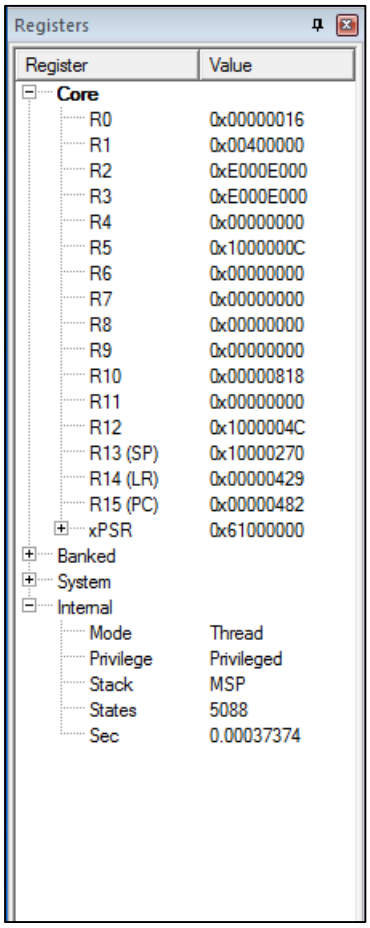
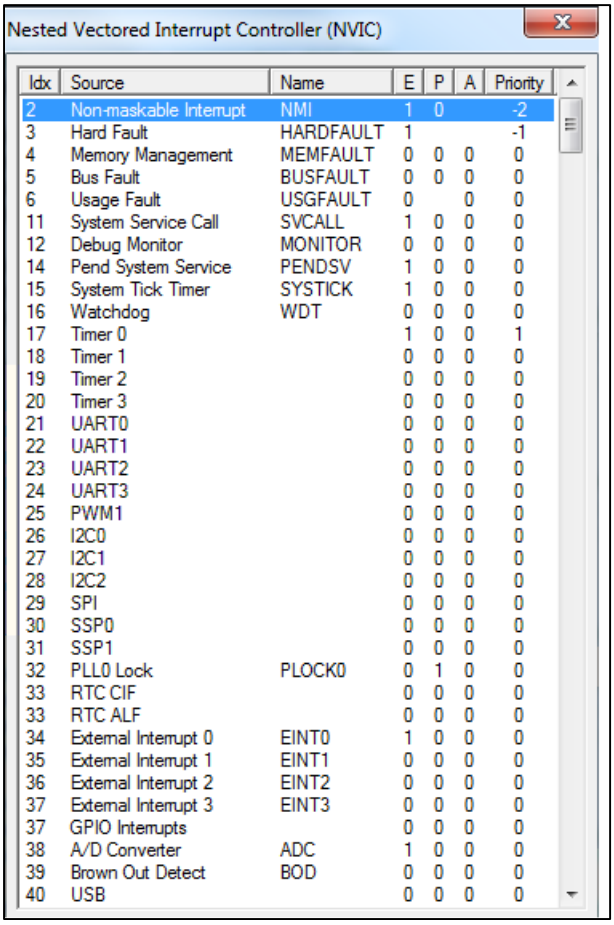
UNIVERSIDAD DE ALCALÁ ESCUELA POLITÉCNICA SUPERIOR DEPARTAMENTO DE ELECTRÓNICA		GRADO EN INGENIERÍA ELECTRÓNICA Y AUTOMÁTICA INDUSTRIAL	
ASIGNATURA	SISTEMAS ELECTRÓNICOS DIGITALES	FECHA	ENERO 2015
APELLIDOS, NOMBRE		GRUPO	

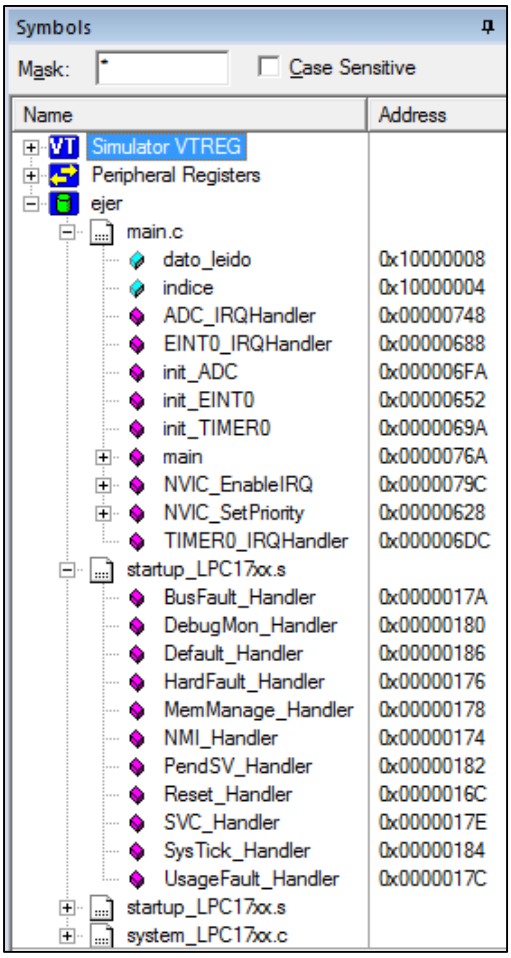
PRUEBA DE EVALUACIÓN GLOBAL

Ejercicio 1 (8 puntos)

A continuación se muestra el contenido de las ventanas capturadas en el simulador Keil, justo después de un RESET de un sistema basado en el LPC1768:







a) Atendiendo a esta información indique **dirección y contenido** de la tabla de vectores referente al puntero de pila, al vector de RESET y a los vectores de las fuentes de interrupción programadas por el usuario.

(2 ptos.)

b) Suponiendo que finalmente se configuran los niveles de prioridades tal y como se muestra a continuación, complete la tabla incluida a continuación con el estado **(X: inactivo; P: pendiente; A: activo; A*: activo y en ejecución)** al que cambia cada una de las excepciones indicadas (columnas) tras la aparición de los eventos indicados (filas). Suponga además que estos eventos se producen en el orden en el que aparecen en la tabla.

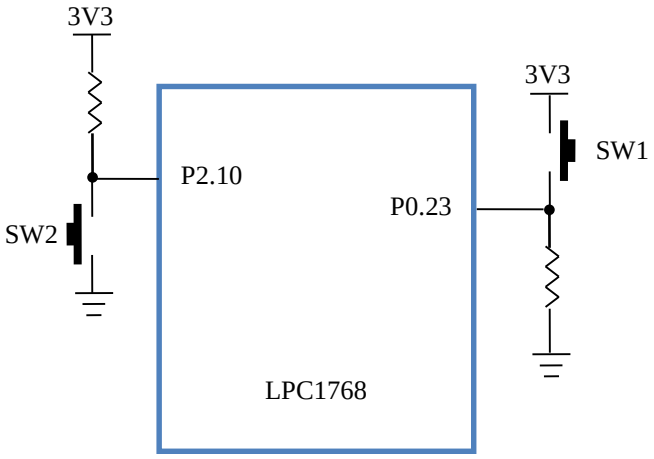
(2 ptos.)

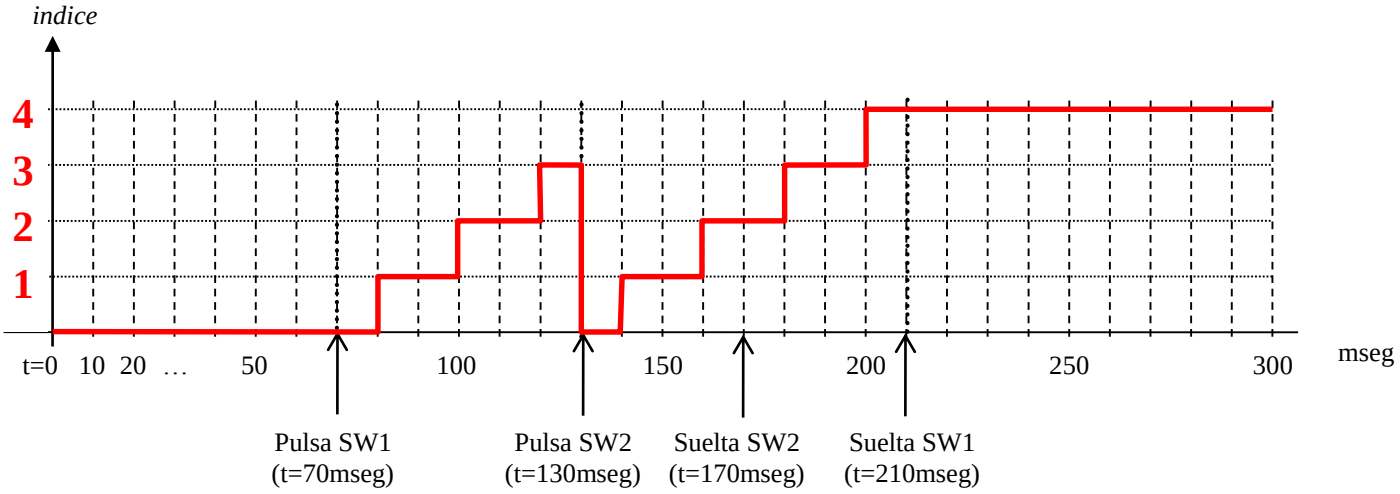
```
Config_Prioridades(void){
   NVIC_SetPriority(TIMER0_IRQn, 4);    //Prioridad: 001.00xxx => 1.0
   NVIC_SetPriority(EINT0_IRQn, 2);     //Prioridad: 000.10xxx => 0.1, (más prioritaria)
   NVIC_SetPriority(SYSTICK_IRQn, 5); //Prioridad: 001.01xxx => 1.1 (menos prioritaria)
   NVIC_SetPriorityGrouping(4);
   }
```

		Estado (X, P, A, A*)		
Orden	**Evento**	**TIMER0**	**EINT0**	**SYSTICK**
1	Solicita atención SYSTICK	**X**	**X**	**A***
2	Solicita atención EINT0	**X**	**A***	**A**
3	Solicita atención TIMER0	**P**	**A***	**A**
4	Finaliza EINT0	**P**	**X**	**A***
5	Finaliza SYSTICK	**A***	**X**	**X**
6	Solicita atención SYSTICK	**A***	**X**	**P**
7	Finaliza TIMER0	**X**	**X**	**A***
8	Finaliza SYSTICK	**X**	**X**	**X**

c) Se realiza además un montaje, en donde se conectan dos pulsadores, SW1 y SW2, a los pines P0.23 y P2.10 respectivamente, tal y como se muestra en la figura inferior. Atendiendo a esta configuración, y al código incluido en el Anexo 1, dibuje sobre el cronograma mostrado más abajo el valor a lo largo del tiempo de la variable "indice" considerando que en **el instante t = 70mseg se pulsa SW1 y se mantiene pulsado hasta el instante t=210mseg, y que en el instante t = 130mseg se pulsa SW2 y se mantiene pulsado hasta el instante t=170mseg**. (4 ptos.)

NOTA: VREF_P=3'3V, VREF_N=0V y CCLK=100MHz.



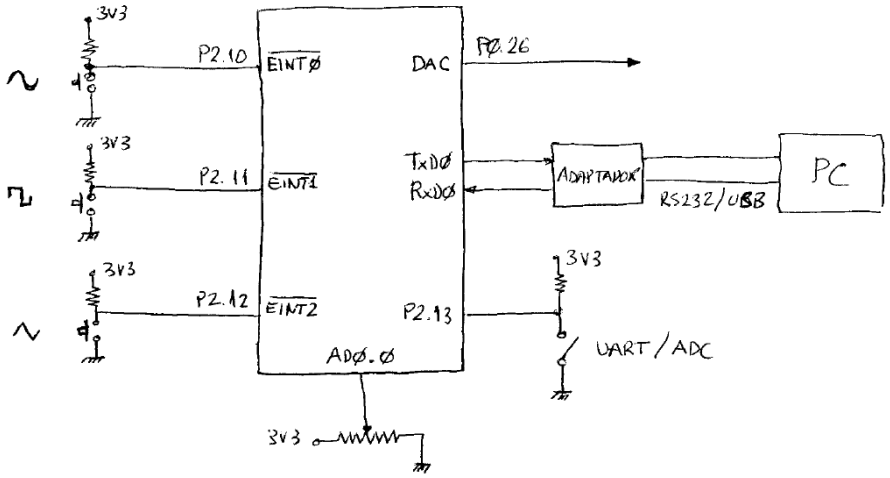


Ejercicio 2 (15 puntos)

Se pretende diseñar un generador de señal senoidal, triangular o cuadrada, de 32 muestras por periodo, basado en el LPC1768, utilizando el DAC y un timer que interrumpe periódicamente para generar las muestras. El tipo de señal se selecciona pulsando uno de tres pulsadores. La frecuencia de la señal de salida deseada, en el rango de 100Hz-10kHz, se introducirá por el puerto serie (UART0) validando la entrada con enter, o en función de la posición de un potenciómetro mientras se mantiene pulsado un pulsador adicional.

NOTA: Considere ya escritas las funciones de control de la UART (incluidas en el Anexo 2).

a) Dibuje el diagrama hardware del generador señalando claramente las entradas o salidas utilizadas del LPC1768. (2 ptos.)



b) Escriba la(s) función(es) de atención a la interrupción de los 3 pulsadores que permiten seleccionar el tipo de señal. (3 ptos.)

```
void EINT0_IRQHander(void)
{
   uint8_t i;
   LPC_SC->EXINT=(1<<0); // Borrar flag Externa 0
   for(i=0;i<32;i++) muestras[i]=(uint16_t)(511+511*sin(2*pi*i/32));
}
void EINT1_IRQHander(void)
{
   uint8_t i;
   LPC_SC->EXINT=(1<<1); // Borrar flag Externa 1
   for(i=0;i<16;i++) muestras[i]=1023;
```

c) Complete la función de configuración del ADC incluida a continuación, indicando cuál es la frecuencia de muestreo configurada. Comente el código. (3 ptos.)

```
void init_ADC(void)
{
   LPC_SC->PCONP|= (1<<12);          // Power ON
   LPC_PINCON->PINSEL1 |=(1<<14);    // P0.23 entrada AD0.0 (canal 0)
   LPC_PINCON->PINMODE1|=(2<<20);    // sin pull-up y pull down
   LPC_ADC->ADCR=  (0x01<< 0)|       // Canal 0
                   (249<<8)|         // F_clk_adc= 25e6/(249+1)=100 KHz
                    (1<<21)|         // PDN=1
                    (1<<16);         // Modo Burst
   LPC_ADC->ADINTEN=0;               //Sin interrupciones
}
```

Fs=F_clk_adc/65= 1538,4 Hz

d) Complete la función de configuración del TIMER incluida a continuación, que se usa para llevar las muestras al DAC, considerando que se ha declarado la variable global **frec_deseada**. Comente el código. (3 ptos.)

```
void init_TIMER1 (void)
{
   LPC_SC->PCONP|=(1<<2);     // Power ON Timer 1
   LPC_TIM1 ->PR = 0x00;      // Prescaler ÷1
   LPC_TIM1 ->MCR = 0x03;     // Int. en Match y reset timer
   LPC_TIM1 ->MR0 =(F_pclk/frec_deseada/32)-1;
   LPC_TIM1 ->EMR = 0x01;     // External Match 0
   LPC_TIM1 ->TCR = 0x01;     // Start timer
   NVIC_SetPriority(TIMER1_IRQn, 0); // Nivel 0 (más alta prioridad)
   NVIC_EnableIRQ(TIMER1_IRQn);      // Hab. Int. Timer 1
}
```

e) Escriba el código del programa principal que actualiza continuamente el valor de la frecuencia de la señal en función del valor leído del ADC (con resolución de 8 bits) o de la entrada recibida por la UART. (3 ptos.)

```
while(1)
{
     if(LPC_GPIO2->FIOPIN&(1<<13)) // P2.13=0?
     {
             frec_deseada=100+(LPC_ADC-ADR0>>8)&0xFF)*9900/255;
             LPC_TIM1->MR0=(F_pclk/frec_deseada/32)-1;
     }
     if(rx_completa==1)
     {
             rx_completa=0;
             frec_deseada=atoi(buffer);       // convertimos a entero
             LPC_TIM1->MR0=(F_pclk/frec_deseada/32)-1;
      {
}
```

Ejercicio 3 (6 puntos)

Se desea diseñar un sistema digital para una aplicación de control de producción de una fábrica, con un microprocesador de 16 bits con bus de direcciones A[1..23] que trabaja con ordenación big endian y cuenta con las señales de control $\overline{BHE}$ (Bus High Enable, validación de parte alta del dato) y $\overline{BLE}$ (Bus Low Enable, validación de parte baja del dato). Las necesidades de mapeo del sistema a diseñar se muestran a continuación:

- 10Mbytes para la aplicación y tablas de datos constantes.
- 4Mbytes para datos variables y pila.
- 2 espacios de 1Mbyte para posibles ampliaciones periféricos externos de 8 bits.

Para realizar el mapeo se debe tener en cuenta que no se pueden dejar espacios libres entre los distintos segmentos y que se puede usar cualquier circuito de memoria comercial.

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

(3 ptos.)

	Funcionalidad	Direcciones (Inicio-Fin)	Chips a implementar
RAM	4Mbytes de DATOS VARIABLES Y PILA	Inicio: 0x000000 Fin: 0x3FFFFF	2 de 2M8
ROM	10Mbytes para APLICACIÓN Y TABLAS DE DATOS CONSTANTES	Inicio: 0x400000 Fin: 0xDFFFFF	2 de 8M8
IO	2 espacios diferentes de 1Mbyte PERIFÉRICOS DE 8 BITS	Inicio: 0xE00000 Fin: 0xFFFFFE	X
		Inicio: 0xE00001 Fin: 0xFFFFFF	

Como el mapa es para un microprocesador de 16 bits se organiza en 2 bancos, incluyendo el área de periféricos que, al ser para periféricos de 8 bits, se organiza ocupando los 2 bancos diferentes.

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 volátil. (3 ptos.)

Construimos una tabla de decodificación simplificada (solo nibble alto de direcciones) según el mapa anterior, y diseñamos el circuito de decodificación (con un decodificador de 3 a 8 p.ej.) a partir de esta tabla:

A23	A22	A21	CS
0	0	X	$\overline{CSRAM}$

el direccionamiento es único (1 dirección sola lógica por cada dirección física)

Ejercicio 4 (11 puntos)

Se desea implementar un espacio de memoria externa para el LPC1788 (CCLK=100MHz) con las siguientes características:

- El espacio incluye una zona de Flash en el rango 0x80000000-0x800FFFFF, y otro de RAM a partir de la dirección 0x90000000 de doble capacidad.
- Para implementar el espacio de Flash se usan dispositivos AT29C010A-12.
- Para implementar el espacio de RAM se usan dispositivos IS62WV25616ALL-55.
- Se desea además conectar un periférico de entrada de 16 bits y 64 registros internos a partir de la dirección 0x9C000000.

NOTA: Las características de las memorias se adjuntan en el Anexo 3.

Se pide:

a) Indique cuantos dispositivos de memoria Flash y RAM son necesarios para implementar el espacio deseado y con qué tipo de ampliación u ordenación sería necesario montarlos, suponiendo que se desea minimizar el tiempo de acceso a datos de 32 bits. (2 ptos.)

AT29C010A-12: chip Flash de 128k8 según su patillaje (bus direcciones, control y datos)
IS62WV25616ALL-55: chip RAM de 256k16 según su patillaje (bus direcciones, control y datos)

Flash: Necesario 1M8 organizado en 4 bancos, por lo que se usan 8 chips, 2 por banco en ampliación en número de palabras, y luego una organización en bancos de los 4 pares (4x2=8 chips)

RAM: Necesarios 2M8 organizados en 4 bancos, por lo que se usan 4 chips (mapeando cada chip en 2 bancos: 4N-4N+1 o 4N+1-4N+3), 2 por cada par de bancos en ampliación en número de palabras, y luego una organización en bancos de los 2 pares (4x2=8 chips)

b) Configure, justificadamente, los registros STATICCONFIG necesarios para generar las señales de **chip select** necesarias. (3 ptos.)

Por las direcciones de mapeo de los 3 espacios, y analizando el manual de usuario del módulo EMC del microcontrolador, la asignación de los espacios al mapa del Cortex es la siguiente:

- Flash mapeada en rango 0x80000000-0x800FFFFF, a través de EMC_CS0, configurado en 32 bits
- RAM mapeada en rango 0x90000000-0x901FFFFF, a través de EMC_CS1, configurado en 32 bits
- Flash mapeada en rango 0x9C000000-0x9C00007F, a través de EMC_CS3, configurado en 16 bits

Se configura, por tanto, los registros STATICCONFIG correspondientes de acuerdo con esto:

c) Configure, justificadamente, **los registros STATICWAITRD** para que los tiempos de selección del microcontrolador se adapten a la velocidad de acceso en lectura de las memorias RAM y Flash. (3 ptos.)

El tiempo de selección del microcontrolador según los cronogramas del EMC es RD_5, cuyo valor es:
$t_{sel\_min} = RD_{5\_min} = (STATICWAITRDx + STATICWAITOEx +1) \times t_{CCLK} - 8.6$ ns;
donde, STATICWAITRDx = 0x1F por defecto tras reset y STATICWAITOEx = 0x00 por defecto tras reset
$t_{sel_min} = 32 \times 10 - 8.6 = 311.4$ns;

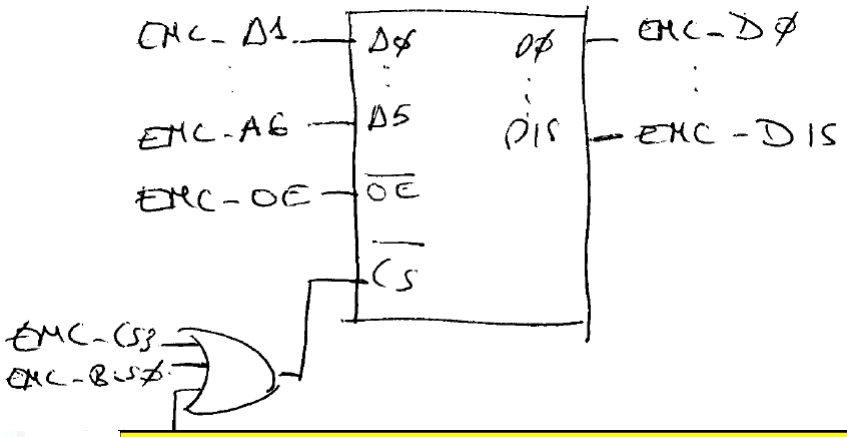
El tiempo de acceso en lectura de las memorias, según sus hojas de carcterísitcas anexas es:
Para la Flash $t_{CE} = t_{ACC} = 120$ns (suponemos el control por OE es menos restrictivo)
Para la RAM $t_{ACE1} = t_{BA} = 55$ns (suponemos el control por OE es menos restrictivo)

Para que los tiempos de selección del microcontrolador se adapten a la velocidad de acceso en lectura de las memorias RAM y Flash conviene modificar el valor de los registros STATICWAITRDx (dejando el de STATICWAITOEx por defecto) de modo que:

Para la Flash $t_{sel\_min} \geq t_{CE}$
$t_{sel\_min} = (STATICWAITRD0 +1) \times 10 - 8.6 \geq 120$ -> STATICWAITRD0 = 11 = 0x0B

Para la Flash $t_{sel\_min} \geq t_{ACE1}$
$t_{sel\_min} = (STATICWAITRD1 +1) \times 10 - 8.6 \geq 55$ -> STATICWAITRD1 = 4 = 0x04

d) Conecte el periférico al microcontrolador según las especificaciones dadas. (3 ptos.)



Anexo 1. Código del programa. (Ejercicio 1)

```
#include <LPC17xx.H>

int indice = 0;
int dato_leido = 0;

void init_EINT0(void)
{
  LPC_PINCON->PINSEL4 |= (0x01<<20);          //P2.10 es entrada interrup. EXT 0
  LPC_SC->EXTMODE |= (1<<1)|(1<<0);           //Por Flanco...
  LPC_SC->EXTPOLAR = 0;                       //de bajada
  NVIC_SetPriority(EINT0_IRQn, 0);
  NVIC_EnableIRQ(EINT0_IRQn);
}

void init_TIMER0(void)
{
  LPC_SC->PCONP |= (1<<1);                    //Power Timer0
  LPC_SC->PCLKSEL0 |= (0x2<<2);               //PCLK = CCLK/2
  LPC_TIM0->MCR = 0x18;                       //Resetea TC en Match_1 e interrumpe
  LPC_TIM0->MR1 = 1000000;
  LPC_TIM0->TCR = 0x01;                       //Enable TC
  NVIC_SetPriority(TIMER0_IRQn,1);
  NVIC_EnableIRQ(TIMER0_IRQn);
}

void init_ADC(void)
{
  LPC_SC->PCONP |= (1<<12);                   //Power ON
  LPC_SC->PCLKSEL0 |= (0x00<<8);              //PCLK = CCLK/4
  LPC_PINCON->PINSEL1 |= (1<<14);             //ADC input = P0.23 (AD0.0)
  LPC_PINCON->PINMODE1 |= (2<<14);            //No resistencia pull-up ni pull-down
  LPC_ADC->ADCR = (0x01<<0)|                  //Canal 0
                  (0x01<<8)|                  //CLKDIV = 1 (Fclk_ADC=25Mhz /(1+1)= 12.5Mhz)
                  (0x01<<21);                 //PDN = 1
  LPC_ADC->ADINTEN = (1<<8);                  //Habilita interrupción
  NVIC_SetPriority(ADC_IRQn,2);
  NVIC_EnableIRQ(ADC_IRQn);
}

void EINT0_IRQHandler(void)
{
  LPC_SC->EXTINT = (1<<0);                    //Borrar flag interrupt.
  indice = 0;
}

void TIMER0_IRQHandler(void)
{
  LPC_TIM0->IR |= (1<<1);                     //Borrar flag interrup.
  LPC_ADC->ADCR |= (1<<24);                   //Inicio de conversión del ADC
}

void ADC_IRQHandler(void)
{
  dato_leido = ((LPC_ADC->ADGDR >>4)&0xFFF)/0xFFF;   //Se borra automat. el flag DONE al leer ADCGDR
  indice |= dato_leido;
}
```

Anexo 2. Funciones de control de la UART (Ejercicio 2)

```
/* uart.c
 * contiene las funciones:
 1  UART0_IRQHandler(void)
 2  tx cadena UART0(char *ptr)
 3  uart0_set_baudrate(unsigned int baudrate)
 4  uart0_init(int baudrate)
*/
#include <LPC17xx.h>
#include "uart.h"

// UART0 interrupt handler
void UART0_IRQHandler(void)
{
    switch(LPC_UART0->IIR&0x0E) {

        case 0x04:                          // RBR, Receiver Buffer Ready
        *ptr_rx=LPC_UART0->RBR;             // lee el dato recibido y lo almacena
            if (*ptr_rx++ ==13)             // Caracter return --> Cadena completa
                {
                        *ptr_rx=0;          // Añadimos null para tratar los datos recibidos como una cadena
                        rx_completa = 1;    // rx completa
                        ptr_rx=buffer;      // puntero al inicio del buffer para nueva recepción
                }
                break;

        case 0x02:                          // THRE, Transmit Holding Register empty
                if (*ptr_tx!=0) LPC_UART0->THR=*ptr_tx++;   // carga un nuevo dato para ser transmitido
                else tx_completa=1;
                break;
    }
}

// Función para enviar una cadena de texto
// El argumento de entrada es la dirección de la cadena, o directamente la cadena de texto entre comillas
void tx_cadena_UART0(char *cadena)
{
        ptr_tx=cadena;
        tx_completa=0;
        LPC_UART0->THR=*ptr_tx++;      // IMPORTANTE: Introducir un carácter para iniciar TX o activar flag IRQ
}

// Función para INICIALIZAR LA UART0
void uart0_init(int baudrate)
{

    LPC_PINCON->PINSEL0|=(1<<4)|(1<<6);             // Change P0.2 and P0.3 mode to TXD0 and RXD0

    LPC_UART0->LCR &= ~STOP_1_BIT & ~PARITY_NONE;   // Set 8N1 mode (8 bits/dato, sin pariad, y 1 bit de stop)
    LPC_UART0->LCR |= CHAR_8_BIT;

    uart0_set_baudrate(baudrate);                   // Set the baud rate (INSERTED IN UART.H)

    LPC_UART0->IER = THRE_IRQ_ENABLE|RBR_IRQ_ENABLE;// Enable UART TX and RX interrupt (for LPC17xx UART)
    NVIC_EnableIRQ(UART0_IRQn);                     // Enable the UART interrupt (for Cortex-CM3 NVIC)

}
```

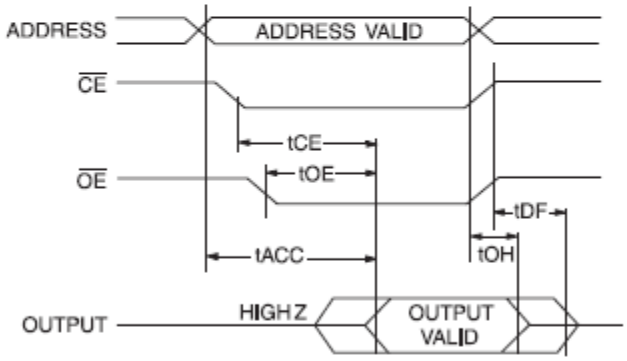
Anexo 3. Características de Memorias (Ejercicio 3)

AT29C010A-12

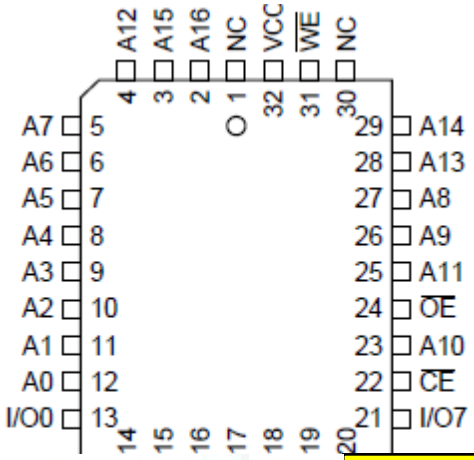
AC Read Characteristics

Symbol	Parameter	AT29C010A-70		AT29C010A-90		AT29C010A-12		AT29C010A-15		Units
		Min	Max	Min	Max	Min	Max	Min	Max	
t_{ACC}	Address to Output Delay		70		90		120		150	ns
t_{CE} (1)	$\overline{CE}$ to Output Delay		70		90		120		150	ns
t_{OE} (2)	$\overline{OE}$ to Output Delay	0	35	0	40	0	50	0	70	ns
t_{DF} (3)(4)	$\overline{CE}$ or $\overline{OE}$ to Output Float	0	25	0	25	0	30	0	40	ns
t_{OH}	Output Hold from $\overline{OE}$, $\overline{CE}$ or Address, whichever occurred first	0		0		0		0		ns

AC Read Waveforms (1)(2)(3)(4)



Notes:
1. $\overline{CE}$ may be delayed up to $t_{ACC} - t_{CE}$ after the address transition without impact on t_{ACC}.
2. $\overline{OE}$ may be delayed up to $t_{CE} - t_{OE}$ after the falling edge of $\overline{CE}$ without impact on t_{CE} or by $t_{ACC} - t_{OE}$ after an address change without impact on t_{ACC}.
3. t_{DF} is specified from $\overline{OE}$ or $\overline{CE}$ whichever occurs first (CL = 5 pF).
4. This parameter is characterized and is not 100% tested.



Pin Name	Function
A0 - A16	Addresses
$\overline{CE}$	Chip Enable
$\overline{OE}$	Output Enable
$\overline{WE}$	Write Enable
I/O0 - I/O7	Data Inputs/Outputs
NC	No Connect

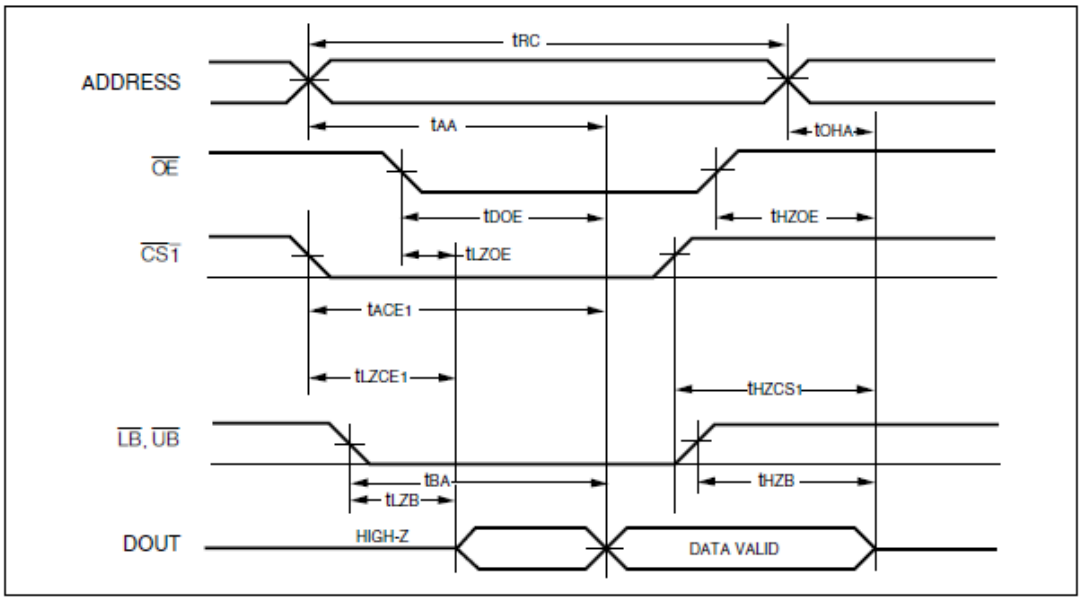
IS62WV25616ALL-55

READ CYCLE SWITCHING CHARACTERISTICS[1] (Over Operating Range)

Symbol	Parameter	55 ns Min.	55 ns Max.	70 ns Min.	70 ns Max.	Unit
t_{RC}	Read Cycle Time	55	—	70	—	ns
t_{AA}	Address Access Time	—	55	—	70	ns
t_{OHA}	Output Hold Time	10	—	10	—	ns
t_{ACS1}	$\overline{CS1}$ Access Time	—	55	—	70	ns
t_{DOE}	$\overline{OE}$ Access Time	—	25	—	35	ns
t_{HZOE}[2]	$\overline{OE}$ to High-Z Output	—	20	—	25	ns
t_{LZOE}[2]	$\overline{OE}$ to Low-Z Output	5	—	5	—	ns
t_{HZCS1}	$\overline{CS1}$ to High-Z Output	0	20	0	25	ns
t_{LZCS1}	$\overline{CS1}$ to Low-Z Output	10	—	10	—	ns
t_{BA}	$\overline{LB}$, $\overline{UB}$ Access Time	—	55	—	70	ns
t_{HZB}	$\overline{LB}$, $\overline{UB}$ to High-Z Output	0	20	0	25	ns
t_{LZB}	$\overline{LB}$, $\overline{UB}$ to Low-Z Output	0	—	0	—	ns

Notes:
1. Test conditions assume signal transition times of 5 ns or less, timing reference levels of 0.9V/1.5V, input pulse levels of 0.4 to V_{DD}-0.2V/V_{DD}-0.3V and output loading specified in Figure 1.
2. Tested with the load in Figure 2. Transition is measured ±500 mV from steady-state voltage. Not 100% tested.

READ CYCLE NO. 2[1,3] ($\overline{CS1}$, $\overline{OE}$, AND $\overline{UB}$/$\overline{LB}$ Controlled)



Notes:
1. $\overline{WE}$ is HIGH for a Read Cycle.
2. The device is continuously selected. $\overline{OE}$, $\overline{CS1}$, $\overline{UB}$, or $\overline{LB}$ = V_{IL}. $\overline{WE}$ = V_{IH}.
3. Address is valid prior to or coincident with $\overline{CS1}$ LOW transition.

VDD	Power
GND	Ground