Informe Puerto serie

Diseño:

Trabajaremos sobre la implementación de la primer parte. Para poder agregar la funcionalidad buscada, necesitamos: _ un protocolo de comunicación para poder luego coordinar la recepción e impresión del mensaje. Definimos uno muy sencillo: los mensajes pueden tener una longitud máxima fija y cualquier carácter a excepción del carácter '\0', que indica fin de mensaje.

_ manejar la recepción de caracteres a través del puerto serie, almacenándolos en un buffer. Para simplificar la implementación emplearemos un buffer circular. Por ejemplo, no debemos entonces implementar mecanismos de control de desborde de búffer.

_ una rutina para trasladar el contenido del buffer de recepción a una instancia del t.a.d. "String", que es la que usará la ISR del puerto paralelo para imprimir el mensaje.

_ un mecanismo para coordinar la rutina de recepción de mensajes y la de traslado del contenido del buffer al display.

Dividiremos las responsabilidades del siguiente modo: _ la ISR del puerto serie se encargará de recibir cada carácter y almacenarlo en el buffer, teniendo en cuenta el protocolo definido.

_ en el programa principal implementaremos la rutina necesaria para trasladar el contenido del buffer a una instancia de t.a.d. "String". La ISR del puerto paralelo se encargará de hacer efectiva la impresión del mensaje. Aquí empleamos lo ya implementado en la primer parte del proyecto.

_ coordinaremos la ISR y el programa principal mediante el uso de variables globales: una para indicar si se ha recibido el carácter de fin de mensaje y otra para indicar que se ha recibido y almacenado un nuevo carácter.

Implementación

Rutina de servicio de interrupción del UART:

El funcionamiento de la rutina sigue los principios básicos: desactivar el flag de interrupciones del micro para asegurar la atomicidad de la ejecución de la rutina, realizar las operaciones requeridas para atender la interrupción, enviar un EOI al PIC y habilitar el flag de interrupciones del micro al terminar.

Para la comunicación a través del puerto serie adoptamos el siguiente protocolo: los mensajes puede contener cualquier largo y cualquier caracter a excepción del caracter '\0' que indicará el fin del mensaje.

Durante la rutina de tratamiento de la interrupción hacemos uso de 2 banderas globales para sincronizar las operaciones de almacenamiento, en un búffer, de los datos recibidos a través del puerto serie e impresión a través del puerto lpt1. Estas banderas son END_MSG (para indicar si se ha recibido el caracter de fin de mensaje) y NEW_MESSAGE (para indicar que se ha recibido un nuevo caracter).

Al recibir un nuevo caracter en el puerto, producirse la interrupción y ejecutarse nuestra rutina, comenzamos por leer el registro de datos del puerto (ubicado en la dirección 0x3F8). Almacenamos lo leído en una nueva posición en nuestro búffer de recepción. Lo manejamos como un búffer circular, de allí que calculemos la próxima posición mediante el cálculo del módulo con el largo del búffer. Si el caracter leído es el de final de mensaje, levantamos la bandera END_MSG.

Programa principal:

En la primera parte de este fragmento de código generamos una serie de inicializaciones. Aquí definimos las variables "int_lpt" y "int_uart" en donde se guardan las direcciones de las IRQ's de los puertos. Se puede observar que se hace una suma de +8 y +4 respectivamente debido a que los IRQ's empiezan en las direcciones 08h. También indicamos las direcciones adonde el pic

tiene que enmascarar o desenmascarar interrupciones en las variables "pic_mask_lpt" y "pic_mask_uart" y la dirección del pic en "picaddr".

Después de definir algunas variables locales, pasamos a hacer un manejo del puerto serie uart. Primero que todo salvamos los estados de los registros LCR e IER del uart y los guardamos en las variables "uart_ctrl" y "uart_int". Deshabilitamos las interrupciones del uart para no tener otras complicaciones y despues hacemos un cambio en el vector de interrupciones para sacar la interrupción que está en ese momento en la dirección definida previamente en "int_uart" y poner la nuestra. La interrupción vieja se guarda para restablecerla mas tarde. Pasamos a configurar el baudeaje de bits. Para ello, primero ponemos a DLAB = 1 para poder acceder al divisor Latch register. Ahora tendremos a DATA_S en LSB y IER en MSB, escogemos un baudeaje de 4800 dándole la dirección 0x18h (0001 1000) habilitando los bits correspondientes. Una vez elegido el baudeaje, escogemos nuestro modo de trabajo poniendo los bits correspondientes en el LCR para los modos de DLAB := 0, Parity enabled, even parity, 2 stop bits y 8 bit word length. Pasamos a que el pic desenmascare el uart para recibir interrupciones y como anteriormente habíamos seteado de vuelta DLAB=0 pasamos a configurar al uart para que solamente reciba datos. Una vez configurado nuestro puerto serie, pasamos a configurar nuestro puerto paralelo. En esta configuración hacemos una rutina parecida, se asegura que el puerto este en la dirección correcta, se reemplaza nuestra interrupción por la que está en ese momento, desenmascaramos el puerto para que el pic reciba interrupciones del puerto y habilitamos las transmisiones del mismo. Una vez que ya tengamos nuestros puertos configurados, seteamos nuestro timer de impresión.

A partir de aquí, ya podemos definir nuestro main program. Lo que se realiza aquí es primero que todo recibir todo el mensaje por el puerto serie uart, todo esto mientras no se realice ningún suceso de teclado. No se imprimirá absolutamente nada hasta no completar el mensaje. Una vez obtenido el mismo, nos fijamos si tenemos un nuevo mensaje a traves de la variable (manejada por nuestra interrupción) "NEW_MESSAGE" y pasamos a imprimir a los displays a través del puerto paralelo de la misma forma en que lo hacíamos antes.

Desinstalación de los puertos paralelo y serie:

Tras la culminación de la recepción e impresión de mensajes: _deshabilitamos las interrupciones del puerto paralelo, poniendo a 0 el bit IRQEN del registro de control. Como no queremos cambiar el estado de los restantes bits, leeremos el contenido del puerto y efectuaremos una operación AND lógica bit a bit con el byte EF. El resultado será lo que escribiremos en el puerto.

_enmascaramos las líneas de interrupción del PIC que corresponden al puerto paralelo (IRQ7) y al puerto serie COM1 (IRQ4). Como no queremos alterar los restantes bits del registro IMR, realizamos una operación OR lógica bit a bit entre el contenido del registro y el byte cuyo único bit activo se corresponde con el número de IRQ de interés. El resulto de esto será lo que escribimos en el registro IMR.

_restablecemos el UART al estado en el que se encontraba antes de ser usado, mediante la escritura en sus registros IER y LCR del contenido que tenían al inicio de nuestro programa. Este contenido fue almacenado temporalmente en las variables uart_int (para el registro IER) y uart_ctrl (para el registro LCR).

_restablecemos las entradas del vector de rutina de interrupción en las que hemos almacenado las direcciones a nuestras ISRs.

_liberamos los recursos de memoria empleados por nuestros tads.

La implementación de la solución propuesta está en el archivo “uart_int_con_bucle.c”.