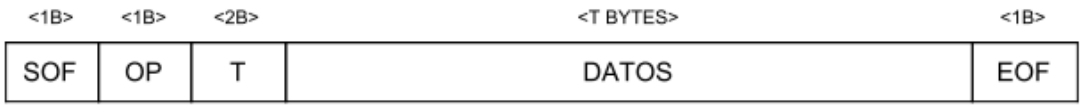
TRABAJO FINAL – RTOS 2 – CESE
GRUPO 1 v24-05-2019

INTEGRANTES:
Julian Bustamante Narvaez
Jacobo Salvador
Gustavo Paredes D.
Rafael Oliva

Introducción

PROBLEMA

Paquetes de datos:



Delimitación de paquete: 00 a 99

SOF: Carácter '{'
EOF: Carácter '}'.

Campos:

OP (Operación):
0: Convertir los caracteres recibidos a mayúsculas. (CMD/RTA)
1: Convertir los caracteres recibidos a minúsculas. (CMD/RTA)
2: Reportar stack disponible (RTA)
3: Reportar heap disponible. (RTA)

T (Tamaño): El tamaño del campo DATOS. "00" a "99" o sea máximo 99 bytes
DATOS: Texto a procesar. Deben ser caracteres ASCII legibles.

1) En FreeRTOSBlinky.c
   Transmisión inicializada utilizando Interrupcion por TxBuf vacío:

```
//Requerimiento 2.5 – Transmision x buffer vacio
        uartTxInterruptCallbackSet( UART_USB, Transmit_UART );
        uartTxInterruptSet( UART_USB, true );

// En la creación de Tasks Mayusculas y minusculas agregar task Handles..
// Esto para ver el stack disponible
            xTaskHandle_MayOP0
            xTaskHandle_MinOP1
```

2) En task.c

```
2.1) Callback TX por ISR
/*====================================================================================
 | Callback IT TX |
  // Falta definir readBuffer(char *buffer, char *ByteToTx);
 ====================================================================================*/
void Transmit_UART ( void* noUsado )
{
        static int start_detected = 0;
        char Txbyte;
        if( readBuffer( &Txbuffer, &Txbyte ) ){

                uartTxWrite( UART_USB, Txbyte );
        }

}

2.2) Task TX - modificaciones
/*====================================================================================
                                                                        | Tarea tx |
 ====================================================================================*/
void TaskTxUart( void* taskParmPtr ){
        char * rx;
        while(true){
                        /*Recibe por la cola*/
                        rx = ModuleDinamicMemory_receive(&ModuleData, xPointerQueue_3, portMAX_DELAY);

                        sprintf(Txbuffer,"%c",Frame_parameters._SOF);
                        sprintf(Txbuffer,"%s",Frame_parameters.Operation);
                        sprintf(Txbuffer,"%s",atoi(Frame_parameters.T));
                        sprintf(Txbuffer,"%s",rx);
                        sprintf(Txbuffer,"%c",Frame_parameters._EOF);

                        /*Libera memoria dinamica*/
                        ModuleDinamicMemory_Free(&ModuleData, rx);
                        if( uartTxReady( UART_USB ) ){
                                // La primera vez – con esto arranca
                                Transmit_UART( 0 );
                        }
                }
}


2.3) Task _toMayusculas - modificaciones
/*====================================================================================
                              | Tarea Mayusculizar |
 ====================================================================================*/
void Task_ToMayusculas_OP0( void* taskParmPtr ){
        char * rx;
        while(1){

                rx = ModuleDinamicMemory_receive(&ModuleData,xPointerQueue_OP0,  portMAX_DELAY);
                //PrintUartMessageMutex("Task_ToMayusculas_OP0", SemMutexUart);
                //PrintUartBuffMutex( "mayus %s\r\n",rx,SemMutexUart);
                // pasar string a mayusculas
                packetToUpper(rx);
                // Enviar a cola de TaskTxUARt
                ModuleDinamicMemory_send(&ModuleData,0,NULL,rx, xPointerQueue_3,portMAX_DELAY);
                /*Libera memoria dinamica*/
                ModuleDinamicMemory_Free(&ModuleData, rx);
```

```
        }
}
```

2.4) Task _toMinusc - modificaciones

```
/*=====================================================================================
                                | Tarea Minusculizar |
 ======================================================================================*/
void Task_ToMinusculas_OP1( void* taskParmPtr ){
        char * rx;
        while(1){

                rx = ModuleDinamicMemory_receive(&ModuleData,xPointerQueue_OP1,  portMAX_DELAY);
                //PrintUartMessageMutex("Task_ToMinusculas_OP1", SemMutexUart);
                //PrintUartBuffMutex( "Minus %s\r\n",rx,SemMutexUart);
                packetToLower(rx);
                // Enviar a cola de TaskTxUARt
                ModuleDinamicMemory_send(&ModuleData,0,NULL,rx, xPointerQueue_3,portMAX_DELAY);
                /*Libera memoria dinamica*/
                ModuleDinamicMemory_Free(&ModuleData, rx);
        }
}
```

2.5) Reportar Stack disponible

```
/*=====================================================================================
                  | Tarea Reportar stack disponible |
 ======================================================================================*/
void Task_ReportStack_OP2( void* taskParmPtr ){
        UBaseType_t uxHighWaterMark;
        char *rx;
        char tempStack[30];
        while(1){
            rx = ModuleDinamicMemory_receive(&ModuleData,xPointerQueue_OP2,  portMAX_DELAY);
            if(Frame_parameters.Operation ==OP0)
                {uxHighWaterMark = uxTaskGetStackHighWaterMark( &xTaskHandle_MayOP0);}
            else{
                uxHighWaterMark = uxTaskGetStackHighWaterMark(&xTaskHandle_MinOP1);
                }
            memset(tempStack, 0, sizeof(tempStack ) );
            sprintf(tempStack, "%d", uxHighWaterMark );
            strncpy(rx, tempStack, strlen(tempStack));
            // Enviar a cola de TaskTxUARt
            ModuleDinamicMemory_send(&ModuleData,0,NULL,rx, xPointerQueue_3,portMAX_DELAY);
            /*Libera memoria dinamica*/
            ModuleDinamicMemory_Free(&ModuleData, rx);
        }
```

2.6) Task reporter HeapDisp

```
/*=====================================================================================
                          | Tarea Reportar heap disponible |
 ======================================================================================*/
void Task_ReportHeap_OP3( void* taskParmPtr ){
        char *rx;
        char tempHeap[30];
        while(1){
            rx = ModuleDinamicMemory_receive(&ModuleData,xPointerQueue_OP3,  portMAX_DELAY);
            memset(tempHeap, 0, sizeof(tempHeap) );
            sprintf(tempHeap, "%d", xPortGetFreeHeapSize() );
            strncpy(rx, tempStack, strlen(tempStack));
            // Enviar a cola de TaskTxUARt
            ModuleDinamicMemory_send(&ModuleData,0,NULL,rx, xPointerQueue_3,portMAX_DELAY);
            /*Libera memoria dinamica*/
            ModuleDinamicMemory_Free(&ModuleData, rx);
        }

}
```

ANEXO

-0-