86.65 - 66.48 - 66.66 - Sistemas Embebidos
Trabajo Práctico N 3: Debounce, Uart , A/D y D/A

Objetivo :

El objetivo de esta actividad es poder testear el correcto funcionamiento de hard y soft de los periféricos UART, A/D y D/A, avanzando sobre el mismo incrementando la complejidad en forma gradual y pensando en que se cuentan con acceso limitado a recursos electrónicos.

Nota importante: Se requerirá para realizar esta práctica solamente de **2 resistores de valor igual y superior a 1 kohm**.

Referencias :

- Ejemplo "switches leds" (gpio) de la sAPI
- Ejemplo "tick_hood" de la sAPI
- Ejemplo "fsm_debounce_non_blocking" (finite_state_machine) de la sAPI
- Ejemplo "adc_dac" de la sAPI
- Ejemplo "rgb_led_uart" (pwm) de la sAPI
- Ejemplo "echo" (uart) de la sAPI
- Ejemplo "rx_interrupt" (uart) de la sAPI

1. En forma idéntica a la solicitada en el TP2, documentar las funciones que se listan a continuación. Identificar la secuencia de funciones invocadas durante la ejecución de la misma, en qué archivo se encuentran, su descripción detallada, qué efecto tiene la aplicación sobre el hardware (identificar circuitos, puertos, pines, niveles, etc.).

   - uartConfig(UART_USB, 115200);
   - adcConfig(ADC_ENABLE);
   - dacConfig(DAC_ENABLE);
   - delayConfig(&delay, 500);
   - muestra = adcRead(CH1);
   - uartReadByte(UART_USB, &dato)
   - uartWriteByte(UART_USB, dato)
   - uartWriteString(UART_USB, "ADC CH1 value: ");
   - dacWrite(DAC, muestra);
   - uartCallbackSet(UART_USB, UART_RECEIVE, onRx, NULL);
   - uartInterrupt(UART_USB, true);

2. A partir de los programas de ejemplo switches_leds, fsm_debounce_non_blocking, tickHook, y uart, programar una aplicación que realice lo siguiente:

   PULSADOR -> LED -> UART (USB).TX -> TERMINAL (PC) -> TEXTO(PANTALLA)

   Es decir, cuando se presione (o se suelte) uno de los pulsadores se deberá prender (o apagar) un led de la placa y enviar un carácter a través de la terminal serie implementada con el puerto USB de debug. Dado que hay 4 pulsadores en la placa, asignar un led y un caracter distinto a cada pulsador y mantener la consistencia entre ellos.

3. Una vez hecho el punto anterior, se desea hacer una aplicación similar pero al revés:

   TECLA (PC) -> TERMINAL (PC) -> UART (USB).RX -> LED

   a. Por medio de la terminal serie implementada con el puerto USB de debug, y basándose en el ejemplo uart.c, asignar 3 caracteres para prender cada uno de los 3 led y 3 caracteres para apagar cada uno de los 3 led.
   b. Implementar el ejercicio a. de nuevo, pero por interrupciones. Basarse en el ejemplo rx_interrupt.
   c. Combinar las funcionalidades de los ejercicios a. y b. en un solo programa:

      TEXTO(PANTALLA) <- TERMINAL (PC) <- UART (USB).TX <- PULSADOR

      TECLA (PC) -> TERMINAL (PC) -> UART (USB).RX -> LED

4. A partir del programa de ejemplo rgb_led_uart:

   a. armar un PWM por software para disponer de distintos niveles de brillo en cada uno de los leds.
   b. Por medio de la terminal serie implementada con el puerto USB de debug, asignar un caracter para incrementar y otro para decrementar el brillo de 1 led.

5. A partir del programa de ejemplo adc_dac, vamos a probar el funcionamiento del conversor A/D. Con un resistor de valor igual o superior a 1k, y según lo que muestra la imagen adc_dac_pins.png del ejemplo:

    a. Conectar a través del resistor la entrada de A/D a 3.3v y verificar por la terminal de debug que indique el fondo de escala.
    b. Conectar a través del resistor la entrada de A/D a 0v y verificar por la terminal de debug que indique el valor nulo.
    c. Conectar 2 resistores de igual valor de la siguiente manera:

    3,3v -/\/\/\-- A/D -/\/\/\-- 0v

    y verificar por la terminal de debug que indique un valor cercano al centro de escala.

    (0v .. 3,3V) -> RESISTOR -> A/D -> UART (USB).TX -> TERMINAL (PC) -> TEXTO (PC)

6. A partir del programa de ejemplo adc_dac, vamos a probar el funcionamiento del conversor D/A y vamos a verificarlo con el ejercicio anterior:

    a. Modificar el ejemplo para que con las teclas numéricas (de 0 a 9) se pueda variar la tension de salida del DAC el valor de la salida DAC (donde el “0” corresponde con 0v y el “9” con 3,3v).

    TECLA (0 - 9) -> TERMINAL (PC) -> UART (USB).RX -> DAC -> (0 - 3,3v)

    b. Agregar al punto anterior la funcionalidad del ejercicio 5 para poder visualizar por la consola la variacion de tension que le aplicamos al A/D. Para ello se debe armar un loopback a través de puerto serie:

    bornera 15 [DAC] --/\/\/\-- bornera 11 [ADC (CH1)]

    con un resistor de valor igual o mayor a 1k.

    TECLA (0..9) -> TERMINAL (PC) -> UART (USB).RX -> DAC -> (0..3,3v) -> RESISTOR -> (0v .. 3,3V) -> RESISTOR -> A/D -> UART (USB).TX -> TERMINAL (PC) -> TEXTO (PANTALLA)

7. En el conector CONN_20x2 de la edu-ciaa, armar un loopback a través de puerto serie Uart3:

    Pin 88 {P2_4} bornera 23 [RX] --/\/\/\-- [TX] 25 bornera {P2$_3$} Pin 87

    a través de un resistor de valor igual o superior a 1k.

    a. Modificar el programa del ejercicio 2 para redireccionar el envió del carácter de la UART2 a la UART3.
    b. Modificar el programa del ejercicio 3 para redireccionar la recepción del carácter de la UART2 a la UART3.
    c. Testear que cuando esté conectado el resistor de loopback, se puedan prender o apagar 2 de los leds con los 4 pulsadores.

    PULSADOR -> LED -> UART3.TX -> RESISTOR -> UART3.RX -> LED

8. En el conector CONN_20x2 de la edu-ciaa armar un loopback a través de puerto serie Uart3:

    Pin 88 {P2_4} bornera 23 [RX] --/\/\/\-- [TX] 25 bornera {P2$_3$} Pin 87

    a través de un resistor de valor igual o superior a 1k.

    a. Modificar el programa 3 para forwardear el carácter que ingresa por RX de la uart USB de debug al RX de la UART3.
    b. Agregar la funcionalidad de forwardear el carácter que ingresa por el pin 88 RX de la UART3 al TX de la uart USB de debug.
    c. Testear que cuando esté conectado el resistor de loopback se reciba por la consola un eco de lo escrito por la misma.

    TECLA -> TERMINAL (PC) -> UART (USB).RX -> LED -> UART3.TX -> RESISTOR -> UART3.RX -> LED -> UART (USB).TX -> TERMINAL (PC) -> TEXTO(PANTALLA)

Implementación para el chip utilizado en el TPF

Luego de haber completado estos pasos, se pretende que estas funciones sirvan de base para la implementación del trabajo final. Proponer cómo van a adaptar la funcionalidades básicas de la SAPI para que sean de utilidad para el trabajo final.