

# CONTADOR UP/DOWN DE 000 A 999

Stellaris Lunchpad LM4F120

## Descripción breve

Contador que va de 000 a 999, el cual funciona con 3 botones el primero para aumentar el contador, el segundo para disminuir el contador y el tercero para iniciar, pausar y reiniciar el contador, desplegando los datos en displays de 7 segmentos.

Juan Miguel Vargas Sánchez  
Jmvarsan@gmail.com

```

Timer0IntHandler,           // Timer 0 subtimer A
IntDefaultHandler,         // Timer 0 subtimer B
Timer1IntHandler,          // Timer 1 subtimer A
    IntDefaultHandler,      // Timer 1 subtimer B

```

## Contador UP/DOWN de 000 a 999 con salida en 7 segmentos

### Objetivo






Con este trabajo se muestra el funcionamiento de las interrupciones externas y el uso de los Timers de la tarjeta LM4F120 así como el uso de los periféricos de entrada y salida mediante un contador up/down de 000 a 999 en displays de 7 segmentos.

### Resumen

Se acondicionara un LM4F120 (StellarisLaunchpad) con 3 displays de 7 segmentos en los cuales se desplegara un contador que valla de 000 hasta 999, dicho contador contara con 3 interruptores, uno que aumente el contador, otro que disminuya el contador y un tercero para iniciara/pausara/reiniciara el sistema.

### Desarrollo

Para el desarrollo de este sistema se utilizaron los siguientes componentes

Stellaris Launchpad LM4F120H5QR	
3 displays de 7segmentos anodo común	
7 resistores de 220 ohms	
1 resistor de 4700 ohms	
1 push button	

1 CI decodificador BCD modelo SN47LS47N	
--	--

## Conexiones

La tarjeta Stellaris Launchpad conecta al resto del sistema por medio de los puertos B

(PB0, PB1, PB2, PB3) y el puerto D (PD0, PD1, PD2, PD3), el puerto PB0 es la entrada para recibir la señal del tercer botón (inicio/pausa/reinicio), los puertos PB1, PB2 y PB3 son los habilitadores de cada uno de los displays de 7 segmentos en el sistema con esto se logra ahorrar componentes y salidas, esto da a la tarjeta la posibilidad de utilizar los puertos restantes en otras aplicaciones.

El puerto D son la salida de los datos que se desplegarán en los displays de 7 segmentos y la alimentación y tierra son tomados igual de la tarjeta para la alimentación del resto del sistema.

Por otro lado el resto del hardware que compone el sistema es el push button externo conectado en un extremo a tierra y el otro a un resistor de 4.7 kohm esto para evitar el efecto de rebotes.

El circuito integrado SN74LS47N el cual recibe los datos del puerto D de la tarjeta, este componente nos sirve para hacer la decodificación de los datos a los displays de 7 segmentos, dichos datos son enviados a través de un bus de 4 bits los cuales son leídos en los pines 7, 1, 2 y 6 (A, B, C, D) respectivamente, dando una salida directa a cada pin de los displays el sus pines 13, 12, 11, 10, 9, 15, 14 (a, b, c, d, e, f, g), estas salidas son conectadas a un resistor de 220 ohms para cada pin y estos resistores al pin correspondiente de cada salida en el display respectivamente.

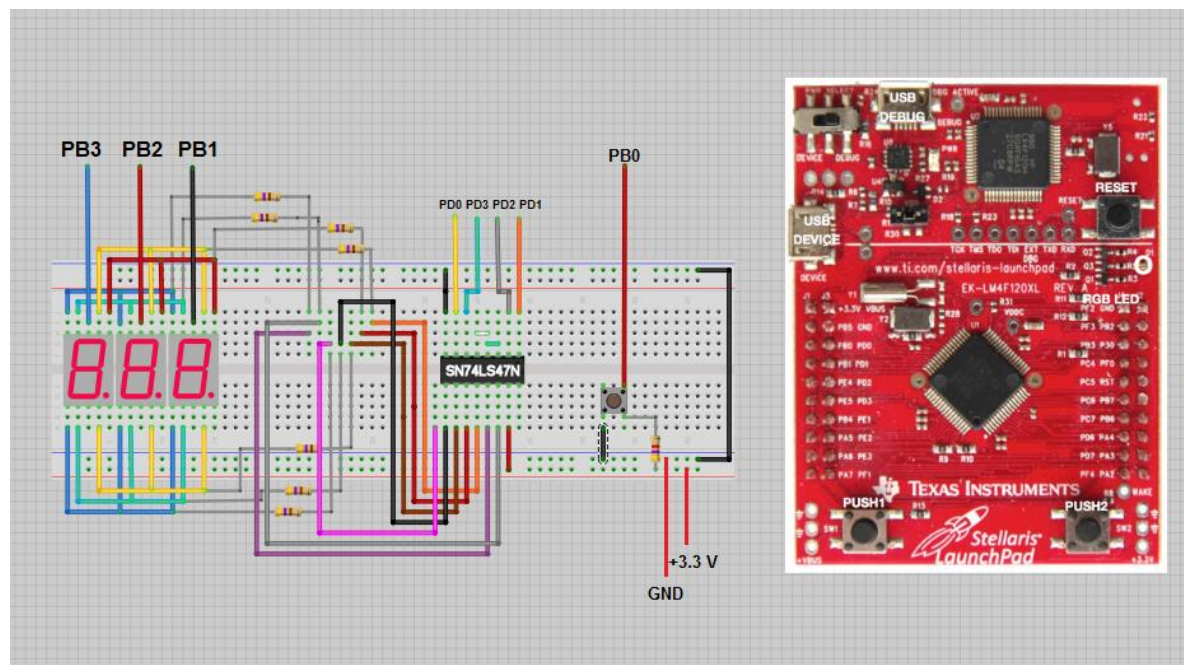
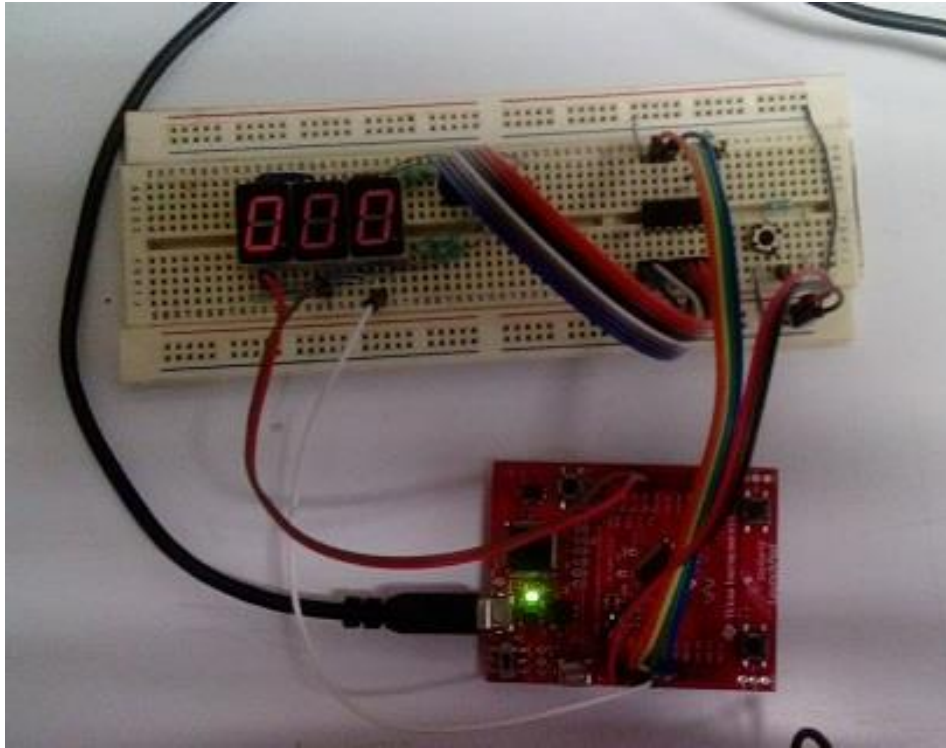


Figura 1. Sistema contador up/down de 000 a 999

El funcionamiento en general es muy simple, el botón de la tarjeta PUSH1 sirve para aumentar el conteo, el PUSH2 para disminuir el conteo y el botón externo tiene la función de iniciar, pausar y reiniciar el sistema con la primera pulsación inicia el sistema, una segunda pulsación lo pausa y si se mantiene presionado durante 3 segundos reinicia a ceros el sistema, en la figura 2 se muestra la parte del hardware del sistema real.



*Figura 2. Hardware del sistema del contador up/down.*

A continuación se muestra el código encargado de hacer funcionar el contador up/down

Primeramente se definen las librerías a utilizar, y las variables a utilizar durante la ejecución del sistema, cont es un contador que lleva el conteo general desde 0 a 999, unidad, decena y centena son contadores que solo cuentan de 0 a 9 y estas mismas son tomadas para como salidas para ser desplegadas en los displays de 7 segmentos.

band es la bandera la cual nos indica que botón ha sido presionado tomando valores de 1, 2 o 3 dependiendo que botón haya sido presionado (PUSH1, PUSH2 o el botón externo), band1 es la salida de los habilitadores dicha bandera es inicializada en 0 y al habilitar la interrupción del Timer1 esta cambia su valor entre 1, 2 y 3 para así ir habilitando cada display por separado durante un intervalo de tiempo de 10ms aproximadamente y por último la band3 es la nos indica las veces que ha sido presionado el botón externo esto para iniciar o pausar el sistema o si ha sido presionado durante 3 segundos esto para reiniciar el sistema.

```
#include "utils/ustdlib.h"
#include "inc/hw_ints.h"
#include "inc/hw_types.h"
```

```

#include "inc/hw_memmap.h"
#include "driverlib/sysctl.h"
#include "driverlib/pin_map.h"
#include "driverlib/debug.h"
#include "driverlib/gpio.h"
#include "driverlib/timer.h"
#include "driverlib/interrupt.h"
#include <inc/hw_gpio.h>

```

```

volatile unsigned int cont = 0;
volatile unsigned int unidad = 0;
volatile unsigned int decena = 0;
volatile unsigned int centena = 0;

```

```

volatile unsigned int band = 0;
volatile unsigned int band1 = 0;
volatile unsigned int band3 = 0;

```

Posteriormente se muestra la parte del código principal del sistema en el cual se configuran los periféricos a utilizar, primeramente se debe definir la frecuencia del reloj del sistema con la que se va trabajar en este caso se trabajó con un frecuencia de 40 MHz, después se debe habilitar los puertos a utilizar con la instrucción “**SysCtlPeripheralEnable**(SYSCTL\_PERIPH\_GPIOX);” donde X es la letra del puerto a habilitar. Con la instrucción “**GPIOPinTypeGPIOInput**(GPIO\_PORTX\_BASE, GPIO\_PIN\_Y);” define entrada un puerto donde X es la letra del puerto y Y el número del pin, para configurarlos como salida solo se cambia Input por Output. Cabe mencionar que para utilizar el PUSH2 como entrada se debe desbloquear pues principalmente funciona por default para sacar de hibernación al sistema, esto se hace con el siguiente fragmento del código.

```

HWREG(GPIO_PORTF_BASE+GPIO_O_LOCK) = GPIO_LOCK_KEY_DD;
HWREG(GPIO_PORTF_BASE+GPIO_O_CR) = 0x01;

```

Ahora con la instrucción “**GPIOPadConfigSet**(GPIO\_PORTX\_BASE, GPIO\_PIN\_Y, GPIO\_STRENGTH\_4MA, GPIO\_PIN\_TYPE\_STD\_WPU);” se configura como van a funcionar los botones de entrada donde X es la letra del puerto y Y el numero del pin. Con “GPIO\_PIN\_TYPE\_STD\_WPU” se indica que los botones funcionaran como pull up.

```

int main(void)
{
    unsigned long ulPeriod;
    unsigned long ulPeriod2;

    /* El reloj del sistema está corriendo utilizando un oscilador de 16
    Mhz conectado al oscilador principal del microcontrolador
    * Este genera una señal de reloj interna de 400Mhz utilizando un PLL
    * La señal es preescalada por el sistema por 2
    * En la siguiente configuración se está preescalando el reloj del
    sistema por 5 para que el oscilador corra a 40 MHz

```

```

    * La frecuencia del reloj del sistema debe ser menor o igual a 80 Mhz
    */
    SysCtlClockSet(SYSCTL_SYSDIV_5|SYSCTL_USE_PLL|SYSCTL_XTAL_16MHZ|SYSCTL_
OSC_MAIN);

    // Se habilitan los periféricos de los puertos F, B y D
    SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOF);
    SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOB);
    SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOD);

    // Se configuran los puertos como entrada
    GPIOPinTypeGPIOInput(GPIO_PORTF_BASE, GPIO_PIN_0|GPIO_PIN_4);
    GPIOPinTypeGPIOInput(GPIO_PORTB_BASE, GPIO_PIN_0);

    // Se configuran los puertos de salida
    GPIOPinTypeGPIOOutput(GPIO_PORTF_BASE,
GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3);
    GPIOPinTypeGPIOOutput(GPIO_PORTB_BASE,
GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3);
    GPIOPinTypeGPIOOutput(GPIO_PORTD_BASE,
GPIO_PIN_0|GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3);

    // Se libera el botón PUSH2 para que trabaje como un periférico normal
de entrada
    HWREG(GPIO_PORTF_BASE+GPIO_O_LOCK) = GPIO_LOCK_KEY_DD;
    HWREG(GPIO_PORTF_BASE+GPIO_O_CR) = 0x01;

    // Se configuran los botones PUSH1,PUSH2 y botón externo configurados
como pull up.
    GPIOPadConfigSet(GPIO_PORTF_BASE, GPIO_PIN_0|GPIO_PIN_4, GPIO_STRENGTH_4MA,
GPIO_PIN_TYPE_STD_WPU);
    GPIOPadConfigSet(GPIO_PORTB_BASE, GPIO_PIN_0, GPIO_STRENGTH_4MA,
GPIO_PIN_TYPE_STD_WPU);

```

En el siguiente fragmento de código se configuran los Timers que se activaran al hacer la interrupciones, el Timer0 se encarga de hacer los conteos up, down e iniciar el contador y el Timer1 nos sirve para ir cambiando los valores de los habilitadores de los displays de 7 segmentos, para habilitar los periféricos de los Timers se utiliza la siguiente instrucción

“**SysCtlPeripheralEnable**(SYSCTL\_PERIPH\_TIMERX);” donde X indica el Timer a habilitar después se hace la configuración con “**TimerConfigure**(TIMER0\_BASE, TIMER\_CFG\_32\_BIT\_PER);” configurando el Timer con 32 bits por periodo.

Las variables ulPeriod y ulPeriod2 toman el valor al cual define el rango de tiempo al cual se ejecutara cada interrupción, en este caso ulPeriod interrumpirá cada 1 segundo y ulPeriod2 cada 10 ms para el Timer0 y Timer1 respectivamente, estos valores son indicados a los timers con la instrucción “**TimerLoadSet**(TIMER0\_BASE, TIMER\_A, ulPeriod -1);” por último se habilitan las interrupciones con “**IntEnable**(INT\_TIMER0A);”, se habilitan los Timers con “**TimerIntEnable**(TIMER0\_BASE, TIMER\_TIMA\_TIMEOUT);” y con “**IntMasterEnable**();” se hace la habilitación general de interrupciones.

```

// Se habilitan los periféricos del Timer0 y Timer1

```



```

SysCtlPeripheralEnable(SYSCTL_PERIPH_TIMER0);
SysCtlPeripheralEnable(SYSCTL_PERIPH_TIMER1);

// Se configuran los Timer0 y Timer1 para que trabajen con 32 bits por
periodo
TimerConfigure(TIMER0_BASE, TIMER_CFG_32_BIT_PER);
TimerConfigure(TIMER1_BASE, TIMER_CFG_32_BIT_PER);

// La variable ulPeriod toma un valor del reloj del sistema
ulPeriod = (SysCtlClockGet() / 1) / 1;
ulPeriod2 = (SysCtlClockGet() / 150) / 1;

// Se establece el tiempo cada cuanto se activaran las interrupciones

// Timer0 se activara cada 1 segundo
TimerLoadSet(TIMER0_BASE, TIMER_A, ulPeriod -1);
// Timer1 se activara cada 10 milisegundos
TimerLoadSet(TIMER1_BASE, TIMER_A, ulPeriod2 -1);

//Habilita las interrupciones del sistema
IntEnable(INT_TIMER0A);
IntEnable(INT_TIMER1A);
TimerIntEnable(TIMER0_BASE, TIMER_TIMA_TIMEOUT);
TimerIntEnable(TIMER1_BASE, TIMER_TIMA_TIMEOUT);
// Habilita el control maestro de interrupciones
IntMasterEnable();

```

Ahora en el loop infinito es donde se leerán las señales de entrada que vienen de los botones con la instrucción “**GPIOPinRead**(GPIO\_PORTF\_BASE, GPIO\_PIN\_4) == 0” indicando puerto y numero de pin que se está leyendo, cuando ingresa en alguna instrucción entonces se enciende el led rgb de la tarjeta rojo para cuando aumenta el contador, azul para cuando disminuye el contador y verde cuando se le da inicio, pausa o reinicio al contador, esto se hace escribiendo en el puerto F e indicando que led encenderá y el valor que representa el color de cada led con la instrucción “**GPIOPinWrite**(GPIO\_PORTF\_BASE, GPIO\_PIN\_1, 2);” cabe mencionar que 2 es para led rojo, 4 para el led azul y 8 para el led verde.

Posteriormente se habilitan las interrupciones de los Timers y se le asigna un valor a las banderas band y band3 las cuales son utilizadas en posteriormente.

```

while(1)
{
    // Si el botón PUSH1 de la tarjeta es presionado entonces ingresa
a la siguiente instrucción
    if(GPIOPinRead(GPIO_PORTF_BASE, GPIO_PIN_4) == 0){
        // El led rgb de la tarjeta enciende de color rojo como
señal de que se presionó el botón de aumento
        GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_1, 2);
        // Se inician las interrupciones del Timer0 y el Timer1
        TimerEnable(TIMER0_BASE, TIMER_A);
        TimerEnable(TIMER1_BASE, TIMER_A);
        band = 1;
    }
}

```

```

        // Si el botón PUSH2 de la tarjeta es presionado entonces ingresa
a la siguiente instrucción
        else if(GPIOPinRead(GPIO_PORTF_BASE, GPIO_PIN_0) == 0){
            // El led rgb de la tarjeta enciende de color azul como
señal de que se presionó el botón de disminución
            GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_2, 4);
            // Se inician las interrupciones del Timer0 y el Timer1
            TimerEnable(TIMER0_BASE, TIMER_A);
            TimerEnable(TIMER1_BASE, TIMER_A);
            band = 2;
        }
        // Si el botón externo es presionado entonces ingresa a la
siguiente instrucción
        else if(GPIOPinRead(GPIO_PORTB_BASE, GPIO_PIN_0) == 0){
            // El led rgb de la tarjeta enciende de color verde como
señal de que se presionó el botón de disminución
            GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_3, 8);
            // Se inician las interrupciones del Timer0 y el Timer1
            TimerEnable(TIMER0_BASE, TIMER_A);
            TimerEnable(TIMER1_BASE, TIMER_A);
            band = 3;
            // La bandera band3 cambia de valor dependiendo de los
pulsos que reciba y el tiempo
            // que se mantenga presionado el botón externo (1 pulso =
inicio, 2 pulsos = pausa y presionado por 3 seg = reinicio)
            band3++;
        }
    }
}

```

Cuando se hace la habilitación de la interrupción del Timer0 en el código anterior se ejecuta el siguiente código cada segundo, primeramente se limpia el Timer con `“TimerIntClear(TIMER0_BASE, TIMER_TIMA_TIMEOUT);”` y se apaga el led rgb si es que esta encendido `“GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3, 0);”`.

El funcionamiento de subir, bajar e iniciar el contador es muy simple, el código aún se puede mejorar simplificando estas funciones, cuando la bandera band = 1 entonces el contador aumenta, si band = 2 disminuye en una unidad y si band = 3 entonces se inicia el conteo. En el tercer botón al pulsarlo la primera vez la bandera band3 toma un valor entre 0 y 70000 ciclos de reloj cuando se presiona por segunda vez esta bandera toma un valor entre 70000 y 140000 ciclos de reloj con esto sabemos que se ha presionado una segunda vez y es cuando se pausa el conteo deteniendo la interrupción del Timer0 entonces se reinicializa la bandera band3 si el botón 3 es presionado durante 3 segundo la bandera band3 toma un valor mayor a 600000 entonces con esto le indicamos al sistema que se reinicie y se reinician a 0 todas las variables del sistema.

```

void Timer0IntHandler(void)
{
    // Limpia la interrupción del Timer0
    TimerIntClear(TIMER0_BASE, TIMER_TIMA_TIMEOUT);
    // Lee el estado actual de los pines GPIO y los regresa al estado
inicial
}

```



```

GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3, 0);

// Si la bandera band = 1 entonces el contador aumenta 1 unidad
if(band == 1){
    if(cont != 999){
        if(unidad == 9)
        {
            if(decena == 9){
                centena++;
                decena = 0;
            }
            else{
                decena++;
                unidad = 0;
            }
            unidad = 0;
        }
        else
        {
            unidad++;
        }
        cont++;
    }
    else{
        cont = 0;
        unidad = 0;
        decena = 0;
        centena = 0;
    }
    band = 0;
}
// Si la bandera band = 2 entonces el contador disminuye 1 unidad
else if(band == 2){
    if(cont != 0){
        if(unidad != 0)
        {
            unidad --;
        }
        else
        {
            if(decena != 0){
                decena --;
            }
            else{
                centena--;
                decena = 9;
            }
            unidad = 9;
        }
        cont--;
    }
    else{
        unidad = 9;
        decena = 9;
        centena = 9;
    }
}

```

```

        cont = 999;
    }
    band = 0;
}
// Si la bandera band = 3 entonces el contador inicia/pausa/reinicio
else if(band == 3){
    // Si la band3 es menor a 70000 y menor a 140000 entonces el
    contador se pausa (dos pulsos del botón externo)
    if(band3 > 70000 && band3 < 140000){
        TimerDisable(TIMERO_BASE, TIMER_A);
        band3 = 0;
    }
    // Si la bandera band3 es mayor a band3 entonces el sistema
    reinicializa sus contadores y banderas a cero
    // (mantener presionado el botón externo durante 3 segundos
    aproximadamente)
    else if(band3 > 600000){
        cont = 0;
        unidad = 0;
        decena = 0;
        centena = 0;
        band = 0;
        band1 = 0;
        band3 = 0;
    }
    // Si no se cumplen ni una de las condiciones anteriores el
    contador inicia el conteo
    else{
        band = 0;
        GPIOPinWrite(GPIO_PORTF_BASE,
GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3, 0);

        if(cont != 999){
            if(unidad == 9)
            {
                if(decena == 9){
                    GPIOPinWrite(GPIO_PORTF_BASE,
GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3, 8);
                    centena++;
                    decena = 0;
                    unidad = 0;
                }
                else{
                    GPIOPinWrite(GPIO_PORTF_BASE,
GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3, 4);
                    decena++;
                    unidad = 0;
                }
            }
            else
            {
                GPIOPinWrite(GPIO_PORTF_BASE,
GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3, 2);
                unidad++;
            }
        }
    }
}

```

```

        cont++;
    }
    else{
        GPIOWrite(GPIO_PORTF_BASE,
GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3, 0);
        cont = 0;
        unidad = 0;
        decena = 0;
        centena = 0;
    }
    band = 3;
}
}
else
    GPIOWrite(GPIO_PORTF_BASE, GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3,
0);
}

```

Por último se tiene la interrupción del Timer1 la cual tiene la tarea de ir habilitando los displays por separado cada 10 ms, primeramente se limpia la interrupción con “`TimerIntClear(TIMER1_BASE, TIMER_TIMA_TIMEOUT);`” y se va haciendo el cambio de valor a la variable band1 cada 10 ms y escribiendo a la salida del puerto B un 2 para habilitar el primer display un 4 para el segundo y un 8 para el tercer display, simultáneamente en el puerto D se van asignando los valores de las variables unidad, decena y centena respectivamente.

```

void Timer1IntHandler(void)
{
    // Limpia la interrupción del Timer1
    TimerIntClear(TIMER1_BASE, TIMER_TIMA_TIMEOUT);

    if(band1 == 0){
        // Se inicializa la bandera que controla los habilitadores de los
displays de 7 segmentos
        band1 = 1;
    }
    else if(band1 == 1){
        // Habilita el primer display de 7 segmentos
        GPIOWrite(GPIO_PORTB_BASE, GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3,
2);
        // Los pines del PIND toman el valor de la variable unidad y es
mostrado en el display de 7 segmentos
        GPIOWrite(GPIO_PORTD_BASE,
GPIO_PIN_0|GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3, unidad);
        // Cambia el valor de la bandera para habilitar el segundo
display de 7 segmentos
        band1 = 2;
    }
    else if(band1 == 2){
        // Habilita el segundo display de 7 segmentos
        GPIOWrite(GPIO_PORTB_BASE, GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3,
4);
        // Los pines del PIND toman el valor de la variable decena y es
mostrado en el display de 7 segmentos
    }
}

```

```

        GPIOPinWrite(GPIO_PORTD_BASE,
GPIO_PIN_0|GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3, decena);
        // Cambia el valor de la bandera para habilitar el tercer display
de 7 segmentos
        band1 = 3;
    }
    else{
        // Habilita el tercer display de 7 segmentos
        GPIOPinWrite(GPIO_PORTB_BASE, GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3,
8);
        // Los pines del PIND toman el valor de la variable centena y es
mostrado en el display de 7 segmentos
        GPIOPinWrite(GPIO_PORTD_BASE,
GPIO_PIN_0|GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3, centena);
        // regresa al valor inicial a la bandera band1
        band1 = 0;
    }
}

```

Cabe mencionar que para utilizar las interrupciones se debe modificar las siguientes líneas en el archivo lm4f120h5qr\_startup\_ccs.c el cual viene junto con el proyecto.

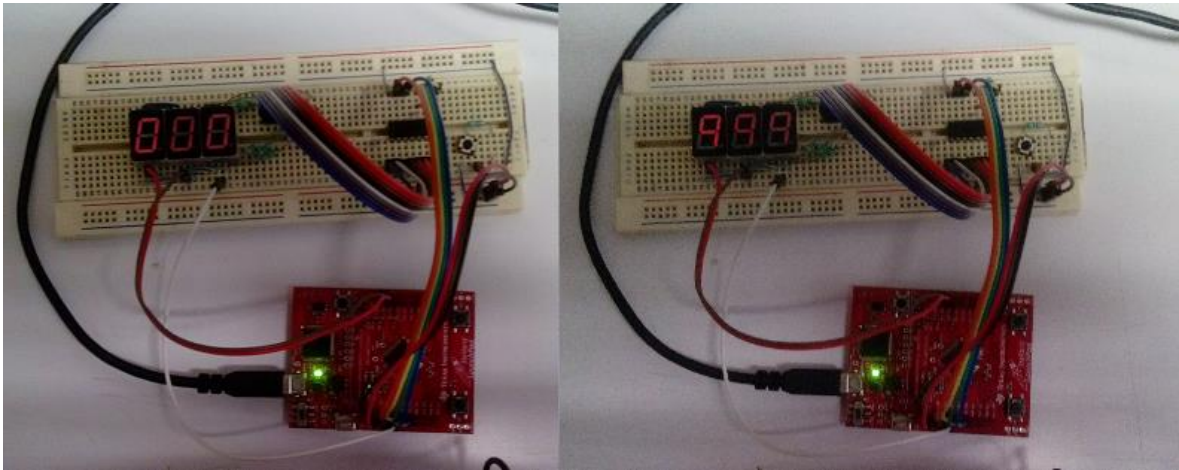
```

extern void Timer0IntHandler(void);
extern void Timer1IntHandler(void);
.
.
.

Timer0IntHandler,           // Timer 0 subtimer A
IntDefaultHandler,          // Timer 0 subtimer B
Timer1IntHandler,           // Timer 1 subtimer A
IntDefaultHandler,          // Timer 1 subtimer B

```

El resultado final de la conjunción del software y hardware es la mostrada en la figura 3, mostrando el valor inicial y final que pueden desplegar los displays de 7 segmentos, el proyecto y código del sistema también está disponible [aquí](#) o también desde [Dropbox](#).



*Figura 3. Sistema de contador up/down en funcionamiento.*

## Conclusiones

Con este sistema se demuestra varios de los recursos básicos de la tarjeta Stellaris Launchpad LM4F120, teniendo en cuenta que las configuraciones usadas en este trabajo pueden variar dependiendo de las necesidades del usuario, dicha tarjeta tiene un amplio número de recursos y configuraciones para estos, este es solo un poco de lo mucho que se puede hacer con esta tarjeta.

## Bibliografía

- [Getting Started with the Stellaris® EK-LM4F120XL LaunchPad Workshop](#)
- [Stellaris® Peripheral Driver Library USER'S GUIDE](#)
- <http://codeforfree.weebly.com/tutorial-2--using-button-switch.html>