| UNIVERSIDAD DE ALCALÁ<br>ESCUELA POLITÉCNICA SUPERIOR<br>DEPARTAMENTO DE ELECTRÓNICA | | GRADO EN INGENIERÍA ELECTRÓNICA Y AUTOMÁTICA INDUSTRIAL | |
|---|---|---|---|
| ASIGNATURA | SISTEMAS ELECTRÓNICOS DIGITALES | FECHA | ENERO 2014 |
| APELLIDOS, NOMBRE | SOLUCIÓN | GRUPO | |

## PRUEBA DE EVALUACIÓN GLOBAL

### Ejercicio 1 (14 puntos)

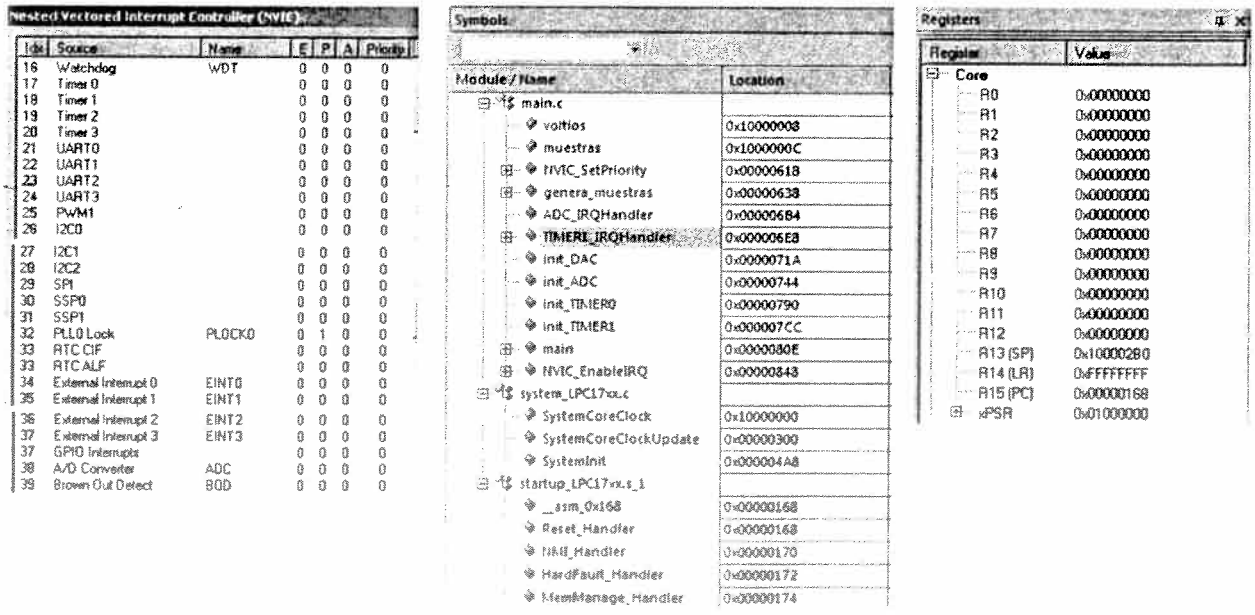
El proyecto cuyo código se muestra como ANEXO trata de demostrar el funcionamiento del ADC y DAC del LPC1768.

El ADC se configura para convertir la señal analógica conectada al canal 0, de forma que el inicio de conversión se lleve a cabo periódicamente gracias al Timer 0 configurado en Modo Match (comparación): MAT0.1. En el registro ADGDR se obtiene tras el fin de la conversión, que genera una interrupción, el valor de 12 bits que se refleja en una variable global traducido a voltios.

Junto a la conversión, el módulo de comparación MAT0.1 se configura para conmutar el valor del pin P1.29, de modo que la conversión se produce cada flanco de subida de la señal digital que se observa en ese pin, en el que se genera así una señal cuadrada de frecuencia igual a la de muestreo del ADC.

El DAC se utiliza para generar una señal senoidal de 1 KHz entre 0 y 3.3V a través del pin P0.23 (AOUT). Para ello se almacenan en un array 32 valores de 10 bits correspondientes al muestreo de la señal senoidal en un periodo, que se convierten con el DAC periódicamente gracias al Timer 1.

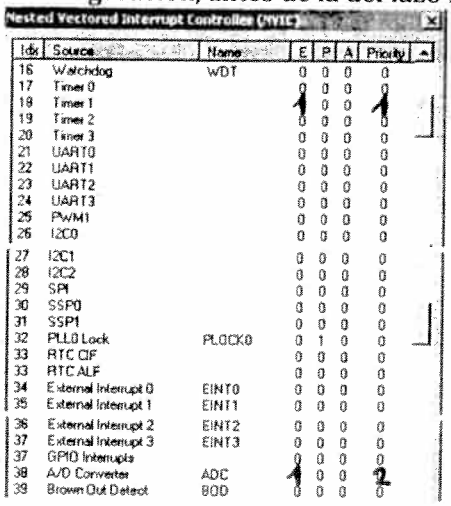
a) A partir de las ventanas capturadas en el simulador, justo después de un RESET, conteste a las siguientes cuestiones:




a.1) Muestre DIRECCIÓN y CONTENIDO de la tabla de vectores, en lo referente al vector de RESET y a los de las fuentes de interrupción incluidas en el proyecto (VTOR=0x00). Justifique la respuesta. (3 ptos.)

| DIRECCIÓN | CONTENIDO | |
|---|---|---|
| 0x0000.0000 | 0x1000.0280 | SSP |
| 0x0000.0004 | 0x0000.0169 | PC |
| ⋮ | ⋮ | |
| 0x0000.0048 | 0x0000.06E9 | TIMER1 → Nº vector = 18 |
| ⋮ | ⋮ | |
| 0x0000.0098 | 0x0000.0685 | ADC → Nº vector = 38 |

a.2) Modifique sobre la ventana del NVIC los cambios que ocurren tras la ejecución de las funciones de configuración, antes de la del lazo infinito del código. (1 pto.)




b) A partir del código desensamblado de la función init_DAC(), mostrado a continuación, conteste a las siguientes cuestiones:

```
    36: void init_DAC(void)
    37: {
0x00000718 4770      BX       lr
    38:         LPC_PINCON->PINSEL1|= (2<<20);          // DAC output = P0.26 (AOUT)
0x0000071A 485F      LDR      r0,[pc,#380]  ; @0x00000898
0x0000071C 6840      LDR      r0,[r0,#0x04]
0x0000071E F4401000  ORR      r0,r0,#0x200000
0x00000722 495D      LDR      r1,[pc,#372]  ; @0x00000898
0x00000724 6048      STR      r0,[r1,#0x04]
    39:         LPC_PINCON->PINMODE1|= (2<<20);         // Deshabilita pullup/pulldown
0x00000726 4608      MOV      r0,r1
0x00000728 6C40      LDR      r0,[r0,#0x44]
0x0000072A F4401000  ORR      r0,r0,#0x200000
0x0000072E 6448      STR      r0,[r1,#0x44]
    40:         LPC_SC->PCLKSEL0|= (0x00<<22);          // CCLK/4 (Fpclk después del reset)
0x00000730 485A      LDR      r0,[pc,#360]  ; @0x0000089C
0x00000732 6800      LDR      r0,[r0,#0x00]
0x00000734 495A      LDR      r1,[pc,#360]  ; @0x000008A0
0x00000736 F8C101A8  STR      r0,[r1,#0x1A8]
    41:         LPC_DAC->DACCTRL=0;                                    // ?
0x0000073A F04F0000  MOV      r0,#0x00
0x0000073E 4955      LDR      r1,[pc,#340]  ; @0x00000894
0x00000740 6048      STR      r0,[r1,#0x04]
    42: }
```

b.1) Indique el tamaño (en bytes) de la función. (2 ptos.)

$0x740 - 0x71A + 2 = 40$ bytes

b.2) Indique qué instrucciones almacenan datos en memoria. (2 ptos.)

Marcadas con "*" en apartado b)

c) ¿Cuál es la frecuencia de interrupción del Timer 1? Complete la instrucción que permite realizar esta configuración en el código que se muestra a continuación. (2 ptos.)

```
void init_TIMER1(void)
{
    LPC_SC->PCONP|=(1<<2);              //
    LPC_TIM1->PR = 0x00;                //
    LPC_TIM1->MCR = 0x03;               // Reset TC on Match, and Interrupt!
    LPC_TIM1->MR0 =(FpcLK/Fout/N_muestras) - 1;
    LPC_TIM1->EMR = 0x02;               //
    LPC_TIM1->TCR = 0x01;               //
    NVIC_EnableIRQ(TIMER1_IRQn);        //
    NVIC_SetPriority(TIMER1_IRQn,1);    //
}
```

$F_{interrup.\ (TIMER\ 1)} = F_{out} * N\_muestras = 1000 * 32 = \underline{32\ kHz.}$

d) Complete la instrucción que configura el MAT0.1 del Timer 0 para conseguir la frecuencia de muestreo del ADC deseada, en el código que se muestra a continuación. (2 ptos.)

```
void init_TIMER0(void)
{
    LPC_SC->PCONP|=(1<<1);              //
    LPC_PINCON->PINSEL3|= 0x0C000000;   //
    LPC_TIM0->PR = 0x00;                //
    LPC_TIM0->MCR = 0x10;               //
    LPC_TIM0->MR1=(FpcLK/F_muestreo/2)-1; Se han de producir DOS Match para iniciar la conversión!!!!
    LPC_TIM0->EMR = 0x00C2;             //
    LPC_TIM0->TCR = 0x01;               //
}
```

e) Calcule el tiempo de conversión del ADC. (2 ptos.)

$$t_{CONV} = 65 * T_{CLK\ (ADC)} = 65 * \frac{1 + CLKDIV}{F_{pcLK}} = \underline{5{,}2\ \mu s.}$$

**Ejercicio 2 (9 puntos)**

Se desea implementar un sistema de memoria SRAM de 256Kx8 para un uP de 8 bits con dispositivos de tipo:

- GS72108AGP.
- CY62256N.

**NOTA**: El patillaje de ambos tipos de memorias, así como las características básicas del módulo EMC del LPC1788 se adjuntan en un ANEXO al final del documento.

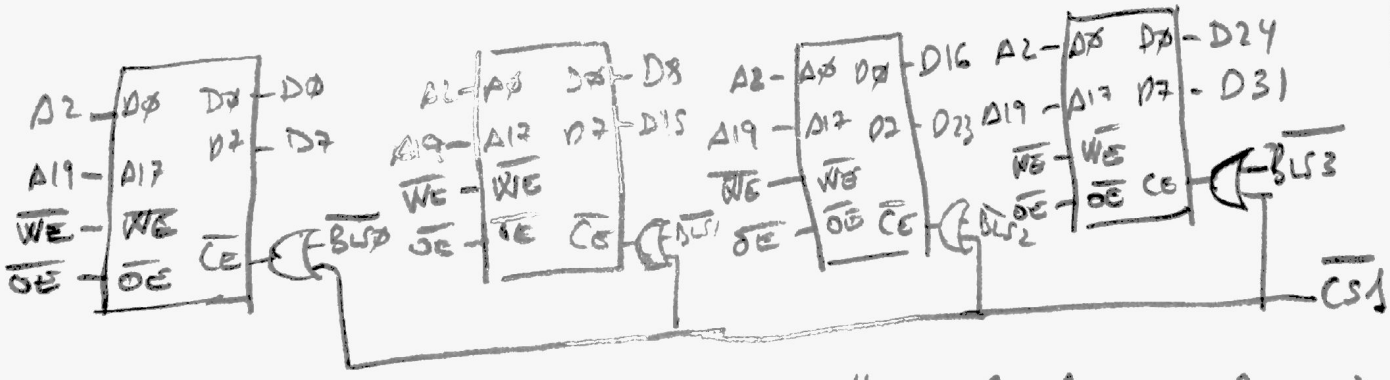
a) Indique número de chips a usar en cada caso y tipo de ampliación/organización que se debería implementar entre ellos. ¿Qué configuración será la más adecuada si se desea minimizar el número de chips (tamaño de la tarjeta) y el espacio de mapa usado? (2 ptos.)

1 chip GS72108AGP, que es de 256K8. Config. óptima
No hay ampliación ni organización

b) Finalmente se realiza el diseño para un LPC1788, configurado en 32 bits y con ordenación Little Endian, a partir de la dirección 0x93000000. Conteste de forma justificada a las siguientes preguntas.

b.1) Dibuje el diseño del sistema de memoria a implementar que minimice el número de chips a usar. ¿Qué tipo de ampliación/organización se está usando en este caso? (3 ptos.)

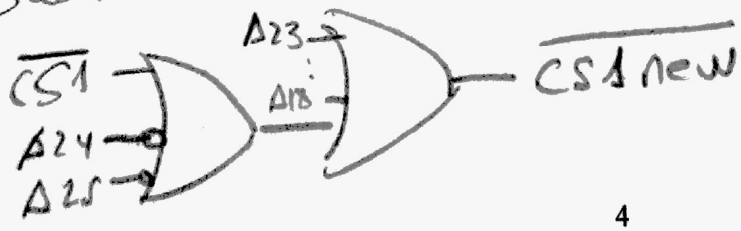
4 chips de 256K8 (GS72108AGP) pues de la CY62256N necesitaría 8 chips.
↓
Ordenación en 4 bancos => 4x256K8 => 1M8 la implementa




(Por simplificar obviamos el prefijo "EMC_" de las líneas del µC)

b.2) A la hora de conectar la línea de $\overline{CE}$ de los chips de memoria realice decodificación completa. ¿Cuál será la última posición usada por este segmento en el mapa del uC? (3 ptos.)

Sustituimos $\overline{CS1}$ por $\overline{CS1new}$:




Direcciones: 0x93000000
0x9303FFFF
↓
Última dirección USADA

b.3) Indique también cuál sería la configuración del registro STATICCONFIGx necesario. (1 pto.)

Usamos CS1 en 32 bits así que:

STATICCONFIG1 = 0x00000082

**Ejercicio 3 (7 puntos)**

Se desea ahora hacer el diseño de un mapa de memoria para un uP de 16 bits con las características siguientes:

- A0-A23; R/W; $\overline{BHE}$ (Bus High Enable, validación de parte alta del dato) y $\overline{BLE}$; ordenación Big Endian.

El diseño debe adaptarse a una aplicación de control de acceso a una fábrica, cuyas características se muestran a continuación:

- Se desea reservar 256Kbytes para la aplicación, tablas de datos constantes y tabla de vectores.
- Se desea reservar 256Kbytes para datos variables y 128Kbytes para la pila.
- Se desea incluir en el sistema 2 periféricos para tarjetas chip de 32 registros y sendos interfaces de gestión de apertura de puertas de 4 registros. Todos ellos son de 8 bits.

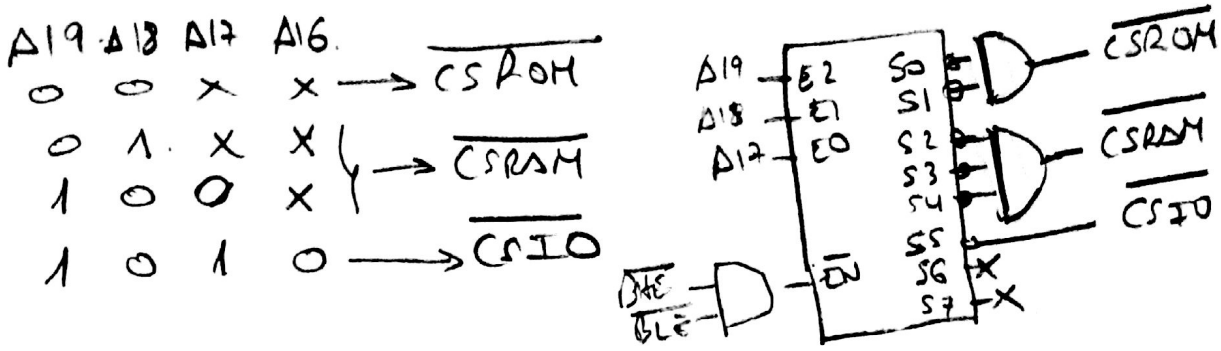
Para realizar el mapeado de los dispositivos necesarios se debe tener en cuenta que el uP ha de tener la tabla de vectores ubicada a partir de la dirección 0, que se puede usar cualquier tipo de chip comercial de memoria, que no se pueden dejar espacios libres entre los distintos segmentos de memoria y que se desea implementar exactamente la cantidad indicada.

a) Diseñe el mapa funcional y físico del sistema, rellenando la tabla mostrada a continuación, de los segmentos de memoria volátil (RAM), no volátil (ROM) y periféricos (IO). (3 ptos.)

| | Funcionalidad | Direcciones (Inicio-Fin) | Chips a implementar |
|---|---|---|---|
| **RAM** | 2 BANCOS DATOS VARIABLES Y PILA 384K8 | 0x040000 ↕ 0x09FFFF | 2 chips de 128K8, 1 en cada banco<br>1 chips de 64K8, 1 en cada banco |
| **ROM** | 2 BANCOS: DATOS CONSTANTES, APLICACIÓN 256K8 | 0x000000 ↕ 0x03FFFF | 2 chips de 128K8<br>1 en cada banco |
| **IO** | 2 BANCOS: 2 x PERIF TARJETAS CHIP 32x8<br>2 x PERIF APERTURA 4x8 | 0x0A0000<br>0x0A003E<br>0x0A0001<br>0x0A003F<br>0x0A0040<br>0x0A0048<br>0x0A0041<br>0x0A0049 | 4 chips.<br>1 de 32 y 1 de 4 bytes en cada banco. |

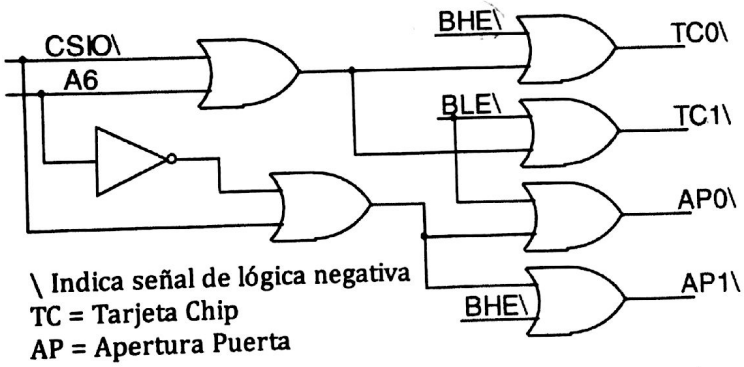
5 líneas A3

b) Realice la lógica de decodificación que genere las señales de $\overline{CS}$ de los 3 segmentos ($\overline{CSRAM}$, $\overline{CSROM}$ y $\overline{CSIO}$) usando decodificación incompleta. Indique con cuántas direcciones diferentes se accede a cada posición de memoria no volátil. (2 ptos.)




c) A la vista del circuito de lógica de decodificación generado a partir de la señal $\overline{CSIO}$ indique en un diagrama la dirección inicial y final de los 4 periféricos e interfaces incluidos en el mapa. (2 ptos.)

**NOTA**: Suponga para este apartado que el rango de mapa direccionado por la señal $\overline{CSIO}$ empieza en la dirección 0x00100000.




\ Indica señal de lógica negativa
TC = Tarjeta Chip
AP = Apertura Puerta

Parte alta BUS → Dirección BAJA
⊕ A6 = Ø.

Parte baja BUS → Dirección ALTA
⊕ A6 = Ø

Dir Alta ⊕ A6 = 1

Dir Baja ⊕ A6 = 1

| 2N | 2N+1 |
|---|---|
| TCØ | TC1 |
| AP1 | APØ |

0x00100000
0x00100001
0x00100003E
0x00100040
0x00100048
0x0010003F
0x00100041
0x00100049

# ANEXO (Código del Programa. Ejercicio 2)

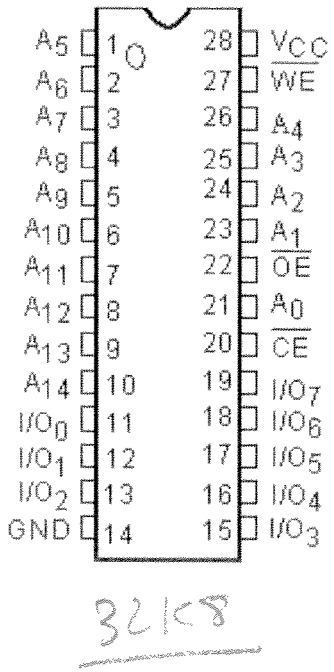
```
#include <LPC17xx.H>
#include <Math.h>
#define F_cpu 100e6   // Defecto Keil (xtal=12Mhz)
#define F_pclk F_cpu/4 // Defecto despues del reset
#define F_muestreo 100 // Fs=100Hz (Cada 10ms se toma una muestra del canal 0)
#define pi 3.14159
#define F_out 1000
#define N_muestras 32
#define V_refp 3.3

uint16_t muestras[N_muestras];      // Array para guardar las muestras de un ciclo de un seno
float voltios;

void genera_muestras(uint8_t muestras_ciclo)
{
  uint8_t i;
  for(i=0;i<muestras_ciclo;i++)
    muestras[i]=1023*(0.5 + 0.5*sin(2*pi*i/muestras_ciclo)); // Ojo! el DAC es de 10bits
}

void ADC_IRQHandler(void)
{
  voltios= ((LPC_ADC->ADGDR >>4)&0xFFF)*3.3/4095; // se borra automat. el flag DONE al leer ADCGDR
}

// Timer 1 interrumpe periódicamente a F = F_out x N_muestras !!!!
// La muestra correpondiente del array,  se saca al DAC en cada interrupción
void TIMER1_IRQHandler(void)
{
  static uint8_t indice_muestra;
  LPC_TIM1->IR|= (1<<0);              //
  LPC_DAC->DACR= muestras[indice_muestra++] << 6; // ?
  indice_muestra&= 0x1F;              //
}

void init_DAC(void)
{
  LPC_PINCON->PINSEL1|= (2<<20);     // DAC output = P0.26 (AOUT)
  LPC_PINCON->PINMODE1|= (2<<20);    // Deshabilita pullup/pulldown
  LPC_SC->PCLKSEL0|= (0x00<<22);     // CCLK/4 (Fpclk después del reset) (100 Mhz/4 = 25Mhz)
  LPC_DAC->DACCTRL=0;                //
}
```

```
void init_ADC(void)
{
  LPC_SC->PCONP|= (1<<12);           // POwer ON
  LPC_PINCON->PINSEL1|= (1<<14);     // ADC input= P0.23 (AD0.0)
  LPC_PINCON->PINMODE1|= (2<<14);    // Deshabilita pullup/pulldown
  LPC_SC->PCLKSEL0|= (0x00<<8);      // CCLK/4 (Fpclk después del reset) (100 Mhz/4 = 25Mhz)
  LPC_ADC->ADCR= (0x01<<0)|          // Canal 0
                 (0x01<<8)|          // CLKDIV=1   (Fclk_ADC=25Mhz /(1+1)= 12.5Mhz)
                 (0x01<<21)|         // PDN=1
                 (4<<24);            // Inicio de conversión con el Match 1 del Timer 0

  LPC_ADC->ADINTEN= (1<<0)|(1<<8);   // Hab. interrupción fin de conversión canal 0
  NVIC_EnableIRQ(ADC_IRQn);          //
  NVIC_SetPriority(ADC_IRQn,2);      //
}

/*  Timer 0 en modo Output Compare (reset TOTC on Match 1)
  Counter clk: 25 MHz   MAT0.1 : On match, Toggle pin/output (P1.29)
  Cada 2 Match se provoca el INICIO DE CONVERSIÓN DEL ADC
  Habilitamos la salida (MAT0.1) para observar la frecuencia de muestreo del ADC */

void init_TIMER0(void)
{
    LPC_SC->PCONP|=(1<<1);             //
    LPC_PINCON->PINSEL3|= 0x0C000000; //
    LPC_TIM0->PR = 0x00;               //
    LPC_TIM0->MCR = 0x10;              //
    LPC_TIM0->MR1 =                  ; // Se han de producir DOS Match para iniciar la conversión!!!!
    LPC_TIM0->EMR = 0x00C2;            //
    LPC_TIM0->TCR = 0x01;              //
}


/*  Timer 1 en modo Output Compare (reset TOTC on Match 0)
  Counter clk: 25 MHz   MAT1.0 : On match, salida de una muestra hacia el DAC */
void init_TIMER1(void)
{
    LPC_SC->PCONP|=(1<<2);             //
    LPC_TIM1->PR = 0x00;               //
    LPC_TIM1->MCR = 0x03;              // Reset TC on Match, and Interrupt!
    LPC_TIM1->MR0 =                  ; //
    LPC_TIM1->EMR = 0x02;              //
    LPC_TIM1->TCR = 0x01;              //
    NVIC_EnableIRQ(TIMER1_IRQn);       //
    NVIC_SetPriority(TIMER1_IRQn,1);   //
}

int main(void)
{
  NVIC_SetPriorityGrouping(2);
  genera_muestras(N_muestras);
  init_ADC();
  init_DAC();
  init_TIMER0();
  init_TIMER1();
  while(1);
}
```

# ANEXO (Patillaje de Memorias. Ejercicio 2)

**GS72108AGP**


**CY62256N**




# ANEXO (Características Básicas del EMC en el LPC1788. Ejercicio 2)

| Name | Description |
|---|---|
| STATICCONFIGn | • **Memory width** (1:0)<br>0x0: 8 bit (reset)<br>0x1: 16 bit<br>0x2: 32 bit<br>• **Page mode** (3). The EMC can burst up to 4 external accesses<br>0x0: Async page mode disable (reset)<br>0x1: Async page mode enabled<br>• **Chip select polarity** (6)<br>0x0: Active LOW chip select<br>0x1: Active HIGH chip select<br>• **Byte lane state (7).** Enables different types of memory to be connected<br>0x0: For reads all the bits in BLSn[3:0] are HIGH. For writes all the bits in BLSn[3:0] are LOW<br>0x1: For reads and writes the respective active bits in BLSn[3:0] are LOW<br>• **Extended wait (8).** Uses StaticExtendedWait register to time both read and write transfers<br>0x0: Extended wait disabled (reset value)<br>0x1: Extended wait enabled |

| Four static memory chip selects. | | Data bus pins | Address bus pins | Control pins |
|---|---|---|---|---|
| 0x8000 0000 - 0x83FF FFFF | Static memory chip select 0 (up to 64 MB) | | | SRAM |
| 0x9000 0000 - 0x93FF FFFF | Static memory chip select 1 (up to 64 MB) | | | |
| 0x9800 0000 - 0x9BFF FFFF | Static memory chip select 2 (up to 64 MB) | EMC_D[31:0] | EMC_A[25:0] | EMC_BLS[3:0], EMC_CS[3:0], EMC_OE, EMC_WE |
| 0x9C00 0000 - 0x9FFF FFFF | Static memory chip select 3 (up to 64 MB) | | | |