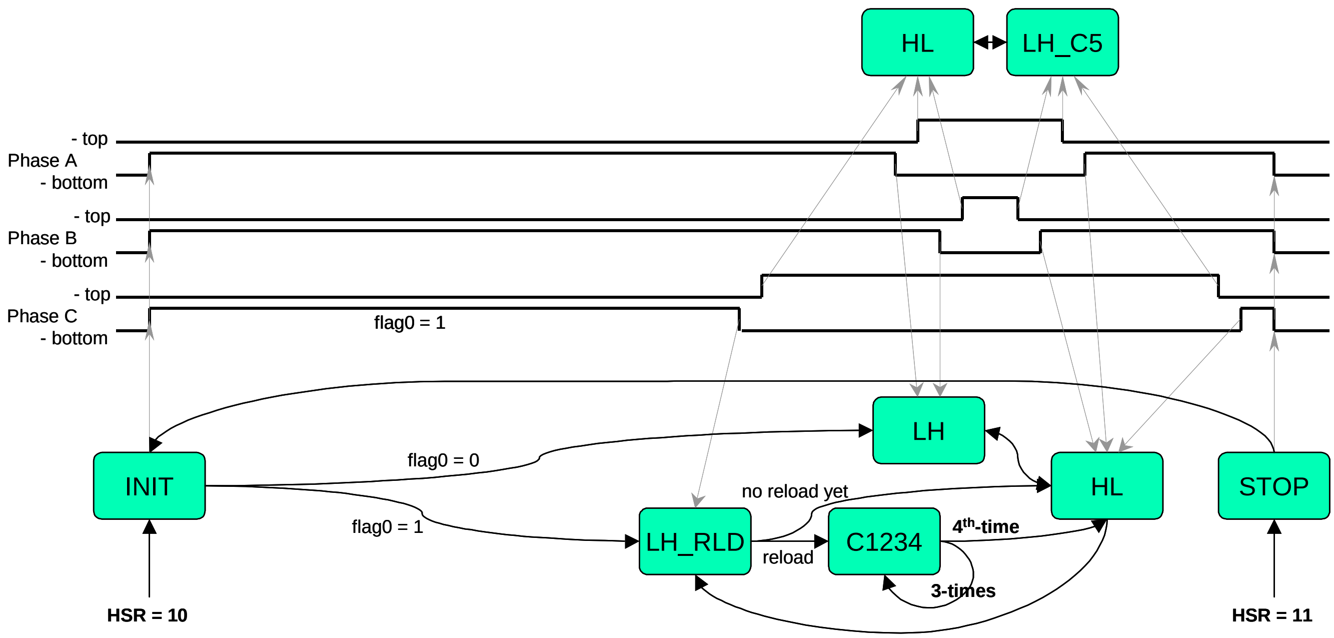


Figure 4. svmStd_top and svmStd_bottom state diagram

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 svmStd_top and svmStd_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 $u_{á}$, $u_{â}$ components. The maximum amplitude of the Stator Reference Voltage Vector should be less than

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

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 5**.

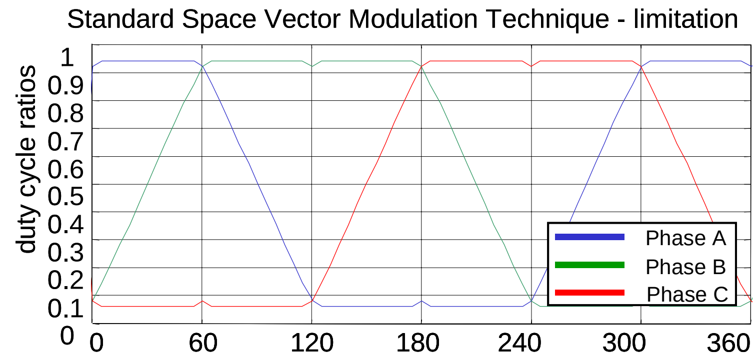


Figure 5. Effect of limitation

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.

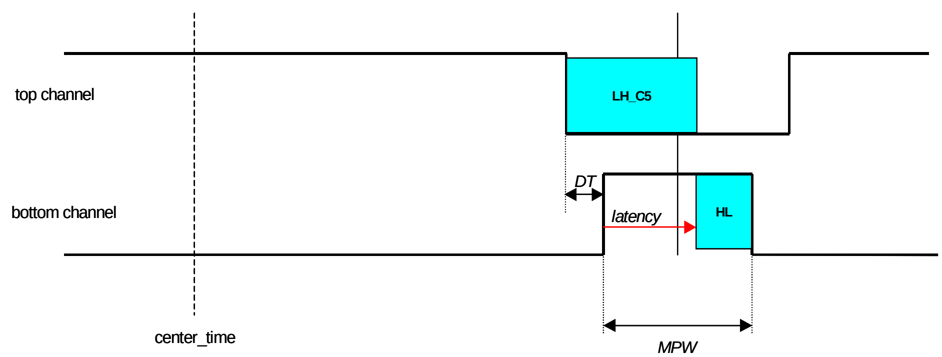


Figure 6. Timing of the worst case

When svmStd_top and svmStd_bottom are running alone on one TPU, the minimum pulse width can be calculated according to **Figure 6**. 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 28 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 6**), so that the latency is 28 IMB clock cycles + 10 IMB clock cycles – *DT*. The svmStd functions are designed so that no other svmStd state can request service at this time. The *MPW*, in the case when only svmStd functions are running on one TPU, is then

$$\text{latency} + 16 \text{ IMB clock cycles} =$$
$$= 28 \text{ IMB clock cycles} + 10 \text{ IMB clock cycles} - DT + 16 \text{ IMB clock cycles} =$$
$$= 54 \text{ IMB clock cycles} - DT$$

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 svmStd function that the TCR1 clck is configured for its maximum speed, which is the IMB clock divided by 2. In this case the *MPW* = 27 – *DT*, with a minimum value of 8.

When other functions are running together on the same TPU as the svmStd functions, the latency could be lengthened. To maintain sufficiently high performance of svmStd, it is recommended that the following rules are followed to configure the TPU:

- assign svmStd PWM channels high priority
- assign svmStd 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 7** and **Figure 8**. Which case occurs depends on the *DT*.

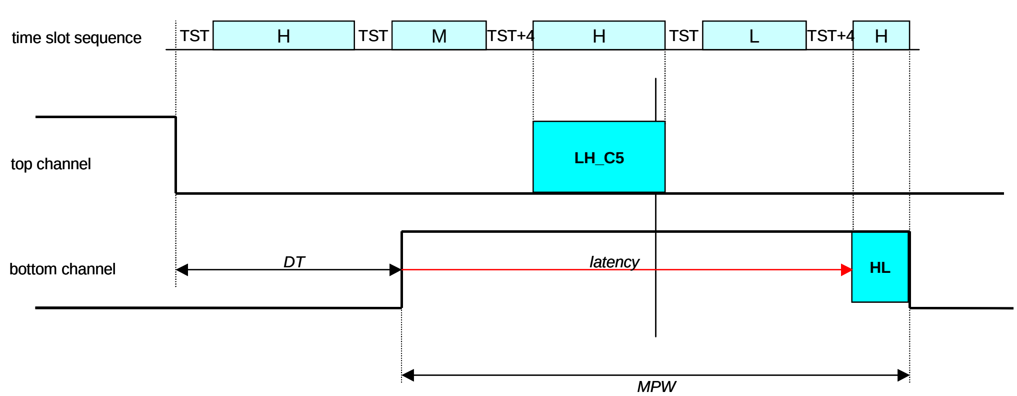


Figure 7. Worst case timing – case one

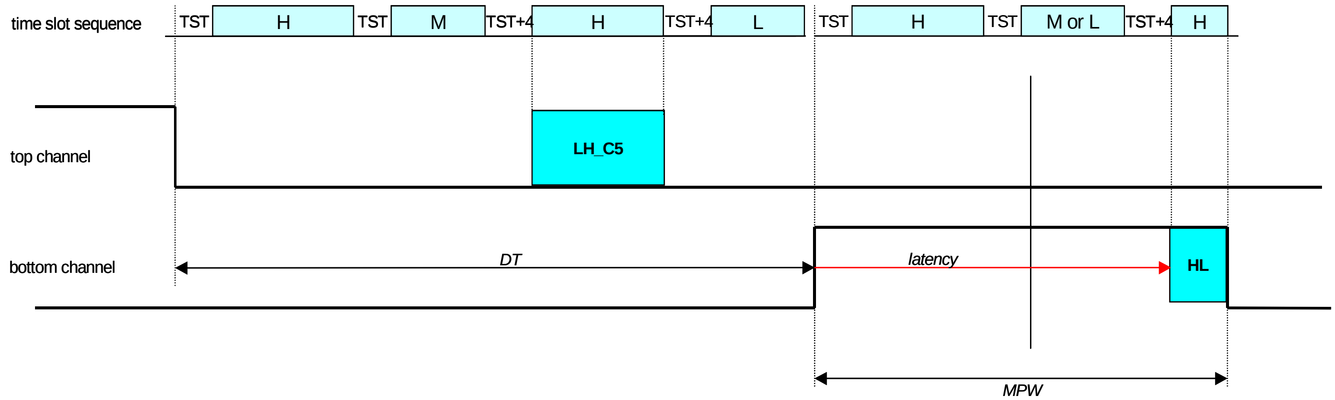


Figure 8. Worst case timing – case two

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 svmStd 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 case of timing can occure.
  - The first case can occure when the *DT* (in IMB clock cycles) is less than TST + *H* + TST + *M* + TST+4 + LH_C5 + TST+4 + *L* (see **Figure 8**) that is 4*TST + 8 + *H* + *M* + *L* + LH_C5 that is 76 + *H* + *M* + *L* IMB clock cycles.

    if *DT* (in IMB clock cycles) < 76 + *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 7**)
    TST + *H* + TST + *M* + TST+4 + LH_C5 + TST + *L* + TST+4 + HL – *DT*
    that is 92 + *H* + *M* + *L* – *DT* IMB clock cycles.

    MPW (in IMB clock cycles) = 92 + *H* + *M* + *L* – *DT*

  - In **case two** the MPW is (according to **Figure 8**)
    TST + *H* + TST + *max*(*M*,*L*) + TST+4 + HL
    that is 40 + *H* + *max*(*M*,*L*) IMB clock cycles.

    MPW (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 considerably 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. Let the WCL be calculated after the configuration of TPU channels and then find the longest WCL value within all svmStd PWM channels. Convert the number, from IMB clock cycles to TCR1 clock cycles, to get the *MPW*.

Synchronization signal for Standard Space Vector Modulation (svmStd_sync)

The svmStd_sync TPU function uses information obtained from svmStd 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.

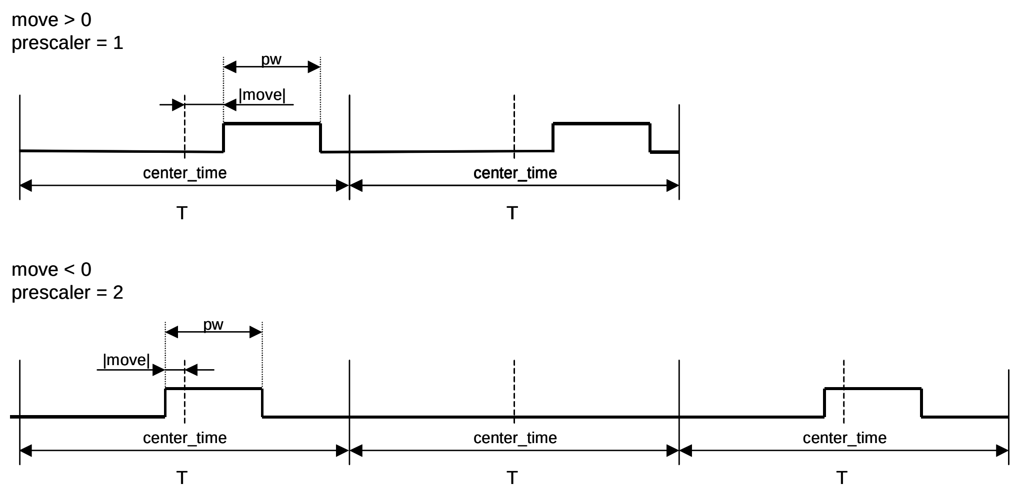


Figure 9. Synchronization signal adjustment examples

Synchronized Change of PWM Prescaler And Synchronization Signal Prescaler

The svmStd_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 svmStd_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. svmStd_sync Control Bits

Name	Options
3 2 1 0 Channel Function Select	svmStd_sync function number (Assigned during assembly the DPTRAM code from library TPU functions)
1 0 Channel Priority	00 – Channel Disabled 01 – Low Priority 10 – Middle Priority 11 – High Priority

Table 11. svmStd_sync Control Bits

Name	Options
1 0 Host Service Bits (HSR)	00 – No Host Service Request 01 – Not used 10 – Initialization 11 – Not used
1 0 Host Sequence Bits (HSQ)	xx – Not used
0 Channel Interrupt Enable	0 – Channel Interrupt Disabled 1 – Channel Interrupt Enabled
0 Channel Interrupt Status	0 – Interrupt Not Asserted 1 – Interrupt Asserted

TPU function svmStd_sync generates an interrupt after each low to high transition.

Table 12. svmStd_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. svmStd_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. svmStd_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. svmStd_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)*

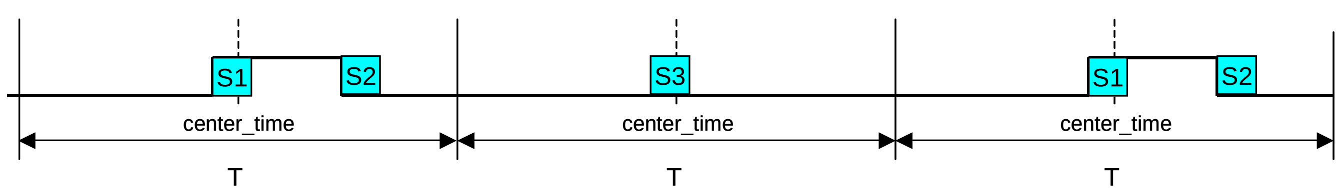


Figure 10. svmStd_sync timing

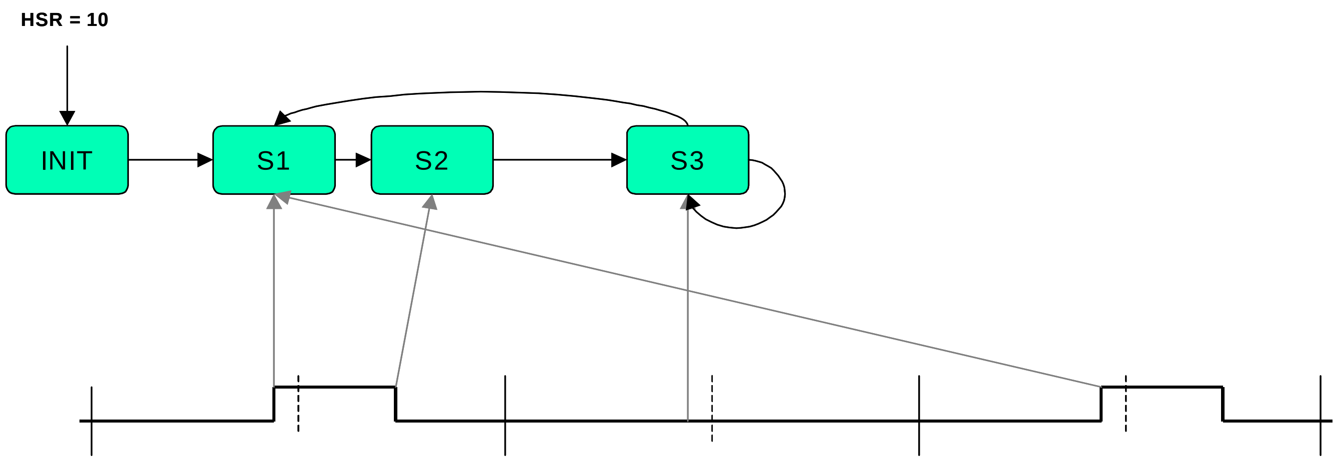


Figure 11. svmStd_sync state diagram

Resolver Reference Signal for Standard Space Vector Modulation (svmStd_res)

The svmStd_res TPU function uses information read from the svmStd 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.

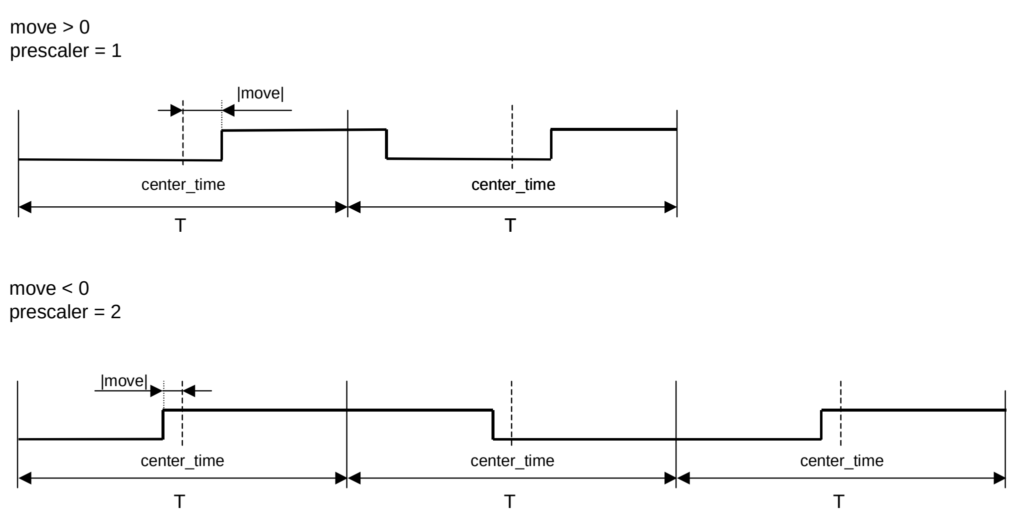


Figure 12. Resolver reference signal adjustment examples

Synchronized Change of PWM Prescaler And Resolver Reference Signals Prescaler

The svmStd_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. svmStd_res Control Bits

Name	Options
3 2 1 0 Channel Function Select	svmStd_res function number (Assigned during assembly the DPTRAM code from library TPU functions)
1 0 Channel Priority	00 – Channel Disabled 01 – Low Priority 10 – Middle Priority 11 – High Priority
1 0 Host Service Bits (HSR)	00 – No Host Service Request 01 – Not used 10 – Initialization 11 – Not used
1 0 Host Sequence Bits (HSQ)	xx – Not used
0 Channel Interrupt Enable	x – Not used
0 Channel Interrupt Status	x – Not used

Table 16. svmStd_res Parameter RAM

<table>
<tr><th>Channel</th><th>Parameter</th><th>15</th><th>14</th><th>13</th><th>12</th><th>11</th><th>10</th><th>9</th><th>8</th><th>7</th><th>6</th><th>5</th><th>4</th><th>3</th><th>2</th><th>1</th><th>0</th></tr>
<tr><td rowspan="8">Resolver</td><td>0</td><td colspan="16">move</td></tr>
<tr><td>1</td><td colspan="16"></td></tr>
<tr><td>2</td><td colspan="16">presc_addr</td></tr>
<tr><td>3</td><td colspan="16">prescaler</td></tr>
<tr><td>4</td><td colspan="16">time</td></tr>
<tr><td>5</td><td colspan="16">dec</td></tr>
<tr><td>6</td><td colspan="16">T_copy</td></tr>
<tr><td>7</td><td colspan="16"></td></tr>
</table>

Table 17. svmStd_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. svmStd_res State Statistics

State	Max IMB Clock Cycles	RAM Accesses by TPU
INIT	12	5
S1	26	9
S3	18	7

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

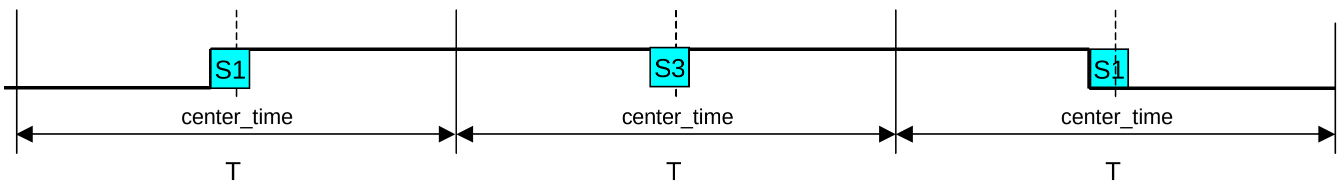


Figure 13. svmStd_res timing

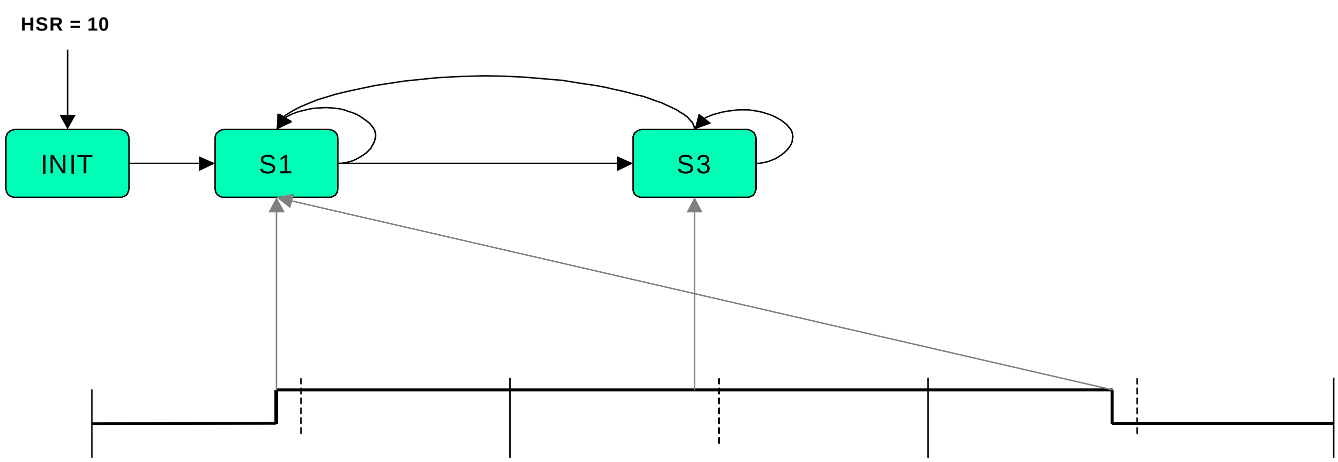


Figure 14. svmStd_res state diagram

Fault Input for Standard Space Vector Modulation (svmStd_fault)

The svmStd_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

(white box)	Written By CPU	(yellow box)	Written by both CPU and TPU
(blue box)	Written By TPU	(green box)	Not Used

Table 19. svmStd_fault Control Bits

Name	Options
Bits 3 2 1 0 — Channel Function Select	svmStd_fault function number (Assigned during assembly the DPTRAM code from library TPU functions)
Bits 1 0 — Channel Priority	00 – Channel Disabled 01 – Low Priority 10 – Middle Priority 11 – High Priority
Bits 1 0 — Host Service Bits (HSR)	00 – No Host Service Request 01 – Not used 10 – Initialization 11 – Not used
Bits 1 0 (not used) — Host Sequence Bits (HSQ)	xx – Not used
Bit 0 — Channel Interrupt Enable	0 – Channel Interrupt Disabled 1 – Channel Interrupt Enabled
Bit 0 (written by TPU) — Channel Interrupt Status	0 – Interrupt Not Asserted 1 – Interrupt Asserted

TPU function svmStd_fault generates an interrupt when a high to low transition appears.

Table 20. svmStd_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. svmStd_fault parameter description

Parameter	Format	Description
Table 22Parameters written by TPU		
fault_pinstate	0 or 1	State of fault pin: 0 ... low 1 ... high

Performance

Table 23. svmStd_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)*

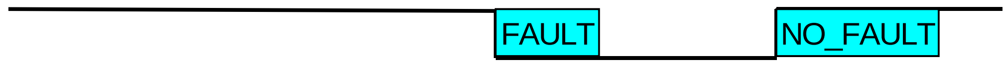


Figure 15. svmStd_fault timing

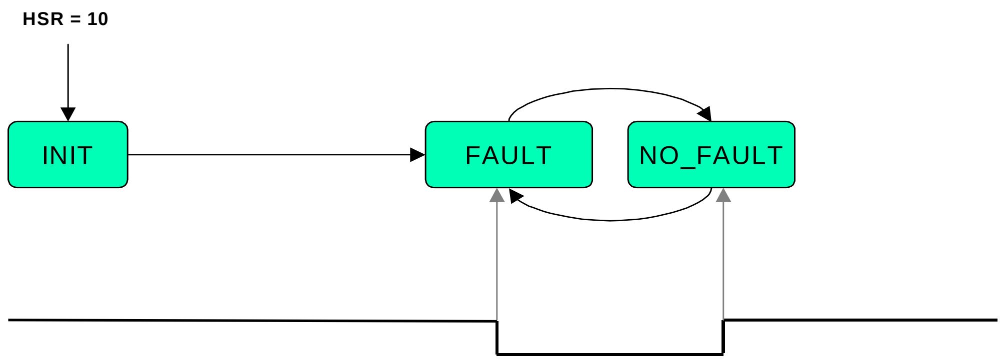


Figure 16. svmStd_fault state diagram

How to Reach Us:

Home Page:
www.freescale.com

E-mail:
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

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

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

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

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

AN2528/D
Rev. 0
5/2003