```c
/*
 * fram_func.c
 *
 *  Created on: 6 juil. 2023
 *      Author: christop.grobety
 */
#include "fram_func.h"
#include "fatfs.h"
SPI_HandleTypeDef hspi_2;

FRESULT res; /* FatFs function common result code */
uint32_t byteswritten, bytesread; /* File write/read counts */
uint8_t  wtext[20] = "DATA FRAM IS : \n"; /* File write buffer */
uint8_t  rtext[_MAX_SS];/* File read buffer */
uint8_t  TX_Data_ADC[2];
uint8_t  TX_Data_PRETRIG = 80;
uint8_t  TX_Data_FRAM_REG = 6;
uint8_t  TX_Read_FRAM_REG = 5;
uint8_t  TX_ReadId_FRAM_REG = 0x9F;
uint8_t  TX_Data_FRAM_RESET = 4;
uint32_t TX_Data_FRAM_ADD = 0x40;
uint8_t  TX_FRAM_TAB[50] ;
uint8_t  TX_FRAM_TEST_R[4] ;
uint8_t  RX_SAMPLE[2];
uint8_t  TX_SAMPLE;
uint8_t  RX_ADD [3];
uint8_t  RX_FRAM_READ[6] ;
uint8_t  TX_FRAM_TEST_R[4] ;
uint16_t device_0,device_1;
TCHAR* fileName_0 = "REG_0.txt";
TCHAR* fileName_1 = "REG_1.txt";
TCHAR* fileName_2 = "REG_2.txt";
TCHAR* fileName_3 = "REG_3.txt";
TCHAR* reg_name;

void FRAM_write_one(enum State state_, uint8_t*TX_FRAM, uint8_t data_size,
SPI_HandleTypeDef hspi_1){
    PIN_reset();
    switch(state_){
        case FRAM_0:
            HAL_GPIO_WritePin(SELECTOR_M2_GPIO_Port, SELECTOR_M2_Pin, GPIO_PIN_SET);
            HAL_SPI_Transmit(&hspi_1,TX_FRAM,data_size, 100);
            HAL_GPIO_WritePin(SELECTOR_M2_GPIO_Port, SELECTOR_M2_Pin, GPIO_PIN_RESET);
            HAL_Delay(100);
            break;
        case FRAM_1:
            HAL_GPIO_WritePin(SELECTOR_M0_GPIO_Port, SELECTOR_M0_Pin, GPIO_PIN_SET);
            HAL_GPIO_WritePin(SELECTOR_M2_GPIO_Port, SELECTOR_M2_Pin, GPIO_PIN_SET);
            HAL_SPI_Transmit(&hspi_1,TX_FRAM,data_size, 100);
            HAL_GPIO_WritePin(SELECTOR_M2_GPIO_Port, SELECTOR_M2_Pin, GPIO_PIN_RESET);
            HAL_GPIO_WritePin(SELECTOR_M0_GPIO_Port, SELECTOR_M0_Pin, GPIO_PIN_RESET);
            HAL_Delay(100);
            break;
        case FRAM_2:
            HAL_GPIO_WritePin(SELECTOR_M1_GPIO_Port, SELECTOR_M1_Pin, GPIO_PIN_SET);
            HAL_GPIO_WritePin(SELECTOR_M2_GPIO_Port, SELECTOR_M2_Pin, GPIO_PIN_SET);
            HAL_SPI_Transmit(&hspi_1,TX_FRAM,data_size, 100);
            HAL_GPIO_WritePin(SELECTOR_M2_GPIO_Port, SELECTOR_M2_Pin, GPIO_PIN_RESET);
            HAL_GPIO_WritePin(SELECTOR_M1_GPIO_Port, SELECTOR_M1_Pin, GPIO_PIN_RESET);
            HAL_Delay(100);
            break;
        case FRAM_3:
            HAL_GPIO_WritePin(SELECTOR_M0_GPIO_Port, SELECTOR_M0_Pin, GPIO_PIN_SET);
            HAL_GPIO_WritePin(SELECTOR_M1_GPIO_Port, SELECTOR_M1_Pin, GPIO_PIN_SET);
            HAL_GPIO_WritePin(SELECTOR_M2_GPIO_Port, SELECTOR_M2_Pin, GPIO_PIN_SET);
            HAL_SPI_Transmit(&hspi_1,TX_FRAM,data_size, 100);
            HAL_GPIO_WritePin(SELECTOR_M2_GPIO_Port, SELECTOR_M2_Pin, GPIO_PIN_RESET);
            HAL_GPIO_WritePin(SELECTOR_M1_GPIO_Port, SELECTOR_M1_Pin, GPIO_PIN_RESET);
            HAL_GPIO_WritePin(SELECTOR_M0_GPIO_Port, SELECTOR_M0_Pin, GPIO_PIN_RESET);
            HAL_Delay(100);
            break;
    }
}
```

```
73  void FRAM_write_all(SPI_HandleTypeDef hspi_1, uint8_t*TX_FRAM, uint8_t data_size){
74      PIN_reset();
75
76      HAL_GPIO_WritePin(SELECTOR_M3_GPIO_Port, SELECTOR_M3_Pin, GPIO_PIN_SET);
77      HAL_SPI_Transmit(&hspi_1, TX_FRAM, data_size, 100);
78      HAL_GPIO_WritePin(SELECTOR_M3_GPIO_Port, SELECTOR_M3_Pin, GPIO_PIN_RESET);
79      HAL_Delay(100);
80
81  }
82  void FRAM_write_ADC_to_FRAM(SPI_HandleTypeDef hspi_1){
83      PIN_reset();
84
85      TX_FRAM_TAB[0] = 2;
86      for(int i =1; i<5;i++){
87          TX_FRAM_TAB[i] = 0;
88      }
89      TX_FRAM_TAB[4] = 0; // "A"
90
91      HAL_GPIO_WritePin(SELECTOR_M3_GPIO_Port, SELECTOR_M3_Pin, GPIO_PIN_SET);
92      HAL_SPI_Transmit(&hspi_1, (uint8_t*)TX_FRAM_TAB,4, 100);
93      //-------------------------------------------------ADC TO
        FRAM-----------------------------------------
94      HAL_GPIO_WritePin(SELECTOR_M1_GPIO_Port, SELECTOR_M1_Pin, GPIO_PIN_SET);
95      //HAL_GPIO_WritePin(SELECTOR_M3_GPIO_Port, SELECTOR_M3_Pin, GPIO_PIN_RESET);
96      //HAL_GPIO_WritePin(SELECTOR_M1_GPIO_Port, SELECTOR_M1_Pin, GPIO_PIN_RESET);
97
98  }
99  void FRAM_write_reg(SPI_HandleTypeDef hspi_1){
100       HAL_GPIO_WritePin(SELECTOR_M3_GPIO_Port, SELECTOR_M3_Pin, GPIO_PIN_SET);
101       HAL_SPI_Transmit(&hspi_1, (uint8_t*)&TX_Data_FRAM_REG,1, 100);
102       HAL_GPIO_WritePin(SELECTOR_M3_GPIO_Port, SELECTOR_M3_Pin, GPIO_PIN_RESET);
103       HAL_Delay(100);
104 }
105 void FRAM_reset_reg(SPI_HandleTypeDef hspi_1){
106
107     HAL_GPIO_WritePin(SELECTOR_M3_GPIO_Port, SELECTOR_M3_Pin, GPIO_PIN_SET);
108     HAL_SPI_Transmit(&hspi_1, (uint8_t*)&TX_Data_FRAM_RESET,1, 100);
109     HAL_GPIO_WritePin(SELECTOR_M3_GPIO_Port, SELECTOR_M3_Pin, GPIO_PIN_RESET);
110     HAL_Delay(100);
111 }
112 void FRAM_read_reg(enum State state_, SPI_HandleTypeDef hspi_1){
113     PIN_reset();
114     switch(state_){
115         case FRAM_0:
116             HAL_GPIO_WritePin(SELECTOR_M2_GPIO_Port, SELECTOR_M2_Pin, GPIO_PIN_SET);
117             HAL_SPI_Transmit(&hspi_1, (uint8_t*)&TX_Read_FRAM_REG,1, 1);
118             HAL_SPI_Receive(&hspi_1, (uint8_t*)&TX_SAMPLE,1, 100);
119             HAL_GPIO_WritePin(SELECTOR_M2_GPIO_Port, SELECTOR_M2_Pin, GPIO_PIN_RESET);
120             //HAL_Delay(100);
121             reg_name = fileName_0;
122             break;
123         case FRAM_1:
124             HAL_GPIO_WritePin(SELECTOR_M0_GPIO_Port, SELECTOR_M0_Pin, GPIO_PIN_SET);
125             HAL_GPIO_WritePin(SELECTOR_M2_GPIO_Port, SELECTOR_M2_Pin, GPIO_PIN_SET);
126             HAL_SPI_Transmit(&hspi_1, (uint8_t*)&TX_Read_FRAM_REG,1, 1);
127             HAL_SPI_Receive(&hspi_1, (uint8_t*)&TX_SAMPLE,1, 100);
128             HAL_GPIO_WritePin(SELECTOR_M2_GPIO_Port, SELECTOR_M2_Pin, GPIO_PIN_RESET);
129             HAL_GPIO_WritePin(SELECTOR_M0_GPIO_Port, SELECTOR_M0_Pin, GPIO_PIN_RESET);
130             //HAL_Delay(100);
131             reg_name = fileName_1;
132             break;
133         case FRAM_2:
134             HAL_GPIO_WritePin(SELECTOR_M1_GPIO_Port, SELECTOR_M1_Pin, GPIO_PIN_SET);
135             HAL_GPIO_WritePin(SELECTOR_M2_GPIO_Port, SELECTOR_M2_Pin, GPIO_PIN_SET);
136             HAL_SPI_Transmit(&hspi_1, (uint8_t*)&TX_Read_FRAM_REG,1, 1);
137             HAL_SPI_Receive(&hspi_1, (uint8_t*)&TX_SAMPLE,1, 100);
138             HAL_GPIO_WritePin(SELECTOR_M2_GPIO_Port, SELECTOR_M2_Pin, GPIO_PIN_RESET);
139             HAL_GPIO_WritePin(SELECTOR_M1_GPIO_Port, SELECTOR_M1_Pin, GPIO_PIN_RESET);
140             //HAL_Delay(100);
141             reg_name = fileName_2;
142             break;
143         case FRAM_3:
144             HAL_GPIO_WritePin(SELECTOR_M0_GPIO_Port, SELECTOR_M0_Pin, GPIO_PIN_SET);
```

```
145                 HAL_GPIO_WritePin(SELECTOR_M1_GPIO_Port, SELECTOR_M1_Pin, GPIO_PIN_SET);
146                 HAL_GPIO_WritePin(SELECTOR_M2_GPIO_Port, SELECTOR_M2_Pin, GPIO_PIN_SET);
147                 HAL_SPI_Transmit(&hspi_1, (uint8_t*)&TX_Read_FRAM_REG,1, 1);
148                 HAL_SPI_Receive(&hspi_1, (uint8_t*)&TX_SAMPLE,1, 100);
149                 HAL_GPIO_WritePin(SELECTOR_M2_GPIO_Port, SELECTOR_M2_Pin, GPIO_PIN_RESET);
150                 HAL_GPIO_WritePin(SELECTOR_M1_GPIO_Port, SELECTOR_M1_Pin, GPIO_PIN_RESET);
151                 HAL_GPIO_WritePin(SELECTOR_M0_GPIO_Port, SELECTOR_M0_Pin, GPIO_PIN_RESET);
152                 //HAL_Delay(100);
153                 reg_name = fileName_3;
154                 break;
155         }
156     }
157     void FRAM_device(SPI_HandleTypeDef hspi_1){
158         PIN_reset();
159         HAL_GPIO_WritePin(SELECTOR_M2_GPIO_Port, SELECTOR_M2_Pin, GPIO_PIN_SET);
160         HAL_SPI_Transmit(&hspi_1, (uint8_t*)&TX_ReadId_FRAM_REG,1, 100);
161         HAL_SPI_Receive(&hspi_1, (uint8_t*)&TX_FRAM_TEST_R,4, 100);
162         HAL_GPIO_WritePin(SELECTOR_M2_GPIO_Port, SELECTOR_M2_Pin, GPIO_PIN_RESET);
163         HAL_Delay(100);
164         reg_name = "DEVICE.txt";
165 
166         SD_create_file(SDFile, reg_name);
167         SD_write_data(SDFile, reg_name, TX_FRAM_TEST_R[0]);
168         SD_write_data(SDFile, reg_name, TX_FRAM_TEST_R[1]);
169         SD_write_data(SDFile, reg_name, TX_FRAM_TEST_R[2]);
170         SD_write_data(SDFile, reg_name, TX_FRAM_TEST_R[3]);
171     }
172     uint8_t* FRAM_read(enum State state_, uint32_t add, SPI_HandleTypeDef hspi_1, uint8_t
    data_size){
173         PIN_reset();
174         RX_FRAM_READ[3] = (uint8_t)(add&255);
175         RX_FRAM_READ[2] = (uint8_t)((add>>8)&255);
176         RX_FRAM_READ[1] = (uint8_t)((add>>16)&7);
177         RX_FRAM_READ[0] = 3;
178         switch(state_){
179             case FRAM_0:
180                 HAL_GPIO_WritePin(SELECTOR_M2_GPIO_Port, SELECTOR_M2_Pin, GPIO_PIN_SET);
181                 HAL_SPI_Transmit(&hspi_1, (uint8_t*)&RX_FRAM_READ,4, 100);
182                 HAL_SPI_Receive(&hspi_1, (uint8_t*)&RX_SAMPLE,data_size, 100);
183                 HAL_GPIO_WritePin(SELECTOR_M2_GPIO_Port, SELECTOR_M2_Pin, GPIO_PIN_RESET);
184                 //HAL_Delay(100);
185                 break;
186             case FRAM_1:
187                 HAL_GPIO_WritePin(SELECTOR_M0_GPIO_Port, SELECTOR_M0_Pin, GPIO_PIN_SET);
188                 HAL_GPIO_WritePin(SELECTOR_M2_GPIO_Port, SELECTOR_M2_Pin, GPIO_PIN_SET);
189                 HAL_SPI_Transmit(&hspi_1, (uint8_t*)&RX_FRAM_READ,4, 100);
190                 HAL_SPI_Receive(&hspi_1, (uint8_t*)&RX_SAMPLE,data_size, 100);
191                 HAL_GPIO_WritePin(SELECTOR_M2_GPIO_Port, SELECTOR_M2_Pin, GPIO_PIN_RESET);
192                 HAL_GPIO_WritePin(SELECTOR_M0_GPIO_Port, SELECTOR_M0_Pin, GPIO_PIN_RESET);
193                 //HAL_Delay(100);
194                 break;
195             case FRAM_2:
196                 HAL_GPIO_WritePin(SELECTOR_M1_GPIO_Port, SELECTOR_M1_Pin, GPIO_PIN_SET);
197                 HAL_GPIO_WritePin(SELECTOR_M2_GPIO_Port, SELECTOR_M2_Pin, GPIO_PIN_SET);
198                 HAL_SPI_Transmit(&hspi_1, (uint8_t*)&RX_FRAM_READ,4, 100);
199                 HAL_SPI_Receive(&hspi_1, (uint8_t*)&RX_SAMPLE,data_size, 100);
200                 HAL_GPIO_WritePin(SELECTOR_M2_GPIO_Port, SELECTOR_M2_Pin, GPIO_PIN_RESET);
201                 HAL_GPIO_WritePin(SELECTOR_M1_GPIO_Port, SELECTOR_M1_Pin, GPIO_PIN_RESET);
202                 //HAL_Delay(100);
203                 break;
204             case FRAM_3:
205                 HAL_GPIO_WritePin(SELECTOR_M0_GPIO_Port, SELECTOR_M0_Pin, GPIO_PIN_SET);
206                 HAL_GPIO_WritePin(SELECTOR_M1_GPIO_Port, SELECTOR_M1_Pin, GPIO_PIN_SET);
207                 HAL_GPIO_WritePin(SELECTOR_M2_GPIO_Port, SELECTOR_M2_Pin, GPIO_PIN_SET);
208                 HAL_SPI_Transmit(&hspi_1, (uint8_t*)&RX_FRAM_READ,4, 100);
209                 HAL_SPI_Receive(&hspi_1, (uint8_t*)&RX_SAMPLE,data_size, 100);
210                 HAL_GPIO_WritePin(SELECTOR_M2_GPIO_Port, SELECTOR_M2_Pin, GPIO_PIN_RESET);
211                 HAL_GPIO_WritePin(SELECTOR_M1_GPIO_Port, SELECTOR_M1_Pin, GPIO_PIN_RESET);
212                 HAL_GPIO_WritePin(SELECTOR_M0_GPIO_Port, SELECTOR_M0_Pin, GPIO_PIN_RESET);
213                 //HAL_Delay(100);
214                 break;
215         }
216         return (uint8_t*)&RX_SAMPLE;
```

```
}
void setPreTrigg(SPI_HandleTypeDef hspi_1, uint8_t pretrig){
    PIN_reset();

    HAL_GPIO_WritePin(FPGA_PRETRIG_GPIO_Port, FPGA_PRETRIG_Pin, GPIO_PIN_SET);
    HAL_SPI_Transmit(&hspi_1, &pretrig, 1, 100);
    HAL_GPIO_WritePin(FPGA_PRETRIG_GPIO_Port, FPGA_PRETRIG_Pin, GPIO_PIN_RESET);

}

void setTriggChannel(enum trig_channel chan){
    switch(chan){
        case CHANNEL_0:
            HAL_GPIO_WritePin(TRIG_SRC0_GPIO_Port, TRIG_SRC0_Pin, GPIO_PIN_RESET);
            HAL_GPIO_WritePin(TRIG_SRC1_GPIO_Port, TRIG_SRC1_Pin, GPIO_PIN_RESET);
            break;
        case CHANNEL_1:
            HAL_GPIO_WritePin(TRIG_SRC0_GPIO_Port, TRIG_SRC0_Pin, GPIO_PIN_RESET);
            HAL_GPIO_WritePin(TRIG_SRC1_GPIO_Port, TRIG_SRC1_Pin, GPIO_PIN_SET);
            break;
        case CHANNEL_2:
            HAL_GPIO_WritePin(TRIG_SRC0_GPIO_Port, TRIG_SRC0_Pin, GPIO_PIN_RESET);
            HAL_GPIO_WritePin(TRIG_SRC1_GPIO_Port, TRIG_SRC1_Pin, GPIO_PIN_SET);
            break;
        case CHANNEL_3:
            HAL_GPIO_WritePin(TRIG_SRC0_GPIO_Port, TRIG_SRC0_Pin, GPIO_PIN_SET);
            HAL_GPIO_WritePin(TRIG_SRC1_GPIO_Port, TRIG_SRC1_Pin, GPIO_PIN_SET);
            break;
    }
}


void PIN_reset(){
    HAL_GPIO_WritePin(SELECTOR_M3_GPIO_Port, SELECTOR_M3_Pin, GPIO_PIN_RESET);
    HAL_GPIO_WritePin(SELECTOR_M2_GPIO_Port, SELECTOR_M2_Pin, GPIO_PIN_RESET);
    HAL_GPIO_WritePin(SELECTOR_M1_GPIO_Port, SELECTOR_M1_Pin, GPIO_PIN_RESET);
    HAL_GPIO_WritePin(SELECTOR_M0_GPIO_Port, SELECTOR_M0_Pin, GPIO_PIN_RESET);
}

void LED_on(enum color_Led color){
    switch(color){
        case RED:
            HAL_GPIO_WritePin(UI_LED_R_GPIO_Port, UI_LED_R_Pin, SET);
            HAL_GPIO_WritePin(UI_LED_B_GPIO_Port, UI_LED_B_Pin, RESET);
            HAL_GPIO_WritePin(UI_LED_G_GPIO_Port, UI_LED_G_Pin, RESET);
            break;
        case BLUE:
            HAL_GPIO_WritePin(UI_LED_R_GPIO_Port, UI_LED_R_Pin, RESET);
            HAL_GPIO_WritePin(UI_LED_B_GPIO_Port, UI_LED_B_Pin, SET);
            HAL_GPIO_WritePin(UI_LED_G_GPIO_Port, UI_LED_G_Pin, RESET);
            break;
        case GREEN:
            HAL_GPIO_WritePin(UI_LED_R_GPIO_Port, UI_LED_R_Pin, RESET);
            HAL_GPIO_WritePin(UI_LED_B_GPIO_Port, UI_LED_B_Pin, RESET);
            HAL_GPIO_WritePin(UI_LED_G_GPIO_Port, UI_LED_G_Pin, SET);
            break;
        case OFF:
            LED_off();
            break;
    }
    LED_STATE = color;
}

void LED_off(){
    HAL_GPIO_WritePin(UI_LED_R_GPIO_Port, UI_LED_R_Pin, RESET);
    HAL_GPIO_WritePin(UI_LED_B_GPIO_Port, UI_LED_B_Pin, RESET);
    HAL_GPIO_WritePin(UI_LED_G_GPIO_Port, UI_LED_G_Pin, RESET);
    LED_STATE = OFF;
}
```