Taller de Sistemas Embebidos
STM32 MCU – Power Management

Información relevante

Taller de Sistemas Embebidos

Asignatura correspondiente a la **actualización 2023** del Plan de Estudios 2020 y resoluciones modificatorias, de Ingeniería Electrónica de FIUBA

Estructura Curricular de la Carrera

El **Proyecto Intermedio** se desarrolla en la asignatura **Taller de Sistemas Embebidos**, la cual tiene un enfoque centrado en la **práctica propia de la carrera** más que en el desarrollo teórico disciplinar, con eje en la **participación de las y los estudiantes**

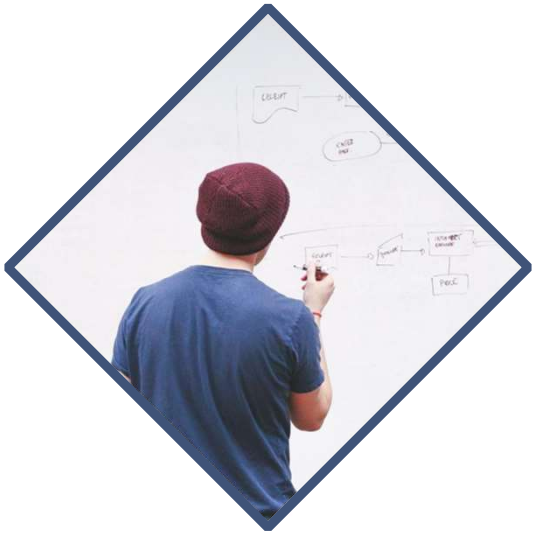
Más información . . .

. . . sobre la **actualización 2023** . . . https://www.fi.uba.ar/grado/carreras/ingenieria-electronica/plan-de-estudios

. . . sobre el **Taller de Sistemas Embebidos** . . . https://campusgrado.fi.uba.ar/course/view.php?id=1217

Por Ing. Juan Manuel Cruz, partiendo de la platilla Salerio de Slides Carnival
Este documento es de uso gratuito bajo Creative Commons Attribution license (https://creativecommons.org/licenses/by-sa/4.0/)

You can keep the Credits slide or mention SlidesCarnival (http://www.slidescarnival.com), Startup Stock Photos (https://startupstockphotos.com/), Ing. Juan Manuel Cruz and other resources used in a slide footer



¡Hola!

Soy Juan Manuel Cruz
Taller de Sistemas Embebidos
Consultas a: jcruz@fi.uba.ar

1

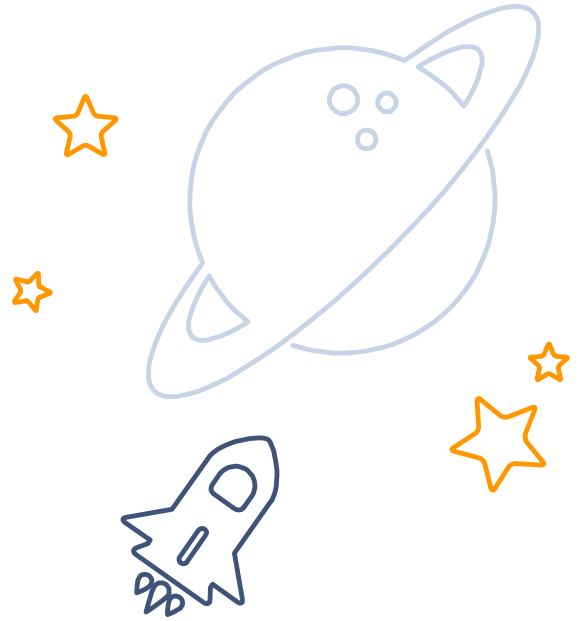
Introducción

Actualización 2023 del Plan de Estudios 2020 y resoluciones . . .

Conceptos básicos

- Referencia:
  - Nucleo Boards Programming with the STM32CubeIDE Hands-on in more than 50 projects - Dogan Ibrahim (Author)
  - Chapter 15: STM32L4 MCU Power Management
    - Este capítulo examina el concepto de Power Management y muestra cómo programarlo.
    - Los microcontroladores Arm implementan Power Management de forma diferente, según los distintos proveedores y pueden ser más complejos.
    - Este capítulo es simplemente una introducción.

Solución Adecuada



. . . lo más simple posible, previa determinación del objetivo de excelencia a cumplir, obviamente contando con la documentación debida y recurriendo a la metodología de trabajo adecuada

2

Documentación debida

1er Cuatrimestre de 2024, dictado por primera vez . . .

“

Nucleo Boards Programming with the STM32CubeIDE Hands-on in more than 50 projects - Dogan Ibrahim (Author)

Chapter 15: STM32L4 MCU Power Management

Overview

- La serie STM32L (L0, L1 y L4) ha sido desarrollada para aplicaciones de bajo consumo.
  - En este capítulo nos centraremos en las funciones de administración de energía de la serie STM32L4.
  - VDD es la fuente de alimentación principal de la MCU.
  - Cuando VDD está apagado, se puede usar una batería opcional conectada al pin VBAT para proporcionar voltaje de reserva.
  - La batería alimenta el módulo de reloj en tiempo real (RTC), el oscilador LSE y los pines de activación para despertar la MCU de los modos de suspensión profunda.
  - Cuando hay VDD presente, es posible cargar la batería externa a través de una resistencia interna.

Overview

- La MCU se coloca en modo Sleep mediante las instrucciones WFI (Wait for Interrupt) y WFE (Wait for Event).
  - Cuando está en modo Sleep, la CPU se reanudará mediante una solicitud de interrupción.
  - WFE es similar a WFI, pero verifica el estado de un registro de eventos y pone la CPU en modo Sleep si el evento aún no está configurado.
  - La WFE puede despertarse por eventos externos.

Low power modes

- Los modos de bajo consumo de la MCU STM32L4 se muestran en la Figura 15.1.

Figure 15.1: STM32L MCU low-power modes.

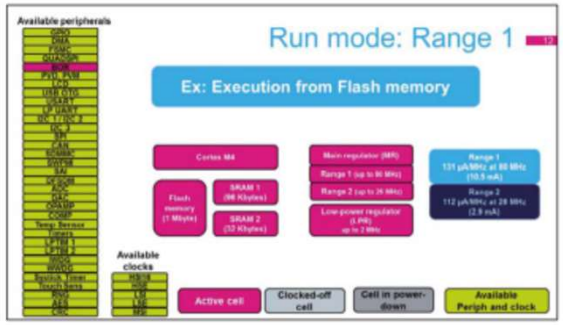


Figure 15.2: Run mode Range 1.

- Modo Run: hay dos modos de ejecución disponibles: Range 1 y Range 2.
- La Figura 15.2 muestra la descripción general del sistema en el Range 1. Este es el modo predeterminado después del encendido o reinicio, donde el reloj funciona a la velocidad máxima de 80 MHz, consumiendo 131 µA/MHz, se pueden activar todos los periféricos, se pueden habilitar todos los relojes, se habilita la memoria flash.

Low power modes

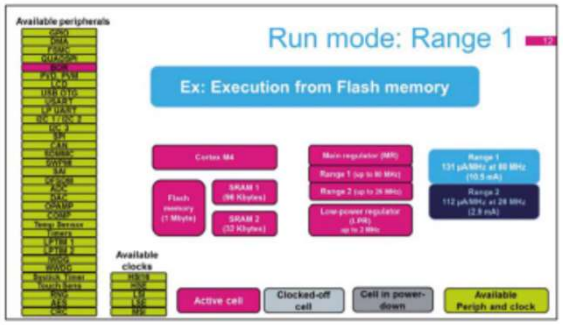


Figure 15.2: Run mode Range 1.

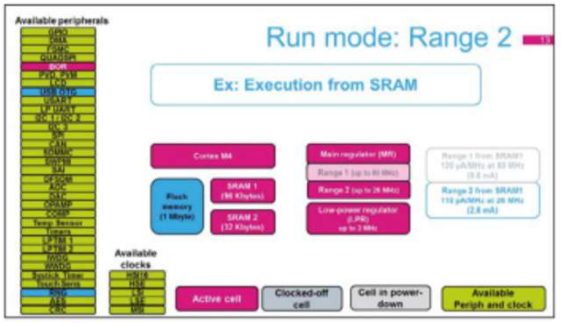


Figure 15.3: Run mode Range 2.

- El modo de ejecución Range 2 (Figura 15.3) es un rango de rendimiento medio con un reloj de hasta 26 MHz, que consume 110 µA/MHz con ejecución desde SRAM y se pueden habilitar todos los relojes.
- Al ejecutar desde SRAM, se ahorra el consumo de energía del Flash. Se pueden activar todos los periféricos excepto el USB OTG y el generador de números aleatorios.

Low power modes

- La memoria flash se puede desactivar en el modo de Run llamando a esta función:
  - `HAL_FLASHEx_EnableRunPowerDown()`
- Y se vuelve a habilitar llamando a esta función:
  - `HAL_FLASHEx_DisableRunPowerDown()`
  - Es un requisito que todas las rutinas se coloquen en SRAM cuando la memoria flash esté apagada; de lo contrario, se producirán fallas en el bus.
- Al Run Range 2 se ingresa llamando a esta función:
  - `HAL_PWREx_EnableVoltageScaling()`

Low power modes

- Modo Low-Power Run: en este modo (Figura 15.4) el regulador principal está APAGADO. El reloj del sistema es de hasta 2 MHz, se pueden habilitar todos los relojes.
  - La ejecución puede ser desde memoria Flash o SRAM. El consumo de Flash se puede reducir apagándolo y apagando su reloj.
  - Se pueden activar todos los periféricos excepto el USB OTG y el generador de números aleatorios.
  - El consumo de energía es de 135 µA/MHz cuando se ejecuta desde memoria Flash y de 112 µA/MHz cuando se opera desde SRAM.
  - A este modo se accede llamando a:
    - `HAL_PWREx_EnableLowPowerRunMode()`

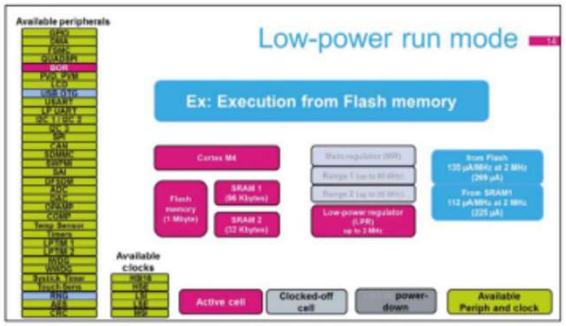


Figure 15.4: Low-power run mode.

Low power modes

- Modo Sleep: en este modo (Figura 15.5), el reloj de la CPU está apagado y el reloj del sistema puede alcanzar hasta 80 MHz.
  - Todos los periféricos se pueden activar.
  - Los relojes SRAM están habilitados, pero se pueden desactivar.
  - A este modo se accede llamando a:
    - `HAL_PWR_EnterSLEEPMode()`
  - La CPU se despierta de la condición de suspensión mediante una interrupción (WFI o WFE).

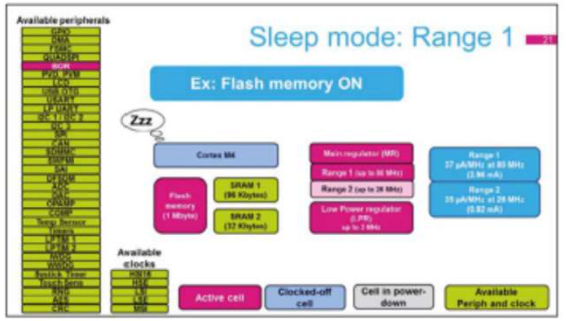


Figure 15.5: Sleep mode.

Low power modes

- Modo Low-Power Sleep: en este modo (Figura 15.6) el reloj de la CPU está APAGADO y el reloj del sistema se reduce a 2 MHz.
  - La memoria flash se puede desactivar.
  - Se pueden activar todos los periféricos excepto el USB OTG y el generador de números aleatorios.
  - El consumo de energía es de 40 µA/MHz a 2 MHz con Flash y SRAM APAGADOS.
  - A este modo se accede llamando a:
    - `HAL_PWR_EnterSLEEPMode()`

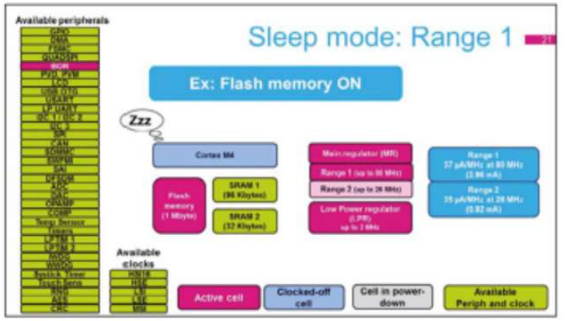


Figure 15.5: Sleep mode.

Low power modes

- Modos Stop 0, Stop 1 y Stop 2: son los modos de menor potencia.
  - Se conservan los contenidos de SRAM y todos los registros de periféricos.
  - Todos los relojes de alta velocidad están detenidos, pero se pueden habilitar los relojes externos de 32.768 kHz e internos de 32 kHz.
  - El reloj del sistema al despertar puede ser el reloj interno de alta velocidad de hasta 48 MHz con un tiempo de activación de 0,7 µs desde SRAM o 5 µs desde Flash.
  - El consumo de energía de Stop 2 es menor que el de Stop 1 y que el de Stop 0.

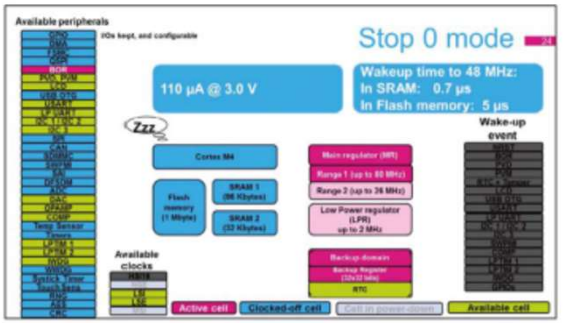


Figure 15.7: Stop mode 0.

Low power modes

- La CPU sale de la condición de Stop mediante cualquier interrupción EXTI o mediante la recepción de interrupciones de UART, I2C, etc.
  - La Figura 15.7 muestra el modo de Stop 0. En este modo, el reloj del sistema está congelado y la mayoría de los relojes periféricos están desactivados.
  - Varios periféricos pueden ser funcionales: detector de voltaje de alimentación, monitor de voltaje de periféricos, controlador LCD, convertidores de digital a analógico, amplificadores operacionales, comparadores, watchdog independiente, temporizadores de baja potencia, I2C, UART y UART de baja potencia.
  - Los eventos de todas las E/S pueden reactivarse desde el modo Stop 0, más la interrupción generada por los periféricos activos.

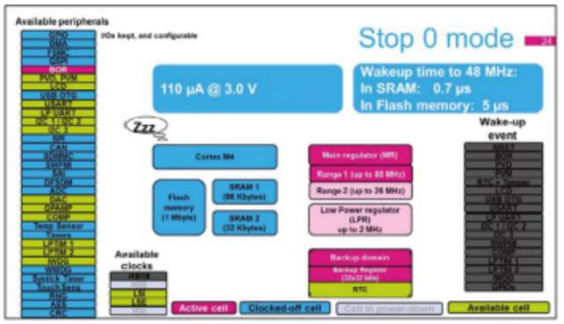


Figure 15.7: Stop mode 0.

Low power modes

- El modo Stop 1 es similar al modo Stop 0, excepto que el consumo de energía es menor. Se ingresa al modo Parada 1 llamando a la función:
  - `HAL_PWREx_EnterSTOP1Modo()`
- El modo Stop 2 es similar al modo Stop 1.
  - Varios periféricos pueden ser funcionales: detector de voltaje de alimentación, monitores de voltaje de periféricos, controlador LCD, comparadores, watchdog independiente, temporizador 1 de baja potencia, I2C3 y UART de baja potencia. Los eventos de todas las E/S pueden despertarse desde el modo Stop 2, más la interrupción generada por los periféricos activos.
  - Se ingresa al modo Parada 1 llamando a la función:
    - `HAL_PWREx_EnterSTOP2Mode()`

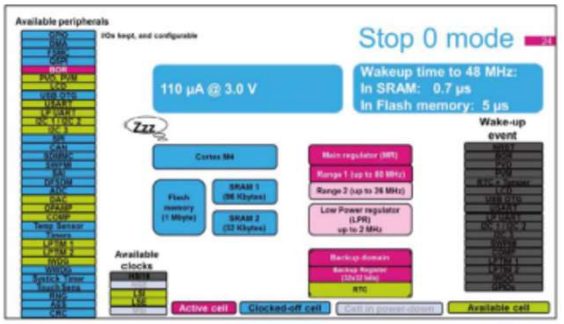


Figure 15.7: Stop mode 0.

Low power modes

- Modo Standby : este es el modo low-power (Figura 15.8).
  - Se pueden conservar los 32 Kbytes de SRAM2.
  - Los reguladores de voltaje están en modo de apagado, la SRAM y los registros de periféricos están perdidos.
  - Se conservan registros de respaldo (128 bytes).
  - La CPU sale de la condición de Standby mediante el flanco ascendente del pin WKUP, alarma RTC, Reset externo o Reset IWDOG.
  - El modo Standby se ingresa llamando a la función:
    - `HAL_PWR_EnterSTANDBYMode()`

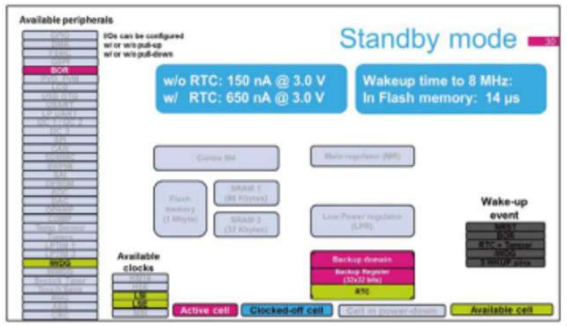


Figure 15.8: Standby mode.

Low power modes

- Modo Shutdown: este es el modo de menor potencia (Figura 15.9) con solo 30 nA a 1,8 V.
  - Este modo es similar al modo Standby, pero sin ningún control de energía: el restablecimiento de la caída de tensión está deshabilitado y el cambio a VBAT no se admite en el modo de apagado.
  - Los registros de respaldo (128 bytes) se conservan en el modo Shutdown.
- La CPU sale del modo de apagado mediante el flanco ascendente del pin WKUP, una alarma de RTC o un Reset externo.
  - El modo Shutdown se ingresa llamando a la función:
    - `HAL_PWREx_EnterSHUTDOWNMode()`

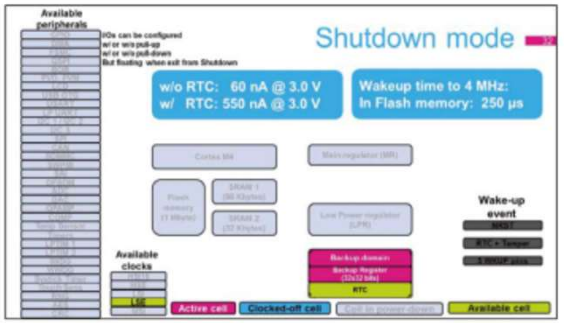


Figure 15.9: Shutdown mode.

Power modes transitions

- La Figura 15.10 muestra las posibles transiciones de un modo de energía a otro.
  - De la figura se desprende claramente que desde el modo Run podemos acceder a todos los modos de bajo consumo excepto al modo Low-Power Sleep.
  - Antes de pasar al modo Low-Power Sleep, primero debemos pasar al modo Low-power Run y ejecutar una instrucción WFI o WFE.
  - Al salir del modo Low-power Sleep, siempre volvemos al modo Low-Power Run.
  - Si el procesador está en los modos de Standby o Shutdown , siempre volverá al modo de Run.

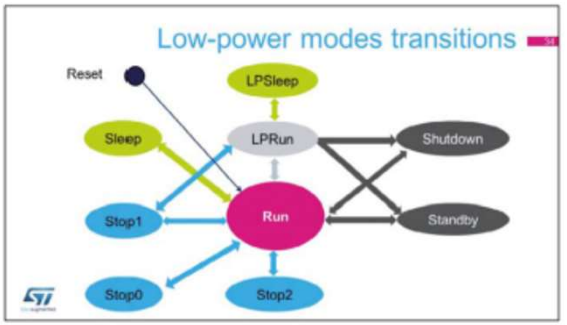


Figure 15.10: Power mode transitions.

Low power peripherals

- Low power peripherals
  - El procesador STM32L4 admite periféricos de bajo consumo, como UART de bajo consumo (LPUART) y temporizador de bajo consumo (LPTIM).

Debugging in low-power modes

- Debugging in low-power modes
  - Si la MCU se coloca en modo sleep, stop 0 standby durante la depuración, la conexión de depuración se pierde.
  - Sin embargo, podemos llamar a algunas funciones HAL para permitir la depuración en modos low-power.
  - Por ejemplo, la llamada a función `HAL_DBGMCU_EnableDBGSleepMode()` habilita la depuración en modo Sleep.
  - Además, `HAL_DBGMCU_EnableDBGStopMode()` habilita la depuración en modo Stop.
  - Debido a que la depuración utiliza los pines GPIO PA13, PA14 y PC3, tenemos que habilitar GPIOA y GPIOC durante la depuración.

Measuring Nucleo current consumption

- Measuring Nucleo current consumption
  - Las placas Nucleo incluyen cabezales de pines IDD que se pueden usar para medir el consumo de corriente de la MCU.
  - Se debe quitar el puente IDD y se debe conectar un amperímetro entre estos pines para medir el consumo de corriente de la MCU en cualquier momento.

Project 1: Sleep Mode Example

- Descripción: En este proyecto entraremos en el modo de Sleep.
- En este programa se utilizan el botón integrado (en PC13, normalmente ALTO) y el LED integrado (en PA5).
  - Recuerde que el estado del botón normalmente es lógico ALTO y pasa a lógico BAJO cuando se presiona el botón.
- En este proyecto, la MCU se activará cada vez que se detecte una interrupción externa presionando el botón. Luego, el estado del LED integrado se alternará dentro del ISR.
- El modo Sleep al salir se configurará para que el programa vuelva al modo Sleep después de salir del ISR. Como resultado, no se ejecutará ningún código fuera del ISR.
- Objetivo: El objetivo de este proyecto es mostrar cómo se puede poner la MCU en modo de Sleep y también cómo se puede salir de este modo.

Project 1: Sleep Mode Example

- Listado de programas: Entraremos en modo Sleep ejecutando instrucciones WFI o WFE.
- Tenga en cuenta que el uso de Wait For Interrupt (WFI) activará la MCU cuando se reconozca cualquier interrupción periférica. En este proyecto usaremos el WFI.
- La llamada a la función para habilitar WFI con el regulador principal habilitado es:
  - `HAL_PWR_EnterSLEEPMode(PWR_MAINREGULATOR_ON, PWR_SLEEPENTRY_WFI);`
- La MCU se activará cada vez que se active una interrupción externa (es decir, cuando se presione el botón) y el ISR tendrá la siguiente función de devolución de llamada. Dentro de la función callback alternaremos el LED.

```
void HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin)
{
      // Toggle the LED
}
```

Project 1: Sleep Mode Example

- Habilitaremos la función de Sleep-on-exit para que cuando ocurra la interrupción se procese dentro del ISR y la MCU vuelva al modo de Sleep cuando el ISR salga.
- La llamada a la función para habilitar Sleep-on-exit es:
  - `HAL_PWR_EnableSleepOnExit();`
- En algunas aplicaciones es posible que tengamos que suspender los ticks desactivando las interrupciones del sistema. Esto se hace llamando a la siguiente función:
  - `HAL_SuspendTick();`
- Para reanudar las interrupciones del sistema, utilice la siguiente función:
  - `HAL_ResumeTick();`

Project 1: Sleep Mode Example

- Los pasos para desarrollar el programa son los siguientes.
  - Inicie STM32CubeIDE.
  - Crear un nuevo espacio de trabajo.
  - Inicie un nuevo proyecto SMT32 y seleccione STM32L476RG como procesador.
  - Nombra el programa como SLEEP.
  - Configure PA5 como salida digital y PC13 como GPIO_EXTI13.
  - Haga clic derecho con el mouse y configure la etiqueta de PA5 como LED y la etiqueta de PC13 como Botón (Figura 15.11).

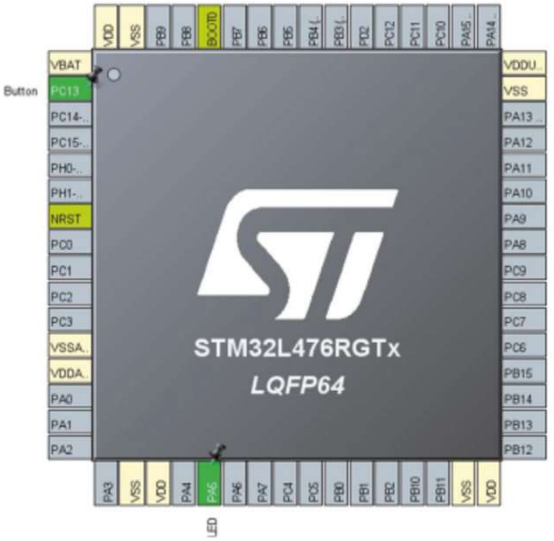


Figure 15.11: Configure PA5 and PC13.

Project 1: Sleep Mode Example

- Haga clic en System Core y seleccione GPIO. Haga clic en PC13 y configure el modo GPIO en Modo de interrupción externa con detección de disparo de flanco descendente.
- Haga clic en NVIC en la pestaña Núcleo del sistema. Haga clic para habilitar las interrupciones de la línea EXTI [15:10]
- Haga clic en Archivo, seguido de Guardar y haga clic en SÍ para generar código.
- Haga clic en Core, seguido de Src y haga doble clic en main.c para abrir el programa principal.

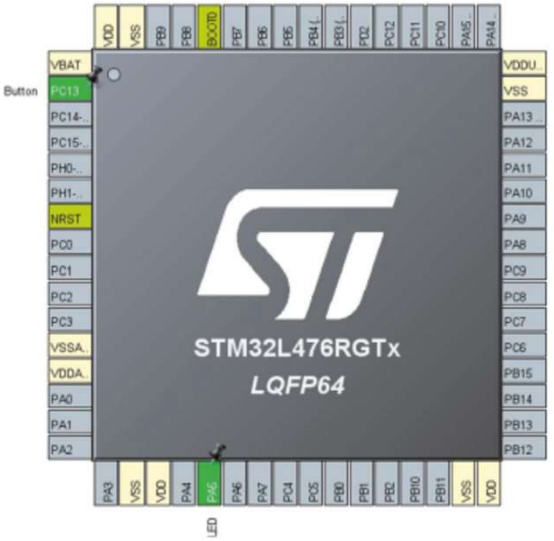


Figure 15.11: Configure PA5 and PC13.

Project 1: Sleep Mode Example

- El código ISR alterna el LED y su código es el siguiente:

```
void HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin)
{
    HAL_GPIO_TogglePin(GPIOA, LED_Pin);
}
```

- Dentro del programa principal, se habilita la Sleep-on-exit y luego la CPU se pone en modo Sleep llamando a las siguientes funciones:

```
HAL_PWR_EnableSleepOnExit();
HAL_PWR_EnterSLEEPMode(PWR_MAINREGULATOR_ON, PWR_SLEEPENTRY_WFI);
```

- Observe que el siguiente código se coloca dentro del loop del programa, pero estas declaraciones no se ejecutarán ya que la CPU vuelve a Sleep después de la ISR.

Project 1: Sleep Mode Example

```
//
// The following code will not be executed since we have enabled sleep-on-exit
//
while (1)
{
      HAL_GPIO_WritePin(GPIOA,LED_Pin, GPIO_PIN_SET);
      HAL_Delay(500);
      HAL_GPIO_WritePin(GPIOA,LED_Pin, GPIO_PIN_RESET);
      HAL_Delay(500);
}
```

- Si insertamos la declaración `HAL_PWR_DiableSleepOnExit()` dentro de la ISR, cuando se presiona el botón, el LED cambiará su estado y luego se ejecutarán las declaraciones dentro del loop del programa, haciendo que el LED parpadee cada 500 ms.

Project 1: Sleep Mode Example

- La Figura 15.12 muestra el listado de programas (Programa: SLEEP), donde el reloj, las E/S y las rutinas de error no se muestran para mayor claridad.

```
#include «main.h»

void SystemClock_Config(void);
static void MX_GPIO_Init(void);

//
// This is the interrupt service routine (ISR). The program jumps her when the
// Button is pressed. Here, we toggle the LED
//
void HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin)
{
  HAL_GPIO_TogglePin(GPIOA, LED_Pin);
}

//
// Start of main program
//
int main(void)
{
  HAL_Init();
  SystemClock_Config();
  MX_GPIO_Init();

  HAL_PWR_EnableSleepOnExit();
  HAL_PWR_EnterSLEEPMode(PWR_MAINREGULATOR_ON, PWR_SLEEPENTRY_WFI);


//
// The following code will not be executed since we have enabled sleep-on-exit
//
  while (1)
  {
    HAL_GPIO_WritePin(GPIOA,LED_Pin, GPIO_PIN_SET);
    HAL_Delay(500);
    HAL_GPIO_WritePin(GPIOA,LED_Pin, GPIO_PIN_RESET);
    HAL_Delay(500);
  }
}
```

Figure 15.12: The SLEEP Program.

Project 2: Stop Mode Example

- Descripción: Este proyecto es similar al anterior donde se utilizan el botón integrado y el LED integrado.
- Objetivo: El objetivo de este proyecto es mostrar cómo se puede utilizar el modo Stop de MCU en un ejemplo.
- Listado de programas: en el modo Stop, todos los relojes se detienen, los PLL, el HIS y el oscilador HSE RC están deshabilitados.
- Se conservan los registros internos de MCU y el contenido de SRAM. En este proyecto apagaremos el regulador principal y usaremos el regulador de low power.
- Podemos ingresar al modo Stop ejecutando WFI o WFE como en el modo de suspensión. Si se utiliza WFI para ingresar al modo Stop, una interrupción externa, un watchdog o RTC pueden activar el dispositivo. En este ejemplo, se utilizará la interrupción externa generada al presionar el botón integrado.

Project 2: Stop Mode Example

- La siguiente función se utilizará para ingresar al modo Stop:

```
HAL_PWR_EnterSTOPMode(PWR_LOWPOWERREGULATOR_ON, PWR_STOPENTRY_WFI);
```

- Debido a que todos los relojes se detienen en este modo, tenemos que reiniciar el reloj del sistema dentro del ISR.
- También puede ser necesario reanudar el sistema en algunas aplicaciones:

```
void HAL_GPIO_EXTI_Callback (uint16_t GPIO_Pin)
{
    if (GPIO_Pin == GPIO_PIN_13)
        {
            SystemClock_Config ();
        }
}
```

Project 2: Stop Mode Example

- Nuevamente, habilitaremos Sleep-on-exit al salir para que la CPU vuelva al modo Sleep después de salir del ISR.
- El programa de este proyecto es muy similar al que se presenta aquí y, como resultado, los pasos de desarrollo del programa no se repiten aquí.
- El listado de programas se muestra en la Figura 15.13. Nuevamente, sólo se muestra la lista del programa principal.
- Observe que el código de parpadeo del LED dentro del bucle del programa no se ejecuta ya que la suspensión al salir está habilitada. Si desactivamos la suspensión al salir dentro del ISR, veremos el LED parpadear cada 500 ms después de presionar el botón una vez.
- Esto requiere que el reloj de la CPU esté funcionando, como es el caso aquí ya que el reloj está configurado dentro del ISR.

Project 2: Stop Mode Ex

```
#include «main.h»

void SystemClock_Config(void);
static void MX_GPIO_Init(void);

//
// This is the interrupt service routine (ISR). The program jumps her when the
// Button is pressed. Here, we toggle the LED
//
void HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin)
{
  SystemClock_Config ();
  HAL_GPIO_TogglePin(GPIOA, LED_Pin);
}


//
// Start of main program
//
int main(void)
{
  HAL_Init();
  SystemClock_Config();
  MX_GPIO_Init();

  HAL_PWR_EnableSleepOnExit();
  HAL_PWR_EnterSTOPMode(PWR_LOWPOWERREGULATOR_ON, PWR_STOPENTRY_WFI);

//
// The following code will not be executed since we have enabled sleep-on-exit
//
  while (1)
  {
    HAL_GPIO_WritePin(GPIOA,LED_Pin, GPIO_PIN_SET);
    HAL_Delay(500);
    HAL_GPIO_WritePin(GPIOA,LED_Pin, GPIO_PIN_RESET);
    HAL_Delay(500);
  }
}
```

Figure 15.13: Program listing.

Project 3: Standby Mode Example

- Descripción: Este proyecto es similar al anterior. Aquí, el LED integrado se activa cuando la CPU sale del modo Standby. El wakeup se produce si el pin de wakeup (pin GPIO PA0) sube de BAJO a ALTO.
- Objetivo: El objetivo de este proyecto es mostrar cómo se puede utilizar el modo Standby de MCU en un ejemplo.
- Listado de programas: En este modo se consume muy poca energía. Los osciladores PLL, HIS y HSE están todos cortados.
- El modo Standby es prácticamente el mismo que el Reset del sistema, donde después de wakeup el programa se ejecuta desde el principio como si se hubiera producido un Reset. La única diferencia entre el Reset y Standby wakeup es que en la activación Standby wakeup se establece el flag de CPU PWR_FLAG_SB. Por lo tanto, al verificar el valor de este flag podemos determinar si se ha producido o no un Reset o un wakeup desde el modo Standby.

Project 3: Standby Mode Example

- La Standby wakeup se puede activar con el flanco ascendente del pin WKUP 1 (pin GPIO PA0 en el microcontrolador STM32L476RG), o puede activarse mediante el RTC o IWDG. En este programa usaremos el pin WKUP para activar la CPU.
- Antes de ingresar a este modo, se deben borrar PWR_FLAG y RTC_FLAG. Dado que usaremos PWR_FLAG, se debe llamar a la siguiente función HAL:

```
__HAL_PWR_CLEAR_FLAG(PWR_FLAG_WU);
```

- Para habilitar el pin de wakeup 1 (pin GPIO PA0), se debe llamar a la siguiente función HAL:

```
HAL_PWR_EnableWakeUpPin(PWR_WAKEUP_PIN1);
```

- Se ingresa al modo de espera llamando a la siguiente función HAL:

```
HAL_PWR_EnterSTANDBYMode();
```

Project 3: Standby Mode Example

- La siguiente declaración se puede utilizar para determinar si se ha producido un Reset o un wakeup desde el modo Standby. Las declaraciones dentro del bloque if se ejecutarán si se ha producido el wakeup desde el modo Standby:

```
If (__HAL_PWR_GET_FLAG(PWR_FLAG_SB)!= RESET)
{
}
```

- La Figura 15.14 muestra el listado de programas, donde solo se muestra el programa principal.
- Al comienzo del programa determinamos si se ha producido un Reset o un wakeup del modo Standby. Si se produjo un wakeup, entonces se borra PWR_FLAG_SB, el LED se alterna y el pin de wakeup se desactiva. El código fuera del bloque if se ejecuta cuando se produce un Reset y también después de que se ejecuta el bloque if. Aquí, se borra el flag de energía, se habilita el wakeup y la CPU se pone en modo Standby.

Project 3: Standby Mode Example

```
#include «main.h»

void SystemClock_Config(void);
static void MX_GPIO_Init(void);

int main(void)
{
  HAL_Init();
  SystemClock_Config();
  MX_GPIO_Init();

  if(__HAL_PWR_GET_FLAG(PWR_FLAG_SB) != RESET)          // If Wakeup from Standby
  {
        __HAL_PWR_CLEAR_FLAG(PWR_FLAG_SB);              // Clear PWR_FLAG_SB
    HAL_GPIO_TogglePin(GPIOA, LED_Pin);
    HAL_PWR_DisableWakeUpPin(PWR_WAKEUP_PIN1);          // Disable WAKEUP_PIN1(PA0)
  }

  __HAL_PWR_CLEAR_FLAG(PWR_FLAG_WU);                    // Clear PWR_FLAG
  HAL_PWR_EnableWakeUpPin(PWR_WAKEUP_PIN1);             // Enable WAKEUP_PIN1(PA0)
  HAL_PWR_EnterSTANDBYMode();

  while (1)
  {
  }
}
```

Figure 15.14: Program listing.

Summary

- En este capítulo hemos aprendido los distintos modos de energía de la MCU STM32L476RG.
- Se han desarrollado varios proyectos de ejemplo para mostrar cómo utilizar el modo de suspensión, el modo de parada y el modo de espera.
- En el próximo capítulo nos centraremos en las placas de expansión Nucleo y desarrollaremos proyectos utilizando varias placas de expansión.

Referencias

- Programming with STM32: Getting Started with the Nucleo Board and C/C++ 1st Edición - Donal Norris (Author)
- Nucleo Boards Programming with the STM32CubeIDE, Hands-on in more than 50 projects - Dogan Ibrahim (Author)
- STM32 Arm Programming for Embedded Systems, Using C Language with STM32 Nucleo - Muhammad Ali Mazidi (Author), Shujen Chen (Author), Eshragh Ghaemi (Author)

Manos a la obra con el . . .

. . . Proyecto Intermedio

. . . un enfoque centrado en la **práctica propia de la carrera** más que en el desarrollo teórico disciplinar, con eje en la **participación de las y los estudiantes**


Las y los estudiantes preguntarán:
¿en qué lío nos metimos?

¡Muchas gracias!

¿Preguntas?

. . .

Consultas a: jcruz@fi.uba.ar