

Modular Microcontroller Family

TPU

TIME PROCESSOR UNIT


REFERENCE
MANUAL



MOTOROLA

OVERVIEW	1
HOST INTERFACE	2
SCHEDULER	3
TPU EMULATION MODE	4
TPU FUNCTIONS	A
REGISTER SUMMARY	B
ESTIMATING WORST-CASE LATENCY	C
CHANNEL HARDWARE DIAGRAM	D
SUMMARY OF CHANGES	S
INDEX	I

TPU TIME PROCESSOR UNIT REFERENCE MANUAL

Motorola reserves the right to make changes without further notice to any products herein. Motorola makes no warranty, representation or guarantee regarding the suitability of its products for any particular purpose, nor does Motorola 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 can and do vary in different applications. All operating parameters, including "Typicals" must be validated for each customer application by customer's technical experts. Motorola does not convey any license under its patent rights nor the rights of others. Motorola 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 Motorola product could create a situation where personal injury or death may occur. Should Buyer purchase or use Motorola products for any such unintended or unauthorized application, Buyer shall indemnify and hold Motorola 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 Motorola was negligent regarding the design or manufacture of the part. Motorola and  are registered trademarks of Motorola, Inc. Motorola, Inc. is an Equal Opportunity/Affirmative Action Employer.

PREFACE

This manual describes the capabilities, operation, and functions of the time processor unit (TPU), an integral module of Motorola's family of modular microcontrollers. Documentation for the Modular Microcontroller Family follows the modular construction of the devices in the product line. Each device has a comprehensive user's manual which provides sufficient information for normal operation of the device. The user's manual is supplemented by module reference manuals, including the TPU reference manual, that provide detailed information about module operation and applications. Refer to Motorola publication *Advanced Microcontroller Unit (AMCU) Literature* (BR1116/D) for a complete listing of documentation.

The following conventions are used throughout the manual.

Logic level one is the voltage that corresponds to Boolean true (1) state.

Logic level zero is the voltage that corresponds to Boolean false (0) state.

To **set** a bit or bits means to establish logic level one on the bit or bits.

To **clear** a bit or bits means to establish logic level zero on the bit or bits.

A signal that is **asserted** is in its active logic state. An active low signal changes from logic level one to logic level zero when asserted, and an active high signal changes from logic level zero to logic level one.

A signal that is **negated** is in its inactive logic state. An active low signal changes from logic level zero to logic level one when negated, and an active high signal changes from logic level one to logic level zero.

LSB means least significant bit or bits. **MSB** means most significant bit or bits. References to low and high bytes are spelled out.

A specific bit or signal within a range is referred to by mnemonic and number. For example, ADDR15 is bit 15 of the address bus. **A range of bits or signals** is referred to by mnemonic and the numbers that define the range. For example, DATA[7:0] form the low byte of the data bus.

TABLE OF CONTENTS

Paragraph	Title	Page
Section 1		
Overview		
1.1	Introduction	1-2
1.2	TPU Components	1-2
1.2.1	Time Bases	1-2
1.2.2	Timer Channels.....	1-2
1.2.3	Host Interface.....	1-2
1.2.4	Parameter RAM.....	1-3
1.2.5	Scheduler.....	1-3
1.2.6	Microengine	1-3
1.3	TPU Features.....	1-3
1.3.1	Emulation Support.....	1-3
1.3.2	Channel Orthogonality.....	1-4
1.3.3	Interchannel Communication	1-4
1.3.4	Coherency.....	1-4
1.4	TPU Memory Map.....	1-4
1.5	Signal Descriptions.....	1-6

Section 2

Host Interface

2.1	System Configuration	2-1
2.1.1	Prescaler Control for TCR1	2-1
2.1.2	Prescaler Control for TCR2.....	2-2
2.1.3	Emulation Control.....	2-3
2.1.4	Low-Power Stop Control.....	2-4
2.1.5	Privilege Levels.....	2-4
2.1.6	TPUMCR Register Diagram	2-4
2.2	Interrupts.....	2-6
2.2.1	Interrupt Levels.....	2-6
2.2.2	Interrupt Arbitration	2-6
2.2.3	Interrupt Vectors	2-7
2.2.4	Enabling Interrupts	2-7
2.2.5	Channel Interrupt Registers	2-8

TABLE OF CONTENTS (Continued)

Paragraph	Title	Page
A.8	Period Measurement with Additional Transition Detection (PMA)	A-14
A.9	Period Measurement with Missing Transition Detection (PMM)	A-16
A.10	Input Capture/Input Transition Counter (ITC)	A-18
A.11	Pulse-Width Modulation (PWM)	A-20
A.12	Discrete Input/Output (DIO)	A-22
A.13	Synchronized Pulse-Width Modulation (SPWM)	A-24
A.14	Quadrature Decode (QDEC)	A-27
A.15	Programmable Time Accumulator (PTA)	A-29
A.16	Queued Output Match TPU Function (QOM)	A-31
A.17	Table Stepper Motor (TSM)	A-33
A.18	Frequency Measurement (FQM)	A-36
A.19	Universal Asynchronous Receiver/Transmitter (UART)	A-38
A.20	New Input Transition Counter (NITC)	A-41
A.21	Multiphase Motor Commutation (COMM)	A-43
A.22	Multichannel Pulse-Width Modulation (MCPWM)	A-45
A.23	Hall Effect Decode (HALLD)	A-52
A.24	Fast Quadrature Decode TPU Function (FQD)	A-54

Appendix B Register Summary

B.1	Register Summary	B-1
B.2	Bit/Field Quick Reference Guide	B-2

Appendix C Estimating Worst-Case Latency

C.1	Introduction to Worst-Case Latency	C-2
C.2	Using Worst-Case Latency Estimates to Evaluate Performance	C-4
C.3	Priority Scheme Details Used in WCL Analysis	C-5
C.3.1	Priority Passing	C-6
C.3.2	Time-Slot Transition	C-6
C.3.3	Channel Number Priority	C-6
C.3.4	RAM Collision Rate	C-7
C.4	First-Pass Worst-Case Latency Analysis	C-8
C.4.1	Worst-Case Assumptions and Formula	C-8
C.4.1.1	Finding the Worst-Case Service Time for Each Active Channel	C-8
C.4.1.2	Mapping the Channels for Each Time Slot	C-9
C.4.1.3	Adding Time for Time-Slot Transitions and NOPs	C-9

LIST OF ILLUSTRATIONS

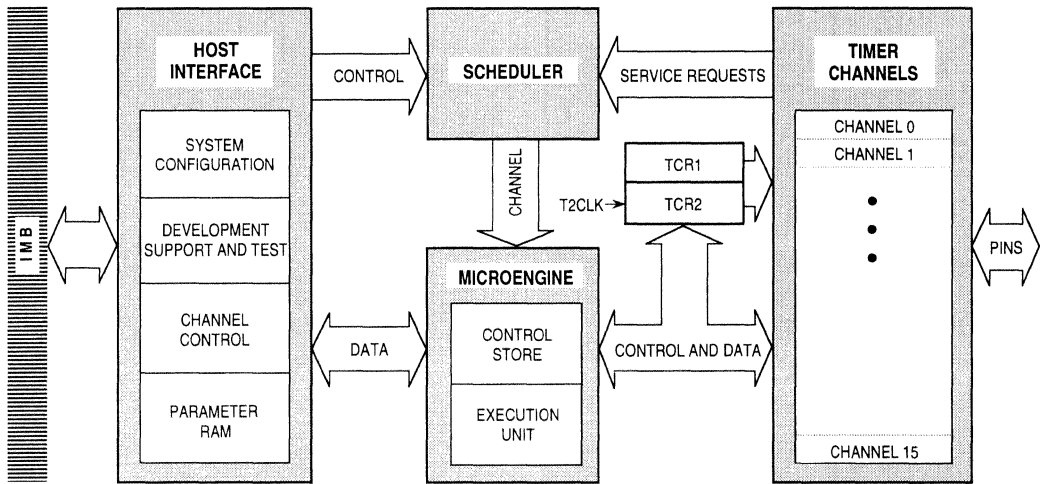
Figure	Title	Page
1-1	TPU Block Diagram	1-1
1-2	TPU Pins.....	1-6
2-1	TCR1 Prescaler Control.....	2-2
2-2	TCR2 Prescaler Control.....	2-2
2-3	RAM Arbitration Timing, Word Access by Host.....	2-14
2-4	RAM Arbitration Timing, Word or Long-Word Access by TPU.....	2-15
2-5	Channel Control and Parameter RAM Configuration	2-17
3-1	Priority Levels.....	3-3
3-2	Priority Passing	3-4
3-3	Time-Slot Variation.....	3-7
4-1	On-Chip RAM Configuration	4-2
4-2	TPU Control Store	4-4
4-3	Extending Microcode	4-5
4-4	1-Kbyte Emulation RAM Memory Map.....	4-6
A-1	PPWA Parameters	A-7
A-2	OC Parameters.....	A-9
A-3	SM Parameters	A-11
A-4	PSP Parameters	A-13
A-5	PMA Parameters.....	A-15
A-6	PMM Parameters	A-17
A-7	ITC Parameters	A-19
A-8	PWM Parameters.....	A-21
A-9	DIO Parameters.....	A-23
A-10	SPWM Parameters.....	A-25
A-11	QDEC Parameters.....	A-28
A-12	PTA Parameters.....	A-30
A-13	QOM Parameters	A-31
A-14	TSM Parameters – Master Mode.....	A-34
A-15	TSM Parameters – Slave Mode.....	A-35
A-16	FQM Parameters.....	A-37
A-17	UART Transmitter Parameters.....	A-39
A-18	UART Receiver Parameters.....	A-40
A-19	NITC Parameters	A-42

LIST OF TABLES

Table	Title	Page
1-1	TPU Address Map.....	1-5
2-1	TCR1 Prescaler Control.....	2-2
2-2	TCR2 Prescaler Control.....	2-3
2-3	Channel Priorities.....	2-11
2-4	Parameter RAM Address Map.....	2-12
3-1	Priority Passing.....	3-3
A-1	Mask Set A Function Encodings.....	A-2
A-2	Mask Set G Function Encodings.....	A-3
A-3	CHANNEL_CONTROL Options.....	A-5
B-1	TPU Register Summary.....	B-1
B-2	Bit/Field Quick Reference Guide.....	B-2
C-1	Longest States and RAM Accesses for Mask Set A Functions.....	C-9

SECTION 1 OVERVIEW

The time processor unit (TPU) is an intelligent, semi-autonomous IMB module designed for timing control. Operating simultaneously with the CPU, the TPU executes microinstructions (microcode) from TPU control ROM. Functions are microcode programs that typically schedule tasks and perform input and output operations. Figure 1-1 is a simplified block diagram of the TPU.



TPU BLOCK

Figure 1-1. TPU Block Diagram

1.2.4 Parameter RAM

Parameter RAM occupies 256 bytes at the top of the TPU module address map. Channel parameters are organized as 128 16-bit words. Only 100 words are actually implemented, however. Channels 0 to 13 use six parameter words, while channels 14 and 15 each use eight parameter words. **1.4 TPU Memory Map** and **2.7.1 Parameter RAM Address Map** show how parameter words are organized in memory.

The parameter RAM is used for communication between the host CPU and the TPU, and for data storage by the TPU. The parameters required by each pre-programmed time function are shown in **APPENDIX A TPU FUNCTIONS** and described in detail in individual programming notes.

1.2.5 Scheduler

Out of reset, all channels are disabled. The host CPU makes a channel active by assigning it one of three priorities: high, middle, or low. The scheduler determines the order in which channels are serviced based on channel number and assigned priority. Refer to **SECTION 3 SCHEDULER** for additional details.

1.2.6 Microengine

The microengine is composed of a control store and an execution unit. Control-store ROM holds the microcode for each factory-masked function. Alternatively, in emulation mode, microcode is executed from the TPURAM module instead of the control store. Emulation mode allows the development of custom TPU functions. Refer to **SECTION 4 TPU EMULATION MODE** for more information.

1.3 TPU Features

Important TPU features are summarized in the following paragraphs.

1.3.1 Emulation Support

To support changing TPU application requirements, Motorola has established a TPU function library. The function library is a collection of TPU functions written for easy assembly in combination with each other or with custom functions. In emulation mode, the TPU uses an on-chip RAM module (TPURAM) as control store instead of the TPU control store ROM. There is no performance loss in emulation mode; functions execute as quickly as they would in TPU control ROM. Notice that in emulation mode, the CPU cannot access the TPURAM. Refer to **SECTION 4 TPU EMULATION MODE** for information on TPU emulation mode. See Motorola Programming Note TPUPN00/D, *Using the TPU Function Library and TPU Emulation Mode*, for details on the TPU function library.

Table 1–1. TPU Address Map

Access	Address	Register
S	\$\$\$E00	TPU MODULE CONFIGURATION REGISTER (TPUMCR)
S	\$\$\$E02	TPU TEST CONFIGURATION REGISTER (TPUTCRCR)
S	\$\$\$E04	DEVELOPMENT SUPPORT CONTROL REGISTER (DSCR)
S	\$\$\$E06	DEVELOPMENT SUPPORT STATUS REGISTER (DSSR)
S	\$\$\$E08	TPU INTERRUPT CONFIGURATION REGISTER (TICR)
S	\$\$\$E0A	CHANNEL INTERRUPT ENABLE REGISTER (CIER)
	\$\$\$E0C	CHANNEL FUNCTION SELECT REGISTER 0 (CFSR0)
S	\$\$\$E0E	CHANNEL FUNCTION SELECT REGISTER 1 (CFSR1)
S	\$\$\$E10	CHANNEL FUNCTION SELECT REGISTER 2 (CFSR2)
S	\$\$\$E12	CHANNEL FUNCTION SELECT REGISTER 3 (CFSR3)
S/U	\$\$\$E14	HOST SEQUENCE REGISTER 0 (HSQR0)
S/U	\$\$\$E16	HOST SEQUENCE REGISTER 1 (HSQR1)
S/U	\$\$\$E18	HOST SERVICE REQUEST REGISTER 0 (HSRR0)
S/U	\$\$\$E1A	HOST SERVICE REQUEST REGISTER 1 (HSRR1)
S	\$\$\$E1C	CHANNEL PRIORITY REGISTER 0 (CPR0)
S	\$\$\$E1E	CHANNEL PRIORITY REGISTER 1 (CPR1)
S	\$\$\$E20	CHANNEL INTERRUPT STATUS REGISTER (CISR)
	\$\$\$E22 – \$\$\$EFF	RESERVED
S/U	\$\$\$F00 – \$\$\$F0E	CHANNEL 0 PARAMETER REGISTERS
S/U	\$\$\$F10 – \$\$\$F1E	CHANNEL 1 PARAMETER REGISTERS
S/U	\$\$\$F20 – \$\$\$F2E	CHANNEL 2 PARAMETER REGISTERS
S/U	\$\$\$F30 – \$\$\$F3E	CHANNEL 3 PARAMETER REGISTERS
S/U	\$\$\$F40 – \$\$\$F4E	CHANNEL 4 PARAMETER REGISTERS
S/U	\$\$\$F50 – \$\$\$F5E	CHANNEL 5 PARAMETER REGISTERS
S/U	\$\$\$F60 – \$\$\$F6E	CHANNEL 6 PARAMETER REGISTERS
S/U	\$\$\$F70 – \$\$\$F7E	CHANNEL 7 PARAMETER REGISTERS
S/U	\$\$\$F80 – \$\$\$F8E	CHANNEL 8 PARAMETER REGISTERS
S/U	\$\$\$F90 – \$\$\$F9E	CHANNEL 9 PARAMETER REGISTERS
S/U	\$\$\$FA0 – \$\$\$FAE	CHANNEL 10 PARAMETER REGISTERS
S/U	\$\$\$FB0 – \$\$\$FBE	CHANNEL 11 PARAMETER REGISTERS

SECTION 2 HOST INTERFACE

The following section describes the host interface to the TPU. The registers described in this section configure the TPU as a whole, the 16 individual channels, and the parameter registers within the RAM used to exchange parameters between the TPU and host CPU.

NOTE

All registers are 16 bits in length and are only accessible through word transfers with one exception: the channel interrupt status register (CISR) can also be accessed on a byte basis. This allows the CPU to perform bit manipulation instructions on the CISR only. A byte write of any TPU register other than the CISR sets the other byte of the word to \$FF.

2.1 System Configuration

The TPU module configuration register (TPUMCR) contains six fields that define module attributes: the interrupt arbitration field, the supervisor/unrestricted control field, the timer count register 2 (TCR2) clock control field, the emulation control field, TCR1 and TCR2 prescaler control fields, and the stop control and status field. The TPUMCR resides in supervisor data space.

2.1.1 Prescaler Control for TCR1

Timer count register one (TCR1) is clocked from the output of a prescaler. Two fields in the TPUMCR control TCR1. The input to the prescaler is the internal TPU system clock divided by either 4 or 32, depending on the value of the PSCK (prescaler clock) bit. The prescaler divides this input by 1, 2, 4, or 8, depending on the value of TCR1P (timer count register 1 prescaler control). TCR1 has the capability to resolve down to the TPU system clock divided by 4. Refer to Figure 2-1 and Table 2-1.

When the T2CG bit is set, the external TCR2 pin functions as a gate of the DIV8 clock (the TPU system clock divided by 8). In this case, when the external TCR2 pin is low, the DIV8 clock is blocked, preventing it from incrementing TCR2. When the external TCR2 pin is high, TCR2 is incremented at the frequency of the DIV8 clock. When the T2CG bit is cleared, an external clock from the TCR2 pin increments TCR2.

The TCR2 pin and each channel configured as an input have an associated synchronizer followed by a digital filter connected to the pin that samples pin transitions. These filter out high and low pulse widths less than the period of two system clocks, preventing these transitions from being input to the transition detect logic. The synchronizer and digital filter are guaranteed to pass pulses that are greater than or equal to the period of four system clocks.

The TCR2 field in TPUMCR specifies the value of the prescaler: 1, 2, 4, or 8. Channels using TCR2 have the capability to resolve down to the TPU system clock divided by 8. Table 2–2 is a summary of prescaler output.

Table 2–2. TCR2 Prescaler Control

TCR2 Prescaler	Divide By	Internal Clock Divided By	External Clock Divided By
00	1	8	1
01	2	16	2
10	4	32	4
11	8	64	8

2.1.3 Emulation Control

Emulation mode is entered by setting the EMU bit in the TPUMCR. In emulation mode, an auxiliary bus connection is made between TPURAM and the TPU module, and access to TPURAM via the intermodule bus is disabled. A 9-bit address bus, a 32-bit data bus, and control lines transfer information between the modules. To ensure exact emulation, TPURAM module access timing remains consistent with access timing of the TPU ROM control store.

The TPU function library is a collection of TPU functions written for easy assembly in combination with each other or with custom functions. TPU emulation capability allows the user to use the TPU library functions and develop new time functions.

Refer to **SECTION 4 TPU EMULATION MODE** for information about emulation mode, and to Motorola Programming Note TPUPN00/D, *Using the TPU Function Library and TPU Emulation Mode*, for details on the TPU function library.

STOP — Stop Bit

- 0 = TPU operating normally
- 1 = Internal clocks shut down

Refer to **2.1.4 Low-Power Stop Control** for more information.

TCR1P — Timer Count Register 1 Prescaler Control

TCR1 is clocked from the output of a prescaler. The input to the prescaler is the internal TPU system clock divided by either 4 or 32, depending on the value of the PSCK bit. The prescaler divides this input by 1, 2, 4, or 8.

- 00 = Divide by 1
- 01 = Divide by 2
- 10 = Divide by 4
- 11 = Divide by 8

Refer to **2.1.1 Prescaler Control for TCR1** for more information.

TCR2P — Timer Count Register 2 Prescaler Control

TCR2 is clocked from the output of a prescaler. The TCR2 field specifies the value of the prescaler: 1, 2, 4, or 8.

- 00 = Divide by 1
- 01 = Divide by 2
- 10 = Divide by 4
- 11 = Divide by 8

Refer to **2.1.2 Prescaler Control for TCR2** for more information.

EMU — Emulation Control

- 0 = TPU and TPURAM in normal mode (operating as separate modules)
- 1 = TPU and TPURAM in emulation mode

After reset, this bit can be written only once. Refer to **2.1.3 Emulation Control** for more information.

T2CG — TCR2 Clock/Gate Control

- 0 = TCR2 pin used as clock source for TCR2
- 1 = TCR2 pin used as gate of DIV8 clock for TCR2

Refer to **2.1.2 Prescaler Control for TCR2** for more information.

STF — Stop Flag

- 0 = TPU operating
- 1 = TPU stopped (STOP bit has been asserted)

Refer to **2.1.4 Low-Power Stop Control** for more information.

SUPV — Supervisor Data Space

- 0 = Assignable registers are accessible from user or supervisor privilege level
- 1 = Assignable registers are accessible from supervisor privilege level only

Refer to **2.1.5 Privilege Levels** for more information.

The IARB field is essentially a second-level priority in case of a tie. Each module that has an interrupt priority level also has an IARB field. Each module must be assigned a unique non-zero IARB value, or operation is undefined when interrupts with the same priority level are issued simultaneously.

IARB fields contain four bits. An IARB value of F is the highest arbitration priority, 1 is the lowest, and 0 disables interrupt arbitration. Whereas zero in the interrupt priority level field disables a module from requesting an interrupt, a zero in the IARB field disables a module from acknowledging its interrupt request. If a module with a non-zero interrupt priority field and a zero IARB field requests an interrupt, the CPU sees a spurious interrupt, because the module requesting the interrupt service never confirms that it made the request. The IARB field for the TPU is located in the TPUMCR.

2.2.3 Interrupt Vectors

The system designer must make sure the CPU knows where to find the service routine for each type of interrupt. The channel interrupt base vector (CIBV) determines where to find the 16 TPU service routines, one for each of the 16 TPU channels. CIBV is the upper nibble of a byte-size vector number; the lower nibble is the channel number itself. CIBV should contain an unreserved vector number. For example, if CIBV contains 8 , the vector number for a channel 0 interrupt is 80 , the vector number for a channel 1 interrupt is 81 , and so on to $8F$ for channel 15. The TPU passes the appropriate vector number to the CPU when it acknowledges its interrupt request.

The system designer must set up the vector table. Refer to the *CPU16 Reference Manual (CPU16RM/AD)* or the *CPU32 Reference Manual (CPU32RM/AD)* for additional information on the vector table.

2.2.4 Enabling Interrupts

The CPU can mask interrupts from individual channels in the channel interrupt enable register (CIER). A zero in a bit field masks the interrupt from a specific channel; a one enables the channel interrupt. If a channel interrupt is masked, the channel interrupt status register (CISR) can be polled to see if an interrupt request is pending. The CISR is the only TPU register that can be accessed by individual byte. This allows CPU32 bit test instructions, which only work on byte accesses, to be used for ease in polling.

To clear a status flag, read the CISR with the bit set, and then write a zero to the appropriate bit.

2.3 Channel Function Select Registers

Each four-bit field within the channel function select registers specifies one of up to 16 time functions to be executed on the corresponding channel. Numbers for predefined functions in both TPU ROM mask sets currently in production are found in Tables A–1 and A–2 in **APPENDIX A TPU FUNCTIONS**. Channel function select registers reside in supervisor data space.

CFSR0 — Channel Function Select Register 0 \$###E0C

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CHANNEL15				CHANNEL14				CHANNEL13				CHANNEL12			
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CFSR1 — Channel Function Select Register 1 \$###E0E

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CHANNEL11				CHANNEL10				CHANNEL9				CHANNEL8			
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CFSR2 — Channel Function Select Register 2 \$###E10

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CHANNEL7				CHANNEL6				CHANNEL5				CHANNEL4			
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CFSR3 — Channel Function Select Register 3 \$###E12

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CHANNEL3				CHANNEL2				CHANNEL1				CHANNEL0			
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CHANNEL[15:0] — Function to Execute on Corresponding Channel

2.4 Host Sequence Registers

The host sequence field helps specify the operation of the time function selected on a given channel. The meaning of the host sequence bits depends on the time function specified. Refer to Tables A–1 and A–2 in **APPENDIX A TPU FUNCTIONS** for a summary of host sequence bits for the predefined functions in the two TPU ROM mask sets currently in production.

The CPU can read or write the host sequence registers, but the TPU can only read them. In systems that support privilege levels, host sequence registers can be assigned to either supervisor or user data space.

2.6 Channel Priority Registers

The channel priority registers (CPR0, CPR1) assign one of three priority levels to a channel or disable the channel. Table 2–3 indicates the priority assignments.

Table 2–3. Channel Priorities

CHX[1:0]	Service
00	Disabled
01	Low
10	Middle
11	High

Access to the channel priority registers may generate a wait state (one clock delay of data transfer acknowledge assertion). The channel priority registers are accessible only from the supervisor privilege level.

It is possible to change the priority level of or disable a channel dynamically. A disabled channel is never scheduled to be serviced. Service requests that are pending before a channel is disabled or occur while a channel is disabled remain asserted until the channel is serviced. It is recommended to configure a host service request for initialization of a channel before that channel is enabled to active priority.

Refer to **SECTION 3 SCHEDULER** for additional information on channel priorities.

CPR0 — Channel Priority Register 0

####E1C

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CH 15	CH 14	CH 13	CH 12	CH 11	CH 10	CH 9	CH 8								
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CPR1 — Channel Priority Register 1

####E1E

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CH 7	CH 6	CH 5	CH 4	CH 3	CH 2	CH 1	CH 0								
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CH[15:0] — Priority Level

2.7.2 Parameter RAM Accesses

The TPU shares parameter RAM with the CPU. Parameter RAM may be accessed by only one source at a time; an arbitration scheme prevents simultaneous accesses, allowing eventual access to all requesting sources. Long-word CPU accesses are coherent. The following rules regulate parameter RAM access.

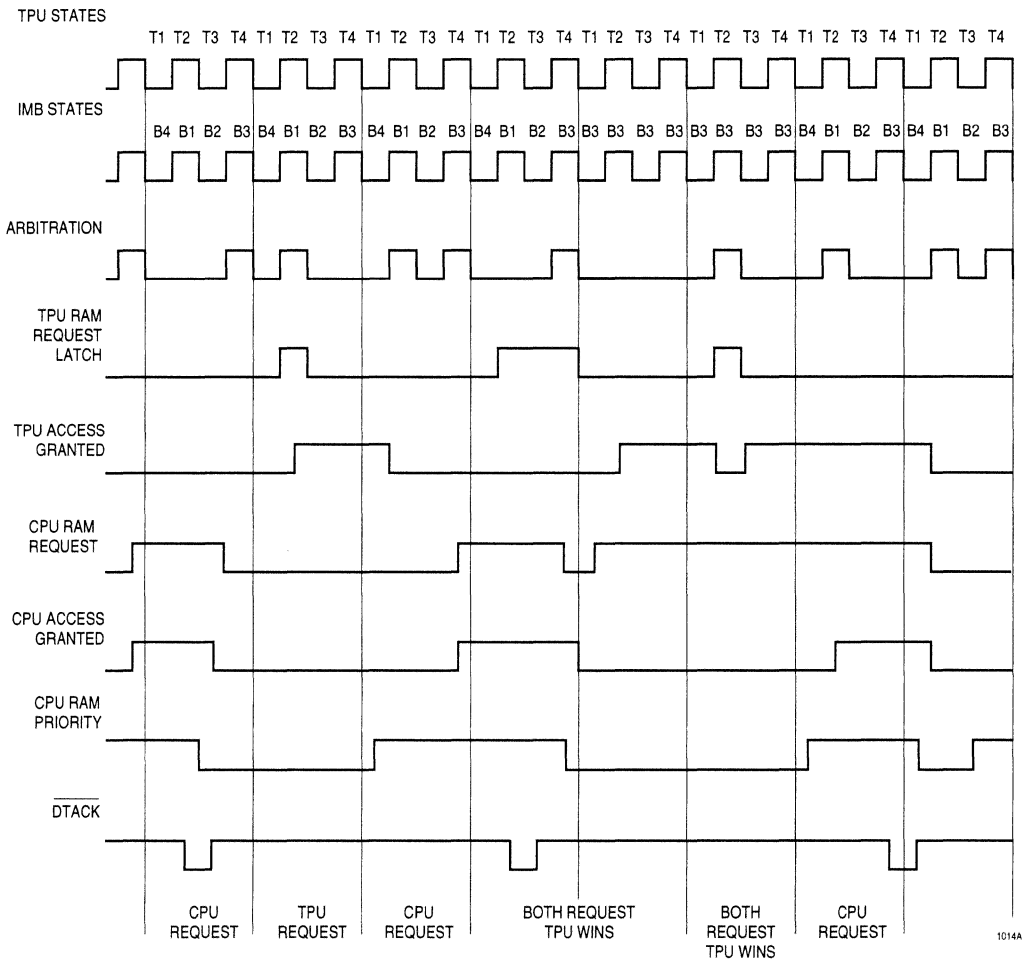
The TPU gives priority to the CPU for parameter RAM accesses under any of the following conditions:

- The TPU has completed accessing the second word of a long-word parameter RAM access.
- The parameter RAM was not accessed during the last arbitration period.
- The CPU is arbitrating for the second word access of a long-word transfer.

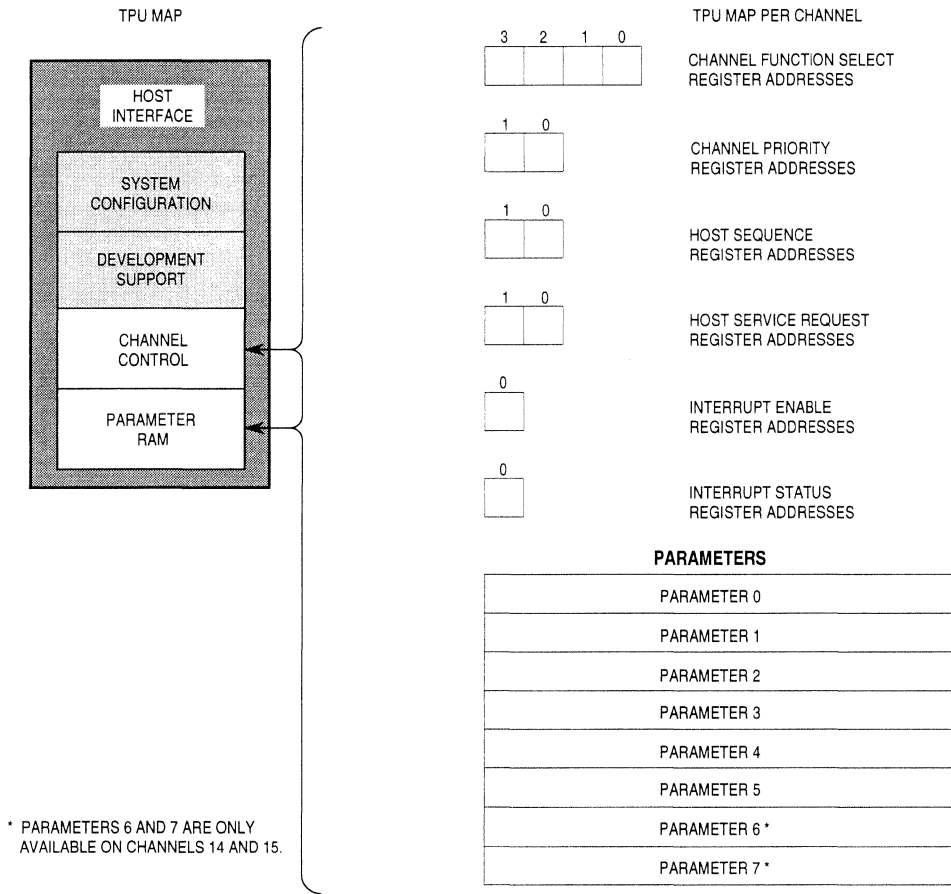
The TPU takes priority for parameter RAM accesses under either of the following conditions:

- The CPU has completed a data transfer during the last access; or
- The TPU is arbitrating for the second access of a data transfer. (A data transfer is defined as word or long-word access.) All even multiples of back-to-back word accesses are coherent.

Figures 2–3 and 2–4 illustrate word accesses by a host, such as a CPU, and word or long-word accesses by the TPU.



**Figure 2-4. Parameter RAM Arbitration Timing,
Word or Long-Word Access by TPU**



1016A

Figure 2-5. Channel Control and Parameter RAM Configuration

```
** Make sure channels 1 and 0 have been initialized before continuing.
(This may not be necessary.) **
```

```
wait:
move.w    hsr1, d0                ;check host service bits for ch. 1 & 0
ori.b    #$0F, d0                ;if host service request bits = 00
                                           ;then channel has been serviced

bne      wait
```

2.9.2 CPU16 Configuration Example

The following code initializes the TPU to run the input transition counter (ITC) function on channel 1 and the pulse-width modulation (PWM) function on channel 0.

```
** Initialize the TPU module configuration register and TPU interrupt
control register to set up for interrupts from TPU and to set up for a
fast clock. **
```

```
ldd      #$004E
std      tpumcr                  ;prescale TCR1 by 4, set IARB to $E
ldd      #$0640
std      ticr                    ;tpu interrupt level = 6, vectors $4X
```

```
** Enable interrupts on channels 0 and 1 only by setting corresponding
bits in channel interrupt enable register (CIER) **
```

```
ldd      #$0003
std      cier                    ;enable interrupts for ch. 0 and 1
```

```
** Choose ITC for channel 1 and PWM for channel 0 by writing to the
channel function select register (CFSR) **
```

```
ldd      #$00A9
std      cfsr3                  ;ITC ($A) to ch. 1, PWM ($9) to ch. 0
```

```
** Choose options for channel 1 by writing to the host sequence
register. Set up parameters for channel 1. **
```

```
ldd      #$0004
std      hsqr1                  ;no link, cont. mode (%01) to ch. 1
ldd      #$0007
std      chlpar0                ;capture TCR1 on rising edges
ldd      #$000E
std      chlpar1                ;bank address pointed to nonexistent addr.
ldd      #$000A
std      chlpar2                ;max count = $A for ch. 1
```

```
** Set up parameters for channel 0 **
```

```
ldd      #$0091
std      ch0par0                ;match TCR1, initialize pin high
ldd      #$1000
std      ch0par2                ;high time = $1000 TCR1 tics
ldd      #$2000
```

SECTION 3 SCHEDULER

Every function is composed of one or more states. A state is constructed of a specific number of microinstructions that cannot be interrupted when executed by the microengine. The intent of every channel is to receive time for state execution (to be serviced). Since one microengine handles up to 16 functions operating concurrently, the function states must be executed serially. The task of the scheduler is to recognize and prioritize the channels needing service and to grant each channel state execution time. The time given to an individual state for execution or service is called a time slot. The duration of a time slot is determined by the number of microinstructions the state contains and, therefore, varies in length.

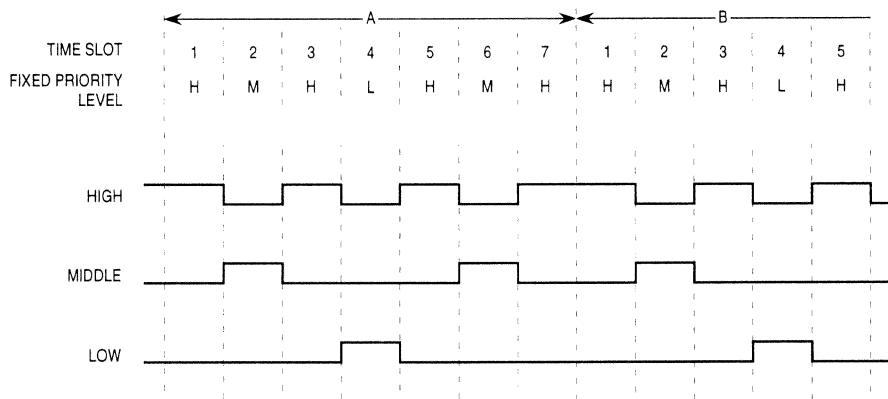
At any time, an arbitrary number of channels can require service by the microengine. To request service, a channel notifies the scheduler by issuing a service request. A service request, which is any occurrence that asserts the service request latch, has four origins:

1. Match Recognition Service Request
2. Transition Detect Service Request
3. Channel Linking Service Request
4. Host Service Request

Once the scheduler grants a channel a time slot, the service grant latch for that channel is asserted, disabling the service request latch. As a result, the channel may request new service but is not serviced again until all other requesting channels have been serviced. The service grant latch then notifies the scheduler that the channel has been granted a time slot. Likewise, while this latch is asserted, the channel is not granted another time slot for new service.

3.1 Priority Scheme

In order to organize incoming requests and ensure that no channel permanently blocks another channel from receiving a time slot, the scheduler requires a priority scheme. Every channel is assigned one of three priority levels: high, middle, or low. Channel priority assignment is discussed in **2.6 Channel**



1053A

Figure 3-1. Priority Levels

3.1.1 Primary Scheme — Priority Among Channels on Different Levels

Although time priority is fixed, the servicing priority is not. The primary scheme acknowledges the priority level assigned to a time slot, granting service first to a channel having the same priority. In Figure 3-1, time slot one has a high-level assignment; therefore, any high-level channel requesting service is recognized first. However, if no high-level channel requests service, the scheduler recognizes a requesting middle-level channel. If this level has no request, the scheduler continues to the low level. If no requests occur, the scheduler remains in the time slot waiting for any channel to request service. Granting service to a different-level channel is called priority passing. The order of passing, which always gives second priority to a high-level channel, is shown in Table 3-1.

Table 3-1. Priority Passing

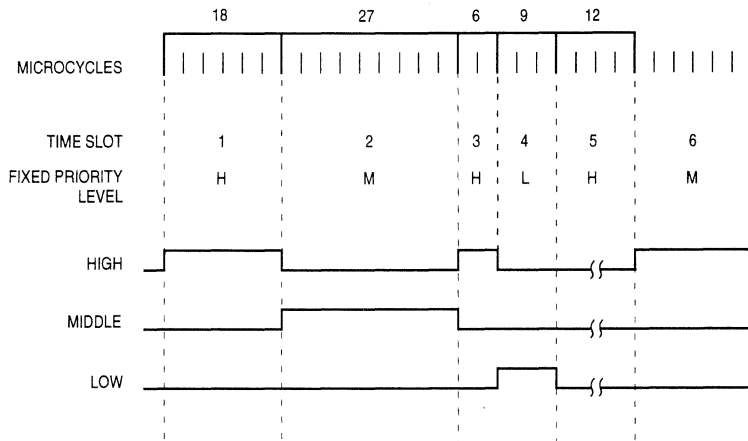
Assigned Priority Level		Next Priority Level		Next Priority Level
High	→	Middle	→	Low
Middle	→	High	→	Low
Low	→	High	→	Middle

the three priority levels, beginning with the lowest numbered channel on that level. It services all requesting same-level channels before clearing any of them for new service.

3.1.3 Correlation of Primary and Secondary Schemes

The overall priority scheme simultaneously incorporates both primary and secondary schemes. Combining both schemes in the following example conveys their correlation.

1. Having its service request latch asserted, a single high-level channel requires service and is granted time slot one, which has high-level priority (primary scheme). Once serviced, the channel's service grant latch is asserted. Next, the service grant and service request latches are negated, and a two-microcycle delay is introduced.
2. The scheduler proceeds to time slot two, which has middle-level priority; however, no middle-level channel is requesting service. Priority is passed to the high level, but no high-level channel is requesting service; therefore, priority is passed again, and service is granted to the single requesting low-level channel. Once scheduled, this channel's service latches are negated, and a two-microcycle delay is introduced.
3. The scheduler resumes with the fixed-priority sequence on time slot three; however, no channels are requesting service. A two-microcycle delay is introduced, and the scheduler remains at time slot three inserting two-microcycle delays while waiting for requests.
4. Three high-level channels simultaneously request service (secondary scheme). The scheduler finds the lowest numbered high-level channel and assigns it to time slot three, which has high-level priority. This channel's service grant latch is asserted; however, the two remaining high-level channels have asserted service request latches.
5. The scheduler continues to time slot four, which has low priority, and allocates the slot to the lowest numbered low-level channel requesting service (primary scheme). The scheduler notes the still unserved low-level channels and proceeds to time slot five (secondary scheme resumes).
6. The next lowest numbered high-level channel is assigned to time slot five, which has high priority. Noting the one remaining high-level channel, the scheduler continues to time slot six.



NOTE: THE MICROCYCLE FIGURES ARE ARBITRARY EXAMPLES.

1055A

Figure 3-3. Time-Slot Variation

If the maximum time slot for each channel and the number of RAM accesses by the host CPU are known, the user can determine the maximum latency that will occur for each channel. Procedures for estimating worst-case latency are given in **APPENDIX C ESTIMATING WORST-CASE LATENCY**.

3.3 Disabling a Function

The CPU disables the function operating on a given channel by clearing the channel priority bits to zero. When the CPU disables a function, if the function is currently being serviced, servicing of the function will complete. This means that it is possible for the output level of a channel pin to change even after the priority bits are cleared. For instance, if an output transition is scheduled, the transition will occur even after the channel is disabled.

SECTION 4 TPU EMULATION MODE

In emulation mode, the TPU uses the on-chip TPURAM, normally used by the host CPU, for the control store. (Refer to Figure 4–1). Emulation mode gives the user flexibility in selecting a TPU function set. A user can write his or her own functions, download the standard mask set and make changes, or select any combination of functions from the TPU function library (see **4.3 TPU Function Library**). Any combination of library functions and custom functions can be assembled together and downloaded to the TPURAM, provided the combined size of the functions does not exceed the limit placed on the TPU.

This section provides an overview of TPU emulation mode. For a complete discussion of TPU emulation mode and the TPU function library, refer to Motorola Programming Note TPUPN00/D, *Using the TPU Function Library and TPU Emulation Mode*.

segment is contiguous with the normal microcode segment. This is the most efficient use of the available space. Figure 4–3 shows unused entry points being used for additional opcode space.

All operations, timing, and conditions that apply to the ROM control store also apply to the RAM when used for emulation.

4.2 Emulation Mode Memory Map

When the TPU enters emulation mode, the TPURAM is dedicated to the TPU and replaces the control store ROM. Most microcontrollers currently available with a TPU have a full two Kbytes of TPURAM, which allows complete emulation of the control store. Figure 4–2 shows the equivalent host CPU byte addresses that are used to load TPURAM with TPU microcode before invoking emulation mode.

Some Motorola MCUs contain only one Kbyte of TPURAM. With these devices, it is only possible to emulate half the TPU control store. Figure 4–4 shows emulation memory map and equivalent TPURAM addresses for these devices.

Some Motorola microcontrollers contain TPURAM modules that are larger than the TPU microcode control store. In these devices, only a portion of the TPURAM will be used for TPU emulation, but the entire TPURAM will be removed from the CPU memory map during emulation. The emulation mode memory map will never be larger than the control store map.

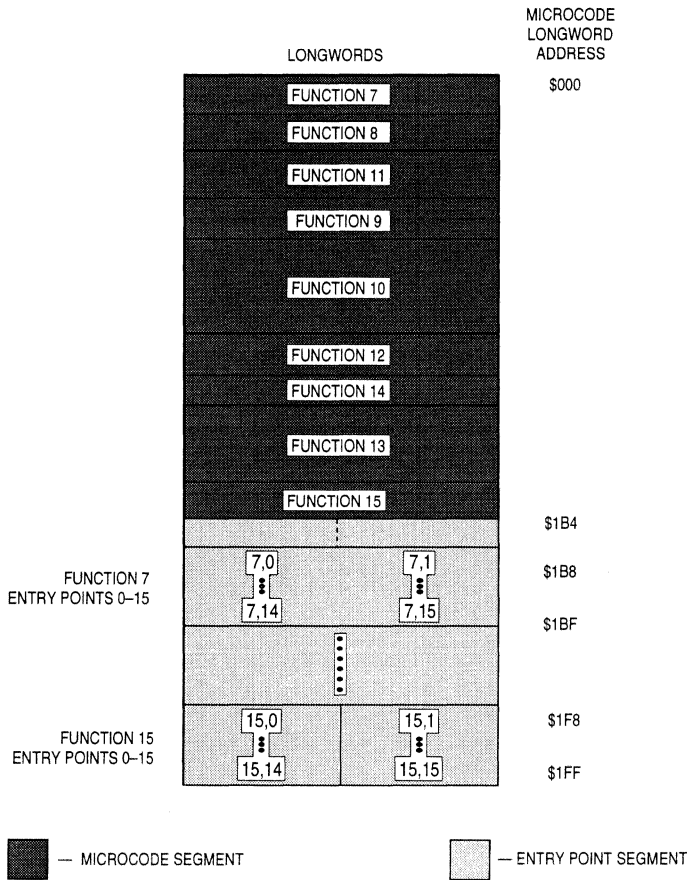


Figure 4-3. Extending Microcode Segment by Means of Unused Entry Points

F3

4.3 TPU Function Library

To support changing TPU application requirements and to allow inclusion of customer-defined TPU functions, Motorola has established a TPU function library. The function library is a collection of TPU functions written for easy assembly in combination with each other or with custom functions. The library currently includes all functions from the standard microcode ROMs and several other functions. Refer to Motorola Programming Note TPUPN00/D, *Using the TPU Function Library and TPU Emulation Mode*, for details on using the function code library and for a list of functions currently available. Other functions are being developed — the list of functions will continue to grow as Motorola responds to requests for new features.

Source code for functions, which is required to combine them into a new set, is available from Motorola (refer to Motorola Programming Note TPUPN00/D, *Using the TPU Function Library and TPU Emulation Mode* for details) or your local Motorola technical representative.

4.4 Emulation Mode Summary

Motorola Programming Note TPUPN00/D, *Using the TPU Function Library and TPU Emulation Mode*, provides detailed instructions on installing and running functions from the function library. Following is a summary of the steps required:

1. Select the desired set of functions and determine that total code size is within the limit (512 long words).
2. Assemble the source code for the desired functions to produce executable code.
3. Configure the on-chip emulation TPURAM by writing to the TPURAM base address register.
4. Load the code into TPURAM.
5. Put the TPU into emulation mode by setting the EMU bit in the TPU module configuration register.

While the TPU is in emulation mode, the TPURAM module is removed from the CPU memory map, and the vacated address space may be allocated to other internal or external peripheral modules.

Once procedures for loading and configuring the TPU for emulation mode operation are completed, the TPU will run the set of newly installed functions as though they were contained in the control store ROM. To run the functions, the CPU must set up control registers and parameter RAM as explained in the documentation for each function. In emulation mode, the functions in microcode ROM are not available to the TPU.

APPENDIX A TPU FUNCTIONS

The following pages provide brief descriptions of the pre-programmed functions in the two TPU mask sets currently in use. For detailed descriptions, refer to the programming note for the individual function. Motorola Programming Note TPUPN00/D, *Using the TPU Function Library and TPU Emulation Mode*, provides a list of available programming notes.

A.1 Mask Set A

Table A-1 gives the function code and lists options for the host sequence bits and the host service request bits for each function in mask set A.

A

A.2 Mask Set G

Table A–2 gives the function code and lists options for the host sequence bits and the HSR bits for each function in mask set G.

Table A–2. Mask Set G Time Function Encodings

Function Name	Function Code	Host Service Request Code	Host Sequence Code*
PTA Programmable Time Accumulator	\$F	0 = No host service 1 = No effect 2 = No effect 3 = Initialize function	0 = High time accumulate 1 = Low time accumulate 2 = Period accumulate – rising 3 = Period accumulate – falling
Queued Output Match (QOM)	\$E	0 = No Host Service 1 = Initialize, No Pin Change 2 = Initialize, Pin Low 3 = Initialize, Pin High	0 = Single-shot mode 1 = Loop Mode 2 = Continuous Mode 3 = Continuous Mode
TSM Table Stepper Motor	\$D	0 = No Host Service 1 = Initialize, Pin Low 2 = Initialize, Pin High 3 = Move Request (Master Only)	0 = Rotate PIN_SEQUENCE once between steps, local mode acceleration table 1 = Rotate PIN_SEQUENCE once between steps, split mode acceleration table 2 = Rotate PIN_SEQUENCE twice between steps, local mode acceleration table 3 = Rotate PIN_SEQUENCE twice between steps, split mode acceleration table
FQM Frequency Measurement	\$C	0 = No Host Service 1 = Undefined 2 = Initialize 3 = Undefined	0 = Begin with Falling Edge –Single- Shot Mode 1 = Begin with Falling Edge – Continuous Mode 2 = Begin with Rising Edge – Single- Shot Mode 3 = Begin with Rising Edge – Continuous Mode
UART Asynchronous Receiver/Transmitter	\$B	0 = No Host Service 1 = Not used 2 = Receive 3 = Transmit	0 = No Parity 1 = No Parity 2 = Even Parity 3 = Odd Parity
NITC New Input Transition Counter	\$A	0 = No Host Service 1 = Initialize TCR Mode 2 = Initialize Parameter Mode 3 = Not Used	0 = Single Shot, No Links 1 = Continual, No Links 2 = Single Shot, Links 3 = Continual, Links
COMM Multiphase Motor Commutation	\$9	0 = No host service request 1 = Not used 2 = Initialize or force state 3 = Initialize or force immediate state test	0 = Sensorless match update mode 1 = Sensorless match update mode 2 = Sensorless link update mode 3 = Sensored mode
HALLD	\$8	0 = No host service 1 = Not used 2 = Initialize – two channel mode 3 = Initialize – three channel mode	0 = Channel A 1 = Channel B 2 = Channel B 3 = Channel C (3-channel mode only)

Table A-3. CHANNEL_CONTROL Options

TBS				PAC			PSC		Action	
8	7	6	5	4	3	2	1	0	Input	Output
							0	0	—	Force Pin as Specified by PAC Latches Force Pin High Force Pin Low Do Not Force Any State
							0	1	—	
							1	0	—	
							1	1	—	
				0	0	0			Do Not Detect Transition	Do Not Change Pin State on Match
				0	0	1			Detect Rising Edge	High on Match
				0	1	0			Detect Falling Edge	Low on Match
				0	1	1			Detect Either Edge	Toggle on Match
				1	x	x			Do Not Change PAC	Do Not Change PAC
0	0	x	x						Input Channel	
0	0	0	0						Capture TCR1, Match TCR1	—
0	0	0	1						Capture TCR1, Match TCR2	—
0	0	1	0						Capture TCR2, Match TCR1	—
0	0	1	1						Capture TCR2, Match TCR2	—
0	1	x	x							Output Channel
0	1	0	0						—	Capture TCR1, Match TCR1
0	1	0	1						—	Capture TCR2, Match TCR2
0	1	1	0						—	Capture TCR2, Match TCR1
0	1	1	1						—	Capture TCR2, Match TCR2
1	x	x	x						Do Not Change TBS	Do Not Change TBS

A

CONTROL BITS

NAME	OPTIONS	ADDRESSES	
<div style="display: flex; justify-content: space-around; width: 100px;"> 3210 </div> <div style="display: flex; justify-content: space-around; width: 100px;"> <div style="border: 1px solid black; width: 20px; height: 20px;"></div> <div style="border: 1px solid black; width: 20px; height: 20px;"></div> <div style="border: 1px solid black; width: 20px; height: 20px;"></div> <div style="border: 1px solid black; width: 20px; height: 20px;"></div> </div>	CHANNEL FUNCTION SELECT	PPWA FUNCTION NUMBER (ASSIGNED DURING MICRO-CODE ASSEMBLY)	\$\$\$E0C-\$\$\$E12
<div style="display: flex; justify-content: space-around; width: 100px;"> 10 </div> <div style="display: flex; justify-content: space-around; width: 100px;"> <div style="border: 1px solid black; width: 20px; height: 20px;"></div> <div style="border: 1px solid black; width: 20px; height: 20px;"></div> </div>	CHANNEL PRIORITY	00 — CHANNEL DISABLED 01 — LOW PRIORITY 10 — MEDIUM PRIORITY 11 — HIGH PRIORITY	\$\$\$E1C-\$\$\$E1E
<div style="display: flex; justify-content: space-around; width: 100px;"> 10 </div> <div style="display: flex; justify-content: space-around; width: 100px;"> <div style="border: 1px solid black; width: 20px; height: 20px;"></div> <div style="border: 1px solid black; width: 20px; height: 20px;"></div> </div>	HOST SEQUENCE BITS	00 — ACCUMULATE 24-BIT PERIODS, NO LINKS 01 — ACCUMULATE 16-BIT PERIODS, LINKS 10 — ACCUMULATE 24-BIT PULSE WIDTHS, NO LINKS 11 — ACCUMULATE 16-BIT PULSE WIDTHS, LINKS	\$\$\$E14-\$\$\$E16
<div style="display: flex; justify-content: space-around; width: 100px;"> 10 </div> <div style="display: flex; justify-content: space-around; width: 100px;"> <div style="border: 1px solid black; width: 20px; height: 20px;"></div> <div style="border: 1px solid black; width: 20px; height: 20px;"></div> </div>	HOST SERVICE BITS	00 — NOT USED 01 — NOT USED 10 — INITIALIZE 11 — NOT USED	\$\$\$E18-\$\$\$E1A
<div style="display: flex; justify-content: space-around; width: 100px;"> 0 </div> <div style="display: flex; justify-content: space-around; width: 100px;"> <div style="border: 1px solid black; width: 20px; height: 20px;"></div> </div>	INTERRUPT ENABLE	0 — INTERRUPT NOT ASSERTED 1 — INTERRUPT ASSERTED	\$\$\$E0A
<div style="display: flex; justify-content: space-around; width: 100px;"> 0 </div> <div style="display: flex; justify-content: space-around; width: 100px;"> <div style="background-color: #cccccc; width: 20px; height: 20px;"></div> </div>	INTERRUPT STATUS		\$\$\$E20

A

PARAMETER RAM

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
\$\$\$FW0	START_LINK_CHANNEL				LINK_CHANNEL_COUNT				CHANNEL_CONTROL							
\$\$\$FW2	MAX_COUNT								PERIOD_COUNT							
\$\$\$FW4	LAST_ACCUM															
\$\$\$FW6	ACCUM															
\$\$\$FW8	ACCUM_RATE								PPWA_UB							
\$\$\$FWA	PPWA_LW															
\$\$\$FWC	[Hatched Pattern]															
\$\$\$FWE	[Hatched Pattern]															

<div style="display: flex; justify-content: space-around;"> <div style="border: 1px solid black; width: 20px; height: 20px;"></div> = WRITTEN BY CPU </div>	<div style="display: flex; justify-content: space-around;"> <div style="background-color: black; width: 20px; height: 20px;"></div> = WRITTEN BY CPU AND TPU </div>
<div style="display: flex; justify-content: space-around;"> <div style="background-color: #cccccc; width: 20px; height: 20px;"></div> = WRITTEN BY TPU </div>	<div style="display: flex; justify-content: space-around;"> <div style="background: repeating-linear-gradient(45deg, transparent, transparent 2px, black 2px, black 4px); width: 20px; height: 20px;"></div> = UNUSED PARAMETERS </div>

W = CHANNEL NUMBER

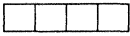





NOTES:

1. THE TPU DOES NOT CHECK THE VALUE OF LINK_CHANNEL_COUNT. IF THIS PARAMETER IS NOT > 0 AND ≤ 8, RESULTS ARE UNPREDICTABLE.
2. MAX_COUNT MAY BE WRITTEN AT ANY TIME BY THE HOST CPU, BUT IF THE VALUE WRITTEN IS ≤ PERIOD_COUNT, A PERIOD OR PULSE-WIDTH ACCUMULATION IS TERMINATED. IF THIS HAPPENS, THE NUMBER OF PERIODS OVER WHICH THE ACCUMULATION IS DONE WILL NOT CORRESPOND TO MAX_COUNT.

1049A





Figure A-1. PPWA Parameters

CONTROL BITS

	NAME	OPTIONS	ADDRESSES
3 2 1 0 	CHANNEL FUNCTION SELECT	OC FUNCTION NUMBER (ASSIGNED DURING MICRO-CODE ASSEMBLY)	###E0C-###E12
1 0 	CHANNEL PRIORITY	00 — CHANNEL DISABLED 01 — LOW PRIORITY 10 — MEDIUM PRIORITY 11 — HIGH PRIORITY	###E1C-###E1E
1 0 	HOST SEQUENCE BITS	0x — MATCHES AND PULSES SCHEDULED 1x — ONLY READ TCR1, TCR2	###E14-###E16
1 0 	HOST SERVICE BITS	00 — NO HOST SERVICE REQUEST 01 — HOST-INITIATED PULSE 10 — NOT USED 11 — INITIALIZE, CONTINUOUS PULSES	###E18-###E1A
0 	INTERRUPT ENABLE	0 — INTERRUPT NOT ASSERTED 1 — INTERRUPT ASSERTED	###E0A
0 	INTERRUPT STATUS		###E20

PARAMETER RAM

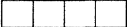





	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
###FW0	UNUSED								CHANNEL_CONTROL									
###FW2	OFFSET																	
###FW4	RATIO								REF_ADDR1								0	
###FW6	REF_ADDR2								0	REF_ADDR3								0
###FW8	REF_TIME																	
###FWA	ACTUAL_MATCH_TIME																	
###FEC	TCR1																	
###FEE	TCR2																	

 = WRITTEN BY CPU	 = WRITTEN BY CPU AND TPU
 = WRITTEN BY TPU	 = UNUSED PARAMETERS
W = CHANNEL NUMBER	

1024A

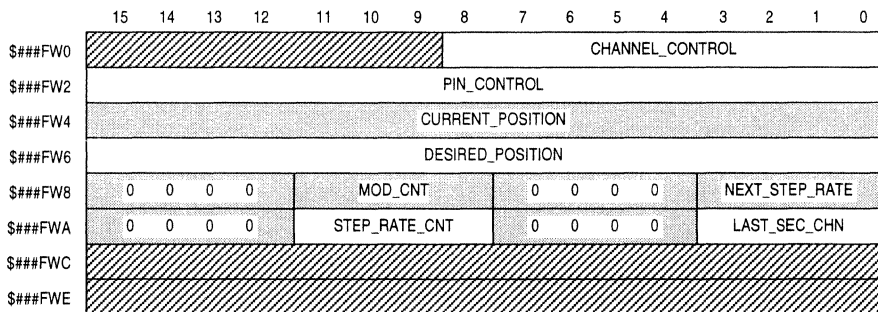
Figure A-2. OC Parameters





CONTROL BITS

	NAME	OPTIONS	ADDRESSES
3 2 1 0 	CHANNEL FUNCTION SELECT	SM FUNCTION NUMBER (ASSIGNED DURING MICRO-CODE ASSEMBLY)	\$\$\$E0C-\$\$\$E12
1 0 	CHANNEL PRIORITY	00 — CHANNEL DISABLED 01 — LOW PRIORITY 10 — MEDIUM PRIORITY 11 — HIGH PRIORITY	\$\$\$E1C-\$\$\$E1E
1 0 	HOST SEQUENCE BITS	xx — NOT USED	\$\$\$E14-\$\$\$E16
1 0 	HOST SERVICE BITS	00 — NO HOST SERVICE REQUEST 01 — NOT USED 10 — INITIALIZE 11 — STEP REQUEST (PRIMARY CHANNEL ONLY)	\$\$\$E18-\$\$\$E1A
0 	INTERRUPT ENABLE	0 — INTERRUPT NOT ASSERTED 1 — INTERRUPT ASSERTED	\$\$\$E0A
0 	INTERRUPT STATUS		\$\$\$E20



PARAMETER RAM (PRIMARY CHANNEL)



-  = WRITTEN BY CPU
-  = WRITTEN BY CPU AND TPU
-  = WRITTEN BY TPU
-  = UNUSED PARAMETERS
- W = CHANNEL NUMBER

1046A-1

Figure A-3. SM Parameters

CONTROL BITS

NAME	OPTIONS	ADDRESSES	
<div style="display: flex; justify-content: space-around; width: 100px;"> 3210 </div> <div style="display: flex; justify-content: space-around; width: 100px;"> <div style="border: 1px solid black; width: 20px; height: 20px;"></div> <div style="border: 1px solid black; width: 20px; height: 20px;"></div> <div style="border: 1px solid black; width: 20px; height: 20px;"></div> <div style="border: 1px solid black; width: 20px; height: 20px;"></div> </div>	CHANNEL FUNCTION SELECT	PSP FUNCTION NUMBER (ASSIGNED DURING MICRO-CODE ASSEMBLY)	###E0C-###E12
<div style="display: flex; justify-content: space-around; width: 100px;"> 10 </div> <div style="display: flex; justify-content: space-around; width: 100px;"> <div style="border: 1px solid black; width: 20px; height: 20px;"></div> <div style="border: 1px solid black; width: 20px; height: 20px;"></div> </div>	CHANNEL PRIORITY	00 — CHANNEL DISABLED 01 — LOW PRIORITY 10 — MEDIUM PRIORITY 11 — HIGH PRIORITY	###E1C-###E1E
<div style="display: flex; justify-content: space-around; width: 100px;"> 10 </div> <div style="display: flex; justify-content: space-around; width: 100px;"> <div style="border: 1px solid black; width: 20px; height: 20px;"></div> <div style="border: 1px solid black; width: 20px; height: 20px;"></div> </div>	HOST SEQUENCE BITS	x0 — ANGLE-ANGLE MODE x1 — ANGLE-TIME MODE	###E14-###E16
<div style="display: flex; justify-content: space-around; width: 100px;"> 10 </div> <div style="display: flex; justify-content: space-around; width: 100px;"> <div style="border: 1px solid black; width: 20px; height: 20px;"></div> <div style="border: 1px solid black; width: 20px; height: 20px;"></div> </div>	HOST SERVICE BITS	00 — NO HOST SERVICE REQUEST 01 — IMMEDIATE UPDATE 10 — INITIALIZE 11 — FORCE PIN	###E18-###E1A
<div style="display: flex; justify-content: space-around; width: 100px;"> 0 </div> <div style="display: flex; justify-content: space-around; width: 100px;"> <div style="border: 1px solid black; width: 20px; height: 20px;"></div> </div>	INTERRUPT ENABLE	0 — INTERRUPT NOT ASSERTED 1 — INTERRUPT ASSERTED	###E0A
<div style="display: flex; justify-content: space-around; width: 100px;"> 0 </div> <div style="display: flex; justify-content: space-around; width: 100px;"> <div style="border: 1px solid black; width: 20px; height: 20px; background-color: #cccccc;"></div> </div>	INTERRUPT STATUS		###E20

A

PARAMETER RAM

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
###FW0	PERIOD_ADDRESS								CHANNEL_CONTROL							
###FW2	R2_A2_TEMP															
###FW4	ANGLE_TIME															
###FW6	RATIO_TEMP															
###FW8	RATIO1 (1)								ANGLE1 (1)							
###FWA	RATIO2 (1, 2)						HIGH_TIME (3)			ANGLE2 (1, 2)						
###FWC																
###FWE																

<div style="display: flex; justify-content: space-around;"> <div style="border: 1px solid black; width: 20px; height: 20px;"></div> = WRITTEN BY CPU </div>	<div style="display: flex; justify-content: space-around;"> <div style="background-color: black; width: 20px; height: 20px;"></div> = WRITTEN BY CPU AND TPU </div>
<div style="display: flex; justify-content: space-around;"> <div style="background-color: #cccccc; width: 20px; height: 20px;"></div> = WRITTEN BY TPU </div>	<div style="display: flex; justify-content: space-around;"> <div style="background: repeating-linear-gradient(45deg, transparent, transparent 2px, black 2px, black 4px); width: 20px; height: 20px;"></div> = UNUSED PARAMETERS </div>

W = CHANNEL NUMBER

NOTES:

1. RESTRICTIONS ON THE ANGLE AND RATIO PARAMETERS:
 - A) ANGLE1 AND ANGLE2 MUST BE A VALUE THAT IS IN THE COUNTING RANGE OF TCR2 FOR THE APPLICATION.
 - B) IF ANGLE1 = ANGLE2, THEN RATIO2 MUST BE GREATER THAN RATIO1.
 - C) A RESTRICTION IS PLACED UPON THESE PARAMETERS BECAUSE OF TPU LATENCY; E.G., IF THE OUTPUT PULSE IS PROGRAMMED TO BE 99.9% DUTY CYCLE, AN OUTPUT PULSE OF A SMALLER PERCENTAGE MAY RESULT.
2. USED ONLY IN ANGLE-ANGLE MODE.
3. USED ONLY IN ANGLE-TIME MODE.

1043A

Figure A-4. PSP Parameters

CONTROL BITS

NAME	OPTIONS	ADDRESSES	
<div style="display: flex; justify-content: space-around; width: 100px;"> 3210 </div> <div style="display: flex; justify-content: space-around; width: 100px;"> <div style="border: 1px solid black; width: 20px; height: 15px;"></div> <div style="border: 1px solid black; width: 20px; height: 15px;"></div> <div style="border: 1px solid black; width: 20px; height: 15px;"></div> <div style="border: 1px solid black; width: 20px; height: 15px;"></div> </div>	CHANNEL FUNCTION SELECT	PMA FUNCTION NUMBER (ASSIGNED DURING MICRO-CODE ASSEMBLY)	###E0C-###E12
<div style="display: flex; justify-content: space-around; width: 100px;"> 10 </div> <div style="display: flex; justify-content: space-around; width: 100px;"> <div style="border: 1px solid black; width: 20px; height: 15px;"></div> <div style="border: 1px solid black; width: 20px; height: 15px;"></div> </div>	CHANNEL PRIORITY	00 — DISABLE 01 — LOW PRIORITY 10 — MEDIUM PRIORITY 11 — HIGH PRIORITY	###E1C-###E1E
<div style="display: flex; justify-content: space-around; width: 100px;"> 10 </div> <div style="display: flex; justify-content: space-around; width: 100px;"> <div style="border: 1px solid black; width: 20px; height: 15px;"></div> <div style="border: 1px solid black; width: 20px; height: 15px;"></div> </div>	HOST SEQUENCE BITS	00 — ADDITIONAL TOOTH BANK MODE 01 — ADDITIONAL TOOTH COUNT MODE 10 — (MISSING TOOTH BANK MODE) 11 — (MISSING TOOTH COUNT MODE)	###E14-###E16
<div style="display: flex; justify-content: space-around; width: 100px;"> 10 </div> <div style="display: flex; justify-content: space-around; width: 100px;"> <div style="border: 1px solid black; width: 20px; height: 15px;"></div> <div style="border: 1px solid black; width: 20px; height: 15px;"></div> </div>	HOST SERVICE BITS	00 — NO HOST SERVICE REQUEST 01 — INITIALIZE 10 — NOT USED 11 — NOT USED	###E18-###E1A
<div style="display: flex; justify-content: space-around; width: 100px;"> 0 </div> <div style="display: flex; justify-content: space-around; width: 100px;"> <div style="border: 1px solid black; width: 20px; height: 15px;"></div> </div>	INTERRUPT ENABLE	0 — INTERRUPT NOT ASSERTED 1 — INTERRUPT ASSERTED	###E0A
<div style="display: flex; justify-content: space-around; width: 100px;"> 0 </div> <div style="display: flex; justify-content: space-around; width: 100px;"> <div style="background-color: #cccccc; width: 20px; height: 15px;"></div> </div>	INTERRUPT STATUS		###E20

A

PARAMETER RAM

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
###FW0	REF_TIME								CHANNEL_CONTROL							
###FW2	MAX_ADDITIONAL								NUM_OF_TEETH							
###FW4	BANK_SIGNAL/ADDITIONAL_COUNT								ROLLOVER_COUNT							
###FW6	RATIO								TCR2_MAX_VALUE							
###FW8	PERIOD_HIGH_WORD															
###FWA	PERIOD_LOW_WORD															
###FFC	ERROR				TCR2_VALUE											

- = WRITTEN BY CPU
- = WRITTEN BY CPU AND TPU
- = WRITTEN BY TPU
- = UNUSED PARAMETERS
- W = CHANNEL NUMBER

1034A

Figure A-5. PMA Parameters

CONTROL BITS

	NAME	OPTIONS	ADDRESSES
3 2 1 0 <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/>	CHANNEL FUNCTION SELECT	PMM FUNCTION NUMBER (ASSIGNED DURING MICRO-CODE ASSEMBLY)	\$\$\$E0C-\$\$\$E12
1 0 <input type="checkbox"/> <input type="checkbox"/>	CHANNEL PRIORITY	00 — DISABLE 01 — LOW PRIORITY 10 — MEDIUM PRIORITY 11 — HIGH PRIORITY	\$\$\$E1C-\$\$\$E1E
1 0 <input type="checkbox"/> <input type="checkbox"/>	HOST SEQUENCE BITS	00 — (ADDITIONAL TOOTH BANK MODE) \$\$\$E14-\$\$\$E16 01 — (ADDITIONAL TOOTH COUNT MODE) 10 — MISSING TOOTH BANK MODE 11 — MISSING TOOTH COUNT MODE	
1 0 <input type="checkbox"/> <input type="checkbox"/>	HOST SERVICE BITS	00 — NO HOST SERVICE REQUEST 01 — INITIALIZE 10 — NOT USED 11 — NOT USED	\$\$\$E18-\$\$\$E1A
0 <input type="checkbox"/>	INTERRUPT ENABLE	0 — INTERRUPT NOT ASSERTED 1 — INTERRUPT ASSERTED	\$\$\$E0A
0 <input checked="" type="checkbox"/>	INTERRUPT STATUS		\$\$\$E20

PARAMETER RAM

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
\$\$\$FW0	REF_TIME							CHANNEL_CONTROL								
\$\$\$FW2	MAX_MISSING							NUM_OF_TEETH								
\$\$\$FW4	BANK_SIGNAL/MISSING_COUNT							ROLLOVER_COUNT								
\$\$\$FW6	RATIO							TCR2_MAX_VALUE								
\$\$\$FW8	PERIOD_HIGH_WORD															
\$\$\$FWA	PERIOD_LOW_WORD															
\$\$\$FFC	ERROR							TCR2_VALUE								

= WRITTEN BY CPU = WRITTEN BY CPU AND TPU
 = WRITTEN BY TPU = UNUSED PARAMETERS
 W = CHANNEL NUMBER

1038A

Figure A-6. PMM Parameters

CONTROL BITS

	NAME	OPTIONS	ADDRESSES
3 2 1 0 <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/>	CHANNEL FUNCTION SELECT	ITC FUNCTION NUMBER (ASSIGNED DURING MICRO-CODE ASSEMBLY)	###E0C-###E12
1 0 <input type="checkbox"/> <input type="checkbox"/>	CHANNEL PRIORITY	00 — DISABLE 01 — LOW PRIORITY 10 — MEDIUM PRIORITY 11 — HIGH PRIORITY	###E1C-###E1E
1 0 <input type="checkbox"/> <input type="checkbox"/>	HOST SEQUENCE BITS	00 — SINGLE-SHOT MODE, NO LINKS 01 — CONTINUOUS MODE, NO LINKS 10 — SINGLE-SHOT MODE, LINKS 11 — CONTINUOUS MODE, LINKS	###E14-###E16
1 0 <input type="checkbox"/> <input type="checkbox"/>	HOST SERVICE BITS	00 — NOT USED 01 — INITIALIZE 10 — NOT USED 11 — NOT USED	###E18-###E1A
0 <input type="checkbox"/>	INTERRUPT ENABLE	0 — INTERRUPT NOT ASSERTED 1 — INTERRUPT ASSERTED	###E0A
0 <input checked="" type="checkbox"/>	INTERRUPT STATUS		###E20

PARAMETER RAM

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
###FW0								CHANNEL_CONTROL								
###FW2	START_LINK_CHANNEL				LINK_CHANNEL_COUNT (2)			BANK_ADDRESS						0		
###FW4	MAX_COUNT (1, 3)															
###FW6	TRANS_COUNT (1)															
###FW8	FINAL_TRANS_TIME															
###FWA	LAST_TRANS_TIME															
###FWC																
###FWE																

<input type="checkbox"/> = WRITTEN BY CPU	<input checked="" type="checkbox"/> = WRITTEN BY CPU AND TPU
<input checked="" type="checkbox"/> = WRITTEN BY TPU	<input type="checkbox"/> = UNUSED PARAMETERS
W = CHANNEL NUMBER	

- NOTES:
- MAX_COUNT AND TRANS_COUNT SHOULD BE ACCESSED COHERENTLY AND RESIDE ON A DOUBLE-WORD BOUNDARY.
 - THE TPU DOES NOT PERFORM CHECKS ON LINK_CHANNEL_COUNT VALUE. IF LINK_CHANNEL_COUNT IS GREATER THAN EIGHT OR EQUAL TO ZERO, RESULTS ARE UNPREDICTABLE.
 - MAX_COUNT SHOULD BE BETWEEN ZERO AND \$FFFF. IF MAX_COUNT EQUALS ZERO, THE TPU COUNTS ONE TRANSITION.

1019A

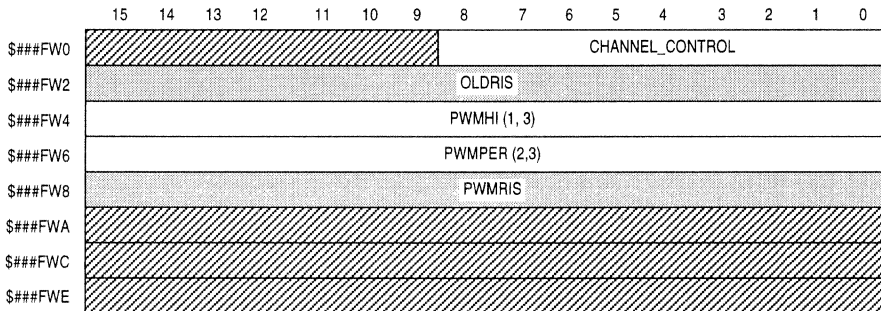
Figure A-7. ITC Parameters

CONTROL BITS

	NAME	OPTIONS	ADDRESSES
3 2 1 0 	CHANNEL FUNCTION SELECT	PWM FUNCTION NUMBER (ASSIGNED DURING MICRO-CODE ASSEMBLY)	###E0C-###E12
1 0 	CHANNEL PRIORITY	00 — DISABLE 01 — LOW PRIORITY 10 — MEDIUM PRIORITY 11 — HIGH PRIORITY	###E1C-###E1E
1 0 	HOST SEQUENCE BITS	xx — NOT USED	###E14-###E16
1 0 	HOST SERVICE BITS	00 — NOT USED 01 — IMMEDIATE UPDATE OF PWM 10 — INITIALIZE 11 — NOT USED	###E18-###E1A
0 	INTERRUPT ENABLE	0 — INTERRUPT NOT ASSERTED 1 — INTERRUPT ASSERTED	###E0A
0 	INTERRUPT STATUS		###E20

A

PARAMETER RAM



= WRITTEN BY CPU = WRITTEN BY CPU AND TPU
 = WRITTEN BY TPU = UNUSED PARAMETERS
 W = CHANNEL NUMBER

NOTES:

1. BEST-CASE MINIMUM FOR PWMHI IS 32 SYSTEM CLOCK CYCLES.
2. BEST-CASE MINIMUM FOR PWMPER IS 48 SYSTEM CLOCK CYCLES.
3. PWMHI AND PWMPER MUST BE ACCESSED COHERENTLY.

1027A

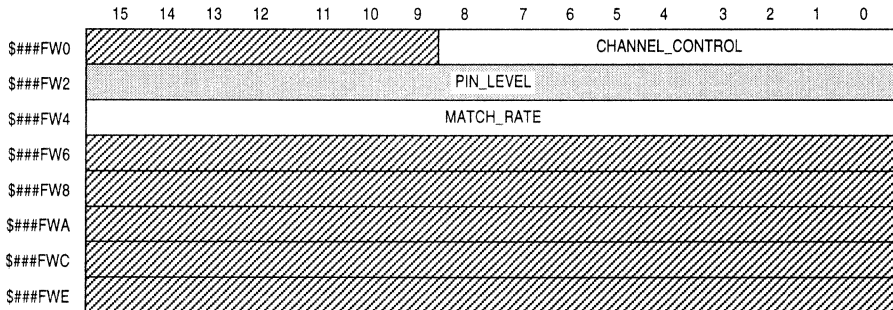
Figure A-8. PWM Parameters

CONTROL BITS

	NAME	OPTIONS	ADDRESSES
3 2 1 0 <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/>	CHANNEL FUNCTION SELECT	DIO FUNCTION NUMBER (ASSIGNED DURING MICRO-CODE ASSEMBLY)	###E0C-###E12
1 0 <input type="checkbox"/> <input type="checkbox"/>	CHANNEL PRIORITY	00 — DISABLE 01 — LOW PRIORITY 10 — MEDIUM PRIORITY 11 — HIGH PRIORITY	###E1C-###E1E
1 0 <input type="checkbox"/> <input type="checkbox"/>	HOST SEQUENCE BITS	00 — UPDATE ON TRANSITION 01 — UPDATE AT MATCH RATE 10 — UPDATE ON HSR 11 11 — NOT USED	###E14-###E16
1 0 <input type="checkbox"/> <input type="checkbox"/>	HOST SERVICE BITS	00 — NOT USED 01 — DRIVE PIN HIGH 10 — DRIVE PIN LOW 11 — INITIALIZE	###E18-###E1A
0 <input type="checkbox"/>	INTERRUPT ENABLE	0 — INTERRUPT NOT ASSERTED 1 — INTERRUPT ASSERTED	###E0A
0 <input checked="" type="checkbox"/>	INTERRUPT STATUS		###E20

A

PARAMETER RAM



<input type="checkbox"/> = WRITTEN BY CPU <input checked="" type="checkbox"/> = WRITTEN BY CPU AND TPU <input checked="" type="checkbox"/> = WRITTEN BY TPU	<input checked="" type="checkbox"/> = WRITTEN BY CPU AND TPU <input checked="" type="checkbox"/> = WRITTEN BY TPU <input checked="" type="checkbox"/> = UNUSED PARAMETERS
---	---

W = CHANNEL NUMBER

1017A

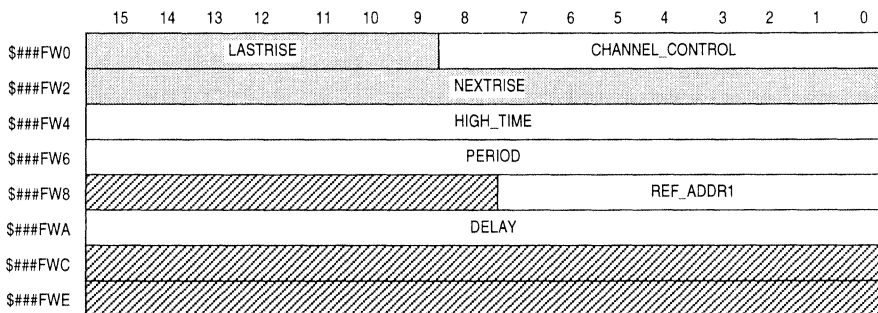
Figure A-9. DIO Parameters

CONTROL BITS

	NAME	OPTIONS	ADDRESSES
3 2 1 0 □ □ □ □	CHANNEL FUNCTION SELECT	SPWM FUNCTION NUMBER (ASSIGNED DURING MICRO-CODE ASSEMBLY)	\$\$\$E0C-\$\$\$E12
1 0 □ □	CHANNEL PRIORITY	00 — DISABLE 01 — LOW PRIORITY 10 — MEDIUM PRIORITY 11 — HIGH PRIORITY	\$\$\$E1C-\$\$\$E1E
1 0 □ □	HOST SEQUENCE BITS	00 — MODE 0 01 — MODE 1 10 — MODE 2 11 — NOT USED	\$\$\$E14-\$\$\$E16
1 0 □ □	HOST SERVICE BITS	00 — NO HOST SERVICE REQUEST 01 — NOT USED 10 — INITIALIZE 11 — IMMEDIATE UPDATE (MODE 1)	\$\$\$E18-\$\$\$E1A
0 □	INTERRUPT ENABLE	0 — INTERRUPT NOT ASSERTED 1 — INTERRUPT ASSERTED	\$\$\$E0A
0 □	INTERRUPT STATUS		\$\$\$E20

A

PARAMETER RAM (MODE 0)



= WRITTEN BY CPU
 = WRITTEN BY CPU AND TPU
 = WRITTEN BY TPU
 = UNUSED PARAMETERS
 W = CHANNEL NUMBER

1030A-1

Figure A-10. SPWM Parameters

A.14 Quadrature Decode (QDEC)

QDEC uses two channels to decode a pair of out-of-phase signals in order to present the CPU with directional information and a position value. It is particularly suitable for use with slotted encoders employed in motor control. The function derives full resolution from the encoder signals and provides a 16-bit position counter with rollover/under indication via an interrupt.

Figure A-11 shows all of the host interface areas for the QDEC function.

A.15 Programmable Time Accumulator (PTA)

PTA accumulates a 32-bit sum of the total high time, low time, or period of an input signal over a programmable number of periods or pulses. The period accumulation can start on a rising or falling edge. After the specified number of periods or pulses, the PTA generates an interrupt request.

From 1 to 255 period measurements can be accumulated before the TPU interrupts the CPU, providing instantaneous or average frequency measurement capability.

Figure A-12 shows all of the host interface areas for the PTA function.

A

A.16 Queued Output Match TPU Function (QOM)

QOM can generate single or multiple output match events from a table of offsets in parameter RAM. Loop modes allow complex pulse trains to be generated once, a specified number of times, or continuously. The function can be triggered by a link from another TPU channel. In addition, the reference time for the sequence of matches can be obtained from another channel. QOM can generate pulse-width modulated waveforms, including waveforms with high times of 0% or 100%. QOM also allows a TPU channel to be used as a discrete output pin.

Figure A-13 shows all of the host interface areas for the QOM function. The bit encodings shown below describe the corresponding fields in parameter RAM.

Bit Encoding

A	Timebase Selection
0	Use TCR1 as Timebase
1	Use TCR2 as Timebase

↓	Edge Selection
0	Falling Edge at Match
1	Rising Edge at Match

B:C	Reference for First Match
00	Immediate TCR Value
01	Last Event Time
10	Value Pointed to by REF_ADDR
11	Last Event Time

Figure A-13. QOM Parameters

A.17 Table Stepper Motor (TSM)

The TSM function provides for acceleration and deceleration control of a stepper motor with a programmable number of step rates up to 58. TSM uses a table in parameter RAM, rather than an algorithm, to define the stepper motor acceleration profile, allowing the user to fully define the profile. In addition, a slew rate parameter allows fine control of the terminal running speed of the motor independent of the acceleration table. The CPU need only write a desired position, and the TPU accelerates, slews, and decelerates the motor to the required position. Full and half step support is provided for two-phase motors. In addition, a slew rate parameter allows fine control of the terminal running speed of the motor independent of the acceleration table.

Figures A-14 and A-15 show all of the host interface areas for the TSM function when operating in master and slave mode, respectively.

CONTROL BITS

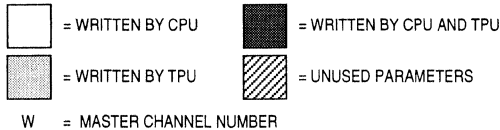
	NAME	OPTIONS	ADDRESSES
<div style="display: flex; justify-content: space-around; width: 100px;"> 3210 </div> <div style="border: 1px solid black; width: 100px; height: 20px; margin-top: 5px;"></div>	CHANNEL FUNCTION SELECT	TSM FUNCTION NUMBER (ASSIGNED DURING MICRO-CODE ASSEMBLY)	\$\$\$E0C-\$\$\$E12
<div style="display: flex; justify-content: space-around; width: 60px;"> 10 </div> <div style="border: 1px solid black; width: 60px; height: 20px; margin-top: 5px;"></div>	HOST SEQUENCE BITS	x0 — LOCAL MODE ACCELERATION TABLE x1 — SPLIT MODE ACCELERATION TABLE 0x — ROTATE PIN_SEQUENCE ONCE BETWEEN STEPS 1x — ROTATE PIN_SEQUENCE TWICE BETWEEN STEP	\$\$\$E14-\$\$\$E16
<div style="display: flex; justify-content: space-around; width: 60px;"> 10 </div> <div style="border: 1px solid black; width: 60px; height: 20px; margin-top: 5px;"></div>	HOST SERVICE BITS	00 — NO HOST SERVICE (RESET CONDITION) 01 — INITIALIZE, PIN LOW 10 — INITIALIZE, PIN HIGH 11 — MOVE REQUEST (MASTER ONLY)	\$\$\$E18-\$\$\$E1A
<div style="display: flex; justify-content: space-around; width: 60px;"> 10 </div> <div style="border: 1px solid black; width: 60px; height: 20px; margin-top: 5px;"></div>	CHANNEL PRIORITY	00 — DISABLED 01 — LOW PRIORITY 10 — MEDIUM PRIORITY 11 — HIGH PRIORITY	\$\$\$E1C-\$\$\$E1E
<div style="display: flex; justify-content: space-around; width: 60px;"> 0 </div> <div style="border: 1px solid black; width: 60px; height: 20px; margin-top: 5px;"></div>	CHANNEL INTERRUPT ENABLE	0 — CHANNEL INTERRUPTS DISABLED 1 — CHANNEL INTERRUPTS ENABLED	\$\$\$E0A
<div style="display: flex; justify-content: space-around; width: 60px;"> 0 </div> <div style="border: 1px solid black; width: 60px; height: 20px; margin-top: 5px; background-color: #cccccc;"></div>	CHANNEL INTERRUPT STATUS	0 — CHANNEL INTERRUPT NOT ASSERTED 1 — CHANNEL INTERRUPT ASSERTED	\$\$\$E20

A

PARAMETER RAM

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
\$\$\$F(W + 1)0	ACCEL_RATIO_2						ACCEL_RATIO_1									
\$\$\$F(W + 1)2	ACCEL_RATIO_4						ACCEL_RATIO_3									
\$\$\$F(W + 1)4	ACCEL_RATIO_6						ACCEL_RATIO_5									
\$\$\$F(W + 1)6	ACCEL_RATIO_8						ACCEL_RATIO_7									
\$\$\$F(W + 1)8	ACCEL_RATIO_10						ACCEL_RATIO_9									
\$\$\$F(W + 1)A	ACCEL_RATIO_12						ACCEL_RATIO_11									
\$\$\$F(W + 1)C *	ACCEL_RATIO_14 *						ACCEL_RATIO_13 *									
⋮	⋮						⋮									
\$\$\$F(W + 3)A *	ACCEL_RATIO_36 *						ACCEL_RATIO_35 *									

* OPTIONAL ADDITIONAL PARAMETERS NOT AVAILABLE IN ALL CASES. REFER TO MOTOROLA PROGRAMMING NOTE TPUPN04 FOR DETAILS.



TPU TSM SLV CHRT

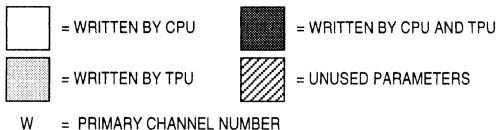
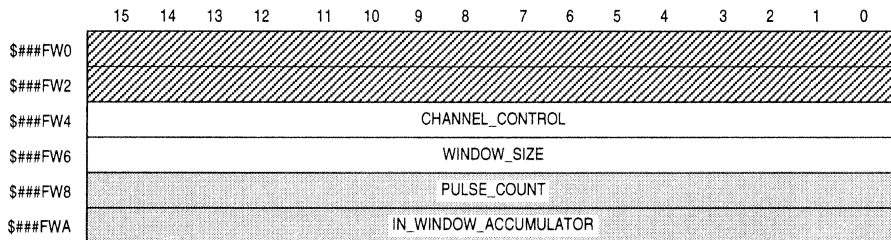
Figure A-15. TSM Parameters — Slave Mode

CONTROL BITS

NAME	OPTIONS	ADDRESSES	
<div style="display: flex; justify-content: space-around; width: 100px;"> 3210 </div> <div style="display: flex; justify-content: space-around; width: 100px;"> <div style="border: 1px solid black; width: 20px; height: 20px;"></div> <div style="border: 1px solid black; width: 20px; height: 20px;"></div> <div style="border: 1px solid black; width: 20px; height: 20px;"></div> <div style="border: 1px solid black; width: 20px; height: 20px;"></div> </div>	CHANNEL FUNCTION SELECT	FQM FUNCTION NUMBER (ASSIGNED DURING MICROCODE ASSEMBLY)	\$###E0C-\$###E12
<div style="display: flex; justify-content: space-around; width: 100px;"> 10 </div> <div style="display: flex; justify-content: space-around; width: 100px;"> <div style="border: 1px solid black; width: 20px; height: 20px;"></div> <div style="border: 1px solid black; width: 20px; height: 20px;"></div> </div>	HOST SEQUENCE BITS	00 — BEGIN WITH FALLING EDGE, SINGLE-SHOT MODE 01 — BEGIN WITH FALLING EDGE, CONTINUOUS MODE 10 — BEGIN WITH RISING EDGE, SINGLE-SHOT MODE 11 — BEGIN WITH RISING EDGE, CONTINUOUS MODE	\$###E14-\$###E16
<div style="display: flex; justify-content: space-around; width: 100px;"> 10 </div> <div style="display: flex; justify-content: space-around; width: 100px;"> <div style="border: 1px solid black; width: 20px; height: 20px;"></div> <div style="border: 1px solid black; width: 20px; height: 20px;"></div> </div>	HOST SERVICE BITS	00 — NO HOST SERVICE (RESET CONDITION) 01 — NOT USED 10 — INITIALIZE 11 — NOT USED	\$###E18-\$###E1A
<div style="display: flex; justify-content: space-around; width: 100px;"> 10 </div> <div style="display: flex; justify-content: space-around; width: 100px;"> <div style="border: 1px solid black; width: 20px; height: 20px;"></div> <div style="border: 1px solid black; width: 20px; height: 20px;"></div> </div>	CHANNEL PRIORITY	00 — DISABLED 01 — LOW PRIORITY 10 — MEDIUM PRIORITY 11 — HIGH PRIORITY	\$###E1C-\$###E1E
<div style="display: flex; justify-content: space-around; width: 100px;"> 0 </div> <div style="display: flex; justify-content: space-around; width: 100px;"> <div style="border: 1px solid black; width: 20px; height: 20px;"></div> </div>	CHANNEL INTERRUPT ENABLE	0 — CHANNEL INTERRUPTS DISABLED 1 — CHANNEL INTERRUPTS ENABLED	\$###E0A
<div style="display: flex; justify-content: space-around; width: 100px;"> 0 </div> <div style="display: flex; justify-content: space-around; width: 100px;"> <div style="background-color: #cccccc; border: 1px solid black; width: 20px; height: 20px;"></div> </div>	CHANNEL INTERRUPT STATUS	0 — CHANNEL INTERRUPT NOT ASSERTED 1 — CHANNEL INTERRUPT ASSERTED	\$###E20

A

PARAMETER RAM



TPU FQM CHRT

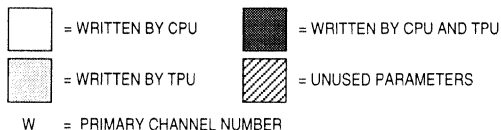
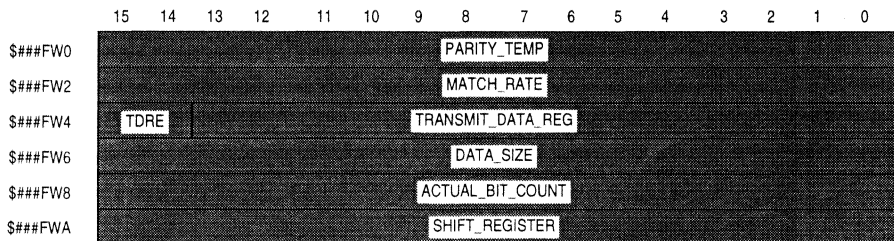
Figure A-16. FQM Parameters

CONTROL BITS

NAME	OPTIONS	ADDRESSES	
<div style="display: flex; justify-content: space-around; width: 100px;"> 3210 </div> <div style="display: flex; justify-content: space-around; width: 100px;"> <div style="border: 1px solid black; width: 20px; height: 20px;"></div> <div style="border: 1px solid black; width: 20px; height: 20px;"></div> <div style="border: 1px solid black; width: 20px; height: 20px;"></div> <div style="border: 1px solid black; width: 20px; height: 20px;"></div> </div>	CHANNEL FUNCTION SELECT	UART FUNCTION NUMBER (ASSIGNED DURING MICRO-CODE ASSEMBLY)	\$###E0C-\$###E12
<div style="display: flex; justify-content: space-around; width: 100px;"> 10 </div> <div style="display: flex; justify-content: space-around; width: 100px;"> <div style="border: 1px solid black; width: 20px; height: 20px;"></div> <div style="border: 1px solid black; width: 20px; height: 20px;"></div> </div>	HOST SEQUENCE BITS	00 — NO PARITY 01 — NO PARITY 10 — EVEN PARITY 11 — ODD PARITY	\$###E14-\$###E16
<div style="display: flex; justify-content: space-around; width: 100px;"> 10 </div> <div style="display: flex; justify-content: space-around; width: 100px;"> <div style="border: 1px solid black; width: 20px; height: 20px;"></div> <div style="border: 1px solid black; width: 20px; height: 20px;"></div> </div>	HOST SERVICE BITS	00 — NOT USED 01 — NOT USED 10 — TRANSMIT 11 — RECEIVE	\$###E18-\$###E1A
<div style="display: flex; justify-content: space-around; width: 100px;"> 10 </div> <div style="display: flex; justify-content: space-around; width: 100px;"> <div style="border: 1px solid black; width: 20px; height: 20px;"></div> <div style="border: 1px solid black; width: 20px; height: 20px;"></div> </div>	CHANNEL PRIORITY	00 — DISABLED 01 — LOW PRIORITY 10 — MEDIUM PRIORITY 11 — HIGH PRIORITY	\$###E1C-\$###E1E
<div style="display: flex; justify-content: space-around; width: 100px;"> 0 </div> <div style="display: flex; justify-content: space-around; width: 100px;"> <div style="border: 1px solid black; width: 20px; height: 20px;"></div> </div>	CHANNEL INTERRUPT ENABLE	0 — CHANNEL INTERRUPTS DISABLED 1 — CHANNEL INTERRUPTS ENABLED	\$###E0A
<div style="display: flex; justify-content: space-around; width: 100px;"> 0 </div> <div style="display: flex; justify-content: space-around; width: 100px;"> <div style="border: 1px solid black; width: 20px; height: 20px; background-color: #cccccc;"></div> </div>	CHANNEL INTERRUPT STATUS	0 — CHANNEL INTERRUPT NOT ASSERTED 1 — CHANNEL INTERRUPT ASSERTED	\$###E20

A

PARAMETER RAM (TRANSMITTER)



TPU UART TRANS CHRT

Figure A-17. UART Transmitter Parameters

A.20 New Input Capture/Transition Counter (NITC)

Any channel of the TPU can capture the value of a specified TCR or any specified location in parameter RAM upon the occurrence of each transition or specified number of transitions, and then generate an interrupt request to notify the bus master. The times of the most recent two transitions are maintained in parameter RAM. A channel can perform input captures continually, or a channel can detect a single transition or specified number of transitions, ceasing channel activity until reinitialization. After each transition or specified number of transitions, the channel can generate a link to other channels.

Figure A-19 shows all of the host interface areas for the NITC function.

A.21 Multiphase Motor Commutation (COMM)

The COMM function generates the phase commutation signals for a variety of brushless motors, including three-phase brushless direct current. It derives the commutation state directly from the position decoded in FQD, thus eliminating the need for hall effect sensors.

The state sequence is implemented as a user-configurable state machine, thus providing a flexible approach with other general applications. A CPU offset parameter is provided to allow all the switching angles to be advanced or retarded on the fly by the CPU. This feature is useful for torque maintenance at high speeds.

Figure A-20 shows all of the host interface areas for the COMM function.

A

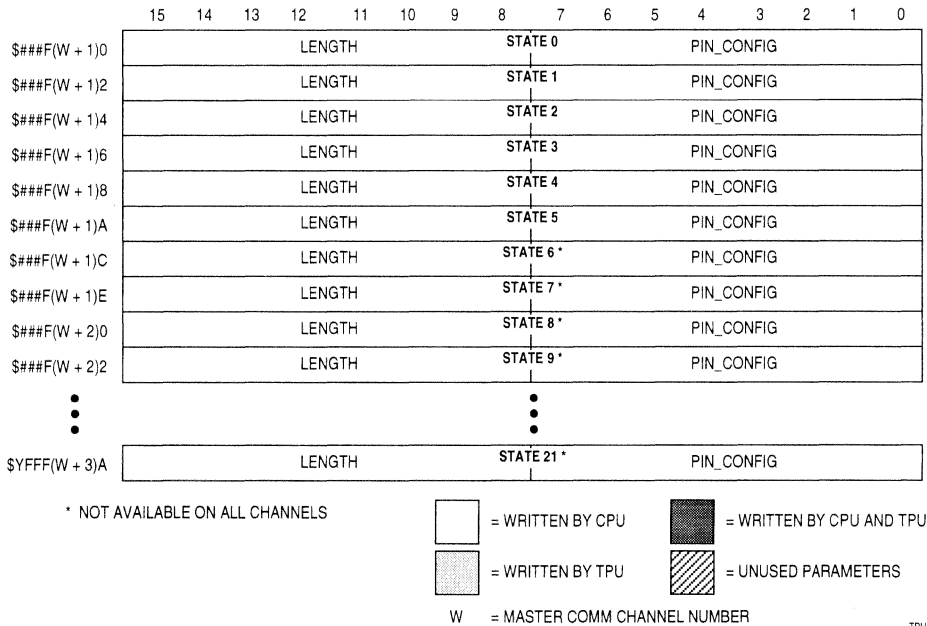


Figure A-20. COMM Parameters (Continued)

A.22 Multichannel Pulse-Width Modulation (MCPWM)

MCPWM generates pulse-width modulated outputs with full 0% to 100% duty cycle range independent of other TPU activity. This capability requires two TPU channels plus an external gate for one PWM channel. (A simple one-channel PWM capability is supported by the QOM function.)

Multiple PWMs generated by MCPWM have two types of high time alignment: edge aligned and center aligned. Edge-aligned mode uses $n + 1$ TPU channels for n PWMs; center-aligned mode uses $2n + 1$ channels. Center-aligned mode allows a user-defined 'dead time' to be specified so that two PWMs can be used to drive an H-bridge without destructive current spikes. This feature is important for motor control applications.

Figures A-21 through A-26 show the host interface areas for the MCPWM function in each mode.



CONTROL BITS

	NAME	OPTIONS	ADDRESSES
3 2 1 0 <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/>	CHANNEL FUNCTION SELECT	MCPWM FUNCTION NUMBER (ASSIGNED DURING MICRO-CODE ASSEMBLY)	\$\$\$E0C-\$\$\$E12
1 0 <input type="checkbox"/> <input type="checkbox"/>	HOST SEQUENCE	00 — EDGE ALIGNED MODE 01 — SLAVE A TYPE CENTER ALIGNED MODE 10 — SLAVE B TYPE CENTER ALIGNED MODE 11 — SLAVE C TYPE CENTER ALIGNED MODE	\$\$\$E14-\$\$\$E16
1 0 <input type="checkbox"/> <input type="checkbox"/>	HOST SERVICE REQUEST	00 — NO HOST SERVICE (RESET CONDITION) 01 — INITIALIZE AS SLAVE (INVERTED) 10 — INITIALIZE AS SLAVE (NORMAL) 11 — INITIALIZE AS MASTER	\$\$\$E18-\$\$\$E1A
1 0 <input type="checkbox"/> <input type="checkbox"/>	CHANNEL PRIORITY	00 — DISABLED 01 — LOW PRIORITY 10 — MEDIUM PRIORITY 11 — HIGH PRIORITY	\$\$\$E1C-\$\$\$E1E
0 <input type="checkbox"/>	CHANNEL INTERRUPT ENABLE	0 — CHANNEL INTERRUPTS DISABLED 1 — CHANNEL INTERRUPTS ENABLED	\$\$\$E0A
0 <input type="checkbox"/>	CHANNEL INTERRUPT STATUS	0 — CHANNEL INTERRUPT NOT ASSERTED 1 — CHANNEL INTERRUPT ASSERTED	\$\$\$E20

PARAMETER RAM

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
\$\$\$FW0	PERIOD															
\$\$\$FW2	HIGH_TIME															
\$\$\$FW4	[Unused]															
\$\$\$FW6	HIGH_TIME_PTR															
\$\$\$FW8	RISE_TIME_PTR															
\$\$\$FWA	FALL_TIME_PTR															
\$\$\$FWC	[Unused]															
\$\$\$FWE	[Unused]															

- = WRITTEN BY CPU
 - = WRITTEN BY CPU AND TPU
 - = WRITTEN BY TPU
 - = UNUSED PARAMETERS
- W = PRIMARY CHANNEL NUMBER

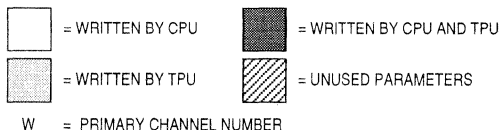
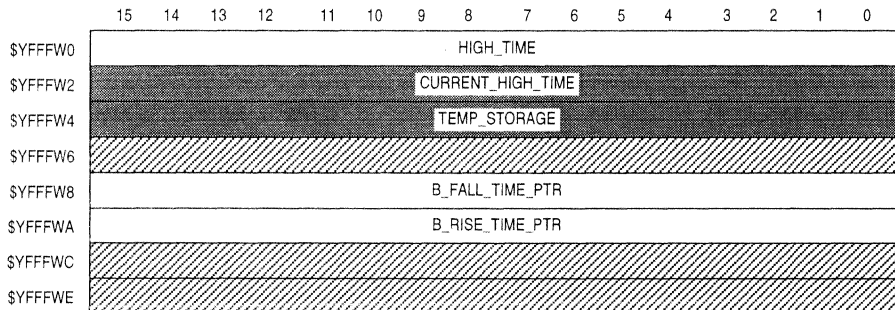
TPU MCPWM S EA CHRT

Figure A-22. MCPWM Parameters — Slave Edge-Aligned Mode

CONTROL BITS

	NAME	OPTIONS	ADDRESSES
0 <input type="checkbox"/>	CHANNEL INTERRUPT ENABLE	0 — CHANNEL INTERRUPTS DISABLED 1 — CHANNEL INTERRUPTS ENABLED	\$YFFE0A
3 2 1 0 <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/>	CHANNEL FUNCTION SELECT	xxxx — FQD FUNCTION NUMBER (ASSIGNED DURING MICROCODE ASSEMBLY)	\$YFFE0C-\$YFFE12
1 0 <input type="checkbox"/> <input type="checkbox"/>	HOST SEQUENCE	00 — EDGE ALIGNED MODE 01 — SLAVE A TYPE CENTER ALIGNED MODE 10 — SLAVE B TYPE CENTER ALIGNED MODE 11 — SLAVE C TYPE CENTER ALIGNED MODE	\$YFFE14-\$YFFE16
1 0 <input type="checkbox"/> <input type="checkbox"/>	HOST SERVICE REQUEST	00 — NO HOST SERVICE (RESET CONDITION) 01 — INITIALIZE AS SLAVE (INVERTED) 10 — INITIALIZE AS SLAVE (NORMAL) 11 — INITIALIZE AS MASTER	\$YFFE18-\$YFFE1A
1 0 <input type="checkbox"/> <input type="checkbox"/>	CHANNEL PRIORITY	00 — DISABLED 01 — LOW PRIORITY 10 — MEDIUM PRIORITY 11 — HIGH PRIORITY	\$YFFE1C-\$YFFE1E
0 <input type="checkbox"/>	CHANNEL INTERRUPT STATUS	0 — CHANNEL INTERRUPT NOT ASSERTED 1 — CHANNEL INTERRUPT ASSERTED	\$YFFE20

PARAMETER RAM



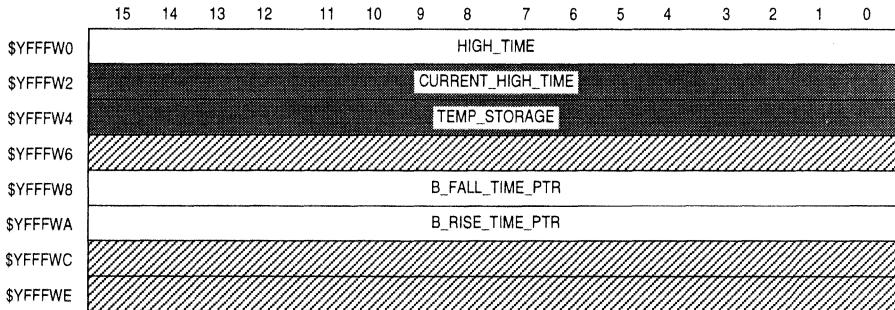
TPU MCPWM SB NICA CHRT

Figure A-24. MCPWM Parameters — Slave Channel B Non-Inverted Center-Aligned Mode

CONTROL BITS

NAME	OPTIONS	ADDRESSES
<div style="display: flex; justify-content: center; align-items: center;"> 0 <div style="border: 1px solid black; width: 20px; height: 15px; display: inline-block;"></div> </div>	CHANNEL INTERRUPT ENABLE 0 — CHANNEL INTERRUPTS DISABLED 1 — CHANNEL INTERRUPTS ENABLED	\$YFFE0A
<div style="display: flex; justify-content: center; align-items: center;"> 3 2 1 0 <div style="display: flex; gap: 5px;"> <div style="border: 1px solid black; width: 20px; height: 15px;"></div> <div style="border: 1px solid black; width: 20px; height: 15px;"></div> <div style="border: 1px solid black; width: 20px; height: 15px;"></div> <div style="border: 1px solid black; width: 20px; height: 15px;"></div> </div> </div>	CHANNEL FUNCTION SELECT xxxx — FQD FUNCTION NUMBER (ASSIGNED DURING MICROCODE ASSEMBLY)	\$YFFE0C—\$YFFE12
<div style="display: flex; justify-content: center; align-items: center;"> 1 0 <div style="display: flex; gap: 5px;"> <div style="border: 1px solid black; width: 20px; height: 15px;"></div> <div style="border: 1px solid black; width: 20px; height: 15px;"></div> </div> </div>	HOST SEQUENCE 00 — EDGE ALIGNED MODE 01 — SLAVE A TYPE CENTER ALIGNED MODE 10 — SLAVE B TYPE CENTER ALIGNED MODE 11 — SLAVE C TYPE CENTER ALIGNED MODE	\$YFFE14—\$YFFE16
<div style="display: flex; justify-content: center; align-items: center;"> 1 0 <div style="display: flex; gap: 5px;"> <div style="border: 1px solid black; width: 20px; height: 15px;"></div> <div style="border: 1px solid black; width: 20px; height: 15px;"></div> </div> </div>	HOST SERVICE REQUEST 00 — NO HOST SERVICE (RESET CONDITION) 01 — INITIALIZE AS SLAVE (INVERTED) 10 — INITIALIZE AS SLAVE (NORMAL) 11 — INITIALIZE AS MASTER	\$YFFE18—\$YFFE1A
<div style="display: flex; justify-content: center; align-items: center;"> 1 0 <div style="display: flex; gap: 5px;"> <div style="border: 1px solid black; width: 20px; height: 15px;"></div> <div style="border: 1px solid black; width: 20px; height: 15px;"></div> </div> </div>	CHANNEL PRIORITY 00 — DISABLED 01 — LOW PRIORITY 10 — MEDIUM PRIORITY 11 — HIGH PRIORITY	\$YFFE1C—\$YFFE1E
<div style="display: flex; justify-content: center; align-items: center;"> 0 <div style="border: 1px solid black; width: 20px; height: 15px; display: inline-block;"></div> </div>	CHANNEL INTERRUPT STATUS 0 — CHANNEL INTERRUPT NOT ASSERTED 1 — CHANNEL INTERRUPT ASSERTED	\$YFFE20

PARAMETER RAM



= WRITTEN BY CPU
 = WRITTEN BY CPU AND TPU
 = WRITTEN BY TPU
 = UNUSED PARAMETERS
W = PRIMARY CHANNEL NUMBER

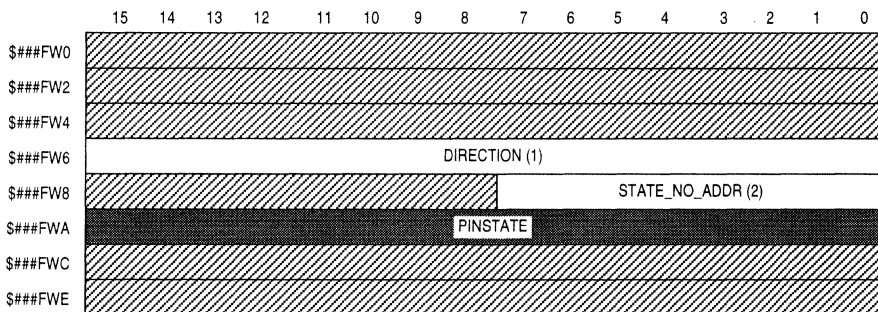
TPU MCPWM SB ICA CHRT

**Figure A-26. MCPWM Parameters —
Slave Channel B Inverted Center-Aligned Mode**

CONTROL BITS

	NAME	OPTIONS	ADDRESSES
3 2 1 0 □ □ □ □	CHANNEL FUNCTION SELECT	HALLD FUNCTION NUMBER (ASSIGNED DURING MICROCODE ASSEMBLY)	\$\$\$E0C-\$\$\$E12
1 0 □ □	HOST SEQUENCE	00 — CHANNEL A 01 — CHANNEL B 10 — CHANNEL B 11 — CHANNEL C (3-CHANNEL MODE ONLY)	\$\$\$E14-\$\$\$E16
1 0 □ □	HOST SERVICE REQUEST	00 — NO HOST SERVICE (RESET CONDITION) 01 — NOT USED 10 — INITIALIZE, 2-CHANNEL MODE 11 — INITIALIZE, 3-CHANNEL MODE	\$\$\$E18-\$\$\$E1A
1 0 □ □	CHANNEL PRIORITY	00 — DISABLED 01 — LOW PRIORITY 10 — MEDIUM PRIORITY 11 — HIGH PRIORITY	\$\$\$E1C-\$\$\$E1E
0 □	CHANNEL INTERRUPT ENABLE	0 — CHANNEL INTERRUPTS DISABLED 1 — CHANNEL INTERRUPTS ENABLED	\$\$\$E0A
0 □	CHANNEL INTERRUPT STATUS	x — NOT USED	\$\$\$E20

PARAMETER RAM



NOTES:

1. CHANNEL A ONLY.
2. 1 CHANNEL ONLY (CHANNEL B IN 2-CHANNEL MODE, CHANNEL C IN 3-CHANNEL MODE).

□	= WRITTEN BY CPU	■	= WRITTEN BY CPU AND TPU
▨	= WRITTEN BY TPU	▩	= UNUSED PARAMETERS
W	= CHANNEL NUMBER		

TPU HALLD CHRT

Figure A-27. HALLD Parameters

CONTROL BITS

	NAME	OPTIONS	ADDRESSES
3 2 1 0 <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/>	CHANNEL FUNCTION SELECT	FQD FUNCTION NUMBER (ASSIGNED DURING MICROCODE ASSEMBLY)	\$###E0C-\$###E12
1 0 <input type="checkbox"/> <input type="checkbox"/>	HOST SEQUENCE BITS	00 — PRIMARY CHANNEL (NORMAL MODE) 01 — SECONDARY CHANNEL (NORMAL MODE) 10 — PRIMARY CHANNEL (FAST MODE) 11 — SECONDARY CHANNEL (FAST MODE)	\$###E14-\$###E16
1 0 <input type="checkbox"/> <input type="checkbox"/>	HOST SERVICE BITS	00 — NO HOST SERVICE (RESET CONDITION) 01 — NOT USED 10 — READ TCR1 11 — INITIALIZE	\$###E18-\$###E1A
1 0 <input type="checkbox"/> <input type="checkbox"/>	CHANNEL PRIORITY	00 — DISABLED 01 — LOW PRIORITY 10 — MEDIUM PRIORITY 11 — HIGH PRIORITY	\$###E1C-\$###E1E
0 <input type="checkbox"/>	CHANNEL INTERRUPT ENABLE	x — NOT USED	\$###E0A
0 <input type="checkbox"/>	CHANNEL INTERRUPT STATUS	xx — NOT USED	\$###E20

PARAMETER RAM

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
\$###FW0	EDGE_TIME															
\$###FW2	POSITION_COUNT															
\$###FW4	TCR1_VALUE															
\$###FW6	CHAN_PINSTATE															
\$###FW8	CORR_PINSTATE_ADDR															
\$###FWA	EDGE_TIME_LSB_ADDR															
\$###FWC	UNUSED PARAMETERS															
\$###FWE	UNUSED PARAMETERS															

- = WRITTEN BY CPU = WRITTEN BY CPU AND TPU
 = WRITTEN BY TPU = UNUSED PARAMETERS
 W = CHANNEL NUMBER

TPU FQD PRI CHRT

Figure A-28. FQD Parameters — Primary Channel

APPENDIX B REGISTER SUMMARY

This appendix summarizes the bit functions in the TPU control registers.

B.1 Register Summary

Table B-1 summarizes the TPU registers, bits, and reset states.

Table B-1. TPU Register Summary

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TPUMCR ###E00	STOP 0	TCR1P PRESCALER 0 0		TCR2P PRESCALER 0 0		EMU 0	T2CG 0	STF 0	SUPV 1	PSCK 0	0	0	INTERRUPT ARBITRATION ID 0 0 0 0			
TCR ###E08	0	0	0	0	0	CHANNEL INTERRUPT REQUEST LEVEL 0 0 0			CHANNEL INTERRUPT BASE VECTOR 0 0 0 0				0	0	0	0
CIER ###E0A	CH15 0	CH14 0	CH13 0	CH12 0	CH11 0	CH10 0	CH9 0	CH8 0	CH7 0	CH6 0	CH5 0	CH4 0	CH3 0	CH2 0	CH1 0	CHA0 0
CFSR0 ###E0C	CHANNEL15 0 0 0 0				CHANNEL14 0 0 0 0				CHANNEL13 0 0 0 0				CHANNEL12 0 0 0 0			
CFSR1 ###E0E	CHANNEL11 0 0 0 0				CHANNEL10 0 0 0 0				CHANNEL9 0 0 0 0				CHANNEL8 0 0 0 0			
CFSR2 ###E10	CHANNEL7 0 0 0 0				CHANNEL6 0 0 0 0				CHANNEL5 0 0 0 0				CHANNEL4 0 0 0 0			
CFSR3 ###E12	CHANNEL3 0 0 0 0				CHANNEL2 0 0 0 0				CHANNEL1 0 0 0 0				CHANNEL0 0 0 0 0			
HSQR0 ###E14	CH15 0 0		CH14 0 0		CH13 0 0		CH12 0 0		CH11 0 0		CH10 0 0		CH9 0 0		CH8 0 0	
HSQR1 ###E16	CH7 0 0		CH6 0 0		CH5 0 0		CH4 0 0		CH3 0 0		CH2 0 0		CH1 0 0		CH0 0 0	
HSRR0 ###E18	CH15 0 0		CH14 0 0		CH13 0 0		CH12 0 0		CH11 0 0		CH10 0 0		CH9 0 0		CH8 0 0	
HSRR1 ###E1A	CH7 0 0		CH6 0 0		CH5 0 0		CH4 0 0		CH3 0 0		CH2 0 0		CH1 0 0		CH0 0 0	
CPR0 ###E1C	CH15 0 0		CH14 0 0		CH13 0 0		CH12 0 0		CH11 0 0		CH10 0 0		CH9 0 0		CH8 0 0	
CPR1 ###E1E	CH7 0 0		CH6 0 0		CH5 0 0		CH4 0 0		CH3 0 0		CH2 0 0		CH1 0 0		CH0 0 0	
CISR ###E20	CH15 0	CH14 0	CH13 0	CH12 0	CH11 0	CH10 0	CH9 0	CH8 0	CH7 0	CH6 0	CH5 0	CH4 0	CH3 0	CH2 0	CH1 0	CH0 0

B

APPENDIX C ESTIMATING WORST-CASE LATENCY

Reliable systems are designed to work under worst-case conditions. This section explains how to estimate worst-case latency (WCL) for any TPU function in any system. The appendix covers the following topics:

- Introduction to Worst-Case Latency
- Using Worst-Case Latency Estimates to Evaluate Performance
- Priority Scheme Details used in WCL Analyses
- First-Pass WCL Analysis
- Second-Pass WCL Analysis

The first-pass WCL analysis is based on a deterministic, generalized formula that is easy to apply. Because of the generalizations in the formula, the first analysis result is almost always much worse than the real worst case. If the desired system performance is within the limits of this first analysis, then no further analysis is required; the system is well within the performance limits of the TPU. If the desired system performance exceeds that indicated by the first analysis, the second-pass WCL analysis should be applied. The second-pass analysis is not a generalized formula, but rather uses specific system details for a realistic worst-case estimation.

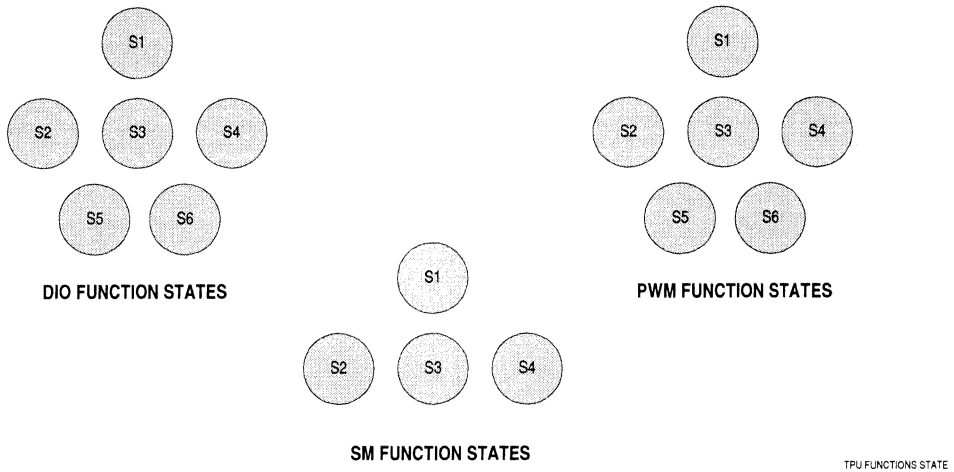


Figure C-2. TPU Function States

Since there is only one TPU execution unit, the TPU cannot actually execute the software for multiple functions simultaneously. However, the hardware for each of the sixteen channels is independent. This means that, for example, all sixteen channel pins can change state at the same moment, provided that the function software sets up the channel hardware to do so beforehand.

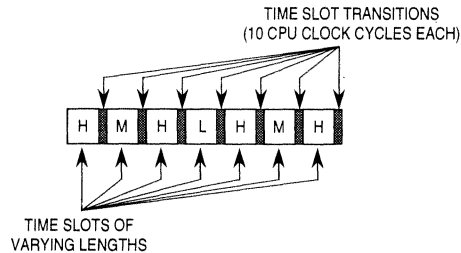
With CPU code, the system designer assigns functions to channels and initializes the functions. After initialization, functions typically run without CPU intervention, except for TPU channel interrupts to the CPU to give or receive information. Most functions can run continuously with periodic servicing from the TPU execution unit. As required, the channels request to be serviced from the TPU execution unit, and the TPU scheduler determines the order in which the channels are serviced. Worst-case latency for a channel can be derived from the details of the priority scheme that the scheduler uses.

C

C.3 Priority Scheme Details Used in WCL Analysis

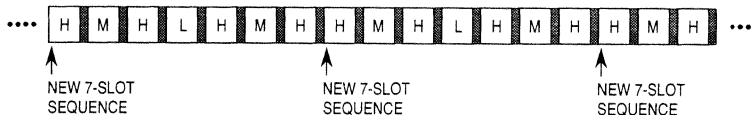
The user assigns functions to channel numbers and gives each active channel a priority level of high, middle, or low. The scheduler uses the channel number and channel priority level to determine the order in which to grant service.

The scheduler allocates time slots to specific priority levels of high, middle, or low. One function state is executed in each time slot. The length of a time slot varies according to the length of the executing state. The scheduler always assigns time slots in a seven-slot sequence (see Figure C-3). After a seven-slot sequence is completed, another seven-slot sequence begins (see Figure C-4).



TPU TIME SLOT TIM

Figure C-3. Time-Slot Sequence



TPU MULT TIME SLOT TIM

Figure C-4. Multiple Time-Slot Sequences

This fixed-sequence scheme gives higher-priority channels more service time than lower-priority channels. High-priority channels are allocated four of seven time slots, middle-priority channels are allocated two of seven time slots, and low-priority channels are allocated one of seven time slots.

For example, if a middle-priority channel has just been serviced (either in a middle-priority time slot or a high or low-priority time slot gained by priority passing), the SRLs and SGLs of all middle-priority channels are compared. If there is no middle-priority channel with its SRL set and SGL cleared, the scheduler clears all middle-level SGLs. If there is a middle-level channel with its SRL set and SGL cleared, the scheduler does not clear the SGL group, and the requesting middle-level channel is serviced on the next middle-level time slot (or possibly sooner by priority passing).

While it is clearing a group of SGLs, the TPU issues a 4-clock no operation (NOP). This four clock delay must be included in worst case latency estimations as appropriate.

C.3.4 RAM Collision Rate

Most function states read or write to the TPU parameter RAM at least once. Because both the TPU and CPU can access the parameter RAM but not at the same time, the TPU may stall during a parameter RAM access while waiting for the CPU to finish accessing the RAM. At other times the CPU may stall for the TPU. A stall can take up to two CPU clocks. TPU stalls must be added into the worst-case latency calculation. The system designer should estimate the percentage of parameter RAM accesses in the system that will result in a TPU stall. This percentage is called the RAM collision rate (RCR).

A 100% RCR for a system is the theoretical worst case. In many systems, however, the RCR is very low, sometimes even near 0%. This is because the TPU is an independent processor capable of servicing most function needs, so that the CPU rarely needs to access the TPU parameter RAM. To find a realistic RCR, the system designer should evaluate the CPU code and find the percentage of time it accesses the TPU parameter RAM. This percentage is a good RCR.

NOTE

The programming practice of polling a flag in the TPU parameter RAM causes a very high RCR and should be avoided in high-performance systems.

After a RAM collision rate for a system is found, it can be applied to the WCL calculations for each channel. The system designer can use the RCR and the number of RAM accesses in the longest state to estimate the TPU stall time for a function. The estimation of TPU stall time is as follows:

$$\text{Function TPU stall time} = \text{number of RAM accesses in the longest state} * \text{RCR} * 2 \text{ CPU clocks.}$$

Table C-1. Longest States and RAM Accesses for Mask Set A Functions

Function	Longest State	RAM Accesses
DIO	10	4
ITC	40 (no linking) 42 (linking)	7
OC	40	7
PWM	24	4
SPWM		
Mode 0	14	4
Mode 1	18	4
Mode 2	20 (no linking) 22 (linking)	4 4
PMA	94	8
PMM	94	8
PSP		
Angle-Angle Mode	76	6
Angle-Time Mode	50	3
SM ¹	160	21
PPWA		
Mode 0	44	9
Mode 1	50 ²	10
Mode 2	44	9
Mode 3	50	10

NOTES:

1. Assumes one master and one slave. For each additional slave
 - a) add 32 clocks and 2 RAM accesses, and
 - b) add (STEP_RATE_CNT * two clocks).
2. With one channel linked. Add two clocks for each additional channel linked.

The worst-case service time for each channel is:

$$\text{Longest state} + (\text{number of RAM accesses in longest state} * \text{RCR} * 2 \text{ clocks})$$



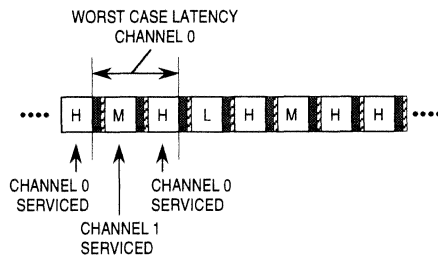
C.4.1.2 Mapping the Channels for Each Time Slot

To determine when a channel will be serviced again, it is necessary to determine which other channels will be serviced first. Do this by assuming all channels are continuously requesting service and mapping the channels into the time-slot sequence.

C.4.1.3 Adding Time for Time-Slot Transitions and NOPs

Add time for time-slot transitions which occur after each time slot and four-clock NOPs which occur when the TPU clears a group of SGLs to start a new cycle for a priority level.

■ = 10-CYCLE TIME SLOT TRANSITION
 ▨ = 4-CYCLE NOP INSTRUCTION



TPU CH0 WCL TIM

Figure C-6. Next Servicing for Channel 0

Channel 1 will be serviced in the middle-priority time slot before channel 0 is serviced again.

3. Add time for the ten-clock CPU time-slot transitions and the four-clock NOPs. See Figure C-6.

A four-clock NOP occurs after each channel is serviced since there is one channel in each priority level, i.e., a new cycle for a priority level is started after each channel is serviced. Time-slot transitions occur after each time slot.

Worst-Case Latency for Channel 0:

Channel 0 worst-case service time	25 clocks
Channel 1 worst-case service time	46 clocks
Two 10-clock time-slot transitions	20 clocks
Two 4-clock NOPs	<u>8 clocks</u>
	99 clocks

$$99 \text{ clocks} * 60 \text{ ns/clock} = 5940 \text{ ns}$$

Conclusion: in this system configuration PWM can run with a minimum high time or low time of 5940 ns.

C

C.4.2.3 Finding the WCL for DIO on Channel 2

1. Find the worst-case service time for each active channel. See step 1 of previous examples.
2. Assume channel 2 has just been serviced and that channels 0 and 1 are continuously requesting service. Using the H-M-H-L-H-M-H time-slot sequence, map the channels that are granted for each time slot. See Figure C-8.

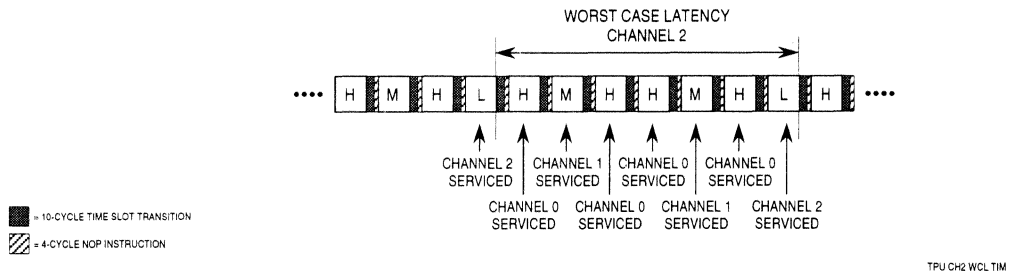


Figure C-8. Next Servicing for Channel 2

Channel 0 will be serviced four times and channel 1 twice before channel 2 is serviced again.

3. Add time for the ten-clock CPU time-slot transitions and the four-clock NOPs. See Figure C-8.

Worst-Case Latency for Channel 2:

Four Channel 0 worst-case service times	100 clocks
Two Channel 1 worst-case service time	92 clocks
Channel 2 worst-case service time	11 clocks
Seven 10-clock time-slot transitions	70 clocks
Seven 4-clock NOPs	<u>28 clocks</u>
	301 clocks

$$301 \text{ clocks} * 60 \text{ ns/clock} = 18060 \text{ ns}$$

Conclusion: in this system configuration DIO can keep track of the input level at a minimum of every 18060 ns.

Make sure that when mapping out channels to the sequence, you choose a worst-case slot to start the mapping. For example, when estimating WCL for a high-priority channel, do not start the mapping in the last high-priority slot in a seven-slot sequence, as that is a best case for a high-priority channel since another high-priority time slot is next.

6. Instead of always using the longest state in the function as the worst-case state, evaluate the states in the function that will be used in the system and use the appropriate worst-case states. For example, in the preceding example of first-pass analysis, the PWM was shown to be able to achieve a high time and low time of 5940 ns under worst-case conditions. This was derived using the longest PWM state of 24 CPU clocks. This longest state is actually state 2, the state that is entered after the pin has just gone high. State 3, the state that is entered after the pin has just gone low, requires only 2 CPU clocks. Therefore, in the first-pass example, the high time was correctly derived, but the low time is actually shorter than was estimated.

C.5.2 Second-Pass Analysis Example

This example requires three 50% PWM waveforms: one 5 kHz (200 μ s/period) and two 50 kHz (20 μ s/period), each running DC motors. (Remember that the PWM function requests service from the TPU after each high time and after each low time, so the TPU must handle a request every 100 μ s for the 5 kHz PWM and every 10 μ s for the 50 MHz PWM.)

NOTE

This example uses square waves for simplicity. Notice that to use a PWM waveform in the typical way, in which the pulse is modulated, the pulse must not be modulated in a way that violates the worst-case latency requirements.

This example also uses one DIO channel monitoring a signal level every millisecond and one PPWA channel in mode 0 monitoring the speed of the 5-kHz DC motor. The PPWA must measure periods of 5 kHz (200 μ s/period).

The CPU is interrupted by the channel running the PPWA function after measuring 200 periods (every 40 ms). The interrupt service routine performs an averaging of the period accumulation and checks it against a known parameter. The interrupt service time is so short and infrequent that it is a tiny fraction of total system time. The interrupt service routine contains no polling of the parameter RAM. Therefore a realistic RCR = 0%.

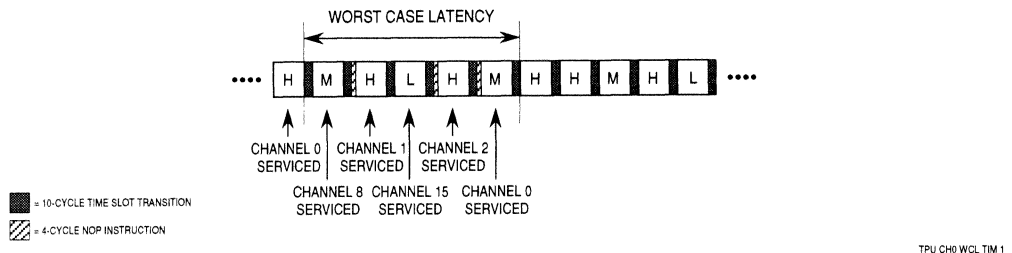


Figure C–9. Worst-Case Latency for Channel 0 (First Try)

Conclusion: with this system configuration, worst-case latencies for channels 0 and 1 are too high (WCL for channel 1 is the same as WCL for channel 0). Try a different system configuration.

C.5.2.2 Second-Try System Configuration

The second-try system configuration is shown below.

Channel	Priority	Function
0	High	PWM at 50 kHz (needs a 10- μ s WCL)
1	High	PWM at 50 kHz (needs a 10- μ s WCL)
2	Middle	PWM at 5 kHz (needs a 100- μ s WCL)
8	Middle	PPWA at 5 kHz (needs a 200- μ s WCL)
15	Low	DIO as input at rate of 1 ms

0% RAM collision rate
 CPU clock rate = 16.67 MHz, or 60 ns per clock period



To find the WCL for channel 0, assume channel 0 has just finished service. Map the channels in the H-M-H-L-H-M-H sequence. See Figure C–10.

Notice that channels 2 and 8 are well within their WCL requirements. The system could be re-configured as follows to give channels 0 and 1 a larger margin while still keeping channels 2, 8 and 15 within their WCL requirements:

Channel	Priority	Function
0	High	PWM at 50 kHz (needs a 10- μ s WCL)
1	High	PWM at 50 kHz (needs a 10- μ s WCL)
2	Low	PWM at 5 kHz (needs a 100- μ s WCL)
8	Middle	PPWA at 5 kHz (needs a 200- μ s WCL)
15	Low	DIO as input at rate of 1 ms

0% RAM collision rate

CPU clock rate = 16.67 MHz, or 60 ns per clock period

APPENDIX D CHANNEL HARDWARE DIAGRAM

Figure D-1 is a block diagram of the TPU channel control hardware. This diagram helps clarify the interaction of the event register, control signals, the I/O, and other channel features.

SUMMARY OF CHANGES

This is a revision with complete reprint. All known errors in the publication have been corrected. The following summary lists significant changes. Typographical errors which do not affect content have not been noted.

Section 2 Host Interface

Page 2-2 Table 2-1 all text written as "ms" changed to "μs".

Appendix A TPU Functions

Page A-3 Table A-2 corrected host sequence code for TSM.

Page A-28 Figure A-11 corrected parameter RAM allocation for QDEC.

INDEX

- A -

Access levels, 1-4, 2-4
Arbitration, interrupt, 2-6

- C -

CFSR[3:0], 2-9
Channel
 function select registers, 2-9
 hardware, D-1
 interrupt
 base vector (CIBV), 2-7, 2-8
 enable register (CIER), 2-7, 2-8
 request level, 2-8
 status bit, 2-8
 status register (CISR), 2-7, 2-8
 number priority, C-6
 parameter, RAM, 2-12
 pins, 1-6
 priority registers (CPR0, CPR1), 2-11
CHANNEL[15:0], 2-9
CHANNEL_CONTROL, A-4
CH[15:0], 2-8, 2-10
CIBV, 2-7, 2-8
CIER, 2-7, 2-8
CIRL, 2-8
CISR, 2-7, 2-8
Coherency, 1-4
COMM, A-43
Configuration, TPU, 2-16
 examples, 2-18
Control store, 4-2
CPR0, 2-11
CPR1, 2-11

- D -

DIO, A-22
Discrete input/output (DIO), A-22

- E -

EMU, 2-3, 2-5
Emulation mode, 1-3, 2-3, 2-5, 4-1
 memory map, 4-3
 summary, 4-7
Entry point segment, 4-2

- F -

Fast quadrature decode TPU function (FQD), A-54
First-pass worst case latency analysis, C-8
FQD, A-54
FQM, A-36
Frequency measurement (FQM), A-36
Functions, TPU, A-1
 disabling, 3-7
 library, 1-3, 2-3, 4-7
 states, C-2

- H -

Hall effect decode (HALLD), A-52
Host interface, 1-2
Host sequence registers (HSQR[1:0]), 2-10
Host service request registers (HSRR[1:0]), 2-10

- I -

IARB, 2-6, 2-7
Initialization, TPU, 2-16
Input capture/input transition counter (ITC), A-18
Interrupts, 2-6
 arbitration, 2-7

Signal descriptions, 1–6
SM, A–10
SPWM, A–24
SRL, 3–1, C–6
States, function, 3–1, C–2
Stepper motor (SM), A–10
STF, 2–4, 2–5
STOP, 2–4, 2–5
Stop flag (STF), 2–4, 2–5
Supervisor privilege level, 1–4, 2–4, 2–5
SUPV, 2–5
Synchronized pulse-width modulation (SPWM), A–24

- T -

T2CG, 2–2, 2–4
T2CLK clock pin, 1–2, 1–6
Table stepper motor (TSM), A–33
TBS, A–4
TCR1, 1–2, 2–1
TCR1P, 2–1, 2–5
TCR2, 1–2, 2–2
TCR2 clock/gate control, 2–2, 2–5
TCR2P, 2–5
TICR, 2–8
Time bases, 1–2, A–4
Time slot, 3–1
 latency, 3–6
 sequence, C–5
 transition, C–6
Timer count register one (TCR1), 2–1
 prescaler control, 2–5
Timer count register two (TCR2), 2–2
 prescaler control, 2–5
TPU
 configuration, 2–16
 examples, 2–18
 emulation, 1–3, 2–3, 4–1
 function library, 1–3, 2–3, 4–7
 functions, A–1
 disabling, 3–7
 host interface, 1–2
 interrupt configuration register (TICR), 2–8
 interrupts, 2–6
 memory map, 1–4

 module configuration register (TPUMCR), 2–4
 parameter RAM, 1–3
 pins, 1–6
 registers, 2–1, B–1
TPUMCR, 2–4
TPURAM, 1–3, 2–3
TSM, A–33

- U -

UART, A–38
User privilege level, 1–4, 2–4

- V -

Vectors, interrupt, 2–7

- W -

Worst-case latency (WCL), C–1
 first-pass estimate, C–8
 second pass analysis, C–14
Worst-case service time, C–8

OVERVIEW **1**

HOST INTERFACE **2**

SCHEDULER **3**

TPU EMULATION MODE **4**

TPU FUNCTIONS **A**

REGISTER SUMMARY **B**

ESTIMATING WORST-CASE LATENCY **C**

CHANNEL HARDWARE DIAGRAM **D**

SUMMARY OF CHANGES **S**

INDEX **I**

Literature Distribution Centers:

USA: Motorola Literature Distribution; P.O. Box 20912; Phoenix, Arizona 85036.

EUROPE: Motorola Ltd.; European Literature Centre; 88 Tanners Drive, Blakelands, Milton Keynes, MK14 5BP, England.

JAPAN: Nippon Motorola Ltd.; 4-32-1, Nishi-Gotanda, Shinagawa-ku, Tokyo 141, Japan.

ASIA PACIFIC: Motorola Semiconductors H.K. Ltd.; Silicon Harbour Center, No. 2 Dai King Street, Tai Po Industrial Estate,
Tai Po, N.T., Hong Kong.



MOTOROLA

TPURM/AD

