Práctica 6

Programación del uC del periférico de comunicación serie utilizando interrupciones.

Objetivo: Mediante esta práctica el alumno aprenderá el uso básico para inicializar y operar, bajo un esquema de interrupciones, el puerto serie del microcontrolador.

Equipo: - Computadora Personal
- Tarjeta de desarrollo ESP32

Teoría: - Manejo del Periférico de Comunicación Serie (UART) del microcontrolador ESP32
- Secuencias de escape ANSI.

Actividades a realizar:

1. Implementar las siguientes funciones:

```
void uartInit(uint8_t com, uint16_t baudrate, uint8_t size,
              uint8_t parity, uint8_t stop)
```

Función que inicializa el periférico del UART en un esquema de interrupciones. Y la configuración es dada por los parámetros, donde:

- **com**: representa el número de UART a configurar. Considerar los cuatro posibles puertos.
- **baudrate**: representa la velocidad en Baud de configuración, puede ser no estándar.
- **size**: representa el número de bits de los datos con los que operará el UARTx. Considerar de 5 a 8 bits.
- **parity**: representa el tipo de paridad con los que operará el UARTx. Considerar 0: No paridad, 1: impar, 2: par.
- **stop**: representa el número de bits de paro con los que operará el UARTx. Considerar 1 ó 2.

```
void uartGets(uint8_t com, char *str)
```

Función que retorna una cadena haciendo uso de *uartgetchar(`uint8_t com`)*, la cadena se retorna en el apuntador *str*.

```
void uartPuts(char *str) → void uartputs(uint8_t com, char *str)
```

Función que imprime una cadena mediante *uartputchar(uint8_t com, char data)*.

```
void myItoa(uint16_t number, char* str, uint8_t base)
```

Función que imprime un numero de 16 bits y lo convierte en la base dada sobre el aputnador de cadena.

```
uint16_t myAtoi(char *str)
```

Función que retorna un numero de 16 bits en base a la cadena dada en el apuntador. Asumir que el numero es decimal.

```
void uartSetColor(uint8_t com, uint8_t color)
```

Función que envía la secuencia de escape para configurar el color del texto que se desplegará en la terminal.

```
void uartGotoxy(uint8_t com, uint8_t x, uint8_t y)
```

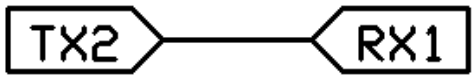
Función que posiciona el cursor en la terminal en la coordenada *x*,*y* que lleguen como parámetro, utilizando la secuencia de escape.

Utilizar el siguiente listado para comprobar el funcionamiento de las funciones.

Listado 1.

```
int main( void )
{
    char cad[20];
    char cadUart[20];
    uint16_t num;
    uartInit(PC_UART_PORT,12345,8,1,2,PC_UART_RX_PIN,PC_UART_TX_PIN);
    uartInit(UART1_PORT,115200,8,0,1,UART1_TX_PIN,UART1_RX_PIN);
    uartInit(UART2_PORT,115200,8,0,1,UART2_TX_PIN,UART2_RX_PIN);
    while(1)
    {
        uartGetchar(0);
        uartClrScr(0);
        uartGotoxy(0,2,2);
        uartSetColor(0,YELLOW);
        uartPuts(0,"Introduce un número:");
        uartGotoxy(0,22,2);
        uartSetColor(0,GREEN);
        uartGets(0,cad);
// For the following code to work, TX2 must be physically
// connected with a jumper wire to RX1
// ----------------------------------------
        // Cycle through UART2->UART1
        uartPuts(2,cad);
        uartPuts(2,"\r");
        uartGets(1,cadUart);
        uartGotoxy(0,5,3);
        uartPuts(0,cadUart);
// ----------------------------------------
        num = myAtoi(cad);
        myItoa(num,cad,16);
        uartGotoxy(0,5,4);
        uartSetColor(0,BLUE);
        uartPuts(0,"Hex: ");
        uartPuts(0,cad);
        myItoa(num,cad,2);
        uartGotoxy(0,5,5);
        uartPuts(0,"Bin: ");
        uartPuts(0,cad);
    }
}
```

Es necesario interconectar la transmisión del UART1 hacia la recepción del UART2:



Comentarios y Conclusiones.

Bibliografía.