

Apellidos:.....

Nombre:.....

P1	P2	P3

Duración 4:00 horas

- 1.- **[3 Puntos]** Analice el siguiente código e indique qué hace. Proponga un código alternativo que haga lo mismo, pero de forma más eficiente. Señale si sigue o no el convenio de llamada de C. En caso afirmativo, muestre el prototipo. En caso negativo, enumere los parámetros de entrada/salida, ubicación y tipo. Calcule el número de ciclos de ejecución, así como el tiempo total para $f_{MCLK}=1\text{MHz}$. Ensamble manualmente el programa y dé el código máquina en hexadecimal (excepto las instrucciones repetidas, que están en negrita). NOTA: la etiqueta C se refiere al bit de carry y está definida como BIT0.

		Emulación	Ciclos	Código máquina

vuas	mov.w r12, r13	;		
	mov.w #0x7FFF, r12	;		
vuasb1	rra.w r13	;		
	jc vuas2	;		
	rrc.w r12	;		
	jc vuasb1	;		
	jmp vuasfin	;		
vuasb2	rra.w r13	;		
	xor.w #C, sr	;		
vuas2	rrc.w r12	;		
	jc vuasb2	;		
vuasfin	ret	;		

- 2.- **[3 Puntos]** Realice en ensamblador del MSP430 las funciones `colaInserta` y `colaExtrae` que siguen la convención de llamada tipo C y atienden a los siguientes prototipos:

```
int colaInserta (cola_t *cola, uint8_t c);
int colaExtrae (cola_t *cola);
```

donde `cola` es el puntero a una cola y `c` es un elemento a insertar en la misma. `colaInserta` devuelve 0 si todo fue bien y -1 en caso de error. `colaExtrae` devuelve el elemento extraído si todo fue bien y -1 si la cola estaba vacía. El tipo `cola_t` se define como sigue:

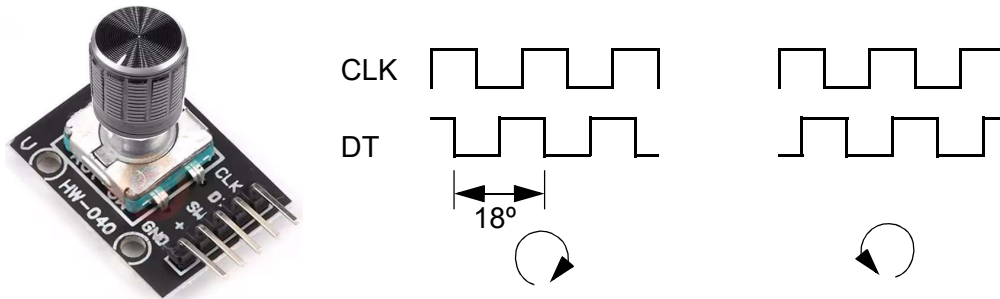
```
typedef struct
{
    uint8_t *pini;           //Puntero a cabeza
    uint8_t *pfin;          //Puntero a cola
    uint8_t cola[TAMCOLA];   //La cola en sí
} cola_t;
```

Suponga que las colas se crean vacías y se inicializan de forma que `pini=pfin=cola[0]` para que ambos punteros apunten al primer elemento de la cola.

- Explique con palabras el algoritmo que usará para resolver el problema.
- Escriba `colaInserta`.
- Escriba `colaExtrae`.

- 3.- **[4 Puntos]** El KY-040 es un encoder diferencial de 20 pulsos por vuelta. Se trata de un dispositivo con un eje accionable que proporciona 2 salidas indicando la dirección y velocidad de giro (CLK y DT). Además, dispone de un pulsador asociado al eje Z (salida SW) y dos

terminales de alimentación (+ y GND).



Es fácil determinar sentido/velocidad de giro (vea la figura). Así, se gira sentido horario si, al detectar un flanco en CLK, ambas salidas son iguales (CLK=DT), mientras que si son complementarias, el sentido es el contrario al de las agujas del reloj. La velocidad de giro se puede medir sabiendo que un ciclo completo de CLK o DT se produce después de girar 18° el eje (20 ciclos por vuelta).

Se ha conectado un KY-040 al Launchpad del MSP430: CLK en P1.1, DT en P1.2 y SW en P1.3, además de los leds rojo (P1.0) y verde (P9.7) habituales.

Realice un programa en ensamblador del MSP430 usando programación por interrupciones y con el mínimo consumo posible que controle el brillo de ambos leds con la ayuda del encoder y los temporizadores. Inicialmente los dos leds están apagados y el led activo es el rojo. Girando el encoder en sentido horario es posible aumentar el brillo del led activo. Se baja el brillo girando en sentido antihorario. Si la velocidad de giro es menor a 60 rpm, cada pulso del encoder cambiará un 1% el brillo. En cambio, si la velocidad es mayor, cada pulso cambiará un 5%. Cada vez que se pulse el eje hacia dentro, se cambiará el led activo.

- La resolución del brillo es de un 1%. Use una frecuencia mínima de 64Hz para evitar ver el parpadeo de los leds.
- Suponga que la función primaria de P1.0 es TA0.1 y de P9.7 es TA0.2. La función secundaria de P1.1 es TA1.0 (CCI1A).
- CLK y DT tienen resistencias de *pullup* a Vcc, pero no SW.
- Suponga SW libre de rebotes.
- Considere el perro guardián desactivado, los puertos desbloqueados, ACLK funcionando a 32768Hz y la pila inicializada.