//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// MSP432P401 Demo - Timer0\_A3, Toggle P1.0, CCR0 Cont Mode ISR, DCO SMCLK

//

// Description: Toggle P1.0 using software and TA\_0 ISR. Timer0\_A is

// configured for continuous mode, thus the timer overflows when TAR counts

// to CCR0. In this example, CCR0 is loaded with 50000.

// ACLK = n/a, MCLK = SMCLK = TACLK = default DCO = ~3MHz

//

// MSP432P401x

// ---------------

// /|\| |

// | | |

// --|RST |

// | |

// | P1.0|-->LED

//

// William Goh

// Texas Instruments Inc.

// June 2016 (updated) | November 2013 (created)

// Built with CCSv6.1, IAR, Keil, GCC

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

#include "msp.h"

#define CYCLETIMER0 750 //Adjust this for lower bit

#define CYCLETIMER1 1500 //Adjust this for upper bit

int main(void) {

WDT\_A->CTL = WDT\_A\_CTL\_PW | // Stop WDT

WDT\_A\_CTL\_HOLD;

// Configure P5.0 P5.1

P5 -> DIR |= BIT0 | BIT1;

P5 -> OUT = 0;

CS->KEY = CS\_KEY\_VAL; // unlock CS registers

CS->CTL0 = 0; // clear register CTL0

CS->CTL0=CS\_CTL0\_DCORSEL\_0; //1.5 MHz

CS->CTL1 = CS\_CTL1\_SELA\_2 | CS\_CTL1\_SELS\_3 | CS\_CTL1\_SELM\_3; // select clock sources

CS->KEY = 0; // lock the CS registers

TIMER\_A0->CCTL[0] = TIMER\_A\_CCTLN\_CCIE; // TACCR0 interrupt enabled

TIMER\_A0->CCTL[1] = TIMER\_A\_CCTLN\_CCIE; // TACCR1 interrupt enabled

TIMER\_A0->CCR[0] = CYCLETIMER0; //Set CCR0 for on cycle

TIMER\_A0->CCR[1] = CYCLETIMER1; //SET CCR1 for cycle

TIMER\_A0->CTL = TIMER\_A\_CTL\_SSEL\_\_SMCLK | // SMCLK, continuous mode

TIMER\_A\_CTL\_MC\_\_CONTINUOUS;

SCB->SCR |= SCB\_SCR\_SLEEPONEXIT\_Msk; // Enable sleep on exit from ISR

// Enable global interrupt

\_\_enable\_irq();

NVIC->ISER[0] = 1 << ((TA0\_0\_IRQn) & 31);

NVIC->ISER[0] = 1 << ((TA0\_N\_IRQn) & 31);

while (1)

{

\_\_sleep();

\_\_no\_operation(); // For debugger

}

}

// Timer A0 interrupt service routine

void TA0\_0\_IRQHandler(void) {

TIMER\_A0->CCTL[0] &= ~TIMER\_A\_CCTLN\_CCIFG;

TIMER\_A0->CCR[0] += CYCLETIMER0; //increment timer

if (P5->OUT & BIT0)

P5 -> OUT &= ~BIT0;

else

P5-> OUT |= BIT0;

}

void TA0\_N\_IRQHandler(void) {

if (TIMER\_A0->CCTL[1] & TIMER\_A\_CCTLN\_CCIFG)

{

TIMER\_A0->CCTL[1] &= ~TIMER\_A\_CCTLN\_CCIFG;

TIMER\_A0->CCR[1] += CYCLETIMER1; //increment timer

if (P5->OUT & BIT1)

P5 -> OUT &= ~BIT1;

else

P5-> OUT |= BIT1;

}

}