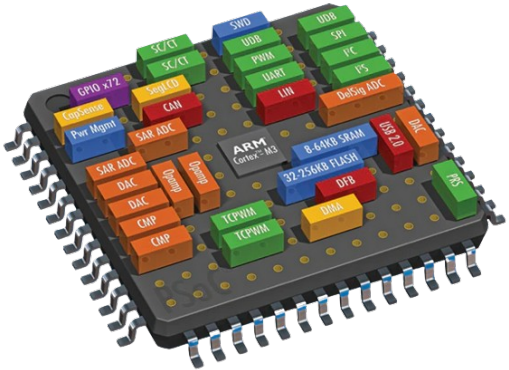
Aplicaciones de Sistemas Embebidos con Doble Núcleo

Fundamentos de Programación para Sistemas Embebidos

UTN FRA
Departamento de Ingeniería Electrónica
Laboratorio de Sistemas Embebidos

14 de agosto de 2024

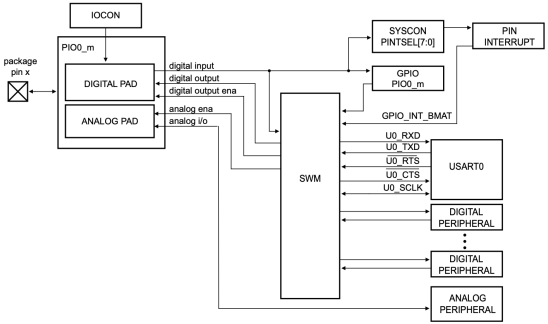
Índice



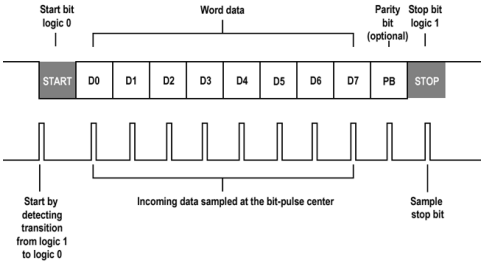
Periféricos comunes

- General Purpose Input Output (GPIO)
- Analog to Digital Converter (ADC)
- Digital to Analog Converter (DAC)
- System Timer (Systick) y otros Timers
- Universal Synchronous Asynchronous Receiver Transmitter (USART)
- Inter-Integrated Circuit (I2C)
- Serial Peripherial Interface (SPI)
- Universal Serial Bus (USB)

Cada pin tiene múltiples funciones asociadas. Se selecciona solo una de ellas de acuerdo a lo que se defina en la SWM y el IOCON.



- Comunicación full-duplex.
- Puede trabajar en modo sincrónico o asincrónico.
- Paquetes de datos de 7, 8 o 9 bits.
- Baudrates estándares incluyen 4800, 9600, 19200, 38400, 57600, 115200.
- Los pines de RX y TX deben cruzarse entre dispositivos.



- Se usa SWM_SetMovablePinSelect para elegir los pines que van a cumplir la función de USART.
- Se pasa primero el puntero a la matriz de conmutación (SWM), luego funcion que cumple (swm_select_movable_t) y luego el puerto y pin (swm_port_pin_type_t).
- Se puede seleccionar estas funciones para prácticamente cualquier pin.

```
// Habilito clock de matriz de conmutacion
CLOCK_EnableClock(kCLOCK_Swm);
// Asigno funciones de TX y RX a P0 25 y P0 24
SWM_SetMovablePinSelect(SWM0,
    kSWM_USART1_TXD,
    kSWM_PortPin_P0_25
);
SWM_SetMovablePinSelect(SWM0,
    kSWM_USART1_RXD,
    kSWM_PortPin_P0_24
);
// Quito el clock de la matriz de conmutacion
CLOCK_DisableClock(kCLOCK_Swm);
```

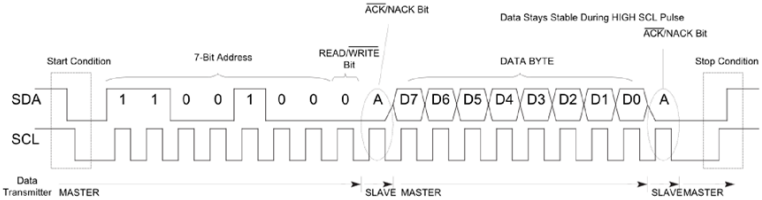
```
CLOCK_Select(kUART1_Clk_From_MainClk);
// Configuracion por defecto:
usart_config_t config = {
    baudRate_Bps = 115200,
    parityMode = kUSART_ParityDisabled,
    stopBitCount = kUSART_OneStopBit,
    bitCountPerChar = kUSART_8BitsPerChar,
    loopback = false,
    enableTx = true,
    enableRx = true,
    syncMode = kUSART_SyncModeDisabled
};

// Habilito USART1
USART_Init(USART1, &usart_config, CLOCK_GetFreq(kCLOCK_MainClk));
```

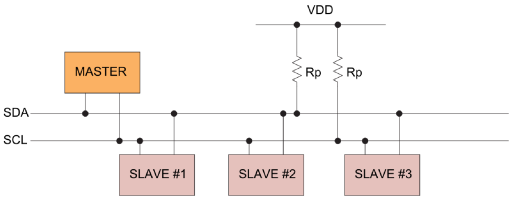
- Con CLOCK_Select se elige la fuente de clock para el periférico.
- En la estructura del tipo usart_config_t se configura: baudios, paridad, bits de stop, tamaño del paquete y la habilitación del RX y TX.

- Se usa USART_WriteBlocking para mandar un array o puntero a uint8_t.
- El puntero debe ser casteado correctamente si el tipo de dato original era otro.
- Se debe indicar la cantidad de bytes que se quieren enviar por USART.
- Se usa USART_ReadBlocking para leer una cantidad determinada de bytes del USART.
- Es necesario indicar un puntero a uint8_t o array de ese tipo para determinar el destino de lo leído.
- Como el nombre lo indica, ambas funciones son bloqueantes.

```
// Variable para mandar
char str[] = "Hola mundo!\n";

// Escribo por USART un byte
USART_WriteBlocking(USART1,
    (uint8_t*) str,
    sizeof(str) / sizeof(char)
);

// Leo byte por USART
USART_ReadBlocking(USART1,
    (uint8_t*) str,
    10
);
```

- Permite la conexión de múltiples dispositivos esclavos con solo un bus de datos (SDA) y clock (SCL).
- Cada dispositivo tiene una dirección única en el bus de 7 bits.
- Soporta velocidades estándares de 100Kb/s (Standard-mode), 400Kb/s (Fast-mode), 1Mb/s (Fast-mode Plus) y 3.4Mb/s (High-speed mode).





- Comunicación half-duplex.
- Para funcionar, el protocolo requiere resistencias de pull-up en SDA y SCL.
- Los pines asociados a I2C suelen ser salidas open-collector u open-drain.
- Las pull-ups ayudan que el estado por defecto del I2C sea high.

- Se usa SWM_SetMovablePinSelect para elegir los pines que van a cumplir la función de I2C.
- En algunos microcontroladores, existen buses de I2C especialmente rápido que solo pueden usarse en pines fijos usando la función SWM_SetFixedPinSelect.
- Exceptuando el caso anterior, cualquier pin suele poder usarse como SDA y SCL para el I2C.

```
// Habilito clock de matriz de conmutacion
CLOCK_EnableClock(kCLOCK_Swm);
// Asigno funciones de SDA y SCL a P0 27 y P0 26
SWM_SetMovablePinSelect(SWM0,
    kSWM_I2C1_SDA,
    kSWM_PortPin_P0_27
);
SWM_SetMovablePinSelect(SWM0,
    kSWM_I2C1_SCL,
    kSWM_PortPin_P0_26
);
// Quito el clock de la matriz de conmutacion
CLOCK_DisableClock(kCLOCK_Swm);
```

```
// Inicializo el clock del I2C1
CLOCK_Select(kI2C1_Clk_From_MainClk);

// Configuracion de master con 400 KHz de clock
i2c_master_config_t config = {
    true,    // Habilito Master
    400000, // Configuro a 400 KHz
    false    // Sin timeout
};
// Clock del sistema de base para generar el clock
I2C_MasterInit(I2C1, &config, SystemCoreClock);
```

- Con CLOCK_Select se elige la fuente de clock para el periférico.
- Con la estructura del tipo i2c_master_config_t se puede configurar si el dispositivo va a funcionar como Master y la velocidad de clock para la comunicación siempre que sea posible generarla en ese pin.

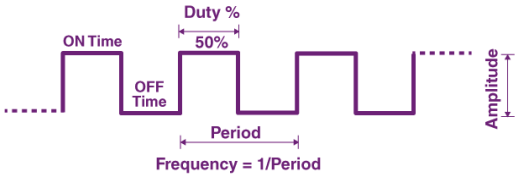
- La función I2C_MasterStart produce la condición de start en el bus.
- Para iniciar se pasa el bus usado, la dirección del esclavo y una constante del tipo i2c_direction_t para indicar si se escribe o se lee (kI2C_Write).
- Para escribir, se usa I2C_MasterWriteBlocking, pasando un array o puntero con los datos en formato uint8_t y la cantidad de bytes a enviar.
- Se termina la comunicación con I2C_MasterStop.

```
// Inicia escritura al slave con direccion 0x48
I2C_MasterStart(I2C1, 0x48, kI2C_Write);

// Registro y valor a escribir
uint8_t buf[] = { 0x66, 0xfa };

// Escribe ambos bytes
I2C_MasterWriteBlocking(I2C1,
    buf,     // Datos para escribir
    2,       // Cuantos bytes
    kI2C_TransferDefaultFlag
);

// Detengo la comunicacion
I2C_MasterStop(I2C1);
```

```
// Inicia escritura al slave con direccion 0x48
I2C_MasterStart(I2C1, 0x48, kI2C_Write);

// Registro y valor a escribir
uint8_t buf[] = { 0x66, 0xfa };

// Escribe ambos bytes
I2C_MasterWriteBlocking(I2C1,
    buf,     // Datos para escribir
    2,       // Cuantos bytes
    kI2C_TransferDefaultFlag
);

// Detengo la comunicacion
I2C_MasterStop(I2C1);
```

- La función I2C_MasterStart inicia la comunicación.
- Se pasa a la función el valor de kI2C_Read para iniciar una lectura.
- Para leer, se usa I2C_MasterReadBlocking, pasando un array o puntero a uint8_t donde guardar los datos y la cantidad de bytes a leer.
- Se termina la comunicación con I2C_MasterStop.

- Generan señales contínuas pulsantes de frecuencia fija y ancho de pulso variable.
- El efecto final es que la carga ve una señal de tensión media entre 0 y el valor de V_{CC}.
- Los valores de ancho de pulso suelen expresarse entre 0 y 1.
- Típicamente usado para controlar velocidad de motores, intensidades lumínicas, salidas analógicas y generación de senoidales.



- Se usa SWM_SetMovablePinSelect para elegir el pin que va a generar una salida PWM.
- Los PWM se generan con Timers, en el caso del LPC845, el Timer por excelencia es el SCTimer.
- En el caso del LPC845, hay disponibles cuatro salidas para PWM.

```
// Habilito clock de matriz de conmutacion
CLOCK_EnableClock(kCLOCK_Swm);
// Asigna la salida 4 del SCT al P1_1
SWM_SetMovablePinSelect(SWM0,
    kSWM_SCT_OUT4,
    kSWM_PortPin_P1_1
);
// Quito el clock de la matriz de conmutacion
CLOCK_DisableClock(kCLOCK_Swm);
```

```
// Configuracion generica del SCT Timer
sctimer_config_t sctimer_config;
SCTIMER_GetDefaultConfig(&sctimer_config);
SCTIMER_Init(SCT0, &sctimer_config);

// Configuro el PWM
sctimer_pwm_signal_param_t pwm_config = {
    kSCTIMER_Out_4,      // Salida del Timer
    kSCTIMER_LowTrue,    // Logica negativa
    50                   // 50% de ancho de pulso
};
```

- Se puede obtener una configuración por defecto con SCTIMER_GetDefaultConfig.
- En la estructura del tipo sctimer_pwm_signal_param_t se pueden configurar cuál de las salidas del Timer se va a usar, la lógica y el porcentaje de ancho de pulso.

```
// Evento al que se asigna el PWM
uint32_t event;
// Eligo el clock para el Timer
uint32_t sctimer_clock = CLOCK_GetFreq(kCLOCK_Fro);

// Inicializo el PWM
SCTIMER_SetupPwm(
    SCT0,
    &pwm_config,      // Estructura anterior
    kSCTIMER_CenterAlignedPwm,
    1000,             // 1KHz de frecuencia
    sctimer_clock,    // Clock de base
    &event
);

// Inicializo el Timer
SCTIMER_StartTimer(SCT0, kSCTIMER_Counter_U);
```

- Con la función SCTIMER_SetupPWM se puede configurar lo establecido en la estructura sctimer_pwm_signal_param_t y otros parámetros como la frecuencia del PWM y frecuencia de base del periférico.
- La función SCTIMER_StartTimer es la encargada de dar inicio al Timer con la configuración establecida.

```
SCTIMER_UpdatePwmDutycycle(
    SCT0,
    kSCTIMER_Out_4,      // Salida usada
    75,                  // Ancho de pulso
    event
);

SCTIMER_UpdatePwmDutycycle(
    SCT0,
    kSCTIMER_Out_4,      // Salida usada
    0,                   // PWM apagado
    event
);

SCTIMER_UpdatePwmDutycycle(
    SCT0,
    kSCTIMER_Out_4, // Salida usada
    75,             // Continua valor maximo
    event
);
```

- La función SCTIMER_UpdatePwmDutycycle es la que actualiza el ancho de pulso del PWM, requiere la salida que se quiere modificar, el ancho de pulso y el evento registrado anteriormente.
- El valor de ancho de pulso se especifica en 0 a 100.
- Se especifica la salida del SCTimer, no el pin mapeado a la salida.

Algunas propuestas para practicar

1. En un proyecto llamado **03_light_control**, hacer un programa que lea la intensidad lumínica del BH1750 y regule el brillo del LED D1 según este valor. El máximo brillo del LED debe encontrarse al máximo de intensidad lumínica y viceversa.
2. En un proyecto llamado **03_pwm_sine_wave**, generar una señal de PWM apropiada para lograr, con un filtro pasa banda, una senoidal de 10KHz.
3. En un proyecto llamado **03_servo_control**, una señal de PWM apropiada para cubrir el ángulo de giro completo de un servomotor SG90 controlado por la indicación del potenciómetro R22.

Cada ejercicio que se resuelva, subirlo al repositorio personal del curso.

Algunos recursos útiles

- Manual del LPC845
- Manual del LPC845 Breakout Board
- Documentación del SDK del LPC845
- Esquemático del kit del laboratorio
- Half-Duplex vs Full-Duplex
- I2C-bus specification and user manual
- A Basic Guide to I2C
- Hoja de datos del BH1750
- Hoja de datos de servo motor SG90