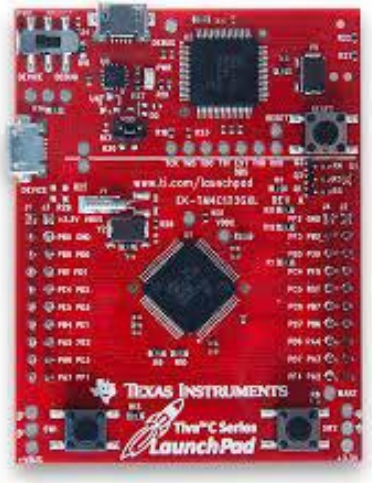
SDU

systick



Use of systick

- Used for precise timing
- Whenever the timer runs out it will trigger an interrupt
- We'll set it for 5ms
- Will be relied upon throughout the semester

SDU

Create a new project

- Create a new project named “systick_test” following instructions from Lab1
- Download and unpack systick_minimum.zip from itslearning
- Add three files from zip to newly created project

SDU

Set up interrupt handler

- Open the following file which was created automatically when you created the project

SDU

Set up handler

- Add the following lines to enable the interrupt handler
- Compile and run
- A blue light should be blinking on your board

```
32 void ResetISR(void);
33 static void NmiSR(void);
34 static void FaultISR(void);
35 static void IntDefaultHandler(void);
36
37 void systick_handler(void);
38
39 #ifndef HWREG
40 #define HWREG(x) (*((volatile uint32_t *)(x)))
41 #endif
42
43 //*****************************************************************************
44 //
45 // The entry point for the application.
46 //
47 //*****************************************************************************
48 extern int main(void);
49
50 //*****************************************************************************
51 //
52 // Reserve space for the system stack.
53 //
54 //*****************************************************************************
55 static uint32_t pui32Stack[128];
56
57 //*****************************************************************************
58 //
59 // External declarations for the interrupt handlers used by the application.
60 //
61 //*****************************************************************************
62 // To be added by user
63
64 //*****************************************************************************
65 //
66 // The vector table.  Note that the proper constructs must be placed on this to
67 // ensure that it ends up at physical address 0x0000.0000 or at the start of
68 // the program if located at a start address other than 0.
69 //
70 //*****************************************************************************
71 __attribute__ ((section(".intvecs")))
72 void (* const g_pfnVectors[])(void) =
73 {
74     (void (*)(void))((uint32_t)pui32Stack + sizeof(pui32Stack)),
75                                             // The initial stack pointer
76     ResetISR,                               // The reset handler
77     NmiSR,                                  // The NMI handler
78     FaultISR,                               // The hard fault handler
79     IntDefaultHandler,                      // The MPU fault handler
80     IntDefaultHandler,                      // The bus fault handler
81     IntDefaultHandler,                      // The usage fault handler
82     0,                                      // Reserved
83     0,                                      // Reserved
84     0,                                      // Reserved
85     0,                                      // Reserved
86     IntDefaultHandler,                      // SVCall handler
87     IntDefaultHandler,                      // Debug monitor handler
88     0,                                      // Reserved
89     IntDefaultHandler,                      // The PendSV handler
90     systick_handler,                        // The SysTick handler
91     IntDefaultHandler,                      // GPIO Port A
```

Systick.c

```c
#include <stdint.h>
#include "tm4c123gh6pm.h"



#define SYSTICK_RELOAD_VALUE 80000              // 5 mS

// Missing definitions in tm4c123gh6pm.h file
#define NVIC_INT_CTRL_PEND_SYST   0x04000000    // Pend a systick int
#define NVIC_INT_CTRL_UNPEND_SYST 0x02000000    // Unpend a systick int

#define SYSTICK_PRIORITY    0x7E

volatile int ticks = 0;

void systick_handler(void)
/*****************************************************************************
*   Function : See module specification (.h-file).
*****************************************************************************/
{
  // Hardware clears systick int request
  ticks++;
}

```


SDU

Main.c

```c
/**
 * main.c
 */

#include <stdint.h>
#include "tm4c123gh6pm.h"
#include "systick.h"

#define TIM_1_SEC     200

extern int ticks;

int main(void)
{

    int alive_timer = TIM_1_SEC;

    disable_global_int();
    init_systick();
    enable_global_int();

    int dummy;
    SYSCTL_RCGC2_R = SYSCTL_RCGC2_GPIOF;        // enable the GPIO port that is used for the on-board LEDs and switches
    dummy = SYSCTL_RCGC2_R;                     // dummy read to insert a few cycles after enabling the peripheral
    GPIO_PORTF_DIR_R = 0x0E;                    // set the direction as output for LED pins on PortF (PF1 - PF3)
    GPIO_PORTF_DEN_R = 0x1E;                    // enable the GPIO pins for digital function (PF1 - PF4)
    GPIO_PORTF_PUR_R = 0x10;                    // enable internal pull-up resistor for switch (PF4)

    while(1)                                    // loop forever
    {
        while( !ticks );

        // The following will be executed every 5mS
        ticks--;

        if( ! --alive_timer )
        {
          alive_timer        = TIM_1_SEC;
          GPIO_PORTF_DATA_R ^= 0x04;
        }
    }
    return 0;
}
```