PRÁCTICA B INTERRUPCIÓN POR TIMER

CÓDIGO

```
#include <Arduino.h>

volatile int interruptCounter;
int totalInterruptCounter;

hw_timer_t * timer = NULL;
portMUX_TYPE timerMux = portMUX_INITIALIZER_UNLOCKED;

void IRAM_ATTR onTimer() {
  portENTER_CRITICAL_ISR(&timerMux);
  interruptCounter++;
  portEXIT_CRITICAL_ISR(&timerMux);

}

void setup() {

  Serial.begin(9600);

  timer = timerBegin(0, 80, true);
  timerAttachInterrupt(timer, &onTimer, true);
  timerAlarmWrite(timer, 1000000, true);
  timerAlarmEnable(timer);

}

void loop() {

  if (interruptCounter > 0) {

    portENTER_CRITICAL(&timerMux);
    interruptCounter--;
    portEXIT_CRITICAL(&timerMux);

    totalInterruptCounter++;

    Serial.print("An interrupt as occurred. Total number: ");
    Serial.println(totalInterruptCounter);

  }
}
```

FUNCIONAMENTO

Empezamos declarando el contador de interrupciones con **volatile**, lo que evitara que se elimine a causa de optimizaciones del compilador.

```
volatile int interruptCounter;
```

Declaramos también un contador para ver cuantas interrupciones han ocurrido desde el principio del programa, este no requiere **volatile**, ya que solo vamos a usarlo en el *loop* principal (*main loop*).

```
int totalInterruptCounter;
```

Para la configuración del timer que haremos más adelante, primero necesitaremos un puntero.

```
hw_timer_t * timer = NULL;
```

Y la última variable que tenemos que declarar es del tipo `portMUX_TYPE`, esta la usaremos para la sincronización entre el *main loop* y la ISR.

```
portMUX_TYPE timerMux = portMUX_INITIALIZER_UNLOCKED;
```

A continuación vamos a declarar el siguiente **void**:

```
void setup() {

  Serial.begin(9600);

  timer = timerBegin(0, 80, true);
  timerAttachInterrupt(timer, &onTimer, true);
  timerAlarmWrite(timer, 1000000, true);
  timerAlarmEnable(timer);

}
```

Dentro de este tenemos la inicialización de nuestro timer llamando a la función `timerBegin`. Esta función recibe como entrada el número del temporizador que queremos usar (en nuestro caso el 0), el

valor del prescaler (en nuestro caso 80) e indicamos si el contador cuenta hacia adelante (*true*) o si lo hace hacia atrás (*false*).

```
timer = timerBegin(0, 80, true);
```

Ahora usaremos la función `timerAttachInterrupt`, esta recibe como entrada un puntero al temporizador, la dirección a la función `onTimer` que más tarde especificaremos y sirve para manejar la interrupción y finalmente el valor *true* para especificar que la interrupción es de tipo *edge*.

```
timerAttachInterrupt(timer, &onTimer, true);
```

A continuación usaremos la función `timerAlarmWrite`, esta también recibe como entrada tres valores: Como primera entrada recibe el puntero al temporizador, como segunda el valor del contador en el que se tiene que generar la interrupción y finalmente un indicador de si el temporizador se ha de recargar automáticamente al generar la interrupción.

```
timerAlarmWrite(timer, 1000000, true);
```

Finalmente llamamos a la función `timerAlarmEnable`, en esta pasamos como entrada nuestra variable de temporizador.

```
timerAlarmEnable(timer);
```

El *main loop* completo es el siguiente:

```
void loop() {

  if (interruptCounter > 0) {

    portENTER_CRITICAL(&timerMux);
    interruptCounter--;
    portEXIT_CRITICAL(&timerMux);

    totalInterruptCounter++;

    Serial.print("An interrupt as occurred. Total number: ");
    Serial.println(totalInterruptCounter);

  }
}
```

Este programa básicamente consiste en incrementar el contador con el número total de interrupciones (`totalInterruptCounter`) i imprimirlo al puerto serie.

Para acabar explicaremos como funciona la función ISR (*interrupt service routine*). Esta consistirá en incrementar el contador de interrupciones que indicará al bucle principal que se ha producido una interrupción. Esto se produce dentro de una sección crítica, declarada con `portENTER_CRITICAL_ISR` y `portEXIT_CRITICAL_ISR` . El código completo del ISR queda:

```
void IRAM_ATTR onTimer() {
  portENTER_CRITICAL_ISR(&timerMux);
  interruptCounter++;
  portEXIT_CRITICAL_ISR(&timerMux);

}
```

IMPRESIÓN SERIE

Este programa va a imprimir por pantalla el número de veces que hay una interrupción. Por lo tanto imprimirá algo como:

```
An intrrupt as occurred. Total number: 1
An intrrupt as occurred. Total number: 2
An intrrupt as occurred. Total number: 3
An intrrupt as occurred. Total number: 4
An intrrupt as occurred. Total number: 5
An intrrupt as occurred. Total number: 6
An intrrupt as occurred. Total number: 7
...
```

Y así indefinidamente hasta que no se pare manualmente.