

Evaluación del Laboratorio Número 4

Sistemas con Microprocesadores y Microcontroladores

20/11/2024

Instrucciones

- La presente evaluación es individual y podrá utilizar todo el material escrito del que disponga.
- La misma consta de 1 ejercicio en total y la duración será de 90 minutos.
- En cada ejercicio se indica el tiempo estimado para resolverlo y el puntaje que otorga el mismo.
- No se permitirá intercambiar ni prestar ningún tipo de elemento entre los estudiantes.
- Cada ejercicio funcionando deberá ser aprobado por un docente para que se compute como válido antes de comenzar con el siguiente ejercicio.
- El código de los ejercicios aprobados por el docente debe ser enviados por correo electrónico a las direcciones *mgiori@herrera.unt.edu.ar* y *ltek@herrera.unt.edu.ar* antes de finalizar la evaluación.

Ejercicios para resolver

1) 90 Minutos - 20, 25, 25, 30 Puntos

Se desea mostrar en el punto de los display distintas intensidades que puede alcanzar el LED y mostrar en el display el valor:

- a) Cuando se presione el **botón 1**, comience el incremento en pasos de a 10 % de la intensidad del punto del display entre 0.10 y 0.90. Cuando alcance la intensidad máxima deberá volver a la intensidad mínima y continuar su incremento.
- b) Cuando se presione el **botón 2**, comience el decremento en pasos de a 10 % de la intensidad del punto del display entre 0.10 y 0.90. Cuando alcance la intensidad mínima deberá volver a la intensidad máxima y continuar su decremento.
- c) Mostrar en display 1 (o todos juntos) la decena de la intensidad en la que se encuentran los puntos del display.
- d) Cuando se presione el botón **aceptar**, funcionara como el botón fijar, quedara fijo el valor de intensidad y no se movera hasta que se presione el botón **cancelar**.

Soluciones completas

```

1)      .cpu cortex-m4           // Indica el procesador de destino
.syntax unified                 // Habilita las instrucciones Thumb-2
.thumb                          // Usar instrucciones Thumb y no ARM

.include "configuraciones/lpc4337.s"
.include "configuraciones/rutinas.s"

/*****
/* Definiciones de macros
*****/

// Recursos utilizados por tecla 1
.equ TEC_1_PORT, 4
.equ TEC_1_PIN, 8
.equ TEC_1_BIT, 12
.equ TEC_1_MASK, (1 << TEC_1_BIT)

// Recursos utilizados por tecla 2
.equ TEC_2_PORT, 4
.equ TEC_2_PIN, 9
.equ TEC_2_BIT, 13
.equ TEC_2_MASK, (1 << TEC_2_BIT)

// Recursos utilizados por tecla 3
.equ TEC_3_PORT, 4
.equ TEC_3_PIN, 10
.equ TEC_3_BIT, 14
.equ TEC_3_MASK, (1 << TEC_3_BIT)

// Recursos utilizados por tecla 4
.equ TEC_4_PORT, 6
.equ TEC_4_PIN, 7
.equ TEC_4_BIT, 15
.equ TEC_4_MASK, (1 << TEC_4_BIT)

//Recursos utilizador por la tecla aceptar
.equ TEC_AC_PORT, 3
.equ TEC_AC_PIN, 2
.equ TEC_AC_BIT, 9
.equ TEC_AC_MASK, (1 << TEC_AC_BIT)

// Recursos utilizados por tecla de cancelar
.equ TEC_CN_PORT, 3
.equ TEC_CN_PIN, 1
.equ TEC_CN_BIT, 8
.equ TEC_CN_MASK, (1 << TEC_CN_BIT)

// Recursos utilizados por el teclado
.equ TEC_GPIO, 5
.equ TEC_OFFSET, ( TEC_GPIO << 2)
.equ TEC_MASK, ( TEC_1_MASK | TEC_2_MASK | TEC_3_MASK | TEC_4_MASK | TEC_AC_MASK |
                TEC_CN_MASK)

// Recursos utilizados por el punto del display
.equ SEG_DP_PORT,      6
.equ SEG_DP_PIN,       8
.equ SEG_DP_MAT,        1
.equ SEG_DP_TMR,        2

/*****
/* Vector de interrupciones
*****/

```

```

.section .isr // Define una seccion especial para el vector
.word stack // 0: Initial stack pointer value
.word reset+1 // 1: Initial program counter value
.word handler+1 // 2: Non maskable interrupt service routine
.word handler+1 // 3: Hard fault system trap service routine
.word handler+1 // 4: Memory manager system trap service routine
.word handler+1 // 5: Bus fault system trap service routine
.word handler+1 // 6: Usage fault system trap service routine
.word 0 // 7: Reserved entry
.word 0 // 8: Reserved entry
.word 0 // 9: Reserved entry
.word 0 // 10: Reserved entry
.word handler+1 // 11: System service call trap service routine
.word 0 // 12: Reserved entry
.word 0 // 13: Reserved entry
.word handler+1 // 14: Pending service system trap service routine
.word handler+1 // 15: System tick service routine
.word handler+1 // 16: IRQ 0: DAC service routine
.word handler+1 // 17: IRQ 1: M0APP service routine
.word handler+1 // 18: IRQ 2: DMA service routine
.word 0 // 19: Reserved entry
.word handler+1 // 20: IRQ 4: FLASHEEPROM service routine
.word handler+1 // 21: IRQ 5: ETHERNET service routine
.word handler+1 // 22: IRQ 6: SDIO service routine
.word handler+1 // 23: IRQ 7: LCD service routine
.word handler+1 // 24: IRQ 8: USB0 service routine
.word handler+1 // 25: IRQ 9: USB1 service routine
.word handler+1 // 26: IRQ 10: SCT service routine
.word handler+1 // 27: IRQ 11: RTIMER service routine
.word handler+1 // 28: IRQ 12: TIMER0 service routine
.word handler+1 // 29: IRQ 13: TIMER1 service routine
.word timer_isr+1 // 30: IRQ 14: TIMER2 service routine
.word handler+1 // 31: IRQ 15: TIMER3 service routine
.word handler+1 // 32: IRQ 16: MCPWM service routine
.word handler+1 // 33: IRQ 17: ADC0 service routine
.word handler+1 // 34: IRQ 18: I2C0 service routine
.word handler+1 // 35: IRQ 19: I2C1 service routine
.word handler+1 // 36: IRQ 20: SPI service routine
.word handler+1 // 37: IRQ 21: ADC1 service routine
.word handler+1 // 38: IRQ 22: SSP0 service routine
.word handler+1 // 39: IRQ 23: SSP1 service routine
.word handler+1 // 40: IRQ 24: USART0 service routine
.word handler+1 // 41: IRQ 25: UART1 service routine
.word handler+1 // 42: IRQ 26: USART2 service routine
.word handler+1 // 43: IRQ 27: USART3 service routine
.word handler+1 // 44: IRQ 28: I2S0 service routine
.word handler+1 // 45: IRQ 29: I2S1 service routine
.word handler+1 // 46: IRQ 30: SPIFI service routine
.word handler+1 // 47: IRQ 31: SGPIO service routine
.word handler+1 // 48: IRQ 32: PIN_INT0 service routine
.word handler+1 // 49: IRQ 33: PIN_INT1 service routine
.word handler+1 // 50: IRQ 34: PIN_INT2 service routine
.word handler+1 // 51: IRQ 35: PIN_INT3 service routine
.word handler+1 // 52: IRQ 36: PIN_INT4 service routine
.word handler+1 // 53: IRQ 37: PIN_INT5 service routine
.word handler+1 // 54: IRQ 38: PIN_INT6 service routine
.word handler+1 // 55: IRQ 39: PIN_INT7 service routine
.word handler+1 // 56: IRQ 40: GINT0 service routine
.word handler+1 // 56: IRQ 40: GINT1 service routine

/*****
/* Programa principal */
*****/

.section .data // Define la sección de variables (RAM)

```

```

flag: .byte 0,0

variables:
.zero 8

.equ periodo,    0      // Variable compartida el valor del periodo del PWM
.equ factor,     4      // Variable compartida con el factor de trabajo del PWM

.global reset        // Define el punto de entrada del código
.section .text       // Define la sección de código (FLASH)
.func main           // Indica al depurador el inicio de una funcion

reset:
// Mueve el vector de interrupciones al principio de la segunda RAM
LDR R1,=VTOR
LDR R0,=#0x10080000
STR R0,[R1]

BL config_tec

// Configura el pin del punto como salida TMAT
LDR R1,=SCU_BASE
MOV R0,#(SCU_MODE_INACT | SCU_MODE_INBUFF_EN | SCU_MODE_ZIF_DIS | SCU_MODE_FUNC5)
STR R0,[R1,#(SEG_DP_PORT << 7 | SEG_DP_PIN << 2)]

// Inicialización de las variables con los tiempos del PWM
LDR R1,=variables
LDR R0,=#200
STR R0,[R1,#periodo]
LDR R2,=#10
STR R2,[R1,#factor]

// Cuenta con clock interno
LDR R1,=TIMER2_BASE
MOV R0,#0x00
STR R0,[R1,#CTCR]

// Prescaler de 9500 para una frecuencia de 10 KHz
LDR R0,=9500
STR R0,[R1,#PR]

// La primera interrupcion ocurre despues de un factor de trabajo
STR R2,[R1,#MR1]

// El registro de match provoca interrupcion
MOV R0,#(MR1I)
STR R0,[R1,#MCR]

// Define el estado inicial y toggle on match del punto
MOV R0,#(3 << (4 + (2 * SEG_DP_MAT)))
STR R0,[R1,#EMR]

// Limpieza del contador
MOV R0,#CRST
STR R0,[R1,#TCR]

// Inicio del contador
MOV R0,#CEN
STR R0,[R1,#TCR]

// Limpieza del pedido pendiente en el NVIC
LDR R1,=NVIC_ICPR0
MOV R0,(1 << 14)
STR R0,[R1]

// Habilitacion del pedido de interrupcion en el NVIC

```

```

LDR R1,=NVIC_ISER0
MOV R0,(1 << 14)
STR R0,[R1]

```

```

CPSIE I      // Rehabilita interrupciones

```

```

main:
LDR R6,=GPIO_PIN0
LDR R0,[R6,#TEC_OFFSET]    //R0 = estado de teclas
LDR R1,=flag
TST R0,#TEC_1_MASK
BEQ tecla2
MOV R0,#1
STRB R0,[R1]
MOV R0,#0
STRB R0,[R1,#1]
B main_fin
tecla2:
TST R0,#TEC_2_MASK
BEQ tecla3
MOV R0,#0
STRB R0,[R1]
MOV R0,#1
STRB R0,[R1,#1]
B main_fin
tecla3:
TST R0,#TEC_3_MASK
BEQ main_fin
MOV R0,#0
STRH R0,[R1]
main_fin:
B main

```

```

.pool      // Almacenar las constantes de código
.endfunc

```

```

/*****
/* Rutina de servicio para la interrupcion del timer */
*****/
.func timer_isr
timer_isr:
// Limpio el flag de interrupcion
LDR R1,=TIMER2_BASE
LDR R0,[R1,#IR]
STR R0,[R1,#IR]
// Cargo el factor de trabajo
LDR R12,=variables
LDR R2,[R12,#factor]

//Me fijo si se apretó alguna tecla
LDR R0,=flag
LDRB R3,[R0]
TST R3,#1
BNE tec_s1
LDRH R3,[R0,#1]
TST R3,#1
BNE tec_s2
B timer_isr_fin
tec_s1:
ADD R2,#2
CMP R2,#190
IT GT
LDRGT R2,=10
STR R2,[R12,#factor]
B timer_isr_fin

```

```

tec_s2:
SUB R2,#2
CMP R2,#10
IT LT
LDRLT R2,=190
STR R2,[R12,#factor]
timer_isr_fin:
// Determino si el punto esta encendido o apagado
LDR R0,[R1,#EMR]
TST R0,#(1 << 1)

ITTE EQ
// Si esta apagado se utiliza periodo menos el factor de trabajo
LDREQ R3,[R12,#periodo]
SUBEQ R3,R2
// Si esta encendido se utiliza el factor de trabajo
MOVNE R3,R2

// Se actualiza el valor de match para la siguiente interrupcion
LDR R0,[R1,#MR1]
ADD R0,R3
STR R0,[R1,#MR1]

// Retorno
BX LR

.pool // Almacenar las constantes de código
.endfunc

/*****/
.func config_tec
config_tec:
PUSH {LR}
//Configura los digitos como gpio sin pull-up
LDR R1,=SCU_BASE
// Configura los pines de las teclas como gpio con pull-UP
MOV R0,#(SCU_MODE_PULLUP | SCU_MODE_INBUFF_EN | SCU_MODE_ZIF_DIS | SCU_MODE_FUNC4)
STR R0,[R1,#(TEC_1_PORT << 7 | TEC_1_PIN << 2)]
STR R0,[R1,#(TEC_2_PORT << 7 | TEC_2_PIN << 2)]
STR R0,[R1,#(TEC_3_PORT << 7 | TEC_3_PIN << 2)]
STR R0,[R1,#(TEC_4_PORT << 7 | TEC_4_PIN << 2)]
STR R0,[R1,#(TEC_AC_PORT << 7 | TEC_AC_PIN << 2)]
STR R0,[R1,#(TEC_CN_PORT << 7 | TEC_CN_PIN << 2)]
// Configura las teclas como entradas
LDR R1,=GPIO_DIR0
LDR R0,[R1,#TEC_OFFSET]
BIC R0,#TEC_MASK
STR R0,[R1,#TEC_OFFSET]
POP {PC} // Se retorna al programa principal
.pool // Se almacenan las constantes de codigo
.endfunc

/*****/
/* Rutina de servicio generica para excepciones */
/* Esta rutina atiende todas las excepciones no utilizadas en el programa. */
/* Se declara como una medida de seguridad para evitar que el procesador */
/* se pierda cuando hay una excepcion no prevista por el programador */
/*****/
.func handler
handler:
LDR R0,=set_led_1 // Apuntar al incio de una subrutina lejana
BLX R0 // Llamar a la rutina para encender el led rojo
B handler // Lazo infinito para detener la ejecucion
.pool // Almacenar las constantes de código
.endfunc

```