Análisis de Programa STM32 con FreeRTOS

1. HAL_Init()

```
/* Reset of all peripherals, Initialize
HAL_Init();

/* USER CODE BEGIN Init */

/* USER CODE END Init */

/* Configure the system clock */
```

¿Qué es?

Función de inicialización de la capa de abstracción de hardware (HAL - Hardware Abstraction Layer).

¿Para qué sirve?

Inicializa todos los periféricos, la interfaz Flash y configura el SysTick como base de tiempo del sistema. Esta función debe ser llamada al inicio del programa antes de configurar otros componentes del sistema.

Parámetros

Esta función no recibe parámetros.

Configuración Interna

- Reset de todos los periféricos del microcontrolador
- Inicialización de la interfaz Flash
- Configuración del temporizador SysTick para generar interrupciones cada 1ms (base de tiempo del sistema)

2. SystemClock_Config()

```
/* Configure the system clock */
SystemClock_Config();

/* USER CODE BEGIN SysInit */
```

```
RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_HSI;
RCC_OscInitStruct.HSIState = RCC_HSI_ON;
RCC_OscInitStruct.HSICalibrationValue = RCC_HSICALIBRATION_DEFAULT;
RCC_OscInitStruct.PLL.PLLState = RCC_PLL_NONE;
if (HAL_RCC_OscConfig(&RCC_OscInitStruct) != HAL_OK)
{
  Error_Handler();
}

/** Initializes the CPU, AHB and APB buses clocks
*/
RCC_ClkInitStruct.ClockType = RCC_CLOCKTYPE_HCLK|RCC_CLOCKTYPE_SYSCLK
                              |RCC_CLOCKTYPE_PCLK1|RCC_CLOCKTYPE_PCLK2;
RCC_ClkInitStruct.SYSCLKSource = RCC_SYSCLKSOURCE_HSI;
RCC_ClkInitStruct.AHBCLKDivider = RCC_SYSCLK_DIV1;
RCC_ClkInitStruct.APB1CLKDivider = RCC_HCLK_DIV1;
RCC_ClkInitStruct.APB2CLKDivider = RCC_HCLK_DIV1;
```

¿Qué es?

Función de configuración del reloj del sistema (System Clock Configuration).

¿Para qué sirve?

Configura todos los relojes del microcontrolador, incluyendo osciladores, PLL (Phase-Locked Loop) y buses de comunicación. Esta configuración determina la velocidad de operación del procesador y periféricos.

Parámetros Configurados

Configuración del Oscilador (RCC_OscInitStruct)

Parámetro	Valor
OscillatorType	RCC_OSCILLATORTYPE_HSI (Oscilador interno)

HSIState	RCC_HSI_ON (HSI activado)
HSICalibrationValue	RCC_HSICALIBRATION_DEFAULT
PLL.PLLState	RCC_PLL_NONE (PLL desactivado)

Configuración de Relojes de Buses (RCC_ClkInitStruct)

Parámetro	Valor
ClockType	HCLK | SYSCLK | PCLK1 | PCLK2
SYSCLKSource	RCC_SYSCLKSOURCE_HSI (16 MHz)
AHBCLKDivider	RCC_SYSCLK_DIV1 (sin división)
APB1CLKDivider	RCC_HCLK_DIV1 (sin división)
APB2CLKDivider	RCC_HCLK_DIV1 (sin división)
Flash Latency	FLASH_LATENCY_0 (0 wait states)

.

3. MX_GPIO_Init()

```
/* Initialize all configured periph
MX_GPIO_Init();
/* USER CODE BEGIN 2 */

/* USER CODE END 2 */
```

```
 /*Configure GPIO pin : LED_GREEN_Pin */
 GPIO_InitStruct.Pin = LED_GREEN_Pin;
 GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
 GPIO_InitStruct.Pull = GPIO_NOPULL;
 GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
 HAL_GPIO_Init(LED_GREEN_GPIO_Port, &GPIO_InitStruct);
```

¿Qué es?

Función de inicialización de pines GPIO (General Purpose Input/Output).

¿Para qué sirve?

Configura los pines de entrada/salida digital del microcontrolador. En este caso, configura un LED verde como salida digital.

Parámetros Configurados

Configuración del pin LED_GREEN:

Parámetro	Valor/Descripción
Pin	LED_GREEN_Pin (pin específico)
Mode	GPIO_MODE_OUTPUT_PP (salida push-pull)
Pull	GPIO_NOPULL (sin resistencias pull-up/down)
Speed	GPIO_SPEED_FREQ_LOW (velocidad baja)
Estado Inicial	GPIO_PIN_RESET (LED apagado)

4. osKernelInitialize()

```
/* Init scheduler */
osKernelInitialize();

/* USER CODE BEGIN RTOS_MUTEX */
/* add mutexes, ... */
/* USER CODE END RTOS_MUTEX */
```

¿Qué es?

Función de inicialización del kernel RTOS (CMSIS-RTOS v2 / FreeRTOS).

¿Para qué sirve?

Prepara el sistema operativo en tiempo real antes de crear tareas. Inicializa todas las estructuras internas necesarias para la gestión de tareas, colas, semáforos y otros mecanismos de sincronización.

Parámetros

Esta función no recibe parámetros.

Funcionalidad

- Inicializa las estructuras internas de FreeRTOS
- Prepara el sistema de gestión de tareas
- Configura los mecanismos de sincronización (colas, semáforos, mutexes)
- Debe llamarse antes de crear cualquier tarea con osThreadNew()

5. osThreadNew()

```
/* Create the thread(s) */
/* creation of blink01 */
blink01Handle = osThreadNew(StartBlink01, NULL, &blink01_attributes);

/* creation of blink02 */
blink02Handle = osThreadNew(StartBlink02, NULL, &blink02_attributes);
```

```
/* Definitions for blink01 */
osThreadId_t blink01Handle;
const osThreadAttr_t blink01_attributes = {
  .name = "blink01",
  .stack_size = 128 * 4,
  .priority = (osPriority_t) osPriorityNormal,
};
/* Definitions for blink02 */
osThreadId_t blink02Handle;
const osThreadAttr_t blink02_attributes = {
  .name = "blink02",
  .stack_size = 128 * 4,
  .priority = (osPriority_t) osPriorityBelowNormal,
};
```

¿Qué es?

Función para crear hilos o tareas en el sistema operativo en tiempo real.

¿Para qué sirve?

Crea una nueva tarea que se ejecutará de forma concurrente con otras tareas. Cada tarea tiene su propia pila de ejecución, prioridad y función asociada.

Parámetro	Descripción
func	Puntero a la función que ejecuta el hilo
argument	Argumento pasado a la función (NULL si no se usa)
attr	Puntero a estructura con atributos del hilo

Thread 1: blink01

Atributo	Valor
name	"blink01"
stack_size	512 bytes (128 * 4)
priority	osPriorityNormal (prioridad normal)

Thread 2: blink02

Atributo	Valor
name	"blink02"
stack_size	512 bytes (128 * 4)
priority	osPriorityBelowNormal (prioridad baja)

La tarea blink01 tiene mayor prioridad que blink02, por lo que el scheduler le dará preferencia cuando ambas estén listas para ejecutarse.

6. osKernelStart()

```
/* Init scheduler */
osKernelInitialize();

/* USER CODE BEGIN RTOS_MUTEX */
/* add mutexes, ... */
/* USER CODE END RTOS_MUTEX */
```

¿Qué es?

Función que inicia el scheduler (planificador) del RTOS.

¿Para qué sirve?

Arranca el planificador de tareas y comienza la ejecución de multitarea. Una vez llamada, el control del programa pasa completamente al sistema operativo.

Parámetros

Esta función no recibe parámetros.

Comportamiento

- Inicia el scheduler de FreeRTOS
- Comienza a ejecutar las tareas creadas según sus prioridades
- Esta función NUNCA retorna en operación normal
- El código después de osKernelStart() no se ejecutará

7. HAL_RCC_OscConfig()

```
RCC_OscInitTypeDef RCC_OscInitStruct = {0};
RCC_ClkInitTypeDef RCC_ClkInitStruct = {0};

/** Configure the main internal regulator output voltage
*/
__HAL_RCC_PWR_CLK_ENABLE();
__HAL_PWR_VOLTAGESCALING_CONFIG(PWR_REGULATOR_VOLTAGE_SCALE3);
```

¿Qué es?

Función de configuración de osciladores del módulo RCC (Reset and Clock Control).

¿Para qué sirve?

Configura las fuentes de reloj del microcontrolador, incluyendo osciladores internos (HSI), externos (HSE), y el PLL (Phase-Locked Loop) para multiplicación de frecuencia.

Parámetros Configurados

Campo	Valor/Descripción
OscillatorType	RCC_OSCILLATORTYPE_HSI - Selecciona oscilador interno
HSIState	RCC_HSI_ON - Activa el oscilador HSI (16 MHz)
HSICalibrationValue	RCC_HSICALIBRATION_DEFAULT - Valor de calibración por defecto
PLL.PLLState	RCC_PLL_NONE - PLL desactivado (no se usa multiplicación)

Tipos de Osciladores Disponibles

- HSI (High Speed Internal): Oscilador interno de 16 MHz
- HSE (High Speed External): Oscilador externo (cristal)
- LSI (Low Speed Internal): Oscilador interno de baja velocidad
- LSE (Low Speed External): Oscilador externo de baja velocidad (32.768 kHz)

8. HAL_RCC_ClockConfig()

```
/** Initializes the CPU, AHB and APB buses clocks
*/
RCC_ClkInitStruct.ClockType = RCC_CLOCKTYPE_HCLK|RCC_CLOCKTYPE_SYSCLK
                            |RCC_CLOCKTYPE_PCLK1|RCC_CLOCKTYPE_PCLK2;
RCC_ClkInitStruct.SYSCLKSource = RCC_SYSCLKSOURCE_HSI;
RCC_ClkInitStruct.AHBCLKDivider = RCC_SYSCLK_DIV1;
RCC_ClkInitStruct.APB1CLKDivider = RCC_HCLK_DIV1;
RCC_ClkInitStruct.APB2CLKDivider = RCC_HCLK_DIV1;
```

¿Qué es?

Función de configuración de los relojes de los buses del sistema.

¿Para qué sirve?

Establece divisores y fuentes de reloj para los diferentes buses del microcontrolador (CPU, AHB, APB1, APB2). También configura la latencia de la memoria Flash según la velocidad del sistema.

Parámetros Configurados

Campo	Valor/Descripción
ClockType	HCLK | SYSCLK | PCLK1 | PCLK2 - Todos los relojes
SYSCLKSource	RCC_SYSCLKSOURCE_HSI - Reloj del sistema desde HSI
AHBCLKDivider	RCC_SYSCLK_DIV1 - Bus AHB sin división (16 MHz)
APB1CLKDivider	RCC_HCLK_DIV1 - Bus APB1 sin división (16 MHz)
APB2CLKDivider	RCC_HCLK_DIV1 - Bus APB2 sin división (16 MHz)

Parámetro adicional:

FLatency	FLASH_LATENCY_0 - 0 wait states para memoria Flash

Jerarquía de Relojes

- SYSCLK: Reloj principal del sistema (16 MHz desde HSI)
- HCLK (AHB): Reloj del bus AHB para CPU, memoria y DMA
- PCLK1 (APB1): Reloj del bus de periféricos de baja velocidad
- PCLK2 (APB2): Reloj del bus de periféricos de alta velocidad

9. HAL_GPIO_WritePin()

```
/*Configure GPIO pin Output Level */
HAL_GPIO_WritePin(LED_GREEN_GPIO_Port, LED_GREEN_Pin, GPIO_PIN_RESET);

/*Configure GPIO pin : LED_GREEN_Pin */
```

¿Qué es?

Función para escribir un valor digital en un pin GPIO.

¿Para qué sirve?

Establece el nivel lógico (HIGH o LOW) de un pin digital configurado como salida. Se utiliza típicamente para controlar LEDs, activar relés, enviar señales digitales, etc.

Parámetros

Parámetro	Descripción
GPIOx	Puerto GPIO (ej: GPIOA, GPIOB, GPIOC, etc.)
GPIO_Pin	Pin específico (ej: GPIO_PIN_0 hasta GPIO_PIN_15)
PinState	Estado del pin: GPIO_PIN_SET (HIGH) o GPIO_PIN_RESET (LOW)

Estados Posibles

Estado	Descripción
GPIO_PIN_RESET	Nivel lógico bajo (0V) - LED apagado
GPIO_PIN_SET	Nivel lógico alto (3.3V) - LED encendido