TP 04 - Eliminando Bloqueos

Bloqueos identificados

Los bloqueos encontrados pertenecen al módulo `pc_serial_com`, en el archivo `modules/pc_serial_com/pc_serial_com.cpp`. En este archivo podemos encontrar el siguiente fragmento de código correspondiente a la funcion `commandSetDateAndTime()`

```
static void commandSetDateAndTime()
{
    char year[5] = "";
    char month[3] = "";
    char day[3] = "";
    char hour[3] = "";
    char minute[3] = "";
    char second[3] = "";

    pcSerialComStringWrite("\r\nType four digits for the current year (YYYY):
");
    pcSerialComStringRead( year, 4);
    pcSerialComStringWrite("\r\n");

    pcSerialComStringWrite("Type two digits for the current month (01-12): ");
    pcSerialComStringRead( month, 2);
    pcSerialComStringWrite("\r\n");

    pcSerialComStringWrite("Type two digits for the current day (01-31): ");
    pcSerialComStringRead( day, 2);
    pcSerialComStringWrite("\r\n");

    pcSerialComStringWrite("Type two digits for the current hour (00-23): ");
    pcSerialComStringRead( hour, 2);
    pcSerialComStringWrite("\r\n");

    pcSerialComStringWrite("Type two digits for the current minutes (00-59):
");
    pcSerialComStringRead( minute, 2);
    pcSerialComStringWrite("\r\n");

    pcSerialComStringWrite("Type two digits for the current seconds (00-59):
");
    pcSerialComStringRead( second, 2);
    pcSerialComStringWrite("\r\n");

    pcSerialComStringWrite("Date and time has been set\r\n");

    dateAndTimeWrite( atoi(year), atoi(month), atoi(day),
        atoi(hour), atoi(minute), atoi(second) );
}
```

Esta funcion es ejecutada cuando el usuario presiona la tecla `s` ó `S` en la terminal para ingresar una fecha y hora en el sistema. Es bloqueante porque puede verse que el programa se quedara en esta funcion hasta que el usuario ingrese todos los caracteres necesarios debido al

uso de la función `pcSerialComStringRead()`, donde el programa se quedará esperando los caracteres correspondientes a año, mes, dia, hora, minuto y segundo.

Durante la espera de la función `pcSerialComStringRead()` el programa no realizará ninguna otra lógica.

El codigo de la funcion `pcSerialComStringRead()` se puede ver a continuación

```
static void pcSerialComStringRead( char* str, int strLength )
{
    int strIndex;
    for ( strIndex = 0; strIndex < strLength; strIndex++) {
        uartUsb.read( &str[strIndex] , 1 );
        uartUsb.write( &str[strIndex] ,1 );
    }
    str[strLength]='\0';
}
```

Solución propuesta

Segun lo visto durante la teórica, la forma más elegante de solucionar este problema, será visto mas adelante en la materia, en donde esta funcion sera reemplazada como una tarea del RTOS.

Por lo tanto, se planteó una logica similiar a la ya existente para el ingreso de una nueva clave para el desbloqueo de la alarma.

De esta forma, se agrego un nuevo estado a la FSM, que detecta si el programa está en la espera de caracteres de la terminal que se activa cuando el usuario presiona la tecla `s` ó `S`.

```
typedef enum{
    PC_SERIAL_COMMANDS,
    PC_SERIAL_GET_CODE,
    PC_SERIAL_SAVE_NEW_CODE,
    PC_SERIAL_SET_DATE_AND_TIME, /// CAMBIO: Nuevo estado para la FSM, agregado
para TP 04
} pcSerialComMode_t;
```

Por lo que la funcion `pcSerialComUpdate()` quedó con las sifuientes modificaciones

```
void pcSerialComUpdate()
{
    char receivedChar = pcSerialComCharRead();
    if( receivedChar != '\0' ) {
        switch ( pcSerialComMode ) {
            case PC_SERIAL_COMMANDS:
                pcSerialComCommandUpdate( receivedChar );
            break;

            case PC_SERIAL_GET_CODE:
                pcSerialComGetCodeUpdate( receivedChar );
            break;

            case PC_SERIAL_SAVE_NEW_CODE:
                pcSerialComSaveNewCodeUpdate( receivedChar );
            break;

            ///CAMBIO: Agregado para TP 04
            case PC_SERIAL_SET_DATE_AND_TIME:
                pcSerialComSetDateAndTime( receivedChar );
            default:
                pcSerialComMode = PC_SERIAL_COMMANDS;
            break;
        }
    }
}
```

Las nuevas funciones agregadas al módulo fueron:

- `pcSerialComSetDateAndTime()`
- `setDateAndTime()`

Y resultaron

```
/// CAMBIO: Funcion agregada para TP 04
static void pcSerialComSetDateAndTime( char receivedChar )
{
    static char dateAndTimeSequence[TIME_AND_DATE_NB_OF_CHARS];

    dateAndTimeSequence[numberOfDateAndTimeChars] = receivedChar;
    numberOfDateAndTimeChars++;

    if ( numberOfDateAndTimeChars >= TIME_AND_DATE_NB_OF_CHARS ) {
        pcSerialComMode = PC_SERIAL_COMMANDS;
        numberOfDateAndTimeChars = 0;
        setDateAndTime( dateAndTimeSequence );
        pcSerialComStringWrite("Date and time has been set\r\n");
    }
}
```

```
/// CAMBIO: Funcion agregada para TP 04
static void setDateAndTime( char *dateAndTimeSequence )
{
    char year[5] = "";
    char month[3] = "";
    char day[3] = "";
    char hour[3] = "";
    char minute[3] = "";
    char second[3] = "";

    strncpy(year, dateAndTimeSequence, 4);
    year[4] = '\0';

    strncpy(month, dateAndTimeSequence+MONTH_OFFSET, 2);
    month[2] = '\0';

    strncpy(day, dateAndTimeSequence+DAY_OFFSET, 2);
    day[2] = '\0';

    strncpy(hour, dateAndTimeSequence+HOUR_OFFSET, 2);
    hour[2] = '\0';

    strncpy(minute, dateAndTimeSequence+MINUTE_OFFSET, 2);
    minute[2] = '\0';

    strncpy(second, dateAndTimeSequence+SECOND_OFFSET, 2);
    second[2] = '\0';

    dateAndTimeWrite( atoi(year), atoi(month), atoi(day),
        atoi(hour), atoi(minute), atoi(second) );

}
```

Por último, se muestra como quedó la modificacion de la funcion `commandSetDateAndTime()`

```
/// CAMBIO: Nueva funcion commandSetDateAndTime(). Modificada para TP 04
static void commandSetDateAndTime()
{
    pcSerialComStringWrite( "Please enter date and time: " );
    pcSerialComStringWrite("\r\n- Type four digits for the current year (YYYY)
");
    pcSerialComStringWrite("\r\n- Type two digits for the current month (MM)
(01-12) ");
    pcSerialComStringWrite("\r\n- Type two digits for the current day (DD) (01-
31) ");
    pcSerialComStringWrite("\r\n- Type two digits for the current hour (hh)
(00-23) ");
    pcSerialComStringWrite("\r\n- Type two digits for the current minutes (mm)
(00-59) ");
    pcSerialComStringWrite("\r\n- Type two digits for the current seconds (ss)
(00-59) ");
    pcSerialComStringWrite("\r\n\r\nRespect the format: ");
    pcSerialComStringWrite("YYYY MM DD hh mm ss");
    pcSerialComStringWrite( "\r\n(Note the space between each of the fields)"
);
    pcSerialComStringWrite("\r\n");

    numberOfDateAndTimeChars = 0;
    pcSerialComMode = PC_SERIAL_SET_DATE_AND_TIME;
}
```

Con todos estos cambios, se puede ver que ahora el programa no interrumpe su ejecución a la espera del ingreso de los caracteres, sino que se quedará en uno de los posibles estados de la FSM y por ende ya no estará en estado de "bloqueo".

> **Nota:**
>
> Se tuvieron que agregar algunos `defines` para hacer mas entendible el código. Estos fueron

```
/** CAMBIO:
Define agregado para TP 04:
 - 4 caracteres por el año + 1 por el espacio = 5
 - 2 caracteres por el mes + 1 por el espacio = 3
 - 2 caracteres por el dia + 1 por el espacio = 3
 - 2 caracteres por las hora + 1 por el espacio = 3
 - 2 caracteres por los minutos + 1 por el espacio = 3
 - 2 caracteres por los segundos = 2
 - 1 caracter mas por el '\0' = 1
 TOTAL DE CARACTERES = 20
*/
#define TIME_AND_DATE_NB_OF_CHARS 20

/// CAMBIO: Defines para el offset de la cadena de caracteres de fecha y
hora. Agregado para TP 04
#define MONTH_OFFSET 5
#define DAY_OFFSET 8
#define HOUR_OFFSET 11
#define MINUTE_OFFSET 14
#define SECOND_OFFSET 17
```