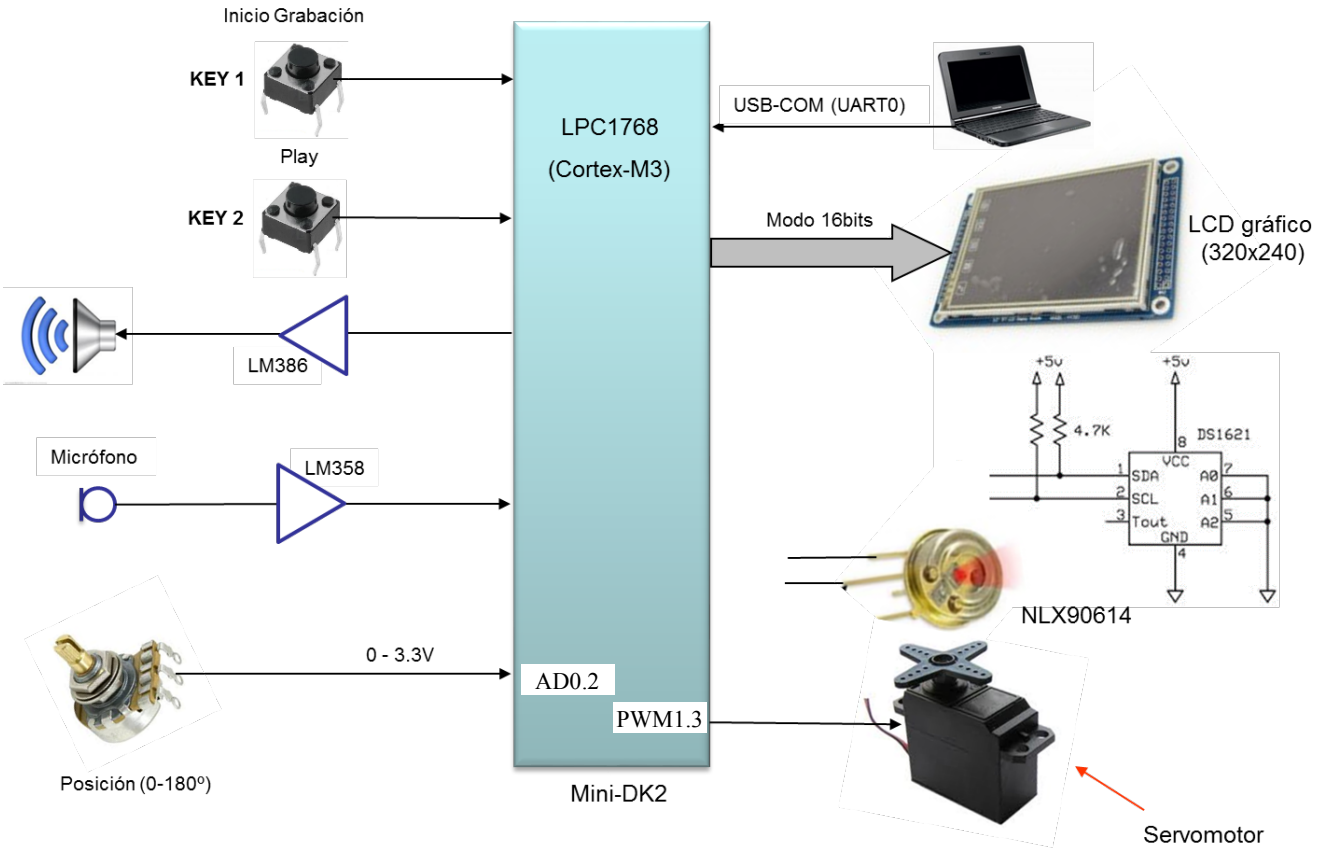
UNIVERSIDAD DE ALCALÁ ESCUELA POLITÉCNICA SUPERIOR DEPARTAMENTO DE ELECTRÓNICA		GRADO EN INGENIERÍA ELECTRÓNICA DE COMUNICACIONES	
ASIGNATURA	SISTEMAS ELECTRÓNICOS DIGITALES AVANZADOS	FECHA	MAYO 2016
APELLIDOS, NOMBRE		GRUPO	

PRUEBA DE EVALUACIÓN FINAL (PARTE I)

Ejercicio 1

El diagrama de la figura muestra un sistema empotrado basado en el LPC1768 para realizar un sistema de medida de la temperatura de una zona de un entorno a partir de un sensor de IR (MLX90614) acoplado a un servomotor, cuya posición se controla manualmente con ayuda de un potenciómetro. El sistema también es capaz de medir la temperatura exterior a partir de un sensor digital (DS1621). Un LCD muestra la información a la vez que un terminal de comunicación conectado a la UART0 permite configurar el sistema y monitorizar las variables.



Condiciones de diseño:

- El valor de la tensión analógica que proporciona el potenciómetro se ha de tomar **periódicamente cada 50ms**.
- Si la temperatura del sensor IR supera un **umbral** (programable) se ha de activar una señal de alarma acústica a través del altavoz indicando de forma verbal el fonema "Alarma" repitiéndose de manera periódica cada **5 segundos**.
- Un menú de configuración permitirá seleccionar una opción para grabar en RAM el mensaje de alarma utilizando un micrófono con el correspondiente circuito acondicionador. La pulsación de KEY 1 iniciará el proceso de grabación a una frecuencia de muestreo de 8 kHz durante 2 segundos. La pulsación de KEY 2 permitirá reproducir el mensaje grabado.
- **Considere que el SysTick está habilitado y que interrumpe periódicamente cada 50ms**, entre otras cosas, para para hacer medidas de tiempos grandes.
- Considere una Fcpu=100MHz y Fpclk=25MHz

a) Complete sobre el diagrama de la figura1, el nombre del pin (**Pn.x**) sobre la línea de conexión y el nombre del recurso utilizado dentro del bloque que representa la Mini-DK2 (ej. **MAT1.0**) excepto el LCD.

b) Complete la función de configuración del ADC (incluyendo los comentarios) para la entrada a la que se conecta el potenciómetro e indique la **frecuencia de muestreo** del canal seleccionado y el **tiempo de conversión** (T1MR0=100000).

```
void init_ADC_potencimetro(void)
{
LPC_SC->PCONP|= (1<<12);             // Power ON
LPC_PINCON->PINSEL1|= (       );     //
LPC_PINCON->PINMODE1|=(       );     // Deshabilita pullup/pulldown
LPC_ADC->ADCR= (    1<< )|           //
                  (24<<8)|           //
                  (1<<21)|           // PDN=1
                   (6<<24);          //
NVIC_DisableIRQ(ADC_IRQn);           //
}
```

c) Complete la función de configuración de la señal PWM (incluyendo los comentarios), y de actualización de la posición del servo en función de la posición del potenciómetro. Considere que el periodo sea de **20ms**, y el tiempo a nivel alto varíe entre **0.8-2.4ms**, para un movimiento de su posición entre 0º y 180º.

```
void config_pwm(void)
{
LPC_PINCON->PINSEL |=(  <<  );
LPC_SC->PCONP|=(1<<6);  //Power PMW module
LPC_PWM1->MR0=
LPC_PWM1->PCR|=
LPC_PWM1->MCR|=
LPC_PWM1->TCR|=
}

void set_servo(void)
{
LPC_PWM1->MR  =
LPC_PWM1->LER|=
}
```

d) Escriba las funciones de lectura de la temperatura externa del DS1621 (considere ya configurado) y del sensor de temperatura IR (MLX90614), a partir de la información del anexo III y IV.
NOTA: Considere para simplificar el código que se desea una **resolución de un grado**, y que ya están escritas las funciones del bus I2C.

```
void I2CSendByte(unsigned char byte);
void I2CSendAddr(unsigned char addr, unsigned char rw);
unsigned char I2CGetByte(unsigned char ACK);
void I2CSendStop(void);
```

char temperatura_DS1621(void)
{

}

char temperatura_NLX(void)
{

}

e) Escriba la función de configuración del ADC para muestrear la señal de audio procedente del micrófono e indique el valor de T1MR0 (Fs=8kHz). Considere que la transferencia de las muestras a memoria se hará por DMA.

```
void init_ADC_microfono(void)
{
LPC_SC->PCONP|= (1<<12);              // Power ON
LPC_PINCON->PINSEL1|= (       );      //
LPC_PINCON->PINMODE1|=(       );      // Deshabilita pullup/pulldown
LPC_ADC->ADCR= (    1<< )|            //
                  (24<<8)|            //
                  (1<<21)|            // PDN=1
                   (6<<24);           //

}
```

f) Explique detalladamente la configuración del DMA para el modo de grabación de la señal de audio considerando que se han de almacenar en memoria las muestras del ADC de 12 bits. Considere la información del **Anexo I**

g) ¿Qué condiciones y/o parámetros limitan la duración máxima del mensaje de alarma?

h) Escriba en pseudocódigo la función de interrupción del DMA en el modo de grabación de la señal de audio.

i) Escriba la función de interrupción del Timer 2 que saca las muestras hacia el DAC para generar la señal de alarma, y el valor del registro **MRx** correspondiente para obtener la señal de salida correctamente.

```
void TIMER3_IRQHandler(void)
{




}
```

j) Escriba el código o pseudocódigo de las funciones de interrupción asociadas a los pulsadores Key 1 y Key 2.

k) Explique qué recurso utilizaría y como lo configuraría para reducir la carga de CPU durante la **generación de la señal de alarma**.

l) Escriba el código de la función de interrupción del **SysTick**, encargada de **posicionar el servo** en función de la posición del potenciómetro, de **mostrar por el LCD** y de **enviar por el puerto serie** los valores de la temperatura del sensor IR y la temperatura externa cada segundo.

```
void SysTick_IRQHandler(void)
{









}
```

m) Completa la función **simplificada** de configuración de la velocidad del puerto serie (considerando FDR=ResetValue) para una velocidad de **38400 baudios** y calcula la **velocidad real** de la comunicación y el **tiempo que tarda en transmitirse un carácter**.

```
void uart0_set_baudrate(int baudrate)
{
LPC_UART0->LCR |= DLAB_ENABLE;
LPC_UART0->DLM =
LPC_UART0->DLL =
LPC_UART0->LCR &= ~DLAB_ENABLE;
}
```

Table 285: UARTn Fractional Divider Register (U0FDR - address 0x4000 C028, U2FDR - 0x4009 8028, U3FDR - 0x4009 C028) bit description

Bit	Function	Value	Description	Reset value
3:0	DIVADDVAL	0	Baud-rate generation pre-scaler divisor value. If this field is 0, fractional baud-rate generator will not impact the UARTn baudrate.	0
7:4	MULVAL	1	Baud-rate pre-scaler multiplier value. This field must be greater or equal 1 for UARTn to operate properly, regardless of whether the fractional baud-rate generator is used or not.	1
31:8	-	NA	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	0

$$UARTn_{baudrate} = \frac{PCLK}{16 \times (256 \times UnDLM + UnDLL) \times \left(1 + \frac{DivAddVal}{MulVal}\right)}$$

ANEXO I (Mapa de memoria)

Table 3. LPC176x/5x memory usage and details

Address range	General Use	Address range details and description	
0x0000 0000 to 0x1FFF FFFF	On-chip non-volatile memory	0x0000 0000 - 0x0007 FFFF	For devices with 512 kB of flash memory.
		0x0000 0000 - 0x0003 FFFF	For devices with 256 kB of flash memory.
		0x0000 0000 - 0x0001 FFFF	For devices with 128 kB of flash memory.
		0x0000 0000 - 0x0000 FFFF	For devices with 64 kB of flash memory.
		0x0000 0000 - 0x0000 7FFF	For devices with 32 kB of flash memory.
	On-chip SRAM	0x1000 0000 - 0x1000 7FFF	For devices with 32 kB of local SRAM.
		0x1000 0000 - 0x1000 3FFF	For devices with 16 kB of local SRAM.
		0x1000 0000 - 0x1000 1FFF	For devices with 8 kB of local SRAM.
	Boot ROM	0x1FFF 0000 - 0x1FFF 1FFF	8 kB Boot ROM with flash services.
0x2000 0000 to 0x3FFF FFFF	On-chip SRAM (typically used for peripheral data)	0x2007 C000 - 0x2007 FFFF	AHB SRAM - bank 0 (16 kB), present on devices with 32 kB or 64 kB of total SRAM.
		0x2008 0000 - 0x2008 3FFF	AHB SRAM - bank 1 (16 kB), present on devices with 64 kB of total SRAM.
	GPIO	0x2009 C000 - 0x2009 FFFF	GPIO.
0x4000 0000 to 0x5FFF FFFF	APB Peripherals	0x4000 0000 - 0x4007 FFFF	APB0 Peripherals, up to 32 peripheral blocks, 16 kB each.
		0x4008 0000 - 0x400F FFFF	APB1 Peripherals, up to 32 peripheral blocks, 16 kB each.
	AHB peripherals	0x5000 0000 - 0x501F FFFF	DMA Controller, Ethernet interface, and USB interface.
0xE000 0000 to 0xE00F FFFF	Cortex-M3 Private Peripheral Bus	0xE000 0000 - 0xE00F FFFF	Cortex-M3 related functions, includes the NVIC and System Tick Timer.

Table 530. ADC registers

Generic Name	Description	Access	Reset value[1]	AD0 Name & Address
ADCR	A/D Control Register. The ADCR register must be written to select the operating mode before A/D conversion can occur.	R/W	1	AD0CR - 0x4003 4000
ADGDR	A/D Global Data Register. This register contains the ADC's DONE bit and the result of the most recent A/D conversion.	R/W	NA	AD0GDR - 0x4003 4004
ADINTEN	A/D Interrupt Enable Register. This register contains enable bits that allow the DONE flag of each A/D channel to be included or excluded from contributing to the generation of an A/D interrupt.	R/W	0x100	AD0INTEN - 0x4003 400C
ADDR0	A/D Channel 0 Data Register. This register contains the result of the most recent conversion completed on channel 0.	RO	NA	AD0DR0 - 0x4003 4010
ADDR1	A/D Channel 1 Data Register. This register contains the result of the most recent conversion completed on channel 1.	RO	NA	AD0DR1 - 0x4003 4014
ADDR2	A/D Channel 2 Data Register. This register contains the result of the most recent conversion completed on channel 2.	RO	NA	AD0DR2 - 0x4003 4018
ADDR3	A/D Channel 3 Data Register. This register contains the result of the most recent conversion completed on channel 3.	RO	NA	AD0DR3 - 0x4003 401C
ADDR4	A/D Channel 4 Data Register. This register contains the result of the most recent conversion completed on channel 4.	RO	NA	AD0DR4 - 0x4003 4020
ADDR5	A/D Channel 5 Data Register. This register contains the result of the most recent conversion completed on channel 5.	RO	NA	AD0DR5 - 0x4003 4024
ADDR6	A/D Channel 6 Data Register. This register contains the result of the most recent conversion completed on channel 6.	RO	NA	AD0DR6 - 0x4003 4028
ADDR7	A/D Channel 7 Data Register. This register contains the result of the most recent conversion completed on channel 7.	RO	NA	AD0DR7 - 0x4003 402C
ADSTAT	A/D Status Register. This register contains DONE and OVERRUN flags for all of the A/D channels, as well as the A/D interrupt/DMA flag.	RO	0	AD0STAT - 0x4003 4030
ADTRM	ADC trim register.	R/W	0x0000 0F00	AD0TRM - 0x4003 4034

ANEXO II (Funciones de control del puerto serie y LCD)

```
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 el caracter 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;
   }
}
```

```
void uart0_init(int baudrate) {

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

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

    uart0_set_baudrate(baudrate);                 // Set the baud rate

    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)

}
```

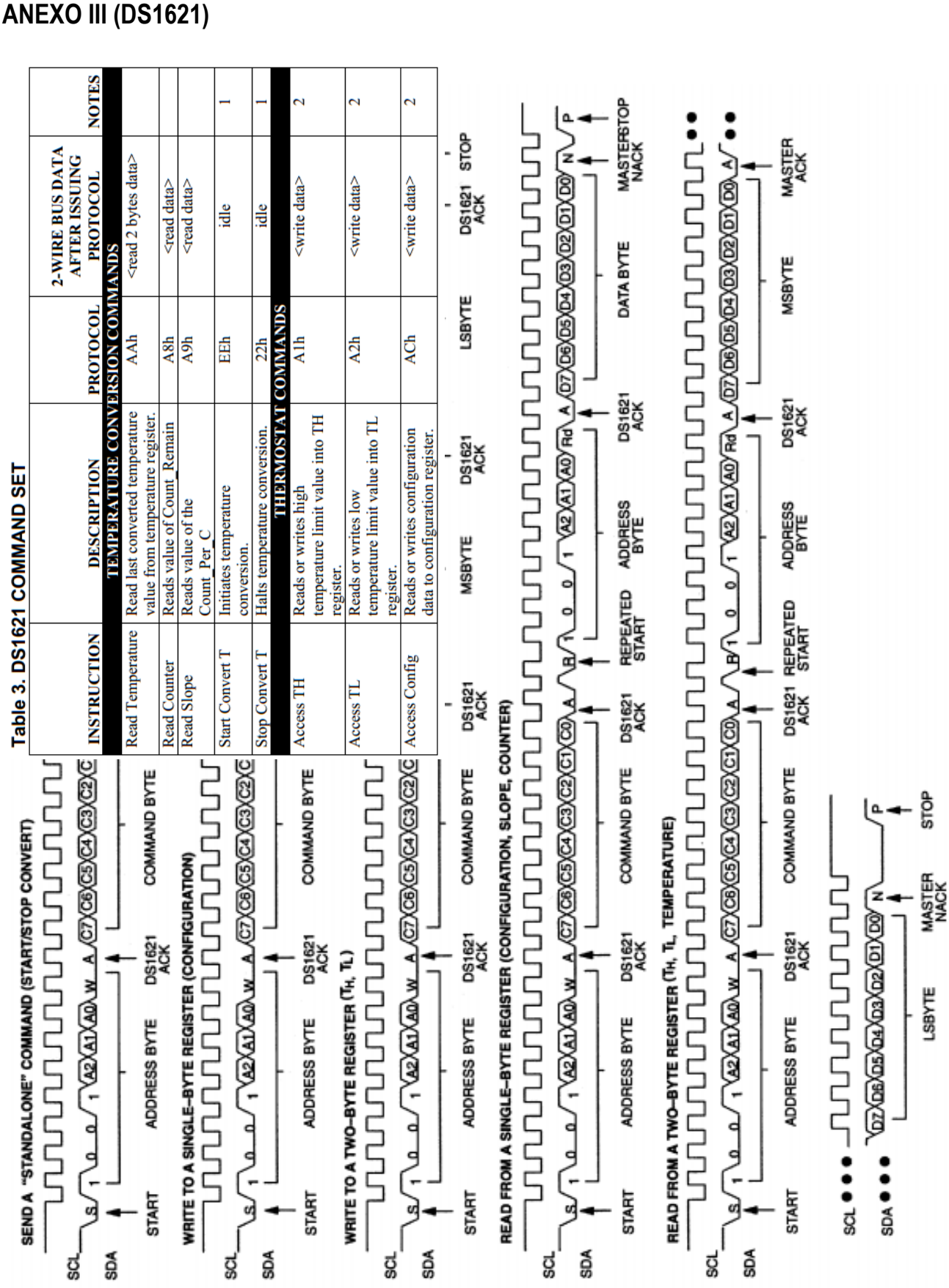
```
void tx_cadena_UART0(char *cadena)
{
   ptr_tx=cadena;
   tx_completa=0;
   LPC_UART0->THR = *ptr_tx;   // IMPORTANTE: Introducir un carácter al comienzo para iniciar TX o
}                              // activar flag interrupción por registro transmisor vacio
```

```
/* Private function prototypes -----------------------------------------------*/
void LCD_Initializtion(void);
void LCD_Clear(uint16_t Color);
uint16_t LCD_GetPoint(uint16_t Xpos,uint16_t Ypos);
void LCD_SetPoint(uint16_t Xpos,uint16_t Ypos,uint16_t point);
void LCD_DrawLine( uint16_t x0, uint16_t y0, uint16_t x1, uint16_t y1 , uint16_t color );
void PutChar( uint16_t Xpos, uint16_t Ypos, uint8_t ASCI, uint16_t charColor, uint16_t bkColor );
void GUI_Text(uint16_t Xpos, uint16_t Ypos, uint8_t *str,uint16_t Color, uint16_t bkColor);
void PutChinese(uint16_t Xpos,uint16_t Ypos,uint8_t *str,uint16_t Color,uint16_t bkColor);
void GUI_Chinese(uint16_t Xpos, uint16_t Ypos, uint8_t *str,uint16_t Color, uint16_t bkColor);
```

ANEXO III (DS1621)

Table 3. DS1621 COMMAND SET

INSTRUCTION	DESCRIPTION	PROTOCOL	2-WIRE BUS DATA AFTER ISSUING PROTOCOL	NOTES
TEMPERATURE CONVERSION COMMANDS				
Read Temperature	Read last converted temperature value from temperature register.	AAh	<read 2 bytes data>	
Read Counter	Reads value of Count_Remain	A8h	<read data>	
Read Slope	Reads value of the Count_Per_C	A9h	<read data>	
Start Convert T	Initiates temperature conversion.	EEh	idle	1
Stop Convert T	Halts temperature conversion.	22h	idle	1
THERMOSTAT COMMANDS				
Access TH	Reads or writes high temperature limit value into TH register.	A1h	<write data>	2
Access TL	Reads or writes low temperature limit value into TL register.	A2h	<write data>	2
Access Config	Reads or writes configuration data to configuration register.	ACh	<write data>	2



ANEXO IV (MLX90614)

Slave address	SA	Factory default		5Ah		hex

RAM (32x17)		
Name	Address	Read access
Melexis reserved	**000h**	**Yes**
…	**…**	**…**
Melexis reserved	**005h**	**Yes**
T_A	**006h**	**Yes**
T_{OBJ1}	**007h**	**Yes**
T_{OBJ2}	**008h**	**Yes**
Melexis reserved	**009h**	**Yes**
…	**…**	**…**
Melexis reserved	**01Fh**	**Yes**

$$To[°K] = Toreg \times 0.02$$, or 0.02°K / LSB.

7.4.3.1.1 Read Word (depending on the command – RAM or EEPROM)

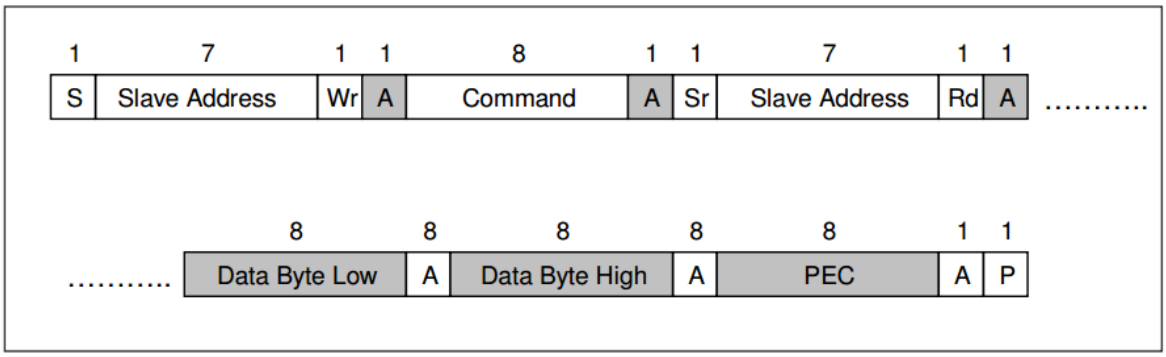


Figure 5: SMBus read word format

7.4.6 Commands

In application mode RAM and EEPROM can be read both with 32x16 sizes. If the RAM is read, the data are divided by two, due to a sign bit in RAM (for example, T_{OBJ1} - RAM address 0x07h will sweep between 0x27ADh to 0x7FFF as the object temperature rises from -70.01°C to +382.19°C). The MSB read from RAM is an error flag (active high) for the linearized temperatures (T_{OBJ1}, T_{OBJ2} and T_a). The MSB for the raw data (e.g. IR sensor1 data) is a sign bit (sign and magnitude format).

Opcode	Command
000x xxxx*	RAM Access
001x xxxx*	EEPROM Access
1111_0000**	Read Flags
1111_1111	Enter SLEEP mode

Note: The xxxxx are the 5 LSBits of the memory map address to be read/written.*
*Note**: Behaves like read command. The MLX90614 returns PEC after 16 bits data of which only 4 are meaningful and if the MD wants it, it can stop the communication after the first byte. The difference between read and read flags is that the latter does not have a repeated start bit.*

Flags read are:

Data[15] – EEBUSY – the previous write/erase EEPROM access is still in progress. High active.
Data[14] – Unused
Data[13] - EE_DEAD – EEPROM double error has occurred. High active.
Data[12] – INIT – POR initialization routine is still ongoing. High active.
Data[11] – not implemented..
Data[10..0] – all zeros.

Flags read is a diagnostic feature. The MLX90614 can be used regardless of these flags.