!	LATH |= _LATH_LATH2_MASK;

0x9D0008D0: LUI V0, -16506

0x9D0008D4: LW V0, 1840(V0)

0x9D0008D8: ORI V1, V0, 4

0x9D0008DC: LUI V0, -16506

0x9D0008E0: SW V1, 1840(V0)

!	LATHbits.LATH2 = 1;

0x9D0008D0: LUI V1, -16506

0x9D0008D4: LHU V0, 1840(V1)

0x9D0008D8: ADDIU A0, ZERO, 1

0x9D0008DC: INS V0, A0, 2, 1

0x9D0008E0: SH V0, 1840(V1)

!	LATHSET = _LATH_LATH2_MASK;

0x9D0008D0: LUI V0, -16506

0x9D0008D4: ADDIU V1, ZERO, 4

0x9D0008D8: SW V1, 1848(V0)

The implementation of the two commands is relatively the same.

They both are 5 instructions long. The SFR is referenced 3 times.

The implementation of the set command is a few steps shorter than the previous approaches.

The LATHbits.LATH2 = 1 command is 5 instructions long and the SFR is referenced 3 times.

The LATHSET = _LATH_LATH2_MASK; command is 3 instructions long and the SFR is referenced twice.

System clock count = 10,500,094

PR2 = 0x0000A037

Prescaler = 256

1/21MHz = 4.761904762 x 10 ^-8 * 10,500,094 = 0.5 seconds

84MHz/21MHz = 4

4 * 10,500,094 = 42,000,476

The measurement did not change from the measurement in step 16.

According to the simulator the PB3DIV was still set.

It should not have been since we removed the code that unlocked it for change.

The simulator is fallible.

SYSKEY = 0; // Ensure lock

SYSKEY = 0xAA996655; // Write Key 1

SYSKEY = 0x556699AA; // Write Key 2

PB3DIV = _PB3DIV_ON_MASK | 3 & _PB3DIV_PBDIV_MASK;

SYSKEY = 0; // Re lock

Target halted. Stopwatch cycle count = 8445938

1/84MHz * 8445938 = 0.100s

It does not correlate with what I see with my eye

0.5s / 0.1s = 5

84 x 10^6 * 5 = 420 x 10^6

The pass count option only halts the program after the breakpoint is reached a specific number of times

I was able to get set data breakpoints for local variables SRFs and global variables.

TABLE 5-1: HARDWARE VS. SOFTWARE BREAKPOINTS

Feature	Hardware Breakpoints	Software Breakpoints
Number of breakpoints	Limited	Unlimited
Breakpoints written to*	Internal debug registers	Flash Program Memory
Breakpoints applied to**	Program Memory/Data Memory	Program Memory only
Time to set breakpoints	Minimal	Dependent on oscillator speed, time to program Flash Memory and page size
Breakpoint skidding	Most devices. See the online help, Limitations section, for details.	No

* Where information about the breakpoint is written in the device.
** What kind of device feature applies to the breakpoint. This is where the breakpoint is set.

Using software breakpoints writes to flash memory so using them can wear out your flash memory.

SA = Source Address

SD = Source Data

S2A = Source 2 Address

DA = Destination Address

DD = Destination Data

LATHSET is used to set the entire SFR so it doesn't make sense to say only one bit should be 1 since they all will be using LATHSET

It would seem you are trying to toggle but LATHSET sets it all 1's so it wont toggle also if you are trying to get all 1's you can just use LATHSET without the = |1.

```
// PIC32MZ2048ECG144, EFM144 or or EFG144 based HMZ144 board Configuration Bit Settings

// DEVCFG2

#if defined(__32MZ2048EFG144__) || defined(__32MZ2048EFM144__)

#pragma config FPLLIDIV = DIV_4         // System PLL Input Divider (4x Divider) for 24MHz clock (Rev C1 board w EFG) 24MHz/4 = 6MHz

                                        // also 24MHz clock rev C board w EFM (weird - went back to C. rev D also is EFM but with Osc)

#pragma config UPLLFSEL = FREQ_24MHZ    // USB PLL Input Frequency Selection (USB PLL input is 24 MHz)

#else

#pragma config FPLLIDIV = DIV_2         // System PLL Input Divider (2x Divider) for 12 MHz crystal (Rev B and C boards w ECG) 12MHz/2 = 6MHz

#pragma config UPLLEN = OFF             // USB PLL Enable (USB PLL is disabled)

#endif

#pragma config FPLLRNG = RANGE_5_10_MHZ // System PLL Input Range (5-10 MHz Input)

#pragma config FPLLICLK = PLL_POSC      // System PLL Input Clock Selection (POSC is input to the System PLL)

#pragma config FPLLMULT = MUL_112       // System PLL Multiplier (PLL Multiply by 112) 6MHz * 112 = 672MHz

#pragma config FPLLODIV = DIV_8         // System PLL Output Clock Divider (8x Divider) 672MHz / 2 = 84MHz


// DEVCFG1

#pragma config FNOSC = SPLL             // Oscillator Selection Bits (Primary Osc (HS,EC))

#pragma config FSOSCEN = OFF            // Secondary Oscillator Enable (Disable SOSC)

#if defined(__32MZ2048EFG144__)
```

```
#pragma config POSCMOD = EC            // Primary Oscillator Configuration
EC - External clock osc

                                       // Rev C1 board w EFG uses an
Oscillator (Rev D boards too))

#else

#pragma config POSCMOD = HS            // Primary Oscillator Configuration
HS - Crystal osc

                                       // Rev B and C (w ECG or EFM) use
Crystals

#endif

#pragma config FCKSM = CSDCMD          // Clock Switching and Monitor
Selection (Clock Switch Disabled, FSCM Disabled)

#pragma config FWDTEN = OFF            // Watchdog Timer Enable (WDT
Disabled)

#pragma config FDMTEN = OFF            // Deadman Timer Enable (Deadman
Timer is disabled)

#pragma config DMTINTV = WIN_127_128   // Default DMT Count Window Interval
(Window/Interval value is 127/128 counter value)

#pragma config DMTCNT = DMT31          // Max Deadman Timer count = 2^31


// DEVCFG0

#pragma config JTAGEN = OFF            // JTAG Enable (JTAG Disabled)

#pragma config ICESEL = ICS_PGx2       // ICD/ICE is on PGEC2/PGED2 pins
(not default)


#include <xc.h>

int main()

{

    SYSKEY = 0; // Ensure lock

    SYSKEY = 0xAA996655; // Write Key 1

    SYSKEY = 0x556699AA; // Write Key 2

    PB3DIV = _PB3DIV_ON_MASK | 3 & _PB3DIV_PBDIV_MASK; // 0 = div by 1, 1 =
div by 2, 2 = div by 3 etc up to 128

    SYSKEY = 0; // Re lock

    TRISH = 1;

    PORTH = _PORTH_RH2_POSITION;

    T2CON = ((7 << _T2CON_TCKPS_POSITION) & _T2CON_TCKPS_MASK);

    PR2 = 10;

    T2CONSET = 0x8000; // Start the timer

    while(1)
```

```
    {
        while(IFS0 & _IFS0_T2IF_MASK)
        {
            LATHINV = _LATH_LATH2_MASK;
            IFS0CLR = _IFS0_T2IF_MASK; // Clear the timer interrupt status
flag
        }
    }
}
```