MOTOROLA
SEMICONDUCTOR
APPLICATION NOTE

Order this document
by AN1200/D

AN1200

Configuring the M68300 Family Time Processing Unit (TPU)

By Al Franceschino

INTRODUCTION

This application note was written to help the design engineer acquire a basic understanding of the operation of the time processing unit. The TPU's basic operation, registers, and parameters are discussed and a sample program is provided.

The TPU is Motorola's latest and most sophisticated embedded peripheral to be added to the M68300 Family of microcontroller units (MCUs). The TPU was designed to offload the core processor from time intensive tasks. With the advent of the TPU the time intensive tasks normally associated with an embedded controller can be performed with minimal processor intervention. Essentially the TPU is a self-contained and complete computer system with its own microengine. There are several preprogrammed time functions residing in the TPU's ROM which the user can call by passing commands and parameters to the TPU through a set of shared registers and dual port RAM. The synthesis of waveforms, capturing pulse duration, and other functions are all microcoded so that only applications-specific parameters need be supplied. The TPU has 16 orthogonal channels each capable of executing any of the supplied time primitives. There are two clocking sources available to the TPU channels, TCR1 and TCR2. TCR1 is an internal clock derived from a prescaled system clock, and TCR2 is an externally supplied clock. Finally, the TPU was designed to interface to Motorola's new intermodule bus (IMB). This bus, with its defined structure and timing, is the opening through which all on-chip peripherals and processors alike communicate. In the M68332 MCU, the TPU, system integration module (SIM), serial modules, and RAM all communicate with the CPU32 (the on-chip host processor) over this bus.

THE TPU PROGRAMMING MODEL

There are a number of memory-mapped registers which require configuration before the TPU can begin operation. In addition to the registers, there are also a number of primitive dependent parameters within the channels which also require configuration. The location of the registers and channel parameters are defined in the SIM's module configuration register (MCR). The state of the modmap bit in this register determines whether the TPU registers and channel parameters are located in the upper portion of the first or second segment of memory. See Table 1 for a bit-level description of the TPU's register set.

freescale™
semiconductor

Table 1. TPU Register Summary

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TMCR $YFFE00	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			
TTCR $YFFE02	0 	0 	0 	INCAD 0	TRC1C 0	ACUTR1-0 0 0		0 	0 	TS OSEL2-0 0 0 0			TS ISE L2-0 0 0 0			TMM 0
DSCR $YFFE04	HOT4 0	0 	0 	0 	0 	BLC 0	CLKS 0	FRZ1-0 0 0		CCL 0	BP 0	BC 0	BH 0	BL 0	BM 0	BT 0
DSSR $YFFE06	0 0	0 0	0 0	0 0	0 0	0 0	0 0	0 0	BKPT 0	PCBK 0	CHBK 0	SRBK 0	TPUF 0	0 0	0 0	0 0
TICR $YFFE08	0 	0 	0 	0 	0 	CHANNEL INTERRUPT REQUEST LEVEL 0 0 0			CHANNEL BASE VECTOR 0 0 0 0				0 	0 	0 	0
CIER $YFFE0A	CHAN 15 0	CHAN 14 0	CHAN 13 0	CHAN 12 0	CHAN 11 0	CHAN 10 0	CHAN 9 0	CHAN 8 0	CHAN 7 0	CHAN 6 0	CHAN 5 0	CHAN 4 0	CHAN 3 0	CHAN 2 0	CHAN 1 0	CHAN 0 0
CFSR0 $YFFE0C	CHANNEL 15 0 0 0 0				CHANNEL 14 0 0 0 0				CHANNEL 13 0 0 0 0				CHANNEL 12 0 0 0 0			
CFSR1 $YFFE0E	CHANNEL 11 0 0 0 0				CHANNEL 10 0 0 0 0				CHANNEL 9 0 0 0 0				CHANNEL 8 0 0 0 0			
CFSR2 $YFFE10	CHANNEL 7 0 0 0 0				CHANNEL 6 0 0 0 0				CHANNEL 5 0 0 0 0				CHANNEL 4 0 0 0 0			
CFSR3 $YFFE12	CHANNEL 3 0 0 0 0				CHANNEL 2 0 0 0 0				CHANNEL 1 0 0 0 0				CHANNEL 0 0 0 0 0			
HSQR0 $YFFE14	CHANNEL 15 0 0		CHANNEL 14 0 0		CHANNEL 13 0 0		CHANNEL 12 0 0		CHANNEL 11 0 0		CHANNEL 10 0 0		CHANNEL 9 0 0		CHANNEL 8 0 0	
HSQR1 $YFFE16	CHANNEL 7 0 0		CHANNEL 6 0 0		CHANNEL 5 0 0		CHANNEL 4 0 0		CHANNEL 3 0 0		CHANNEL 2 0 0		CHANNEL 1 0 0		CHANNEL 0 0 0	
HSRR0 $YFFE18	CHANNEL 15 0 0		CHANNEL 14 0 0		CHANNEL 13 0 0		CHANNEL 12 0 0		CHANNEL 11 0 0		CHANNEL 10 0 0		CHANNEL 9 0 0		CHANNEL 8 0 0	
HSRR1 $YFFE1A	CHANNEL 7 0 0		CHANNEL 6 0 0		CHANNEL 5 0 0		CHANNEL 4 0 0		CHANNEL 3 0 0		CHANNEL 2 0 0		CHANNEL 1 0 0		CHANNEL 0 0 0	
CPR0 $YFFE1C	CHANNEL 15 0 0		CHANNEL 14 0 0		CHANNEL 13 0 0		CHANNEL 12 0 0		CHANNEL 11 0 0		CHANNEL 10 0 0		CHANNEL 9 0 0		CHANNEL 8 0 0	
CPR1 $YFFE1E	CHANNEL 7 0 0		CHANNEL 6 0 0		CHANNEL 5 0 0		CHANNEL 4 0 0		CHANNEL 3 0 0		CHANNEL 2 0 0		CHANNEL 1 0 0		CHANNEL 0 0 0	
CISR $YFFE20	CHAN 15 0	CHAN 14 0	CHAN 13 0	CHAN 12 0	CHAN 11 0	CHAN 10 0	CHAN 9 0	CHAN 8 0	CHAN 7 0	CHAN 6 0	CHAN 5 0	CHAN 4 0	CHAN 3 0	CHAN 2 0	CHAN 1 0	CHAN 0 0

Y = m111, where m is the state of the modmap bit in the module configuration register of the system integration module (Y = $7 or $F). (For information on the system integration module, refer to the MC68332 user's manual (formerly *MC68332 SIM User's Manual*), document number MC68332UM/AD).

REGISTER DESCRIPTION

A brief description of the more commonly used registers is given below. For a more detailed description of these registers consult the TPU Reference Manual (TPURM/AD). The register descriptions supplied herein can be used as a reference while reviewing the sample program.

The module configuration register contains the following bits:

STOP — if this is asserted it causes both clock sources to be shut down.

TCR1 prescaler — this is one of two prescalers applied to TCR1. Settings available are divide by 1, 2, 4, and 8.

TCR2 prescaler — the only prescaler applied to TCR2 incoming clock. Settings available are divide by 1, 2, 4, and 8.

EMU — this bit controls TPU emulation. When set, the TPU executes code out of internal RAM only. This precludes RAM use by other modules on the IMB.

T2CG — this bit determines whether the TCR2 pin functions as an external clock source or as a gate of the system clock divided by 8.

STF — this bit is the stop flag indicating that the stop bit has been set and that the TPU will enter the stop state described above.

SUPV — this bit is set to allow only supervisor mode access to TPU registers.

PSCK — this bit selects either a divide by 4 or divide by 32 clock as input to the TCR1 prescaler. (The divide by 4 is available in Rev. G parts and above.)

IARB — this field determines the interrupt arbitration priority assigned to the TPU in the event that simultaneous interrupts are generated by multiple modules on the IMB.

The test/development support registers and their usage are beyond the scope of this paper.

The interrupt configuration register consists of two bit fields. The first is the channel interrupt request level which specifies the interrupt level for all TPU channels. Levels zero through seven are supported. A level seven interrupt is nonmaskable. Level zero indicates no interrupts passed on to the IMB host (CPU32).

The second is the channel interrupt base vector which specifies the upper nibble of the supplied interrupt vector. The lower nibble of the vector is equal to the interrupting channel number.

The 16 bits of the channel interrupt enable register allow for channel interrupt enable/disable on a channel by channel basis. Channel interrupts are enabled if the corresponding bit is set.

The channel interrupt status register has 16 bits which indicate when a particular TPU channel has an interrupt pending. Setting a bit in this register along with setting the appropriate bit in the channel interrupt enable register will cause an interrupt, if the interrupt request level for the TPU is not zero.

There are four channel function select registers. Four of the 16 TPU channels are specified in each of the select registers by a 4-bit field. Eight preprogrammed (microcoded into the TPU ROM) time primitives (functions) available to the user are selected in this 4-bit field. These are the only functions normally available to the user. They are:

DIO – discrete I/O	function code $8
ITC – input capture/input transition counter	function code $A
OC – output compare	function code $E
PWM – pulse width modulation	function code $9
PMA/PMM – period measurement/additional/missing pulse detection	function code $B
PSP – position-synchronized pulse generator	function code $C
SM – stepper motor	function code $D
PPWA – period/pulse accumulator	function code $F

The functionality of the host sequence registers is time-primitive dependent. Each channel in the TPU occupies two bits in these registers. These bits affect channel operation by changing the flow of microcode (program flow) through the TPU while it is running in a particular channel. As an example, if a channel is set up to run the input capture function, the host sequence field can modify channel operation such that it does one of the following:

a) count a specified number of transitions (events) and stop.
b) count events continuously (no time limit).
c) count a specified number of events, generate a link signal to another channel, and stop.
d) continuously count events and issue a link signal to another channel each time a specified number of events is counted. The specified count for this function is contained in a parameter called MAX-COUNT.

NOTE

A link is a signal to another channel which causes that channel to perform some function of its own (assuming it has been configured properly). The capability of generating links is function dependent. Some functions can generate a link, others receive a link.

There are also two host service request registers. These registers are divided into fields of two bits each, one for each TPU channel. The functionality of these bits is also time-primitive dependent. For example — these fields could direct a channel to initialize based on preprogrammed parameter values, or force an output high or low upon completion of the request. The host writes this field to activate a channel. The channel scheduler within the TPU then schedules the particular channel for service.

The channel priority registers specify the relative priority of a channel. The 2-bit field for each channel allows the channel to be high, middle, low, or no priority (disabled). There are seven time slots available for channel service. High priority channels get four of the seven slots, middle level priority channels get two of the seven time slots, and low level priority channels get one slot in seven. Thus low priority channels cannot be permanently held off from service.

The channel parameter register (RAM) can be thought of as a mail box through which the host processor and the TPU communicate. Every channel has at least six RAM locations that are each 16 bits wide. Channels fourteen and fifteen have two additional parameters. The base address of a channel parameter is equal to $YFFFX0, where y = modmap bit setting (7 or F) and where x = the channel number as shown below in Figure 1. These parameters will be reviewed in more detail in the next section.

Channel	Parameter RAM Addresses							
	0	1	2	3	4	5	6	7
0	PRAM0.0 $YFFF00	PRAM0.1 $YFFF02	PRAM0.2 $YFFF04	PRAM0.3 $YFFF06	PRAM0.4 $YFFF08	PRAM0.5 $YFFF0A	— —	— —
1	PRAM1.0 $YFFF10	PRAM1.1 $YFFF12	PRAM1.2 $YFFF14	PRAM1.3 $YFFF16	PRAM1.4 $YFFF18	PRAM1.5 $YFFF1A	— —	— —
2	PRAM2.0 $YFFF20	PRAM2.1 $YFFF22	PRAM2.2 $YFFF24	PRAM2.3 $YFFF26	PRAM2.4 $YFFF28	PRAM2.5 $YFFF2A	— —	— —
3	PRAM3.0 $YFFF30	PRAM3.1 $YFFF32	PRAM3.2 $YFFF34	PRAM3.3 $YFFF36	PRAM3.4 $YFFF38	PRAM3.5 $YFFF3A	— —	— —
4	PRAM4.0 $YFFF40	PRAM4.1 $YFFF42	PRAM4.2 $YFFF44	PRAM4.3 $YFFF46	PRAM4.4 $YFFF48	PRAM4.5 $YFFF4A	— —	— —
5	PRAM5.0 $YFFF50	PRAM5.1 $YFFF52	PRAM5.2 $YFFF54	PRAM5.3 $YFFF56	PRAM5.4 $YFFF58	PRAM5.5 $YFFF5A	— —	— —
6	PRAM6.0 $YFFF60	PRAM6.1 $YFFF62	PRAM6.2 $YFFF64	PRAM6.3 $YFFF66	PRAM6.4 $YFFF68	PRAM6.5 $YFFF6A	— —	— —
7	PRAM7.0 $YFFF70	PRAM7.1 $YFFF72	PRAM7.2 $YFFF74	PRAM7.3 $YFFF76	PRAM7.4 $YFFF78	PRAM7.5 $YFFF7A	— —	— —
8	PRAM8.0 $YFFF80	PRAM8.1 $YFFF82	PRAM8.2 $YFFF84	PRAM8.3 $YFFF86	PRAM8.4 $YFFF88	PRAM8.5 $YFFF8A	— —	— —
9	PRAM9.0 $YFFF90	PRAM9.1 $YFFF92	PRAM9.2 $YFFF94	PRAM9.3 $YFFF96	PRAM9.4 $YFFF98	PRAM9.5 $YFFF9A	— —	— —
10	PRAMA.0 $YFFFA0	PRAMA.1 $YFFFA2	PRAMA.2 $YFFFA4	PRAMA.3 $YFFFA6	PRAMA.4 $YFFFA8	PRAMA.5 $YFFFAA	— —	— —
11	PRAMB.0 $YFFFB0	PRAMB.1 $YFFFB2	PRAMB.2 $YFFFB4	PRAMB.3 $YFFFB6	PRAMB.4 $YFFFB8	PRAMB.5 $YFFFBA	— —	— —
12	PRAMC.0 $YFFFC0	PRAMC.1 $YFFFC2	PRAMC.2 $YFFFC4	PRAMC.3 $YFFFC6	PRAMC.4 $YFFFC8	PRAMC.5 $YFFFCA	— —	— —
13	PRAMD.0 $YFFFD0	PRAMD.1 $YFFFD2	PRAMD.2 $YFFFD4	PRAMD.3 $YFFFD6	PRAMD.4 $YFFFD8	PRAMD.5 $YFFFDA	— —	— —
14	PRAME.0 $YFFFE0	PRAME.1 $YFFFE2	PRAME.2 $YFFFE4	PRAME.3 $YFFFE6	PRAME.4 $YFFFE8	PRAME.5 $YFFFEA	PRAME.6 $YFFFEC	PRAME.7 $YFFFEE
15	PRAMF.0 $YFFFF0	PRAMF.1 $YFFFF2	PRAMF.2 $YFFFF4	PRAMF.3 $YFFFF6	PRAMF.4 $YFFFF8	PRAMF.5 $YFFFFA	PRAMF.6 $YFFFFC	PRAMF.7 $YFFFFE

— = Not implemented.
Y = m111, where m is the modmap signal state on the IMB, which reflects the state of the modmap in the module configuration register of the system integration module (Y = $7 or $F).

Figure 1. Parameter RAM MAP

AN EXAMPLE APPLICATION

Consider the following example: A controller board utilizing the MC68332 is being designed with a synthesized constant-frequency (32.552 kHz) waveform. Additionally at least one other sub-multiple frequency, synchronized to the original waveform, must be generated. How many channels of TPU are required? How are these channels properly configured for the task?

Solution

GENERAL INFORMATION

The PWM function is ideal for generating output frequencies with either a fixed or variable duty cycle.

The PPWA function can be used either to accumulate the high time of a pulse or the period of the pulse. This function can therefore accumulate data in the form of clock (TCR1 or TCR2) "ticks" as to the period of the waveform. It also offers the added functionality of being able to link synchronously to another channel.

The OC function can be used to generate a waveform that is referenced on the PPWA channel.

PWM PARAMETER RAM

The PWM function uses five of the six channel parameters. Three of these parameter locations are updated by the host CPU prior to initialization. These are:

CHANNEL_CONTROL at offset channel address + 01 (used to configure the channel as an output).

PWMHI at offset channel address + 04 (used to control the high time of the pulse).

PWMPER at offset channel address + 06 (used to specify the waveform's overall period).

PPWA PARAMETER RAM

This function uses all six of the channel parameters. Five of these parameter locations are updated by the host CPU prior to initialization. These are:

START_LINK_CHANNEL at offset channel address + 0 (used to specify the first channel number with which to link).

LINK_CHANNEL_COUNT at offset channel address + 0 (used to specify how many channels are being linked).

CHANNEL_CONTROL at offset channel address + 0 (used to direct the channel to accumulate periods or high time as well as which TCR to capture and match).

MAX_CNT at offset channel address + 02 (used to specify how many periods or high pulses to accumulate).

ACCUM_RATE at offset channel address + 08 (used to specify the real-time update rate for para-menter accum). For those applications where a real-time accumulation value is not required, this parameter should be set to $FF. This value causes the least amount of service requests from the microengine between period accumulations, which yields better channel synchronization (no edge jitter).

OUTPUT COMPARE PARAMETER RAM

This function uses all six of the normally supplied parameters, plus has the option of using two additional parameters when configured in host mode. These six parameter locations are updated by the host CPU. They are:

CHANNEL_CONTROL at offset channel address + 0, used to specify the toggling action of the output after a link, as well as which TCR to use when calculating the next toggle time.

OFFSET at offset channel address + 02 is calculated by the TPU in continuous mode.

RATIO at offset channel address + 04 is a fractional value used by the channel to scale the incoming accumulated period. Thus the synchronized frequency can be less than or greater than the original PWM reference frequency.

REF_ADDR1 at offset channel address + 04 is a pointer to a value in another channel, namely LAST_ACCUM in the PPWA channel, which is used in all TPU calculations subsequent to the first link. When a link occurs, LAST_ACCUM is presented to the output compare channel and is the captured TCR count. This is used as a reference point in the calculation for the next transition time.

REF_ADDR2 at offset channel address + 06 is a pointer to a value in another channel, namely PPWA_LW in the PPWA channel, which is used in all TPU calculations. When a link occurs, PPWA_LW is presented to this channel through parameter REF_ADDR2. The information contained in this parameter represents the total number of TCR "ticks" during the accumulated period(s). This information is multiplied by RATIO to form the parameter OFFSET (the next transition time).

REF_ADDR3 at offset channel address + 06 has the same functionality as REF_ADDR1 except that it is only used as a reference on the first link to the channel. All subsequent links to this channel use REF_ADDR1.

SUGGESTED CHANNEL SETUP

The procedure for configuring the TPU channels is described below. For this application the channel numbers have been arbitrarily chosen. Interrupts on all channels are disabled.

Step 1 — period analysis: The M68332 is running on a 16.67 MHz clock. This is a 60 ns period. By configuring the module configuration register for TCR1 running internal clock/32 and setting parameter PWMPER to $10, a 30.72 μs period can be obtained on a channel running the PWM function. To obtain a 50% duty cycle on this channel, an 8 is written to parameter PWMHI.

$$(60\ \text{ns} \times 32 \times 16 = 30.72\ \mu\text{s})$$

Step 2 — Configure another channel for the PPWA function and hardwire this channel to the PWM channel. Writing function code $F to the function select register of the channel will activate the PPWM function. This channel should be configured to accumulate the period (in TCR1 clock ticks) of the PWM channel. This is accomplished by writing $010B to the CHANNEL_CONTROL register. This channel is set up to link to the output compare channel. Parameter MAX_CNT defines how many periods will be accumulated (counted) before the link will take place. This effectively produces the divide by (n) synchronous counter that will be the source of the other frequencies to be produced.

Step 3 — Configure a third channel for the output compare function. This is done by writing function code $E to the function select register of the selected channel. This channel should run in continuous mode, with its parameter RAM pointers set looking into the PPWA channel as described above.

Sample Program

```
*****************************************************************************
*              Program:       Sync_Waveform
*              Author:        Al Franceschino
*              Date:          6/25/90
*              Rev.           1.0
*
* TPU init routine to set up ch. 0 as a continuous output. Interrupts are disabled.
*
                   org $9000
MCR                EQU $FFFE00              MODULE CONFIG. REG
ICR                EQU $FFFE08              INTERRUPT CONFIG. REG
CIE                EQU $FFFE0A              CH. INTERRUPT ENABLE REG
CFS3               EQU $FFFE12              CH0-3 FUNCTION SELECT REG
HSEQ7              EQU $FFFE16              CH0-7 HOST SEQUENCE REG
HSYR7              EQU $FFFE1A              CH0-7 HOST SERVICE REQ. REG
CHPR7              EQU $FFFE1E              CH0-7 PRIORITY REG
CHINTS             EQU $FFFE20              CH0-15 INTERRUPT STATUS REG
CHOC               EQU $FFFF00              CH0 CONTROL REG
CHOMXC             EQU $FFFF04              CH0 RATIO + REF_ADD_1
CHORA2             EQU $FFFF06              CH0 REF_ADD_2,3
*
*
START              MOVE.W #$0007,MCR        TCR1 2US,TCR2 DIRECT,
*                                           NO EMUL,USER OK,SYSCLK/32
                   MOVE.W #0,CIE            DISABLE INTERRUPTS ON ALL
*                                           CHANNELS
*   THIS SECTION OF CODE CONFIGURES CHANNEL 0 FOR THE OUTPUT COMPARE FUNCTION
*
                   MOVE.W CFS3,D0           GET FUNCTIONS SELECTED
                   ORI.W #$000E,D0
                   MOVE.W D0,CFS3           SELECT OC FOR CH0
                   MOVE.W #$003F,CHPR7      SET CH0-2 = HIGHEST PRIORITY
                   MOVE.W #$008F,CHOC       TOGGLE ON LINK,MATCH TCR1
                   MOVE.B #$FF,CHOMXC       SET RATIO APPROX = 1,REF_ADD
                   MOVE.B #$24,CHOMXC+1     1 POINTS TO LAST_ACCUM
                   MOVE.W #$2A24,CHORA2     REF_ADD_2 POINTS TO PPWA_
*                                           LW,REF_ADD_3 POINTS TO
*                                           LAST_ACCUM
GET_STAT           MOVE.W HSVR7,D4          SEE IF ANY PENDING COMMANDS
                   ANDI.W #$03,D4           TO EXECUTE
                   BNE GET_STAT             IF SO WAIT UNTIL FINISHED
                   MOVE.W HSVR7,D0
                   ORI.W #$0003,D0          QUE UP COMMAND
                   MOVE.W D0,HSVR7          ISSUE HOST SERVICE REQUEST
*
```

```
*   THIS SECTION OF CODE CONFIGURES CHANNEL 2 FOR THE PPWA FUNCTION. CHANNEL
*   2 IS SET UP TO ACCUMULATE TCR1 "TICKS". A LOOP BACK SHOULD BE INSTALLED
*   BETWEEN THIS CHANNEL AND CHANNEL 1 (PWM CHANNEL). CHANNEL 2 LINKS TO
*   CHANNEL 0 RUNNING THE OUTPUT COMPARE FUNCTION.
*
CH2C                EQU $FFFF20             CH 2 CONTROL + LINK REGISTER
CH2MXC              EQU $FFFF22             CH 2 MAX COUNT REGISTER
CH2ACRT             EQU $FFFF28             CH 2 ACCUM_RATE REGISTER
*
START1              MOVE.W CFS3,D0          GET ACTIVATED FUNCTIONS

                    ORI.W #$0F00,D0         SELECT PPWA FUNCTION
                    MOVE.W D0,CFS3
                    MOVE.W HSEQ7,D0         GET EXISTING HOST SEQ. BITS
                    ORI.W #$0010,D0
                    MOVE.W DO,HSEQ7         MAKE CH LINK AND ACCUM UP
*                                           TO 16 BIT OF PERIOD.
*                                           PERIODS,NO FORCE PIN STATE
                    MOVE.W #$010B,CH2C      CAPTURE PERIODS,NO FORCE
*                                           PIN STATE
                    MOVE.B #$02,CH2MXC      SET CH TO DIVIDE PWM PERIOD
*                                           BY 2
                    MOVE.B #$FF,CH2ACRT     SET ACCUM_RATE TO APPROX
*                                           510 µs.
GET_STAT1           MOVE.W HSVR7,D0         ANY PENDING COMMANDS ?
                    ANDI.W #$0010,D0
                    BNE GET_STAT1           IF SO WAIT TILL FINISHED
                    MOVE.W HSVR7,D0
                    ORI.W #$0020,D0
                    MOVE.W D0,HSVR7         ISSUE HOST SERVICE REQUEST
*                                           FOR CHANNEL TO START PPWA
*
*   THIS SECTION OF CODE CONFIGURES CHANNEL 1 FOR THE PWM FUNCTION. A
*   50% DUTY CYCLE PULSE WITH A PERIOD OF APPROX. 31 µs IS GENERATED
*   BY SETTING PARAMETER PWMHI = 8 AND PWMPER = $10.
*
CH1C                EQU $FFFF10             CH 1 CONTROL REGISTER
PWMHI               EQU $FFFF14             CH 1 PWM HI PARAMETER
PWMPER              EQU $FFFF16             CH 1 PWM PERIOD PARAMETER
*
START2              MOVE.W CFS3,D0          GET CURRENT FUNCTIONS
                    ORI.W #$0090,D0
                    MOVE.W D0,CFS3          SELECT PWM FUNCTION
                    MOVE.W #$0090,CH1C      MATCH (USE) TCR1 FOR TIMING
                    MOVE.W #$0010,PWMPER    SET PERIOD = (16)X1.92 µs
                    MOVE.W #$0008,PWMHI     SET DUTY CYCLE = 50%
GET_STAT2           MOVE.W HSVR7,D0         ANY COMMANDS PENDING ?
                    ANDI.W #$08,D0
                    BNE GET_STAT2           IF SO WAIT TILL FINISHED
                    MOVE.W HSVR7,D0
                    ORI.W #$0008,D0         DON'T AFFECT OTHER REQUESTS
                    MOVE.W D0,HSVR7         START PWM FUNCTION
                    TRAP #15
                    DC.W $0063              332 EVM SYSTEM CALL
                    END
```

CONCLUSION

The program above demonstrates the ease with which the TPU can be configured to perform a variety of functions. Several additional functions can be executed concurrently with the ones provided, after the associated channels have been configured. In embedded control applications where there are many time-critical functions, the M68332 and its TPU can be an invaluable resource. The preprogrammed functions and the ability to customize time primitives makes the M68300 Family TPU a viable alternative to discrete multichip control.

This page intentionally left blank.

How to Reach Us:

Home Page:
www.freescale.com

E-mail:
support@freescale.com

USA/Europe or Locations Not Listed:
Freescale Semiconductor
Technical Information Center, CH370
1300 N. Alma School Road
Chandler, Arizona 85224
+1-800-521-6274 or +1-480-768-2130
support@freescale.com

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

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

Asia/Pacific:
Freescale Semiconductor Hong Kong Ltd.
Technical Information Center
2 Dai King Street
Tai Po Industrial Estate
Tai Po, N.T., Hong Kong
+800 2666 8080
support.asia@freescale.com

For Literature Requests Only:
Freescale Semiconductor Literature Distribution Center
P.O. Box 5405
Denver, Colorado 80217
1-800-441-2447 or 303-675-2140
Fax: 303-675-2150
LDCForFreescaleSemiconductor@hibbertgroup.com

freescale™
semiconductor

A30176 PRINTED IN USA (1994) MPS/POD MCU