Microcontroladores: Periféricos

PID_00247319

Màrius Montón Macián

Tiempo mínimo de dedicación recomendado: 2 horas

Universitat Oberta de Catalunya

Índice

Introducción

En este contenido se enumeran los periféricos más comunes que se pueden encontrar en los microcontroladores actuales. Se hace una descripción genérica de cada uno, y para más información o detalles se deberán consultar los manuales del fabricante en concreto.

1. GPIO

Un *general purpose input output* es el periférico o conjunto de periféricos que controlan las entradas y salidas de los pines del microcontrolador.

Habitualmente es posible configurar cada uno de los pines para que actúe como entrada o salida y cada uno de ellos con distintas configuraciones. Así, un pin configurado como entrada puede tener asociado un pequeño filtro *anti-glitches*, o un pin de salida puede tener conectada una resistencia de *pull-up*.

Glitch

Un *glitch* es una variación momentánea y errónea del valor de una señal. Un filtro *anti-glitch* filtra la señal de manera que estos *glitches* no aparezcan una vez pasada la señal por el filtro.

Además, este periférico es el encargado de configurar si un pin en concreto debe generar o no una interrupción cuando cambie de valor, o reciba un cambio de flanco, etc.

2. Timers

Este tipo de periférico no es más que un contador accesible y controlado por la CPU. Así, una vez inicializado, el Timer empezará a incrementar o decrementar un contador a cada pulso de reloj. Cuando el contador llegue a un valor determinado, el Timer generará una interrupción para alertar a la CPU de este evento.

Este tipo de periférico se usa habitualmente para llevar un control fino del tiempo sin tener a la CPU dedicada en exclusiva a esta tarea. Así, por ejemplo, se puede configurar el Timer para que genere una interrupción cada medio segundo y de esta manera hacer parpadear un LED.

Hay variantes de Timers que en lugar de incrementar o decrementar el contador con cada pulso de reloj, lo hacen con cada pulso de una entrada externa. Así, es posible contar el número de pulsos de una entrada determinada sin tener que manejar interrupciones o ningún tipo de control adicional.

También es habitual encontrarse Timers capaces de generar señales externas cuando el contador llega a un valor determinado, de manera que se puede generar una señal dada una condición previa sin ningún tipo de intervención de la CPU.

3. USART

Los periféricos tipo USART (*universal synchronous / asynchronous receiver / transmitter*) son los encargados de manejar las comunicaciones tipo serie síncronas y asíncronas y SPI.

Estos periféricos suelen tener un pequeño *buffer* de datos y una vez configurados reciben y transmiten los datos según la configuración dada.

Habitualmente pueden ser configurados para que generen interrupciones por cada paquete recibido o enviado y, en algunos fabricantes, pueden generar interrupción cuando se finaliza la recepción de un paquete (mediante la configuración previa de cómo se marca la finalización de un paquete).

Debido a las altas tasas de transferencias de datos que se pueden llegar a dar en una comunicación serie (del orden de centenares de miles de bits por segundo), estos periféricos pueden trabajar junto con el DMA (véase el apartado 9) para descargar la CPU del trabajo de procesar cada uno de los datos recibidos, y solo alertarla cuando se ha recibido todo un conjunto de datos grande.

4. I2C

Es muy habitual encontrar un periférico integrado en los microcontroladores actuales capaz de conectarse al bus I2C.

Suelen ser periféricos muy sencillos de usar, donde se indica la dirección del esclavo que hay que interrogar y los datos que enviar o recibir. Debido a la lentitud del bus respecto al funcionamiento interno del microcontrolador (kHz frente a MHz), estos periféricos generan una interrupción cuando han acabado la transferencia.

Aunque el uso normal es que el microcontrolador sea el *master* del bus, es posible trabajar en modo esclavo y que sea otro *master* el que maneje el bus y nos lance peticiones.

5. RTC

Es también común encontrar un periférico RTC (*real-time clock*). Este periférico se encarga de mantener un calendario interno, normalmente mediante un contador muy grande que va contando los segundos transcurridos. Suelen llevar conectados un reloj externo de bajo consumo y baja frecuencia, y lo más habitual es que necesiten un cristal de 32.768 kHz.

Estos periféricos, aparte de llevar la cuenta del tiempo, pueden ser programados para que generen una interrupción cada cierto número de segundos transcurridos.

6. Watchdog

Este periférico consiste en un contador (normalmente decreciente) que al llegar a término (normalmente a 0) genera un *reset* al sistema. A este periférico se le debe ir «alimentando» cada cierto tiempo para reiniciar su contador y así que no reinicie el sistema.

Este periférico se usa como salvaguarda de todo el sistema, de manera que el desarrollador va «alimentando» el Watchdog cada cierto tiempo para que el sistema funcione con normalidad. Si llega el caso en el que, por un error del tipo que sea, el código no «alimenta» durante un periodo de tiempo el Watchdog, este reiniciará el dispositivo y el sistema volverá a funcionar desde cero.

7. ADC

Muchos microcontroladores llevan integrado algún tipo de periférico ADC (*analog-to-digital converter*). Aunque no suelen ser de una gran calidad debido a su integración con el resto de la circuitería digital, pueden resultar útiles para la conversión de voltajes tales como la alimentación de entrada, señales analógicas provenientes de sensores tipo 0-10 voltios, etc.

En un microprocesador común de 32 bits podemos encontrar un periférico ADC, siendo este de tipo SAR (aproximaciones sucesivas) y de 12 bits de resolución. Este ADC es capaz de muestrear hasta un millón de muestras por segundo. En algunos casos, delante del ADC hay un MUX analógico, de modo que se pueden convertir distintas señales analógicas provenientes de distintos pines de entrada/salida del microcontrolador.

Como es habitual en los periféricos que pueden generar gran cantidad de datos, es posible configurarlo para que trabaje junto con el DMA; en este caso para que almacene el resultado de la conversión directamente en la memoria de datos y que, ocasionalmente, genere una interrupción para notificar a la CPU que los datos están disponibles.

8. DAC

Lo dicho para los ADC puede decirse para los DAC (*digital-to-analog converter*): hay microcontroladores que integran un periférico de este tipo para generar señales analógicas directamente, sin necesidad de componentes externos.

Parámetros habituales de DAC integrados en microcontroladores son resolución de 12 bits y hasta quinientas mil muestras por segundo. También es habitual que la salida analógica pueda ser dirigida a varios pines de entrada/salida.

De nuevo, y debido a la necesidad de transferir datos periódicamente hacia el periférico DAC, es posible usar el DMA para que realice esta tarea descargando así a la CPU.

9. DMA

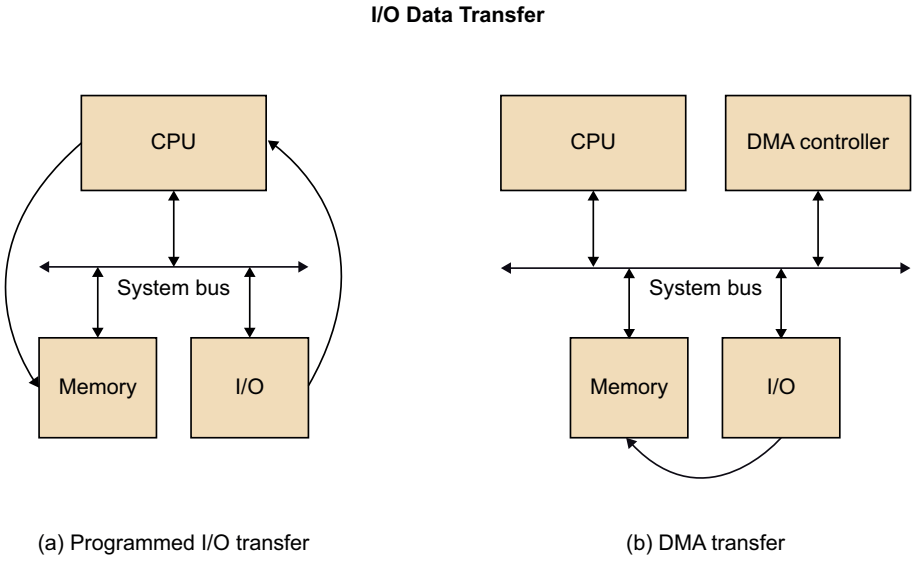
El periférico DMA (*direct memory access*) es un módulo programable que puede encargarse de hacer transferencias de datos entre zonas de memoria o entre zonas de memoria y periféricos sin la intervención directa de la CPU.

Así, por ejemplo, el DMA puede configurarse para que copie una zona de memoria a otra zona de memoria, siendo necesario pasarle las direcciones de origen y destino, y la cantidad de datos que copiar (figura 1).

Otra aplicación común es la de almacenar datos generados por otro periférico e ir guardando esos datos de manera consecutiva en un *buffer*. Para este caso, los parámetros que debemos configurar son la fuente de datos, esto es, qué periférico generará los datos, la dirección de inicio del *buffer* donde almacenarlos y la cantidad máxima que se pueden almacenar. En este caso, es habitual también especificarle si se quiere recibir un aviso en forma de interrupción cuando el *buffer* se haya llenado o esté a punto de hacerlo.

En todo caso, el objetivo es que la CPU quede descargada de este tipo de tareas y pueda dedicarse a otras operaciones o permanecer dormida. El DMA genera una interrupción para notificar a la CPU que ha terminado la transferencia y así la CPU puede o bien pasar a procesar esos datos, o despertarse y empezar a trabajar.

Figura 1. Transferencia DMA



(a) Programmed I/O transfer

(b) DMA transfer

Los microcontroladores de últimas generaciones llevan DMA que pueden manejar múltiples transferencias simultáneamente organizadas en lo que se suele llamar canales. De este modo, es posible, por ejemplo, usar el DMA para ir guardando los datos recibidos por un puerto serie a la vez que otro canal del DMA se está usando para generar una señal analógica mediante la transferencia de un *buffer* en memoria precalculado hacia el periférico DAC; todo esto mientras la CPU está en modo de bajo consumo, sin hacer ninguna operación.

10. Debug

Por *debug* se entiende toda actividad orientada a encontrar y solucionar *bugs* y errores en el código desarrollado. Para ello, los microcontroladores actuales contienen periféricos especiales para estas tareas.

Así, es posible dentro de una sesión de Debug indicar al procesador que se pare si se da una condición determinada o si se llega a una línea de código dada.

Una vez parada la ejecución, es posible recuperar el valor de las variables en ese momento, o hacer un volcado de ciertas partes de la memoria para saber qué datos hay almacenados. El desarrollador también puede, a partir de ese punto, ejecutar su código paso a paso, y así poder comprobar qué se está ejecutando y en qué orden, etc.

Habitualmente para acceder a las funciones de Debug se requiere un dispositivo externo específico, que es el encargado de comunicar nuestro ordenador personal donde se hace el desarrollo con el microcontrolador (ver figura 2). Esto se lleva a cabo a través del puerto de Debug, que son una serie de pines de entrada y salida especiales para la programación y el Debug del microcontrolador.

Figura 2. Dispositivo de Debug

Fuente:
https://blog.adafruit.com/2013/05/16/new-product-segger-j-link-edu-jtagswd-debugger/

11. Otros periféricos

Si bien hemos presentado los periféricos más habituales en un microcontrolador actual, existen otros periféricos no tan habituales pero que cada vez más los fabricantes van añadiendo a sus dispositivos.

Así, podemos encontrar:

11.1. Módulos para criptografía

Algunos fabricantes añaden módulos especiales para manejar operaciones *ad hoc* o algoritmos de criptografía. Algunos de estos módulos pueden cifrar o descifrar directamente un *buffer* de memoria con una clave dada, calcular un CRC o llevar a cabo operaciones similares [Silicon Labs (a)].

De esta manera se descarga la CPU de estas operaciones, pudiendo incluso estar en modo de bajo consumo y parada mientras se realizan las operaciones criptográficas.

11.2. Contador de pulsos

Algunos fabricantes dotan a alguno de sus *timers* de la capacidad de llevar la cuenta no según un reloj, sino con una entrada externa y calcular así los pulsos que ha generado dicha señal [Silicon Labs (2013)].

Esto puede usarse para leer sensores que indican su lectura a través de pulsos sin la intervención directa de la CPU.

11.3. Comparadores de voltaje

Hay fabricantes que ofrecen este periférico como módulo distinto del ADC, de manera que existe un comparador de cierto voltaje de entrada y un voltaje fijo y que es capaz de generar una interrupción si dicho valor cambia [STMicroelectronics (2017b)].

De este modo resulta muy sencillo monitorizar, por ejemplo, el valor de tensión de entrada de una batería, y lanzar una alarma o indicación de que la batería se está agotando.

11.4. LCD

La mayoría de los fabricantes ofrecen alguna familia de sus dispositivos con un periférico de control para pantallas tipo LCD [STMicroelectronics (2017c)].

Estas pantallas requieren unas señales de control con unos tiempos y protocolos determinados. Estos periféricos realizan la tarea de controlar estas señales para un correcto manejo de la pantalla sin cargar de tareas a la CPU.

11.5. USB

También se suelen encontrar familias de dispositivos que llevan integrado un controlador de puerto USB. De esta manera se puede tener disponible un puerto USB en el microcontrolador de forma sencilla y usando solamente algunos componentes externos menores [Texas Instruments (2009)].

Debido a la complejidad del protocolo USB, los periféricos USB llevan asociada una librería proporcionada por el mismo fabricante para su manejo de forma sencilla*.

*** Véase el apartado «Librerias habituales» dentro del material titulado «Programación» para más detalles.**

Bibliografía

Silicon Labs (a). «AN0955: CRYPTO».
URL: https://www.silabs.com/documents/ public/application-notes/AN0955.pdf.

Silicon Labs (2013). «Pulse Counter».
URL: https://www.silabs.com/documents/ public/application-notes/AN0024.pdf.

STMicroelectronics (2017b). «Getting started with analog comparators for STM32F3 Series devices».
URL: http://www.st.com/resource/en/application_note/dm00074240.pdf.

STMicroelectronics (2017c). «LCD-TFT display controller (LTDC) on STM32 MCUs».
URL: http://www.st.com/resource/en/application_note/dm00287603.pdf.

Texas Instruments (2009). «The ultra-low-power USB revolution: Bringing power efficiency and simplicity to USB for portable embedded applications».
URL: http://www.ti.com/lit/wp/slay014/slay014.pdf.