UNIVERSIDAD CARLOS III DE MADRID (Dpto. de Tecnología Electrónica)

Sist. Dig. Basados en Microprocesador **(Gr. Ing. Telemática)**

24 de febrero de 2012 1er PARCIAL (75 minutos)

No se permiten ni libros, ni apuntes, ni calculadoras programables. Sólo se permite el manual del microcontrolador
Se contestará sólo en el espacio reservado al efecto, pudiendo utilizar la cara posterior de la misma hoja.
No se pueden separar las hojas ni presentar hojas adicionales. ***Todas las respuestas deben estar justificadas.***

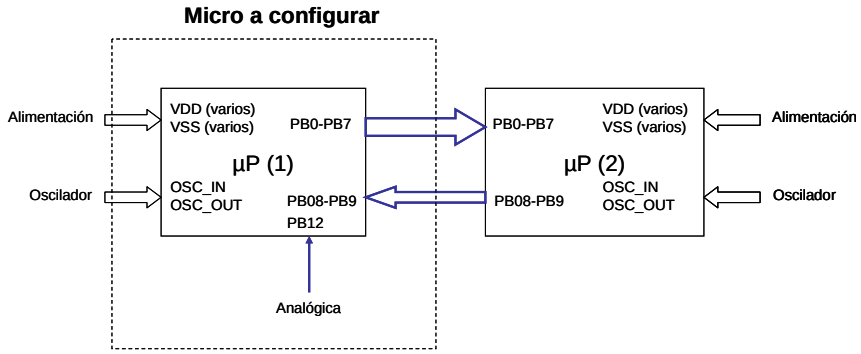
APELLIDOS	NOMBRE	NIA
	SOLUCIÓN	

CUESTIÓN 1 (2.5 puntos):

Para una determinada aplicación se necesita tener conectado un dispositivo al STM32L152RB a través de los pines PB0 – PB7, configurados para que el dispositivo le mande datos al STM32L152RB. Además, dicho dispositivo necesita que el STM32L152RB le envíe instrucciones a través de dos pines (PB8 – PB9). El sistema se conectará también a un sensor que suministra una tensión analógica (PB12).
Del resto de pines no se tiene conocimiento de lo que hacen.
Con toda esta información, configure los pines del STM32L152RB de forma que se satisfagan las necesidades anteriores, y configure el GPIO con los criterios dados.
Escriba las sentencias de código que realizarán dicha configuración y justifique brevemente su respuesta.

SOLUCIÓN

El diagrama de bloques sería el siguiente:



Los pines a configurar serán:

- PB0 a PB7 (8 pines): GPIO de salida (push-pull y velocidad lenta).
- PB8 a PB9 (2 pines): GPIO de entrada (sin pull-up ni pull-down).
- PB12: Entrada analógica.

APELLIDOS	NOMBRE	NIA
	SOLUCIÓN	

Por tanto:

```
//********** PB0 a PB7 *********
// PB0 a PB7 como GPIO de salida  -> …0101.0101.0101.0101 (binario) = 0x00005555
GPIOB->MODER  |=  0x00005555;       // pongo o uno sólo los bits que me interesan
GPIOB->MODER  &= ~0x00005555;       // pongo a cero sólo los bits que me interesan

// PB0 a PB7 como Push-pull -> …0000.0000 (binario) = 0x00000000
GPIOB->OTYPER &= ~0x000000FF; // pongo a cero solo los bits que me interesan

// PB0 a PB7 con velocidad lenta -> …0000.0000.0000.0000 (binario) = 0x00000000
GPIOB->OSPEEDR &= ~0x0000FFFF;      // pongo a cero solo los bits que me interesan

//********** PB8 a PB9 *********
// PB8 a PB9 como GPIO de entrada  -> …0000.0000.0000.0000 (binario) = 0x00000000
GPIOB->MODER  &=  ~0x0000FFFF;      // pongo a cero sólo los bits que me interesan

// PB8 a PB9 sin pull-up ni pull-down> …0000.0000.0000.0000 (binario) = 0x00000000
GPIOB->PUPDR  &=  ~0x0000FFFF;      // pongo a cero sólo los bits que me interesan

//********** PB12 *********
// PB12 como entrada analógica -> bits 24 y 25 de MODER a 1
GPIOB->MODER  |=  (0x3 << 24);
```

Criterios de corrección:

- *Resta 0.1 no poner los datos en hexadecimal*
- *Resta 0.5 no mencionar OTYPER, OSPEED y PUPDR*
- *Resta 0,5 errores en pines o bits*
- *Resta 0,5 errores puntuales en mascaras, y 2.5 si no son errores puntuales sino conceptuales*
- *Resta 0,5 limpieza/claridad*
- *Resta 0,1 otros fallos puntales de importancia menor*
- *Resta 0,5 otros fallos de importancia*

CUESTIÓN 2 (2.5 puntos):

Para una determinada aplicación, se requiere que se muestree continuamente una señal analógica, entre 0 y 3,3 voltios, a una frecuencia de muestreo de 100Hz, utilizando 256 niveles. Determine cómo configuraría el conversor ADC para dicha aplicación (justifique brevemente su respuesta). Considere que el lapso de tiempo entre una conversión y otra, para conseguir conversiones cada 10ms, viene dado por una función externa llamada `void lapso(void)`. Escriba las sentencias necesarias para la configuración del ADC.

SOLUCIÓN

256 niveles => 8 bits de precisión
100 Hz => 100 muestras por segundo => 10 milisegundos entre muestra

Dado que el tiempo de conversión es muy pequeño (decenas de ciclos * 1/32MHz => µsegundos) comparado con el intervalo entre muestra y muestra (10 ms), podemos usar dos opciones:

- Utilizamos la conversión en modo simple.
- Utilizamos la conversión en modo continuo, forzando a que no inicie la conversión hasta que hayamos leído el dato anterior.

Sin embargo, la mejor (la perfecta), es la segunda, dado que el tiempo entre muestra y muestra será exactamente 10 milisegundos.

Por tanto la configuración sería:

```
// Configuramos el pin de entrada (suponemos el PA4)
GPIOA->MODER |= 0x00000300; //PA4 analógico

// Configuramos el ADC
ADC1->CR2 &= ~(0x00000001);   // ADON=0 (me aseguro que está parado)
ADC1->CR1 = 0x02000000; // 8 bits de resolución, sin scan y sin interrupciones
ADC1->CR2 = 0x00000412; // SWSTART [30] = 0b (no arranco todavía)
                        // ALIGN [11] = 0b (alineo a la derecha)
                        // EOCS [10] = 1b (pone EOC a 1 al final de cada conversión)
                        // DELS [6:4] = 001b (no convierto hasta que lea el anterior)
                        // CONT [1] = 1b (conversión continua)
                        // ADON [0] = 0b (no inicializo todavía)
ADC1->SMPR1 = 0;        // 4 ciclos de reloj por conversión y canal
ADC1->SMPR2 = 0;
ADC1->SMPR3 = 0;
ADC1->SQR1 = 0x00000000; // 1 elemento solo
ADC1->SQR5 = 0x00000004; // Canal AIN4
ADC1->CR2 |= 0x00000001; // ADON=1 (enciendo el conversor)
while ((ADC1->SR&0x0040)==0); // ADONS [6] = 1b (espero que el conversor esté listo)
ADC1->CR2 |= (1 << 30);        // SWSTART [30] = 1b (arranco)


Y para usarlo basta con hacer lo siguiente:

while(1){
      lapso();                // espero 10ms
      valor = ADC1->DR;       // leo
}
```

Si quisiéramos, podríamos comprobar el bit "EOC" del status, dado que lo hemos configurado para que nos avise de cada conversión, pero no es necesario dado que en 10ms es seguro que habrá terminado.

```
While(1){
      lapso();                // espero 10ms

      // leo y gestiono el error en caso de que no haya dato
      if ((ADC1->SR&0x0002) != 0) valor = ADC1->DR;
      else [aquí gestionariamos el supuesto error];
}
```

Criterios de corrección:

- *Errores configuración: -0.5*
- *No justificar: -0.5*
- *Código no comentado o poco comprensible: -0.5*
- *Errores menores: -0.1*
- *Errores de concepto: -2.5*

CUESTIÓN 3 (2 puntos):

Indique los errores que observa en el siguiente código (justifique su respuesta). Considere que pclk=15MHz.

Fichero Biblioteca_SDM.h:

```
void Init_SDM(void);
void Init_LCD(void);
void LCD_Limpia(void);
void LCD_Texto(unsigned char *texto);
```

Fichero Principal.c:

```
#include "stm32l1xx.h"
#include "Biblioteca_SDM.h"
int main(void){
   unsigned short valor = 0;
   char texto[5];                  // necesito string
   Init_SDM();
   Init_LCD();
   GPIOA->MODER &= ~(1 << (0*2 +1));
   GPIOA->MODER &= ~(1 << (0*2));
   GPIOA->PUPDR &= ~(11 << (0*2));
   GPIOA->MODER |= 0x00000300;
   ADC1->CR2 &= ~(0x00000001);
   ADC1->CR1 = 0x00000002;
   ADC1->CR2 = 0x00000400;
   ADC1->SMPR1 = 0;
   ADC1->SMPR2 = 0;
   ADC1->SMPR3 = 0;
   ADC1->SQR1 = 0x00000000;
   ADC1->SQR5 = 0x00000004;
   if ((GPIOA->IDR&0x00000001)!=0) {
      while ((ADC1->SR&0x0040)==0);
      ADC1->CR2 |= 0x40000000;
      while ((ADC1->SR&0x0002)==0);
      valor = ADC1->DR;
      LCD_Texto(valor);
      }
   }
```

No se permiten ni libros, ni apuntes, ni calculadoras programables. Sólo se permite el manual del microcontrolador
Se contestará sólo en el espacio reservado al efecto, pudiendo utilizar la cara posterior de la misma hoja.
No se pueden separar las hojas ni presentar hojas adicionales. ***Todas las respuestas deben estar justificadas.***

SOLUCIÓN

```
#include "stm32l1xx.h"
#include "Biblioteca_SDM.h"

void bin2ascii(unsigned short valor, char *texto){
      …
}

int main(void){
   unsigned short valor = 0;
   Init_SDM();
   Init_LCD();

   // pin PA0 como entrada sin pull-up, ni pull-down
   GPIOA->MODER &= ~(1 << (0*2 +1));
   GPIOA->MODER &= ~(1 << (0*2));
   GPIOA->PUPDR &= ~(11 << (0*2));

   // configuro ADC
   GPIOA->MODER |= 0x00000300;      // pin PA4 como entrada analógica
   ADC1->CR2 &= ~(0x00000001);      // me aseguro que ADC está apagado
   ADC1->CR1 = 0x00000000;          // sin scan, 12 bits de resolución
   ADC1->CR2 = 0x00000400;          // modo simple, aviso en cada conversión
   ADC1->SMPR1 = 0;                 // 4 ciclos por conversión
   ADC1->SMPR2 = 0;
   ADC1->SMPR3 = 0;
   ADC1->SQR1 = 0x00000000;         // canal 4
   ADC1->SQR5 = 0x00000004;
   ADC1->CR2 |= 0x00000001;         // arranco el conversor ADON = 1
   while ((ADC1->SR&0x0040)==0);    // el conversor está listo ¿ADONS=1?
                                    // (no estaba mal en su posición original)

   while (1){                       // bucle infinito
      if ((GPIOA->IDR&0x00000001)!=0) {  // espero que pulse el botón
         ADC1->CR2 |= 0x40000000;        // inicio la conversión (SWSTART=1)
         while ((ADC1->SR&0x0002)==0);   // ¿dato listo? (¿EOC=1?)
         valor = ADC1->DR;               // cojo el dato
         LCD_limpia();                   // limpio el LCD
         Bin2ascii(valor, texto);        // convierto el valor a ASCII
         LCD_Texto(texto);               // lo muestro
      }
   }
}
```

Criterios de corrección:

- *Falta while(1)*
- *Falta limpiar LCD*
- *Falta convertir dato*
- *Falta arrancar conversor*
- *Falta definir string*

CUESTIÓN 4 (3 puntos):

Resuelva la creación de un potenciómetro digital. Un potenciómetro digital es un dispositivo tal que, por un pin ofrece una tensión que puede variarse según se pulsen dos teclas, una para subir su valor y otra para bajarlo. En este caso la tensión a generar estará entre 0 y 3V, y los incrementos y decrementos de tensión serán, como mucho, de 1mV. Partiendo de la placa STM32L-Discovery (), realice:
a) el diagrama de bloques de la solución
b) indique cómo configuraría cada uno de los periféricos (no escriba el código)
c) realice el diagrama de flujo.

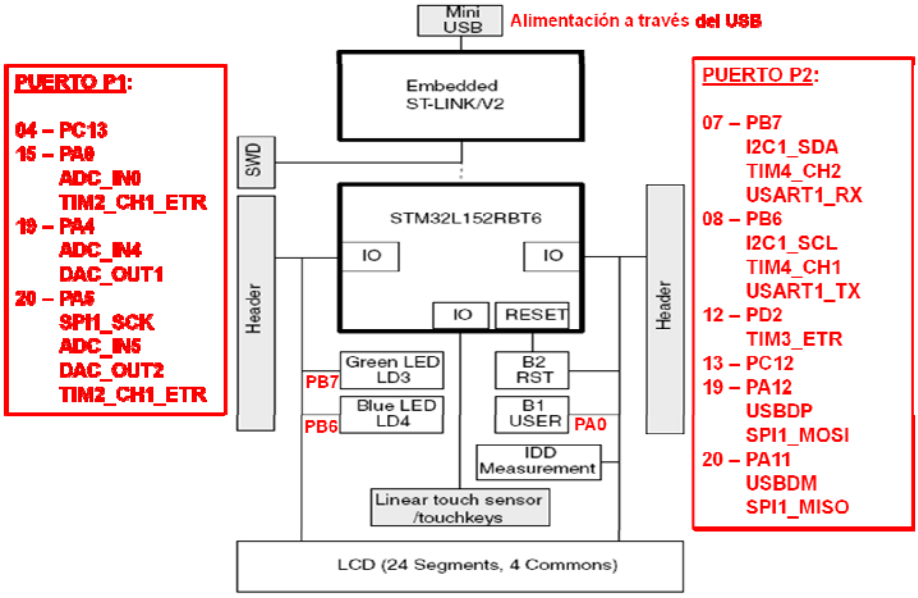
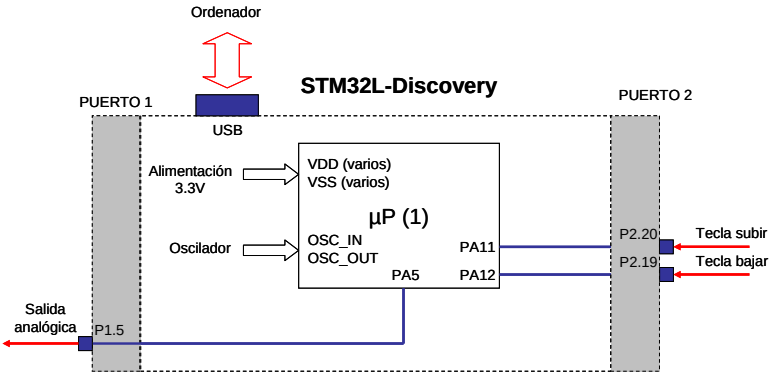


Fig. 1: Diagrama de bloques de la STM32L-Discovery. Puerto P1 se refiere al puerto de expansión de dicha placa, mientras P2 es el segundo puerto de expansión de dicha placa, nada que ver con los puertos GPIO (PA – PD) del STM32L152RB.

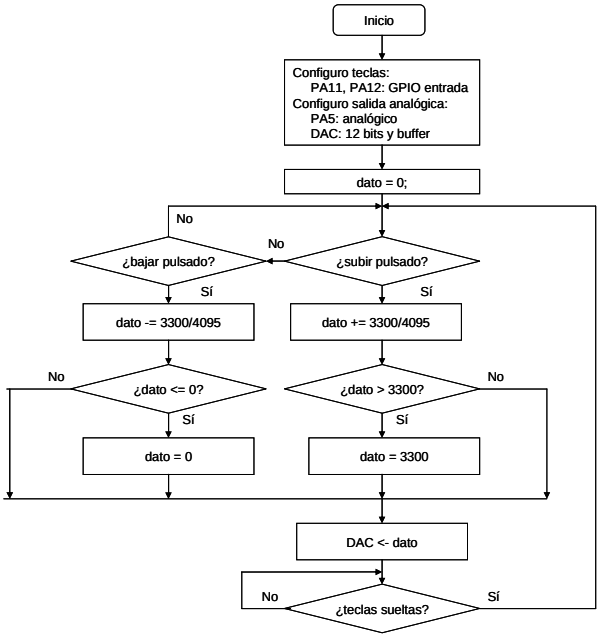
SOLUCIÓN

a)Diagrama de bloques.(1 pto)



b) Configuración de los periféricos (0.5 pto).
- Puerto A, pin 11 y 12 como GPIO de entrada sin pull-up, ni pull-down.
- Puerto A, pin 5 como analógico.
- DAC: habilitando buffer y con 12 bits de precisión(3,3/0,001 = 3300 < 2^12).

c) Diagrama de flujo (1 pto.).
Consideraciones:
- No ponemos antirebotes
- No consideramos error pulsar dos teclas
- La tecla subir tiene prioridad sobre la tecla bajar
- Solo se cuenta un incremento por pulsación (pulsar-soltar).



No se permiten ni libros, ni apuntes, ni calculadoras programables. Sólo se permite el manual del microcontrolador
Se contestará sólo en el espacio reservado al efecto, pudiendo utilizar la cara posterior de la misma hoja.
No se pueden separar las hojas ni presentar hojas adicionales. ***Todas las respuestas deben estar justificadas.***

Criterios de corrección:

- *Errores de concepto anulan la puntuación de un apartado.*
- *Errores mayores 0.5*
- *Errores menores 0.1*
- *Realizar el apartado "a" usando los botones de la placa, sin saber el pineado o, sin considerar el efecto del reset anula el apartado.*