```c

/*
 * File:    RN4678_driver.c
 * Author:  M.Ricchieri
 *
 * Inspired by the code of S. Giuseppe
 * Created on 12. avril 2023
 *
 * This code uses USART with FIFO
 */


//-----------------------------------------------------------------------------// Includes
#include <stdbool.h>
#include <stdint.h>
#include "RN4678_driver.h"
#include "app.h"


//-----------------------------------------------------------------------------// Constants
// Commands
#define CMD_MODE_ENTER      "$$$\r"
#define CMD_MODE_EXIT       "---\r"
#define CMD_BLE_DISCOV_EN   "Q,0\r"
// The module is able to connect, but is undiscoverable in Bluetooth Classic
#define CMD_BT_DISCOV_DIS   "Q,2\r"
#define CMD_BLE_ONLY        "SG,1\r"
#define CMD_BT_CLASSIC_ONLY "SG,2\r"
#define CMD_PREFIX_SUFIX    "SO,<,>\r"
#define CMD_REBOOT_DEVICE   "R,1\r"
#define CMD_BITMAP          "SQ,8000\r"
#define CMD_SCAN_DURATION   "SL,01\r"

// Answers
#define CMD_MODE_ANSWER      "CMD> "
#define CMD_EXIT_ANSWER      "END\r\n"
#define CMD_POS_ANSWER       "AOK\r\nCMD> "
#define CMD_NEG_ANSWER       "ERR\r\nCMD> "
//#define CMD_REBOOT_ANSWER   "<REBOOT>"
#define CMD_REBOOT_ANSWER   "Rebooting\r\n"

// Device name
#define DEVICE_NAME          "SN,TubePitotDeporte_v1.0.0\r"



//-----------------------------------------------------------------------------// init_RN4678
bool init_RN4678(void){

    bool initIsDone = 1;

    //Resets the module for a reboot
    RESET_BLEOff();
    inv_imu_sleep_ms(1000);
    RESET_BLEOn();
    inv_imu_sleep_ms(2000);

    appData.isBluetoothInCommandMode = true;
    // Enters in command mode
    initIsDone = sendCMD_RN4678(CMD_MODE_ENTER, sizeof(CMD_MODE_ENTER), CMD_MODE_ANSWER,
            sizeof(CMD_MODE_ANSWER));
    // Sets the name of the device
    initIsDone &= sendCMD_RN4678(DEVICE_NAME, sizeof(DEVICE_NAME), CMD_POS_ANSWER,
            sizeof(CMD_POS_ANSWER));
    // Sets the Bluetooth mode in Classic
    initIsDone &= sendCMD_RN4678(CMD_BT_CLASSIC_ONLY, sizeof(CMD_BT_CLASSIC_ONLY), CMD_POS_ANSWER,
            sizeof(CMD_POS_ANSWER));
    // Sets the prefix and the sufix of status
    initIsDone &= sendCMD_RN4678(CMD_PREFIX_SUFIX, sizeof(CMD_PREFIX_SUFIX), CMD_POS_ANSWER,
            sizeof(CMD_POS_ANSWER));
    // Sets the scan duration to 10 seconds
    initIsDone &= sendCMD_RN4678(CMD_SCAN_DURATION, sizeof(CMD_SCAN_DURATION), CMD_POS_ANSWER,
            sizeof(CMD_POS_ANSWER));
    // Sets the RN4678 into Fast mode
    initIsDone &= sendCMD_RN4678(CMD_BITMAP, sizeof(CMD_BITMAP), CMD_POS_ANSWER,
            sizeof(CMD_POS_ANSWER));
    // Lauches a reboot command
    initIsDone &= sendCMD_RN4678(CMD_REBOOT_DEVICE, sizeof(CMD_REBOOT_DEVICE), CMD_REBOOT_ANSWER,
            sizeof(CMD_REBOOT_ANSWER));

    inv_imu_sleep_ms(2000);

    // Flag is discoverable true
    appData.isBluetoothDiscoverable = true;

    appData.isBluetoothInCommandMode = false;

    return initIsDone;
}

//---------------------------------------------------------------------------// turnOffDiscoverBT
// For the moment, this function isn't used. The Fast mode affects the detection
// of the "$$$\r" message and it is impossible to inter in command mode when the
// data mode is enable.
bool turnOffDiscoverBT(void){

    appData.isBluetoothInCommandMode = true;
//    // Enters in command mode
    sendCMD_RN4678(CMD_MODE_ENTER, sizeof(CMD_MODE_ENTER), CMD_MODE_ANSWER,
            sizeof(CMD_MODE_ANSWER));
    // Turn off the discoverable mode of the module
    sendCMD_RN4678(CMD_BT_DISCOV_DIS, sizeof(CMD_BT_DISCOV_DIS), CMD_POS_ANSWER,
            sizeof(CMD_POS_ANSWER));
    // Exits command mode
    sendCMD_RN4678(CMD_MODE_EXIT, sizeof(CMD_MODE_EXIT), CMD_EXIT_ANSWER,
            sizeof(CMD_EXIT_ANSWER));

    appData.isBluetoothInCommandMode = false;

    return 1;
}


//---------------------------------------------------------------------------// sendCMD_RN4678
bool sendCMD_RN4678(char* pArrayToSend, size_t arraySize, char* pArrayExpected,
        size_t answerSize){

    int8_t a_answer[20];

    // Save data in TX FIFO
    putStringInFifo(&usartFifoTx, arraySize, pArrayToSend);
    // Enable USART TX interrupt
    PLIB_INT_SourceEnable(INT_ID_0, INT_SOURCE_USART_1_TRANSMIT);

    do{
        // If the number of new char in FIFO is the same as the answer size
        if(getReadSize(&usartFifoRx) >= answerSize - 1){

            // Reads the answere received
            getStringFromFifo(&usartFifoRx, &a_answer[0]);
        }

    }while((strstr((char*)a_answer, pArrayExpected) == NULL));
        //if(strstr((char*)a_answer, pArrayExpected) != NULL) isInitDone = 1;

    //}while(isInitDone != 1);

    clearInt8Array(sizeof(a_answer), &a_answer[0]);

    return 1;
}


//---------------------------------------------------------------------------// getUsartData
void getUsartData(int8_t* pArrayToModify){

    do{
        // Reads the answere received
        getStringFromFifo(&usartFifoRx, &pArrayToModify[0]);

    }while(getReadSize(&usartFifoRx));
}


//----------------------------------------------------------------------------// sendData_RN4678
void sendData_RN4678(int8_t* pArrayToSend){

    int cursor = 0;

    // Does until character '\r' is sent
    do{
        // Wait for the Transmit buffer to be empty.
        if(!PLIB_USART_TransmitterBufferIsFull(USART_ID_1)){

            // Sends all data of the array
            PLIB_USART_TransmitterByteSend(USART_ID_1, pArrayToSend[cursor]);
            cursor++;
        }
    }while(pArrayToSend[cursor-1] != '\r');
}


//----------------------------------------------------------------------------// readStatus
void readStatus(char *pArrayStatus){

    int cursor = 0;

    while(PLIB_USART_ReceiverDataIsAvailable(USART_ID_1)){
        // Reads and saves the characters received in an array
        pArrayStatus[cursor] = PLIB_USART_ReceiverByteReceive(USART_ID_1);
        // Increments the cursor value
        cursor++;
    }
}


//----------------------------------------------------------------------------// clearInt8Array
void clearInt8Array(size_t arraySize, int8_t* arrayToClear){

    int i;

    for (i = 0; i < arraySize; i++){

        arrayToClear[i] = NULL;
    }
}
```