Funciones de la práctica 3.2

Juego Mastermind con consola serie

ec

Ficheros que contiene la práctica 3.2

- Hay que usar los ficheros que habéis programado para la practica 2
  - **gpio.c**
  - **D8Led.c**
  - **intcontroller.c**
  - **timer.c**
  - **Keyboard.c**
  - **…**

- Se añaden los ficheros (**uart.c y uart.h)** que contienen las **funciones necesarias para configurar el dispositivo de transmisión en serie** (**UART): tiene dos puertos serie uart0 y uart1**
  - **Tenéis que programar estas funciones**

Programa principal de la práctica 3.2

- **main.c: partiendo del de la práctica 3.1 hay que añadir lo siguiente:**
  - **En la función loop ():** en el **estado DOGUESS** hay que añadir las siguientes modificaciones
    - **Transmitir (enviar) por el puerto serie uart0** al terminal del ordenador **la cadena** "Introduzca passwd: "
    - **Recibir (leer) caracteres de** la clave (escrita en el teclado del ordenador)**a través del puerto serie uart0**
    - **Copiar los 4 últimos caracteres de la línea leída en el buffer guess** para que el siguiente estado pueda visualizarlos
      - » Hay que **convertir estos caracteres de ASCII a valor decimal**
  - Función **int readline (char *buffer, int size)**
    - Lee los caracteres recibidos por Uart0 y los escribe en el parámetro "buffer"
      - » Lee caracteres hasta encontrar '\r' o hasta llenar el buffer que se le pasa (tamaño indicado por size), lo que antes suceda
  - Función **static char ascii2digit(char c)** **(Esta función se os da hecha)**
    - Para convertir un carácter de ASCII a valor decimal
  - **En la función setup ()**
    - **Configurar la uart**

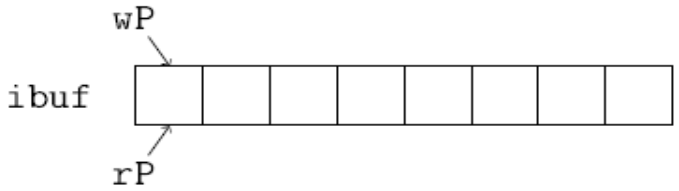
Estructuras definidas

```
struct ulconf {
        enum ONOFF ired;          // Infrarrojos
        enum UPARITY par;         // Paridad
        enum USTOPB stopb;        // Bit de stop
        enum UWORDLEN  wordlen;   //Tamaño de la trama
        enum ONOFF  echo;
        int baud;
};
```

Contiene los valores para configurar la trama y la velocidad de trabajo de la uart

```
struct port_stat {
enum  URxTxMode rxmode;  //Modo de recepción (DIS, POLL, INT, DMA)
enum URxTxMode txmode;   //Modo de envío (DIS, POLL, INT, DMA)
unsigned char ibuf[BUFLEN];
int rP;
int wP;

char *sendP;             //Puntero a la cadena de envío (modo INT)
enum ONOFF echo;         //Flag para echo de caracteres recibidos
};
```



Para gestionar la transmisión y recepción en modo Interrupción

Subrutinas uart.c: rutinas de configuración

- **void uart_init (void)**
  - Inicializa estructura port_stat
  - Inicializa la tabla de direcciones de las subrutinas de tratamiento de interrupción
  - Configure las líneas de interrupción de los puertos series uart1 y uart0 por IRQ

- **int uart_lconf (enum UART port, struct ulconf *lconf)**
  - **El parámetro lconf contiene los valores para configurar la trama y la velocidad de trabajo**
  - Configura la trama
    - Infrarrojos
    - Paridad
    - Bit de stop
    - Tamaño de la trama
  - Asigna la velocidad de trabajo de la uart
  - Configura los pines de los puertos E y C de la GPIO que es por donde la uart0 y la uart1 se comunican con el exterior (el ordenador)

Subrutinas uart.c: rutinas de configuración

- **int uart_conf_txmode (enum UART port, enum URxTxMode mode)**
  - Configura en qué modo se realiza la transmisión
    - Polling
    - Interrupciones
    - DMA
  - Configura el tipo de interrupción por nivel

- **int uart_conf_rxmode (enum UART port, enum URxTxMode mode)**
  - Configura en qué modo se realiza la recepción
    - Polling
    - Interrupciones
    - DMA
  - Configura el tipo de interrupción por pulso

Subrutinas uart.c: RTIs

- Cuando la transmisión/recepción es configurada en modo interrupciones son necesarias las siguientes subrutinas de interrupción:

- **void Uart0_RxInt (void)**
  - RTI de recepción para Uart0

- **void Uart1_RxInt (void)**
  - RTI de recepción para Uart1

- **void Uart0_TxInt (void)**
  - RTI de transmisión para Uart0

- **void Uart1_TxInt (void)**
  - RTI de transmisión para Uart0

Subrutinas uart.c: Rutinas de recepción

- **Recepción de un carácter por el puerto serie UARTn**
  - Se almacena en el **registro buffer de recepción** URXHn

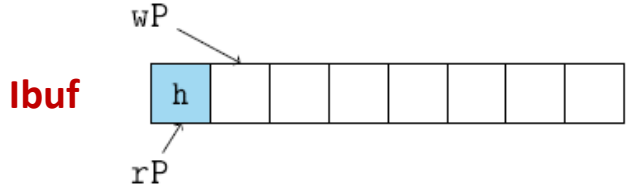
  - Si la recepción está configurada en modo interrupción
    - La línea de interrupción de transferencia UTXDn se activa cada vez que haya un carácter escrito el registro buffer de recepción

Cuando se activa la línea URXDn

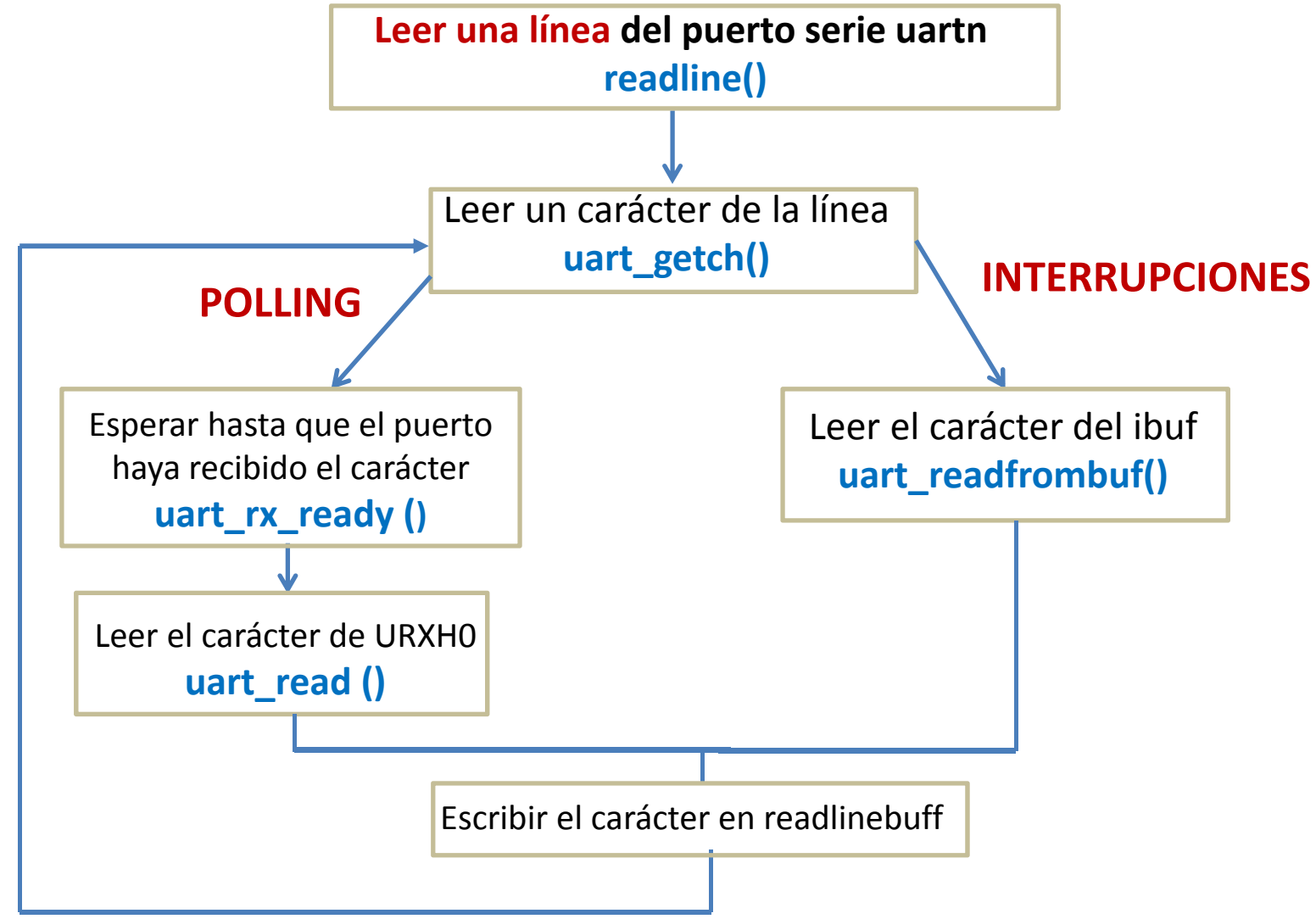
RTI: Uartn_RxInt ()

Leer el carácter de URXHn y copiarlo en ibuf

uart_readtobuf ()



Subrutinas uart.c: Rutinas de recepción



Subrutinas uart.c: Rutinas de recepción

- **static void uart_rx_ready (enum UART port)**
  - Espera a que se haya recibido un carácter por el puerto serie UARTn (n=port)
- **static char uart_read (enum UART port)**
  - Lee un carácter del registro buffer de recepción URXHn de la UARTn (n=port)
  - Se puede usar macro RdURXHn
- **static void uart_readtobuf (enum UART port)**
  - Copia un carácter del registro buffer URXHn al buffer ibuf del puerto “port”
  - Se llama cada vez que hay una interrupción por recepción de la UARTn
- **static char uart_readfrombuf (enum UART port)**
  - Lee un carácter del buffer ibuf del puerto “port”
- **int uart_getch (enum UART port, char *c)**
  - Lee un carácter del registro buffer de URXH, o del buffer ibuf, del puerto “port” (depende del modo en que se realiza la recepción Polling o Interrupción) y lo devuelve por el parámetro “c”

Subrutinas uart.c: Rutinas de transmisión

Transmitir por el puerto serie uart0 al terminal del ordenador **una cadena**
uart_printf()

Envía una cadena por el puerto UARTn
uart_send_str ()

POLLING

INTERRUPCIONES

Envía un carácter por el puerto UARTn
uart_sendch ()

Esperar hasta que el **registro buffer de transmisión** UTXHn esté vacío
uart_tx_ready ()

Escribe el carácter en el **registro buffer de transmisión** UTXHn
uart_write ()

Hasta que se ha enviado la cadena entera

- **Habilita las interrupciones de transmisión**
- Espera hasta que se haya enviado todo la cadena

La línea de interrupción de transferencia UTXDn se activa cada vez que se vacíe el registro buffer de transmisión

Subrutinas uart.c: Rutinas de transmisión

- **Se produce una interrupción por transferencia** cada vez que se vacíe el registro buffer de transmisión

Cuando se **activa la línea UTXDn** salta la **RTI:**

Uartn_TxInt ()

Envía el carácter al que señala el puntero sendP
uart_dotxint ()

Esperar hasta que el **registro buffer de transmisión** UTXHn esté vacío
uart_tx_ready ()

Escribe el carácter en el **registro buffer de transmisión** UTXHn
uart_write ()

Si envía el último carácter de la cadena
desactiva las interrupciones

Subrutinas uart.c: Rutinas de transmisión

- **static void uart_tx_ready (enum UART port)**
  - Espera a que el registro buffer de transmisión UTXHn de la UARTn (n=port) esté vacío
- **static char uart_write (enum UART port, char c)**
  - Escribe el carácter que se pasa como argumento (c) en el registro buffer de transmisión UTXHn de la UARTn (n=port)
  - Se puede usar la macro WrURXHn
- **int uart_sendch (enum UART port, char c)**
  - Envía el carácter que se pasa como argumento (c) por el puerto serie (port)
  - Depende del modo en que se realiza la transmisión Polling o Interrupción
    - Si es interrupción envía el caracter llamando a uart_send_str ()
- **int uart_send_str (enum UART port, char *str)**
  - Envía la cadena de caracteres apuntada por str por el puerto serie (port)
  - Depende del modo en que se realiza la transmisión Polling o Interrupción

Subrutinas uart.c: Rutinas de transmisión

- **void uart_printf (enum UART port, char *fmt, …)**
  - Envía por el puerto serie una cadena de caracteres con formato fmt, al estilo de printf

- **static void uart_dotxint (enum UART port)**
  - Envía un carácter al que señala el puntero send en UTXH0
  - Incrementa el puntero
  - Si el puntero queda apuntando al final de la cadena deshabilita la línea de interrupción por envío de la uart
  - Pone el puntero sendP a NULL para indicar el final del envío