// Lab 7: Concurrency via Interrupts

// Programming multiple interrupt events 🡪 Having interaction of the interrupts with each other!

// Having a sense of concurrency in the code according to which multiple events are processed in an overlapping way Vs. Sequential way!

// 7.1: Long Pulse on the LED (non-renewing interval)

// Writing a program that turns ON the LED for three seconds. @ pushing of button!

// Having non-renewing interval 🡪 Pushing the button again during the interval 🡪 The timer does not renew!

// Timing the three-second interval using Timer\_A with interrupts!

// Using any mode (up-mode Vs. continuous-mode) + the preferred channel!

// Engaging a low-power mode 🡪 Waiting for the button push and during the three-second interval.

// Program 🡪 Having two ISRs: the button’s and the timer’s!

// Two interrupt events interaction with one another 🡪 The button’s ISR enables the timer’s interrupt.

// Three-Second Interval for LED: (a) Button Interrupt; (b) Button Pushes; (c) OFF LED

// The program has two ISRs 🡪 (A) the button; and (b) the timer.

// Two interrupt events have interaction with each other.

// The button’s ISR enables the timer’s interrupt.

// The timer’s ISR @ the end of the three-second interval 🡪 Re-enabling the button’s interrupt.

// The timer’s interrupt 🡪 Re-enabling the button’s interrupt.

// Interrupt events 🡪 They are enabled/disabled multiple times!

// Disabling the button’s interrupt at the end of the three second interval!

// The timer 🡪 It does not keep raising interrupts if the button is not pushed again!

// Timer’s interrupt 🡪 Disabling at the end of the three-second interval 🡪 The timer does not keep raising interrupts if the button is not pushed again!

**#include** <msp430fr6989.h>

**#define** BUT1 BIT1 //Push Button S1 location is P1.1

**#define** redLED BIT0 // Red LED location is P1.0

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

// Reconfigures ACLK to be rerouted to the 32 KHz crystal on the LaunchPad

**void** **config\_ACLK\_to\_32KHz\_crystal**() {

// The default mode of the ACLK is a built-in oscillator at a frequency of 39KHz normally.

// Rerouted the pins to LFXIN/LFXOUT functionality so that the ACLK can be routed to the 32KHz crystal.

// This information can be found using the LaunchPad user's guide (page 29) and the chip's data sheet (page 123).

PJSEL1 &= ~BIT4;

PJSEL0 |= BIT4;

// We need to for the crystal to settle, once it has started.

// Therefore, we will wait until the local and global oscillator fault flags are cleared and remain cleared.

CSCTL0 = CSKEY; // Unlock CS registers, to divert the pins for the crystal functionality.

//Clears the flag and will do so until they remain cleared.

**do** {

CSCTL5 &= ~LFXTOFFG; // Local oscillator fault flag

SFRIFG1 &= ~OFIFG; // Global oscillator fault flag

} **while**((CSCTL5 & LFXTOFFG) != 0);

CSCTL0\_H = 0; // Lock CS registers, returns the pins.

**return**;

}

/\*\*

\* main.c

\*/

**void** **main**(**void**)

{

WDTCTL = WDTPW | WDTHOLD; // Stops the watchdog timer. We do this so the MCU doesn't reset itself periodically.

PM5CTL0 &= ~LOCKLPM5; // We enable the general purpose I/O pins.

// Led Configuration

P1DIR |= redLED;

P1OUT &= ~redLED;

// Configuring buttons with interrupt

P1DIR &= ~BUT1;// Sets the push button S1 as input

P1REN |= BUT1; // Enables the internal resistor found in P1.1

P1OUT |= BUT1; // Sets the resistor as a pull-up

P1IE |= BUT1; // Enables the interrupt enable bit of push button S1

P1IES |= BUT1; // Configures the interrupt raise event as falling edge

P1IFG &= ~BUT1;// Clears the interrupt flag of push button S1

// Reroutes the ACLK to the 32kHz crystal

config\_ACLK\_to\_32KHz\_crystal();

// Timer configuration (ACLK) (frequency division by 2) (Stop mode) (clear TAR).

TA0CTL = TASSEL\_1 | ID\_1 | MC\_0 | TACLR;

TA0CCR0 = 49151; // @16KHz this will generate a delay of 3 seconds.

TA0CCTL0 |= CCIE; // Enables the interrupt enable bit of Timer A module 0 channel 0

TA0CCTL0 &= ~CCIFG; // Clears the interrupt flag of Timer A module 0 channel 0

// Engages low power mode 3 since we are using the ACLK.

\_low\_power\_mode\_3();

}

// Push Button raised an interrupt event

**#pragma** vector = PORT1\_VECTOR

**\_\_interrupt** **void** **PORT1\_ISR**() {

TA0CTL |= TACLR; // Clears the TAR of the timer

TA0CTL ^= MC\_1; // Toggles the timer between Stop Mode and Up Mode

P1OUT |= redLED; // Turns on the Red LED

P1IFG &= ~BUT1; // Clears the interrupt flag of the push button S1

P1IE &= ~BUT1; // Disables the interrupt enable bit of the push button S1

}

// Timer raised an interrupt event

**#pragma** vector = TIMER0\_A0\_VECTOR

**\_\_interrupt** **void** **T0A0\_ISR**() {

P1OUT &= ~redLED; // Turns off the Red LED

P1IE |= BUT1; // Enables the interrupt enable bit of the push button S1

P1IFG &= ~ BUT1; // Clears the interrupt flag of the push button S1

TA0CTL ^= MC\_1; // Toggles the timer between Stop Mode and Up Mode

TA0CTL |= TACLR; // Clears the TAR of the timer

// Hardware clears Channel 0 flag (CCIFG in TA0CCTL0)

}

// 7.2: Long Pulse on the LED (renewing interval)

// Making a renewing interval

// Pushing the button during the three-second interval 🡪 the timer renews from the moment it is pushed the last time!

// User 🡪 Keep pushing the button midway through the interval 🡪 the LED remains lit continuously!

**#include** <msp430fr6989.h>

**#define** BUT1 BIT1 //Push Button S1 location is P1.1

**#define** redLED BIT0 // Red LED location is P1.0

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

// Reconfigures ACLK to be rerouted to the 32 KHz crystal on the LaunchPad

**void** **config\_ACLK\_to\_32KHz\_crystal**() {

// The default mode of the ACLK is a built-in oscillator at a frequency of 39KHz normally.

// Rerouted the pins to LFXIN/LFXOUT functionality so that the ACLK can be routed to the 32KHz crystal.

// This information can be found using the LaunchPad user's guide (page 29) and the chip's data sheet (page 123).

PJSEL1 &= ~BIT4;

PJSEL0 |= BIT4;

// We need to for the crystal to settle, once it has started.

// Therefore, we will wait until the local and global oscillator fault flags are cleared and remain cleared.

CSCTL0 = CSKEY; // Unlock CS registers, to divert the pins for the crystal functionality.

//Clears the flag and will do so until they remain cleared.

**do** {

CSCTL5 &= ~LFXTOFFG; // Local oscillator fault flag

SFRIFG1 &= ~OFIFG; // Global oscillator fault flag

} **while**((CSCTL5 & LFXTOFFG) != 0);

CSCTL0\_H = 0; // Lock CS registers, returns the pins.

**return**;

}

/\*\*

\* main.c

\*/

**void** **main**(**void**)

{

WDTCTL = WDTPW | WDTHOLD; // Stops the watchdog timer. We do this so the MCU doesn't reset itself periodically.

PM5CTL0 &= ~LOCKLPM5; // We enable the general purpose I/O pins.

// Led Configuration

P1DIR |= redLED; // Sets the LED as output

P1OUT &= ~redLED; // Starts Off

// Configuring buttons with interrupt

P1DIR &= ~BUT1;// Sets the push button S1 as input

P1REN |= BUT1; // Enables the internal resistor found in P1.1

P1OUT |= BUT1; // Sets the resistor as a pull-up

P1IE |= BUT1; // Enables the interrupt enable bit of push button S1

P1IES |= BUT1; // Configures the interrupt raise event as falling edge

P1IFG &= ~BUT1;// Clears the interrupt flag of push button S1

// Reroutes the ACLK to the 32kHz crystal

config\_ACLK\_to\_32KHz\_crystal();

// Timer configuration (ACLK) (frequency division by 2) (Stop mode) (clear TAR).

TA0CTL = TASSEL\_1 | ID\_1 | MC\_0 | TACLR;

TA0CCR0 = 49151; // @16KHz this will generate a delay of 3 seconds.

TA0CCTL0 |= CCIE; // Enables the interrupt enable bit of Timer A module 0 channel 0

TA0CCTL0 &= ~CCIFG; // Clears the interrupt flag of Timer A module 0 channel 0

// Engages low power mode 3 since we are using the ACLK.

\_low\_power\_mode\_3();

}

// Push Button raised an interrupt event

**#pragma** vector = PORT1\_VECTOR

**\_\_interrupt** **void** **PORT1\_ISR**() {

TA0CTL |= TACLR; // Clears the TAR of the Timer

TA0CTL |= MC\_1; // Sets the timer from Stop Mode to Up Mode

P1OUT |= redLED; // Turns on the Red LED

P1IFG &= ~BUT1; // Clears the interrupt flag of the push button S1

}

**#pragma** vector = TIMER0\_A0\_VECTOR

**\_\_interrupt** **void** **T0A0\_ISR**() {

P1OUT &= ~redLED; // Turns off the Red LED

TA0CTL &= ~MC\_3; // Sets the timer to Stop Mode

// Hardware clears Channel 0 flag (CCIFG in TA0CCTL0)

}

// 7.3: Button Debouncing

// Implementing a push button debouncing algorithm.

// A code is written for debouncing the button so that it works every time!

// The algorithm 🡪 Taking two samples of the button separated by the maximum bounce duration.

// Pushing button 🡪 Raising of an interrupt in the first falling edge (button is active low)!

// 20ms Timer 🡪 Representing the maximum bounce duration!

// Disabling the button interrupt during this interval 🡪 Causing further interrupts!

// The 20ms interval 🡪 Checking the button status + Interpreting a button push (toggling the LED)!

// Waiting for the maximum bounce duration 🡪 Releasing the button and interpreting a button push!

// Bouncing severity 🡪 Depending on the quality of the button!

// Push buttons of higher quality have shorter bounce durations!

// Buttons with built-in debouncers 🡪 No bounces at all exist!

// 20ms Duration 🡪 Our algorithm represents the maximum bounce duration!

// Continuation of bouncing 🡪 Observed!

// No observation of bouncing 🡪 Decreasing this duration until the button works reliably.

// Introduction of a delay of 20ms into our algorithm effectively.

// The algorithm capability 🡪 Toggling the LED when the button is pushed.

// Reading the button via interrupt

// Timing the interval using Timer\_A with interrupt!

// Engaging a low-power mode while waiting for the push button and during the timed interval!

**#include** <msp430fr6989.h>

**#define** BUT1 BIT1

**#define** redLED BIT0

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

// Reconfigures ACLK to be rerouted to the 32 KHz crystal on the LaunchPad

**void** **config\_ACLK\_to\_32KHz\_crystal**() {

// The default mode of the ACLK is a built-in oscillator at a frequency of 39KHz normally.

// Rerouted the pins to LFXIN/LFXOUT functionality so that the ACLK can be routed to the 32KHz crystal.

// This information can be found using the LaunchPad user's guide (page 29) and the chip's data sheet (page 123).

PJSEL1 &= ~BIT4;

PJSEL0 |= BIT4;

// We need to for the crystal to settle, once it has started.

// Therefore, we will wait until the local and global oscillator fault flags are cleared and remain cleared.

CSCTL0 = CSKEY; // Unlock CS registers, to divert the pins for the crystal functionality.

//Clears the flag and will do so until they remain cleared.

**do** {

CSCTL5 &= ~LFXTOFFG; // Local oscillator fault flag

SFRIFG1 &= ~OFIFG; // Global oscillator fault flag

} **while**((CSCTL5 & LFXTOFFG) != 0);

CSCTL0\_H = 0; // Lock CS registers, returns the pins.

**return**;

}

/\*\*

\* main.c

\*/

**void** **main**(**void**)

{

WDTCTL = WDTPW | WDTHOLD; // Stops the watchdog timer. We do this so the MCU doesn't reset itself periodically.

PM5CTL0 &= ~LOCKLPM5; // We enable the general purpose I/O pins.

// Led Configuration

P1DIR |= redLED; // Sets the LED as output

P1OUT &= ~redLED; // Starts Off

// Configuring buttons with interrupt

P1DIR &= ~BUT1;// Sets the push button S1 as input

P1REN |= BUT1; // Enables the internal resistor found in P1.1

P1OUT |= BUT1; // Sets the resistor as a pull-up

P1IE |= BUT1; // Enables the interrupt enable bit of push button S1

P1IES |= BUT1; // Configures the interrupt raise event as falling edge

P1IFG &= ~BUT1;// Clears the interrupt flag of push button S1

// Reroutes the ACLK to the 32kHz crystal

config\_ACLK\_to\_32KHz\_crystal();

// Timer configuration (ACLK) (frequency division by 1) (Continuous mode) (clear TAR).

TA0CTL = TASSEL\_1 | ID\_0 | MC\_2 | TACLR;

TA0CCTL1 &= ~CCIE; // Disables the interrupt enable bit of Timer A module 0 channel 1

TA0CCTL1 &= ~CCIFG; // Clears the interrupt flag of Timer A module 0 channel 1

// Engages low power mode 3 since we are using the ACLK

\_low\_power\_mode\_3();

}

// Push Button raised an interrupt event

**#pragma** vector = PORT1\_VECTOR

**\_\_interrupt** **void** **PORT1\_ISR**() {

TA0CCTL1 |= CCIE; // Enable the interrupt enable bit of Timer A module 0 channel 1

TA0CCTL1 &= ~CCIFG; // Clears the interrupt flag of Timer A module 0 channel 1

TA0CCR1 = TA0R + 655; // Schedule the next interrupt event to 20ms

P1IFG &= ~BUT1; // Clears the interrupt flag of the push button S1

P1IE &= ~BUT1; // Disables the interrupt enable bit of push button S1

}

// Timer raised an interrupt event

**#pragma** vector = TIMER0\_A1\_VECTOR

**\_\_interrupt** **void** **T0A1\_ISR**() {

// Checks to see if the button is still pushed

**if**((P1IN & BUT1) == 0)

P1OUT ^= redLED; // Toggles Red LED

P1IE |= BUT1; // Enables the interrupt enable bit of push button S1

TA0CCTL1 &= ~CCIE; // Disables the interrupt enable bit of Timer A module 0 channel 1

TA0CCTL1 &= ~CCIFG; // Clears the interrupt flag of Timer A module 0 channel 1

}