



Mod5234 DMA Timers

Application Note

Revision 1.0
September 11, 2007
Document Status: Second Release

Table of Contents

Introduction	3
DMA Timer Registers	3
DMA Timer Mode Register (TMR)	3
DMA Timer Extended Mode Register (TXMR)	4
DMA Timer Event Register (TER)	4
DMA Timer Reference Register (TRR)	4
DMA Timer Capture Register (TCR)	4
DMA Timer Counter Register (TCN)	5
Configuring a 1-Millisecond DMA Timer	5
Program Example	6

Introduction

The Mod5234 has four direct memory access timer modules, DTIM0-DTIM3. Each 32-bit DMA timer provides input capture and reference compare capabilities with optional signaling of events using interrupts or triggers.

This application note will describe the use of the timers, as well as provide an example program that uses one of the timers to create interrupts at a regular interval. The example is useful both for learning how to use interrupts as well as how the DMA timer functions. Additional information concerning the DMA timers are available in the MCF5235 Reference Manual, Chapter 24 (Revision 2.0).

DMA Timer Registers

The DMA timers involve the use of six types of registers for configuration and control (depending on how the timer is used, not all of them may be required):

- DMA Timer Mode Register (TMR - 16-bit)
- DMA Timer Extended Mode Register (TXMR - 8-bit)
- DMA Timer Event Register (TER - 8-bit)
- DMA Timer Reference Register (TRR - 32-bit)
- DMA Timer Capture Register (TCR - 32-bit)
- DMA Timer Counter Register (TCN - 32-bit)

Each timer module has its own set of registers. They may be configured to assert an output signal, generate an interrupt, or initiate a DMA transfer on a particular event. All registers support read/write capability, with the exception of the Timer Capture Register, which supports read-only access. The following subsections describe these registers.

DMA Timer Mode Register (TMR)

This is where most of the configuration of the DMA timer takes place. A DMA timer can be configured to operate from the system clock or from an external clocking source using an available DMA timer input signal (only DTIM0 and DTIM1 input signals are made available). If the system clock is selected, it can be divided by 16 or 1. The selected clock source is routed to an 8-bit programmable prescaler that clocks the actual DMA timer counter register (TCN). This register also controls timer reset and enable. Refer to Section 24.2.6 of the MCF5235 Reference Manual for more information.

DMA Timer Extended Mode Register (TXMR)

This register enables and disables DMA request output on a counter reference match or capture edge event. It also selects the increment mode for the timer, whether it increments by 1 or by 65,537. When incrementing by 65,537 is used, only the upper bits of the 32-bit timer is exercised while it mirrors the lower 16 bits. Refer to Section 24.2.7 of the MCF5235 Reference Manual for more information.

DMA Timer Event Register (TER)

This register reports capture or reference events by setting the reference or capture bit. This reporting is done regardless of the corresponding DMA request or interrupt enable values in the TXMR and TMR registers, respectively.

Writing a 1 to either the reference or capture bit of this register clears it (writing a 0 does not affect bit value); both bits can be cleared at the same time. If configured to generate an interrupt request, the reference and capture bits must be cleared early in the interrupt service routine so that the timer module can negate the interrupt request signal to the interrupt controller. If configured to generate a DMA request, the processing of the DMA data transfer automatically clears both the reference and capture flags via the internal DMA ACK signal. Refer to Section 24.2.8 of the MCF5235 Reference Manual for more information.

DMA Timer Reference Register (TRR)

This register contains the reference value compared with the respective free-running timer counter (TCN) as part of the output-compare function. The reference value is not matched until TCN equals TRR, and the prescaler indicates that TCN should be incremented again. Refer to Section 24.2.9 of the MCF5235 Reference Manual for more information.

DMA Timer Capture Register (TCR)

This register latches the corresponding TCN value during a capture operation when an edge occurs on DTIN0 or DTIN1 (the input signals available for DTIM0 and DTIM1, respectively), as programmed in the DMA Timer Mode Register (TMR). The system clock is assumed to be the clock source. DTIN0/1 cannot simultaneously function as a clocking source and as an input capture pin. Indeterminate operation will result if DTIN0/1 is set as the clock source when the input capture mode is used. Refer to Section 24.2.10 of the MCF5235 Reference Manual for more information.

DMA Timer Counter Register (TCN)

The current value of this register can be read at any time without affecting counting. Writing to the counter register clears it. The timer counter increments on the clock source rising edge. Refer to Section 24.2.11 of the MCF5235 Reference Manual for more information.

Configuring a 1-Millisecond DMA Timer

This section describes how to set a timer to interrupt about every millisecond by using the system clock as an input clock source and utilizing reference compare values to generate interrupts. Before writing to registers to configure and initialize the timer, one needs to determine the desired resolution through calculation. The values needed in configuring a DMA timer can be determined by using the formula described in Section 24.3.2 of the MCF5235 Reference Manual:

T = Time-out period in seconds
CLOCKFREQ = 73,728,000 Hz is used for CLKOUT, which is half of the internal system clock (147,456,000 Hz)
DIVIDER = 1 or 16 for system clock (set via Mode Register CLK [2:1])
PRESCALER = 1 (0x00) to 256 (0xFF) (set via Mode Register PS [15:8])
REF_VAL = Reference compare value (set via 32-bit Reference Register)

$$T = (1 / \text{CLOCKFREQ}) * \text{DIVIDER} * (\text{PRESCALAR} + 1) * (\text{REF_VAL} + 1)$$

The known values for the above variables at this point are: T (0.001 seconds) and the CLOCKFREQ (73,728,000 Hz). For the DIVIDER, Table 24-2 is referenced on the available values for the CLK segment of the DMA Timer Mode Register [2:1] - the value of '1' (01) will be used to divide the system clock. Next is the PRESCALER value - any value between 1 (0x00) to 256(0xFF) can be used to divide the clock input. The value of '1' (0x00) is arbitrarily used in this case for simplicity.

Having determined a sufficient set of values, all that is left is to figure out what reference compare value will be used to set the 32-bit DMA Timer Reference Register:

$$0.001 = (1 / 73,728,000) * 1 * (0 + 1) * (\text{REF_VAL} + 1)$$

The value for REF_VAL comes out to be 73,727 (0x11FFF). There is now sufficient information to configure a DMA Timer. The following section contains a program example which shows how the registers are configured and initialized in conjunction to using interrupts.

Program Example

```
/* *****
 * This example program exercises one of the DMA timer modules to be *
 * used to generate a 1-millisecond resolution interrupt for the *
 * MCF5234 CPU. A counter is displayed every second via the debug *
 * serial port in MTTY on how many interrupts are being generated, *
 * which is almost 1000 times per second. *
 * *****/

#include "predef.h"
#include <stdio.h>
#include <ctype.h>
#include <startnet.h>
#include <autoupdate.h>
#include <dhcpcclient.h>
#include <sim5234.h> // For access to DMA timer registers
#include <cfinter.h> // For utilizing interrupts

//
// Function prototypes - Instruct the C++ compiler not to mangle the
// function names
//
extern "C"
{
    void UserMain( void *pd );

    //
    // This function sets up the 5234 interrupt controller
    //
    void SetIntc( int intc, long func, int vector, int level,
                 int prio );
}

//
// Name for development tools to identify this application
//
const char *AppName = "MOD5234 DMA Timer Example";

DWORD uint32_counter = 0; // Initialize interrupt counter

////////////////////////////////////
// INTERRUPT() - Interrupt service routine. Take note that when the
// DTIM0 interrupt is initialized via SetIntc() in UserMain(), it is
// configured to be a level 1 interrupt, while the INTERRUPT() macro
// below specifies in the input parameter that all interrupts at level
// 6 and below are blocked. Once INTERRUPT() is entered after the
// level 1 DTIM0 interrupt is triggered, level 6 and below interrupts
// are blocked during the execution of this service routine until
// INTERRUPT() exits.
//
INTERRUPT( func_isr, 0x2600 )
{
    sim.timer[0].ter |= 0x02; // Clear the DTIM0 reference event flag
    uint32_counter++; // Increment interrupt counter
}
```

```

}

////////////////////////////////////////////////////////////////////////////////////////////////////////////////
// UserMain() - Main task
//
void UserMain( void *pd )
{
    InitializeStack();
    if ( EthernetIP == 0 ) GetDHCPAddress();
    OSChangePrio( MAIN_PRIO );
    EnableAutoUpdate();

    fprintf( "Application started\r\n" );

   ////////////////////////////////////////////////////////////////////////////////////////////////////////////////
    // Initialize DMA timer registers (refer to MCF5235 Reference Manual
    // Section 24.2 for descriptions on the bit configurations of the
    // registers).
    //
    // To calculate the time-out period for the interrupt interval, the
    // following formula from Section 24.3.2 (Calculating Time-Out
    // Values) is used:
    //
    // T          = Time-out period in seconds
    // DIVIDER    = 1 or 16 (set via Mode Register [CLK])
    // PRESCALAR  = 1 (0x00) to 256 (0xFF) (set via Mode Register
    //             [PS])
    // REF_COMP_VAL = Reference compare value (set via 32-bit Reference
    //             Register)
    //
    // 
$$T = ( 1 / \text{CLOCKFREQ} ) * \text{DIVIDER} * ( \text{PRESCALAR} + 1 ) * ( \text{REF\_COMP\_VAL} + 1 )$$

    //
    // To get 0.001 second (1 millisecond) as the interrupt interval in
    // this example, the following is used:
    //
    // 
$$0.001 = ( 1 / 73728000 ) * 1 * ( 0 + 1 ) * ( 73727 + 1 )$$

    //
    // CLOCKFREQ    = 73,728,000 Hz is used for CLKOUT, which is half of
    //               the internal system clock (147,456,000 Hz)
    // DIVIDER      = 1 (set as "01" in Mode Register [CLK] bits 2-1)
    // PRESCALAR    = 1 (set as "0000 0000" in Mode Register [PS] bits
    //               15-8)
    // REF_COMP_VAL = 73727 (set in 32-bit Reference Register - an
    //                   interrupt will occur when the Counter Register
    //                   value matches the set reference compare value
    //                   before the counter resets)
    //
    //
    //
    // DMA Timer 0 Mode Register - Refer to Table 24-2 to see what each
    // of the bits are configuring.
    //
    sim.timer[0].tmr = 0x001A;           // 0000 0000 0001 1010

    //

```

```

// DMA Timer 0 Extended Mode Register - Refer to Table 24-3; DMA
// requests disabled and timer increments by 1.
//
sim.timer[0].txmr = 0x00; // 0000 0000

//
// DMA Timer 0 Timer Event Register - Refer to Table 24-4; clears
// the DTIM0 output reference event flag.
//
sim.timer[0].ter |= 0x02; // 0000 0010

//
// DMA Timer 0 Timer Reference Register - This is where the
// calculated REF_COMP_VAL is used for interval timing.
//
sim.timer[0].trr = 73727;

//
// DMA Timer 0 Timer Counter Register - Any writing to this register
// clears the counter; done just to be thorough, though not
// necessary.
//
sim.timer[0].tcn = 1; // Clear counter

SetIntc( 0, ( long ) &func_isr, 19, 1, 1 ); // Set the ISR and
// source vector

sim.timer[0].tmr |= 0x0001; // Enable the DMA timer

while ( 1 )
{
    OSTimeDly( TICKS_PER_SECOND );
    iprintf( "DTIM0 Count = %d\r\n", uint32_counter );
}
}

```