LEDButton.asm

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;                                                                               ;
;                                LEDButton                                      ;
;                                EE110a HW1                                     ;
;                                                                               ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

; Description:      This program toggles the CC26xR Launch Pad LEDs based on
;                   the button that is pressed. When BTN1 (left) is pressed,
;                   the RED LED is on, and when it is released, the RED LED is off.
;                   Respectively applied for BTN2 (right) and GREEN LED.
;
; Operation:        The program sets up the hardware by initializing power, timers,
;                   and GPIO. The push buttons are read as inputs via GPIO pins
13/14,
;                   while the LEDs are written to as outputs via GPIO pins 6/7.
;                   Note that the push buttons are pulled low when pressed, so
;                   they are hooked up to pull up resistors via the MCU.
;                   This program does NOT setup a stack, since it is not needed
;                   (no nested code).
;
; References:       CC26xR launchpad pin mapping and schematic:
;                   https://www.ti.com/tool/LAUNCHXL-CC26X2R1#tech-docs
;
; Input:            BTN1 and BTN2 push buttons on the CC2xR Launchpad.
; Output:           The RED and GREEN LEDs on the CC26xR launch pad are toggled.
;
; User Interface:   Two buttons (BTN1 and BTN2) can be pressed on the CC26xR
;                   launch pad.
;
; Error Handling:   None.
;
;
; Revision History:
;    10/27/25  Steven Lei       initial revision
;    10/30/25  Steven Lei       final revision, HW1



; local include

; utilities
 .include  "GeneralMacros.inc"
 .include  "GeneralConstants.inc"

; CC26×2 hardware
 .include  "CPUreg.inc"
 .include  "GPIOreg.inc"
 .include  "IOCreg.inc"

```

```
; This program specific
 .include  "LEDButton.inc"


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
; data
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

        .data

; Stack goes here normally, but not needed since no nested subroutines

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
; code
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

        .text

        .global resetISR

resetISR:

Init:                                   ; setup CC26x2 hardware

        BL      InitPower               ;turn on power to everything
        BL      InitClocks              ;turn on clocks to everything
        BL      InitGPIO                ;setup the I/O (only output)
                                        ;initialize the variables

        MOV32   R2, GPIO_BASE_ADDR      ;use R2 to access GPIO registers
        STREG   ((1 << RED_LED_IO_BIT) |(1 << GREEN_LED_IO_BIT)), R2,
GPIO_DCLR31_0_OFF
                                        ;       and turn both LEDs off

HandleButtonPresses:                    ; Toggle LEDs when button pressed/released

        MOV32   R1, GPIO_BASE_ADDR              ;read button input from base addr
        LDR     R0, [R1, #GPIO_DIN31_0_OFF]     ;       + offset

        ; Just shift button bits down to where LED bits are, both 32 bit aligned
        LSR     R0, R0, #(LEFT_BTN_IO_BIT - RED_LED_IO_BIT)
        EOR     R0, R0, #ALL_ONES               ;LED bit on when button bit is low,
                                                ;       and vice versa, so negate
        STR     R0, [R1, #GPIO_DOUT31_0_OFF]    ;write to LEDs via GPIO output

DoneHandleButtonPresses:                ;done checking button presses
        B       HandleButtonPresses             ;so do it again, loop forever

```

```
101         BX      LR                              ;should never get here. just ret
102
103
104 ; InitPower
105 ;
106 ; Description:       Turn on the power to the peripherals.
107 ;
108 ; Operation:         Setup PRCM registers to turn on power to the peripherals.
109 ;
110 ; Arguments:         None.
111 ; Return Value:      None.
112 ;
113 ; Local Variables:   None.
114 ; Shared Variables:  None.
115 ; Global Variables:  None.
116 ;
117 ; Input:             None.
118 ; Output:            None.
119 ;
120 ; Error Handling:    None.
121 ;
122 ; Algorithms:        None.
123 ; Data Structures:   None.
124 ;
125 ; Registers Changed: flags, R0, R1
126 ; Stack Depth:       0 words
127 ;
128 ; Revision History:  02/17/21   Glen George      initial revision
129
130 InitPower:
131
132         MOV32   R1, PRCM_BASE_ADDR              ;get base for power registers
133
134         STREG   PD_PERIPH_EN, R1, PDCTL0_OFF    ;turn on peripheral power
135
136 WaitPowerOn:                                    ;wait for power on
137         LDR     R0, [R1, #PDSTAT0_OFF]          ;get power status
138         ANDS    R0, #PD_PERIPH_STAT             ;check if power is on
139         BEQ     WaitPowerOn                     ;if not, keep checking
140         ;BNE    DonePeriphPower                 ;otherwise done
141
142
143 DonePeriphPower:                                ;done turning on peripherals
144         BX      LR
145
146
147 ; InitClocks
148 ;
149 ; Description:       Turn on the clock to the peripherals.
150 ;
151 ; Operation:         Setup PRCM registers to turn on clock to the peripherals.
152 ;
```

```
153 ; Arguments:         None.
154 ; Return Value:      None.
155 ;
156 ; Local Variables:   None.
157 ; Shared Variables:  None.
158 ; Global Variables:  None.
159 ;
160 ; Input:             None.
161 ; Output:            None.
162 ;
163 ; Error Handling:    None.
164 ;
165 ; Algorithms:        None.
166 ; Data Structures:   None.
167 ;
168 ; Registers Changed: flags, R0, R1
169 ; Stack Depth:       0 words
170 ;
171 ; Revision History:  02/17/21   Glen George      initial revision
172 ;                    10/28/25   Steven Lei       remove GPT0CLK, not used
173
174 InitClocks:
175
176
177         MOV32   R1, PRCM_BASE_ADDR              ;get base for power registers
178
179         STREG   GPIOCLK_EN, R1, GPIOCLKGR_OFF   ;turn on GPIO clocks
180         STREG   GPT0CLK_EN, R1, GPTCLKGR_OFF    ;turn on Timer 0 clocks
181         STREG   GPTCLKDIV_1, R1, GPTCLKDIV_OFF  ;timers get system clock
182
183         STREG   CLKLOADCTL_LD, R1, CLKLOADCTL_OFF  ;load clock settings
184
185 WaitClocksLoaded:                               ;wait for clocks to be loaded
186         LDR     R0, [R1, #CLKLOADCTL_OFF]       ;get clock status
187         ANDS    R0, #CLKLOADCTL_STAT            ;check if clocks are on
188         BEQ     WaitClocksLoaded                ;if not, keep checking
189         ;BNE    DoneClockSetup                  ;otherwise done
190
191
192 DoneClockSetup:                                 ;done setting up clock
193         BX      LR
194
195
196 ; InitGPIO
197 ;
198 ; Description:       Initialize the I/O pins for the LEDs and push buttons.
199 ;                    Note that the push buttons need to be pulled up since
200 ;                    they are pulled low when pressed (from schematics).
201 ;
202 ; References:        Schematics for CC26XR1 Launch Pads
203 ;                    https://www.ti.com/tool/LAUNCHXL-CC26X2R1
204 ;
```

```
205 ; Operation:         Setup GPIO pins 6 and 7 to be 4 mA outputs for the LEDs,
206 ;                    pins 13 and 14 to be inputs with pullups for the push buttons.
207 ;
208 ; Arguments:         None.
209 ; Return Value:      None.
210 ;
211 ; Local Variables:   None.
212 ; Shared Variables:  None.
213 ; Global Variables:  None.
214 ;
215 ; Input:             None.
216 ; Output:            None.
217 ;
218 ; Error Handling:    None.
219 ;
220 ; Algorithms:        None.
221 ; Data Structures:   None.
222 ;
223 ; Registers Changed: flags, R0, R1
224 ; Stack Depth:       0 words
225 ;
226 ; Revision History:  02/17/21   Glen George      initial revision
227 ;                    10/28/25   Steven Lei       fork from EHDemo.s, add push buttons
228
229 InitGPIO:
230
231         MOV32   R1, IOC_BASE_ADDR       ;get base addr for I/O control registers
232         MOV32   R0, IOCFG_GEN_DOUT_4MA  ;setup for general 4 mA outputs
233         STR     R0, [R1, #IOCFG6]       ;write config for red LED I/O
234         STR     R0, [R1, #IOCFG7]       ;write config for green LED I/O
235
236                                         ;R1 still has base addr for IOCFG regs
237         MOV32   R0, IOCFG_DIN_PULL_UP   ;input with pullup, since button down is low
238         STR     R0, [R1, #IOCFG13]      ;write config for left push button I/O
239         STR     R0, [R1, #IOCFG14]      ;write config for right push button I/O
240
241                                         ;enable outputs for the GPIO pins
242         MOV32   R1, GPIO_BASE_ADDR      ;get base addr for GPIO registers
243         STREG   ((1 << RED_LED_IO_BIT) | (1 << GREEN_LED_IO_BIT)), R1,
    GPIO_DOE31_0_OFF
244                                         ;       and write the enable
245
246         BX      LR                      ;done so return
```