Tutorial Introduction

PURPOSE
- To explain how to configure and use the Timebase Module

OBJECTIVES:
- Describe the uses and features of the Timebase Module.
- Identify the steps to configure the Timebase Module.
- Write a program to configure the Timebase Module to generate periodic interrupt events.

CONTENT:
- 13 pages
- 2 questions
- 1 off-line programming exercise

LEARNING TIME:
- 20 minutes

PREREQUISITES:
- 68HC08 CPU training module

DigitalDNA
from Motorola

Welcome to this tutorial on the 68HC08 Timebase Module (TBM). This tutorial describes the features and configuration of the TBM. Please note that on subsequent pages, you will find reference buttons in the upper right of the content frame that access additional content.

Upon completion of this tutorial, you'll be able to describe the uses and features of the TBM. You'll also be able to configure the TBM to generate periodic events like auto-wake up from the low-power STOP mode.

The recommended prerequisite for this tutorial is the 68HC08 CPU training module. Click the Forward arrow when you're ready to begin the tutorial.

TBM Capabilities

- Generates periodic interrupts at user selectable rates
  - Independent of the 68HC08 timer system
  - Clocked directly from external crystal
  - Software programmable at 1 Hz, 4 Hz, 16 Hz, 256 Hz, 512 Hz, 1024 Hz, 2048 Hz, or 4096 Hz
  - Can operate as a low-power real-time clock

- Can be enabled in STOP mode
  - Allows periodic auto wake-up from STOP mode
  - Eliminates external circuitry
  - Provides very low-power operation in micro-amps

Let's begin this tutorial with a review of the TBM capabilities.

You can use the 68HC08 TBM to generate periodic interrupts at user selectable rates.   This is useful for running periodic diagnostics, servicing peripherals, performing maintenance, or any other regularly scheduled event.   You could use the output compare function of the timer system to generate a periodic interrupt, but this method includes some software overhead. Because the TBM operates independently of the timer system, it doesn't require any overhead. Using the TBM also keeps the timer channels available for the application.

In the 68HC908GP and GR product families, the TBM is clocked directly from the external crystal and not the oscillator output. MCUs with the Internal Clock Generator (ICG), such as the 68HC908KX product family, operate slightly differently than what is described in this tutorial. Check the product documentation for more information about specific differences.

When the clock source is a 32 kHz crystal, the TBM can be configured to generate periodic interrupts at a wide range of frequencies between1 Hz and 4096 Hz. Also, the TBM can generate timing events like those used in digital watches. With simple software, you can use the TBM as a low-power, cost-efficient, real-time clock. As an example, the TBM can be configured to generate an interrupt once per second. The TBM interrupt service routine would update seconds, minutes, hour, day, and year variables stored in RAM or non-volatile memory.

The TBM can be very useful in low-power applications.   To minimize current consumption and to extend battery life, the 68HC08 can be placed in STOP mode to stop the on-chip oscillator. For example, a battery operated application may enter STOP mode to wait for user input, such as from a keypad. The 68HC08 incorporates keyboard interrupt circuitry to eliminate the resistors and gating required to implement a keypad interrupt. Other battery operated applications may require the MCU to periodically wake-up to check status, take a sensor reading, or perform a diagnostic before returning to low-power STOP mode.

During STOP mode, none of the on-chip peripherals are operating since they don't have a clock source. Therefore, an external interrupt or reset is normally required to wake up from this mode and resume normal operation. The TBM can be enabled to operate directly from the crystal during STOP mode. Using the auto wake-up capability of the TBM, the CPU can wake-up from STOP mode without any external circuitry, reducing system cost.

Timebase Control Register (TBCR)

Address: $001C

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	TBIF	TBR2	TBR1	TBR0	0	TBIE	TBON	TBTST*
Write:					TACK			
Reset:	0	0	0	0	0	0	0	0

= Unimplemented   * PTM test mode

TBON — Timebase Enabled
1 = Timebase enabled
0 = Timebase disabled and the counter initialized to 0s

The Timebase enabled bit, TBON, is a read/write bit that enables the TBM. When the TBM is not needed, you can turn the Module off to reduce power consumption. The TBM counter can be initialized by clearing and then setting this bit. Reset clears the TBON bit.

The Timebase interrupt flag, TBIF, is a read-only bit that is set when the Timebase counter rolls over. When Timebase interrupts are enabled, a value of 1 in this bit position indicates that an interrupt is pending.

Timebase acknowledge bit, TACK, is a write-only bit that always reads as 0. Writing 1 to this bit clears the TBIF bit. Writing 0 to this bit has no effect.

The Timebase interrupt enabled bit, TBIE, is a read/write bit that enables Timebase interrupts. When this bit is set to 1, the TBM will generate an interrupt when the TBIF bit is set. Reset clears the TBIE bit.

The Timebase rate selection bits, TBR2-TBR0, are read/write bits that select the rate of periodic interrupts. Note that you should not change the values of these bits while the Timebase Module is enabled.

The TBTST pin is not implemented .

You can use the OSCSTOPEN bit in the CONFIG register to configure the TBM to operate during STOP mode. If the OSCSTOPEN bit is set, the TBM continues to be clocked directly from the crystal after execution of the STOP instruction. If this bit is cleared, the TBM will not operate in STOP mode.

Timebase Rate Selection

Timebase Rate Selection for OSC1 = 32.768 kHz

TBR2	TBR1	TBR0	Divider	Timebase Interrupt Rate	
				Hz	ms
0	0	0	32,768	1	1000
0	0	1	8192	4	250
0	1	0	2048	16	62.5
0	1	1	128	256	~ 3.9
1	0	0	64	512	~2
1	0	1	32	1024	~1
1	1	0	16	2048	~0.5
1	1	1	8	4096	~0.24

Digital DNA™
from Motorola

The table shows the different bit codes for the Timebase rate selection bits (TBR2-TBR0). This rate selection table is based on a 32.768 kHz external clock. As you can see, we can generate frequencies between 1 Hz and 4096 Hz. Changing the external clock will generate sub-multiples of these frequencies.
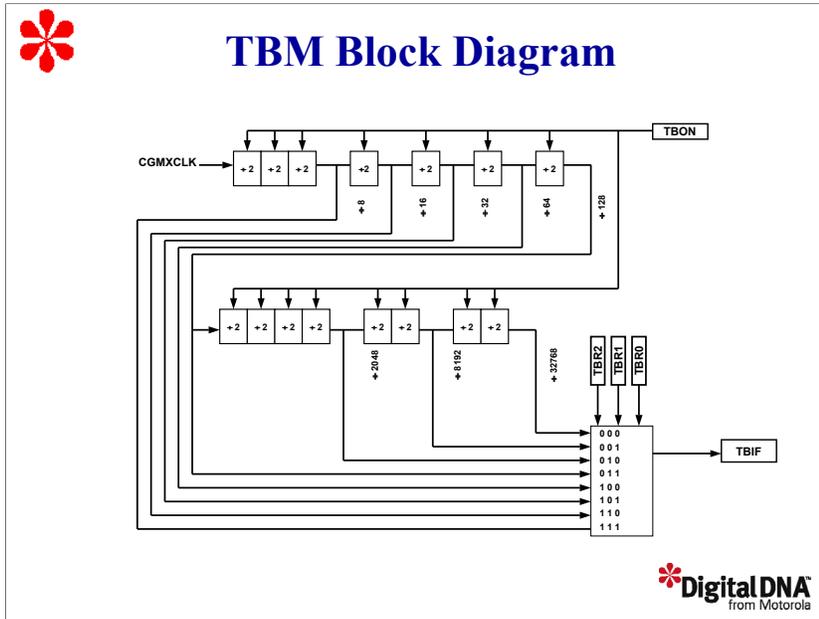
Now, let's review TBM operations using the TBM block diagram.

The TBM generates a periodic interrupt by dividing the external crystal frequency, CGMXCLK. The counter is made up of 15 divider stages that are initialized to 0 when the TBON bit is cleared.

Eight of the 15 stages are user-selectable with the Timebase rate selection bits (TBR2-TBR0). As we discussed earlier, the rate selection bits enable you to select one of eight periodic interrupt rates.

The counter starts counting when the TBON bit is set.

When the TBM has counted up to the the tap selected by TBR2-TBR0, the TBIF bit is set. If the TBIE bit is set, an interrupt request is sent to the CPU. The TBIF flag is cleared by writing a 1 to the TACK bit. Note that the first interrupt that is generated after enabling the Timebase Module will occur at approximately half of the overflow period. Subsequent events will occur at the exact period.

TBM Interrupt Vector

Source	Vector Address	Flag	Mask	Priority
TimeBase	$FFDC - $FFDD	TBIF	TBIE	16
ADC Conv. Complete	$FFDE - $FFDF	COCO	AIEN	15
Keyboard Pin	$FFE0 - $FFE1	KEYF	IMASKK	14
SCI Trans. Complete	$FFE2 - $FFE3	TC	TCIE	13
SCI Transmitter Empty		SCTE	SCTIE	
SCI Input Idle	$FFE4 - $FFE5	IDLE	ILIE	12
SCI Receiver Full		SCRF	SCRIE	
SCI Receiver Overrun	$FFE6 - $FFE7	OR	ORIE	11
SCI Noise Flag		NF	NEIE	
SCI Framing Error		FE	FEIE	
SCI Parity Error		PE	PEIE	
SPI Transmitter Empty	$FFE8 - $FFE9	SPTE	SPTIE	10
SPI Receiver Full	$FFEA - $FFEB	SPRF	SPRIE	9
SPI Overflow		OVRF	ERRIE	
SPI Mode Fault		MODF	ERRIE	
TIM2 Overflow	$FFEC - $FFED	TOF	TOIE	8
TIM2 Channel 1	$FFEE - $FFEF	CH1F	CH1IE	7
TIM2 Channel 0	$FFF0 - $FFF1	CH0F	CH0IE	6
TIM1 Overflow	$FFF2 - $FFF3	TOF	TOIE	5
TIM1 Channel 1	$FFF4 - $FFF5	CH1F	CH1IE	4
TIM1 Channel 0	$FFF6 -$FFF7	CH0F	CH0IE	3
PLL	$FFF8 - $FFF9	PLLF	PLLIE	2
IRQ	$FFA - $FFFB	IRQF	IMASK1	1
SWI	$FFFC - $FFFD	None	None	0
Reset	$FFFD - $FFFF	None	None	0

Digital DNA™ from Motorola

The TBM has its own interrupt vector to eliminate polling for the interrupt source. The vector map shown is for the 68HC908GP32 MCU. Check the device technical data book to determine the specific vector address for different 68HC08 derivatives. Note that the TBM has the lowest priority as its periodic rate is relatively slow.

6

Question

With the TBM enabled and the TBIE bit set to 0, what happens when the TBM counter rolls over? Click on your BEST choice.

a) The TBON bit is set to 1

b) The TBIF bit is set to 1

c) The TBIF bit is cleared

d) A TBM interrupt is generated

e) a and d

f) b and d

Digital DNA
from Motorola

Let's review what we've discussed so far with a couple of questions to check your understanding of the material. With the TBM enabled and the TBIE bit set to 0, what happens when the TBM counter rolls over? Click on your best choice.

Answer: When the TBM counter rolls over, it sets the TBIF flag to 1. Note that in this example, TBM interrupts are disabled because the TBIE bit is set to 0.

Question

Using a 32.768 KH crystal for the external clock source, what value do you need to set the Timebase rate selection bits to in order to generate an event every 62.5 ms? Select the values for TBR2-TBR0. Click on your choice.

a) 010
b) 101
c) 011
d) 001

Digital DNA
from Motorola

Using a 32.768 kHz crystal for the external clock source, what values do you need to set the Timebase rate selection bits to in order to generate an event every 62.5 ms? Select the values for TBR2-TBR0. Click on your choice.

Answer: An event rate of 62.5 ms gives an event frequency of 16 Hz. Dividing the clock frequency by the event frequency yields a divider value of 2048. You can select this divider value by setting TBR2-TBR0 to 010.

Programming Exercise

- Write a program to configure the TBM to generate a periodic signal.
  - Generate a 1 Hz square signal.
  - Use port C of the 68HC908GP32 MCU for output.
  - Set XTAL = 4.9152 MHz.
  - Enable TBM interrupts.
- Write a subroutine to initialize the MCU out of reset.
- Write an interrupt service routine.
  - Acknowledge Timebase interrupt.
  - Toggle port C.
  - Reset Timebase counter.

Digital DNA™
from Motorola

Now that we've discussed how to configure the TBM, let's write a program to generate a periodic signal.

For this example, let's generate a 1 Hz square signal using port C of the 68HC908GP32 MCU for the output. Use a Timebase rate, XTAL, of 4.9152 MHz and enable Timebase interrupts.

In addition to your main program, write two subroutines. Use one subroutine to initialize the MCU out of reset. Use a second subroutine as an interrupt service routine. In the interrupt service routine, acknowledge the Timebase interrupt, toggle port C, and reset the Timebase counter.

Take a moment to review the exercise instructions. When you're finished writing your program, click the Forward arrow to continue the tutorial and review the exercise solution.

Solution - MCU Initialization

```
* ------------------------------------------------------------------
* 68HC908GP32 Initialization
* ------------------------------------------------------------------
        MOV     #%00001011,CONFIG1  ; Configure the CONFIG1 Register
;                    \\\\\\\_____ COP Module disabled
;                    \\\\\\_____ STOP Instruction Enabled
;                    \\\\\_____ Stop mode recovery after 4096 CGMXCLK cycles
;                    \\\\_____ LVI operates in 5V mode
;                    \\\_____ LVI module enabled
;                    \\_____ LVI module Resets enabled
;                    \_____ LVI disabled during STOP mode
;                    _____ COP time period 262,128 cycles

        MOV     #%00000011,CONFIG2  ; Configure the CONFIG2 Register
;                    \\\\\\\_____ Internal Bus Clock used as a source for SCI
;                    \\\\\\_____ Oscillator enabled to operate during Stop Mode
;                    \\\\\_____ Voltage regulator ON (Vdd > 3.6v)
;                    \\\\_____ Unimplemented

        CLRA                        ;Initialize the Accumulator

        LDHX    #ENDRAM             ;Stack Pointer -> End of RAM
        TXS                         ;        (H:X -> SP)


; END MCU Initialization
```

Digital DNA
from Motorola

Let's review the exercise solution beginning with the MCU initialization subroutine. You can use this subroutine as a template for all of the programs you write.

This subroutine initializes the MCU using the CONFIG1 and CONFIG2 registers. Recall that these registers can be configured only once after reset to avoid inadvertently changing the MCU configuration during an application. Therefore, configuring these registers should be the first step when starting. For this example, we use CONFIG1 to disable the COP so that we don't have to feed the counter to avoid reset. We also enable the STOP instruction and enable the LVI to operate at 5V.

In CONFIG2, we select the internal bus clock as the source for the serial communication. We also enable the oscillator to operate during stop mode so that we can use the TBM periodic wake-up feature. Note that the configuration of the CONFIG2 register is not critical for this application. Alternatively, we could use the default initialization for this example.

The next instruction, CLRA, initializes the accumulator to avoid uninitialized register warnings in the simulator/debugger, although this may not be necessary in your application. The next two instructions redirect the stack pointer to the end of the RAM. Remember that in order to keep compatibility with the HC05 family, the HC08 family automatically initializes the stack pointer to $00FF after reset.

10

Solution - Main Program

```
* -------------------------------------------------------------------
* Application
* -------------------------------------------------------------------
        MOV     #$FF,DDRC               ; Enable Port C as output
        MOV     #%00000110,TBCR         ; Program the Time Base Register
;                    \\\\\\_____ Time Base Module ON
;                      \\\ _____ Enable Time Base Interrupt
;                       \\_____ Select Timebase Rate (Div 2^15)
;                                              @XTAL=4.9152MHz -> Freq=150Hz

        MOV     #COUNT,Counter          ; Initializes counter
        CLI                             ; Enable Interrupts

        BRA     *                       ; Waits for interrupt

; END Application
```

Next, compare your main program with the one provided in the solution. This section of code is where we start to work with the Timebase Module.

To generate a 1Hz square signal, we have to toggle the output every 500ms. To measure 500ms with a 4.9152MHz crystal, we need to use the higher rate divisor of 2^15, or a value of 32,768. Dividing the frequency by 32,768 gives us 150 events per second, or 75 events to generate 500ms.

In order to facilitate the process, we measure the events using interrupts. We'll configure port C to output the square signal.

Next, configure the Timebase register. Enable the Timebase interrupt, select the default Timebase rate, and turn the Module on. We need a counter to count the 75 events before we toggle the output. Reserve one byte in RAM for this counter and initialize it using a value of 75.

Finally, we enable interrupts with CLI and wait for the TBM interrupt.

Solution - Interrupt Service Routine

```
TBM_Int_Serv:
        BSET    TACK,TBCR         ; Acknowledge TB Interrupt
        DEC     Counter
        BNE     Exit              ; Decrements COUNTER and exits
                                  ; if NOT Zero
        COM     PORTC             ; If counter=0 -> Toggle Port C
                                  :   (time = 0.5 secs)
        MOV     #COUNT,Counter    ; Resets COUNTER

Exit:   RTI

; END TBM_Int_Serv
```

In the interrupt service routine, we check whether it's time to toggle the output. To do this, we first acknowledge the Timebase interrupt to avoid reentering after RTI. Next, we decrement the counter and check whether it has reached a value of zero. If not, we exit the subroutine and wait for the next event. If the counter has reached a value of zero, this means that 500 msecs have elapsed. We then toggle port C, reinitialize the counter to 75, and start over again.

Tutorial Completion

- Timebase Module Uses and Capabilities
- Timebase Configuration

Digital DNA™
from Motorola

In this tutorial, you've had an opportunity to learn typical uses and capabilities of the 68HC08 Timebase Module. You've learned how to configure and use the TBM Module and demonstrated this by writing a program to generate a 1khz square wave.