

Application Note

AN2523/D
Rev. 0, 5/2003

DC Motor – 2 outputs version
TPU Function Set (DCm2)

By Milan Brejl, Ph.D.

Freescale Semiconductor, Inc.

Functional Overview

The DC Motor – 2 outputs version (DCm2) TPU function is a variant of the DCm function, which in unlike the DCm, generates only the top channel signal for each PWM pair. The bottom channel signal can be derived from the top channel signal by external hardware.

The DCm2 function drives a DC Motor, independently of the CPU. The CPU is only required to set a duty-cycle (*dc*) parameter in the range (-1,1), which determines both the speed and the direction. The function generates unipolar-switched center-aligned PWM signals.

The function set consists of 4 TPU functions:

- DC Motor – 2 outputs version (DCm2)
- Synchronization Signal for DC Motor – 2 outputs version (DCm2_sync)
- Resolver Reference Signal for DC Motor – 2 outputs version (DCm2_res)
- Fault Input for DC Motor – 2 outputs version (DCm2_fault)

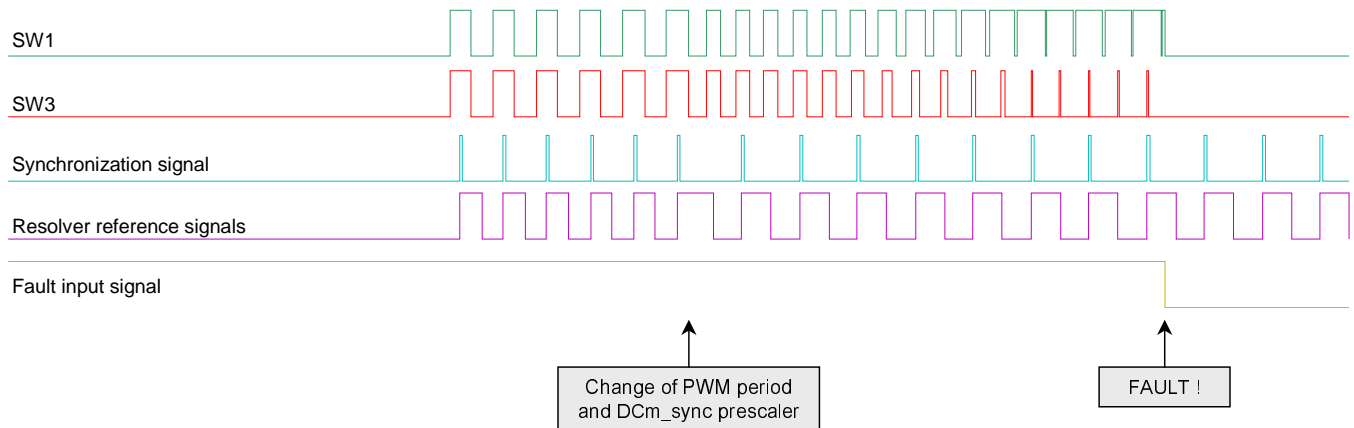


Figure 1. Signals processed by DCm2 TPU function set

The DCm2 TPU function generates a 2-channel 2-phase center-aligned PWM signal. The Synchronization Signal for the DCm2 function can be used to generate one or more adjustable signals for a wide range of uses, which are synchronized to the PWM, and track changes in the PWM period. The Resolver Reference Signal for the DCm2 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 DCm2 function is a TPU input function that sets all PWM outputs low when the input signal goes low. See [Figure 1](#).

Function Set Configuration

The DCm2 function must be used on 2 output channels. One or more channels running Synchronization Signal for DCm2 as well as Resolver Reference Signals for the DCm2 functions can be added. They can run with different settings on each channel. The function Fault Input for the DCm2 can also be added. 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. DCm2 TPU function set configuration options and restrictions

TPU function	Optional/ Mandatory	How many channels	Assignable channels
DCm2	mandatory	2	any 2 channels
DCm2_sync	optional	1 or more	any channels
DCm2_res	optional	1 or more	any channels
DCm2_fault	optional	1	any, recommended is 15 and DTPU bit set

[Table 2](#) shows an example of configuration.

Table 2. Example of configuration

Channel	TPU function	Priority
0	DCm2	high
1	DCm2	high
10	DCm2_sync	low
11	DCm2_res	low
15	DCm2_fault	high

Table 3 shows the TPU function code sizes.

Table 3. TPU function code sizes

TPU function	Code size
DCm2	78 μ instructions + 8 entries = 86 long words
DCm2_sync	26 μ instructions + 8 entries = 34 long words
DCm2_res	38 μ instructions + 8 entries = 46 long words
DCm2_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*, *MPW* and *sync_presc_addr* must be set before initialization. If a DCm2_sync channel or a DCm2_res channel is used, then its parameters must also be set before initialization.
4. Issues an HSR (Host Service Request) type %10 to one of the DCm2 channels to initialize all PWM channels. Issues an HSR type %10 to the DCm2_sync channels, DCm2_res channels and DCm2_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 DCm2_sync or DCm2_res function is initialized after the initialization of DCm2:
 - assigns a priority to the PWM channels to enable their initialization
 - if a Synchronization Signal or a Resolver Reference Signal channel is used, waits until the HSR bits are cleared to indicate that initialization of the PWM channels has completed and
 - assigns a priority to the DCm2_sync or DCm2_res channel to enable its initialization

NOTE: *A CPU routine that configures the TPU can be generated automatically using the MPC500_Quick_Start Graphical Configuration Tool.*

Detailed Function Description

DC Motor – 2 outputs version (DCm2)

The DCm2 TPU function generates a 2-channel, 2-phase unipolar-switched center-aligned PWM signal. Unlike DCm, the generated signals are not top-bottom complementary pairs with dead-times but only top-like signals without dead-times. 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 a value 0 of the duty-cycle ratio *dc* until the first *dc* value is processed, or for at least one PWM period.

The CPU controls the PWM output by setting the TPU parameters. The duty-cycle ratio *dc* and PWM period *T* can be adjusted during run time. Conversely, minimum pulse width (*MPW*) is not supposed to be changed during run time. The duty-cycle ratio *dc* can gain a value in the range (-1, 1). The sign controls the motion system direction, while the absolute value controls the amplitude of the applied voltage.

The following figures show the input *dc* value and corresponding output PWM signals:

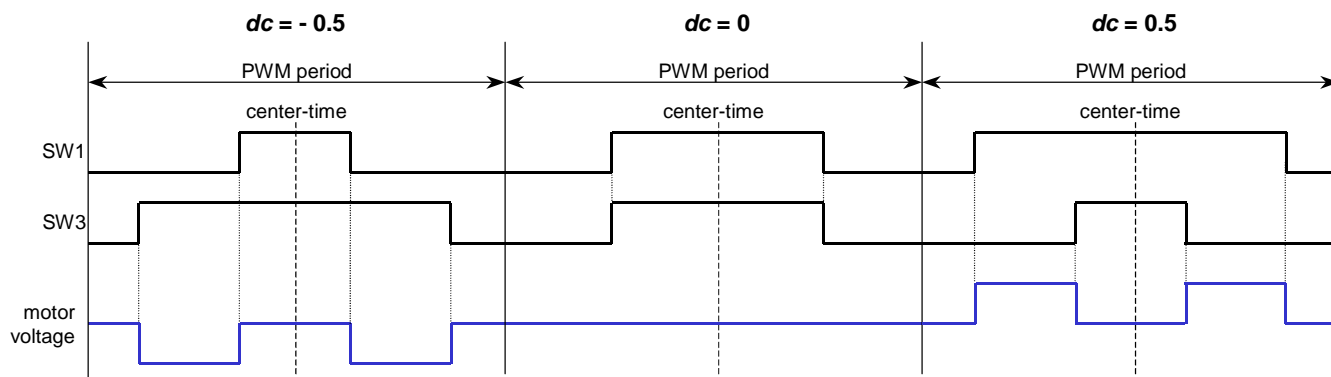


Figure 2. Unipolar switching

The following equations describe how the PWM signal transition times $SW1_{LH}$, $SW3_{LH}$, $SW1_{HL}$ and $SW3_{HL}$ are calculated:

$$T_{dc} = T \cdot dc$$

$$X = \frac{T + Tdc}{4}$$

$$Y = \frac{T - Tdc}{4}$$

$$SWI_{LH} = center_time - X$$

$$SWI_{HL} = center_time + X$$

$$SW3_{LH} = center_time - Y$$

$$SW3_{HL} = center_time + Y$$

Host Interface





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

Table 4. DCm2 Control Bits







Name	Options
<div style="display: flex; justify-content: space-around; width: 100px;"> 3210 </div>  Channel Function Select	DCm2 function number (Assigned during assembly the DPTRAM code from library TPU functions)
<div style="display: flex; justify-content: space-around; width: 100px;"> 10 </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;"> 10 </div>  Host Service Bits (HSR)	00 – No Host Service Request 01 – Not used 10 – Initialization 11 – Stop
<div style="display: flex; justify-content: space-around; width: 100px;"> 10 </div>  Host Sequence Bits (HSQ)	xx – Not used
<div style="display: flex; justify-content: space-around; width: 100px;"> 0 </div>  Channel Interrupt Enable	x – Not used
<div style="display: flex; justify-content: space-around; width: 100px;"> 0 </div>  Channel Interrupt Status	x – Not used

Table 5. DCm2 Parameter RAM

Channel	Parameter	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
SW1	0	LHtime_1																
	1	HLtime_1																
	2	Tdc																
	3	T_copy																
	4	dc																
	5	T																
	6	other_ch_1																
	7	fault_pinstate																
SW3	0	LHtime_3																
	1	HLtime_3																
	2	L																
	3	center_time																
	4	sync_presc_addr																
	5	MPW																
	6	other_ch_3																
	7																	

Table 6. DCm2 parameter description

Parameter	Format	Description
Parameters written by CPU		
dc	16-bit fractional	duty-cycle ratio in the range <-1,1)
T	16-bit unsigned integer	PWM period in number of TCR1 TPU cycles
MPW	16-bit unsigned integer	Minimum pulse width in number of TCR1 TPU cycles. See Performance for details.
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 TPU		
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 7. DCm2 State Statistics

State	Max IMB Clock Cycles	RAM Accesses by TPU
INIT	54	15
STOP	14	0
C1	78	13
C2	32	9
LH	2	1
HL	2	1

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

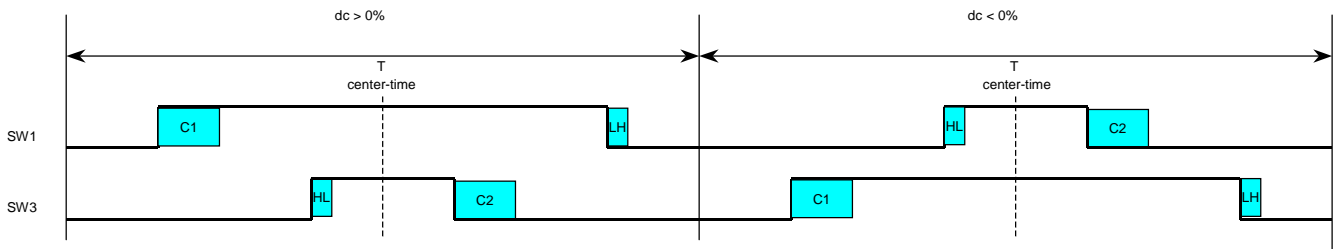


Figure 3. DCm2 timing

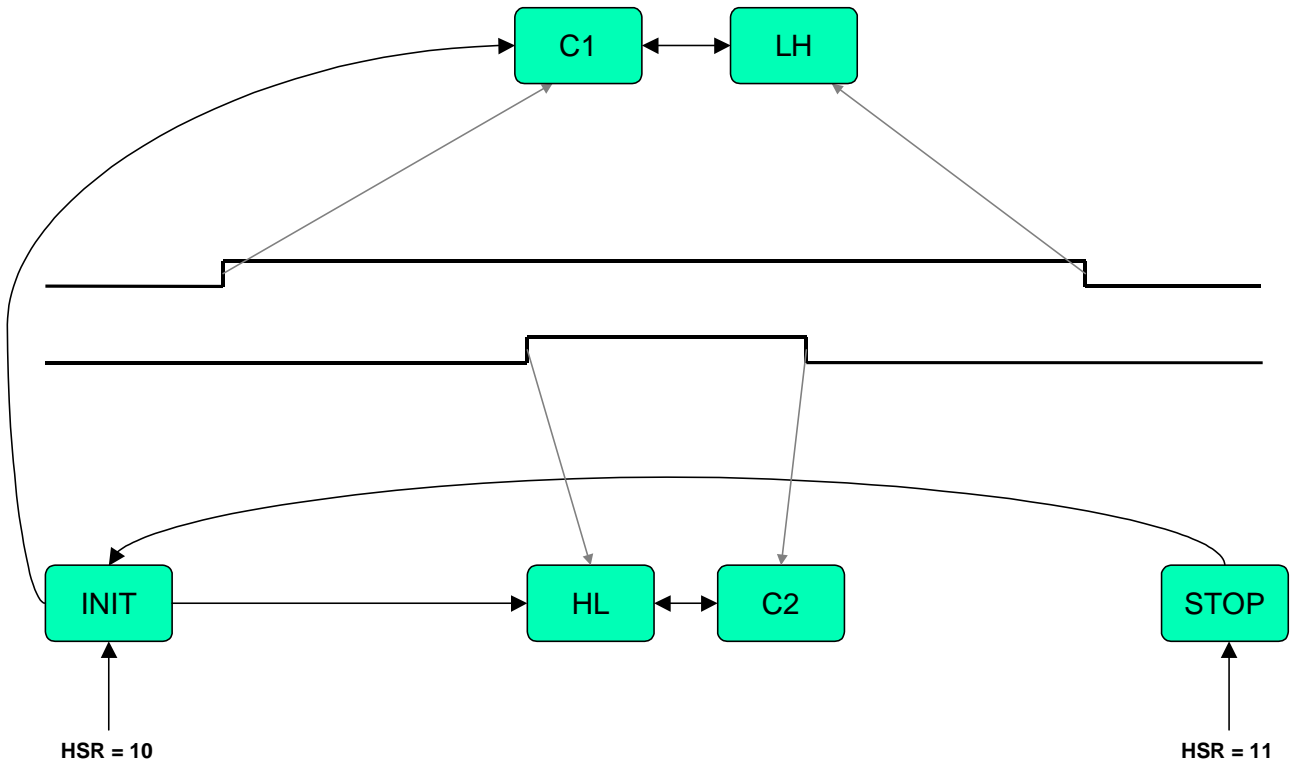


Figure 4. DCm2 state diagram

Minimum Pulse Width

The TPU cannot generate PWM signals with duty cycle ratios very close to 0% or 100%. This is the case when the *dc* value is close to 1 or -1. 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 function DCm2 limits the narrowest generated pulse widths to *MPW*. The CPU program should check the maximum absolute value of *dc* to prevent this limitation, or take into account a non-linear performance when *dc* moves towards the boundary values and the limitation is reached by the TPU. The maximum absolute value of *dc* should satisfy:

$$|dc| \leq 1 - \frac{2MPW}{T}$$

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

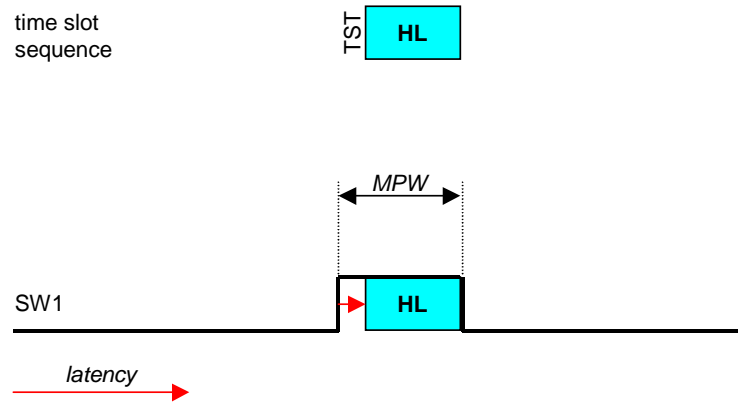


Figure 5. Worst case timing

The minimum pulse width can be calculated according to [Figure 5](#). It illustrates the worst cases of timing in the case when only the DCm2 function is running on one TPU.

According to the [Figure 5](#) the *MPW* is 12 IMB clock cycles.

Note that the *MPW*, as well as the *DT*, are entered into the parameter RAM in TCR1 clock cycles rather than IMB clock cycles. It is recommended for the DCm2 function to configure the TCR1 clock to its maximum speed, which is the IMB clock divided by 2. In this case the *MPW* = 6. This is the absolute minimum value, that the *MPW* can have.

When other functions are running concurrently on the same TPU, the longest state of each function with its time-slot transition can increase the calculated *MPW* value. The DCm2_fault function does not affect the *MPW*. The DCm2_sync, if used, increase the *MPW* value by 22 (44 IMB clock cycles). The DCm2_res, if used, increase the *MPW* value by 20 (40 IMB clock cycles).

If a value lower than the one calculated, but not less than 6 (12 IMB cycles), is assigned to the *MPW* parameter, the motion system can run with a higher motor voltage amplitude, but with a risk, that the maximum amplitude and the center-alignment are not always kept.

You can also use the Worst-Case Latency (WCL), which is automatically calculated by the MPC500_Quick_Start Graphical Configuration Tool. It can serve as a good approximation of *MPW*. The calculated WCL is always longer than the real-case is. Let the WCL be calculated after the configuration of the TPU channels and then read the WCL value on DCm2 PWM channels. Convert the number, from IMB clock cycles to TCR1 clock cycles, to get the *MPW*.

Synchronization signal for DC Motor – 2 outputs version (DCm2_sync)

The DCm2_sync TPU function uses information obtained from DCm2 PWM functions, the actual PWM center times and the PWM periods. This allows a signal to be generated, that 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 before or after the PWM period center time of a number of TCR1 TPU cycles. The pulse width *pw* is another synchronization signal parameter.

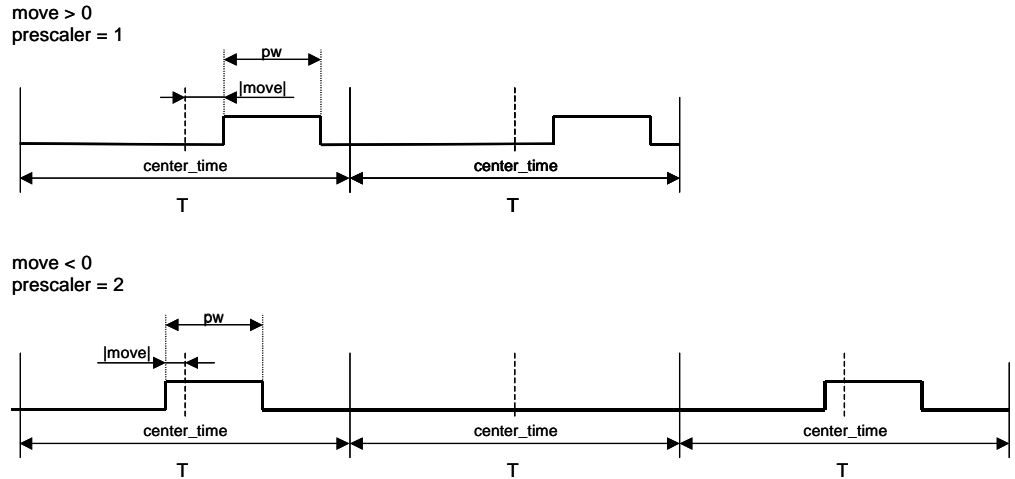


Figure 6. Synchronization signal adjustment examples

Synchronized Change of PWM Prescaler And Synchronization Signal Prescaler

The DCm2_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 DCm2_bottom function at the time of the PWM parameters reload. 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 signals *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 8. DCm2_sync Control Bits

Name		Options
<div style="display: flex; justify-content: space-around; width: 100px;"> 3210 </div> <div style="display: flex; align-items: center;"> <div style="border: 1px solid black; width: 20px; height: 20px; margin-right: 5px;"></div> <div style="border: 1px solid black; width: 20px; height: 20px; margin-right: 5px;"></div> <div style="border: 1px solid black; width: 20px; height: 20px; margin-right: 5px;"></div> <div style="border: 1px solid black; width: 20px; height: 20px;"></div> </div> Channel Function Select	DCm2_sync function number (Assigned during assembly the DPTRAM code from library TPU functions)	
<div style="display: flex; justify-content: space-around; width: 100px;"> 10 </div> <div style="display: flex; align-items: center;"> <div style="border: 1px solid black; width: 20px; height: 20px; margin-right: 5px;"></div> <div style="border: 1px solid black; width: 20px; height: 20px;"></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;"> 10 </div> <div style="display: flex; align-items: center;"> <div style="border: 1px solid black; width: 20px; height: 20px; margin-right: 5px;"></div> <div style="border: 1px solid black; width: 20px; height: 20px;"></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;"> 10 </div> <div style="display: flex; align-items: center;"> <div style="background-color: green; width: 20px; height: 20px; margin-right: 5px;"></div> <div style="background-color: green; width: 20px; height: 20px;"></div> </div> Host Sequence Bits (HSQ)	xx – Not used	
<div style="display: flex; justify-content: space-around; width: 100px;"> 0 </div> <div style="display: flex; align-items: center;"> <div style="border: 1px solid black; width: 20px; height: 20px; 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;"> 0 </div> <div style="display: flex; align-items: center;"> <div style="background-color: blue; width: 20px; height: 20px; margin-right: 5px;"></div> </div> Channel Interrupt Status	0 – Interrupt Not Asserted 1 – Interrupt Asserted	

TPU function DCm2_sync generates an interrupt after each low to high transition.

Table 9. DCm2_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 10. DCm2_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.
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 11. DCm2_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)

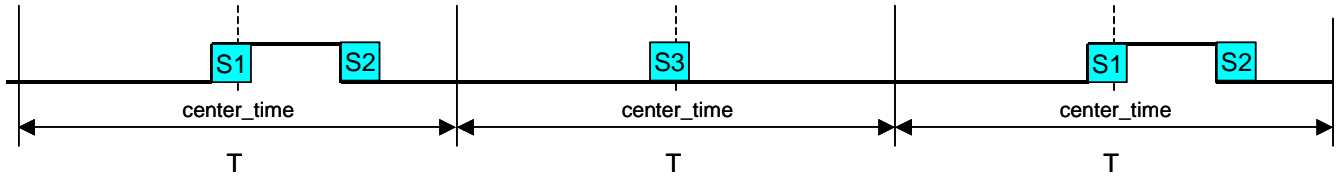


Figure 7. DCm2_sync timing

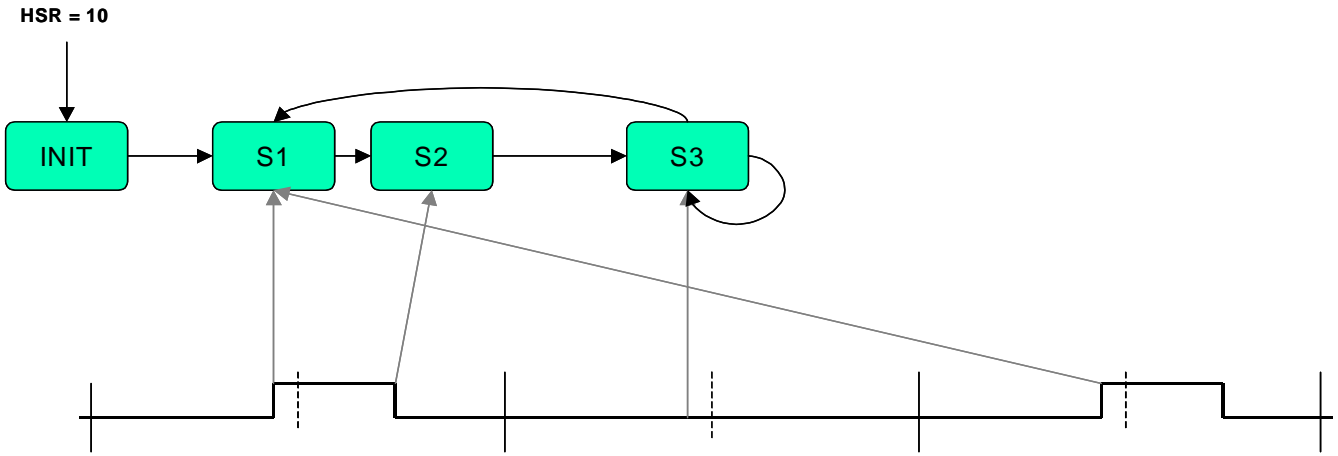


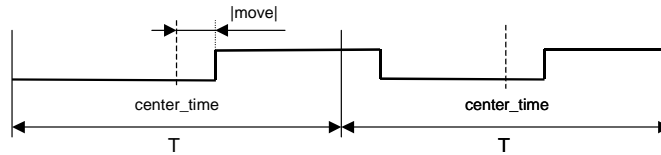
Figure 8. DCm2_sync state diagram

Resolver Reference Signal for DC Motor – 2 outputs version (DCm2_res)

The DCm2_res TPU function uses information read from the DCm2 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 before or after the PWM period center time of a number of TCR1 TPU cycles.

Freescale Semiconductor, Inc.

move > 0
prescaler = 1



move < 0
prescaler = 2

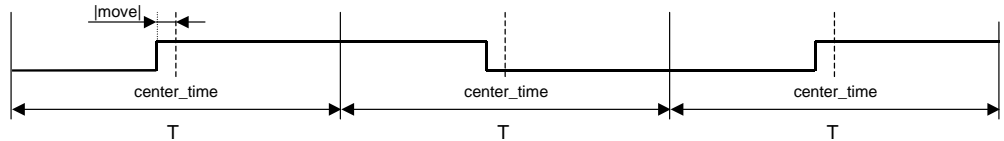


Figure 9. Resolver reference signal adjustment examples

Synchronized Change of PWM Prescaler And Resolver Reference Signals Prescaler

The DCm2_res TPU function can inherit the Synchronization Signal prescaler that is synchronously changed with 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 *prescaler* parameter to directly specify prescaler value.

Host Interface

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

Table 12. DCm2_res Control Bits

Name	Options
<div style="display: flex; justify-content: space-around; font-size: small;"> 3210 </div> <div style="display: flex; align-items: center;"> <div style="border: 1px solid black; width: 20px; height: 15px; margin-right: 5px;"></div> <div style="border: 1px solid black; width: 20px; height: 15px; margin-right: 5px;"></div> <div style="border: 1px solid black; width: 20px; height: 15px; margin-right: 5px;"></div> <div style="border: 1px solid black; width: 20px; height: 15px;"></div> </div>	DCm2_res function number (Assigned during assembly the DPTRAM code from library TPU functions)
Channel Function Select	
<div style="display: flex; justify-content: space-around; font-size: small;"> 10 </div> <div style="display: flex; align-items: center;"> <div style="border: 1px solid black; width: 20px; height: 15px; margin-right: 5px;"></div> <div style="border: 1px solid black; width: 20px; height: 15px;"></div> </div>	00 – Channel Disabled 01 – Low Priority 10 – Middle Priority 11 – High Priority
Channel Priority	
<div style="display: flex; justify-content: space-around; font-size: small;"> 10 </div> <div style="display: flex; align-items: center;"> <div style="border: 1px solid black; width: 20px; height: 15px; margin-right: 5px;"></div> <div style="border: 1px solid black; width: 20px; height: 15px;"></div> </div>	
Host Service Bits (HSR)	
	00 – No Host Service Request 01 – Not used 10 – Initialization 11 – Not used

Table 12. DCm2_res 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;"> 10 </div> <div style="display: flex; justify-content: space-around; width: 20px;"> <div style="width: 10px; height: 10px; background-color: red;"></div> <div style="width: 10px; height: 10px; background-color: red;"></div> </div> </div> <div> Host Sequence Bits (HSQ) </div> <div style="margin-left: 20px;"> xx – Not used </div> </div>	
<div style="display: flex; align-items: center;"> <div style="margin-right: 10px;"> <div style="width: 20px; height: 10px; background-color: red;"></div> </div> <div> Channel Interrupt Enable </div> <div style="margin-left: 20px;"> x – Not used </div> </div>	
<div style="display: flex; align-items: center;"> <div style="margin-right: 10px;"> <div style="width: 20px; height: 10px; background-color: red;"></div> </div> <div> Channel Interrupt Status </div> <div style="margin-left: 20px;"> x – Not used </div> </div>	

Table 13. DCm2_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																
	5																
	6																
	7																

Table 14. DCm2_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		

Table 14. DCm2_res parameter description

Parameter	Format	Description
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 15. DCm2_res State Statistics

State	Max IMB Clock Cycles	RAM Accesses by TPU
INIT	12	5
S1	26	9
S3	16	7

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

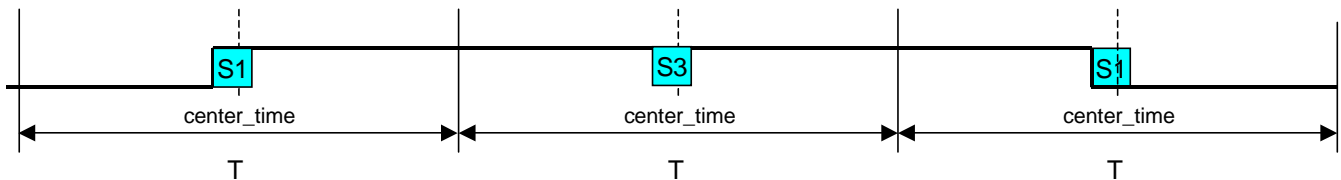


Figure 10. DCm2_res timing

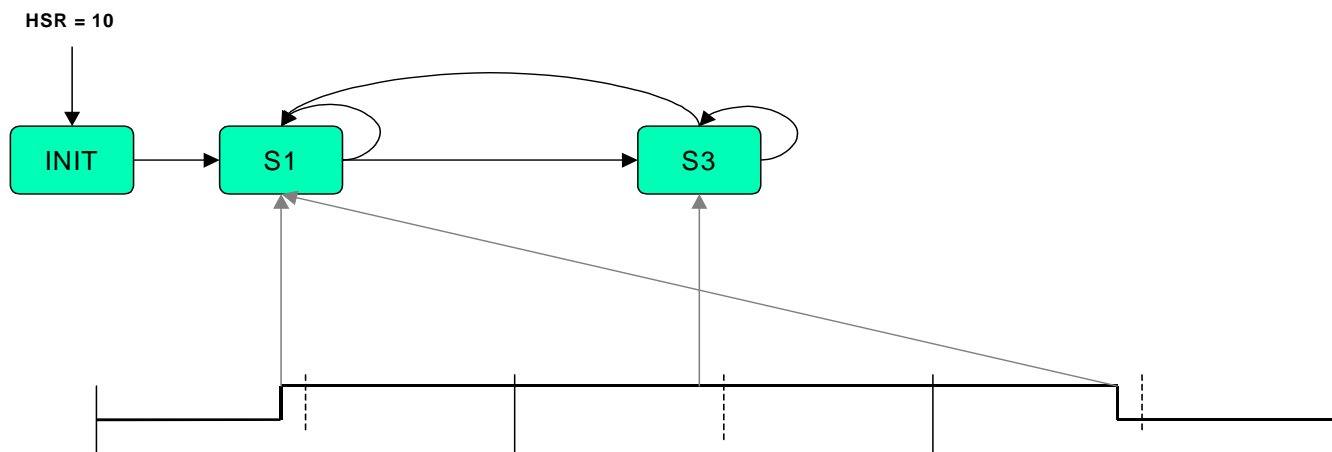


Figure 11. DCm2_res state diagram

Fault Input for DC Motor – 2 outputs version (DCm2_fault)

The DCm2_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 SW1 channel to keep the fault channel parameter space free.

Host Interface

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

Table 16. DCm2_fault Control Bits

Name	Options
<div style="display: flex; justify-content: space-around; font-size: small;"> 3210 </div> <div style="display: flex; align-items: center;"> <div style="border: 1px solid black; width: 20px; height: 15px; margin-right: 5px;"></div> <div style="border: 1px solid black; width: 20px; height: 15px; margin-right: 5px;"></div> <div style="border: 1px solid black; width: 20px; height: 15px; margin-right: 5px;"></div> <div style="border: 1px solid black; width: 20px; height: 15px;"></div> </div>	DCm2_fault function number (Assigned during assembly the DPTRAM code from library TPU functions)
<div style="display: flex; justify-content: space-around; font-size: small;"> 10 </div> <div style="display: flex; align-items: center;"> <div style="border: 1px solid black; width: 20px; height: 15px; margin-right: 5px;"></div> <div style="border: 1px solid black; width: 20px; height: 15px;"></div> </div>	00 – Channel Disabled 01 – Low Priority 10 – Middle Priority 11 – High Priority

Table 16. DCm2_fault 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: 40px;"> 10 </div> <div style="display: flex; justify-content: space-around; width: 40px;"> <div style="border: 1px solid black; width: 15px; height: 15px;"></div> <div style="border: 1px solid black; width: 15px; height: 15px;"></div> </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: 40px;"> 10 </div> <div style="display: flex; justify-content: space-around; width: 40px;"> <div style="background-color: green; border: 1px solid black; width: 15px; height: 15px;"></div> <div style="background-color: green; border: 1px solid black; width: 15px; height: 15px;"></div> </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: center; width: 40px;"> 0 </div> <div style="border: 1px solid black; width: 15px; height: 15px; margin: 0 auto;"></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: center; width: 40px;"> 0 </div> <div style="background-color: blue; border: 1px solid black; width: 15px; height: 15px; margin: 0 auto;"></div> </div> <div> Channel Interrupt Status </div> </div>	0 – Interrupt Not Asserted 1 – Interrupt Asserted

TPU function DCm2_fault generates an interrupt when a high to low transition appears.

Table 17. DCm2_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 18. DCm2_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 19. DCm2_fault State Statistics

State	Max IMB Clock Cycles	RAM Accesses by TPU
INIT	8	2
FAULT	20	1
NO_FAULT	4	1

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

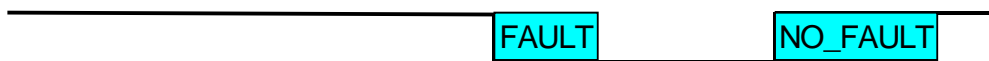


Figure 12. DCm2_fault timing

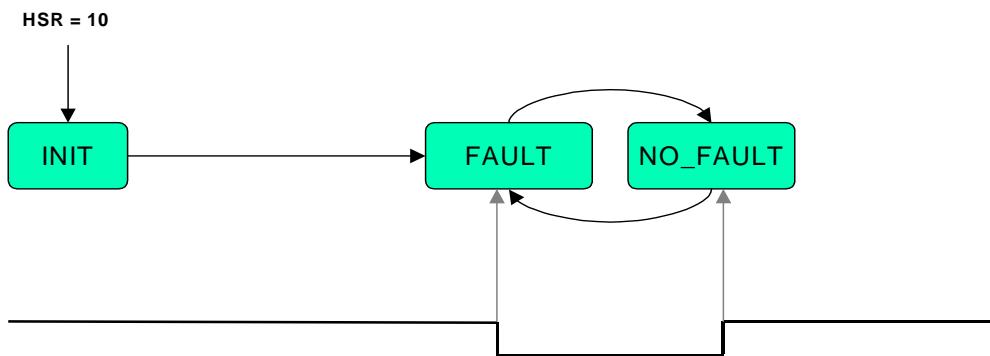


Figure 13. DCm2_fault state diagram

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

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.

