EDUARDO MORA GONZÁLEZ

PRACTICA 3

SISTEMAS EMPOTRADOS Y UBICUOS

Ejercicio 1

Este ejercicio solo tiene código en el fichero *main.c* el código es el siguiente:

```
char message1[] = "Hola mundo";
char message2[] = "Adios mundo";                                   1

for (;;) {
    if (AS1_RecvChar(65) == ERR_OK) {
        for (int i = 0; i < sizeof(message1); i++) {
            while (AS1_SendChar((byte) message1[i]) != ERR_OK) {
            }
        }
    }                                                              2
    if (AS1_RecvChar(66) == ERR_OK) {
        for (int i = 0; i < sizeof(message2); i++) {
            while (AS1_SendChar((byte) message2[i]) != ERR_OK) {
            }
        }
    }                                                              3
}
```

El código se divide en 3 bloques distintos:

1) En este bloque están los mensajes que queremos enviar dependiendo que carácter recibamos.
2) En este bloque esperamos a recibir la letra ‘A’, si lo recibimos respondemos con el mensaje correspondiente.
3) En este bloque esperamos a recibir la letra ‘B’, si lo recibimos respondemos con el mensaje correspondiente.

El funcionamiento del programa es el siguiente:

```
05/03/2021 15:37:27.676 [TX] - b
05/03/2021 15:37:27.689 [RX] - Adios mundo<NUL>
05/03/2021 15:37:28.482 [TX] - a
05/03/2021 15:37:28.492 [RX] - Adios mundo<NUL>
05/03/2021 15:38:11.909 [TX] - b
05/03/2021 15:38:12.034 [RX] - <NUL> Hola mundo<NUL>
05/03/2021 15:38:12.817 [TX] - b
05/03/2021 15:38:12.829 [RX] - Hola mundo<NUL>
05/03/2021 15:38:13.619 [TX] - b
05/03/2021 15:38:13.628 [RX] - Hola mundo<NUL>
```

Respondiendo a la pregunta *¿Qué problemas aparecen?* Como se observa en la imagen a veces la comunicación no es precisa y cuando cambio el carácter me envía un <NUL>.

Ejercicio 2

Este es similar al anterior, pero esta vez la recepción se hace mediante el uso de interrupciones. El código del archivo Events.c es:

```
int opcion = 0;
void AS1_OnRxChar(void) {

    byte rev;

    if (AS1_RecvChar(&rev) == ERR_OK) {

        if (rev == 65)
            opcion = 1;
        if (rev == 66)
            opcion = 2;
    }

}
```

He definido una variable compartida para que dependiendo el carácter que reciba tenga una opción u otra.

Una vez dentro del método, declaro una variable para almacenar los valores que voy recibiendo, en el caso de recibir el carácter A (65) o el B (66) la variable opción toma un valor, en el resto de los casos se ignoran los valores.

El código del fichero main.c es:

```
char message1[] = "Hola mundo";
char message2[] = "Adios mundo";
int i;

for (;;) {
    if (opcion == 1) {
        for (i = 0; i < sizeof(message1); i++)
            while (AS1_SendChar((byte) message1[i]) != ERR_OK) {
            }

        opcion = 0;
    }
    if (opcion == 2) {
        for (i = 0; i < sizeof(message2); i++)
            while (AS1_SendChar((byte) message2[i]) != ERR_OK) {
            }

        opcion = 0;
    }
}
```

1
2
3

El código se divide en 3 bloques distintos:

1) En este bloque están los mensajes que queremos enviar dependiendo que carácter recibamos.
2) Este bloque ejecuta la opción 1 donde se envía el primer mensaje.
3) Este bloque ejecuta la opción 2 donde se envía el segundo mensaje.

La salida del programa es la siguiente:

```
06/03/2021 15:09:01.669 [TX] - A
06/03/2021 15:09:01.679 [RX] - <NUL>Hola mundo<NUL>
06/03/2021 15:09:03.189 [TX] - B
06/03/2021 15:09:03.193 [RX] - Adios mundo<NUL>
06/03/2021 15:09:04.898 [TX] - A
06/03/2021 15:09:04.902 [RX] - Hola mundo<NUL>
06/03/2021 15:09:05.600 [TX] - A
06/03/2021 15:09:05.605 [RX] - Hola mundo<NUL>
06/03/2021 15:09:06.702 [TX] - B
06/03/2021 15:09:06.716 [RX] - Adios mundo<NUL>
06/03/2021 15:09:07.388 [TX] - B
06/03/2021 15:09:07.397 [RX] - Adios mundo<NUL>
```

Ejercicio 3

En este ejercicio se recibe un bloque con la siguiente estructura:

```
struct {
    byte comando;
    byte valor;
} datos;
```

Para recibir la estructura usamos el método del fichero events.c cuyo código es el siguiente:

```
void AS1_OnFullRxBuf(void) {
    word Recibido;

    while (AS1_RecvBlock((byte*) &datos, sizeof(datos), &Recibido) != ERR_OK)
        ;
}
```

Dependiendo de los valores que vaya tomando la estructura se realizará una acción u otra. Las acciones están definidas en el archivo del enunciado:

0x00 y “A”, debe enviarse por el puerto serie el mensaje: “Dame 5 minutos”.

0x00 y “B”, debe enviarse por el puerto serie el mensaje: “Eso no es asunto mio”

0x0F y “A”, debe enviarse por el puerto serie el mensaje: “Ahora mismo”.

0x0F y “B”, debe enviarse por el puerto serie el mensaje: “Ya lo he hecho”

Esto se controla en el fichero main.c:

```
char message1[] = "Dame 5 minutos";
char message2[] = "Eso no es asunto mio";
char message3[] = "Ahora mismo";
char message4[] = "Ya lo he hecho";
```
1

```
word env;
int i;
byte err;
word Sent;

for (;;) {

    comando = datos.valor;
    valor = datos.comando;

    if (comando == 0x00 && valor == 0X41) {

        for (i = 0; i < sizeof(message1); i++)
            while (AS1_SendChar((byte) message1[i]) != ERR_OK) {
            }
    }
```
2

```
    if (comando == 0x00 && valor == 0X42) {

        for (i = 0; i < sizeof(message2); i++)
            while (AS1_SendChar((byte) message2[i]) != ERR_OK) {
            }

    }
```
3

```
    if (comando == 0x0F && valor == 0X41) {

        for (i = 0; i < sizeof(message3); i++)
            while (AS1_SendChar((byte) message3[i]) != ERR_OK) {
            }

    }
```
4

```
    if (comando == 0x0F && valor == 0X42) {

        for (i = 0; i < sizeof(message4); i++)
            while (AS1_SendChar((byte) message4[i]) != ERR_OK) {
            }

    }
```
5

```
    datos.comando = 0x0;
    datos.valor = 0x0;
```
6

```
}
```

El código se divide en 6 bloques distintos:

1) En este bloque están los mensajes que queremos enviar dependiendo de los datos de la estructura.
2) Este bloque ejecuta la primera acción donde se envía el primer mensaje, para saber si el contenido del bloque es el de la primera acción en la estructura de **if** se hace la comparación de los elementos contenidos en el bloque, pero traducidos a Hexadecimal.
3) Igual que el anterior, pero comparando con la segunda opción.
4) Igual que el anterior, pero comparando con la tercera opción.
5) Igual que el anterior, pero comparando con la cuarta opción.
6) Limpiamos los datos de la estructura a la espera de una nueva recepción,

La salida es la siguiente:

```
06/03/2021 13:13:12.787 [TX] - <SI>B
06/03/2021 13:13:12.795 [RX] - Dame 5 minutos<NUL>
06/03/2021 13:13:12.990 [TX] - <SI>B
06/03/2021 13:13:13.367 [TX] - <SI>B
06/03/2021 13:13:13.375 [RX] - Eso no es asunto mio<NUL>
06/03/2021 13:13:13.555 [TX] - <SI>B
06/03/2021 13:13:13.986 [TX] - <SI>B
06/03/2021 13:13:13.994 [RX] - Ya lo he hecho<NUL>
06/03/2021 13:13:14.177 [TX] - <SI>B
```

Notad que a veces la comunicación no es perfecta del todo y tarda en responder o responde un dato incorrecto.

Ejercicio 4

En este ejercicio usamos dos pulsadores, dependiendo de cual pulsemos se realiza una acción u otra, para controlar esto usamos las interrupciones al pulsar y dependiendo de cual pulsemos y las veces que lo hagamos la variable **opción** toma un valor u otro.

```
int cont = 0;
int opcion = 0;
void switch3_OnInterrupt(void) {
    opcion = -1;
}
```

1

```
void switch2_OnInterrupt(void) {

    cont++;
    if (cont > 3)
        cont = 0;

    switch (cont) {
    case 1:
        opcion = 1;
        break;
    case 2:
        opcion = 2;
        break;
    case 3:
        opcion = 3;
        break;
    }

}
```

2

En el archivo main.c dependido del valor de la variable **opción** se realiza una acción u otra. Lo primero que hacemos es definirlos mensajes y las variables que usamos para asignar los valores. También activamos el acelerómetro.

```
FX1_Enable();

unsigned int x;
unsigned int y;
unsigned int z;

int i;

char mensajeX[] = " X: ";
char mensajeY[] = " Y: ";
char mensajeZ[] = " Z: ";
```

El siguiente código muestra el flujo que se sigue dependiendo del pulsador que se pulse:

```
for (;;) {
    x = FX1_MeasureGetRawX();
    y = FX1_MeasureGetRawY();
    z = FX1_MeasureGetRawZ();                                   1
    if (opcion > 0) {
        switch (opcion) {
        case 1:
            for (i = 0; i < sizeof(mensajeX); i++)
                while (AS1_SendChar((byte) mensajeX[i]) != ERR_OK) {
                }
            UART_Write_Numero_Int(x);
            while (AS1_SendChar(10) != ERR_OK) {
            }
            while (AS1_SendChar(13) != ERR_OK) {
            }
            break;
        case 2:
            for (i = 0; i < sizeof(mensajeY); i++)
                while (AS1_SendChar((byte) mensajeY[i]) != ERR_OK) {
                }
            UART_Write_Numero_Int(y);
            while (AS1_SendChar(10) != ERR_OK) {
            }
            while (AS1_SendChar(13) != ERR_OK) {
            }
            break;
        case 3:
            for (i = 0; i < sizeof(mensajeZ); i++)
                while (AS1_SendChar((byte) mensajeZ[i]) != ERR_OK) {
                }
            UART_Write_Numero_Int(z);
            while (AS1_SendChar(10) != ERR_OK) {
            }
            while (AS1_SendChar(13) != ERR_OK) {
            }
            break;
        }                                                       2
        opcion = 0;
    } else if (opcion < 0) {
        for (i = 0; i < sizeof(mensajeX); i++)
            while (AS1_SendChar((byte) mensajeX[i]) != ERR_OK) {
            }
        UART_Write_Numero_Int(x);

        for (i = 0; i < sizeof(mensajeY); i++)
            while (AS1_SendChar((byte) mensajeY[i]) != ERR_OK) {
            }
        UART_Write_Numero_Int(y);

        for (i = 0; i < sizeof(mensajeZ); i++)
            while (AS1_SendChar((byte) mensajeZ[i]) != ERR_OK) {
            }
        UART_Write_Numero_Int(z);
        while (AS1_SendChar(10) != ERR_OK) {
        }
        while (AS1_SendChar(13) != ERR_OK) {
        }                                                       3
        WAIT1_Waitms(500);                                      4
    }
```

El código se divide en 4 bloques distintos:

1) En este bloque se obtiene los valores del acelerómetro.

2) Este bloque ejecuta la primera acción donde cada vez que se pulse el pulsador se envía el valor de X, Y, Z respectivamente.
3) Este bloque ejecuta la segunda acción cuando se pulse el pulsador. La segunda acción envía los valores de X, Y, Z.
4) Este bloque es la espera que se realiza antes de volver a realizar el envío (como en el bloque 3).

Para enviar los datos, se hace uso del método *UART_Write_Numero_Int* que dada la salida del acelerómetro lo convierte a entero y lo envía.

La salida del programa es la siguiente: