2. Migrar el proyecto sapi\_examples/edu-ciaa-nxp/bare\_metal/gpio/tick\_01\_tickHook (uso de tickHooks c/sAPI) a: projects/TP1

a. Mediante compilación condicional, mantener en el archivo TP1.c los fuentes del TP1-1, TP1-2 y TP1-3

b. Identificar funciones de librería sAPI útiles para el colgarse de un tick

i. Documentar mediante tablas c/texto e imágenes la secuencia de funciones invocadas durante la ejecución del ejemplo de aplicación, en qué archivo se encuentran, su descripción detallada, qué efecto tiene la aplicación sobre el hardware (identificar circuitos, puertos, pines, niveles, etc.) así como la interacción entre las mismas (tanto en ResetISR() como en main())

ii. Idem c pero con datos (definiciones, constantes, variables, estructuras, etc.) (tanto en ResetISR() como en main())

Respuesta

2.b

Es conveniente arrancar por la funcion que realizara todo el trabajo del programa. myTickHook es una funcion que denominada funcion de callback, es decir, una funcion que se declara por fuera del main que es llamada bajo ciertas circunstancias, generalmente por interrupciones.

La funcion se encuentra en el mismo directorio que el main

/firmware\_v2/sapi\_examples/edu-ciaa-nxp/bare\_metal/tick\_01\_tickHook/src/tickHook.c

La funcion como se ve en la fig.1 recive un puntero void (un puntero el cual no esta definido el tipo de variable) y devuelve un void tras terminar su ejecucion. (Algo no muy recomendable si se trabaja con punteros)

Por dentro se declara la variable LesState como tipo booleano y se la precarga con off (false). Es importante notar que tiene el modificador static lo cual hace que una vez que se llame de nuvo a la funcion no se volvera a declarar y permanecera con off hasta que se lo cambie. Luego se declara la variable led de tipo gpioMap\_t y se la inicializa con el valor del puntero (es decir led es un puntero que apunta a lo que se halla pasado como argumento).

Luego aparece una instrucción condicional IF que aprovecha el tipo booleano de LedState y pregunta por defecto si la instrucción LedState es verdadera, lo cual es falso y cargara posteriormente a LedState como On (true). Como LesState tiene el modificador statico cuando vuelva a entrada en esa funcion inicializara en On y al entrar al bloque condicional se postcargara con Off haciendo que el estado cambie entre On y Off cada vez que se entra a la funcion.

Por ultimo utilizara gpioWrite para escribir el estado sobre el led que se halla pasado como argumento.

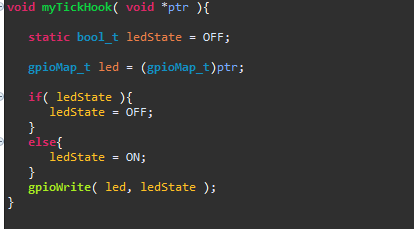


Figura 1

La funcion main se detalla en la figura 2

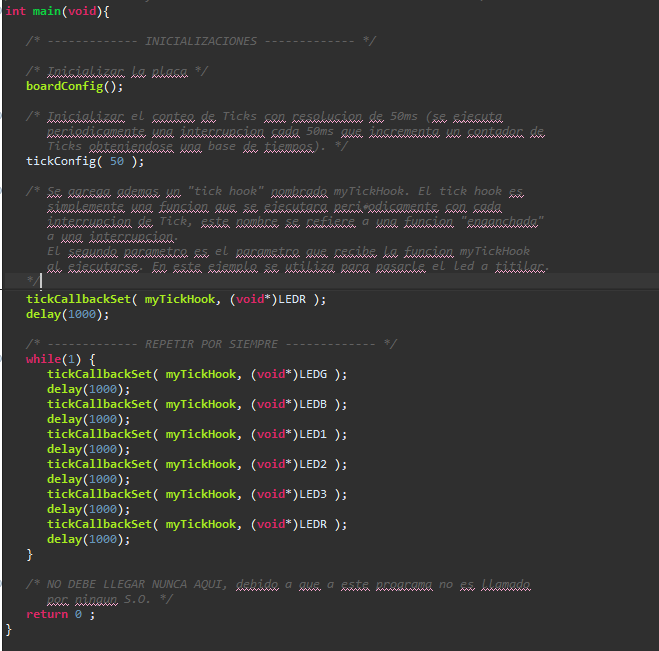


Figura 2.

La misma comenzara con un boardConfig() como es necesario para inicializar los puertos de la EduCiaa, luego llamara a la funcion tickConfig().

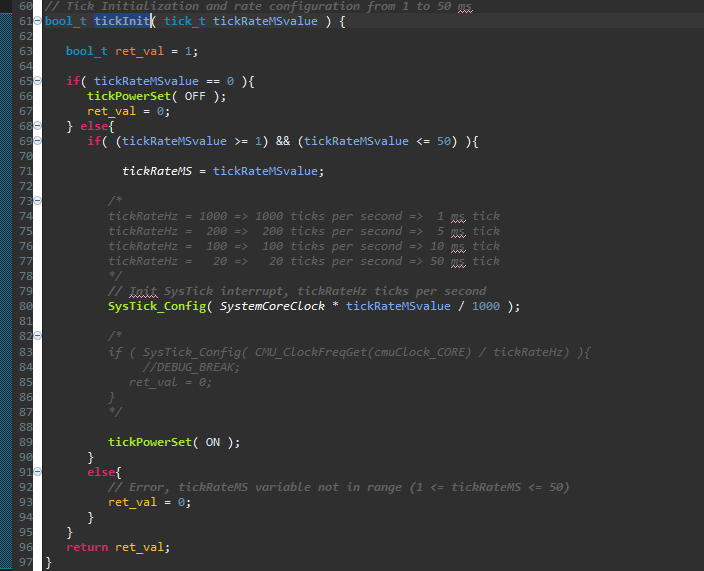


La funcion tickConfig() esta declarada en

/firmware\_v2/modules/lpc4337\_m4/sapi/inc/sapi\_tick.h

Basicamente es un cambio de nombre lo que se hizo. El precompilador cuando ve tickConfig lo reemplaza por tickInit, es decir, en el main se esta llamando implicitamente a la funcion tickInit(), declarada en el mismo lugar que tickConfig(), donde su contenido esta en su correspondiente .c

/firmware\_v2/modules/lpc4337\_m4/sapi/src/sapi\_tick.c

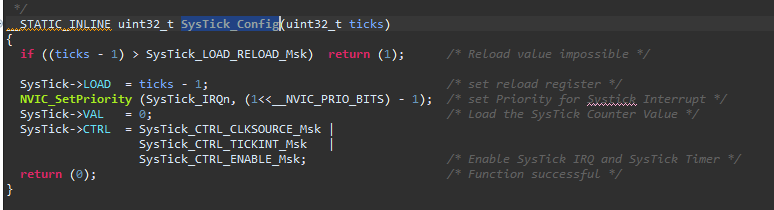


La función recibe un tipo de dato tick\_t, que en efecto es un entero de 64 bits, es decir, 2 ints (un long sin signo) y devuelve un booleano al salir de su ejecución.



Luego la función genera un booleano ret\_val en 1 (true), ret\_val funciona como variable para definir el tipo de retorno de tickInit(). Luego entra en un bloque condicional que verifica que el argumento de entrada no sea cero, pues si lo es se apagara el uso de tick (un timer) y se retorna 0 (false).

En el caso que el argumento sea valido se entrara en un seguno condicional. Este comprobara que dicho argumento este entre 1 y 50. De ser verdad se inicializara el sistemas de interrupciones por tiempo y se activara el timer correspondiente con la función SysTick\_Config(). De ser falso, se retornara 0 pues el valor del tiempo definido supera lo que se permite.

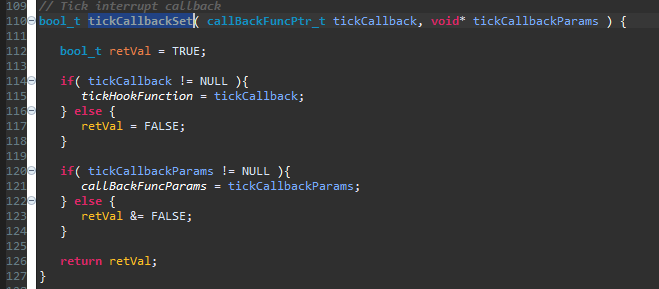


Por lo tanto la función tickInit(50) define una interrupción cada 50 milisegundos.

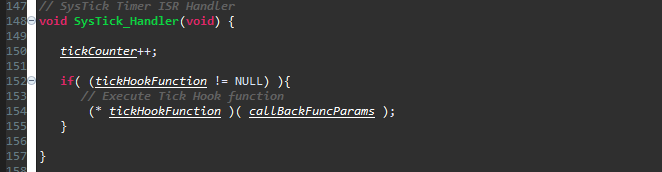
Luego se define la función tickCallbackSet(). Dicha función se encuentra en

/firmware\_v2/modules/lpc4337\_m4/sapi/src/sapi\_tick.c

Dicha función carga a dos variables globales tickHookFunction y callBackParams con los argumentos de entrada.



Cuando la interrupción se lanza pasados los 50 ms se va al vector de interrupciones y se busca la función de interrupción denominada SysTick\_Handler() que básicamente es la encargada de llamar a la función de Callback pasada como argumento a tickCallbackSet(), en este caso, myTickHook().



Por lo tanto mirando nuevamente el main() se observa que el código va cargando en el handler siempre la función de callback pero con distintos parámetros (en este caso los enum de todos los led), provocando que cada 50 ms se vayan apagando y prendiendo todos los leds.