Practica 3:

Puertos de Entrada/Salida

Hecho por Vladislav Kravchenko

En el desarrollo de la practica 3 hemos utilizado diversos registros para realizar correctamente lo que nos pedían. Además, se siguen los pasos indicados en el documento de la práctica; estos pasos consisten en realizar diferentes funciones que realicen diferentes acciones para concluir todo ello en realizar el objetivo de la práctica. Ahora pasamos a analizar lo que realiza cada función independientemente:

Primeramente, configuramos los pines de entrada/salida de la placa MiniDK2. Para ellos disponemos de la función **ConfigPines ();** Para realizar esta acción sabemos que las entradas serán 0 y las salidas 1. Ayudándonos del registro FIODIR y lo dicho anteriormente configuramos todos los pines según lo descrito en el documento de la práctica.

- La función **delay ();** sirve para realizar el retardo necesario que nos piden en la práctica; estos son 125ms y 400ms. Esta función produce un retardo de 25ms.
- Para recibir los datos de los switches tenemos la función **leedato ();** que gracias al registro FIOPIN lee los correspondientes puertos y los asocia a un bit de la variable dato, que será el dato de partida desde el cual calcularemos los números primos.
- La función **muestrabinario ();** es la encargada de mostrar en los 16 leds el número primo hallado, para ello nos ayudamos de los registros FIOSET y FIOCLR.
- La función **muestrabcd ();** tiene el mismo objetivo que la función **muestrabinario ();** pero esta vez solo nos ayudamos del registro FIOSET. En este registro la operación realizada es la de convertir el número en binario con las sucesivas divisiones entre 2 y ver el resto de esta operación, lo que nos da el número en binario. Para realizar la correcta representación en binario o BCD, en estas dos últimas funciones tenemos que realizar una limpieza de los puertos encargados de representar el número primo, ya que si no lo hacemos tendremos una sobre escritura que tendrá como consecuencia la errónea representación en estos dos formatos.
- La función **binario_a_bcd ();** realiza la conversión del número de binario a BCD. Con esto tendremos identificados las decenas de millar, unidades de millar, centenas, decenas y unidades de cada número primo.
- La función **isp ();** nos devuelve el valor 1 si el botón ISP no está pulsado, y un 0 si ISP esta pulsado. Para saber si esta pulsado o no el botón ISP en cada momento, nos ayudamos del registro FIOPIN que leerá del pin asociado a ISP.
- La función **muestrabin_o_bcd ();** nos determina según si el botón ISP esta pulsado o no, si el número primo se representa en binario o BCD. Si la función **isp ();** devuelve un 1 (no está pulsado) se representa en BCD, y por el contrario en binario.

- La función **display ();** realiza la representación de cada número primo en el display de 7 segmentos. En esta función nos ayudamos de los registros FIOCLR y FIOSET, que encienden o apagan los leds del display para representar el número correspondiente.
- La función **displaydigito ();** sirve para representar el número en el display, esta función cada 400ms nos muestra un digito de nuestro número primo, empezando por las decenas de millar, continuando con las unidades de millar, centenas, decenas y unidades.
- La función **ld2 ();** hace parpadear el LED1 de la placa MiniDK2 cada 125ms. Para ello nos ayudamos de esta sentencia:

LPC_GPIO3->FIOPIN ^= (1<<25);

- La función **ld3 ();** recibe como información el número primo y la posición del bit a considerar. Si el bit en esa posición es un 1 el LED de la placa se enciende. Por el contrario, si el bit es 0 el LED se apaga. Esta función considera todos los bits del número primo.
- En el **main ();** nos ayudamos del programa realizado en la practica 2. Cuando detectamos un número primo, realizamos un código que tarda 2 segundos en realizar la completa representación del número primo en los LEDS, en el display y en los LEDS de la placa MiniDK2. Para seguir la temporización que nos viene indicada en el documento de la práctica nos ayudamos de la función **delay ();** y la función for que tendrá 80 ciclos de 25ms.