Discrete Input/Output TPU Function (DIO)

by Charles Melear

1 Functional Overview

The discrete input/output (DIO) function allows the user to configure a time processor unit (TPU) channel as an input or output. As an input, the channel can be read at any time or sampled at a periodic rate. As an output, the channel can be driven high or low upon command by the CPU.

A parameter RAM location, PIN_LEVEL, is used to record the 16 most recent states of the TPU channel pin. The programmer may choose one of the four following conditions to update the parameter: 1) when a transition (positive, negative, or either) occurs, 2) when the CPU makes a request to read the logical value driving the pin, 3) when the CPU makes a request to drive the pin to a specified logical value or 4) at a periodic rate specified in the MATCH_RATE register.

2 Detailed Description

The DIO function allows a TPU channel pin to emulate a discrete input or output pin. As an input, the pin can be read either on command or at a periodic rate. As an output, the pin can be driven high or low on command. The DIO function can be used in the following ways:

1. A TPU channel pin can be configured as an output and programmed to update on a host service request. An HSR %10 outputs a low level onto the pin; an HSR %01 outputs a high level. Both types of requests update bit 15 of PIN_LEVEL with the pin level driven on the pin. Each time a host service request is issued to write a new value to the pin, the contents of the PIN_LEVEL register are shifted to the right by one bit and the new level on the TPU channel is written into bit 15 of that register. Note that if 16 consecutive commands were issued to drive a TPU channel high, the PIN_LEVEL register would contain $FFFF. Likewise, if 16 consecutive commands were issued to write a TPU channel alternately high and low, the PIN_LEVEL register would contain $5555. The host sequence bits are not used for HSRs of %10 and %01.

2. A TPU channel can be configured as an input and programmed to update on positive transitions only, negative transitions only, or all transitions. This mode is entered by setting the host sequence bits to %00 and issuing an HSR %11 (initialization).

Transition mode is normally used with the PAC field set to detect either transition. After 16 selected transitions occur, the PIN_LEVEL register contains $FFFF (positive edges only), $0000 (negative edges only) or $5555 or $AAAA (both positive and negative edges). The update of the PIN_LEVEL register occurs any time a selected transition occurs. The time of the update is not recorded.

3. A TPU channel can be configured as an input and programmed to update at match rate by setting the host sequence bits to %01 and then issuing an HSR %11. The match rate is specified in TPU clock cycles; either timer count register 1 (TCR1) or timer count register 2 (TCR2) can be selected. At the rate specified in the MATCH_RATE register, the contents of the associated PIN_LEVEL register are shifted to the right by one place and the logic level of the TPU channel is loaded into bit 15 of the PIN_LEVEL register. Match mode operation is normally used with the input pin configured by PAC to ignore transitions.

Match mode always uses TCR1 to set up the first compare interval during initialization even if TCR2 is specified in the TBS field of the channel control register. Subsequent matches occur at the TCR2 rate as specified by MATCH_RATE.
4. A TPU channel can be configured as an input and programmed to update on a host service request. This mode is entered by setting the host sequence bits to %00 and issuing an HSR %11 (initialization). To read the pin, set the host sequence bits to %10 and then issue an HSR %11 (initialization). Each time the appropriate HSR is issued the contents of the associated PIN_LEVEL register are shifted to the right by one place and the logic level of the TPU channel is loaded into bit 15 of the PIN_LEVEL register.

If a host service request for initialization coincides with a scheduled request by a match or transition, the host request receives priority and the other request is ignored. HSR %11 should not be used when the host sequence bits are also %11, as errors result.

3 Function Code Size

Total TPU function code size determines what combination of functions can fit into a given ROM or emulation memory microcode space. DIO function code size is:

12 μ instructions + 7 entries = 19 long words

4 DIO Function Parameters

This section provides detailed descriptions of discrete input/output function parameters stored in channel parameter RAM. Figure 1 shows TPU parameter RAM address mapping. Figure 2 shows the parameter RAM assignment used by the DIO function. In the diagrams, $Y = M111$, where M is the value of the module mapping bit (MM) in the system integration module configuration register ($Y = \$7$ or $\$F$).

<table>
<thead>
<tr>
<th>Channel Number</th>
<th>Base Address</th>
<th>Parameter Address</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>$YFFF##</td>
<td>00</td>
</tr>
<tr>
<td>1</td>
<td>$YFFF##</td>
<td>10</td>
</tr>
<tr>
<td>2</td>
<td>$YFFF##</td>
<td>20</td>
</tr>
<tr>
<td>3</td>
<td>$YFFF##</td>
<td>30</td>
</tr>
<tr>
<td>4</td>
<td>$YFFF##</td>
<td>40</td>
</tr>
<tr>
<td>5</td>
<td>$YFFF##</td>
<td>50</td>
</tr>
<tr>
<td>6</td>
<td>$YFFF##</td>
<td>60</td>
</tr>
<tr>
<td>7</td>
<td>$YFFF##</td>
<td>70</td>
</tr>
<tr>
<td>8</td>
<td>$YFFF##</td>
<td>80</td>
</tr>
<tr>
<td>9</td>
<td>$YFFF##</td>
<td>90</td>
</tr>
<tr>
<td>10</td>
<td>$YFFF##</td>
<td>0A</td>
</tr>
<tr>
<td>11</td>
<td>$YFFF##</td>
<td>B0</td>
</tr>
<tr>
<td>12</td>
<td>$YFFF##</td>
<td>C0</td>
</tr>
<tr>
<td>13</td>
<td>$YFFF##</td>
<td>D0</td>
</tr>
<tr>
<td>14</td>
<td>$YFFF##</td>
<td>E0</td>
</tr>
<tr>
<td>15</td>
<td>$YFFF##</td>
<td>F0</td>
</tr>
</tbody>
</table>

— = Not Implemented (reads as $00$)

Figure 1 TPU Channel Parameter RAM CPU Address Map
4.1 CHANNEL_CONTROL

The CPU should write CHANNEL_CONTROL prior to issuing an initialization HSR. The CHANNEL_CONTROL parameter configures the PSC, PAC, and TBS fields. The PSC field is not used by the DIO function and should be set to “no change”. The PAC field specifies the pin logic response for a timer channel input. PAC should be set to “detect either edge” for transition mode and to “no transitions detected” for match mode. For discrete input, the TBS field selects the time base to be used for MATCH_RATE comparisons. (TCR1 is recommended.) For discrete output, this field is a “don't care”. The following table defines the allowable data for this parameter.

<table>
<thead>
<tr>
<th>TBS</th>
<th>PAC</th>
<th>PSC</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>8 7 6 5</td>
<td>4 3 2</td>
<td>1 0</td>
<td>Input</td>
</tr>
<tr>
<td></td>
<td></td>
<td>11</td>
<td>Do Not Force</td>
</tr>
<tr>
<td>0 0 0</td>
<td></td>
<td></td>
<td>Do Not Detect Transition</td>
</tr>
<tr>
<td>0 0 1</td>
<td></td>
<td></td>
<td>Detect Rising Edge</td>
</tr>
<tr>
<td>0 1 0</td>
<td></td>
<td></td>
<td>Detect Falling Edge</td>
</tr>
<tr>
<td>0 1 1</td>
<td></td>
<td></td>
<td>Detect Either Edge</td>
</tr>
<tr>
<td>1 x x</td>
<td></td>
<td></td>
<td>Do Not Change PAC</td>
</tr>
<tr>
<td>0 0 x x</td>
<td></td>
<td></td>
<td>Input Channel</td>
</tr>
<tr>
<td>0 0 0 0</td>
<td></td>
<td></td>
<td>Capture TCR1, Match TCR1</td>
</tr>
<tr>
<td>0 0 0 1</td>
<td></td>
<td></td>
<td>Capture TCR1, Match TCR2</td>
</tr>
<tr>
<td>0 0 1 0</td>
<td></td>
<td></td>
<td>Capture TCR2, Match TCR1</td>
</tr>
<tr>
<td>0 0 1 1</td>
<td></td>
<td></td>
<td>Capture TCR2, Match TCR2</td>
</tr>
<tr>
<td>1 x x x</td>
<td></td>
<td></td>
<td>Do Not Change TBS</td>
</tr>
</tbody>
</table>

W = Channel number
Parameter Write Access:

- Written by CPU
- Written by TPU
- Written by CPU and TPU
- Unused parameters

Figure 2 DIO Function Parameter RAM Assignment
4.2 PIN_LEVEL

PIN_LEVEL indicates the 16 most recent pin values, with the most recent pin value contained in bit 15 and the least recent pin value contained in bit 0. The TPU writes this parameter when a specified transition occurs, or a match or host request to read the pin state occurs, depending on the mode of operation. When updated, the 16 most recent pin values are shifted right by one and the most recent pin value is placed into bit 15. The pin value contained in bit 0 before the right shift is lost after the data is shifted right by one.

4.3 MATCH_RATE

MATCH_RATE indicates the rate, expressed in cycles of the selected TCR, at which the pin value is recorded in PIN_LEVEL when the channel is executing match mode operation.

5 Host Interface to Function

This section provides information concerning the TPU host interface to the DIO function. Figure 3 is a TPU address map. Detailed TPU register diagrams follow the figure. In the diagrams, Y = M111, where M is the value of the module mapping bit (MM) in the system integration module configuration register (Y = $7 or $F).

<table>
<thead>
<tr>
<th>Address</th>
<th>15</th>
<th>8</th>
<th>7</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>$YFFE00</td>
<td></td>
<td></td>
<td></td>
<td>TPU MODULE CONFIGURATION REGISTER (TPUMCR)</td>
</tr>
<tr>
<td>$YFFE02</td>
<td></td>
<td></td>
<td></td>
<td>TEST CONFIGURATION REGISTER (TCR)</td>
</tr>
<tr>
<td>$YFFE04</td>
<td></td>
<td></td>
<td></td>
<td>DEVELOPMENT SUPPORT CONTROL REGISTER (DSCR)</td>
</tr>
<tr>
<td>$YFFE06</td>
<td></td>
<td></td>
<td></td>
<td>DEVELOPMENT SUPPORT STATUS REGISTER (DSSR)</td>
</tr>
<tr>
<td>$YFFE08</td>
<td></td>
<td></td>
<td></td>
<td>TPU INTERRUPT CONFIGURATION REGISTER (TICR)</td>
</tr>
<tr>
<td>$YFFE0A</td>
<td></td>
<td></td>
<td></td>
<td>CHANNEL INTERRUPT ENABLE REGISTER (CIER)</td>
</tr>
<tr>
<td>$YFFE0C</td>
<td></td>
<td></td>
<td></td>
<td>CHANNEL FUNCTION SELECTION REGISTER 0 (CFSR0)</td>
</tr>
<tr>
<td>$YFFE0E</td>
<td></td>
<td></td>
<td></td>
<td>CHANNEL FUNCTION SELECTION REGISTER 1 (CFSR1)</td>
</tr>
<tr>
<td>$YFFE10</td>
<td></td>
<td></td>
<td></td>
<td>CHANNEL FUNCTION SELECTION REGISTER 2 (CFSR2)</td>
</tr>
<tr>
<td>$YFFE12</td>
<td></td>
<td></td>
<td></td>
<td>CHANNEL FUNCTION SELECTION REGISTER 3 (CFSR3)</td>
</tr>
<tr>
<td>$YFFE14</td>
<td></td>
<td></td>
<td></td>
<td>HOST SEQUENCE REGISTER 0 (HSQR0)</td>
</tr>
<tr>
<td>$YFFE16</td>
<td></td>
<td></td>
<td></td>
<td>HOST SEQUENCE REGISTER 1 (HSQR1)</td>
</tr>
<tr>
<td>$YFFE18</td>
<td></td>
<td></td>
<td></td>
<td>HOST SERVICE REQUEST REGISTER 0 (HSRR0)</td>
</tr>
<tr>
<td>$YFFE1A</td>
<td></td>
<td></td>
<td></td>
<td>HOST SERVICE REQUEST REGISTER 1 (HSRR1)</td>
</tr>
<tr>
<td>$YFFE1C</td>
<td></td>
<td></td>
<td></td>
<td>CHANNEL PRIORITY REGISTER 0 (CPR0)</td>
</tr>
<tr>
<td>$YFFE1E</td>
<td></td>
<td></td>
<td></td>
<td>CHANNEL PRIORITY REGISTER 1 (CPR1)</td>
</tr>
<tr>
<td>$YFFE20</td>
<td></td>
<td></td>
<td></td>
<td>CHANNEL INTERRUPT STATUS REGISTER (CISR)</td>
</tr>
<tr>
<td>$YFFE22</td>
<td></td>
<td></td>
<td></td>
<td>LINK REGISTER (LR)</td>
</tr>
<tr>
<td>$YFFE24</td>
<td></td>
<td></td>
<td></td>
<td>SERVICE GRANT LATCH REGISTER (SGLR)</td>
</tr>
<tr>
<td>$YFFE26</td>
<td></td>
<td></td>
<td></td>
<td>DECODED CHANNEL NUMBER REGISTER (DCNR)</td>
</tr>
</tbody>
</table>

Figure 3 TPU Address Map
### CIER — Channel Interrupt Enable Register

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
</tr>
<tr>
<td>13</td>
<td>12</td>
</tr>
<tr>
<td>11</td>
<td>10</td>
</tr>
<tr>
<td>9</td>
<td>8</td>
</tr>
<tr>
<td>7</td>
<td>6</td>
</tr>
<tr>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CH</th>
<th>Interrupt Enable</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Channel interrupts disabled</td>
</tr>
<tr>
<td>1</td>
<td>Channel interrupts enabled</td>
</tr>
</tbody>
</table>

### CFSR[0:3] — Channel Function Select Registers

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
</tr>
<tr>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
</tr>
<tr>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CFS</th>
<th>$YFFE0C – $YFFE12</th>
</tr>
</thead>
<tbody>
<tr>
<td>CFS (CH 15, 11, 7, 3)</td>
<td>CFS (CH 14, 10, 6, 2)</td>
</tr>
<tr>
<td>CFS (CH 13, 9, 5, 1)</td>
<td>CFS (CH 12, 8, 4, 0)</td>
</tr>
</tbody>
</table>

### HSQR[0:1] — Host Sequence Registers

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
</tr>
<tr>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
</tr>
<tr>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CH[15:0]</th>
<th>Action Taken</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>Transition mode: record pin on transition</td>
</tr>
<tr>
<td>01</td>
<td>Match mode: record pin at MATCH_RATE</td>
</tr>
<tr>
<td>10</td>
<td>Record pin state on HSR%11</td>
</tr>
<tr>
<td>11</td>
<td>Indeterminate operation</td>
</tr>
</tbody>
</table>

### HSRR[0:1] — Host Service Request Registers

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
</tr>
<tr>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
</tr>
<tr>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CH[15:0]</th>
<th>Initialization</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>No host service (reset condition)</td>
</tr>
<tr>
<td>01</td>
<td>Force high output</td>
</tr>
<tr>
<td>10</td>
<td>Force low output</td>
</tr>
<tr>
<td>11</td>
<td>Initialize as per host sequence bits</td>
</tr>
</tbody>
</table>

### CPR[1:0] — Channel Priority Registers

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
</tr>
<tr>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
</tr>
<tr>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CH[15:0]</th>
<th>Channel Priority</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>Disabled</td>
</tr>
<tr>
<td>01</td>
<td>Low</td>
</tr>
<tr>
<td>10</td>
<td>Middle</td>
</tr>
<tr>
<td>11</td>
<td>High</td>
</tr>
</tbody>
</table>
6 Function Configuration

For discrete input, the host CPU initializes the channel by:

1. Writing parameters CHANNEL_CONTROL and MATCH_RATE;
2. Writing host sequence bits to configure transition or match mode, as desired;
3. Writing host service request bits to request initialization (%11).

The TPU then executes initialization and accepts an input transition type specified by the PAC field in CHANNEL_CONTROL, or samples the state of the pin at the rate specified by MATCH_RATE. The CPU should monitor the HSR register until the TPU clears the service request to %00 before changing any parameters or before issuing a new service request to this channel.

For discrete output, the host CPU initializes the channel by the following:

1. Writing %01 to the HSR bits; causing the TPU to output a high level to the pin; or,
2. Writing %10 to the HSR bits; causing the TPU to output a low level to the pin.

No other initialization is required.

For all modes of discrete input, once initialized, and discrete output, configuring the host sequence bits to %10 and issuing %11 to the HSR bits causes the TPU to read the pin level and to record the level read in bit 15 of PIN_LEVEL. In the case of discrete output, the pin level read is the state of the output latch, and not the level of the pin at the pad. In all cases, whenever the pin level is recorded in bit 15 of PIN_LEVEL, an interrupt is generated (if enabled).

Note that to switch from discrete output to discrete input, the host sequence bits must be configured for the proper mode of operation and initialization executed. To switch from discrete input to discrete output, no initialization is required; only the proper HSR must be initiated to force the proper output pin level.

7 Performance and Use of Function

7.1 Performance

Like all TPU functions, DIO function performance in an application is to some extent dependent upon the service time (latency) of other active TPU channels. This is due to the operational nature of the scheduler. The more TPU channels are active, the more performance decreases. Worst-case latency in any TPU application can be closely estimated. To analyze the performance of an application that appears to approach the limits of the TPU, use the guidelines given in the TPU reference manual and the information in the DIO state timing table below.
7.2 Changing Mode

The host sequence bits are used to select DIO function operating mode. Change host sequence bit values only when the function is stopped or disabled (channel priority bits = %00). Disabling the channel before changing mode avoids conditions that cause indeterminate operation.

8 Function Examples

The following examples give an indication of the capabilities of the DIO function. Each example includes a description of the example, a diagram of the initial parameter RAM content, a diagram of the output waveform, and a program listing.

8.1 Example A

8.1.1 Description

This example sets up TPU channels 0, 5, 10, and 15 as outputs and causes them to go high or low by issuing the appropriate host service requests. Four square waves are generated at frequencies of f, 2f, 4f, and 8f.

8.1.2 Initialization

Configure the CHANNEL_CONTROL register for channels 0, 5, 10, and 15 as follows:

<table>
<thead>
<tr>
<th>State Number and Name</th>
<th>Max CPU Clock Cycles</th>
<th>RAM Accesses by TPU</th>
</tr>
</thead>
<tbody>
<tr>
<td>S1 Init</td>
<td>18</td>
<td>4</td>
</tr>
<tr>
<td>S2 Match_PS</td>
<td>10</td>
<td>3</td>
</tr>
<tr>
<td>S3 Trans_PS_Low</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>S4 Trans_PS_High</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>S5 Low_Pin_Request</td>
<td>8</td>
<td>2</td>
</tr>
<tr>
<td>S6 High_Pin_Request</td>
<td>8</td>
<td>2</td>
</tr>
</tbody>
</table>

Table 2 DIO State Timing

Table 3 DIO CHANNEL_CONTROL Parameter

<table>
<thead>
<tr>
<th>Table 3 DIO CHANNEL_CONTROL Parameter</th>
</tr>
</thead>
<tbody>
<tr>
<td>$YFFF00 15 8 7 0</td>
</tr>
<tr>
<td>Channel 0</td>
</tr>
</tbody>
</table>

For More Information On This Product, Go to: www.freescale.com
8.1.3 Output Waveforms

8.1.4 Program Listing

1 * The following program uses TPU channels 0, 5, 10 and 15 in the Discrete
2 * Input/Output mode. The purpose of this example is to
3 * set up the channels as outputs and then cause them to go high or low
4 * by issuing the appropriate Host Service Requests. Four square waves will
5 * be generated at frequencies of f, 2f, 4f and 8f.
6
7 * This section of the program assigns names to the registers address.
8
9 00000000 10 TPUMCR equ $fffe00 ;TPU Module Configuration Reg
00000000 11 TTCR equ $fffe02
00000000 12 DSCR equ $fffe04
00000000 13 DSSR equ $fffe06
00000000 14 TICR equ $fffe08 ;TPU Interrupt Config. Reg
00000000 15 CIER equ $fffe0a ;TPU Ch. Interrupt Enable Reg
00000000 16 CFSR0 equ $fffe0c ;TPU Ch. Function Select Reg 0
00000000 17 CFSR1 equ $fffe0e ;TPU Ch. Function Select Reg 1
00000000 18 CFSR2 equ $fffe10 ;TPU Ch. Function Select Reg 2
00000000 19 CFSR3 equ $fffe12 ;TPU Ch. Function Select Reg 3
00000000 20 HSQR0 equ $fffe14 ;TPU Host Sequence Register 0
00000000 21 HSQR1 equ $fffe16 ;TPU Host Sequence Register 1
00000000 22 HSRR0 equ $fffe18 ;TPU Host Service Req. Reg 0
00000000 23 HSRR1 equ $fffe1a ;TPU Host Service Req. Reg 1
00000000 24 CPR0 equ $fffe1c ;TPU Channel Priority Reg 0
00000000 25 CPR1 equ $fffe1e ;TPU Channel Priority Reg 1
00000000 26 CISR equ $fffe20 ;Channel Interrupt Status Reg
00000000 27 CH0_CNTL equ $ffff00 ;Channel 0 Control Reg
00000000 28 CH0_PINL equ $ffff02 ;Channel 0 Pin Level Reg
00000000 29 CH0_MATCH equ $ffff04 ;Channel 0 Match Rate Reg
00000000 30 CH5_CNTL equ $ffff50 ;Channel 5 Control Register
00000000 31 CH5_PINL equ $ffff52 ;Channel 5 Pin Level Register
00000000 32 CH5_MATCH equ $ffff54 ;Channel 5 Match Rate Register
00000000 33 CH10_CNTL equ $ffffa0 ;Channel 10 Control Register
00000000 34 CH10_PINL equ $ffffa2 ;Ch. 10 Pin Level Register
00000000 35 CH10_MATCH equ $ffffa4 ;Ch. 10 Match Rate Register
00000000 36 CH15_CNTL equ $ffffff0 ;Ch. 15 Control Register
00000000 37 CH15_PINL equ $ffffff2 ;Ch. 15 Pin Level Register
00000000 38 CH15_MATCH equ $ffffff4 ;Channel 15 Match Rate Register
00000000 39
00005000 40 org $5000 ;program origin
41 * This portion of the program initializes the TPU reg
42
43 00005000 33FC8000 42 init move.w #$8000,(CFSR0).l ;init ch. 15 to DIO function
00005008 33FC8000 43 move.w #$0800,(CFSR1).l ;init ch. 10 to DIO function
00005010 33FC0080  44 move.w #$0080,(CFSR2).l ; init ch. 5 to DIO function
00005018 33FC0008  45 move.w #$0008,(CFSR3).l ; init ch. 0 to DIO function
00005020 33FC0000  46 move.w #$0000,(HSQR0).l ; ch. 15,10 pin level
00005028 33FC0000  47 move.w #$0000,(HSQR1).l ; ch. 5,0 pin level
00005030 33FC0030  48 move.w #$c030,(CPR0).l ; ch. 15,10 - hi priority
00005038 33FC0C03  49 move.w #$0c03,(CPR1).l ; ch. 5,0 - hi priority
00005040 33FC0003  50 move.w #$0003,(CH15_CNTL).l ; ch. 15 - use TCR1
00005048 33FC0003  51 move.w #$0003,(CH10_CNTL).l ; ch. 10 use TCR1
00005050 33FC0003  52 move.w #$0003,(CH5_CNTL).l ; ch. 5 use TCR1
00005058 33FC0003  53 move.w #$0003,(CH0_CNTL).l ; ch. 0 use TCR1
00005060 33FC0003  54 move.w #$0003,(CH15_CNTL).l ; ch. 15 - use TCR1
00005068 33FC0003  55 * This portion of the program only initializes the DIO channels as outputs.
00005070 33FC0003  56 * No action will be taken on an external pin by executing the next two
00005078 33FC0003  57 * instructions.
00005080 4EB90000  58 move.w #$8020,(HSRR0).l ; HSR - ch. 15-lo, c
00005086 33FC0802  59 move.w #$0802,(HSRR1).l ; HSR ch 5-lo, ch 0-lo
00005090 4EB90000  60 jsr wait
00005096 33FC0003  61 move.w #$8010,(HSRR0).l ; HSR - ch. 15 - lo, c
0000509E 33FC0003  62 * This portion of the program generates four square waves using software
000050A2 33FC0003  63 * timing loops.
000050A8 4EB90000  64 move.w #$8020,(HSRR0).l ; HSR - ch. 15-lo, c
000050B0 33FC0003  65 move.w #$0802,(HSRR1).l ; HSR ch 5-lo, ch 0-lo
000050B4 4EB90000  66 jsr wait
000050B8 33FC0003  67 move.w #$8010,(HSRR0).l ; HSR - ch. 15 - lo, c
000050C2 33FC0003  68 jsr wait
000050CC 4EB90000  69 move.w #$8020,(HSRR0).l ; HSR - ch. 15-lo, c
000050D0 33FC0003  70 jsr wait
000050D4 4EB90000  71 move.w #$0802,(HSRR1).l ; HSR ch 5-lo, ch 0-lo
000050D8 33FC0003  72 jsr wait
000050E0 4EB90000  73 move.w #$8010,(HSRR0).l ; HSR - ch. 15 - lo, c
000050E4 33FC0003  74 jsr wait
000050E8 4EB90000  75 move.w #$8020,(HSRR0).l ; HSR - ch. 15-lo, c
000050F2 33FC0003  76 jsr wait
000050F6 4EB90000  77 move.w #$0802,(HSRR1).l ; HSR ch 5-lo, ch 0-lo
000050F8 33FC0003  78 move.w #$8010,(HSRR0).l ; HSR - ch. 15 - lo, c
000050FA 33FC0003  79 move.w #$8010,(HSRR0).l ; HSR - ch. 15 - lo, c

55 * This portion of the program only initializes the DIO channels as outputs.
56 * No action will be taken on an external pin by executing the next two
57 * instructions.
61 62 * This portion of the program generates four square waves using software
63 * timing loops.
64 65 strt move.w #$8020,(HSRR0).l ; HSR - ch. 15-lo, c
65 66 jsr wait
67 68 jsr wait
69 70 jsr wait
71 72 jsr wait
73 74 jsr wait
74 75 jsr wait
76 77 jsr wait
78 79 jsr wait
00FFFE18 00050EDC 4EB90000 518C 78 jsr wait
00FFFE1A 00050F22 33FC0401 00005198 4EB90000 518C 79 move.w #$0401,(HSRR1).l ; HSR ch 5-hi, ch 0-hi
00FFFE1E 00050F9A 33FC8010 000051A6 4EB90000 518C 80 move.w #$8010,(HSRR0).l ; HSR - ch. 15 - lo, c
00FFFE18 00050F22 33FC0401 000051C2 4EB90000 518C 80 jsr wait
00FFFE1A 00050F9A 33FC8010 000051DE 4EB90000 518C 81 move.w #$0802,(HSRR1).l ; HSR ch. 5-lo, ch 0-lo
00FFFE18 00050F22 33FC0401 000051E4 4EB90000 518C 82 move.w #$4020,(HSRR0).l ; HSR ch. 15-hi, c
00FFFE1A 00050F22 33FC0401 000051F0 4EB90000 518C 82 jsr wait
00FFFE18 00050F22 33FC0401 00005198 4EB90000 518C 83 move.w #$0401,(HSRR1).l ; HSR ch. 5-hi, ch 0-lo
00FFFE1A 00050F22 33FC0401 000051A6 4EB90000 518C 84 move.w #$8010,(HSRR0).l ; HSR - ch. 15 - hi, c
00FFFE18 00050F22 33FC0401 000051C2 4EB90000 518C 85 jsr wait
00FFFE1A 00050F22 33FC0401 000051DE 4EB90000 518C 86 move.w #$0802,(HSRR1).l ; HSR ch. 5-lo, ch 0-hi
00FFFE18 00050F22 33FC0401 000051E4 4EB90000 518C 87 move.w #$4020,(HSRR0).l ; HSR ch. 15-hi, c
00FFFE1A 00050F22 33FC0401 000051F0 4EB90000 518C 88 jsr wait
00FFFE18 00050F22 33FC0401 00005198 4EB90000 518C 89 move.w #$0401,(HSRR1).l ; HSR ch. 5-hi, ch 0-lo
00FFFE1A 00050F22 33FC0401 000051A6 4EB90000 518C 90 move.w #$8010,(HSRR0).l ; HSR - ch. 15 - hi, c
00FFFE18 00050F22 33FC0401 000051C2 4EB90000 518C 90 jsr wait
00FFFE1A 00050F22 33FC0401 000051DE 4EB90000 518C 91 move.w #$0401,(HSRR1).l ; HSR ch. 5-hi, ch 0-hi
00FFFE1E 00050F9A 33FC8010 000051E4 4EB90000 518C 92 move.w #$8010,(HSRR0).l ; HSR - ch. 15 - lo, c
00FFFE18 00050F22 33FC0401 000051C2 4EB90000 518C 93 jsr wait
00FFFE1A 00050F22 33FC0401 000051DE 4EB90000 518C 94 move.w #$0401,(HSRR1).l ; HSR ch. 5-hi, ch 0-lo
00FFFE1E 00050F9A 33FC8010 000051E4 4EB90000 518C 95 move.w #$8010,(HSRR0).l ; HSR - ch. 15 - hi, c
00FFFE18 00050F22 33FC0401 000051C2 4EB90000 518C 96 jsr wait
00FFFE1A 00050F22 33FC0401 000051DE 4EB90000 518C 97 jmp strt
00FFFE18 00050F22 203C0000 000051F0 4EB90000 518C 98 wait move.l #$fff,d0 ; wait loop
00FFFE1A 00050F22 203C0000 000051E4 4EB90000 518C 99 loop1 sub.l #$1,d0 ; wait loop
00FFFE1E 00050F22 33FC8010 000051C2 4EB90000 518C 100 bne loop1 ; wait loop
00FFFE18 00050F22 33FC8010 000051DE 4EB90000 518C 101 rts
8.2 Example B

8.2.1 Description
This program uses TPU channels 0, 2, 5, 6, 10, 11, 14, and 15 with the discrete output function. The program sets up channels 0, 5, 10, and 15 as outputs and then causes them to go high or low by issuing the appropriate host service requests. Four square waves are generated at frequencies of f, 2f, 4f, and 8f. In addition, TPU channels 2, 6, 11, and 14 are configured as inputs and update their PIN_LEVEL registers on each transition. Channel 0 drives channel 2, channel 5 drives channel 6, channel 10 drives channel 11, and channel 16 drives channel 14. The PIN_LEVEL registers of the input channels will all contain $5555 or $AAAA after 16 transitions have been detected on the slowest channel.

8.2.2 Hardware Configuration

8.2.3 Initialization
Configure the CHANNEL_CONTROL registers as follows:

<table>
<thead>
<tr>
<th>$YFFFO0</th>
<th>$YFF50</th>
<th>$YFFA0</th>
<th>$YFFFO0</th>
<th>$YFF20</th>
<th>$YFFF60</th>
<th>$YFFB0</th>
<th>$YFFE0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 1 1</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1</td>
</tr>
<tr>
<td>Channel 0</td>
<td>Channel 5</td>
<td>Channel 10</td>
<td>Channel 15</td>
<td>Channel 2</td>
<td>Channel 6</td>
<td>Channel 11</td>
<td>Channel 14</td>
</tr>
</tbody>
</table>

---

Table 4 DIO CHANNEL_CONTROL Parameter
8.2.4 Output Waveforms

8.2.5 Program Listing

1 * The following program uses TPU channels 0, 2, 5, 6, 10, 11, 14 and 15 in the
2 Discrete Input/Output mode. This program sets up channels 0, 5, 10 and 15
3 as outputs and then cause them to go high or low
4 * by issuing the appropriate Host Service Requests. Four square waves will
5 * be generated at frequencies of f, 2f, 4f and 8f. TPU channels 2, 6,
6 * 11 and 14 will be configured as inputs and will update their PIN_LEVEL
7 * Registers on each transition. Channel 0 will drive channel 2, channel 5
8 * will drive channel 6, channel 10 will drive channel 11 and channel 15 will
9 * drive channel 14. The PIN_LEVEL Registers of the input channels should all
10 * contain $5555 or $AAAA after 16 transitions have been detected on the
11 * slowest channel.

14 * This section of the program assigns register names to the reg. address.
15
00000000 16 TPUMCR equ $ffe000 ;TPU Module Config. Reg
00000000 17 TTCR equ $ffe0e2 ;TPU Interrupt Config. Reg
00000000 18 DSCR equ $ffe0e4
00000000 19 DSR equ $ffe6e6
00000000 20 TICR equ $ffe080 ;TPU Channel Interrupt EnableReg
00000000 21 CFSR0 equ $ffe0ac ;TPU Ch. Function Select Reg 0
00000000 22 CFSR1 equ $ffe0e0e ;TPU Ch. Function Select Reg 1
00000000 23 CFSR2 equ $ffe100 ;TPU Ch. Function Select Reg 2
00000000 24 CFSR3 equ $ffe120 ;TPU Ch. Function Select Reg 3
00000000 25 HSQR0 equ $ffe140 ;TPU Host Sequence Reg 0
00000000 26 HSQR1 equ $ffe160 ;TPU Host Sequence Reg 1
00000000 27 HSQR2 equ $ffe180 ;TPU Host Svc. Request Reg 0
00000000 28 HSQR3 equ $ffe1a0 ;TPU Host Svc Request Reg 1
00000000 29 CPR0 equ $ffe1c0 ;TPU Channel Priority Reg 0
00000000 30 CPR1 equ $ffe1e0 ;TPU Channel Priority Reg 1
00000000 31 CISR equ $ffe20 ;Channel Int. Status Reg
00000000 32 CH0_CNTL equ $ffe000 ;Channel 0 Control Register
00000000 33 CH0_PINL equ $ffe002 ;Channel 0 Pin Level Reg
00000000 34 CH0_MATCH equ $ffe004 ;Channel 0 Match Rate Reg
00000000 35 CH2_CNTL equ $ffe202 ;Channel 2 Control Reg
00000000 36 CH2_PINL equ $ffe222 ;Channel 2 Pin Level Reg
00000000 37 CH2_MATCH equ $ffe240 ;Channel 2 Match Rate Reg
00000000 38 CH5_CNTL equ $ffe550 ;Channel 5 Control Reg
00000000 39 CH5_PINL equ $ffe552 ;Channel 5 Pin Level Reg
00000000 40 CH5_MATCH equ $ffe554 ;Channel 5 Match Rate Reg
00000000 41 CH6_CNTL equ $ffe600 ;Channel 6 Control Reg
00000000 42 CH6_PINL equ $ffe602 ;Channel 6 Pin Level Reg
00000000 43 CH6_MATCH equ $ffe604 ;Channel 6 Match Rate Reg
00000000 44 CH10_CNTL equ $ffe6a0 ;Channel 10 Control Reg
00000000 45 CH10_PINL equ $ffe6a2 ;Channel 10 Pin Level Reg
00000000 46 CH10_MATCH equ $ffe6a8 ;Channel 10 Match Rate Reg
00000000  47 CH10_MATCH equ $ffffa4 ;Channel 10 Match Rate Reg
00000000  48 CH11_CNTL equ $ffffb0 ;Channel 11 Control Reg
00000000  49 CH11_PINL equ $ffffb2 ;Channel 11 Pin Level Reg
00000000  50 CH11_MATCH equ $ffffb4 ;Channel 11 Match Rate Reg
00000000  51 CH14_CNTL equ $ffffe0 ;Channel 14 Control Reg
00000000  52 CH14_PINL equ $ffffe2 ;Channel 14 Pin Level Reg
00000000  53 CH14_MATCH equ $ffffe4 ;Channel 14 Match Rate Reg
00000000  54 CH15_CNTL equ $fffff0 ;Channel 15 Control Reg
00000000  55 CH15_PINL equ $fffff2 ;Channel 15 Pin Level Reg
00000000  56 CH15_MATCH equ $fffff4 ;Channel 15 Match Rate Reg

00005000  57 org $5000 ;program origin

00005000  58 * This portion of the program initializes the TPU registers
00005008  59 init move.w #$8800,(CFSR0).l ;init ch 15,14 to DIO
00005010  60 move.w #$8800,(CFSR1).l ;init ch 10,11 to DIO
00005018  61 move.w #$0808,(CFSR2).l ;init ch 5,6 to DIO
00005020  62 move.w #$0000,(HSQR0).l ;ch. 15,14,11,10 pin level
00005028  63 move.w #$0000,(HSQR1).l ;reg. update on transition
00005030  64 move.w #$f0f0,(CPR0).l ;ch. 15,14,10,11 have
00005038  65 move.w #$3c33,(CPR1).l ;ch. 5,6 and 2,0 have
00005040  66 move.w #$0003,(CH15_CNTL).l ;ch. 15 use TCR1
00005048  67 move.w #$0003,(CH14_CNTL).l ;ch. 14 use TCR1
00005050  68 move.w #$0003,(CH11_CNTL).l ;ch. 11 use TCR1
00005058  69 move.w #$0003,(CH10_CNTL).l ;ch. 10 use TCR1
00005060  70 move.w #$000f,(CH6_CNTL).l ;ch. 6 use TCR1
00005068  71 move.w #$000f,(CH5_CNTL).l ;ch. 5 use TCR1
00005070  72 move.w #$000f,(CH2_CNTL).l ;ch. 2 use TCR1
00005078  73 move.w #$000f,(CH0_CNTL).l ;ch. 0 use TCR1

00005080  74 strt move.w #$f0f0,(HSRR0).l ;HSR-ch 15,14,11,10
00005088  75 move.w #$3c33,(HSRR1).l ;HSR-ch.5,6,2,0

00005090  76 * This portion of the program initializes the DIO channels as inputs and
00005098  77 * outputs, as required. No action
00005098  78 * will be taken on an external pin by executing the next two instructions.
00005098  79
00005098  80
00005098  81 move.w #$000f,(CH14_CNTL).l ;ch. 14 use TCR1
00005098  82 move.w #$000f,(CH6_CNTL).l ;ch. 6 use TCR1
00005098  83

00005098  84 * This portion of the program generates four square waves using software
00005098  85 * timing loops on channels 0, 5, 10 and 15. The transitions from these
00005098  86 * channels are recorded on channels 2, 6, 11 and 14, respectively.
00005098  87
00005098  88 strt move.w #$8020,(HSRR0).l ;HSR-ch 15,14,11,10, c
00005098  89 move.w #$0802,(HSRR1).l ;HSR-ch 5-10, ch 0-10
000050A0  90 jsr wait
000050A6 33FC0001 90 move.w #$0001,(HSRR1).l ;HSR - ch. 0 - hi
000050AE 4EB90000 91 jsr wait
000050B4 33FC0402 92 move.w #$0402,(HSRR1).l ;HSR-ch 5-hi, ch 0 lo
000050BC 4EB90000 93 jsr wait
000050C2 33FC0001 94 move.w #$0001,(HSRR1).l ;HSR-ch 5-hi, ch 0-hi
000050CA 4EB90000 95 jsr wait
000050D0 33FC0802 96 move.w #$0802,(HSRR1).l ;HSR-ch. 5-lo, ch 0-lo
000050D8 33FC0010 97 move.w #$0010,(HSRR0).l ;HSR - ch. 15 - lo, c
000050E0 4EB90000 97 jsr wait
000050E6 33FC0001 98 move.w #$0001,(HSRR1).l ;HSR-ch 5-lo, ch 0-hi
000050EE 4EB90000 100 jsr wait
000050F4 33FC0402 101 move.w #$0402,(HSRR1).l ;HSR-ch 5-hi, ch 0-lo
000050FC 4EB90000 103 jsr wait
00005102 33FC0001 104 move.w #$0001,(HSRR1).l ;HSR-ch 5-hi, ch 0-hi
0000510A 4EB90000 105 jsr wait
00005110 33FC0802 107 move.w #$0802,(HSRR1).l ;HSR-ch. 5-lo, ch0-lo
00005118 33FC0402 108 move.w #$0402,(HSRR1).l ;HSR-ch 5-hi, ch 0-lo
00005120 4EB90000 108 jsr wait
00005126 33FC0001 109 move.w #$0001,(HSRR1).l ;HSR-ch. 5-lo, ch 0-hi
0000512E 4EB90000 110 jsr wait
00005134 33FC0402 111 move.w #$0402,(HSRR1).l ;HSR-ch 5-hi, ch0-lo
0000513C 4EB90000 114 jsr wait
00005142 33FC0001 113 move.w #$0001,(HSRR1).l ;HSR-ch 5-hi, ch 0-hi
0000514A 4EB90000 114 jsr wait
00005150 33FC0802 115 move.w #$0802,(HSRR1).l ;HSR-ch 5-lo, ch 0-lo
00005158 33FC0010 116 move.w #$0010,(HSRR0).l ;HSR-ch 15-hi, c
00005160 4EB90000 116 jsr wait
00005166 33FC0001 117 move.w #$0001,(HSRR1).l ;HSR-ch 5-lo, ch 0 hi
0000516E 4EB90000 118 jsr wait
00005174 33FC0402 119 move.w #$0402,(HSRR1).l ;HSR-ch 5-hi, ch 0-lo
0000517C 4EB90000 120 jsr wait
00005182 33FC0001 121 move.w #$0001,(HSRR1).l ;HSR-ch 5-hi, ch 0-lo
8.3 Example C

8.3.1 Description

This program uses TPU channels 0, 2, 5, 6, 10, 11, 14, and 15 with the discrete output function. The program sets up channels 0, 5, 10, and 15 as outputs and then causes them to go high or low by issuing the appropriate host service requests. Four square waves are generated at frequencies of f, 2f, 4f, and 8f. In addition, TPU channels 2, 6, 11, and 14 are configured as inputs and programmed to update at match rate. Channel 0 drives channel 2, channel 5 drives channel 6, channel 10 drives channel 11, and channel 16 drives channel 14.

8.3.2 Hardware Configuration

Configure the CHANNEL_CONTROL and MATCH_RATE registers as follows:

```
0FFFE1A                    122 jsr wait
0000518A 4EB90000 51AC     123 move.w (CH2_PINL).l,d4 ;pin level 2 to D4
00005190 383900FF FF22     124 move.w (CH6_PINL).l,d5 ;pin level 5 to D5
00005196 3A3900FF FF62     125 move.w (CH11_PINL).l,d6 ;pin level 11 to D6
0000519C 3C3900FF FFB2     126 move.w (CH14_PINL).l,d7 ;pin level 14 to D7
000051A2 3E3900FF FFE2     127 jmp strt
000051A8 4EF85090          128 wait move.l #$fff,d0 ;wait loop
000051AC 203C0000 0FFF     129 lop1 sub.l #$1,d0 ;wait loop
000051B2 90BC0000 0001     130 bne lop1 ;wait loop
000051B8 6600FFF8          131 rts
000051BC 4E75              132
```
8.3.4 Output Waveforms

Table 5 DIO CHANNEL_CONTROL Parameter

<table>
<thead>
<tr>
<th>Address</th>
<th>Channel 0</th>
<th>Channel 5</th>
<th>Channel 10</th>
<th>Channel 15</th>
</tr>
</thead>
<tbody>
<tr>
<td>$YFF00</td>
<td>0 0 0 0 0 0 0 0 0 0 1 1</td>
<td>0 0 0 0 0 0 0 0 0 0 1 1</td>
<td>0 0 0 0 0 0 0 0 0 0 1 1</td>
<td>0 0 0 0 0 0 0 0 0 0 1 1</td>
</tr>
<tr>
<td>$YFF50</td>
<td>0 0 0 0 0 0 0 0 0 0 1 1</td>
<td>0 0 0 0 0 0 0 0 0 0 1 1</td>
<td>0 0 0 0 0 0 0 0 0 0 1 1</td>
<td>0 0 0 0 0 0 0 0 0 0 1 1</td>
</tr>
<tr>
<td>$YFFA0</td>
<td>0 0 0 0 0 0 0 0 0 0 1 1</td>
<td>0 0 0 0 0 0 0 0 0 0 1 1</td>
<td>0 0 0 0 0 0 0 0 0 0 1 1</td>
<td>0 0 0 0 0 0 0 0 0 0 1 1</td>
</tr>
<tr>
<td>$YFF00</td>
<td>0 0 0 0 0 0 0 0 0 0 1 1</td>
<td>0 0 0 0 0 0 0 0 0 0 1 1</td>
<td>0 0 0 0 0 0 0 0 0 0 1 1</td>
<td>0 0 0 0 0 0 0 0 0 0 1 1</td>
</tr>
<tr>
<td>$YFFF20</td>
<td>0 0 0 0 0 0 0 0 0 0 1 1</td>
<td>0 0 0 0 0 0 0 0 0 0 1 1</td>
<td>0 0 0 0 0 0 0 0 0 0 1 1</td>
<td>0 0 0 0 0 0 0 0 0 0 1 1</td>
</tr>
<tr>
<td>$YFF60</td>
<td>0 0 0 0 0 0 0 0 0 0 1 1</td>
<td>0 0 0 0 0 0 0 0 0 0 1 1</td>
<td>0 0 0 0 0 0 0 0 0 0 1 1</td>
<td>0 0 0 0 0 0 0 0 0 0 1 1</td>
</tr>
<tr>
<td>$YFFB0</td>
<td>0 0 0 0 0 0 0 0 0 0 1 1</td>
<td>0 0 0 0 0 0 0 0 0 0 1 1</td>
<td>0 0 0 0 0 0 0 0 0 0 1 1</td>
<td>0 0 0 0 0 0 0 0 0 0 1 1</td>
</tr>
<tr>
<td>$YFFE0</td>
<td>0 0 0 0 0 0 0 0 0 0 1 1</td>
<td>0 0 0 0 0 0 0 0 0 0 1 1</td>
<td>0 0 0 0 0 0 0 0 0 0 1 1</td>
<td>0 0 0 0 0 0 0 0 0 0 1 1</td>
</tr>
</tbody>
</table>

Table 6 DIO MATCH_RATE Parameter

<table>
<thead>
<tr>
<th>Address</th>
<th>Channel 2</th>
<th>Channel 6</th>
<th>Channel 11</th>
<th>Channel 14</th>
</tr>
</thead>
<tbody>
<tr>
<td>$YFF24</td>
<td>0 0 0 0 1 0 0 1 0 0 0 0 0 0 0 0 0 0 0</td>
<td>0 0 0 0 1 0 0 1 0 0 0 0 0 0 0 0 0 0 0</td>
<td>0 0 0 0 1 0 0 1 0 0 0 0 0 0 0 0 0 0 0</td>
<td>0 0 0 0 1 0 0 1 0 0 0 0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>$YFF64</td>
<td>0 0 0 0 1 0 0 1 0 0 0 0 0 0 0 0 0 0 0</td>
<td>0 0 0 0 1 0 0 1 0 0 0 0 0 0 0 0 0 0 0</td>
<td>0 0 0 0 1 0 0 1 0 0 0 0 0 0 0 0 0 0 0</td>
<td>0 0 0 0 1 0 0 1 0 0 0 0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>$YFFB4</td>
<td>0 0 0 0 1 0 0 1 0 0 0 0 0 0 0 0 0 0 0</td>
<td>0 0 0 0 1 0 0 1 0 0 0 0 0 0 0 0 0 0 0</td>
<td>0 0 0 0 1 0 0 1 0 0 0 0 0 0 0 0 0 0 0</td>
<td>0 0 0 0 1 0 0 1 0 0 0 0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>$YFFE4</td>
<td>0 0 0 0 1 0 0 1 0 0 0 0 0 0 0 0 0 0 0</td>
<td>0 0 0 0 1 0 0 1 0 0 0 0 0 0 0 0 0 0 0</td>
<td>0 0 0 0 1 0 0 1 0 0 0 0 0 0 0 0 0 0 0</td>
<td>0 0 0 0 1 0 0 1 0 0 0 0 0 0 0 0 0 0 0</td>
</tr>
</tbody>
</table>

8.3.5 Program Listing

```
1
2
3 * The following program uses TPU channels 0, 2, 5, 6, 10, 11, 14 and 15
4 * in the Discrete Input/Output mode.
5 *
6 * Channels 0, 5, 10 and 15 are set up as outputs and then caused to go
7 * high or low by issuing the appropriate Host Service Requests. Four
8 * square waves will be generated at frequencies of f, 2f, 4f and 8f.
9 * Channels 2, 6, 11 and 14 are set up in the Discrete Input/Output mode and
10 * programmed to Update at Match Rate. Channel 0 is connected to channel 2,
11 * channel 5 is connected to channel 6, channel 10 is connected to channel 11
12 * and channel 15 is connected to channel 14.
13 *
14 * This section of the program assigns names to the register's address.
15 00000000
16 TPUMCR equ $ffe00 ;TPU Module Config.Reg
17 TTCR equ $ffe02 ;
```

Freescale Semiconductor, Inc.
TPU Programming Library
TPUPN18/D

For More Information On This Product, Go to: www.freescale.com
This portion of the program initializes the DIO channels as inputs and outputs as specified. No action will be taken on an external pin by executing the next two instructions.

This portion of the program generates four square waves using software timing loops.

For More Information On This Product,
Go to: www.freescale.com
9 Function State Descriptions

This section describes the states entered for each of the four DIO cases (request for initialization, update on match rate, update on transition, and set pin low or high). Refer to 10 Function Algorithm for detailed descriptions of each state.

A host service request can be issued at any time and from any state. To begin, assume that the channel function select register, host sequence register, channel priority register and the channel parameter RAM have all been programmed. At this point the channel will be ready to receive its first host service request via the host service request register.

9.1 CASE 1: Request for Initialization

HSR = 11, INITIALIZATION
HSQ = 10, UPDATE ON HSR = 11

State 1 is entered. The channel pin is not configured as an output or an input; it is simply left in its current condition. The contents of the PIN_LEVEL register are shifted to the right by one bit and the new pin state is recorded in bit 15 of the PIN_LEVEL register. Out of reset, the TPU channel pins are inputs. In general, a TPU channel could be either an input or an output. The level that is written into bit 15 of the PIN_LEVEL register will either be the level driven into the TPU channel if the channel is configured as an input or the level being driven by the TPU channel if the channel is configured as an output.
It is most important to recognize that for the case being discussed, i.e., the HSR bits = 11 (initialization) and the HSQ bits = 10 (mode 2), the S1 state is simply executed, exited, and no further action is taken. Realize that S2, S3 and S4 cannot be entered unless the algorithm is currently in S1. Therefore, the HSQ bits cannot be equal to %10 if a channel is to be configured as an input and sampled at the match rate or on each transition.

9.2 CASE 2: Update on MATCH_RATE

<table>
<thead>
<tr>
<th>HSR</th>
<th>HSQ</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>11</td>
<td>01</td>
<td>UPDATE ON MATCH RATE</td>
</tr>
</tbody>
</table>

State 1 is entered where the channel pin is configured as an input. The algorithm exits state 1 and goes to state 2. The algorithm will remain in state 2 until a new host service request is issued. While in state 2, the TPU channel pin, already configured as an input, is continually sampled at the periodic rate determined by the MATCH_RATE register. Each time the TPU channel is sampled the contents of the PIN_LEVEL register are shifted to the right by one bit and the new pin state is recorded in bit 15 of the PIN_LEVEL register. The algorithm remains in state 2.

9.3 CASE 3: Update on Transition

<table>
<thead>
<tr>
<th>HSR</th>
<th>HSQ</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>11</td>
<td>00</td>
<td>UPDATE ON TRANSITION</td>
</tr>
</tbody>
</table>

State 1 is entered where the channel pin is configured as in input. The algorithm exits state 1 and goes to either state 3 if the TPU channel pin is currently at a logic 0 or state 4 if the TPU channel pin is currently at a logic 1.

The DIO algorithm can be programmed to recognize positive edges only, negative edges only or both positive and negative edges. If negative edges only are selected, the algorithm goes to and remains in state 3. The PIN_LEVEL register is updated with a logic 0 each time a negative transition occurs. If positive edges only are selected, the algorithm goes to and remains in state 4. The PIN_LEVEL register is updated with a logic 1 each time a positive transition occurs. If both positive and negative edges are selected, either state 3 or state 4 is entered depending upon whether the next transition is negative or positive, respectively. After either state 3 or state 4 is entered, the algorithm alternates between the two states with each new transition.

9.4 CASE 4A: Set Pin Low

<table>
<thead>
<tr>
<th>HSR</th>
<th>HSQ</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>10</td>
<td>SET PIN LOW</td>
</tr>
</tbody>
</table>

State 5 is entered where the channel pin is configured as an output and the pin is driven to a logic 0. The contents of the PIN_LEVEL register are shifted to the right by one bit and a logic 0 is recorded in bit 15 of the PIN_LEVEL register. The algorithm remains in state 5 until a new host service request is issued.

9.5 CASE 4B: Set Pin High

<table>
<thead>
<tr>
<th>HSR</th>
<th>HSQ</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>01</td>
<td>10</td>
<td>SET PIN HIGH</td>
</tr>
</tbody>
</table>

State 6 is entered where the channel pin is configured as an output and the pin is driven to a logic 1. The contents of the PIN_LEVEL register are shifted to the right by one bit and a logic 1 is recorded in bit 15 of the PIN_LEVEL register. The algorithm remains in state 6 until a new host service request is issued.
10 Function Algorithm

The DIO time function consists of six states, described in the following paragraphs. The following description is provided as a guide only, to aid understanding of the function. The exact sequence of operations in microcode may be different from that shown, in order to optimize speed and code size. TPU microcode source listings for all functions in the TPU function library can be downloaded from the Motorola Freeware bulletin board. Refer to *Using the TPU Function Library and TPU Emulation Mode (TPUPN00/D)* for detailed instructions on downloading and compiling microcode.

10.1 State 1: *InitRecord_PS*

Condition: HSR1, HSR0, M/TSR, LSR, Pin, Flag0 = 11xxxx

Match Enable: Don't Care

Summary:

This state is entered as a result of HSR %11. In this state the channel is either configured to perform modes 1 or 2 discrete input or, if host sequence equals 10, the channel is not configured and the pin state is recorded in bit 15 of PIN_LEVEL. The previous pin states are shifted right by one, losing the least recent pin state in bit 0. An interrupt request is then generated. Flag0 is used internally to indicate one of two modes of operation when configured for discrete input. When clear, flag0 indicates transition mode operation; when set, it indicates match mode operation.

\[
\text{If host sequence bit 1} = 1 \{ \\
\quad \text{Bit N replaced by bit N+1 of parameter PLV} \\
\quad \text{Bit 15 of parameter PLV gets pin state} \\
\quad \text{Assert interrupt request} \\
\} \quad \text{Else} \{ \\
\quad \text{Configure the channel latches via CHANNEL_CONTROL} \\
\quad \text{Clear flag0} \\
\quad \text{ERT replaced by TCR1} \\
\quad \text{If host sequence bit 0} = 0 \{ \\
\quad \quad \text{Bit N replaced by bit N+1 of parameter PLV} \\
\quad \quad \text{Bit 15 of parameter PLV gets pin state} \\
\quad \quad \text{Assert interrupt request} \\
\quad \} \quad \text{Else} \{ \\
\quad \quad \text{Generate match at ERT + MATCH_RATE} \\
\quad \quad \text{Assert flag0} \\
\quad \quad \text{Bit N replaced by bit N+1 of parameter PLV} \\
\quad \quad \text{Bit 15 of parameter PLV gets pin state} \\
\quad \quad \text{Assert interrupt request} \\
\} \}
\]

10.1.1 State 2: *Match_PS*

Condition: HSR1, HSR0, M/TSR, LSR, Pin, Flag0 = 001xx1

Match Enable: Disable

Summary:

This state is entered due to a match when the channel is configured for discrete input match mode operation. In this state a new match time is scheduled by adding the last match time to MATCH_RATE. Bit 15 of PIN_LEVEL is updated with the pin level recorded at the time of service, and the previous pin states are shifted right by one, losing the least recent pin state contained in bit 0. An interrupt request is then generated.
Generate match at ERT + MATCH_RATE
Bit N replaced by bit N+1 of parameter PLV
Bit 15 of parameter PLV gets pin state
Negate MRL, TDL, LSR
Assert interrupt request

10.1.2 State 3: Trans_PS_Low

Condition: HSR1, HSR0, M/TSR, LSR, Pin, Flag0 = 001x00

Match Enable: Don’t Care

Summary:
This state is entered in transition mode only, after a transition is detected, when the pin is low at the time of service. Bit 15 of PIN_LEVEL is updated with the pin level, and the previous pin states are shifted right by one, losing the least recent pin state contained in bit 0. An interrupt request is then generated.

Bit N replaced by bit N+1 of parameter PLV
Bit 15 of parameter PLV gets pin state 0
Negate MRL, TDL, LSR
Assert interrupt request

10.1.3 State 4: Trans_PS_High

Condition: HSR1, HSR0, M/TSR, LSR, Pin, Flag0 = 001x10

Match Enable: Don’t Care

Summary:
This state is entered in transition mode only, after a transition is detected, when the pin is high at the time of service. Bit 15 of PIN_LEVEL is updated with the pin level, and the previous pin states are shifted right by one, losing the least recent pin state contained in bit 0. An interrupt request is then generated.

Bit N replaced by bit N+1 of parameter PLV
Bit 15 of parameter PLV gets pin state 1
Negate MRL, TDL, LSR
Assert interrupt request

10.1.4 State 5: Low_Pin_Request

Condition: HSR1, HSR0, M/TSR, LSR, Pin, Flag0 = 10xxxx

Match Enable: Don’t Care

Summary:
This state is entered due to HSR %10. In this state the pin becomes an output and is forced low, and the match and capture time bases are forced to TCR1. Bit 15 of PIN_LEVEL is updated with the pin level, which is forced, and the previous pin states are shifted right by one, losing the least recent pin state contained in bit 0. An interrupt request is then generated.

Set pin low
Bit N replaced by bit N+1 of parameter PLV
Bit 15 of parameter PLV gets pin state 0
Negate MRL, TDL, LSR
Assert interrupt request
10.1.5 State 6: High_Pin_Request
Condition: HSR1, HSR0, M/TSR, LSR, Pin, Flag0 = 01xxxx
Match Enable: Don't Care

Summary:
This state is entered as a result of HSR %01. In this state the pin becomes an output and is forced high, and the match and capture time bases are forced to TCR1. Bit 15 of PIN_LEVEL is updated with the pin level, which is forced, and the previous pin states are shifted right by one, losing the least recent pin state contained in bit 0. An interrupt request is then generated.

- Set pin high
- Bit N replaced by bit N+1 of parameter PLV
- Bit 15 of parameter PLV gets pin state 0
- Negate MRL, TDL, LSR
- Assert interrupt request

The following table shows the DIO transitions listing the service request sources and channel conditions from current state to next state. Figure 4 illustrates the flow of DIO states.

<table>
<thead>
<tr>
<th>Current State</th>
<th>HSR</th>
<th>M/TSR</th>
<th>LSR</th>
<th>Pin</th>
<th>Flag0</th>
<th>Flag1</th>
<th>Next State</th>
</tr>
</thead>
<tbody>
<tr>
<td>Any State</td>
<td>11</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>S6 High_Pin_Request_PS</td>
</tr>
<tr>
<td>Any State</td>
<td>01</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>S5 Low_Pin_Request</td>
</tr>
<tr>
<td>Any State</td>
<td>10</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>S1 Init/Record_PS</td>
<td>00</td>
<td>1</td>
<td>—</td>
<td>1</td>
<td>—</td>
<td>—</td>
<td>S2 Match_PS</td>
</tr>
<tr>
<td>S2 Match_PS</td>
<td>00</td>
<td>1</td>
<td>—</td>
<td>0</td>
<td>0</td>
<td>—</td>
<td>S3 Trans_PS_Low</td>
</tr>
<tr>
<td>S3 Trans_PS_Low</td>
<td>00</td>
<td>1</td>
<td>—</td>
<td>0</td>
<td>0</td>
<td>—</td>
<td>S4 Trans_PS_High</td>
</tr>
<tr>
<td>S4 Trans_PS_High</td>
<td>00</td>
<td>1</td>
<td>—</td>
<td>0</td>
<td>0</td>
<td>—</td>
<td>S3 Trans_PS_Low</td>
</tr>
<tr>
<td>Unimplemented</td>
<td>00</td>
<td>0</td>
<td>1</td>
<td>x</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
</tbody>
</table>

NOTES:
1. Conditions not specified are “don’t care.”
2. LSR = Link service request
   - HSR = Host service request
   - M/TSR = Either a match or transition (input capture) service request occurred (M/TSR = 1) or neither occurred (M/TSR = 0).
Figure 4 DIO State Flowchart

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