UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Configuración de la frecuencia de trabajo de un microcontrolador STM32F4

Apellidos, nombre	Yuste Pérez, Pedro (pyuste@disca.upv.es)
Departamento	Departamento de Informática de Sistemas y Computadores
Centro	Universidad Politécnica de Valencia

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

1 Resumen de las ideas clave

En este artículo se introduce el sistema de relojes de la familia STM32F4 de los microcontroladores ARM Cortex. Se enumeran las fuentes de reloj que se pueden utilizar para generar el reloj del sistema y se explica, paso a paso, cómo hacer la configuración más habitual para conseguir la frecuencia deseada.

2 Introducción

Una característica fundamental de los sistemas empotrados basados en microcontrolador es la capacidad para trabajar con tiempos con una gran precisión. Para ello es necesario que la frecuencia que se toma como base en el sistema sea la adecuada. Por un lado, una frecuencia demasiado baja hará que la cantidad de instrucciones que el sistema puede ejecutar por unidad de tiempo no sea suficiente para la aplicación o que los temporizadores que dependen de esa frecuencia no consigan la resolución necesaria para su función. Por otro, una frecuencia excesivamente alta incrementa de manera innecesaria el consumo energético, lo que puede ser determinante en, por ejemplo, aplicaciones móviles que dependan de baterías.

En este artículo se introduce el sistema de relojes de la familia STM32F4 de microcontroladores ARM Cortex y se explica cómo configurarlo para que el sistema funcione a la frecuencia deseada. Se asume que el lector tiene un conocimiento básico de los STM32F4 y de las librerías CMSIS.

3 Objetivos

Una vez que el alumno se lea con detenimiento este documento, será capaz de:

- Comprender el funcionamiento del bloque de relojes de los microncontroladores STM32F4
- Calcular los parámetros necesarios para conseguir una frecuencia arbitraria.
- Escribir programas que trabajen a la frecuencia deseada.

4 Desarrollo

El sistema de relojes del STM32F4 es un circuito que tiene varias posibles fuentes de reloj y permite utilizar una cualquiera de ellas para generar la salida. La salida que queremos obtener es SYSCLK, y las fuentes posibles son las siguientes:

- Circuito de reloj principal:
    - HSI: High Speed Internal Oscillator. Es un oscilador RC de 16MHz interno al microcontrolador. Tiene poca precisión pero es económico ya que no necesita componentes externos.
    - HSE: High Speed External Oscillator. Es un oscilador RC que necesita un cristal externo. Permite conseguir una señal de reloj más estable.

- PLL: Phase Locked Loop. Toma como entrada uno de los dos anteriores y permite obtener una frecuencia de salida que sea múltiplo de la que tiene en su entrada. Es el más utilizado.

- Circuito de reloj secundario:
  - Aunque se sale de los objetivos de este artículo, conviene saber que hay una serie de periféricos que, en modos de bajo consumo, obtienen su entrada a partir de un segundo circuito de reloj. Este circuito también tiene dos posibles fuentes: LSI y LSE.

En la figura siguiente se puede ver un esquema de las posibles fuentes de reloj, extraído de [1].

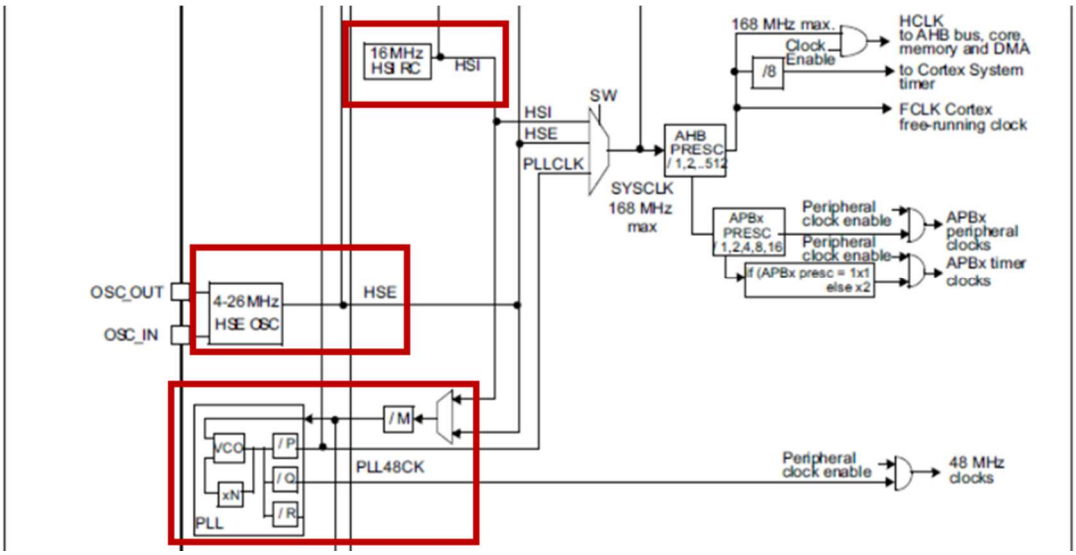

Figura 1. Esquema del circuito de reloj principal

4.1 PLL

Centrándonos en el PLL podemos ver que la señal atraviesa los siguientes bloques:

- Un multiplexor, para poder elegir HSI o HSE como entrada.
- Un divisor, que divide la frecuencia de entrada por un valor configurable (M)
- El PLL, que consigue en su salida una frecuencia igual a la de su entrada multiplicada por un valor N.
- Al final se obtienen las salidas P y Q tras pasar la señal por otros dos divisores de frecuencia.

La frecuencia que se obtiene en la salida sigue las fórmulas:

$$f_{(VCO\ Clock)} = f_{(PLL\ Clock\ Input)} x \left(\frac{N}{M}\right)$$

$$f_{(PLL\ Clock\ Output)} = \frac{f_{(VCO\ Clock)}}{P}$$

Ecuación 1. Frecuencia de salida del PLL.

Los parámetros M, N, P y Q tienen las siguientes limitaciones:

1. M: Se debe ajustar para que la entrada al VCO esté entre 1 y 2MHZ, cuanto más alta sea esta frecuencia, mejor estabilidad tendrá el VCO.
2. M debe cumplir: 2≤M≤63.
3. N: Se debe ajustar para que la salida del VCO esté entre 192 y 432MHz
4. N debe cumplir: 192≤N≤432
5. P: Se debe ajustar para que a su salida la frecuencia máxima sea de 168MHz.
6. P debe ser uno de los valores siguientes: 2, 4, 6, 8

Con todas estas condiciones ya se puede deducir que el conjunto de frecuencias utilizables es finito. Podemos hacernos la siguiente pregunta, por ejemplo: ¿Cuáles podrían ser las frecuencias máximas y mínimas?

- De acuerdo con la condición 5, la máxima sería 168MHz.
- Teniendo en cuenta las condiciones 3 y 6, la frecuencia mínima (utilizando el PLL) sería 192/8=24MHz

4.2 Configuración

Vamos a ver qué pasos tenemos que seguir para configurar el reloj y dónde se configura cada cosa. A grandes rasgos deberíamos seguir los siguientes pasos:

1. Activar la fuente de reloj
2. Configurar el PLL si es necesario
3. Elegir de dónde tomamos SYSCLK

4.2.1 Activar fuente de reloj

Todas las configuraciones del reloj se hacen escribiendo en los registros del bloque RCC. En concreto, las diferentes fuentes de reloj se activan en el registro RCC_CR (Configuration Register) que se puede ver en la figura siguiente:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved				PLLI2S RDY	PLLI2S ON	PLLRDY	PLLON	Reserved				CSS ON	HSE BYP	HSE RDY	HSE ON
				r	rw	r	rw					rw	rw	r	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
HSICAL[7:0]								HSITRIM[4:0]					Res.	HSI RDY	HSION
r	r	r	r	r	r	r	r	rw	rw	rw	rw	rw		r	rw

Figura 2. Registro de configuración del reloj (RCC_CR)

En primer lugar debemos activar la fuente que, como hemos visto, puede ser HSI, HSE o PLL. Para ello tenemos los bits HSI_ON, HSE_ON y PLL_ON. Cuando la fuente se haya

estabilizado, el sistema nos activa los bits que confirman que la fuente correspondiente está en marcha: HSI_RDY, HSE_RDY y PLL_RDY.

4.2.2 Configurar el PLL

Si hemos decidido utilizar el PLL habrá que escribir sus parámetros. Se hace en el registro RCC_PLLCFGR cuyo contenido se puede ver en la figura siguiente:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved				PLLQ3	PLLQ2	PLLQ1	PLLQ0	Reserved	PLLSRC	Reserved				PLLP1	PLLP0
				rw	rw	rw	rw		rw					rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	PLLN									PLLM5	PLLM4	PLLM3	PLLM2	PLLM1	PLLM0
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Figura 3. Registro de configuración del PLL (RCC_PLLCFGR)

Siguiendo la secuencia que ya hemos visto, los parámetros a escribir son los siguientes:

- Configuración del multiplexor de entrada: Escribiremos un 0 o un 1 en PLL_SRC si queremos partir de HSI o HSE respectivamente.
- Divisor M: Escribimos en los bits PLLM[0-5] el valor del divisor M.
- Multiplicador N: Escribimos en los bits PLLN[0-8] el valor del multiplicador N.
- Divisor P: Escribimos en PLLP[0,1] uno de los siguientes valores:
  - 00- P=2
  - 01- P=4
  - 10- P=6
  - 11- P=8

4.2.3 Elegir salida de reloj hacia SYSCLK

Por último, cuando el reloj elegido se ha estabilizado hay que decirle al sistema cuál de ellos es el que queremos conectar a SYSCLK. Se configura en el registro RCC_CFGR.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MCO2		MCO2 PRE[2:0]			MCO1 PRE[2:0]			I2SSCR	MCO1		RTCPRE[4:0]				
rw		rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PPRE2[2:0]			PPRE1[2:0]			Reserved		HPRE[3:0]				SWS1	SWS0	SW1	SW0
rw	rw	rw	rw	rw	rw			rw	rw	rw	rw	r	r	rw	rw

Figura 4. Registro de configuración de salidas (RCC_CFGR)

En este registro tenemos los bits SW[0,1] y los bits SWS[0,1]. En SW pedimos un cambio y en SWS comprobamos si el cambio se ha hecho efectivo. El significado de estos bits es el siguiente:

- SW (System Clock Switch):
  - 00- Elegimos HSI como SYSCLK
  - 10- Elegimos HSE como SYSCLK

  - 10- Elegimos PLL como SYSCLK
- SWS (System Clock Switch Status):
  - 00- La salida de HSI está conectada a SYSCLK
  - 01- La salida de HSE está conectada a SYSCLK
  - 10- La salida del PLL está conectada a SYSCLK

4.3 ¿Y todo esto cómo lo hago?

Para implementarlo podemos utilizar las librerías CMSIS que nos ayudarán bastante al tener predefinidos todos los registros y máscaras necesarios. Vamos a ver ejemplos de los pasos anteriores:

4.3.1 Activar fuente de reloj

En el siguiente trozo de código vemos cómo activar el HSE:

```
/* Enable HSE */
RCC->CR |= ((uint32_t)RCC_CR_HSEON);

/* Wait till HSE is ready and if Time out is reached exit */
do
{
  HSEStatus = RCC->CR & RCC_CR_HSERDY;
  StartUpCounter++;
} while((HSEStatus == 0) && (StartUpCounter != HSE_STARTUP_TIMEOUT));
```

En la primera línea sumamos la máscara HSE_ON al registro RCC_CR. Con eso ponemos a 1 el bit correspondiente a la activación. En el resto esperamos a que el bit HSE_RDY se ponga a 1, aunque si pasa mucho tiempo (HSE_STARTUP_TIMEOUT) asumimos que hay un error y acabamos.

4.3.2 Configurar el PLL

Ahora veamos el código necesario para configurar el PLL:

```
/* Configure the main PLL */
RCC->PLLCFGR = PLL_M | (PLL_N << 6) | (((PLL_P >> 1) -1) << 16) |
               (RCC_PLLCFGR_PLLSRC_HSE) | (PLL_Q << 24);

/* Enable the main PLL */
RCC->CR |= RCC_CR_PLLON;

/* Wait till the main PLL is ready */
while((RCC->CR & RCC_CR_PLLRDY) == 0)
{
}
```

En la primera línea escribimos los parámetros M, N, P (y Q) en RCC_PLLCFGR. Los desplazamientos se deben a la posición del bit en el que empieza el campo correspondiente en el registro. M empieza en el bit 0 y no se desplaza, N empieza en el bit 6 y P empieza en el bit 16.

Una vez escritos los parámetros de configuración, al igual que en el caso anterior, activamos el PLL y esperamos a que se estabilice.

4.3.3 Elegir salida de reloj hacia SYSCLK

Por último, en el siguiente código elegimos el PLL como SYSCLK:

```
/* Select the main PLL as system clock source */
RCC->CFGR &= (uint32_t)((uint32_t)~(RCC_CFGR_SW));
RCC->CFGR |= RCC_CFGR_SW_PLL;

/* Wait till the main PLL is used as system clock source */
while ((RCC->CFGR & (uint32_t)RCC_CFGR_SWS ) != RCC_CFGR_SWS_PLL);
{
}
```

En la primera línea ponemos a 0 los bits que forman parte del campo SW. A continuación escribimos la máscara correspondiente al PLL y esperamos a que el campo SWS confirme que se ha actualizado correctamente.

4.4 Pero, ¿No se puede hacer de manera más sencilla?

En realidad CMSIS tiene una función que nos facilita mucho las cosas. Si sólo queremos configurar el sistema para que el reloj sea el deseado y no vamos a necesitar cambiar la frecuencia mientras el programa se ejecuta, podemos editar el archivo system_stm32f4xx.c de las librerías CMSIS. Dentro de este archivo nos encontramos con la función SystemInit() que es llamada en el arranque del sistema. Esta función configura varios parámetros del microcontrolador y llama a la función SetSysClock(). Esta última función ejecuta el código que hemos visto más arriba, por lo que sólo será necesario modificar, dentro del archivo system_stm32f4xx.c las líneas siguientes:

```
/************************ PLL Parameters *************************************/
/* PLL_VCO = (HSE_VALUE or HSI_VALUE / PLL_M) * PLL_N */
#define PLL_M      25
#define PLL_N      336

/* SYSCLK = PLL_VCO / PLL_P */
#define PLL_P      2
```

5 Cierre

A lo largo de este objeto de aprendizaje hemos tratado los fundamentos del sistema de relojes de la familia STM32F4 y se ha visto el código necesario para cambiar la frecuencia por la deseada. Para comprobar que se ha entendido se recomienda hacer los siguientes ejercicios:

- A partir de la configuración por defecto cambiar los parámetros del PLL para conseguir una frecuencia de 84MHz.

- Conseguir la misma frecuencia de 84 MHz, pero utilizando el HSI como base de tiempos.
- Utilizar directamente el HSI o el HSE como frecuencia del sistema, sin utilizar el PLL.

6 Bibliografía

[1] STMicroelectronics: "RM0090 Reference manual. STM32F40xxx, STM32F41xxx, STM32F42xxx, STM32F43xxx advanced ARM-based 32-bit MCUs", Doc ID 018909 Rev 4. 2013. URL: http://www/st.com