```c
/* ************************************************************************** */
/** Descriptive File Name

  @Company
    ETML-ES

  @File Name
    mc32_serComm.c

 */
/* ************************************************************************** */

/* ************************************************************************** */
/* ************************************************************************** */
/* Section: Included Files                                                    */
/* ************************************************************************** */
/* ************************************************************************** */
#include "Mc32_serComm.h"
#include <stdio.h>

/* This section lists the other files that are included in this file.
 */

/* TODO:  Include other files here if needed. */


/* ************************************************************************** */
/* ************************************************************************** */
/* Section: File Scope or Global Data                                         */
/* ************************************************************************** */
/* ************************************************************************** */

/*  A brief description of a section can be given directly below the section
    banner.
 */

/* ************************************************************************** */
/** Descriptive Data Item Name

  @Summary
    Brief one-line summary of the data item.

  @Description
    Full description, explaining the purpose and usage of data item.
    <p>
    Additional description in consecutive paragraphs separated by HTML
    paragraph breaks, as necessary.
    <p>
    Type "JavaDoc" in the "How Do I?" IDE toolbar for more information on tags.

  @Remarks
    Any additional remarks
 */


/* ************************************************************************** */
/* ************************************************************************** */
// Section: Local Functions                                                   */
/* ************************************************************************** */
/* ************************************************************************** */



/* ************************************************************************** */
/* ************************************************************************** */
// Section: Interface Functions                                               */
/* ************************************************************************** */
/* ************************************************************************** */

/*  A brief description of a section can be given directly below the section
    banner.
 */

```

```c
// ***************************************************************************


void serDisplayValues ( s_bno055_data *bno055_data )
{
    char sendBuffer[66] = {0};
    uint8_t i = 0;
    static uint32_t ctnTimeout = 0;

    /* Preapare Gravity string */
    sprintf(sendBuffer, "DT: %d0 ms\tGravity : X = %04.03lf\tY = %04.03lf\tZ = %04.03lf \n\r", (bno055_data->d_time), bno055_data->gravity.x, bno055_data->gravity.y, bno055_data->gravity.z);
    /* Transmit Gravity string */
    do{
        if(!PLIB_USART_TransmitterBufferIsFull(USART_ID_1))
        {
            PLIB_USART_TransmitterByteSend(USART_ID_1, sendBuffer[i]);
            i++;
        }
        ctnTimeout++;
    }while((sendBuffer[i-1] != '\r')&&(ctnTimeout<TIME_OUT));
    i = 0;

    /* Preapare gyroscope string */
    sprintf(sendBuffer, "Gyro    : X = %04.03lf\tY = %04.03lf\tZ = %04.03lf \n\r", bno055_data->gyro.x, bno055_data->gyro.y, bno055_data->gyro.z);
    /* Transmit Gravity string */
    do{
        if(!PLIB_USART_TransmitterBufferIsFull(USART_ID_1))
        {
            PLIB_USART_TransmitterByteSend(USART_ID_1, sendBuffer[i]);
            i++;
        }
        ctnTimeout++;
    }while((sendBuffer[i-1] != '\r')&&(ctnTimeout<TIME_OUT));
    i = 0;

    /* Preapare magnitude string */
    sprintf(sendBuffer, "Mag     : X = %04.03lf\tY = %04.03lf\tZ = %04.03lf \n\r", bno055_data->mag.x, bno055_data->mag.y, bno055_data->mag.z);
    /* Transmit Gravity string */
    do{
        if(!PLIB_USART_TransmitterBufferIsFull(USART_ID_1))
        {
            PLIB_USART_TransmitterByteSend(USART_ID_1, sendBuffer[i]);
            i++;
        }
        ctnTimeout++;
    }while((sendBuffer[i-1] != '\r')&&(ctnTimeout<TIME_OUT));
    i = 0;

    /* Preapare linear acceleration string */
    sprintf(sendBuffer, "Accel   : X = %04.03lf\tY = %04.03lf\tZ = %04.03lf \n\r", bno055_data->linear_accel.x, bno055_data->linear_accel.y, bno055_data->linear_accel.z);
    /* Transmit Gravity string */
    do{
        if(!PLIB_USART_TransmitterBufferIsFull(USART_ID_1))
        {
            PLIB_USART_TransmitterByteSend(USART_ID_1, sendBuffer[i]);
            i++;
        }
        ctnTimeout++;
    }while((sendBuffer[i-1] != '\r')&&(ctnTimeout<TIME_OUT));
    i = 0;

    /* Preapare euler string */
    sprintf(sendBuffer, "Euler   : H = %04.03lf\tP = %04.03lf\tR = %04.03lf \n\r", bno055_data->euler.h, bno055_data->euler.p, bno055_data->euler.r);
    /* Transmit Gravity string */
    do{
        if(!PLIB_USART_TransmitterBufferIsFull(USART_ID_1))
        {
            PLIB_USART_TransmitterByteSend(USART_ID_1, sendBuffer[i]);
            i++;
        }
        ctnTimeout++;
    }while((sendBuffer[i-1] != '\r')&&(ctnTimeout<TIME_OUT));
    i = 0;

    /* Preapare quaternion string */
    sprintf(sendBuffer, "Quater. : W = %05d\tX = %05d\tY = %05d\tZ = %05d \n\n\r",
    bno055_data->quaternion.w, bno055_data->quaternion.x, bno055_data->quaternion.y,
    bno055_data->quaternion.z);
    /* Transmit Gravity string */
    do{
        if(!PLIB_USART_TransmitterBufferIsFull(USART_ID_1))
        {
            PLIB_USART_TransmitterByteSend(USART_ID_1, sendBuffer[i]);
            i++;
        }
        ctnTimeout++;
    }while((sendBuffer[i-1] != '\r')&&(ctnTimeout<TIME_OUT));
    i = 0;

}

void serTransmitString ( USART_MODULE_ID usartId, const char * msg )
{
    char bufferMsg[60] = {0};
    static uint32_t i = 0;
    static uint32_t ctnTimeout = 0;

    strncpy(bufferMsg, msg, strlen(msg));

    /* Transmit string */
    do{
        if(!PLIB_USART_TransmitterBufferIsFull(usartId))
        {
            PLIB_USART_TransmitterByteSend(usartId, bufferMsg[i]);
            i++;
        }
        ctnTimeout++;
    }while((bufferMsg[i-1] != '\0')&&(ctnTimeout<TIME_OUT));
    i = 0;
}

void serTransmitbuffer ( USART_MODULE_ID usartId, char msg[], uint32_t lenght )
{
    uint32_t i = 0;
    uint32_t ctnTimeout = 0;

    /* Transmit string */
    do{
        if(!PLIB_USART_TransmitterBufferIsFull(usartId))
        {
            PLIB_USART_TransmitterByteSend(usartId, msg[i]);
            i++;
        }
        ctnTimeout++;
    }while((i < lenght)&&(ctnTimeout<TIME_OUT));
    i = 0;
}

bool pollSerialSingleCmd(USART_MODULE_ID usartID, const char * command1)
{
    static char charRead[30] = {0};
    static uint32_t readCnt = 0;

    // Get command's characters
    while((PLIB_USART_ReceiverDataIsAvailable(usartID))&&(readCnt < 30)){
        charRead[readCnt] = PLIB_USART_ReceiverByteReceive(usartID);
        readCnt++;
    }
    // Command
    if(readCnt >= 30)
    {
        /* Reset read counter */
        readCnt = 0;
        /* Clear read buffer */
        memset(charRead,0,strlen(charRead));
    }
    // Check occurence with commands
    if(strstr(charRead, command1) != NULL) {
        /* Reset read counter */
        readCnt = 0;
        /* Clear read buffer */
        memset(charRead,0,strlen(charRead));
        /* Command detected */
        return true;
    }
    else{
        return false;
    }
}

bool pollSerialCmds(USART_MODULE_ID usartID, const char * command1, const char *
command2, const char * command3,
                       const char * command4)
{
    static char charRead[CHAR_READ_BUFFER_SIZE] = {0};
    static uint32_t readCnt = 0;

    // Get command's characters
    while((PLIB_USART_ReceiverDataIsAvailable(usartID))&&(readCnt <
    CHAR_READ_BUFFER_SIZE)){
        charRead[readCnt] = PLIB_USART_ReceiverByteReceive(usartID);
        readCnt++;
    }
    // Command
    if(readCnt >= CHAR_READ_BUFFER_SIZE)
    {
        /* Reset read counter */
        readCnt = 0;
        /* Clear read buffer */
        memset(charRead,0,CHAR_READ_BUFFER_SIZE);
    }
    // Check occurence with commands
    if((strstr(charRead, command1) != NULL) || (strstr(charRead, command2) != NULL)
        || (strstr(charRead, command3) != NULL) || (strstr(charRead, command4) != NULL
        )) {
        /* Reset read counter */
        readCnt = 0;
        /* Clear read buffer */
        memset(charRead,0,CHAR_READ_BUFFER_SIZE);
        /* Command detected */
        return true;
    }
    else{
        return false;
    }
}
/* *******************************************************************************
 End of File
 */

```