2. Relojes.

El procesador es un sistema digital secuencial, es decir, que necesita de un reloj para su operación. El microcontrolador STM32F103C8T6 tiene varias fuentes de reloj internas y dispone de pines para colocar osciladores de cristal de cuarzo o introducir frecuencias externas como fuentes de reloj. También tiene un reloj interno y externo de tiempo real, RTC, como referencia temporizada en segundos.

2.1. Relojes para los periféricos y ejecución de instrucciones.

El microprocesador del STM32F103C8T6 usa un reloj llamado SYSCLK como base para operaciones con sus periféricos. La frecuencia máxima de operación de este reloj es de 72 Mhz. El reloj SYSCLK usa cualquiera de las tres fuentes de reloj: HSI, HSE o la salida de un PLL, como muestra la figura 28. El primero, HSI (High Speed Internal) es un reloj interno de alta velocidad de 8 Mhz. HSE es un reloj externo de alta velocidad, que en las tarjetas negra y azul es cristal de cuarzo de 8 Mhz. El PLL (Phase- Locked Loop) es un multiplicador de frecuencias que tiene como entradas el reloj interno HSI de 8 Mhz dividido en dos, o sea 4 Mhz, el reloj HSE externo de 8 Mhz o este mismo dividido en dos. La multiplicación de frecuencia del PLL puede ser hasta por 16, pero la frecuencia máxima de operación del procesador es de 72 Mhz. Esto quiere decir que si se escoge el reloj de alta velocidad externo de 8 Mhz, el PLL puede multiplicar esta frecuencia hasta por 9.

El cristal de cuarzo que se conecta entre los pines 5 y 6 del integrado, el cual representa el reloj HSE, puede tener una frecuencia entre 4 y 16 Mhz. La diferencia entre el reloj HSI interno de 8 Mhz y el HSE externo de 8 Mhz es la exactitud en la frecuencia; el externo es más estable que el interno, cuya frecuencia se basa en un cristal de cuarzo, mientras que el oscilador interno es un circuito con resistencia y condensador, RC, que puede variar con la temperatura.

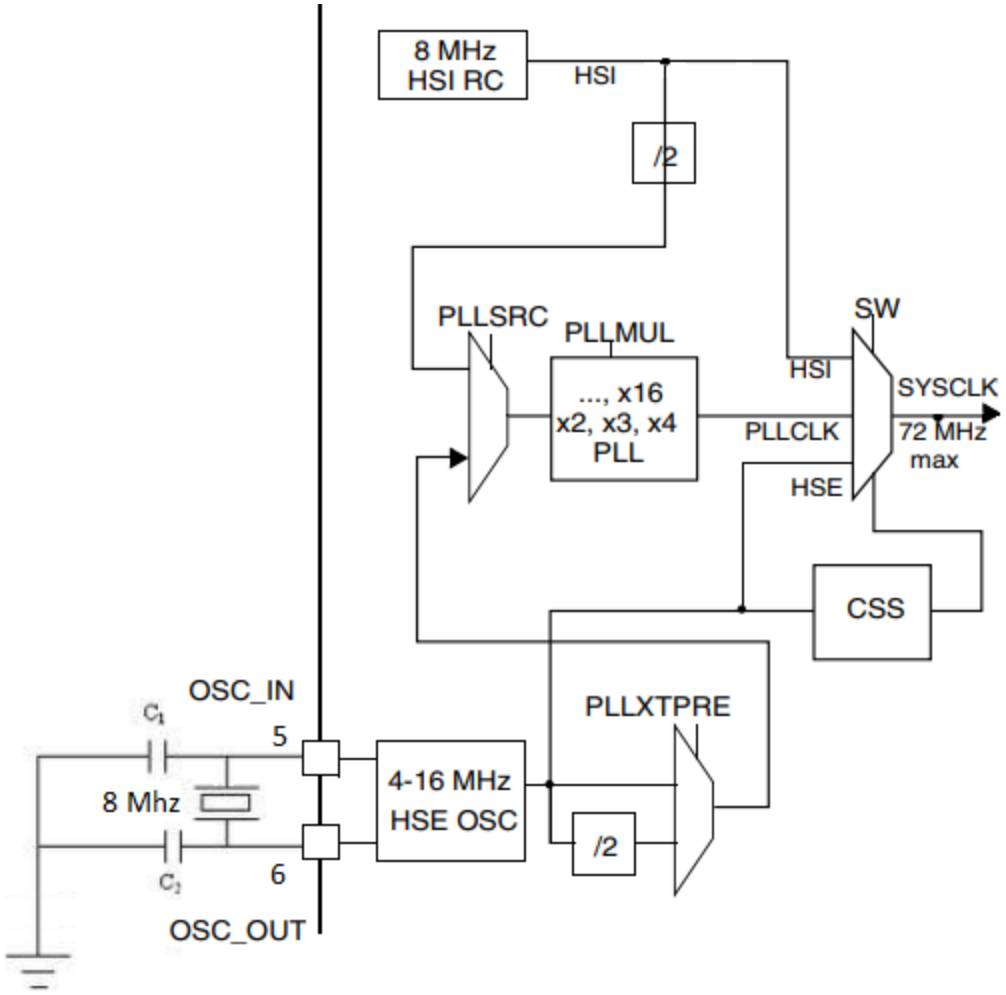

Figura 28. Relojes fuente para la generación del reloj Sysclk.

La programación para escoger la frecuencia en Sysclk se hace con la configuración de los registros o usando las funciones disponibles en la librería SPL. Como se observa en la figura, el Sysclk puede venir del reloj HSI, de la salida del PLL o del reloj externo HSE. De donde proviene, se puede especificar con la función:

```
RCC_SYSCLKConfig(RCC_SYSCLKSource);
```

Donde `RCC_SYSCLKSource` puede ser `RCC_SYSCLKSource_HSI`, para escoger el reloj interno de 8 MHz, `RCC_SYSCLKSource_HSE`, para escoger el reloj externo, que en el caso de las tarjetas negra y azul es de 8 Mhz, o `RCC_SYSCLKSource_PLLCLK`, para escoger la salida de PLL. Cuando se va a cambiar el factor del multiplicador de frecuencia PLL, primero este no debe estar siendo usado por el sistema. Un ejemplo de uso del PLL con una frecuencia en Sysclk de 12 Mhz se presenta a continuación:

```
1. RCC_SYSCLKConfig(RCC_SYSCLKSource_HSI); // Asegura PLL no esté en uso
2. RCC_PLLCmd(DISABLE);// para cambiar multiplicador, debe deshabilitarse el PLL
```

```
3.  RCC_PLLConfig(RCC_PLLSource_HSE_Div2, RCC_PLLMul_3);// 4Mz x 3= 12Mhz
4.  RCC_PLLCmd(ENABLE); // habilita el PLL
5.  while((RCC->CR & RCC_CR_PLLRDY) == 0); // espera que se estabilice.
6.  RCC_SYSCLKConfig(RCC_SYSCLKSource_PLLCLK); // se escoge PLL como Sysclk
```

En la primera línea se asegura que el PLL no sea el Sysclk si escogemos que Sysclk sea el reloj interno HSI. En la segunda línea se deshabilita el PLL para poder cambiarle el factor de multiplicación. En la tercera línea la función tiene como parámetros la fuente de reloj que entra al PLL, que puede ser `RCC_PLLSource_HSI_Div2` (reloj interno HSI dividido por 2), `RCC_PLLSource_HSE_Div2` (reloj externo HSE dividido por 2) o `RCC_PLLSource_HSE_Div1` (reloj externo HSE sin división), y el factor de multiplicación, `RCC_PLLMul_2, RCC_PLLMul_3,` hasta `RCC_PLLMul_16`. En la cuarta línea del código se habilita el PLL, pero es necesario dar una espera de estabilización del reloj, por lo que la quinta línea da esa espera observando el bit correspondiente. En la última línea se deja el PLL como fuente del Sysclk. De esta forma el reloj del sistema, Sysclk, se ha configurado de tal manera que tiene como entrada el reloj externo de 8 Mhz entrando a un divisor por 2 y luego un multiplicador por 3, Sysclk= (8 Mhz / 2) * 3= 12 Mhz.

Existe un pin llamado MCO (Microcontroller Clock Output), ubicado en el pin 8 del puerto GPIOA que se conecta con cualquiera de las fuentes de reloj, como la salida del PLL dividida en dos, el reloj externo HSE, el reloj interno HSI o el reloj del sistema Sysclk. Esto se muestra en la figura 29.

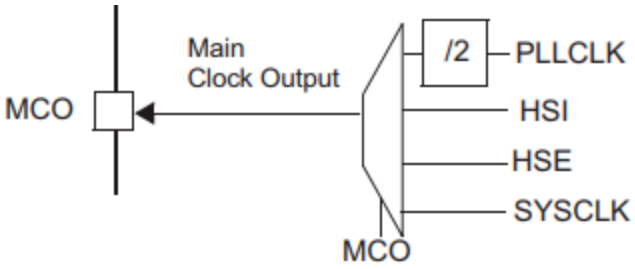

Figura 29. Selección de reloj que va al pin de salida MCO (A8).

Para habilitar la salida de cualquiera de estos relojes al pin MCO, se necesita hacer la configuración del pin en modo alterno y seleccionar el reloj que va a ir conectado al pin MCO. El código que hace esto se muestra a continuación:

```
1.  RCC_MCOConfig(RCC_MCO_PLLCLK_Div2); // sale el reloj del PLL dividido en dos
2.  GPIO_InitTypeDef GPIOA_Struct; // define una estructura
3.  GPIOA_Struct.GPIO_Pin  = GPIO_Pin_8; // inicializa en la estructura, pin A8
4.  GPIOA_Struct.GPIO_Speed = GPIO_Speed_50MHz; // máxima velocidad
5.  GPIOA_Struct.GPIO_Mode = GPIO_Mode_AF_PP; // pin con funciones alternas.
6.  RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOA, ENABLE);
7.  GPIO_Init(GPIOA, &GPIOA_Struct);
```

La primera línea del código dice que se conecte al pin MCO la salida del PLL dividida en dos. Este parámetro puede ser `RCC_MCO_NoClock` (sin reloj), `RCC_MCO_SYSCLK` (el reloj del sistema), `RCC_MCO_HSI` (el reloj interno HSI), `RCC_MCO_HSE` (el reloj externo HSE), o `RCC_MCO_PLLCLK_Div2` (la salida del PLL dividida en dos). Las líneas siguientes a esta son para configurar el puerto A, pin 8, como salida de Push- Pull, modo de funciones alternas y una frecuencia de reloj máxima de 50 Mhz.

Desde el Sysclk como frecuencia base, los relojes para los periféricos se pueden configurar independientemente por medio de la frecuencia del bus donde están unidos, como lo muestra la figura 30.

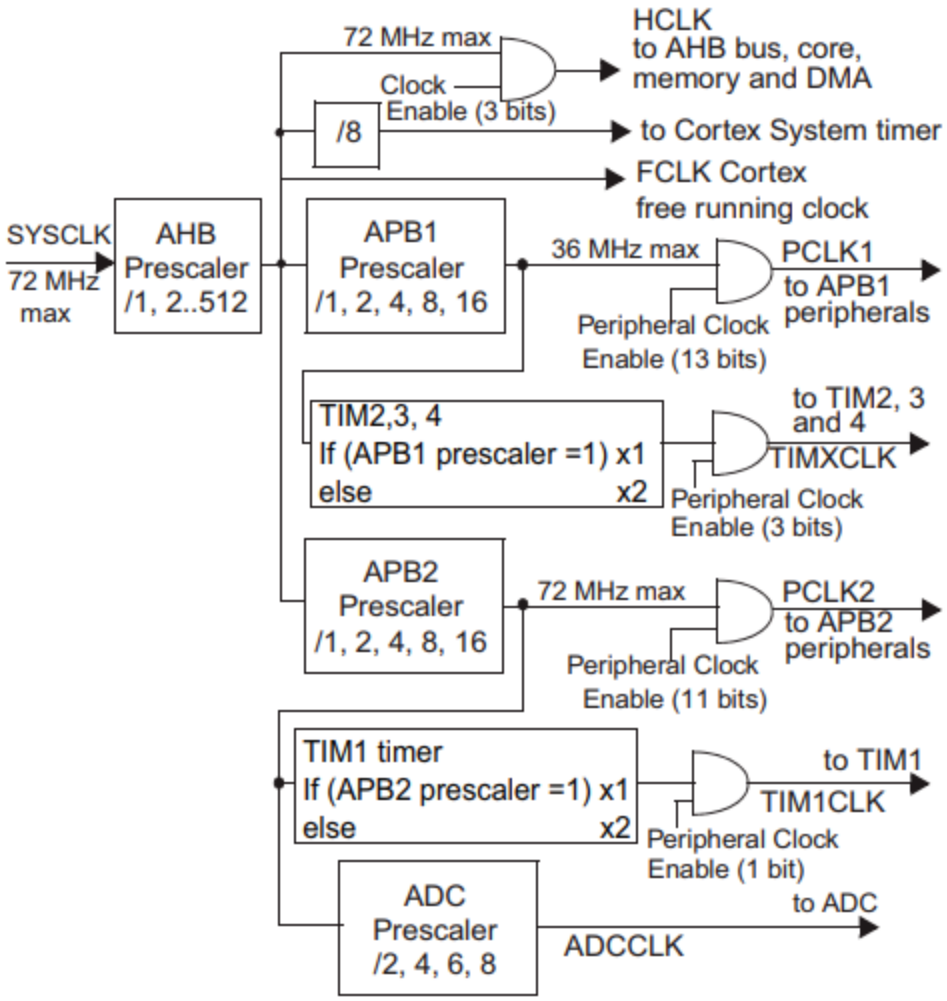

Figura 30. Bloques de configuración de la frecuencia de los buses para periféricos.

Con fuente el reloj Sysclk, el reloj FCLK depende de un bloque divisor AHB que utiliza la siguiente función para configurar el factor de división:

```
RCC_HCLKConfig(RCC_SYSCLK_DivX)
```

Donde X puede ser 1, 2, 4, 8, 16, 64, 128, 256, o 512 e indican el factor de división de la frecuencia en el reloj Sysclk.
Igualmente para conseguir las frecuencias de los relojes PCLK1 y PCLK2 se usan las siguientes funciones respectivamente para determinar los factores de división en cada caso:

```
RCC_PCLK1Config(RCC_HCLK_DivX)
RCC_PCLK2Config(RCC_HCLK_DivX)
```

Donde X puede ser 1, 2, 4, 8 o 16, como factores de división en cada bloque. Cada uno de estos relojes alimentan con frecuencia los módulos periféricos, como los temporizadores, ADCs, líneas de entrada y salida, y periféricos de comunicaciones, DMA y memorias.

Taller 2. Cambios de tiempo con las fuentes de reloj.

Usar el código en lenguaje ensamblador siguiente.

```
int main(void)
{
    int conteo= 10000;
    //configuración del reloj del puerto
    asm("MOVW r0, #0x1000");
    asm("MOVT r0, #0x4002"); //dirección base RCC= 0x4002.1000
    asm("MOVW r2, #0x18"); //offset de RCC_APB2ENR
    asm("LDR r1, [r0, r2]"); //lee su contenido
    asm("ORR r1, r1, #0x08"); //bandera de habilitación del puerto B
    asm("STR r1, [r0, r2]");

    //configuración de los bits como salida
    asm("MOVW r0, #0x0C00"); // base puerto B
    asm("MOVT r0, #0x4001"); //dirección base 0x4001.0C00
    asm("MOVW r2, #0x04"); //offset del registro GPIOB_CRH
    asm("MOVW r1, #0x0000");
    asm("MOVT r1, #0x0001");
    asm("STR r1, [r0, r2]");

    // encender led
    asm("MOVW r2, #0x10"); //offset de este registro
    asm("MOVT r1, #0x1000");
    asm("STR r1, [r0, r2]");

  while(1)
  {
      for (int i = 0; i < conteo; ++i) asm("nop");
      // apagar led
      asm("MOVW r0, #0x0C00"); // base puerto B
      asm("MOVT r0, #0x4001");
      asm("MOVW r2, #0x10");
      asm("MOVT r1, #0x0000");
```

```
      asm("MOVW r1, #0x1000");
      asm("STR r1, [r0, r2]");
      for (int i = 0; i < conteo; ++i) asm("nop");
      // encender led
      asm("MOVW r0, #0x0C00"); // base puerto B
      asm("MOVT r0, #0x4001");
      asm("MOVW r2, #0x10");
      asm("MOVW r1, #0x0000");
      asm("MOVT r1, #0x1000");
      asm("STR r1, [r0, r2]");
  }
}
```

1. Ponga como fuente del reloj la frecuencia externa de 8 Mhz y use el PLL para obtener una frecuencia de Sysclk de 72 Mhz.
2. Ajuste el valor del registro “conteo” de forma que aproximadamente dure un segundo apagar y encender el led.
3. Ponga como fuente la frecuencia interna de 8 Mhz y haga que el reloj Sysclk sea de 72 Mhz. Observe si el led continúa parpadeando con la misma frecuencia.
4. Use el oscilador externo de 8 Mhz y ajuste el reloj Sysclk a una frecuencia igual a 36 Mhz. Observe que la velocidad de parpadeo bajo a la mitad, es decir, apagar y encender el led toma un tiempo de dos segundos.
5. Ahora use el oscilador interno de 8 Mhz y ajuste la frecuencia del reloj Sysclk a 8 Mhz. Observe que ahora encender y apagar el led tarda menos.
6. Sin variar el valor del registro “conteo” del paso 2, calcule el valor de la frecuencia del reloj Sysclk de forma que encender y apagar el led tarde 6 segundos.
7. Calcule la ecuación de tiempo que toma encender y apagar el led en función del valor del PLL y la frecuencia fuente.

Taller 3. Cambio de reloj y observación de la señal en un pin externo.

1. Crea un proyecto llamado reloj, siguiendo las instrucciones de cómo se crea un proyecto, con el microprocesador STM32F103C8T6.
2. Escriba las instrucciones para la configuración del reloj Sysclk con una frecuencia de 12 Mhz y configure el pin A8 para que muestre la frecuencia del PLL/2, poniendo este código dentro de main(), por fuera de while(1):

```
      // configuración del reloj Sysclk a 12Mhz:
      RCC_SYSCLKConfig(RCC_SYSCLKSource_HSI); // Asegura PLL no esté en uso
      RCC_PLLCmd(DISABLE);// deshabilitar el PLL
      RCC_PLLConfig(RCC_PLLSource_HSE_Div2, RCC_PLLMul_3);// 4Mz x 3= 12Mhz
      RCC_PLLCmd(ENABLE); // habilita el PLL
      while((RCC->CR & RCC_CR_PLLRDY) == 0);
      RCC_SYSCLKConfig(RCC_SYSCLKSource_PLLCLK); // se escoge PLL como Sysclk

      // configura la salida del reloj an A8:
```

```
RCC_MCOConfig(RCC_MCO_PLLCLK_Div2); // saca el reloj del PLL dividido en dos
GPIO_InitTypeDef GPIOA_Struct; // define la estructura para programar A8
GPIOA_Struct.GPIO_Pin  = GPIO_Pin_8;
GPIOA_Struct.GPIO_Speed = GPIO_Speed_50MHz; // máxima velocidad
GPIOA_Struct.GPIO_Mode = GPIO_Mode_AF_PP;// funciones alternas
RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOA, ENABLE);// habilitación del puerto.
GPIO_Init(GPIOA, &GPIOA_Struct);// configuración del puerto.
```

3. Tome un osciloscopio y conecte el alambre de tierra del osciloscopio a la tierra del módulo microprocesador. La punta de señal del osciloscopio la coloca en el pin A8 de la tarjeta Black o Blue Pill. Mida la frecuencia de la señal que sale y anótela aquí_____
4. ¿Cuál debería ser la frecuencia que sale en el pin A8?_____
5. ¿La forma de onda de la señal en el pin A8 debe ser cuadrada?_____
6. Modifique la instrucción para que ahora salga la señal del reloj interno por el pin A8.
   Escriba aquí la instrucción:_____ y la frecuecia en A8_____.
   Con un cautín, caliente el microprocesador un poco. Anote aquí la frecuencia_____
7. Haga lo mismo de (6) para el reloj externo.
   Escriba aquí la instrucción:_____ y la frecuecia en A8_____.
   Con un cautín, caliente el microprocesador un poco. Anote aquí la frecuencia_____
   Y lo mismo para el Sysclk.
   Escriba aquí la instrucción:_____ y la frecuecia en A8_____.

2.2. Reloj de tiempo real, RTC.

El módulo de reloj de tiempo real contiene varios contadores que continuamente están funcionando y que se usan para funciones de calendario o para interrupciones de alarma o interrupciones periódicas. El RTC tiene dos pines de entrada como oscilador de un cristal externo con una frecuencia de 32.768 hertz, LSE, un oscilador RC interno de 40 Khz, LSI, o la entrada de un oscilador externo de 8 Mhz dividido por 128. El módulo RTC tiene un contador programable de 32 bits con un registro de comparación. La entrada de pulsos al contador se puede dividir hasta por 20 bits programables también. Esto se puede ver en la figura 31.

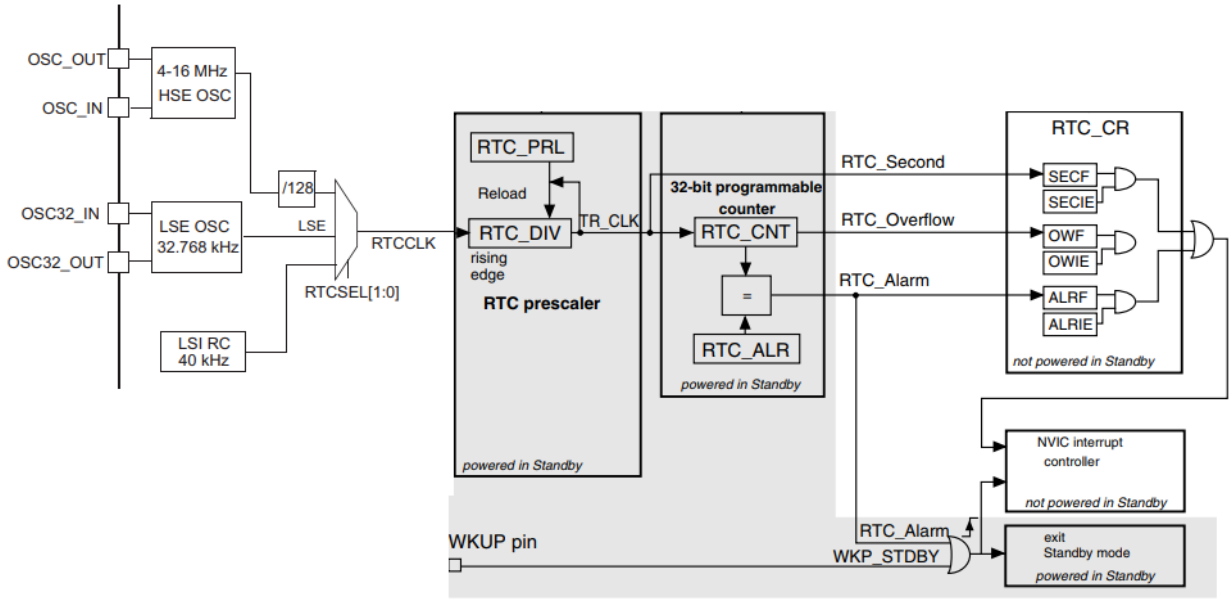

Figura 31. Fuentes de reloj del RTC.

El módulo RTC dispone de un pin del circuito integrado, que está alambrado en el módulo azul, pero no en el negro y que aparece como Vbat. Este pin sirve para conectar una batería, diferente a la fuente de alimentación de todo el circuito, para mantener los datos almacenados en los registros del módulo.
El módulo RTC se puede configurar para usarlo como:

- Registro de tiempo y fecha
- Programación de alarmas
- Programación para despertar en un estado de dormido
- Configuración de salida
- Configuración de registro de tiempo (TimeStamp).

Para poner a trabajar el reloj de tiempo real es necesario hacer una configuración previa. La configuración base es la siguiente:

- Habilitar el controlador de potencia con la función `RCC_APB1PeriphClockCmd(RCC_APB1Periph_PWR, ENABLE)`
- Habilitar el reloj externo con la función `RCC_LSEConfig(RCC_LSE_ON)`. Debe esperar que el reloj se estabilice con `while(RCC_GetFlagStatus(RCC_FLAG_LSERDY) == RESET)`.
- Seleccionar el reloj del módulo RTC con la función `RCC_RTCCLKConfig(RCC_RTCCLKSource_LSE)`. El parámetro `RCC_RTCCLKSource_LSE` es para seleccionar el reloj externo de 32.768 hertz, pero también se puede usar el parámetro `RCC_RTCCLKSource_LSI` para seleccionar el reloj interno de 40 khz o el parámetro `RCC_RTCCLKSource_HSE_Div128` para seleccionar el reloj externo de 8 Mhz dividido por 128.
- Habilitar el reloj del módulo RTC con la función `RCC_RTCCLKCmd(ENABLE)`. Después de esta función se debe esperar por la sincronización con la función `RTC_WaitForSynchro()`.

El siguiente ejemplo muestra la configuración del reloj RTC para usarlo como alarma. Parte de este ejemplo usa las instrucciones en lenguaje ensamblador que se vieron en los capítulos anteriores para manejar el led de la tarjeta.

```
int main(void)
{
    //configuración del reloj del puerto
    asm("MOVW r0, #0x1000");
    asm("MOVT r0, #0x4002"); //dirección base RCC= 0x4002.1000
    asm("MOVW r2, #0x18"); //offset de RCC_APB2ENR
    asm("LDR r1, [r0, r2]"); //lee su contenido
    asm("ORR r1, r1, #0x08"); //bandera de habilitación del puerto B
    asm("STR r1, [r0, r2]");

    //configuración de los bits como salida
    asm("MOVW r0, #0x0C00"); // base puerto B
    asm("MOVT r0, #0x4001"); //dirección base 0x4001.0C00
    asm("MOVW r2, #0x04"); //offset del registro GPIOB_CRH
    asm("MOVW r1, #0x0000");
    asm("MOVT r1, #0x0001");
    asm("STR r1, [r0, r2]");

    // habilitacion del reloj
    RCC_APB1PeriphClockCmd(RCC_APB1Periph_PWR, ENABLE);
    // habilita acceso al rtc
    PWR_BackupAccessCmd(ENABLE);
    // habilita el reloj externo
    RCC_LSEConfig(RCC_LSE_ON);
    // espera hasta que el reloj se estabilice
    while(RCC_GetFlagStatus(RCC_FLAG_LSERDY) == RESET);
    // selecciona el reloj externo como fuente
    RCC_RTCCLKConfig(RCC_RTCCLKSource_LSE);
    // habilita el rtc
    RCC_RTCCLKCmd(ENABLE);
    // espera por sincronizacion
    RTC_WaitForSynchro();
    while(RTC_GetFlagStatus(RTC_FLAG_RTOFF)== RESET);
    // programa alarma para 5 segundos
    RTC_SetAlarm(5);
    RTC_ClearFlag(RTC_FLAG_ALR);
    while(RTC_GetFlagStatus(RTC_FLAG_RTOFF)== RESET);

    while(1)
    {
        // apaga led
        asm("MOVW r0, #0x0C00"); // base puerto B
        asm("MOVT r0, #0x4001");
        asm("MOVW r2, #0x10");
        asm("MOVT r1, #0x0000");
        asm("MOVW r1, #0x1000");
        asm("STR r1, [r0, r2]");

        RTC_ClearFlag(RTC_FLAG_ALR);
        // pone contador en ceros
        RTC_SetCounter(0);
        // espera alarma
```

```
        while(RTC_GetFlagStatus(RTC_FLAG_ALR)== RESET);
        RTC_ClearFlag(RTC_FLAG_ALR);

        // enciende led
        asm("MOVW r0, #0x0C00"); // base puerto B
        asm("MOVT r0, #0x4001");
        asm("MOVW r2, #0x10");
        asm("MOVW r1, #0x0000");
        asm("MOVT r1, #0x1000");
        asm("STR r1, [r0, r2]");

        // pone contador en ceros
        RTC_SetCounter(0);
        // espera alarma
        while(RTC_GetFlagStatus(RTC_FLAG_ALR)== RESET);

    }
}
```

En el código de arriba, la primera parte es la configuración del pin del led en lenguaje ensamblador, como se había demostrado en el capítulo anterior. Luego aparece la configuración del reloj RTC programando la alarma para que dure cinco segundos. Dentro del "while(1)" se apaga el led, se inicia el contador del RTC en ceros y luego se espera por la alarma. Cuando la alarma activa su bandera, el led se enciende y entonces se inicia otra vez el contador en ceros para la espera de la nueva alarma. El led enciende y apaga continuamente cada cinco segundos.

Las siguientes son las funciones de la librería SPL disponibles para el RTC. Las primeras cinco funciones corresponden a configuración del reloj fuente; las que siguen son funciones propias del RTC.

- `void RCC_LSEConfig(uint8_t RCC_LSE);`

Esta función configura el reloj externo de baja velocidad que se usa en el RTC. El parámetro `RCC_LSE` puede ser `RCC_LSE_OFF`, para apagar el oscilador, `RCC_LSE_ON` para encenderlo, y `RCC_LSE_Bypass` cuando la señal de frecuencia viene conectada al pin de entrada del reloj.

- `void RCC_LSICmd(FunctionalState NewState);`

Esta función habilita o deshabilita el oscilador interno de baja velocidad, LSI. El parámetro puede ser `ENABLE` o `DISABLE`.

- `void RCC_RTCCLKConfig(uint32_t RCC_RTCCLKSource);`
- `void RCC_RTCCLKCmd(FunctionalState NewState);`

Estas dos funciones se usan para configurar el reloj fuente para el RTC y para habilitar el reloj del RTC, respectivamente. El parámetro de la primera función, `RCC_RTCCLKSource`, puede ser `RCC_RTCCLKSource_LSE`, para escoger el reloj externo de baja velocidad, o sea, el reloj de 32.768 Hz conectado en la tarjeta, `RCC_RTCCLKSource_LSI`, para escoger el reloj interno de baja velocidad, o sea, 40 khz, y `RCC_RTCCLKSource_HSE_Div128`, para escoger el reloj de alta velocidad externo dividido en 128, o sea, 62.500 hz.

- `void RTC_ITConfig(uint16_t RTC_IT, FunctionalState NewState);`

Esta función sirve para configurar la fuente de interrupción del RTC y habilitar la interrupción. El parámetro `RTC_IT` puede ser `RTC_IT_OW`, para fijar interrupción por rebasamiento, es decir, que el contador pasa de unos a ceros, `RTC_IT_ALR`, para fijar la interrupción por alarma, y `RTC_IT_SEC`, para que interrumpa cada segundo.

- `void RTC_EnterConfigMode(void);`
- `void RTC_ExitConfigMode(void);`

Estas dos funciones se usan para poder manipular los registros del contador RTC. Es una forma de proteger la información que está en los registros. De todas formas estas dos funciones son llamadas por las demás funciones del RTC para acceder a los registros del RTC.

- `uint32_t  RTC_GetCounter(void);`
- `void RTC_SetCounter(uint32_t CounterValue);`

Estas dos funciones se usan para leer el valor del contador y para darle un nuevo valor al contador del RTC, respectivamente.

- `void RTC_SetPrescaler(uint32_t PrescalerValue);`

Esta función sirve para programar el valor del Preescaler, o divisor del reloj de entrada. El Prescaler es un registro que guarda el número de pulsos `RTCCLK` que debe contar. El contador de pulsos se llama `RTC_DIV` y cada número de pulsos en el Prescaler es un segundo. Estos pulsos de un segundo alimentan el contador `RTC_CNT`, o sea que este registro cuenta el número de segundos transcurridos.

- `void RTC_SetAlarm(uint32_t AlarmValue);`

Esta función fija el valor de la alarma en segundos. Cuando el valor de la alarma es igual al contador de segundos, `RTC_CNT`, se activa la bandera de alarma.

- `uint32_t  RTC_GetDivider(void);`

Con esta función se puede leer el valor del contador de pulsos `RTCCLK` que es el número de pulsos que se ha contado durante el transcurso de un segundo. El valor del contador está entre cero y el valor guardado en el Prescaler.

- `void RTC_WaitForLastTask(void);`

Esta función se usa dentro de las otras funciones que modifican los registros del RTC para ver la disponibilidad del registro a ser modificado.

- `void RTC_WaitForSynchro(void);`

Esta función se usa dentro de las funciones que modifican los registros `RTC_CNT`, `RTC_ALR` y `RTC_PRL` para esperar la sincronización de estos registros con el reloj del bus APB.

- `FlagStatus RTC_GetFlagStatus(uint16_t RTC_FLAG);`
- `void RTC_ClearFlag(uint16_t RTC_FLAG);`

Estas dos funciones se usan para obtener información de las banderas de estado del RTC y para borrar cualquier bandera del RTC. El parámetro de entrada es la bandera que se quiere leer o borrar, respectivamente, y puede ser `RTC_FLAG_RTOFF`, que indica que el RTC no está contando, `RTC_FLAG_RSF`, bandera de sincronización, `RTC_FLAG_OW`, para indicar que el contador de segundos pasó de unos a ceros, `RTC_FLAG_ALR`, que indica que el contador de segundos se hizo igual al valor del registro de alarma, y `RTC_FLAG_SEC`, que es la activación de esta bandera cada segundo.

- `ITStatus RTC_GetITStatus(uint16_t RTC_IT);`
- `void RTC_ClearITPendingBit(uint16_t RTC_IT);`

Estas dos funciones se usan para leer el estado de las banderas de interrupción del RTC y para borrar estos estados, respectivamente. El parámetro de entrada a estas dos funciones puede ser `RTC_IT_OW`, para indicar que el contador de segundos, `RTC_CNT`, pasó de unos a ceros, `RTC_IT_ALR`, que hay una interrupción por alarma y `RTC_IT_SEC`, que la interrupción ocurrió por haber transcurrido un segundo.

El siguiente ejemplo es una función para configurar el RTC a un segundo.

```
uint8_t RTC_Init(void)
{
    // habilitación del reloj
    RCC_APB1PeriphClockCmd(RCC_APB1Periph_PWR, ENABLE);
    // habilita acceso al rtc
    PWR_BackupAccessCmd(ENABLE);
    // habilita el reloj externo
    RCC_LSEConfig(RCC_LSE_ON);
    // espera hasta que el reloj se estabilice
    while(RCC_GetFlagStatus(RCC_FLAG_LSERDY) == RESET);
    // selecciona el reloj externo como fuente
    RCC_RTCCLKConfig(RCC_RTCCLKSource_LSE);
    // habilita el rtc
    RCC_RTCCLKCmd(ENABLE);
    // espera por sincronización
    RTC_WaitForSynchro();
    while(RTC_GetFlagStatus(RTC_FLAG_RTOFF)== RESET);
    RTC_SetPrescaler(32768); // divisor del reloj externo
    RTC_SetCounter(0);

    return 0;
}
```

Aquí se usa la función `RTC_SetPrescaler(32768)` para dividir la frecuencia que entra al contador RTC. Esta frecuencia se puede escoger de tres fuentes, pero para este ejemplo se escogió la frecuencia del oscilador externo de baja frecuencia, LSE, con la función `RCC_RTCCLKConfig(RCC_RTCCLKSource_LSE)`.
Esta frecuencia es de un oscilador de cristal de 32.768 hertz, por tanto, si se divide esta frecuencia por el mismo valor, va a tener una salida de pulsos con una frecuencia de un segundo. Inicialmente el contador de pulsos se pone en cero con la función `RTC_SetCounter(0)`, pero este contador tiene 32 bits, por lo que puede contar pulsos de un segundo por muchos años. Se puede leer todo el tiempo el contador con la función RTC_GetCounter(), que devuelve

el conteo de segundos. Dentro del while(1) se puede actualizar el valor de un registro con la lectura del contador para tomar el tiempo en segundos. En este ejemplo, cada dos segundos se enciende y apaga el led del módulo.

```
uint32_t reloj= RTC_GetCounter();
while(1)
{
    if((RTC_GetCounter()- reloj)> 2)
    {
        reloj= RTC_GetCounter();
        if(GPIO_ReadOutputDataBit(GPIOB, GPIO_Pin_12)== Bit_RESET)
           GPIO_SetBits(GPIOB, GPIO_Pin_12);
        else
           GPIO_ResetBits(GPIOB, GPIO_Pin_12);
    }
}
```

Cuestionario:

1. ¿Cuál debe ser el valor del divisor del reloj del RTC si la frecuencia de entrada es el reloj interno de 40 khz y se desea tener una frecuencia de entrada al contador del RTC de 1 hz?

2. ¿Cuál debe ser el valor del divisor del reloj del RTC si la frecuencia de entrada es el oscilador externo de alta velocidad con una frecuencia de 8 Mhz y se desea tener una frecuencia de entrada al contador RTC de 1 hz?

3. La frecuencia de entrada al contador RTC es de 1 Hz. Suponga que se necesita una alarma que se active a los 37 minutos. ¿Cuál debe ser el valor que se programa en el registro de alarma?

4. Dado el programa de inicialización de arriba, ¿Cuáles líneas se cambian para cambiar el reloj de entrada al RTC del reloj externo de baja velocidad de 32.768 hz al reloj interno de baja velocidad de 40 khz?

5. Dado el programa de inicialización de arriba, ¿Cuáles líneas se cambian para cambiar el reloj de entrada al RTC del reloj externo de baja velocidad de 32.768 hz al reloj externo de alta velocidad dividido en 128?