**MEMORIA FLASH**

La memoria flash es un periférico silencioso que usamos sin preocuparnos demasiado por él. Una vez que estamos seguros de que la flash tiene suficiente espacio para almacenar el firmware, cargamos la imagen binaria utilizando el depurador o una herramienta de flash dedicada. Y lo olvidamos por completo.

Sin embargo, la flash interna provista por todos los microcontroladores STM32 funciona de la misma manera que otros periféricos. Puede programarse directamente desde el firmware mediante la configuración de registros específicos, y esto nos permite actualizar el firmware utilizando el mismo código a bordo o almacenar datos de configuración relevantes sin usar hardware externo dedicado (una EEPROM I²C externa o una flash SPI).

Este capítulo muestra cómo programar la memoria flash interna de STM32 utilizando el módulo HAL\_FLASH dedicado de CubeHAL. Describe cómo se organiza generalmente la flash en un microcontrolador STM32 típico, ilustrando brevemente las diferencias entre cada familia y los pasos involucrados para programar áreas específicas de esta memoria directamente desde el mismo microcontrolador.

Finalmente, se describe el papel del Acelerador ARTTM, junto con las evoluciones de esta tecnología propietaria de ST en los microcontroladores STM32F7.

**INTRODUCCIÓN A LA MEMORIA FLASH EN LA STM32**

diferencia de otras arquitecturas embebidas¹, todos los microcontroladores STM32 proporcionan una memoria flash dedicada para almacenar el código del programa y datos constantes. Actualmente existen once tamaños de memoria, que van desde 16KB hasta 2MB. El último dígito del número de parte de un determinado MCU STM32 define el tamaño de la memoria flash, como se muestra en la Tabla 21.1. Por ejemplo, un MCU STM32F401RE tiene 512KB de memoria flash.

Tabla

Descripción generada automáticamente

Tabla

Descripción generada automáticamente

Nuestra placa tiene 2 Mbytes de memoria Flash

Dependiendo de la familia STM32, el tipo de venta y el paquete utilizado, la memoria flash de un MCU STM32 se puede organizar en:

• **Uno o dos bancos**: la mayoría de los microcontroladores STM32 proporcionan solo un banco de memoria flash, mientras que los más potentes tienen hasta dos bancos. La arquitectura de múltiples bancos permite operaciones duales y simultáneas: mientras se programa o borra en un banco, las operaciones de lectura son posibles en el otro. Este enfoque proporciona mayor flexibilidad para operaciones duales, especialmente para aplicaciones de alto rendimiento. En algunos MCUs STM32 más recientes, el múltiple banco es una característica programable que se puede habilitar opcionalmente, y los tamaños de los bancos se pueden configurar según sea necesario.

• **Cada banco se divide a su vez en sectores**: cada banco de memoria flash se divide en varios subbloques, llamados sectores. Algunos MCUs STM32 proporcionan memoria flash con todos los sectores del mismo tamaño (generalmente igual a 1KB o 2KB). Algunos otros proporcionan varios sectores con diferentes tamaños (generalmente los primeros sectores tienen un tamaño más pequeño que los restantes).

• **Cada sector se puede dividir en páginas**: en algunos MCUs STM32, un sector se divide aún más en varias páginas más pequeñas. A veces, esto ocurre solo para el primer sector, lo que permite borrar y luego programar solo una fracción del sector. La Tabla 21.2² muestra cómo se organiza la memoria flash en algunos microcontroladores STM32F0. Como se puede ver, pueden proporcionar hasta diecisiete sectores, cada uno a su vez dividido en cuatro páginas. Además, se asigna una área dedicada, llamada Bloque de información, a otro rango de direcciones: esta memoria no volátil se utiliza para almacenar registros de configuración especiales (llamados bytes de opción) y algunos programadores de arranque preprogramados de fábrica, que estudiaremos en el próximo capítulo. En MCUs STM32 más potentes, la región de Bloque de información también contiene la memoria programable solo una vez (OTP) (que puede variar de 512 a 1024 bytes): esta es una memoria no volátil que se puede usar para almacenar parámetros de configuración relevantes del dispositivo. ¿Por qué tener tal organización de memoria? Antes de poder responder a esta pregunta, necesitamos presentar algunos conceptos fundamentales sobre tecnologías de memoria flash. Sin detallarlo demasiado, hay dos familias principales de memorias flash: NAND y NOR. Las memorias flash NAND ofrecen una arquitectura física más compacta, lo que permite almacenar más celdas de memoria en la misma área de silicio. Las memorias NAND están disponibles en densidades de almacenamiento mayores y a costos más bajos por bit que la memoria NOR-flash (recuerde que, aparte de los costos de investigación y desarrollo, el costo de producción de un CI es todo acerca del tamaño de la matriz). Las memorias NAND también tienen hasta diez veces la resistencia de la memoria NOR-flash. NAND es más adecuada como medio de almacenamiento para archivos grandes, incluidos video y audio. Las unidades flash USB, las tarjetas SD y las tarjetas MMC son de tipo NAND.

Tabla

Descripción generada automáticamente

La NAND-flash no proporciona un bus de dirección externo de acceso aleatorio, por lo que los datos deben leerse por bloques, donde cada bloque contiene cientos a miles de bits, lo que se asemeja a un tipo de acceso de datos secuencial. Esto hace que la tecnología NAND-flash no sea adecuada para microcontroladores integrados, porque la mayoría de los microprocesadores y microcontroladores requieren acceso aleatorio a nivel de byte.

Una cosa importante a tener en cuenta sobre las tecnologías de memoria flash es que una operación de escritura en cualquier tipo de dispositivo flash solo se puede realizar en una unidad vacía o borrada. Entonces, en la mayoría de los casos, una operación de escritura debe ir precedida de una operación de borrado. Si bien la operación de borrado es sencilla en el caso de los dispositivos NAND-flash, en NOR-flash es obligatorio que todos los bytes en el bloque objetivo se escriban con todos los ceros antes de que puedan borrarse. Por el contrario, las memorias NOR-flash ofrecen buses de dirección y datos completos para acceder aleatoriamente a cualquier ubicación de memoria (accesible a cada byte). Esto las hace adecuadas para almacenar código y datos constantes porque rara vez necesitan actualizarse. La resistencia de las memorias NOR es de 10,000 a 100,000 ciclos de borrado. Las memorias NOR-flash son más lentas en operaciones de borrado y escritura en comparación con las NAND-flash. Eso significa que las NAND-flash tienen tiempos de borrado y escritura más rápidos. Además, NAND tiene unidades de borrado más pequeñas. Entonces, se necesitan menos borrados, lo que las hace más adecuadas para almacenar sistemas de archivos. Las memorias NOR-flash pueden leer datos ligeramente más rápido que las NAND.

Los dispositivos NOR-flash se dividen en unidades de borrado, también llamadas bloques, páginas o sectores. Esta división es necesaria para reducir los precios y superar las limitaciones físicas. Es posible escribir información en un bloque específico solo si ese bloque está vacío/borrado, como se dijo antes. En la mayoría de las memorias NOR-flash, después de un ciclo de borrado, una célula individual contiene el valor "1", y una operación de escritura permite cambiar su valor a "0". Esto significa que una ubicación de memoria de palabra se establece en 0xFFFF FFFF después de un borrado. Sin embargo, existen algunas memorias NOR-flash donde el valor predeterminado de la célula después de un borrado es "0", y podemos establecerlo en "1" con una operación de escritura.

La partición de la memoria flash en varios bloques nos da una ventaja indirecta: podemos borrar y luego reprogramar solo pequeñas fracciones de la memoria flash. Esto es especialmente útil cuando usamos la memoria flash para almacenar parámetros de configuración no volátiles, sin usar memorias EEPROM dedicadas y externas³.

**Para evitar completamente las escrituras no deseadas en la memoria no volátil (NVM),** la memoria flash en todos los STM32 MCUs está protegida contra escritura, y existe una secuencia de desbloqueo específica que debe seguirse para desactivarla: se proporcionan dos registros de clave dedicados en la región de Opciones de Bytes, que permiten desactivar la protección de escritura flash emitiendo.

**MODULO HAL\_FLASH :**

Al igual que todos los demás periféricos STM32, incluso la memoria flash proporciona varios registros utilizados para manipular su configuración, como se mencionó antes. El módulo HAL\_FLASH, junto con el módulo relacionado HAL\_FLASHEx, permite borrar y reprogramar fácilmente la memoria NVM sin lidiar demasiado con los detalles de su implementación. Los siguientes subpárrafos presentan las funciones más relevantes de esos módulos.

**DESBLOQUEO DE LA MEMORIA FLASH:**

La memoria flash está protegida contra escritura por defecto, para evitar escrituras accidentales causadas por perturbaciones eléctricas o fallos en el programa. Para habilitar el modo de escritura se debe realizar una secuencia de operaciones, y esto es específico de la familia STM32 en cuestión. Para llevar a cabo esta tarea, CubeHAL proporciona la función:



que nos permite ignorar completamente la arquitectura específica de la memoria flash. Una vez que la protección de escritura/borrado de la memoria flash está deshabilitada, podemos realizar una operación de borrado o escritura. El procedimiento inverso al desbloqueo se realiza utilizando la función:



La protección de escritura se establece automáticamente al reiniciar el sistema. Sin embargo, se sugiere encarecidamente volver a bloquear explícitamente la memoria cuando se hayan completado todas las operaciones de escritura. Esto evita cualquier escritura accidental causada por un mal funcionamiento del firmware o una inestabilidad de energía.

**BORRADO DE LA MEMORIA FLASH:**

Antes de poder cambiar el contenido de una ubicación de memoria flash, debemos restablecer sus bits al valor predeterminado ("0" o "1" según el tipo de NOR-flash). Esto se realiza mediante una operación de borrado en la granularidad de sector/página. Alternativamente, se puede realizar un borrado masivo de todo el banco: esto significa que en aquellos microcontroladores STM32 que proporcionan dos bancos, se puede borrar cada banco a la vez. En la mayoría de los microcontroladores STM32, las celdas individuales de un bloque de memoria flash (sector o página) se establecen en "1" después de una operación de borrado, con solo dos excepciones notablemente: los microcontroladores STM32L0 y STM32L1, cuyo valor predeterminado es "0". El CubeHAL proporciona dos formas de realizar una operación de borrado de flash: borrado de flash en modo de encuesta y en modo de interrupción. La función:



Permite realizar un borrado de flash en modo de encuesta (polling mode). Acepta un puntero a una instancia de la estructura FLASH\_EraseInitTypeDef, que veremos en un momento, y un puntero a la variable (SectorError) que devuelve la identificación de los sectores/páginas defectuosos en caso de error durante el procedimiento de borrado (por ejemplo, si el procedimiento de borrado falla en la cuarta página, el parámetro SectorError contendrá el valor 3). La estructura **FLASH\_EraseInitTypeDef difiere mucho entre cada familia STM32. Por esta razón, echa un vistazo al archivo stm32XXxx\_hal\_flash\_ex.h del CubeHAL para tu MCU**. Aquí, vamos a considerar la implementación encontrada en CubeHALs para los MCU STM32 más potentes como los F2/F4/F7.

Texto, Carta

Descripción generada automáticamente

• **TypeErase**: especifica si estamos realizando un borrado de todo el banco de memoria o un borrado de sector/página. Puede asumir los valores FLASH\_TYPEERASE\_SECTORS o FLASH\_TYPEERASE\_MASSERASE.

• **Banks**: este parámetro, que solo está disponible en aquellas series de STM32 que proporcionan una memoria flash interna de varios bancos, especifica el banco involucrado en un borrado masivo. Puede asumir los valores FLASH\_BANK\_1, FLASH\_BANK\_2 o FLASH\_BANK\_BOTH para eliminar ambos bancos.

• Sector (Página): este campo se refiere al identificador del sector involucrado en un borrado basado en sectores. Puede asumir el valor FLASH\_SECTOR\_0, FLASH\_SECTOR\_1 y así sucesivamente (el número máximo de sectores depende del microcontrolador específico). En aquellos STM32 que proporcionan una memoria flash con granularidad de página, este campo se reemplaza por la primera dirección de la página involucrada en un procedimiento de borrado. Consulte el código fuente de CubeHAL para obtener más información al respecto.

• **NbSectors** (NbPages): el número de sectores (páginas) que se borrarán a partir del sector especificado.

• **VoltageRange**: aunque estemos borrando un sector completo (o página), el procedimiento de borrado se realiza en un subconjunto del mismo (generalmente dos bytes). Los microcontroladores STM32 más avanzados permiten borrar varios bytes a la vez. Esta característica se llama paralelismo de flash y está relacionada con el voltaje de funcionamiento del MCU: cuanto mayor sea VDD, más bytes se borran a la vez⁵. Este campo puede asumir un valor de la Tabla 21.3. Sin embargo, siempre consulte el manual de referencia de su MCU para obtener más información al respecto.

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

La función HAL\_FLASHEx\_Erase() es una función de bloqueo: esperará hasta que se haya completado el procedimiento de borrado. Esto puede ser un procedimiento bastante "largo", dependiendo de la familia de STM32, la velocidad de HCLK, el número de sectores/páginas involucradas en el borrado y el voltaje VDD en aquellos STM32 MCUs que proporcionan paralelismo de programación/borrado. Para evitar bloquear las actividades del firmware durante este procedimiento, el HAL proporciona la función:



que realiza un procedimiento de borrado en modo de interrupción. Podemos ser notificados del final del procedimiento de borrado habilitando la interrupción FLASH\_IRQn e implementando el ISR correspondiente.

**Se debe tener especial cuidado en caso de que estemos borrando una ubicación de memoria flash que contiene código de programa, especialmente si estamos eliminando el primer sector/página que contiene la tabla de vectores (esto siempre es cierto si estamos realizando un borrado en masa). Si este es el caso, entonces necesitamos mover el código del programa y reubicar toda la tabla de vectores dentro de la SRAM, como se muestra en el Capítulo 20, de lo contrario ocurrirá una falla una vez que se active la interrupción.**

**PROGRAMACIÓN DE LA MEMORIA FLASH:**

Una vez que un sector/página es borrado, podemos proceder a programar su contenido. En teoría, es perfectamente posible acceder directamente a una ubicación de memoria flash para cambiar su contenido escribiendo un código C como el siguiente:

Imagen que contiene Texto

Descripción generada automáticamente

Sin embargo, esto no es práctico por dos razones principales. En primer lugar, en algunos STM32 MCUs pueden requerirse operaciones preliminares (como establecer registros específicos) antes de que podamos programar una ubicación en flash. En segundo lugar, dependiendo de la serie STM32 específica y el rango de voltaje VDD, la cantidad de bytes que se pueden transferir simultáneamente a la memoria flash puede diferir significativamente. Por estas razones, HAL define la función:



La función HAL\_FLASH\_Program() está diseñada para abstraer todos los detalles de implementación específicos. Analicemos los argumentos de la función:

• **TypeProgram**: indica cuántos bytes se transfieren durante la operación de escritura, y puede tomar los valores FLASH\_TYPEPROGRAM\_HALFWORD, FLASH\_TYPEPROGRAM\_WORD y FLASH\_TYPEPROGRAM\_DOUBLEWORD. Tenga en cuenta que este parámetro especifica solo la cantidad de datos transferidos usando la función HAL\_FLASH\_Program(). El número efectivo de bytes transferidos en una sola transacción depende de la familia STM32 y del grado de paralelismo, si está disponible. • **Address**: es la dirección de memoria inicial donde comenzar a colocar el contenido.

• **Data**: es el dato a almacenar dentro de la ubicación de memoria flash (**representado como una variable de doble palabra**). Al igual que con el procedimiento de borrado visto anteriormente, es posible realizar un procedimiento de programación de flash en modo de interrupción utilizando la función:



**Acceso de lectura durante la programación y el borrado**

Un acceso de lectura a la memoria flash mientras se está realizando una operación de borrado o escritura provocará una parada en el bus, al menos en la mayoría de los microcontroladores STM32⁷. Esto significa que si necesita llevar a cabo otras operaciones en paralelo, es necesario reubicar en la SRAM el código que se ejecutará durante una operación de programación en flash. Un escenario típico es el de un cargador de arranque personalizado: podemos programar nuestro código para que intercambie el nuevo firmware en la flash utilizando la UART en modo de interrupción o DMA. Si este es el caso, no podemos perder eventos asíncronos (por ejemplo, una interrupción que nos notifica una transferencia de datos) porque el MCU se detiene esperando la operación en curso. En ese caso, es mejor reubicar el código en la SRAM (y, eventualmente, también reubicar la tabla de vectores).

**OPCION BYTES:**

Los bytes de opción son dos o más bytes cuyos bits son valores de configuración especial. El concepto de bytes de opción es similar al que se encuentra en otras arquitecturas de microcontroladores, como los fusibles en la serie AVR de Atmel o los bits de configuración que se encuentran en los microcontroladores PIC de Microchip. Cada bit individual de estos bytes especiales en la región del Bloque de Información tiene un significado especial. El número y tipo de parámetros de configuración dependen del MCU STM32 específico. Los parámetros de configuración más comunes están relacionados con:

• **BOOT**: en la mayoría de los microcontroladores STM32, dos bits de opción permiten seleccionar el origen de arranque (FLASH, memoria del sistema o SRAM).

• **RDP**: estos bits establecen el nivel de protección de lectura de la memoria flash, y los analizaremos con más detalle más adelante en este capítulo.

• **BOR\_LEVEL:** estos bits contienen el umbral del nivel de suministro que activa/libera el reinicio. Se pueden escribir para programar un nuevo nivel de BOR. De forma predeterminada, BOR está desactivado. Cuando la tensión de suministro (VDD) cae por debajo del nivel de BOR seleccionado, se genera un reinicio del dispositivo.

• **Comportamiento del MCU al entrar en algunos modos de bajo consumo de energía:** en casi todos los microcontroladores STM32, es posible configurar el MCU para que genere un reinicio al entrar en los modos de bajo consumo de energía stop o sleep.

• **Watchdog de hardware:** en algunos MCUs STM32, existen uno o dos bits utilizados para configurar el WWDG y el IWDG en "modo hardware", es decir, se inician automáticamente al reiniciar el MCU.

• **Protección de escritura de flash**: estos bits permiten proteger individualmente algunos sectores/páginas de flash, evitando escribir en ellos incluso si la memoria flash está desbloqueada. Si un bit determinado está establecido en '1', el sector/página correspondiente no está protegido contra escritura; si, en cambio, el bit está establecido en '0', entonces el sector/página está protegido contra escritura.

Para programar los bytes de opción hay un procedimiento específico a seguir, que es independiente de la programación de toda la memoria flash. Por lo tanto, CubeHAL proporciona rutinas dedicadas para usar. En primer lugar, esta región debe desbloquearse llamando a la función:



La siguiente opción, un byte se programa completamente usando la función:



El valor de un byte de opción se modifica automáticamente mediante la eliminación previa del bloque de información y luego programando todos los bytes de opción con los valores pasados a la rutina HAL\_FLASHEx\_OBProgram(). La función acepta una instancia de la estructura C FLASH\_OBProgramInitTypeDef, cuyos campos representan el contenido del byte de opción dado. Para obtener más información sobre el tipo y número exacto de campos, consulte el código fuente de CubeHAL.

De manera similar, para recuperar el contenido de un byte de opción dado, utilizamos la función:



Una vez que se modifica un byte de opción, debemos forzar al MCU a recargar su contenido usando la función:



Tenga en cuenta que cambiar algunos bits de opción en algunos STM32 MCUs puede provocar un reinicio del chip. Por último, el ST STM32CubeProgrammer proporciona la capacidad de modificar fácilmente los bytes de opción. Una vez que haya conectado el depurador ST-LINK al MCU de destino, haga clic en el icono de bytes de opción (el tercer icono verde a la izquierda). Aparece la sección de bytes de opción, como se muestra en la Figura 21.1. La misma herramienta STM32CubeProgrammer también permite borrar sectores/páginas de flash seleccionados.

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

**PROTECCIÓN DE MEMORIA DE LA MEMORIA FLASH:**

Un byte de opción (llamado RDP) merece un párrafo separado: el byte de configuración relacionado con la protección de lectura de la memoria flash. Para evitar el acceso no deseado a la memoria flash a través de la interfaz de depuración, es posible desactivar temporal o permanentemente el acceso de lectura a esta memoria desde el mundo exterior (claro está, el acceso desde el núcleo de la CPU y los controladores DMA siempre es posible). Existen tres niveles de protección, que corresponden a tres valores diferentes que se deben almacenar en el byte de opción:

• **Nivel 0** (sin protección de lectura): cuando se establece el nivel de protección de lectura en Nivel 0 escribiendo 0xAA en el byte de opción de protección de lectura (RDP), todas las operaciones de lectura/escritura (si no se establece protección de escritura) desde/hacia la memoria flash o la SRAM de respaldo son posibles en todas las configuraciones de arranque (arranque de usuario de flash, depuración o arranque desde RAM).

• **Nivel 1** (protección de lectura habilitada): es el nivel de protección de lectura predeterminado después del borrado de los bytes de opción (que se realiza automáticamente mediante la rutina HAL\_FLASHEx\_OBProgram()). La protección de lectura Nivel 1 se activa escribiendo cualquier valor (excepto 0xAA y 0xCC utilizados para establecer Nivel 0 y Nivel 2, respectivamente) en el byte de opción RDP. Cuando se establece el Nivel 1 de protección de lectura, no se puede realizar ningún acceso (lectura, borrado, programación) a la memoria flash o la SRAM de respaldo mientras el depurador esté conectado o mientras se arranca desde la RAM o el cargador de arranque de la memoria del sistema. Se genera un error de bus en caso de solicitud de lectura. En cambio, cuando se arranca desde la memoria flash, se permiten los accesos (lectura, borrado, programación) a la memoria flash y la SRAM de respaldo desde el código de usuario. Cuando se activa el Nivel 1, la programación del byte de opción de protección (RDP) al Nivel 0 hace que se borren en masa la memoria flash y la SRAM de respaldo. Como resultado, se borra el área de código de usuario antes de que se elimine la protección de lectura. El borrado en masa solo borra el área de código de usuario. Los otros bytes de opción, incluidas las protecciones de escritura, permanecen sin cambios antes de la operación de borrado en masa. El área OTP no se ve afectada por el borrado en masa y permanece sin cambios. El borrado en masa solo se realiza cuando el Nivel 1 está activo y se solicita el Nivel 0. Cuando se aumenta el nivel de protección (0->1, 1->2, 0->2), no hay borrado en masa.

• **Nivel 2 (!!!protección de lectura de depuración/CHIP desactivada permanentemente!!!**): la protección de lectura Nivel 2 se activa escribiendo 0xCC en el byte de opción RDP. Cuando se establece la protección de lectura Nivel 2:

* Todas las protecciones proporcionadas por el Nivel 1 están activas.
* Ya no se permite el arranque desde RAM.
* Es posible el arranque del bootloader de la memoria del sistema y todos los comandos no están disponibles excepto Get, GetID y GetVersion. Consulte AN2606.
* JTAG, SWV (visor de un solo cable), ETM y la prueba de límites están deshabilitados.
* Los bytes de opción del usuario ya no se pueden cambiar.
* Cuando se arranca desde la memoria flash, se permiten accesos (lectura, borrado y programación) a la memoria flash y SRAM de respaldo desde el código de usuario.

Una captura de pantalla de un celular

Descripción generada automáticamente con confianza media

**EN ESTE DOCUMENTO RECOGE TODA LA INFORMACION DE LA MEMORIA FLASH HASTA EL PUNTO 21.4 (OPTIONAL OTP AND TRUE-EEPROM MEMORIES)!!**