FRAM and persistent data serialization

here is library which serializes data to Non-Volatile FRAM (adafruit https://www.adafruit.com/product/1895) which uses MB85RC256V chip) we need four libraries to be installed before this library (FRAMStream.hpp) (header only library) can be used, they are

arduino Wire (Wire.h) , Adafruit_FRAM_I2C (Adafruit_FRAM_I2C.h) , ArduinoJson (ArduinoJson.h), StreamUtils (StreamUtils.h)

here is source code of library (FRAMStream.hpp)

```
//
//  FRAMStream.c
//
//
//  Created by Alok Kumar Mishra on 7/12/20.
//

#pragma once

#include "../Configuration.hpp"



#include<Adafruit_FRAM_I2C.h>
#include <Stream.h>


extern Adafruit_FRAM_I2C fram;


namespace StreamUtils {

class FRAMStream : public Stream {
 public:
     FRAMStream(size_t address, size_t size)
         : _readAddress(address), _writeAddress(address), _end(address
+ size) {}

     int available() override {
       return _end - _readAddress;
     }

     int read() override {
       if (_readAddress >= _end)
         return -1;
       return fram.read8(_readAddress++);
     }
```

```
     int peek() override {
       if (_readAddress >= _end)
         return -1;
       return fram.read8(_readAddress);
     }

    // using Print::write;

    size_t write(const uint8_t *buffer, size_t size) override {
      size_t remaining = _end - _writeAddress;
      if (size > remaining)
        size = remaining;
      for (size_t i = 0; i < size; i++) {
          fram.write8(_writeAddress++, buffer[i]);
      }
        return size;
    }
    size_t write(uint8_t data) override {
      if (_writeAddress >= _end)
        return 0;

        fram.write8(_writeAddress++, data);
        return 1;

    }
     void flush() override {



     }
    size_t readBytes(char* buffer, size_t length)  {
        for ( size_t i =0; i < length; i++) {
            buffer[i] = fram.read8(_readAddress++);

        }
        return length;

    }


private:
     size_t _readAddress, _writeAddress, _end;};

} // namespace StreamUtils
```

here is arduino sketch (jason_serialize_fram_jul12a_alok3.ino))which show how to use this library

```
#include <Wire.h>
#include "Adafruit_FRAM_I2C.h"
#include <ArduinoJson.h>
#include <StreamUtils.h>
#include "StreamUtils/Streams/FRAMStream.hpp"
/* Example code for the Adafruit I2C EEPROM breakout */

/* Connect SCL    to SCL
   Connect SDA    to SDA
   Connect VDD    to 3 - 5V DC
   Connect GROUND to common ground */

Adafruit_FRAM_I2C fram     = Adafruit_FRAM_I2C();
// FRAMStream framStream(0, 512);

StaticJsonDocument<256> doc; // Document we will hold the num in
int num = 0;

void setup(void) {
  Serial.begin(115200);
  DeserializationError jsonerror;
  if (fram.begin()) {  // you can stick the new i2c addr in here, e.g. begin(0x51);
    Serial.println("Found I2C Fram");
  } else {
    Serial.println("I2C Fram not identified ... check your connections?\r\n");

  }
 FRAMStream framStream(0, 512);
 jsonerror = deserializeJson(doc, framStream);

 if ( jsonerror == DeserializationError::Ok)
     Serial.println("deserialization success");
 else if ( jsonerror == DeserializationError::IncompleteInput)
     Serial.println("IncompleteInput error ");
 else if  ( jsonerror == DeserializationError::InvalidInput)
     Serial.println("InvalidInput error ");
 else if  ( jsonerror == DeserializationError::NoMemory)
     Serial.println("NoMemory error ");
 else if  ( jsonerror == DeserializationError::NotSupported)
     Serial.println("NotSupported error");
 else if  ( jsonerror == DeserializationError::TooDeep)
     Serial.println("TooDeep  error");
 else
     Serial.println("unknown deserialization error");


 if(doc["num"]){
    Serial.println("Loaded doc's num value.");
    num = doc["num"];
    Serial.print("num in setup = ");
    Serial.println(num);
```

```
  } else {
    Serial.println("No 'num' variable in fram.");
  }
 if (doc["string"]) {

   Serial.println("Loaded doc's string value");
   Serial.print("string is = " );
   // Serial.println(String(doc["string"]));
   serializeJson(doc["string"], Serial);

 }
 else
     Serial.println("No string  variable in fram.");

}


void loop() {
  // put your main code here, to run repeatedly:
  int bytes_written = 0;
  if ( Serial.available() ) {
      FRAMStream framStream(0, 512);
      fram.begin();
      char command = Serial.read();
      if ( command == 'w') {
          num++;
          doc["num"] = num;
          doc["string"] = "Thus is test";
          Serial.print("\nserializing num = " );
          Serial.println(num);
          bytes_written = serializeJson(doc,framStream);
          Serial.print("\nbytes_written = " );
          Serial.println(bytes_written);

      }
      else if ( command == 'r') {
          deserializeJson(doc, framStream);
          if(doc["num"])  {
              Serial.println("Loaded doc's num value.");
              num = doc["num"];
              Serial.print("num = " );
              Serial.println( num);
          } else {
              Serial.println("No 'num' variable in fram.");
          }

      }
      else if ( command == 'd' ) {
        uint8_t val;
        for (uint16_t addr = 0; addr < 256; addr++) {
          val = fram.read8(addr);
          if ((addr % 32) == 0) {
            Serial.print("\n 0x"); Serial.print(addr, HEX); Serial.print(": ");
          }
```

```
        Serial.print("0x");
        if (val < 0x10)
          Serial.print('0');
        Serial.print(val, HEX); Serial.print(" ");
      }

    }
    else if ( command == 'c' ) {
      // clear first 256 bytes and write 0xFF into it
      uint8_t val = 0xFF;
      for (uint16_t addr = 0; addr < 256; addr++) {
        fram.write8(addr,val);
      }

    }

  }


}
```