8.

*Migrar el proyecto sapi\_examples/edu-ciaa-nxp/bare\_metal/gpio/gpio\_02\_blinky (parpadeo del LEDs c/sAPI) a: projects/TP1*

*a. Documentar los pasos a seguir para concretar una migración exitosa*

*b. Identificar funciones de librería sAPI útiles para el parpadeo de un led*

*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**

Para realizar el análisis de las funciones invocadas se hara desde un sentido *top–down,* es decir, desde mas alto nivel y luego analizando como son por dentro bajando cada vez mas de nivel.

**Main()**

La secuencia de funciones invocadas para el proyecto gpio\_blinky\_02 es la que se muestra en la fig. 1. La función main se encuentra en

/firmware\_v2/projects/Tp1/gpio\_02\_blinky/src/blinky.c

La misma consiste en una función que inicializa la placa, boardConfig() y luego posee en un loop infinito en el que se llama básicamente a dos funciones gpioWrite() y delay(). Dentro del loop las funciones gpioWrite() siempre utilizan el macro LEDB y van variando el estado del led en prendido y apagado. La función delay tiene *harcodeado* 500 que simboliza 500ms (medio segundo) en donde el sistema esperara dicho tiempo para continuar la ejecución. El código integral al ejecutarlo prendera y apagara un led cada medio segundo.

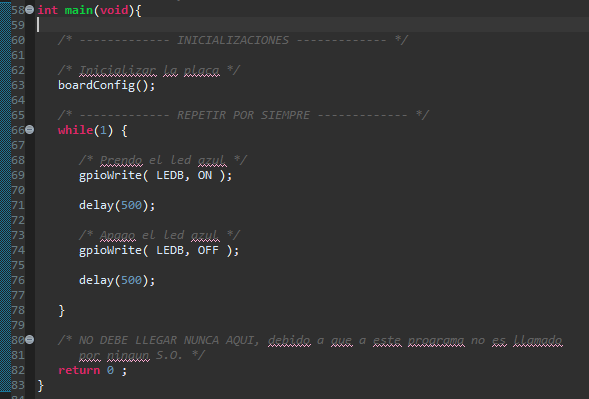


Fig. 1 - Estructura del Main()

1. **boardConfig()**

BoardConfig() es una función importante en cada proyecto de embebidos. Se encuentra declarada en

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

Como se observa en la fig. 2 posee funciones en donde configura el Clock interno del core SystemCoreClockUpdate() y cyclesCounterConfig() , tambien inicializa perifericos con Board\_init() y luego inicializa las entradas y salidas de propósito general (GPIO) utilizando la función gpioConfig(), la cual necesita de macros para definir *como* es configurado el pin.

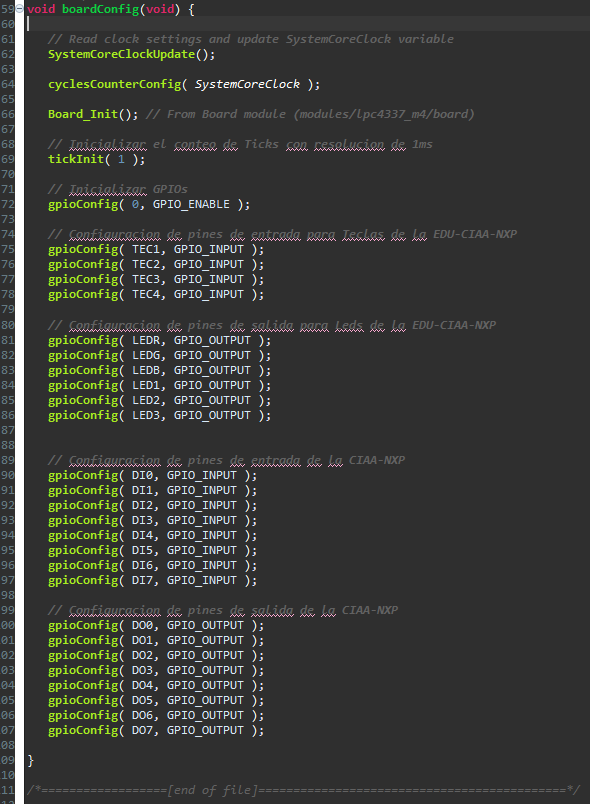
****

Fig. 2

* 1. **SystemCoreClockUpdate**

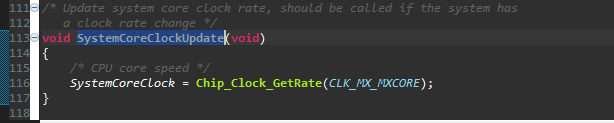
Esta función define el clock a bajo nivel y se llama antes de iniciar el sistema o en algún caso particular que se cambie el clock durante la ejecución. Dicha función se encuentra en

/firmware\_v2/modules/lpc4337\_m4/chip/src/chip\_18xx\_43xx.c

Posee como argumento de entrada un tipo enumerativo definido junto a otros en :

/firmware\_v2/modules/lpc4337\_m4/chip/inc/chip\_clocks.h

Dicho tipo enumerativo junto con la función Chip\_Clock\_GetRate() obtienen la frecuencia de clock del micro. A pesar que la salida de la función sea void, SystemCoreClock es una variable GLOBAL por lo que aquellas funciones que estén en su alcance podrán usarla.

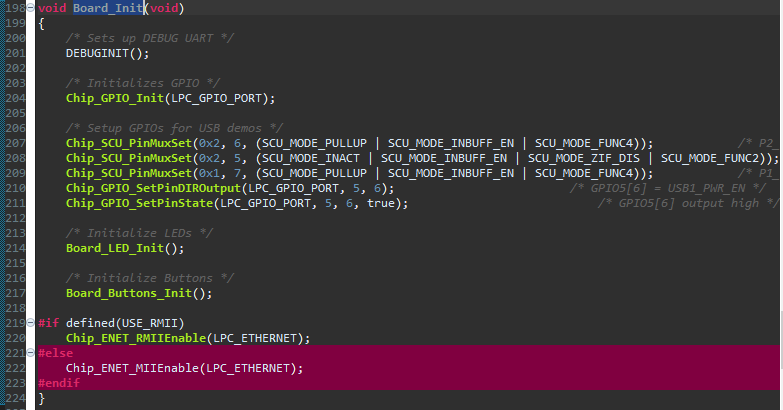


* 1. **Board\_Init()**

Board\_Init() es una función que agrupa la inicialización de distintos periféricos (Gpio como Leds y Buttons, USB, etc). Se encuentra en

/firmware\_v2/modules/lpc4337\_m4/board/src/board.c

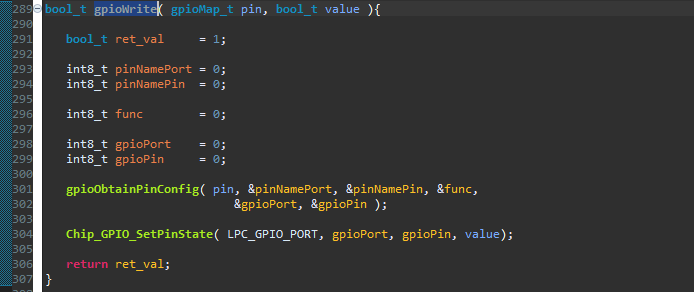
A priori cada pin del micro puede tener varios tipos de funcionamientos y es en esta función donde se dice cual de todas es la función del pin.



* 1. **gpioConfig()**

gpioConfig



1. **gpioWrite()**
2. **delay()**

