



Interrupciones en FreeRTOS.



Agenda.

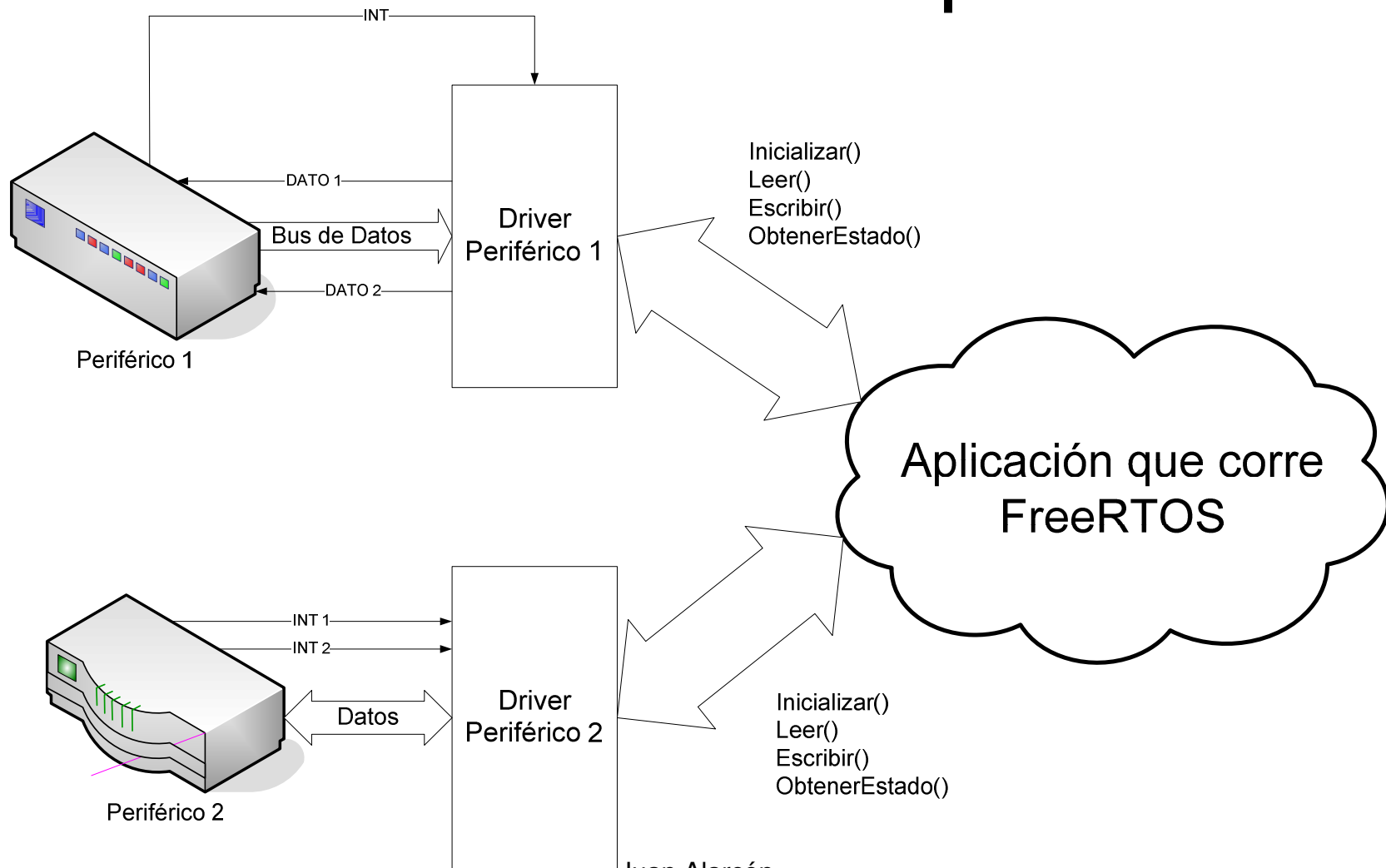
- Concepto de controlador de hardware (drivers).
- Eventos.
- Semáforos en Interrupciones.
- Colas para sincronizar interrupciones
- Anidando interrupciones.
- Ejemplos.
 - Driver I²C. Ejemplo con memoria serie.



Controlador de Hardware. Driver.

- La estrategia propuesta para manejar los diferentes periféricos va a ser la siguiente:
 - Manejar las particularidades del hardware en código bien definido y separado de la lógica del programa (sólo tratar con el periférico, no introducir la lógica propia del programa en el código del driver).
 - Generar interfaces tan generales y claras como sea posible. Por ejemplo, generar funciones: inicializar(), escribir(), leer(), obtenerEstado().
 - Utilizar las primitivas del RTOS para la interfaz del sistema (semáforos, colas de mensajes).
 - En la medida de lo posible, generar interfaces bloqueantes (ser “gentil” avisar cuando se espera por un periférico)

Drivers. Ámbito de aplicación.





Drivers. Objetivo.

- El objetivo principal de escribir el código de los controladores de los diferentes periféricos con interfaces comunes y con código bien delimitado es ***uniformizar el tratamiento de todos (o casi todos) los periféricos.***



Eventos.

- Los sistemas en tiempo real necesitan responder a los eventos generados por su entorno en tiempo conocido y acotado.
- ¿Cómo se implementa?
 - Encuesta del dispositivo (polling).
 - Por interrupciones.



Atención de eventos por interrupciones.

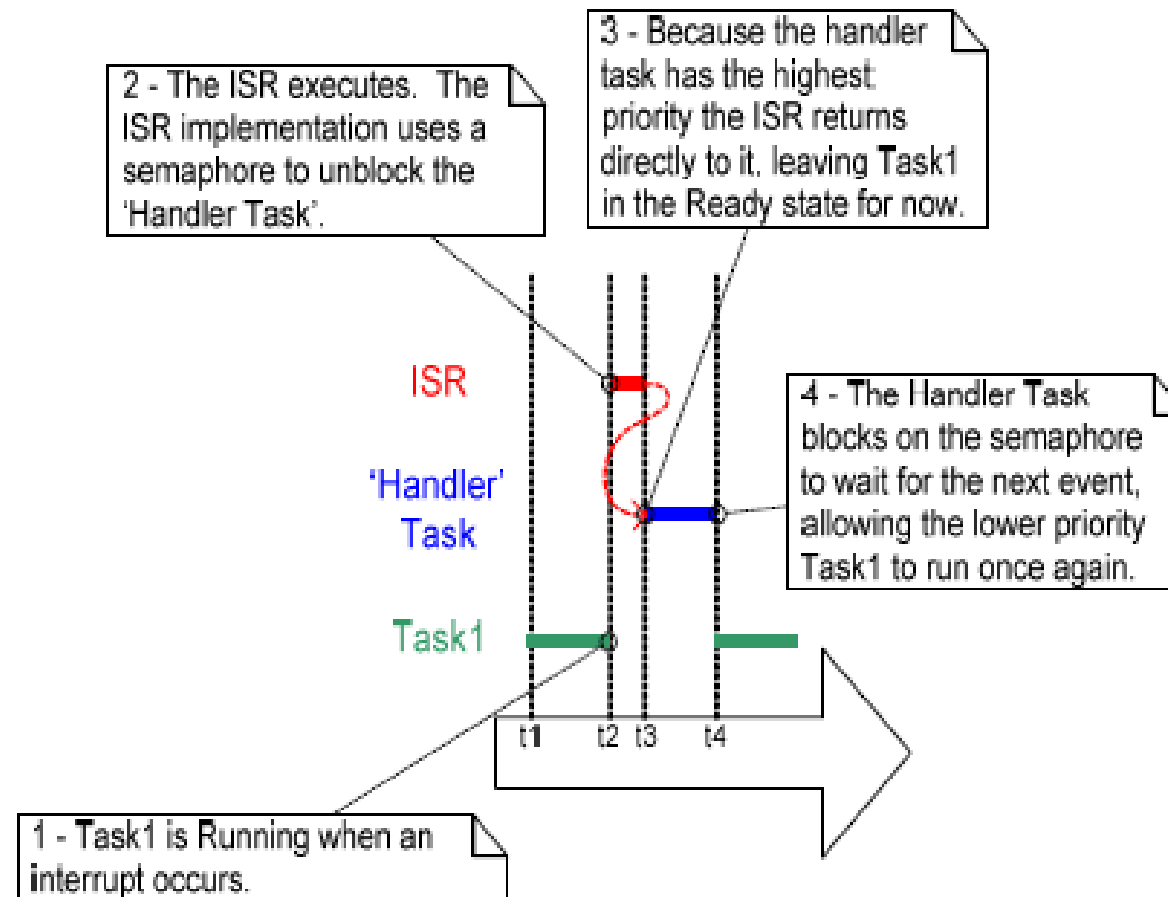
- Al introducir el uso de interrupciones, ¿Cuánto tiempo usar en ellas? ¿Por qué?.
- ¿Cómo comunicar el código de las interrupciones con el del resto del sistema?
- ¿Cómo hacer esta comunicación segura?.



Uso de semáforos en interrupciones.

- Un semáforo binario se lo puede utilizar para sincronizar una tarea con una interrupción.
- Si es necesario procesar con muy baja latencia un evento externo, el código de la interrupción puede desbloquear una tarea de alta prioridad para atenderlo.

Semáforos en interrupciones



Semáforos en interrupciones.

- xSemaphoreGiveFromISR
(
 xSemaphoreHandle xSemaphore,
 signed portBASE_TYPE *pxHigherPriorityTaskWoken
)
- portEND_SWITCHING_ISR
(
 portBASE_TYPE HigherPriorityTaskWoken
)

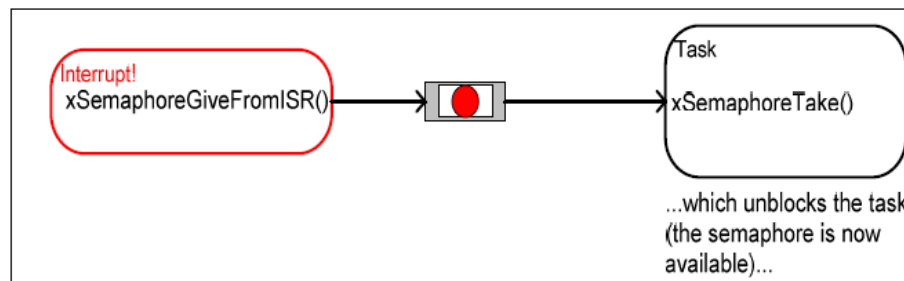
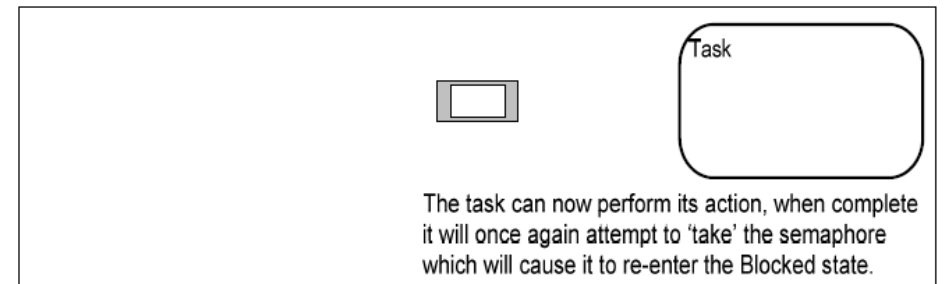
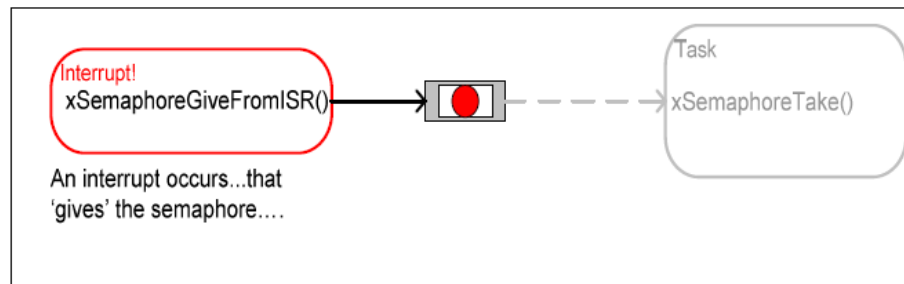
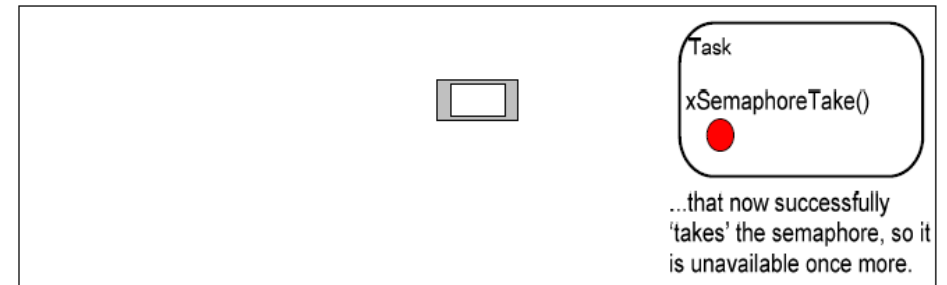
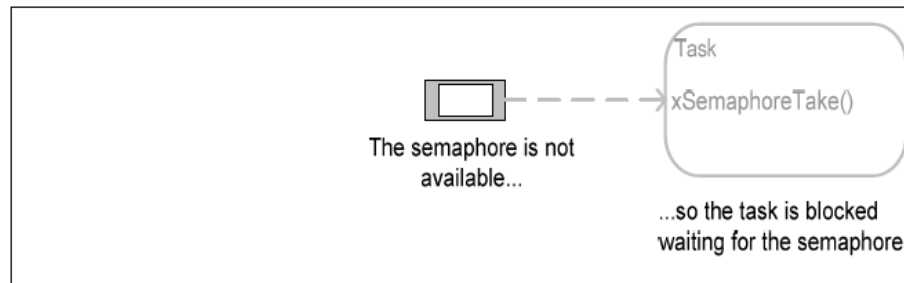
Semáforos en interrupciones.

```
void ISR(void)
{
    portBASE_TYPE    ccontexto;
    xSemaphoreGiveFromISR(sem,&ccontexto);
    portEND_SWITCHING_ISR(ccontexto);
    devolvercontrol();
}
```

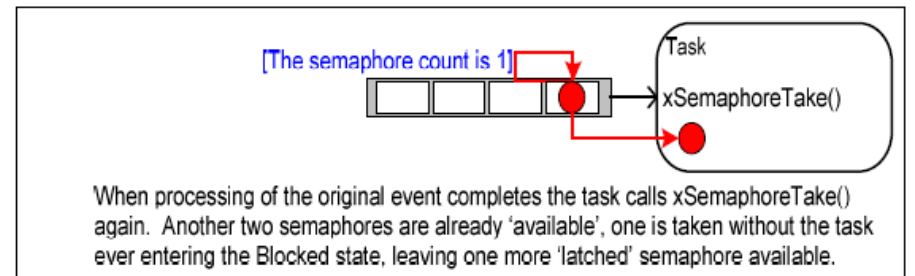
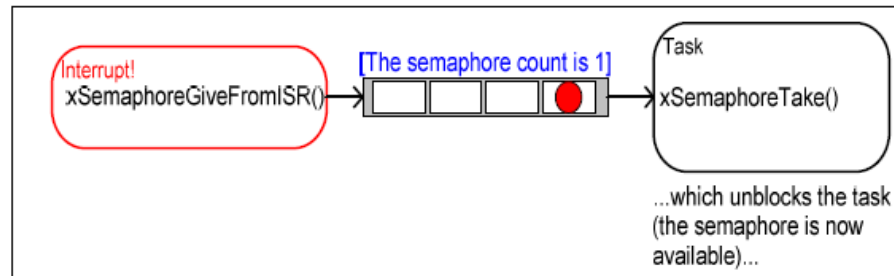
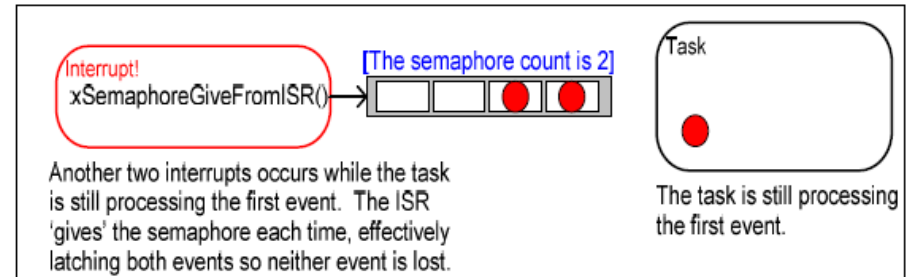
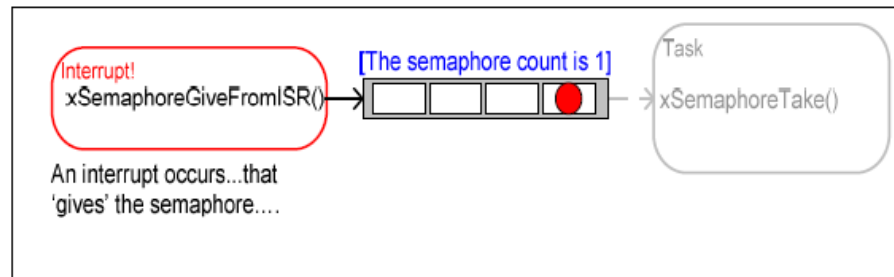
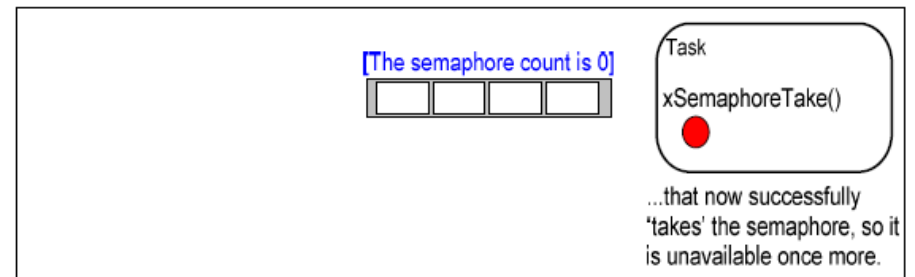
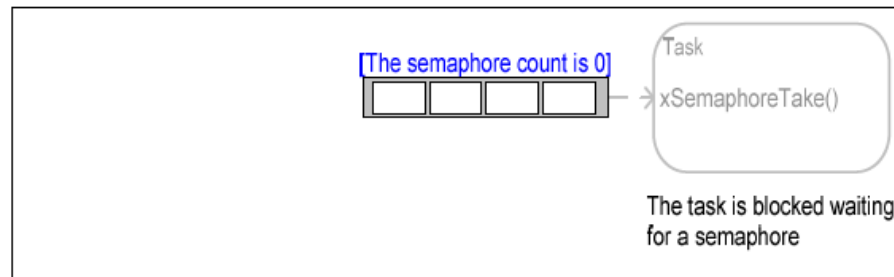
```
void Task1(void *parametros)
{
    for(;;)
    {
        codigoaejecutar();
        mascodigo();
    }
}
```

```
void HandlerTask(void *parametros)
{
    xSemaphoreTake(sem,portMAX_DELAY);
    for(;;)
    {
        xSemaphoreTake(sem,portMAX_DELAY);
        hacerloquehayquehacer();
    }
}
```

Semáforos en interrupciones.



Semáforos de conteo en int.





Colas de mensajes en interrupciones.

- Los semáforos son usados para comunicar eventos entre tareas y entre tareas e interrupciones.
- Las colas de mensajes son usadas para comunicar eventos **y transferir datos** entre tareas y entre tareas e interrupciones

Colas de mensajes. Funciones.

- ***portBASE_TYPE*** xQueueReceiveFromISR
(
 xQueueHandle pxQueue,
 void *pvBuffer,
 portBASE_TYPE *pxTaskWoken
)

Colas de mensajes. Funciones.

- ***portBASE_TYPE*** xQueueSendFromISR
(
 xQueueHandle pxQueue,
 const void *pvItemToQueue,
 portBASE_TYPE *pxHigherPriorityTaskWoken
)
- ***portBASE_TYPE*** xQueueSendToBackFromISR
(
 xQueueHandle pxQueue,
 const void *pvItemToQueue,
 portBASE_TYPE *pxHigherPriorityTaskWoken
)
- ***portBASE_TYPE*** xQueueSendToFrontFromISR
(
 xQueueHandle pxQueue,
 const void *pvItemToQueue,
 portBASE_TYPE *pxHigherPriorityTaskWoken
)



Uso eficiente de las colas.

- Las colas de FreeRTOS son colas que trabajan por ***copia*** por lo que se puede caer en uso poco eficiente de las colas si:
 - Se transfieren elementos de muchos bytes.
 - Se transfieren elementos a alta frecuencia.



Uso eficiente de las colas.

- Para tener un uso eficiente en condiciones de muchos datos o mucha frecuencia de datos:
 - Poner en un buffer común los elementos y utilizar semáforos para sincronizar las tareas.
 - Interpretar los datos en la ISR y encolar la cantidad mínima (teniendo en cuenta que el tiempo en las ISR debe ser muy poco).

Anidamiento de interrupciones.

- Recordando.... Los procesadores LPC17xx tienen 5 bits para configurar la prioridad de las interrupciones.
- La CMSIS tiene funciones para configurar la prioridad de las interrupciones
 - void NVIC_SetPriority (**IRQn_t** IRQn, **uint32_t** priority)
- Configuración del archivo FreeRTOSConfig.h

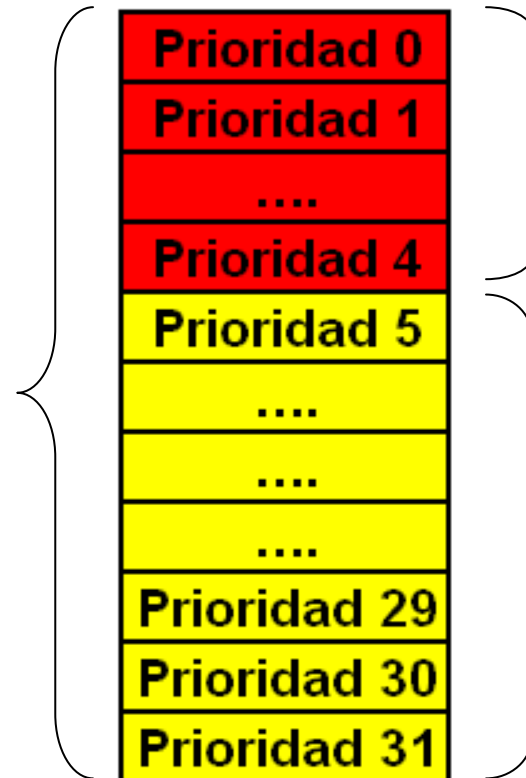
FreeRTOSConfig.h

- Las constantes que pueden afectar el anidamiento de interrupciones:
 - ***configKERNEL_INTERRUPT_PRIORITY***. Define la prioridad del núcleo del sistema operativo en si mismo.
 - ***configMAX_SYSCALL_INTERRUPT_PRIORITY***. Define la prioridad máxima que puede tener una interrupción para utilizar las funciones terminadas en FromISR.
- Si ***configMAX_SYSCALL_INTERRUPT_PRIORITY*** tiene más prioridad que ***configKERNEL_INTERRUPT_PRIORITY*** se va a trabajar con un esquema de anidamiento de interrupciones.

Configuración por defecto.

- `configMAX_SYSCALL_INTERRUPT_PRIORITY` 5
- `configKERNEL_INTERRUPT_PRIORITY` 31

Las interrupciones que no necesitan usar ninguna función del sistema operativo pueden tener cualquier prioridad

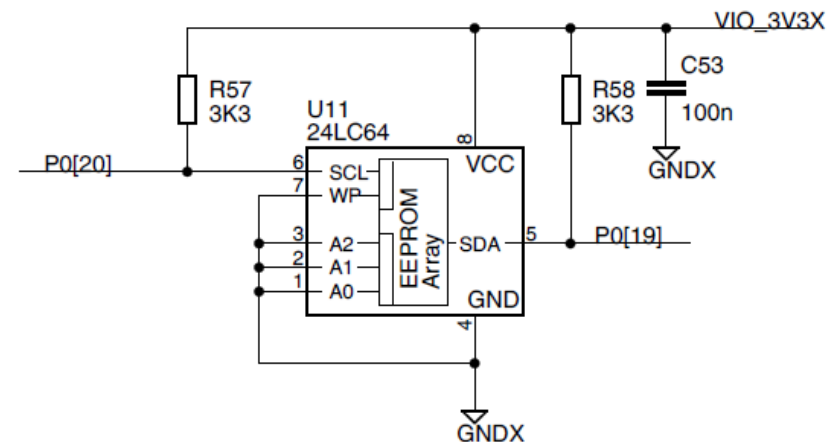


Las interrupciones que tienen prioridad desde 0 (***por defecto para todos los periféricos***) a 4 ***No pueden usar ninguna función de FreeRTOS!!!!!!***.

Las interrupciones que tienen prioridad desde 5 hasta 31 ***pueden llamar a las funciones de FreeRTOS que terminan en FromISR***

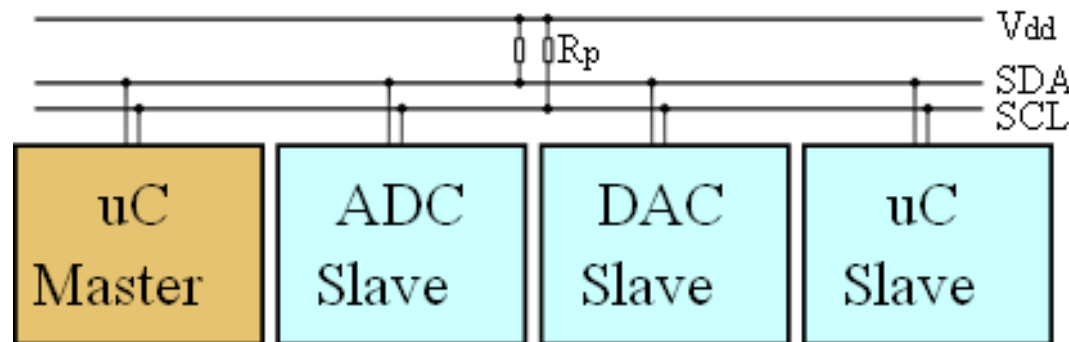
Ejemplo. Driver I²C.

- La intención de este ejemplo es generar un driver simple (pero completamente funcional) de I²C para utilizar la memoria EEPROM (24LC64) disponible en los sticks de LPC1769



I²C. ¿Qué es?

- I²C es un bus diseñado por Philips en los 80s.
- Es un bus que se lo utiliza para comunicar varios dispositivos compartiendo el mismo bus, usualmente dentro de la misma placa de circuito impreso.
- La velocidad de este bus originalmente era de 100KHz, hay un modo de 400KHz y existe un modo de hasta 3.4 Mbit/s.
- Es un protocolo de comunicación multimaestro, multiesclavo
- Es un bus que soporta muchos dispositivos esclavos y varios maestros en el mismo bus.

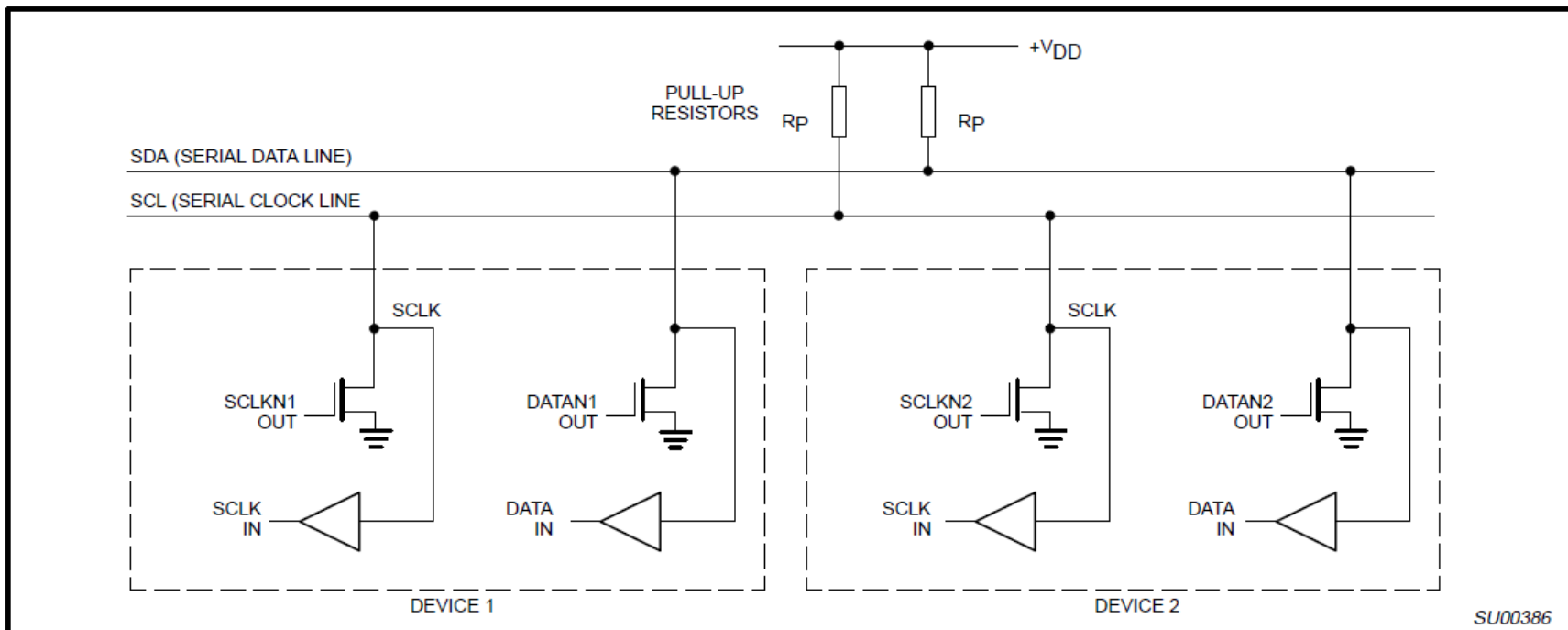




I²C. Breve Descripción eléctrica

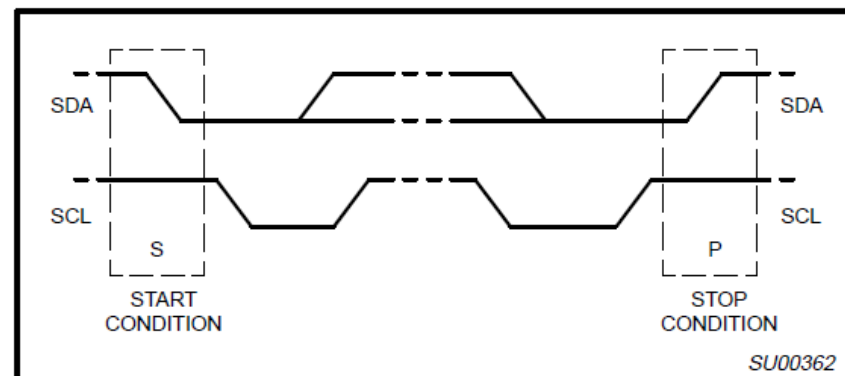
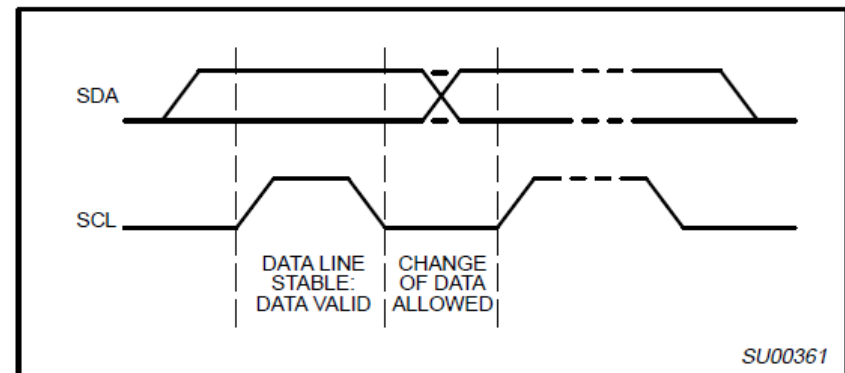
- El bus consta de dos pines: SCL y SDA.
- Estos pines son bidireccionales y trabajan en configuración de drain abierto, por lo que requieren necesariamente de resistencias de pull-up.

I²C. Breve Descripción eléctrica

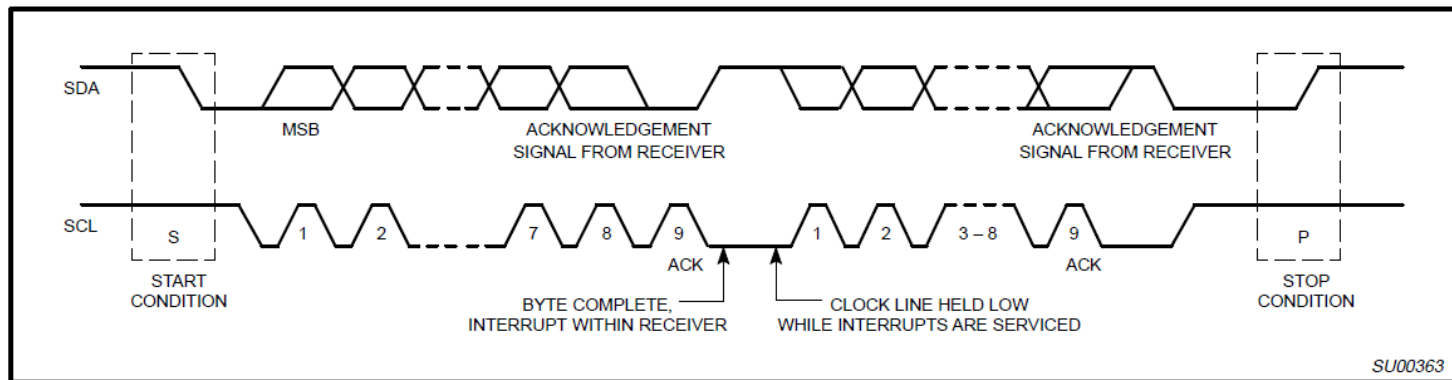


I²C. Señales.

- La Señal SDA es la señal de datos que va a ser válida cuando SCL está alta.
- Todas las transferencias de datos comienzan con una condición de arranque (bit de START) y finalizan con la condición de parada (bit de STOP).

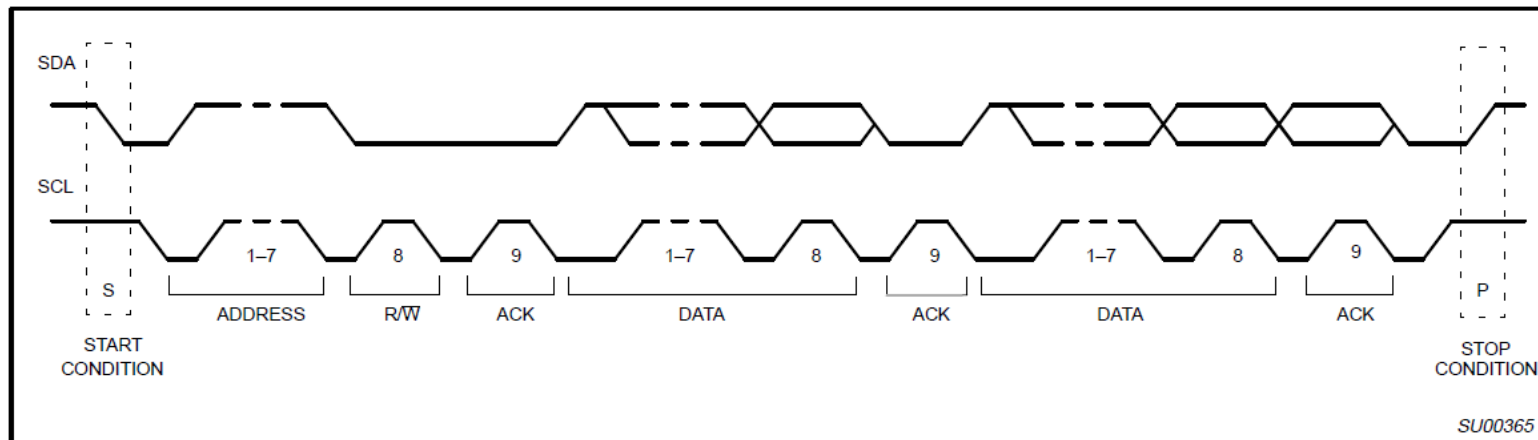


I²C. Señales



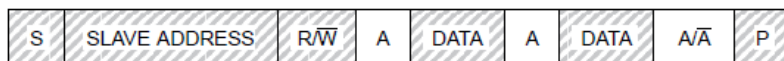
- La unidad de transferencia del protocolo I²C es el byte.
- Los bytes se transfieren enviando primero el bit más significativo MSB.

I²C. Transferencias de datos



- Las transferencias del I²C comienzan enviando una dirección de dispositivo (propia de cada tipo de dispositivo I²C), un bit de R/W y luego los datos. Luego de cada dato el dispositivo que los recibe genera un pulso de recibido (ACK) o de no recibido (NACK).

I²C. Operaciones entre maestros y esclavos.



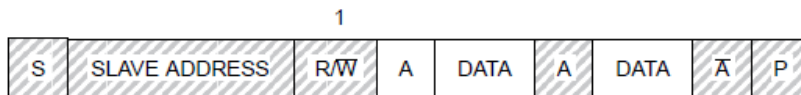
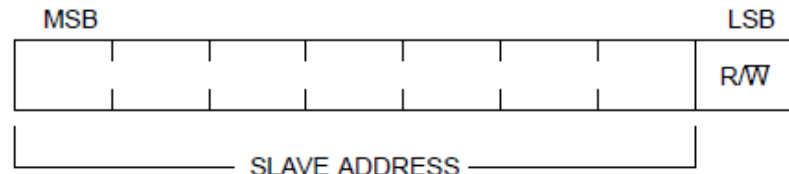
'0' (WRITE)

DATA TRANSFERRED
(n BYTES + ACKNOWLEDGE)

FROM MASTER TO SLAVE

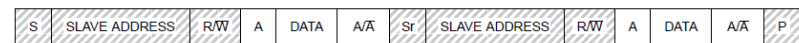
FROM SLAVE TO MASTER

A = ACKNOWLEDGE (SDA LOW)
 \bar{A} = NOT ACKNOWLEDGE (SDA HIGH)
 S = START CONDITION
 P = STOP CONDITION



(READ)

DATA TRANSFERRED
(n BYTES + ACKNOWLEDGE)



READ OR WRITE

(n BYTES + ACK.)*

READ OR WRITE

(n BYTES + ACK.)*

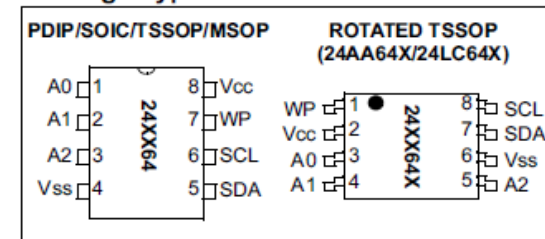
Sr = REPEATED START CONDITION

DIRECTION OF TRANSFER
MAY CHANGE AT THIS POINT

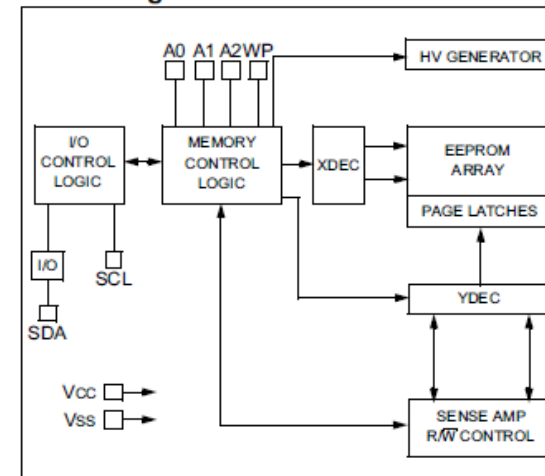
Memoria EEPROM 24LC64.

- Es una memoria de 8KBytes con interfaz I²C. Puede funcionar con un SCL máximo de 400KHz entre 2.5V y 5.5V.
- Tiene tres bits de direccionamiento además de la palabra de configuración de I²C
- Se pueden escribir hasta 32 bytes en una sola operación.

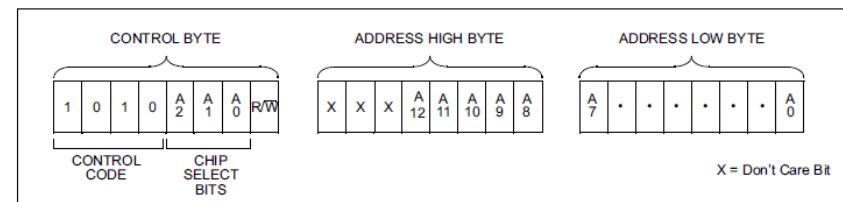
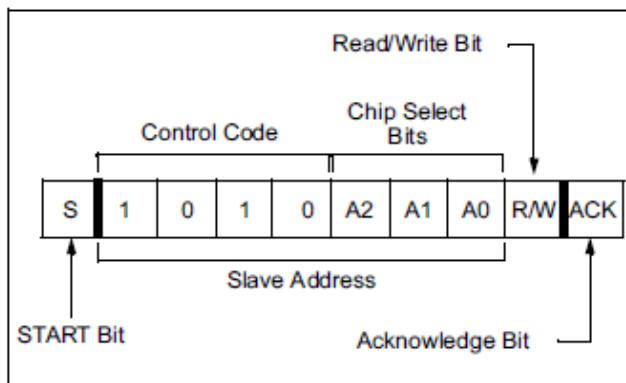
Package Types



Block Diagram



24LC64. Bytes de control.



24LC64. Escritura

FIGURE 4-1: BYTE WRITE

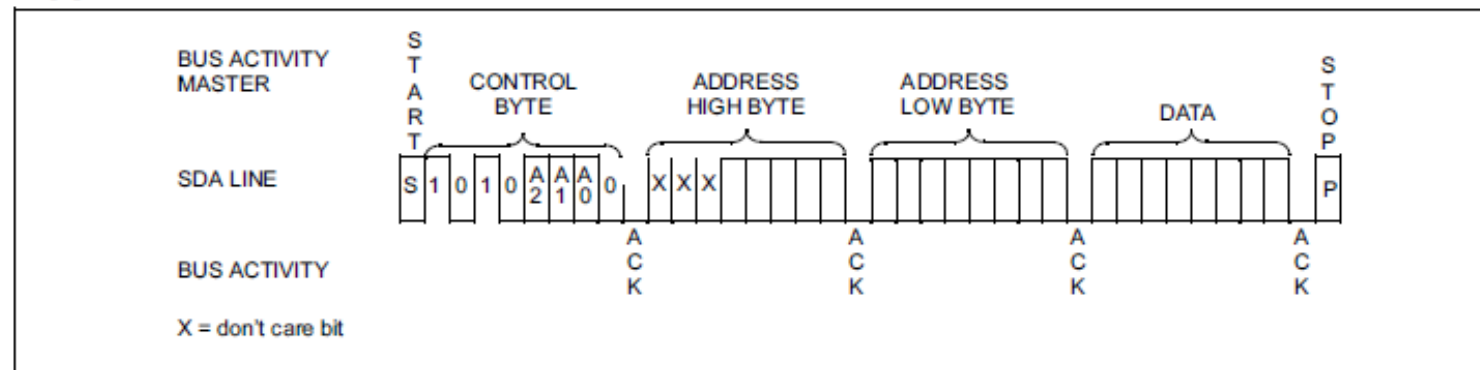
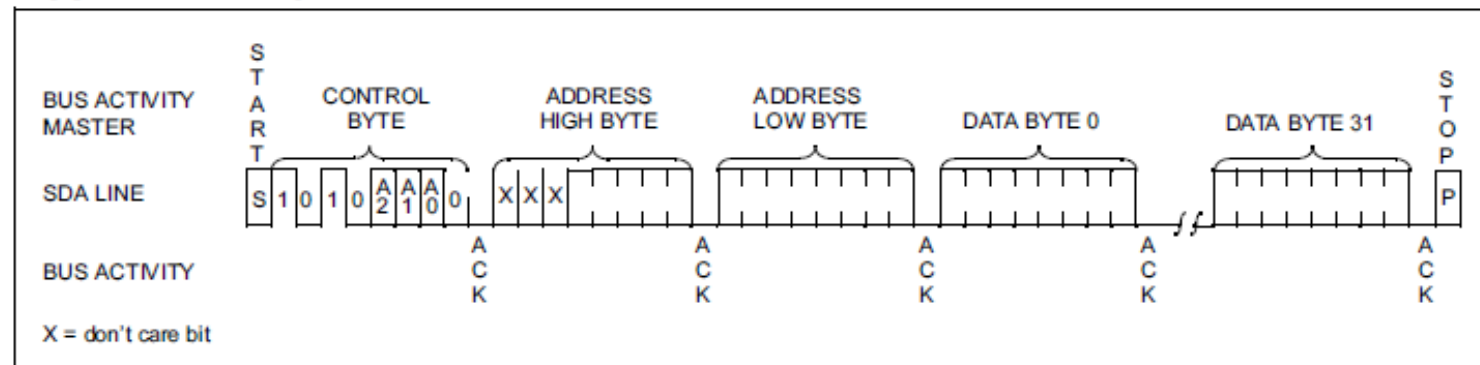


FIGURE 4-2: PAGE WRITE



24LC64. Lectura.

FIGURE 6-2: RANDOM READ

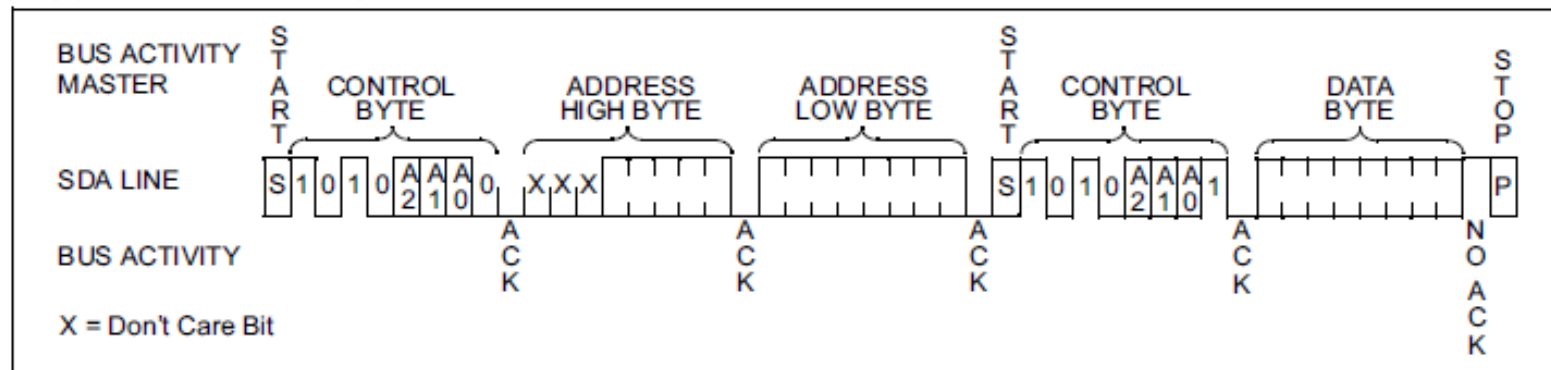
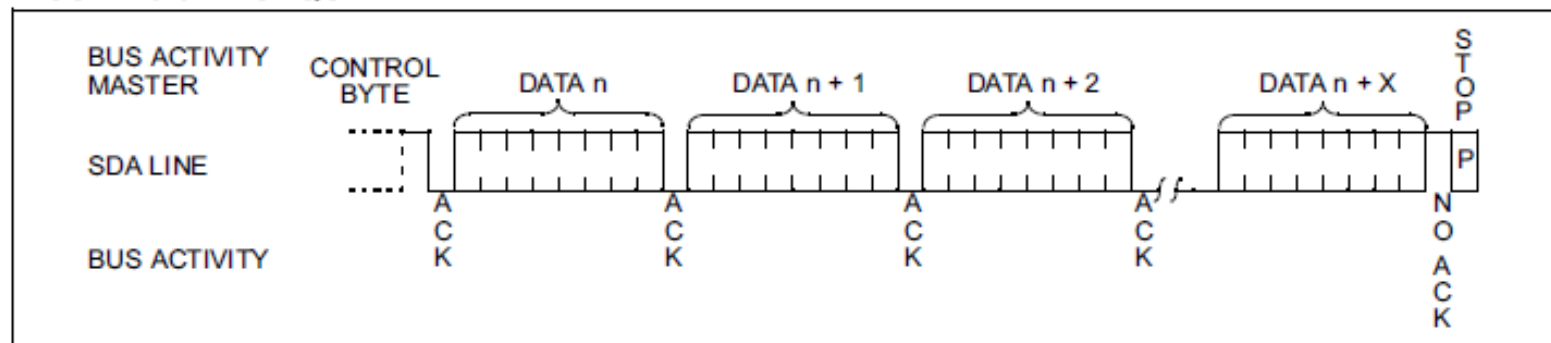


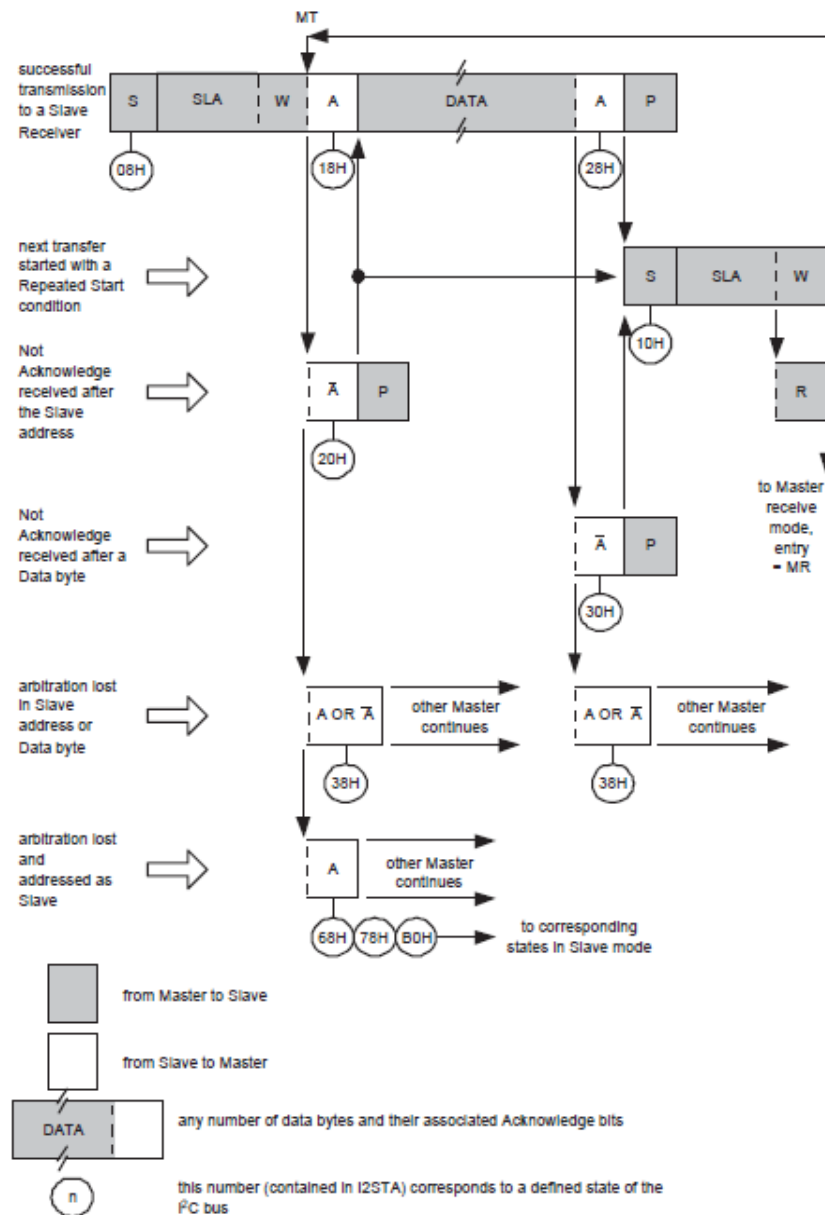
FIGURE 6-3: SEQUENTIAL READ

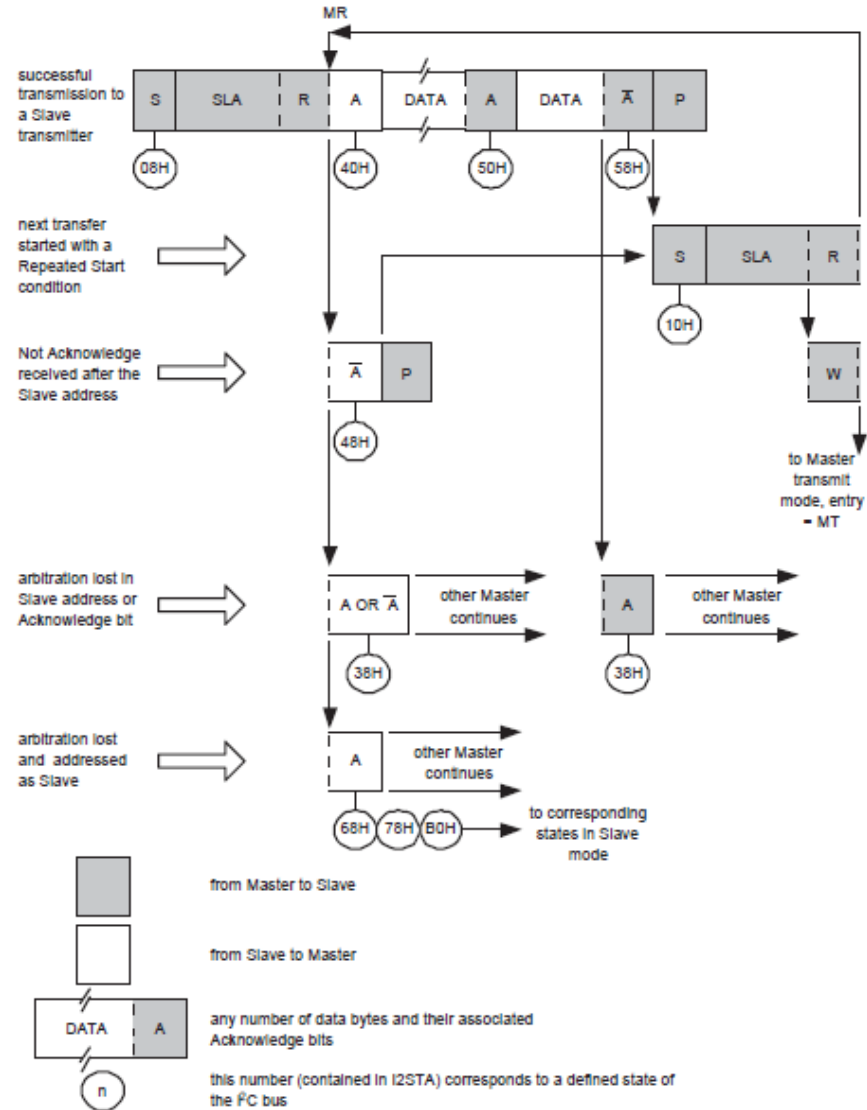




Controlador I²C. LPC1769

- El LPC1769 tiene 3 controladores I²C disponibles.
- La placa LPCXPRESSO que tiene el LPC1769 tiene una memoria 24LC64 conectado al controlador I²C1.
- El controlador se puede configurar tanto como maestro o como esclavo.








Consideraciones para el driver I²C

- El driver que se propone para manejar el I²C desde FreeRTOS contiene los siguientes supuestos:
 - No va a hacer controles de errores o timeout (para simplificar el código).
 - Los datos se van a transferir por copia.
 - La prioridad de la interrupción del driver es fija y no va a cambiar a lo largo del programa.
 - Las funciones de lectura y escritura son bloqueantes.
 - Está basado en el driver de CodeRed.
 - La sincronización se va a realizar utilizando un semáforo para la lectura y otro para la escritura.



Driver I²C. Código.

- Ver programa de FreeRTOS:
 - main.c
 - i2cdriverRTOS.c
 - i2c.h
 - FreeRTOSConfig.h

Bibliografía.

- Using the FreeRTOS Real Time Kernel. NXP LPC17xx Edition. Richard Barry.
- FreeRTOS <http://www.freertos.org/>
- LPC1769 LPCXPRESSO BOARD SPECIFICATION
http://http://www.embeddedartists.com/products/lpcxpresso/lpc1769_xpr.php
- Hoja de datos de la memoria EEPROM 24LC64
<http://ww1.microchip.com/downloads/en/devicedoc/21189f.pdf>
- Manual del procesador LPC17xx
http://www.nxp.com/download/pip/LPC1769FBD100/user_manual/
- Especificación de I²C. http://www.nxp.com/documents/user_manual/UM10204.pdf