Electrónica Digital II

Santiago Rúa Pérez, PhD.

18 de septiembre de 2022

TEMPORIZADORES

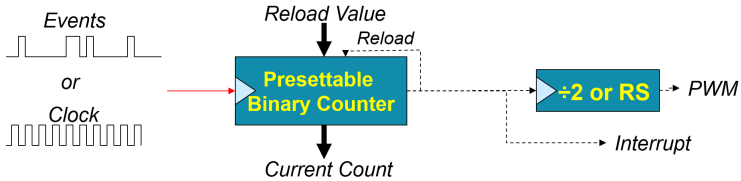
Objetivos

- Entender como funciona el temporizador de todo procesador Cortex-M
- Comprender el funcionamiento del PIT
- Entender el funcionamiento del PWM

Temporizadores del K64

- SYSTICK
  - Parte de la CPU, y específicamente en la arquitectura Cortex-M
  - Genera interrupción periódica
- PIT - Periodic Interrupt Timer
  - Puede ser generado periódicamente por interrupción o triggerear transferencia DMA (acceso directo a memoria)
- FTM - FlexTimer/Módulo PWM
  - Conectarse a I/O pines y soportar captura de entrada y comparación de salida.
  - Puede generar señales PWM
  - Puede generar interrupciones y peticiones del DMA
- LPTMR - Low Power Timer
  - Puede operar como temporizador o contador en todos los modos de potencia
  - Puede despertar el sistema con interrupción
  - Puede triggerear hardware.
- Real-Time Clock
  - Energizado por cristal externo de $32.768\,\text{kHz}32.768\ \text{kHz}$
  - Mantiene dato del tiempo en registro de 32 bits
  - Puede configurar alarma
  - Puede generar señal de $1\,\text{Hz}$
  - Puede despertar el sistema

Introduccion a los temporizadores y contadores



- Es muy común en todos los microcontroladores
- Basado en un contador binario
- Dos modos básicos
  - **Modo contador**: cuenta pulsos los cuales indican eventos (eg. odómetro)
  - **Modo temporizador**: la fuente del reloj es periódica, entonces el valor del contador el proporcional al tiempo transcurrido.

- Principales configuraciones
  - Valor del contador actual
  - Valor de carga del contador
  - Dirección de conteo
  - Fuente del reloj
  - Acciones de desbordamiento

SysTick Timer

- La arquitectura del procesador Cortex M contiene un temporizador integrado
- Contador de 24 bits en decremento.
- Tiene cuatro registros (SysTick): CTRL, LOAD, VAL, y CALIB.
- Cuando se habilita el contador de 24 bits es cargado y cada pulso de reloj hace que decremente
- En el registro VAL se puede leer el valor actual del contador
- Se puede configurar su funcionamiento con el registro de control CTRL
  - El CLKSOURCE define la fuente de reloj. Para el K64 está siempre conectado al core clock.
  - El TICKINT define si se activa la interrupción
  - ENABLE habilita el contador con 1
  - COUNTFLAG retorna 1 si el contador ha llegado a cero

Ejemplo - SysTick Timer

- Realizar un programa que genere una interrupción del SysTick cada 1 ms. Adicionalmente, que togglee un led cada 1 s.
- **Solución**
  - Inicialice el clock del procesador.
  - Inicialice el GPIO como salida para el led.
  - Configura el SysTick
  - Escriba la función de atención a la interrupción

Nota 1: la inicialización del clock del procesador se hará con el ayudante del MCU Expresso y específicamente del SDK del procesador.

Nota 2: el CMSIS-core nos presenta una función para la configuración del SysTick. Activa de una vez la interrupción.

```
__STATIC_INLINE uint32_t SysTick_Config(uint32_t ticks)
```

Ejemplo - SysTick Timer

- Inicializacion del rgb.

```
#define LED_BLUE 21
#define MASK(x) (1UL << (x))

void init_rgb(void);

void init_rgb(void){
  SIM->SCGC5 |= SIM_SCGC5_PORTB_MASK; //| SIM_SCGC5_PORTC_MASK;
  PORTB->PCR[LED_BLUE] = PORT_PCR_MUX(1);
  PTB->PDDR |= MASK(LED_BLUE);
  PTB->PCOR |= MASK(LED_BLUE);
}

```

- ISR SysTick.

```
void SysTick_Handler(void){
  band++;
}

```

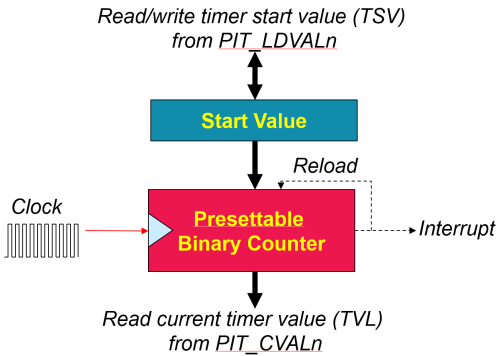
Ejemplo - SysTick Timer

- Main.

```
1  int main(void) {
2    /* Init board hardware. */
3    BOARD_InitBootClocks();
4    init_rgb();
5    /* Init SysTick to 120000 of ticks = 1 ms */
6    SysTick_Config(120000UL);
7    while(1) {
8      if(band>=1000){
9        PTB->PTOR |= MASK(LED_BLUE);
10       band = 0;
11     }
12   }
13   return 0;
14 }
15
```

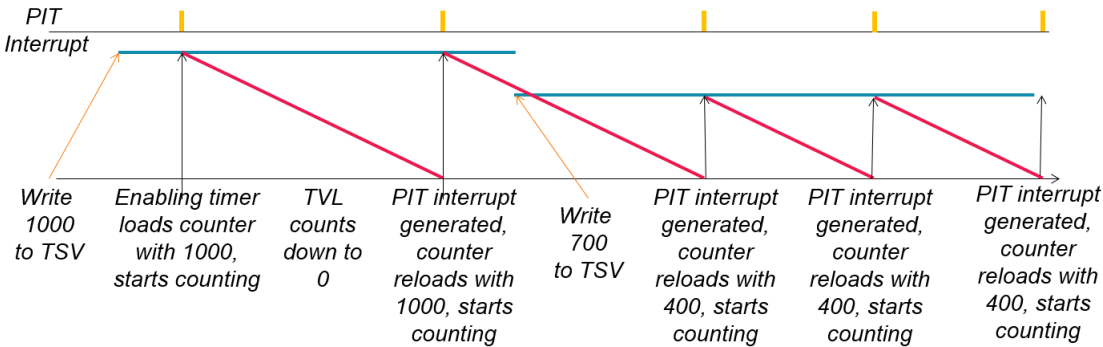
Limitaciones: solo cuenta con uno y de 24 bits. Decrementa.

Periodic Interrupt Timmer - PIT



- Genera una interrupción periodica usando contador de 32 bits
- Se carga el valor inicial en el registro LDVAL
- Contador decrementa con cada pulso del reloj
  - Fuente de reloj fija para el PIT - Clock del Bus.
- Cuando el temporizador llega a cero (CVAL)
  - Genera interrupción
  - Carga el temporizador de nuevo con el valor inicial

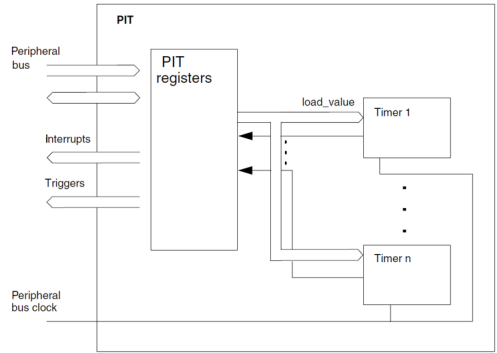
PIT Diagrama de tiempo



Nota: solo se actualiza el valor hasta que termina de contar completamente.

Configuración del PIT

- Habilitar el clock del periférico
  - SIMCGC6 PIT
- Registro de control del módulo (PIT->MCR)
  - MDIS - 0: habilitado, 1: deshabilitado
- FRZ - stop timmer in debug
  - 0: timmer corre en debug
  - 1: timmer parados
- Múltiples canales en el PIT. Para el k64 tiene 4 canales
- Puede hacer cadena de contadores para aumentar la capacidad.



Control de cada canal

- Interfaz CMSIS
  - Acceso general con la estructura PIT->MCR
  - Los canales son accedidos como vectores de estructuras: PIT->CHANNEL[n].LDVAL
- PIT->CHANNEL[n].LDVAL para cargar el valor
- PIT->CHANNEL[n].CVAL valor actual del contador
- PIT->CHANNEL[n].TCTRL registro de control
  - CHN: 0 como independiente, 1 como cadena el cual cambia por el timer n-1
  - TIE: habilita la interrupción. 0 no se habilita, 1 si.
  - TEN: habilita el conteo. 0 comienza el conteo, 1 no cuenta.
  - PIT_TFLGn: bandera del timer. 1 es porque el tiempo se ha alcanzado.

Configurando el PIT

- Habilitar el clock del PIT

```
1 SIM->SCGC6 |= SIM_SCGC6_PIT_MASK;
2
```

- Habilita el modulo, congela el timer mientras DEBUG

```
1 PIT->MCR &= ~PIT_MCR_MDIS_MASK;
2 PIT->MCR |= PIT_MCR_FRZ_MASK;
3
```

- Inicializa el PIT para contar en forma descendiente.

```
1 PIT->CHANNEL[channel].LDVAL = PIT_LDVAL_TSV(TimeValue); // 60 MHz
2
```

- No se hace cadena de timmers y no se habiita por el momento el PIT

```
1 PIT->CHANNEL[channel].TCTRL &= ~PIT_TCTRL_CHN_MASK & ~PIT_TCTRL_TEN_MASK;
2
```

Obteniendo el valor de carga

- Meta: generar una interrupcion cada T segundos
- LDVAL: redondeo($T * f_{count} - 1$)
  - -1 ya que el contador va hasta cero.
  - Redondear ya que es entero el registro.
- **Ejemplo**: Interrupción cada 137.41 ms
  - LDVAL $= 137.41\,\text{ms} * 24\,\text{MHz} - 1 = 3297839$
- **Ejemplo**: Interrupción con frecuencia de 91 Hz
  - LDVAL $= 1/91\,\text{Hz} * 24\,\text{MHz} - 1 = 263734$

Configurando las interrupciones del PIT

- Configure el PIT

```
1 PIT->CHANNEL[channel].TCTRL |= PIT_TCTRL_TIE_MASK;
2
```

- Configure el NVIC
  - Configure la prioridad
  - Limpie cualquier IRQ del PIT pendiente.
  - Habilite la interrupción

```
1 PIT->CHANNEL[channel].TCTRL |= PIT_TCTRL_TIE_MASK;
2 NVIC_SetPriority(PIT0_IRQn+channel, 64); // 0, 64, 128 or 192
3 NVIC_ClearPendingIRQ(PIT0_IRQn+channel);
4 NVIC_EnableIRQ(PIT0_IRQn+channel);
5
```

- Asegúrese que las interrupciones no estén enmascaradas globalmente.

```
1 __enable_irq();
2
```

Handler de la interrupción

- Una función de atención a interrupción por cada canal
- Nombre del CMSIS
  - `void PIT0_IRQHandler(void)`
  - `void PIT1_IRQHandler(void)`
  - `void PIT2_IRQHandler(void)`
  - `void PIT3_IRQHandler(void)`
- Actividades dentro del ISR
  - Determinar cual canal genero la interrupción.
    `if (PIT->CHANNEL[0].TFLG & PIT_TFLG_TIF_MASK)`
  - Limpiar la interrupción.
    `IT->CHANNEL[0].TFLG &= PIT_TFLG_TIF_MASK;`
  - Hacer la tarea del ISR

Iniciar y parar el timer

- Iniciar el timer

  ```
  PIT->CHANNEL[0].TCTRL |= PIT_TCTRL_TEN_MASK;
  ```

- Parar el timer

  ```
  PIT->CHANNEL[0].TCTRL &=  PIT_TCTRL_TEN_MASK;
  ```

Ejemplo PIT

Hacer un programa que genere una interrupción cada 1 s. Una vez se presione el pulsador, el led comienza a parpadear cada segundo. El led debe parpadear 10 veces y luego parar. El parpadeo inicia de nuevo cuando se presiona el pulsador.

Solución:

- Inicializar los clock
- Inicializar el pulsador
- Inicializar el rgb
- Inicializar el PIT
- Tener en cuenta las interrupciones tanto del pulsador como del PIT.

Ejemplo PIT - Solución

```
1  void Init_PIT(uint8_t channel, unsigned period_us, uint8_t gen_interrupts) {
2    // Enable clock to PIT module
3    SIM->SCGC6 |= SIM_SCGC6_PIT_MASK;
4
5    // Enable module, freeze timers in debug mode
6    PIT->MCR &= ~PIT_MCR_MDIS_MASK;
7    PIT->MCR |= PIT_MCR_FRZ_MASK;
8
9    // Initialize PIT to count down from argument
10   PIT->CHANNEL[channel].LDVAL = PIT_LDVAL_TSV((period_us*60)-1); // 60 MHz bus
11
12   // No chaining
13   PIT->CHANNEL[channel].TCTRL &= PIT_TCTRL_CHN_MASK;
14
15   if (gen_interrupts) {
16     // Generate interrupts
17     PIT->CHANNEL[channel].TCTRL |= PIT_TCTRL_TIE_MASK;
18     // Configure NVIC
19     NVIC_SetPriority(PIT0_IRQn+channel, 64); // 0, 64, 128 or 192
20     NVIC_ClearPendingIRQ(PIT0_IRQn+channel);
21     NVIC_EnableIRQ(PIT0_IRQn+channel);
22   }
23 }
```

Ejemplo PIT - Solución

```
1  void Start_PIT(uint8_t channel) {
2    // Enable counter
3    PIT->CHANNEL[channel].TCTRL |= PIT_TCTRL_TEN_MASK;
4  }
5
6  void Stop_PIT(uint8_t channel) {
7    // Disable counter
8    PIT->CHANNEL[channel].TCTRL &= ~PIT_TCTRL_TEN_MASK;
9  }
10
11 void PIT0_IRQHandler(void){
12   // check to see which channel triggered interrupt
13   if (PIT->CHANNEL[0].TFLG & PIT_TFLG_TIF_MASK) {
14     // clear status flag for timer channel 0
15     PIT->CHANNEL[0].TFLG &= PIT_TFLG_TIF_MASK;
16     bandPIT = 1;
17     PTE->PTOR |= MASK(LED_GREEN);
18   }
19 }
20
```

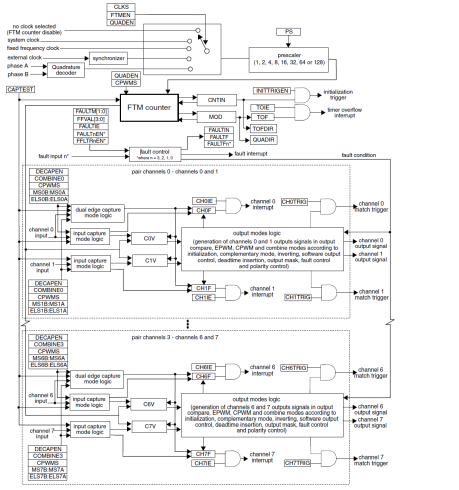
Laboratorio 10 %

Realizar el programa de las maquinas de estado para parpadear tanto led RGB como blanco y negro. Esta vez deberá realizar el delay con interrupción y no bloqueantes. Adicionalmente, los pulsadores tambien deberan tener interrupciones. Solo debe quedar el programa principal con dos tareas: la maquina de estados del RGB y la de Blanco y negro.

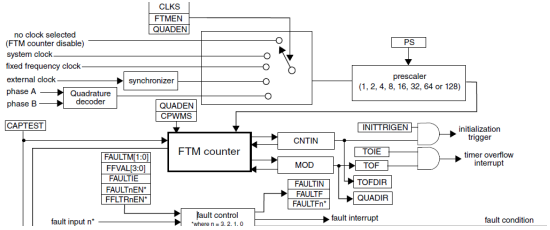
Adicionalmente, tendrá otra tercera tarea que se encarga de leer el valor de un potenciometro. El objetivo es que con dicho valor del potenciometro se cambie la intensidad de brillo de los leds.

FlexTimer FTM / PMW

- Core: contador
  - Diferentes opciones de reloj.
  - Se puede escalar el reloj por 1 hasta 128
  - Contador de 16-bit
    - Puede contar ascendente o ascendente/descendente
    - Se puede cargar el valor del contador
- Hasta 4 FTM con múltiples canales independientes (hasta 8)
  - 3 modos
    - Modo de captura: captura la señal de tiempo cuando la entrada cambia
    - Comparación de salida: cambia la señal de salida cuando el temporizador alcanza cierto valor
    - PWM: genera una señal por modulación de ancho de pulso.
  - Cada canal puede generar interrupcion, peticion DMA
  - Un pin I/O por canal: FTMx_CHn

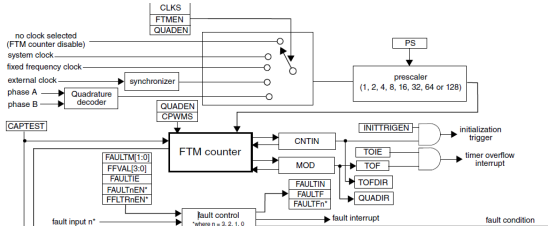


Configuración del FTM



- Fuente de reloj
  - CLKS: no clock, clock del sistema, clock de frecuencia fija o clock externo
- Prescaler
  - PS: divide el clock por 1, 2, 4, 8, 16, 32,64, 128
- Modo de conteo
  - CPWMS: conteo arriba (0) or arriba & abajo (1)
- Módulo de conteo
  - MOD: 16-bit
  - Timer overflow cuando pasa dicho valor
  - Conteo ascendente: 0, 1, ... MOD, 0/Overflow, 1, ...
  - Up/down: 0, 1 ... MOD, MOD-1/ Interrupt, MOD-2, ..., 1, 0, 1, 2 ...
- TOF: bandera que indica el temporizador a finalizado

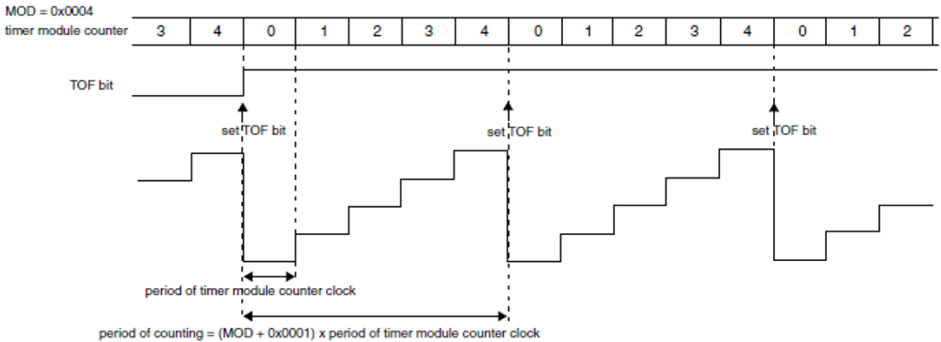
Configuración como contador básico



- Contar los evento externos aplicados al pin
  - Poner CLKS=11 para seleccionar contador externo
  - Poner PS=000 a no ser que se necesite división.
- La bandera de desbordamiento TOF se pone en 1 hasta recibir la cantidad de pulsos MOD
- Puede generar interrupción si TOIE está activada

2-0 PS	Prescaler Factor
000	1
001	2
010	4
011	8
100	16
101	32
110	64
111	128

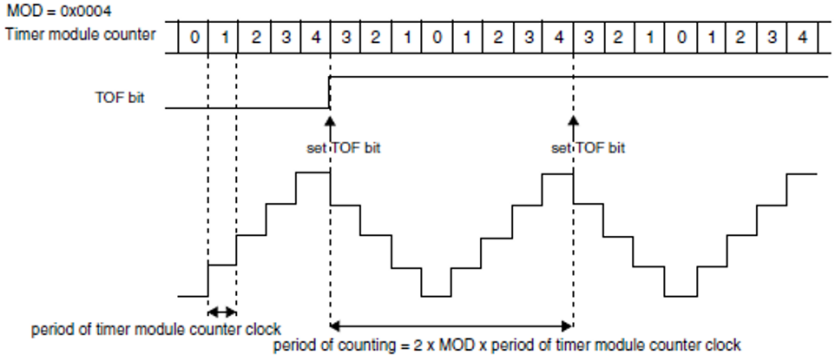
Modo de conteo - Conteo Ascendente



- Contador incrementa con cada clock
- Cuando el contador llega al MOD
  - Poner TOF en 1
  - Resetea el valor a cero.

- La frecuencia será: frecuencia del timer/(1+MOD)

Modo de conteo - Conteo Ascendente/Descendente



- Fase ascendente
  - Contador incrementa con cada clock
  - Cuando llega al MOD, pone uno en TOF, y pasa a modo descendente
- Conteo descendente
  - Contador decrementa con cada clock
  - Cuando alcanza a cero, se pasa a modo ascendente.
- La frecuencia de desbordamiento es: clock/(2*MOD)

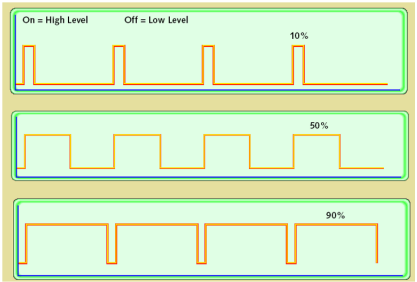
Configuración del canal y valor

- Configuración: FTMx_CnSC
  - CHF: ha ocurrido un evento (1) o no (0)
  - CHIE: habilita el canal para generar interrupción
  - MSB:MSA selección del modo
  - ELSB:ELSA selección por flanco o nivel
  - DMA
- Valor: FTMx_CnV
  - Valor de salida de 16-bit.
- El CMSIS agrupa estos dos campos es una estructura llamada CONTROLS para cada canal
  - `FTM0->CONTROLS[0].CnSC`

CPWMS	MSnB:A	ELSnB:A	Modo	Config
0	00	01	Input Capture	Captura Flanco de subida
		10		Captura Flanco de bajada
		11		Captura ambos flancos
0	01	01	Output Capture	Toggle ouput on match
		10		Clear ouput on match
		11		Set ouput on match
0	1X	10	Edge-Aligned PWM	High-true pulses (clear Output on match)
		X1		Low-true pulses (set Output on match)
1	XX	10	Center-Aligned PWM	High-true pulses (clear Output on match)
		X1		Low-true pulses (set Output on match)

Modulación por ancho de pulso

- Posibilita una señal digital mandar dos valores 0,1.
- Facil codificación: valor es una fracción del tiempo.
- La señal puede ser promediada para crear una señal analógica.
- PWM características
  - Modulación en frecuencia - cuantos pulsos ocurren en un segundo
  - Periodo
  - On-time - Cantidad de tiempo que el pulso esta activo
  - Duty-cycle - on-time/periodo
  - Configurar on-time para representar un valor análogo.



Modulación por ancho de pulso

- **Comunicaciones digitales**: es menos sensible al ruido que los métodos análogos.
  - PWM da una codificación digital para un valor análogo
  - Menos vulnerable al ruido
- **Amplificadores de potencia digitales**: son mas eficiente y menos costosos que los análogos.
  - Aplicaciones: control de velocidad deun motor, dimerizado de luces, conversión de potencia.
  - Carga responde lentos, señal promedio.

Ejemplo - PWM

Hacer que un led se dimerice cada tiempo yendo de total intensidad hasta apagarse. Volver a repetirse

Solución:

- Inicializar los clock del procesador.
- Inicializar el FTM
  - Activar el clock del puerto
  - Activar el clock del FTM
  - Poner el PCR en la alternativa de FTM
  - Poner el modulo de conteo
  - Seleccionar el preescaler
  - Seleccionar el modo de operación (PWN alineado con flanco)
  - poner el valor del duty-cycle
  - Activar el clock del PWM

Ejemplo - Solución

```
1  void Init_Green_LED_PWM(uint16_t period){
2    // Enable clock to port C
3    SIM->SCGC5 |= SIM_SCGC5_PORTC_MASK;;
4
5    // Blue FTM0_CH0, Mux Alt 4
6    PORTC->PCR[LED_GREEN] &= ~PORT_PCR_MUX_MASK;
7    PORTC->PCR[LED_GREEN] |= PORT_PCR_MUX(4);
8
9    // Configure TPM
10   SIM->SCGC6 |= SIM_SCGC6_FTM0_MASK;
11
12   //load the counter and mod
13   FTM0->MOD = period-1;
14
15   //set TPM count direction to up (CPWMS = 0), Disable TOF interrupt, No clock
        selected, divide by 2 prescaler
16   FTM0->SC =  FTM_SC_PS(2);
17
18   // Set channel 0 to edge-aligned low-true PWM
19   FTM0->CONTROLS[0].CnSC = FTM_CnSC_MSB_MASK | FTM_CnSC_ELSA_MASK;
20   // Set initial duty cycle
21   FTM0->CONTROLS[0].CnV = 0;
22   // Start TPM
23   FTM0->SC |= FTM_SC_CLKS(1);
24 }
25
```

Ejemplo - Solución

```
1  int main(void) {
2
3    BOARD_InitBootClocks();
4
5    uint16_t i=0;
6    volatile int32_t delay;
7    Init_Green_LED_PWM(PWM_PERIOD);
8
9    // Flash forever
10   while (1) {
11     for (i=0; i<PWM_PERIOD; i++) {
12       FTM0->CONTROLS[0].CnV = i;
13       for (delay=0; delay<1000; delay++)
14       ;
15     }
16     for (i=PWM_PERIOD-1; i>0; i--) {
17       FTM0->CONTROLS[0].CnV = i;
18       for (delay=0; delay<1000; delay++)
19       ;
20     }
21   }
22   return 0 ;
23 }
24
```

TEMPORIZADORES

GRACIAS