```c
/*!
 * @file        RF_usart.h
 *
 * @brief       This file contains all the functions for the charger PWM
 *
 * @version     V1.0
 *
 * @date        2023-12-24
 */

#include "RF_usart.h"
#include "main.h"

extern Flag_Check          Flag;
extern uint32_t Addr;
extern uint16_t Curr_Def;
uint16_t Prt_CurrMax = 2300;
float Pload = 0;

void Set_Parameter(uint8_t num_rf){
    if(USART_ReadStatusFlag(USART1, USART_FLAG_TXBE) != RESET){
        if(num_rf == 73 || Flag.Set_Curr == 1){       // I -> set current
            Flag.Set_Curr = Flag_ON;
            Flag.Set_AC_SW = Flag_OFF;
            Flag.Set_Dis_BAT = Flag_OFF;
            Set_Current(num_rf);
            Flash_Write();
        }
        else if(num_rf == 83 || Flag.Set_AC_SW == 1){     // S -> Switch AC-DC
            Flag.Set_Curr = Flag_OFF;
            Flag.Set_AC_SW = Flag_ON;
            Flag.Set_Dis_BAT = Flag_OFF;

            Set_ACDC(num_rf);
            Flash_Write();
        }
        else if(num_rf == 68 || Flag.Set_Dis_BAT == 1){        // D -> Cut-Off Battery
            Flag.Set_Curr = Flag_OFF;
            Flag.Set_AC_SW = Flag_OFF;
            Flag.Set_Dis_BAT = Flag_ON;

            Set_DisBattery(num_rf);
            Flash_Write();
        }
        else if(num_rf == 80){      // P -> check parameter
            Check_Para();
        }
        else if(num_rf == 82){      // R -> Reset parameter to default
            Rst_Default();
        }
```

```
51          else if(num_rf == 81){       // Q-> Read data for Application
52              Read_Para();
53          }
54          else {
55              Flag.Set_Curr = Flag_OFF;
56              Flag.Set_AC_SW = Flag_OFF;
57              Flag.Set_Dis_BAT = Flag_OFF;
58              printf("\n\rError! Import: I, S, D, P");
59          }
60      }
61  }
62  void Set_Current(uint8_t Curr){
63      if(Curr == 73){
64          printf("\n\rSet Current: import '+' or '-'");
65          printf("\n\rCurrent: %.2fA", Val_CovCurr);      // Current <=> quy đổi ra
    điện áp sau cầu phân áp * 3 lần trừ đi hệ số tính toán
66      }
67      else if(Curr == 0x2d){
68          I_Load -= 50;
69          Curr_Def -= 50;
70          if(I_Load <= 0){I_Load = 0; Curr_Def = 0;}
71          printf("\n\rCurrent: %.2fA", Val_CovCurr);
72          printf("@D%.2f#",   Val_CovCurr);
73
74      }
75      else if(Curr == 0x2b){
76          I_Load += 50;
77          Curr_Def += 50;
78          if(I_Load >= Prt_CurrMax){I_Load = Prt_CurrMax; Curr_Def = Prt_CurrMax;}
79          printf("\n\rCurrent: %.2fA", Val_CovCurr);
80          printf("@D%.2f#",   Val_CovCurr);
81      }
82      else if(Curr == 66){
83          Flag.Set_Curr = Flag_OFF;
84          printf("\n\rReset OK!");
85      }
86      else{
87          printf("\n\rFail; import 'B' -> Reset");
88      }
89  }
90  void Set_ACDC(uint8_t Sw){
91      if(Sw == 83){
92          printf("\n\rSet DC to AC Voltage: import '+' or '-'");
93          printf("\n\rDC to AC Voltage: %.2fV", AC_SW_Volt/Val_CovBat_DC);
94      }
95      else if(Sw == 0x2d){
96          AC_SW_Volt -= 10;
97          printf("\n\rDC to AC Voltage: %.2fV", (AC_SW_Volt/Val_CovBat_DC));
98          Flag.Cov_ACDC = Flag_OFF;      // Reset Recovery flag
99      }
```

```
100     else if(Sw == 0x2b){
101         AC_SW_Volt += 10;
102         printf("\n\rDC to AC Voltage: %.2fV", AC_SW_Volt/Val_CovBat_DC);
103         Flag.Cov_ACDC = Flag_OFF;       // Reset Recovery flag
104     }
105     else if(Sw == 66){
106         Flag.Set_AC_SW = Flag_OFF;
107         printf("\n\rReset OK!");
108     }
109     else{
110         printf("\n\rFail; import 'B' -> Reset");
111     }
112 }
113 void Set_DisBattery(uint8_t Dis_Bat){
114     if(Dis_Bat == 68){
115         printf("\n\rSet Battery Cut-Off voltage: import '+' or '-'");
116         printf("\n\rBattery Cut-Off voltage: %.2fV", (Dis_BAT/Val_CovBat_DC));
117 
118     }
119     else if(Dis_Bat == 0x2d){
120         Dis_BAT -= 10;
121         printf("\n\rBattery Cut-Off voltage: %.2fV", (Dis_BAT/Val_CovBat_DC));
122     }
123     else if(Dis_Bat == 0x2b){
124         Dis_BAT += 10;
125         printf("\n\rBattery Cut-Off voltage: %.2fV", Dis_BAT/Val_CovBat_DC);
126     }
127     else if(Dis_Bat == 66){
128         Flag.Set_Dis_BAT = Flag_OFF;
129         printf("\n\rReset OK!");
130     }
131     else{
132         printf("\n\rFail; import 'B' -> Reset");
133     }
134 }
135 void Check_Para(void){
136     Pload = (Val_CovCurr)*(Re_Adc_LED/Val_CovLoad);
137     printf("\n\rParameter:");
138     printf("\n\rBattery Cut-Off voltage: %.2fV", (Dis_BAT/Val_CovBat_DC));
139     printf("\n\rDC to AC Voltage: %.2fV", AC_SW_Volt/Val_CovBat_DC);
140     printf("\n\rCurrent: %.2fA", Val_CovCurr);
141     printf("\n\rRecovery_Volt: %.2fV", Recovery_Volt/Val_CovBat_DC);
142     printf("\n\rTemp: %.2foC", Re_TempNTC());
143     printf("\n\rVoltage PV: %.2fV", Re_Adc_PV/1.0);
144     printf("\n\rVoltage LED: %.2fV", Re_Adc_LED/Val_CovLoad);
145     printf("\n\rVoltage ACDC: %.2fV", Re_Adc_DC/Val_CovBat_DC);
146     printf("\n\rVoltage Battery: %.2fV", Re_Adc_BAT/Val_CovBat_DC);
147     printf("\n\rCong suat LOAD: %.2fV", Pload);
148 }
149 void Read_Para(void){
```

```
150     Pload = (Val_CovCurr)*(Re_Adc_LED/Val_CovLoad);
151     printf("@B%.2f#\n", (Dis_BAT/Val_CovBat_DC));
152     printf("@C%.2f#\n",  AC_SW_Volt/Val_CovBat_DC);
153     printf("@D%.2f#\n",  Val_CovCurr);
154     printf("@E%.2f#\n", Recovery_Volt/Val_CovBat_DC);
155     printf("@U%.2f#\n", Re_TempNTC());
156     printf("@I%.2f#\n", Re_Adc_PV/1.0);
157     printf("@O%.2f#\n", Re_Adc_LED/Val_CovLoad);
158     printf("@T%.2f#\n", Re_Adc_DC/Val_CovBat_DC);
159     printf("@X%.2f#\n", Re_Adc_BAT/Val_CovBat_DC);
160     printf("@Y%.2f#\n", Pload);
161     Delay(0x2fff);
162 }
163 void Rst_Default(void){
164     switch(Check_Bat12_24()){
165         case 0:
166             FMC_Unlock();
167             FMC_ErasePage(Addr);
168             FMC_ProgramWord(Flash_Float, 3040/2);     // default 27V (2700 = 24V)
169             FMC_ProgramWord(Flash_Bulk, 3200/2);      // default 28.5V = 3200 (2810
    = 25V)
170             FMC_ProgramWord(Flash_DisPv, 555);       // PV = ~9V
171             FMC_ProgramWord(Flash_EnPv, 620);        // PV  = ~10V
172 
173             FMC_ProgramWord(Flash_ReVol, 2810/2);     // BAT = 25V (Recovery Voltage)
174             FMC_ProgramWord(Flash_ACDC, 2585/2);      // BAT = 23V (Set point
    Swiching DC-AC)
175             FMC_ProgramWord(Flash_DisBat, 2475/2);    // BAT < 22V ( Disable PIN)
176             FMC_ProgramWord(Flash_Curr, 1310);       // Current Load I_load =
    2A(1700) - 24V/5A/120W-Load/3.24A/35.3V(2180)
177             Flicker = 3000;
178             FMC_Lock();
179             break;
180         case 1:
181             FMC_Unlock();
182             FMC_ErasePage(Addr);
183             FMC_ProgramWord(Flash_Float, 3040);     // default 27V (2700 = 24V)
184             FMC_ProgramWord(Flash_Bulk, 3200);      // default 28.5V = 3200 (2810 =
    25V)
185             FMC_ProgramWord(Flash_DisPv, 555);       // PV = ~9V
186             FMC_ProgramWord(Flash_EnPv, 620);        // PV  = ~10V
187 
188             FMC_ProgramWord(Flash_ReVol, 2810);     // BAT = 25V (Recovery Voltage)
189             FMC_ProgramWord(Flash_ACDC, 2585);      // BAT = 23V (Set point Swiching
    DC-AC)
190             FMC_ProgramWord(Flash_DisBat, 2475);    // BAT < 22V ( Disable PIN)
191             FMC_ProgramWord(Flash_Curr, 1700);       // Current Load I_load =
    2A(1700) - 24V/5A/120W-Load/3.24A/35.3V(2180)
192             Flicker = 5000;
193             FMC_Lock();
```

```
194             break;
195         default:
196             break;
197     }
198     Point_Float    = *(volatile uint32_t*)Addr;
199     Point_Bulk     = *(volatile uint32_t*)(Addr+4);
200     SetPoin_DisPV  = *(volatile uint32_t*)(Addr+8);
201     Point_En_PV    = *(volatile uint32_t*)(Addr+12);
202 
203     Recovery_Volt  = *(volatile uint32_t*)(Addr+16);
204     AC_SW_Volt     = *(volatile uint32_t*)(Addr+20);
205     Dis_BAT        = *(volatile uint32_t*)(Addr+24);
206     I_Load         = *(volatile uint32_t*)(Addr+28);
207     printf("\n\rReset parameter to default: OK!");
208 }
209 void Flash_Write(void){
210     FMC_Unlock();
211     FMC_ErasePage(Addr);
212     FMC_ProgramWord(Flash_Float, Point_Float);
213     FMC_ProgramWord(Flash_Bulk, Point_Bulk);
214     FMC_ProgramWord(Flash_DisPv, SetPoin_DisPV);
215     FMC_ProgramWord(Flash_EnPv, Point_En_PV);
216     FMC_ProgramWord(Flash_ReVol, Recovery_Volt);
217     FMC_ProgramWord(Flash_ACDC, AC_SW_Volt);
218     FMC_ProgramWord(Flash_DisBat, Dis_BAT);
219     FMC_ProgramWord(Flash_Curr, I_Load);
220     FMC_Lock();
221 }
222 void USART_Write(USART_T* usart, uint8_t* dat)
223 {
224     while (*dat)
225     {
226         while (USART_ReadStatusFlag(usart, USART_FLAG_TXBE) == RESET);
227         USART_TxData(usart, *dat++);
228     }
229 }
230 #if defined (__CC_ARM) || defined (__ICCARM__) || (defined(__ARMCC_VERSION) &&
    (__ARMCC_VERSION >= 6010050))
231 
232 int fputc(int ch, FILE* f)
233 {
234     /* send a byte of data to the serial port */
235     USART_TxData(DEBUG_USART, (uint8_t)ch);
236     /* wait for the data to be send  */
237     while (USART_ReadStatusFlag(DEBUG_USART, USART_FLAG_TXBE) == RESET);
238     return (ch);
239 }
240 #elif defined (__GNUC__)
241 #endif
242 
```