# Opis kodu i problemów sterowania członami pasywnymi

Tomasz Święchowicz

26 października 2020

## 1 Globalne struktury alarmów, statusów i kontroli

Sterowanie przekształtnikiem przez użytkownika zostało skupione w 3 strukturach: **alarm**, **alarm_snapshot**, **status** i **control**. W **alarm** znajdują się flagi błędów, które powodują zatrzymanie pracy przekształtnika. **Alarm_snapshot** zawiera w sobie błędy, które jako pierwsze spowodowały zatrzymanie przekształtnika. Struktura **status** przechowuje flagi informujące o stanie przekształtnika i jego elementów peryferyjnych, ale nie powodujące zatrzymania pracy. Struktura **control** zawiera natomiast parametry na które użytkownik ma wpływ - może to być załączenie danego typu kompensacji lub zainicjowanie procedury zapisu danych na kartę SD.

### 1.1 Struktura alarm

Znaczenie poszczególnych flag:

1. **I_conv** - pierwsze 16 bitów flag informuje o przekroczeniu prądu przekształtnika. Litera H informuje o przekroczeniu górnego poziomu, natomiast L dolnego poziomu. Dopisek SD oznacza błąd z komparatora badającego prąd za okno $2\mu s$, czyli jest w stanie zareagować przekroczenie prądu krótsze niż okres sterowania wynoszący $32\mu s$. Brak dopisku oznacza błąd przekroczenia średniej wartości prądu w poprzednim okresie.
2. **Temperature** - Limity temperatury są porównywane względem największej wartości z trzech dostępnych termistorów - daje to możliwość podpięcia dowolnej liczby termistorów. Całkowity brak termistorów daje domyślną wartość -27°$C$, przez zostanie wywołany błąd przekroczenia dolnego poziomu temperatury.
3. **U_dc** - Dolny poziom sprawdzenia przekroczenia napięcia DC zmienia swój stan w trakcie pracy - domyślnie jest on ustawiony na wartość -5$V$, a w trakcie pracy przekształtnika wynosi 610V.
4. **Driver** - dotyczą błędów przychodzących z poszczególnych sterowników bramkowych. Litera A oznacza górny, a B dolny tranzystor gałęzi.
5. **No_calibration** - brak danych kalibracyjnych na karcie SD jak i w pamięci FLASH. Uruchomienie przekształtnika spowoduje przejście w tryb pół-automatycznej kalibracji.
6. **CT_char_error** - brak charakterystyki przekładnika na karcie SD jak i w pamięci FLASH lub niepoprawnie wypełnione jej dane. Flaga jest sprawdzana tylko w trybie pracy z przekładnikiem.
7. **PLL_unsync** - utracenie synchronizacji w trakcie pracy przekształtnika.
8. **CONV_SOFTSTART** - nieudana procedura startu przekształtnika.

```
union ALARM
{
    Uint32 all[2];
    struct
    {
        Uint32 I_conv_a_H:1;
        Uint32 I_conv_a_L:1;
        Uint32 I_conv_b_H:1;
        Uint32 I_conv_b_L:1;

        Uint32 I_conv_c_H:1;
        Uint32 I_conv_c_L:1;
        Uint32 I_conv_n_H:1;
        Uint32 I_conv_n_L:1;

        Uint32 I_conv_a_SDH:1;
        Uint32 I_conv_a_SDL:1;
        Uint32 I_conv_b_SDH:1;
        Uint32 I_conv_b_SDL:1;

        Uint32 I_conv_c_SDH:1;
        Uint32 I_conv_c_SDL:1;
        Uint32 I_conv_n_SDH:1;
        Uint32 I_conv_n_SDL:1;
        //16bits

        Uint32 Temperature_H:1;
        Uint32 Temperature_L:1;
        Uint32 U_dc_H:1;
        Uint32 U_dc_L:1;

        Uint32 Driver_PWM_a_A : 1;
        Uint32 Driver_PWM_a_B : 1;
        Uint32 Driver_PWM_b_A : 1;
        Uint32 Driver_PWM_b_B : 1;
        Uint32 Driver_PWM_c_A : 1;
        Uint32 Driver_PWM_c_B : 1;
        Uint32 Driver_PWM_n_A : 1;
        Uint32 Driver_PWM_n_B : 1;

        Uint32 Not_enough_data : 1;
        Uint32 CT_char_error : 1;
        Uint32 PLL_UNSYNC : 1;
        Uint32 CONV_SOFTSTART : 1;
        //32bits
        Uint32 FUSE_BROKEN : 1;
        Uint32 TZ_FLT_SUPPLY : 1;
        Uint32 TZ_DRV_FLT : 1;
        Uint32 TZ_CLOCKFAIL : 1;
        Uint32 TZ_EMUSTOP : 1;
        Uint32 TZ_SD_COMP : 1;
        Uint32 TZ : 1;

        Uint32 I_conv_rms_a:1;
        Uint32 I_conv_rms_b:1;
        Uint32 I_conv_rms_c:1;
        Uint32 I_conv_rms_n:1;

        Uint32 U_grid_rms_a_L:1;
        Uint32 U_grid_rms_b_L:1;
        Uint32 U_grid_rms_c_L:1;

        Uint32 U_grid_abs_a_H:1;
        Uint32 U_grid_abs_b_H:1;
        Uint32 U_grid_abs_c_H:1;

        Uint32 rsvd1:15;
    }bit;
};
```

Wydruk 1: Struktura **alarm**

9. **FUSE_BROKEN** - wykryty przepalony bezpiecznik.
10. **TZ_FLT_SUPPLY** - zanikające lub niepoprawne napięcie 24V.
11. **TZ_DRV_FLT** - zbiorcza flaga błędów sterowników bramkowych.
12. **TZ_CLOCKFAIL** - problemy z zegarem mikrokontrolera.
13. **TZ_EMUSTOP** - program zatrzymany przez breakpoint podczas debugowania.
14. **TZ_SD_COMP** - zbiorcza flaga przekroczeń prądu SD.
15. **TZ** - zbiorcza flaga TripZone.
16. **I_conv_rms** - błąd przekroczenia wartości RMS prądu przekształtnika ponad 120% nominalnej wartości. Prąd rms jest badany z oknem $1s$.
17. **U_grid_rms** - błąd przekroczenia wartości RMS napięcia sieci poniżej $110Vrms$. Napięcie jest badane z oknem $20ms$.
18. **U_grid_abs** - błąd przekroczenia chwilowej wartości absolutnej napięcia sieci ponad $380V$. Napięcie jest próbkowane co $32\mu s$.
19. **rsvd1** - przestrzeń na dodatkowe flagi.

## 1.2 Struktura status

Znaczenie poszczególnych flag:

1. **Init_done** - zakończona inicjalizacja kodu i modułów peryferyjnych mikrokontrolera.
2. **ONOFF** - stan włącznika monostabilnego/bistabilnego.
3. **DS1_switch_SD_CT** - stan DipSwitch decydującego o użyciu przekładników prądowych.
4. **DS2_enable_Q_comp** - stan DipSwitch decydujący o załączeniu kompensacji mocy biernej. Ustawienie aktywne, gdy DS8 jest załączony.
5. **DS3_emable_P_sym** - stan DipSwitch decydujący o załączeniu symetryzacji prądów czynnych. Ustawienie aktywne, gdy DS8 jest załączony.
6. **DS4_enable_H_comp** - stan DipSwitch decydujący o załączeniu kompensacji harmonicznych. Ustawienie aktywne, gdy DS8 jest załączony.
7. **DS5_limit_to_9odd_harmonics** - stan DipSwitch ograniczający kompensowane harmoniczne do 19. Ustawienie aktywne, gdy DS8 jest załączony.
8. **DS6_limit_to_14odd_harmonics** - stan DipSwitch ograniczający kompensowane harmoniczne do 29. Ustawienie aktywne, gdy DS8 jest załączony.
9. **DS7_limit_to_19odd_harmonics** - stan DipSwitch ograniczający kompensowane harmoniczne do 39. Ustawienie aktywne, gdy DS8 jest załączony.
10. **DS8_DS_override** - stan DipSwitch decydujący o użyciu ustawień z DipSwitch zamiast karty SD/pamięci FLASH.
11. **calibration_procedure_error** - niepowodzenie danego etapu kalibracji.
12. **L_grid_measured** - badanie impedancji zostało przeprowadzone. Wykonuje się ono tylko raz po uruchomieniu zasilania $24V$.
13. **Scope_snapshot_pending** - trwa zapisywanie przebiegów z oscyloskopu zainicjowane przez użytkownika.

14. **Scope_snapshot_error** - błąd podczas zapisywania przebiegów z oscyloskopu.
15. **SD_card_not_enough_data** - brak karty SD lub niewystarczające dane do uruchomienia przekształtnika.
16. **SD_no_CT_characteristic** - brak pliku z charakterystyką przekładników.
17. **SD_no_calibration** - brak pliku z danymi kalibracji.
18. **SD_no_harmonic_settings** - brak pliku z informacją o kompensowanych harmonicznych.
19. **SD_no_settings** - brak pliku z ogólnymi ustawieniami pracy przekształtnika.
20. **FLASH_not_enough_data** - brak lub niewystarczające dane zapisane w pamięci FLASH.
21. **FLASH_no_CT_characteristic** - brak danych z charakterystyką przekładników w pamięci FLASH.
22. **FLASH_no_calibration** - brak danych z danymi kalibracji w pamięci FLASH.
23. **FLASH_no_harmonic_settings** - brak danych z informacją o kompensowanych harmonicznych w pamięci FLASH.
24. **FLASH_no_settings** - brak danych z ogólnymi ustawieniami pracy przekształtnika w pamięci FLASH.

```
union STATUS
{
    Uint32 all[2];
    struct
    {
        Uint32 Init_done:1;
        Uint32 ONOFF:1;
        Uint32 DS1_switch_SD_CT:1;
        Uint32 DS2_enable_Q_comp:1;
        Uint32 DS3_enable_P_sym:1;
        Uint32 DS4_enable_H_comp:1;
        Uint32 DS5_limit_to_9odd_harmonics:1;
        Uint32 DS6_limit_to_14odd_harmonics:1;
        Uint32 DS7_limit_to_19odd_harmonics:1;
        Uint32 DS8_DS_override:1;
        Uint32 calibration_procedure_error:1;
        Uint32 L_grid_measured:1;
        Uint32 Scope_snapshot_pending:1;
        Uint32 Scope_snapshot_error:1;
        Uint32 SD_card_not_enough_data:1;
        Uint32 SD_no_CT_characteristic : 1;

        Uint32 SD_no_calibration : 1;
        Uint32 SD_no_harmonic_settings : 1;
        Uint32 SD_no_settings : 1;
        Uint32 FLASH_not_enough_data : 1;
        Uint32 FLASH_no_CT_characteristic : 1;
        Uint32 FLASH_no_calibration : 1;
        Uint32 FLASH_no_harmonic_settings : 1;
        Uint32 FLASH_no_settings : 1;
        Uint32 in_limit_Q : 1;
        Uint32 in_limit_P : 1;
        Uint32 in_limit_H : 1;
        Uint32 Conv_active : 1;
        Uint32 PLL_sync : 1;
        Uint32 Grid_present : 1;
        Uint32 rsvd1 : 2;
        Uint32 rsvd2 : 32;
    }bit;
};
```

Wydruk 2: Struktura **status**

25. **in_limit_Q** - przekształtnik w limicie kompensacji mocy biernej.
26. **in_limit_P** - przekształtnik w limicie symetryzacji mocy czynnej.
27. **in_limit_H** - przekształtnik w limicie kompensacji harmonicznych.
28. **Conv_active** - przekształtnik przeszedł procedurę ładowania obwodu DC i dokonał synchronizacji z siecią - układ pracuje.
29. **PLL_sync** - PLL zsynchronizowany z siecią.
30. **Grid_present** - flaga zapalona, gdy napięcia każdej z faz są większe od $120V_{rms}$.
31. **rsvd1** - przestrzeń na dodatkowe flagi.
32. **rsvd2** - przestrzeń na dodatkowe flagi.

## 1.3 Struktura control

Znaczenie poszczególnych pozycji:

1. **H_odd_a, H_odd_b, H_odd_c** - zmienne 32-bitowe dla trzech faz, każda zawierająca flagi selektywnej kompensacji nieparzystych harmonicznych.
2. **H_even_a, H_even_b, H_even_c** - zmienne 32-bitowe dla trzech faz, każda zawierająca flagi selektywnej kompensacji parzystych harmonicznych.
3. **Q_set** - 3 wartości typu float definiujące przesunięcie od zera kompensacji mocy biernej.
4. **Scope_snapshot** - wyzwolenie zapisu oscyloskopu na kartę SD. Oscyloskop jest synchronizowany względem przejścia napięcia fazy A przez zero.
5. **Modbus_FatFS_repeat** - opcja pozwalająca na szybki odczyt danych z karty SD. .
6. **save_to_FLASH** - zapisz do pamięci FLASH obecnie używane dane (kalibracja, chrakterystyka przekładnika, opcje harmonicznych, ogólne opcje).
7. **SD_save_H_settings** - zapisz na kartę SD obecnie ustawione harmoniczne do kompensacji.
8. **SD_save_settings** - zapisz na kartę SD obecnie ustawione ogólne opcje.
9. **CPU_reset** - wykonaj sprzętowy reset przekształtnika.
10. **ONOFF_override** - przejmij kontrolę nad przełącznikiem ONOFF. Kontrola jest wyłączana gdy nastąpi użycie rzeczywistego przełącznika.
11. **ONOFF** - wirtualny przełącznik. Jego wartość jest aktywna gdy flaga **ONOFF_override** jest załączona.
12. **enable_Q_comp_a** - załączenie kompensacji mocy biernej w pierwszej fazie.
13. **enable_Q_comp_b** - załączenie kompensacji mocy biernej w drugiej fazie.
14. **enable_Q_comp_c** - załączenie kompensacji mocy biernej w trzeciej fazie.
15. **enable_P_sym** - załączenie symetryzacji mocy czynnej.
16. **enable_H_comp** - załączenie kompensacji harmonicznych.
17. **version_Q_comp_a** - wersja kompensacji mocy biernej fazy pierwszej. 0: skompensuj moc bierną sieci do wartości zadanej przez **Q_set**; 1: wprowadź do sieci moc bierną o wartości zadanej przez **Q_set**.

18. **version_Q_comp_b** - wersja kompensacji mocy biernej fazy drugiej. 0: skompensuj moc bierną sieci do wartości zadanej przez **Q_set**; 1: wprowadź do sieci moc bierną o wartości zadanej przez **Q_set**.
19. **version_Q_comp_c** - wersja kompensacji mocy biernej fazy trzeciej. 0: skompensuj moc bierną sieci do wartości zadanej przez **Q_set**; 1: wprowadź do sieci moc bierną o wartości zadanej przez **Q_set**.
20. **version_P_sym** - wersja symetryzacji mocy czynnej. 0: symetryzuj prądy czynne sieci; 1: symetryzuj moc czynną sieci.
21. **tangens_range** - zadane wartości tangensa dla każdej fazy, które ma utrzymywać przekształtnik. Przy zadaniu wartości (0,0) dla danej fazy, moc jest kompensowana do zera mocy biernej. Zadanie wartości (-0.05,0.15) spowoduje skompensowanie mocy biernej przekraczającej tangens powyżej 0.15 i mocy biernej poniżej -0.05. Pomiędzy tymi zakresami moc bierna nie jest kompensowana. Zalecane wartości to (0.05,0.15).
22. **rsvd1** - przestrzeń na dodatkowe flagi.

# 2 Sygnalizacja diodami

Na płycie znajduje znajduje się 5 diod LED:

- LED1 - dioda zielona
- LED2 - dioda żółta
- LED3 - dioda czerwona
- LED4 - dioda zielona
- LED5 - dioda żółta

## 2.1 Stan normalny

LED1(zielona) odpowiada stanowi włącznika ON/OFF i częściowo stanowi przekształtnika. Wyłączona dioda oznacza stan OFF. Migająca dioda z częstotliwością $1Hz$ oznacza oczekiwanie na ponowne załączenie (powrót sieci lub upłynięcie czasu ograniczającego częstotliwość restartów). Podczas uruchamiania przekształtnika dioda będzie migać z częstotliwością $0.5Hz$, a po uruchomieniu zapali się. LED5 (żółta) zapala się gdy przekształtnik pracuje, ale ostatnio wystąpił błąd. Tę sygnalizację można zresetować poprzez utrzymanie przełącznika w pozycji ON na $2s$. LED3(czerwona) migając oznacza, że urządzenie jest w stanie błędu. LED2(pomarańczowa) wypełnieniem przebiegu o częstotliwości $0.5Hz$ informuje o obecnym limicie kompensacji:

- 0%(dioda zgaszona) - przekształtnik nie jest w limicie.
- 33% - limit kompensacji harmonicznych.
- 66% - limit symetryzacji mocy czynnej.
- 100%(dioda zapalona) - limit kompensacji mocy biernej.

## 2.2 Stan kalibracji

Podczas kalibracji diody przyjmują inne funkcje. LED1(zielona) jest zapalona na czas kalibracji. LED3(czerwona) zapala się, gdy nie udało się przejść do następnego etapu kalibracji. LED2(żółta) sygnalizuje liczbą mignięć na którym etapie kalibracji obecnie jest urządzenie:

1. Usuwanie przesunięć zera.
2. Kalibracja pomiarów prądu wzorcem $5A$.
3. Kalibracja pomiarów napięcia sieci wzorcem $30V$.
4. Kalibracja pomiaru napięcia obwodu DC wzorcem $30V$.

# 3 Komunikacja Modbus

W mapie pamięci zostały wykorzystane tylko **holding_registers** i **input_registers**, ale **discrete_inputs** i **coils** są zaimplementowane. W **holding_registers** umieszczona została wcześniej opisana struktura **control** i przestrzeń do pracy z kartą SD.

W **input_registers** znajdują się wartości pomiarowe i określające stan urządzenia. Jest tutaj także umieszczona na początku tablica **FatFS_response**. Jej pozycja została wybrana w ten sposób, aby była elementem niezmiennym w wypadku modyfikacji mapy pamięci. W tablicy **L_grid_previous** znajduje się 10 ostatnich pomiarów impedancji sieci. Parametr **KALMAN_HARMONICS** ma wartość 26 - tablica harmonicznych zawiera składową stałą pod indeksem zero, a następnie nieparzyste harmoniczne od 1 do 49. **harmonic_rms_values** to są wartości bezwzględne RMS harmonicznych, natomiast **harmonic_THD_individual** są obliczone według wzoru:

$$\frac{U_{n\_RMS}}{U_{1\_RMS}}$$

gdzie $n$ to numer harmonicznej pod danym indeksem tablicy.

Wartości są uporządkowane w strukturach **abc_struct** i **abcn_struct**, pokazanych na Wydrukach 7 i 8. Wartości w strukturze **Grid_filter** są uśrednione za okres $1s$. Wartości THD w strukturze **Grid_filter** uwzględniają tylko nieparzyste harmoniczne od 1 do 49.

Czas do zapisu i odczytu z RTC jest umieszczony w zmiennej 48-bitowej **time_BCD_struct**, widocznej na Wydruku 9. Po zapisie całego czasu w jednej ramce Modbus jest on automatycznie programowany w układzie RTC.

```
union CONTROL
{
    Uint32 all[10];
    struct
    {
        struct harmonic_odd_struct
        {
            Uint32 rsrvd1:1;
            Uint32 harm3:1;
            Uint32 harm5:1;
            Uint32 harm7:1;
            Uint32 harm9:1;
            Uint32 harm11:1;
            Uint32 harm13:1;
            Uint32 harm15:1;
            Uint32 harm17:1;
            Uint32 harm19:1;
            Uint32 harm21:1;
            Uint32 harm23:1;
            Uint32 harm25:1;
            Uint32 harm27:1;
            Uint32 harm29:1;
            Uint32 harm31:1;
            Uint32 harm33:1;
            Uint32 harm35:1;
            Uint32 harm37:1;
            Uint32 harm39:1;
            Uint32 harm41:1;
            Uint32 harm43:1;
            Uint32 harm45:1;
            Uint32 harm47:1;
            Uint32 harm49:1;
            Uint32 rsvd2:7;
        }H_odd_a, H_odd_b, H_odd_c;

        struct harmonic_even_struct
        {
            Uint32 harm2:1;
            Uint32 harm4:1;
            Uint32 rsvd1:30;
        }H_even_a, H_even_b, H_even_c;

        struct abc_struct Q_set;

        struct control_bits_struct
        {
            Uint32 Scope_snapshot:1;
            Uint32 save_to_FLASH:1;
            Uint32 SD_save_H_settings:1;
            Uint32 SD_save_settings:1;
            Uint32 CPU_reset:1;
            Uint32 rsvd1:11;
            //16bits
            Uint32 Modbus_FatFS_repeat:1;
            Uint32 ONOFF_override:1;
            Uint32 ONOFF:1;
            Uint32 enable_Q_comp_a:1;
            Uint32 enable_Q_comp_b:1;
            Uint32 enable_Q_comp_c:1;
            Uint32 enable_P_sym:1;
            Uint32 enable_H_comp:1;
            Uint32 version_Q_comp_a:1;
            Uint32 version_Q_comp_b:1;
            Uint32 version_Q_comp_c:1;
            Uint32 version_P_sym:1;
            Uint32 rsvd2:4;
            //16bits
        }control_bits;

        struct abc_struct tangens_range[2];
    }fields;
};
```

Wydruk 3: Struktura **control**

```
    struct
    {
        Uint16 FatFS_response[128];
        union ALARM alarm;
        union ALARM alarm_snapshot;
        union STATUS status;
        enum Machine_state_enum Machine_state;//32bit
        enum Converter_state_enum Converter_state;//32bit
        int16 Temp1;
        int16 Temp2;
        int16 Temp3;
        int16 Temp_CPU;
        float L_grid_previous[10];
        Uint16 file_head;
        struct time_BCD_struct RTC_current_time;//48bit
        Uint16 padding1[128-32-2*sizeof(union ALARM)-sizeof(union STATUS)];
        struct Grid_parameters_struct Grid_filter;
        float frequency;
        Uint16 padding2[256-sizeof(struct Grid_parameters_struct)-2];
        struct
        {
            float I_grid_a[KALMAN_HARMONICS];
            float I_grid_b[KALMAN_HARMONICS];
            float I_grid_c[KALMAN_HARMONICS];
            float U_grid_a[KALMAN_HARMONICS];
            float U_grid_b[KALMAN_HARMONICS];
            float U_grid_c[KALMAN_HARMONICS];
        }harmonic_rms_values;
        struct
        {
            float I_grid_a[KALMAN_HARMONICS];
            float I_grid_b[KALMAN_HARMONICS];
            float I_grid_c[KALMAN_HARMONICS];
            float U_grid_a[KALMAN_HARMONICS];
            float U_grid_b[KALMAN_HARMONICS];
            float U_grid_c[KALMAN_HARMONICS];
        }harmonic_THD_individual;
    }input_registers;
```

Wydruk 4: Struktura **input_registers** będąca elementem **Modbus_converter**

```
    struct
    {
        Uint16 FatFS_request[128];
        union CONTROL control;
        struct time_BCD_struct RTC_new_time;//48bit
        Uint16 even_address_padding;
    }holding_registers;
```

Wydruk 5: Struktura **holding_registers** będąca elementem **Modbus_converter**

```
struct Grid_parameters_struct
{
    struct abc_struct I_grid_1h;
    struct abc_struct U_grid_1h;
    struct abc_struct P_grid_1h;
    struct abc_struct P_load_1h;
    struct abc_struct P_conv_1h;
    struct abc_struct Q_grid_1h;
    struct abc_struct Q_load_1h;
    struct abc_struct Q_conv_1h;
    struct abc_struct S_grid_1h;
    struct abc_struct S_load_1h;
    struct abc_struct S_conv_1h;
    struct abc_struct PF_grid_1h;
    struct abc_struct THD_I_grid;
    struct abc_struct THD_U_grid;
    struct abc_struct U_grid;
    struct abc_struct I_grid;
    struct abcn_struct I_conv;
    struct abc_struct S_grid;
    struct abc_struct S_conv;
    struct abcn_struct Used_resources;
    struct
    {
        float PF_grid_1h;
        float P_load_1h;
        float Q_load_1h;
        float S_load_1h;
        float P_grid_1h;
        float Q_grid_1h;
        float S_grid_1h;
        float U_grid_1h;
    }average;
};
```

Wydruk 6: Struktura **Grid_filter** będąca elementem **input_registers**

```
struct abc_struct
{
    float a;
    float b;
    float c;
};
```

Wydruk 7: Struktura **abc_struct**

```
struct abcn_struct
{
    float a;
    float b;
    float c;
    float n;
};
```

Wydruk 8: Struktura **abcn_struct**

```
struct time_BCD_struct
{
    Uint16 second : 4;
    Uint16 second10 : 4;
    Uint16 minute : 4;
    Uint16 minute10 : 4;
    Uint16 hour : 4;
    Uint16 hour10 : 4;
    Uint16 day : 4;
    Uint16 day10 : 4;
    Uint16 month : 4;
    Uint16 month10 : 4;
    Uint16 year : 4;
    Uint16 year10 : 4;
};
```

Wydruk 9: Struktura **time_BCD_struct**

## 3.1 Numery identyfikacyjne urządzenia

W funkcji Modbus **0x11 Report Server ID** zostały zawarte numery identyfikujące urządzenie i jego elementy. Wygenerowanie pliku version-id.h wymaga posiadania Git Bash. Elementy **modbus_id**, **board_id**, **software_id** są to wartości zakodowane BCD. Wartość 0x0101 oznacza wersję 1.01. **fw_descriptor.dsc** zawiera w sobie znak obecności bootloadera (0xD=Loaded_FW, 0xB=Basic_FW(no BLD)), 6 znaków git sha_hex i ostatni znak oznaczający czy repozytorium było czyste (brak różnic git). Kolejna jest tablica 16 znaków typu urządzenia, a za nią 32bity unikalnego numeru urządzenia nadawanego przez producenta mikrokontrolera.

```
struct dsc_s{
  uint32_t    type:4;
  uint32_t    sha_hex:24;
  uint32_t    rsvd:3;
  uint32_t    clean:1;
} dsc;
```

Wydruk 10: struktura **fw_descriptor.dsc**

```
static void MdbReportServerID(void)
{
    Mdb_slave_ADU.error = No_Error;

    memcpy(Mdb_slave_ADU.data_out, Mdb_slave_ADU.data_in, 2);
    Uint16 temp_index = 3;
    Mdb_slave_ADU.data_out[temp_index++] = (fw_descriptor.modbus_id >> 8) & 0xFF;
    Mdb_slave_ADU.data_out[temp_index++] = (fw_descriptor.modbus_id >> 0) & 0xFF;
    Mdb_slave_ADU.data_out[temp_index++] = (fw_descriptor.board_id >> 8) & 0xFF;
    Mdb_slave_ADU.data_out[temp_index++] = (fw_descriptor.board_id >> 0) & 0xFF;
    Mdb_slave_ADU.data_out[temp_index++] = (fw_descriptor.software_id >> 8) & 0xFF;
    Mdb_slave_ADU.data_out[temp_index++] = (fw_descriptor.software_id >> 0) & 0xFF;
    Mdb_slave_ADU.data_out[temp_index++] = (*(Uint32 *)&fw_descriptor.dsc >> 24) & 0xFF;
    Mdb_slave_ADU.data_out[temp_index++] = (*(Uint32 *)&fw_descriptor.dsc >> 16) & 0xFF;
    Mdb_slave_ADU.data_out[temp_index++] = (*(Uint32 *)&fw_descriptor.dsc >> 8) & 0xFF;
    Mdb_slave_ADU.data_out[temp_index++] = (*(Uint32 *)&fw_descriptor.dsc >> 0) & 0xFF;

    Mdb_slave_ADU.data_out[temp_index++] = fw_descriptor.dev_type[0] & 0xFF;
    Mdb_slave_ADU.data_out[temp_index++] = fw_descriptor.dev_type[1] & 0xFF;
    Mdb_slave_ADU.data_out[temp_index++] = fw_descriptor.dev_type[2] & 0xFF;
    Mdb_slave_ADU.data_out[temp_index++] = fw_descriptor.dev_type[3] & 0xFF;
    Mdb_slave_ADU.data_out[temp_index++] = fw_descriptor.dev_type[4] & 0xFF;
    Mdb_slave_ADU.data_out[temp_index++] = fw_descriptor.dev_type[5] & 0xFF;
    Mdb_slave_ADU.data_out[temp_index++] = fw_descriptor.dev_type[6] & 0xFF;
    Mdb_slave_ADU.data_out[temp_index++] = fw_descriptor.dev_type[7] & 0xFF;
    Mdb_slave_ADU.data_out[temp_index++] = fw_descriptor.dev_type[8] & 0xFF;
    Mdb_slave_ADU.data_out[temp_index++] = fw_descriptor.dev_type[9] & 0xFF;
    Mdb_slave_ADU.data_out[temp_index++] = fw_descriptor.dev_type[10] & 0xFF;
    Mdb_slave_ADU.data_out[temp_index++] = fw_descriptor.dev_type[11] & 0xFF;
    Mdb_slave_ADU.data_out[temp_index++] = fw_descriptor.dev_type[12] & 0xFF;
    Mdb_slave_ADU.data_out[temp_index++] = fw_descriptor.dev_type[13] & 0xFF;
    Mdb_slave_ADU.data_out[temp_index++] = fw_descriptor.dev_type[14] & 0xFF;
    Mdb_slave_ADU.data_out[temp_index++] = fw_descriptor.dev_type[15] & 0xFF;
    //unique ID procesora - unikalny dla tego samego partnumber
    Mdb_slave_ADU.data_out[temp_index++] = (*(Uint32 *)(0x000703C0 + 0xC) >> 24) & 0xFF;
    Mdb_slave_ADU.data_out[temp_index++] = (*(Uint32 *)(0x000703C0 + 0xC) >> 16) & 0xFF;
    Mdb_slave_ADU.data_out[temp_index++] = (*(Uint32 *)(0x000703C0 + 0xC) >> 8) & 0xFF;
    Mdb_slave_ADU.data_out[temp_index++] = (*(Uint32 *)(0x000703C0 + 0xC) >> 0) & 0xFF;
    Mdb_slave_ADU.data_out[temp_index++] = 0x00;

    Mdb_slave_ADU.data_out_length = temp_index;
    Mdb_slave_ADU.data_out[2] = temp_index - 3;
}
```

Wydruk 11: Funkcja Modbus **0x11 Report Server ID**

## 3.2 Modbus FatFS

W **holding_registers** i **input_registers** znajdują się pola służące do obsługi FatFS poprzez Modbus. W **FatFS_request** należy podać funkcję i jej parametry, a zwrócone wartości pojawią się w **FatFS_response**.

Budowanie ramki należy rozpocząć od podania wywoływanej funkji. Typ enum z Wydruku 15 zawiera wszystkie funkcje dostępne w bibliotece FatFS, ale nie wszystkie są zaimplementowane. Wybraną wartość należy umieścić w pozycji 0 tabeli **FatFS_request**, traktując ją jako 8-bitową - 256 pozycji względem 128 przy 16-bitowej interpretacji. W celu pozostawienia swobody implementacji po stronie zewnętrznego sterownika, parametry do funkcji są podawane przez wskaźniki. Na przykładzie funkcji f_open:

```
FRESULT f_open (
FIL* fp,           /* [OUT] Pointer to the file object structure */
const TCHAR* path, /* [IN] File name */
BYTE mode          /* [IN] Mode flags */
);
```

Wydruk 12: Funkcja f_open

pierwszym parametrem jest FRESULT, czyli wartość zwracana funkcji. Na pozycji 1 tabeli **FatFS_request** należy umieścić liczbę, która będzie odpowiadać pozycji w tabeli **FatFS_response** - pod tym adresem znajdzie się FRESULT. Kolejnym parametrem jest struktura, do której zostanie przypisany plik. Obecnie dopuszczalne są operacje tylko na jednym pliku na raz, dlatego nie jest konieczne wypełnianie wskaźnika tego elementu. Podobnie wygląda sytuacja ze strukturą DIR, która też jest tylko jedna. Następnie mamy string z nazwą pliku do otwarcia. W pozycji 3 tabeli **FatFS_request** należy umieścić indeks, pod którym będzie się zaczynał string. Ostatnim elementem jest tryb otwierania pliku i jak w poprzednich elementach należy podać indeks tabeli, pod którym znajduje się potrzebna wartość. Pozostałe funkcje są obsługiwane w analogiczny sposób, przykład f_read:

```
FRESULT f_read (
FIL* fp,     /* [IN] File object */
void* buff,  /* [OUT] Buffer to store read data */
UINT btr,    /* [IN] Number of bytes to read */
UINT* br     /* [OUT] Number of bytes read */
);
```

Wydruk 13: Funkcja f_read

8-bitowa tabela wskaźników w **FatFS_request**, zapisana także w pseudokodzie na wydruku 14:

0. enum FatFS_function_enum function = FatFS_f_read;
1. indeks tablicy **FatFS_response**, pod którym się znajdzie FRESULT;
2. istnieje tylko jeden plik, a więc wartość może być dowolna;
3. indeks tablicy **FatFS_response**, od którego będą zaczynać się odczytane dane;
4. indeks tablicy **FatFS_request**, pod którym znajduje się liczba bajtów do odczytania;
5. indeks tablicy **FatFS_response**, pod którym znajdzie się liczba odczytanych bajtów;

Aby przyspieszyć odczyt danych z karty SD, można użyć flagi **Modbus_FatFS_repeat**. Sprawia ona, że jeśli poprzednią funkcją było f_read to po odczycie **FatFS_response** następuje jej automatyczne ponowienie. Kolejny odczyt **FatFS_response** będzie zawierał nowe dane i nie jest potrzebny zapis do **FatFS_request**, aby je zaktualizować.

```
Uint8 *FatFs_response8 = &FatFs_response;
Uint8 *FatFs_request8 = &FatFs_request;
//wywolanie
FatFs_request8[0] = FatFS_f_read;
FatFs_request8[1] = 1;
FatFs_request8[2] = x;
FatFs_request8[3] = 3;
FatFs_request8[4] = 6;
FatFs_request8[5] = 2;

//wewnetrzne wywolanie funkcji
FatFs_response8[1] = f_read(&fp, &FatFs_response8[3], FatFs_request8[6], &FatFs_response8[2])

//odczyt
FRESULT fresult = FatFs_response8[1];
Uint8 bytes_written = FatFs_response8[2];
Uint8 *data = &FatFs_response8[3];
```

Wydruk 14: Pseudokod wykorzystania FatFS przez Modbus

```
enum FatFS_function_enum
{
    FatFS_no_function = 0,
    FatFS_f_open = 1,
    FatFS_f_close = 2,
    FatFS_f_read = 3,
    FatFS_f_write = 4,
    FatFS_f_lseek = 5,
    FatFS_f_truncate = 6,     //not available
    FatFS_f_sync = 7,         //not available
    FatFS_f_forward = 8,      //not available
    FatFS_f_expand = 9,       //not available
    FatFS_f_gets = 10,        //not available
    FatFS_f_putc = 11,        //not available
    FatFS_f_puts = 12,        //not available
    FatFS_f_printf = 13,      //not available
    FatFS_f_tell = 14,
    FatFS_f_eof = 15,         //not available
    FatFS_f_size = 16,
    FatFS_f_error = 17,       //not available
//Directory Access
    FatFS_f_opendir = 18,
    FatFS_f_closedir = 19,
    FatFS_f_readdir = 20,
    FatFS_f_findfirst = 21,
    FatFS_f_findnext = 22,
//File and Directory Management
    FatFS_f_stat = 23,
    FatFS_f_unlink = 24,
    FatFS_f_rename = 25,
    FatFS_f_chmod = 26,       //not available
    FatFS_f_utime = 27,       //not available
    FatFS_f_mkdir = 28,       //not available
    FatFS_f_chdir = 29,       //not available
    FatFS_f_chdrive = 30,     //not available
    FatFS_f_getcwd = 31,      //not available
//Volume Management and System Configuration
    FatFS_f_mount = 32,       //not available
    FatFS_f_mkfs = 33,        //not available
    FatFS_f_fdisk = 34,       //not available
    FatFS_f_getfree = 35,     //not available
    FatFS_f_getlabel = 36,    //not available
    FatFS_f_setlabel = 37,    //not available
    FatFS_f_setcp = 38        //not available
};
```

Wydruk 15: Typ enum określający wywoływaną funkcję.

| harmonic | on_off phase A | on_off phase B | on_off phase C |
|---|---|---|---|
| 2 | 1 | 1 | 1 |
| 3 | 1 | 1 | 1 |
| 4 | 1 | 1 | 1 |
| 5 | 1 | 1 | 1 |
| 7 | 1 | 1 | 1 |
| 9 | 1 | 1 | 1 |
| 11 | 1 | 1 | 1 |
| 13 | 1 | 1 | 1 |
| 15 | 1 | 1 | 1 |
| 17 | 1 | 1 | 1 |
| 19 | 1 | 1 | 1 |
| 21 | 1 | 1 | 1 |
| 23 | 1 | 1 | 1 |
| 25 | 1 | 1 | 1 |
| 27 | 1 | 1 | 1 |
| 29 | 1 | 1 | 1 |
| 31 | 1 | 1 | 1 |
| 33 | 1 | 1 | 1 |
| 35 | 1 | 1 | 1 |
| 37 | 1 | 1 | 1 |
| 39 | 1 | 1 | 1 |
| 41 | 1 | 1 | 1 |
| 43 | 1 | 1 | 1 |
| 45 | 1 | 1 | 1 |
| 47 | 1 | 1 | 1 |
| 49 | 1 | 1 | 1 |

Tabela 1: Przykładowa tablica z załączoną kompensacją wszystkich harmonicznych.

# 4 Pliki używane przez przekształtnik

Formaty plików używane na karcie SD to CSV (MS-DOS), txt lub binarne. Nazwy plików nie mogą przekraczać 12 znaków. Wszystkie pliki znajdują się w głównym katalogu.

## 4.1 Plik harmon.csv

Ten plik zawsze jest wymagany do uruchomienia, nawet gdy nie jest włączona kompensacja harmonicznych. Przy odczycie pliku pierwszy wiersz z nazwami jest pomijany. Parametry są wprowadzane poprzez podanie numeru harmonicznej (obsługiwane nieparzyste od 3 do 49, parzyste 2 i 4), a następnie wartości zero lub jeden oznaczającej wyłączenie lub załączenie kompensacji harmonicznej w danej fazie. Przykład podano w tabeli 1.

## 4.2 Plik settings.csv

W tym pliku podawane są ogólne opcje działania przekształtnika. Ten plik także jest zawsze wymagany do działania urządzenia. Dostępne opcje ustawień są podane w tabeli 2. W pierwszej kolumnie powinien się znajdować dokładny tekst danej opcji (wielkie litery),

| | |
|---|---|
| STATIC Q COMPENSATION A | 10 |
| STATIC Q COMPENSATION B | -10 |
| STATIC Q COMPENSATION C | 0 |
| ENABLE Q COMPENSATION A | 1 |
| ENABLE Q COMPENSATION B | 1 |
| ENABLE Q COMPENSATION C | 1 |
| ENABLE P SYMMETRIZATION | 0 |
| ENABLE H COMPENSATION | 0 |
| VERSION P SYMMETRIZATION | 0 |
| VERSION Q COMPENSATION A | 0 |
| VERSION Q COMPENSATION B | 0 |
| VERSION Q COMPENSATION C | 0 |
| C | 0.00099 |
| L | 0.00047 |
| I | 16 |
| TANGENS RANGE A HIGH | 0.05 |
| TANGENS RANGE B HIGH | 0.05 |
| TANGENS RANGE C HIGH | 0.05 |
| TANGENS RANGE A LOW | 0.15 |
| TANGENS RANGE B LOW | 0.15 |
| TANGENS RANGE C LOW | 0.15 |
| BAUDRATE | 115200 |

Tabela 2: Przykładowa tablica z opcjami działania urządzenia.

natomiast w drugiej kolumnie parametr tej opcji. Opisy poszczególnych opcji znajdują się w strukturze **control**. Parametry C, L, I powinny być ustawione przy produkcji. Oznaczają kolejno pojemność obwodu DC, indukcyjność dławików, znamionowy prąd przekształtnika. Parametr BAUDRATE ustawia prędkość Modbus.

## 4.3 Plik calib.csv

Ten plik jest tworzony przez urządzenie po przejściu procesu kalibracji. Jest on niezbędny do działania urządzenia. Nie należy go tworzyć samodzielnie.

## 4.4 Plik CT_CHAR.TXT

W tym pliku znajduje się charakterystyka przekładników użytych w urządzeniu. Jest on wymagany, gdy DIPSWITCH konfigurujący wskazuje na pracę z przekładnikami. Domyślnie ten plik jest generowany przez urządzenie do badania przekładników, ale jest możliwość ręcznego wypełnienia go. Algorytm dokonuje liniowej interpolacji między punktami, a więc zadanie dwóch punktów spowoduje liniową zmianę przesunięcia między zadanymi punktami. Nie jest dokonywana ekstrapolacja - jeśli zmierzony prąd przekładnika wychodzi poza opisaną charakterystykę przyjmuje się punkt skrajny do obliczeń.

Pierwszy wiersz jest zawsze pomijany (kolejność kolumn jest z góry zdefiniowana). Maksymalna liczba punktów pomiarowych wynosi 60, a ich kolejność nie ma znaczenia. Możliwe jest także podanie jednego punktu pomiarowego, uzyskując kompensację przekładnika niezależną od jego prądu. Pierwsza kolumna zawiera prąd przekładnika, przy którym został wykonany

pomiar. 3 kolejne kolumny zawierają przekładnię przekładnika dla zadanego prądu. W ostatnich 3 kolumnach znajduje się przesunięcie przekładnika dla $50Hz$ wymuszenia.

| Current [A] | CT A ratio [A/A] | CT B ratio [A/A] | CT C ratio [A/A] | CT A phase [degrees] | CT B phase [degrees] | CT C phase [degrees] |
|---|---|---|---|---|---|---|
| 96 | 20.045 | 20.0784 | 20.2585 | 0.295057 | 0.308327 | 0.855952 |
| 85.5601 | 20.0505 | 20.0815 | 20.2606 | 0.316761 | 0.333645 | 0.786549 |
| 76.2555 | 20.0545 | 20.0853 | 20.2676 | 0.341358 | 0.36046 | 0.793838 |
| 67.9628 | 20.0584 | 20.0893 | 20.2757 | 0.367688 | 0.389332 | 0.825237 |
| 60.5719 | 20.0616 | 20.093 | 20.2841 | 0.39356 | 0.418437 | 0.873105 |
| 53.9848 | 20.0645 | 20.0967 | 20.2929 | 0.420092 | 0.447429 | 0.932033 |
| 48.114 | 20.0675 | 20.1003 | 20.3015 | 0.445378 | 0.475754 | 0.999408 |
| 42.8816 | 20.0703 | 20.1037 | 20.3087 | 0.470222 | 0.503979 | 1.0751 |
| 38.2183 | 20.0727 | 20.1063 | 20.3162 | 0.494301 | 0.53102 | 1.15654 |
| 34.0621 | 20.075 | 20.1088 | 20.3226 | 0.518138 | 0.55701 | 1.24109 |
| 30.3579 | 20.0772 | 20.1114 | 20.3294 | 0.541239 | 0.583441 | 1.32927 |
| 27.0565 | 20.0792 | 20.1134 | 20.3363 | 0.562897 | 0.608012 | 1.41942 |
| 24.1141 | 20.0808 | 20.1156 | 20.3422 | 0.586405 | 0.633909 | 1.5103 |
| 21.4917 | 20.0828 | 20.1174 | 20.3481 | 0.609687 | 0.659467 | 1.60142 |
| 19.1545 | 20.0842 | 20.1191 | 20.3527 | 0.634126 | 0.685697 | 1.69168 |
| 17.0715 | 20.0861 | 20.1209 | 20.3574 | 0.658828 | 0.711843 | 1.78217 |
| 15.215 | 20.088 | 20.1237 | 20.3624 | 0.684166 | 0.739172 | 1.8699 |
| 13.5604 | 20.0901 | 20.1259 | 20.367 | 0.71032 | 0.765752 | 1.95554 |
| 12.0857 | 20.0929 | 20.1288 | 20.3721 | 0.736042 | 0.793394 | 2.04001 |
| 10.7714 | 20.097 | 20.1327 | 20.3783 | 0.761018 | 0.819627 | 2.12281 |
| 9.6 | 20.1017 | 20.1377 | 20.3854 | 0.789246 | 0.848453 | 2.20344 |
| 8.55601 | 20.1071 | 20.1422 | 20.3939 | 0.818019 | 0.876597 | 2.28638 |
| 7.62555 | 20.1129 | 20.1485 | 20.4029 | 0.844877 | 0.906718 | 2.3663 |
| 6.79628 | 20.1187 | 20.1554 | 20.4135 | 0.874263 | 0.940678 | 2.44781 |
| 6.05719 | 20.1259 | 20.1609 | 20.4251 | 0.903875 | 0.974618 | 2.52564 |
| 5.39848 | 20.1338 | 20.1695 | 20.4378 | 0.938458 | 1.01356 | 2.60968 |
| 4.8114 | 20.1417 | 20.1778 | 20.452 | 0.972541 | 1.05112 | 2.69315 |
| 4.28817 | 20.1495 | 20.1871 | 20.4658 | 1.01092 | 1.09196 | 2.78059 |
| 3.82183 | 20.1586 | 20.197 | 20.4812 | 1.05194 | 1.13856 | 2.87539 |
| 3.40621 | 20.1658 | 20.2077 | 20.4985 | 1.09237 | 1.18092 | 2.96432 |
| 3.03579 | 20.1766 | 20.2197 | 20.5194 | 1.13854 | 1.22919 | 3.0603 |
| 2.70565 | 20.1873 | 20.2342 | 20.5426 | 1.18669 | 1.28156 | 3.16011 |
| 2.41141 | 20.2019 | 20.248 | 20.5674 | 1.24612 | 1.34942 | 3.26749 |
| 2.14917 | 20.2154 | 20.2623 | 20.5931 | 1.29991 | 1.40468 | 3.37931 |
| 1.91545 | 20.2313 | 20.2781 | 20.6206 | 1.35978 | 1.47139 | 3.49853 |
| 1.70715 | 20.2469 | 20.2967 | 20.6501 | 1.44044 | 1.54247 | 3.62777 |
| 1.5215 | 20.2692 | 20.3219 | 20.6842 | 1.5313 | 1.63374 | 3.75475 |
| 1.35604 | 20.2899 | 20.3469 | 20.7239 | 1.61721 | 1.72612 | 3.90636 |
| 1.20857 | 20.313 | 20.3676 | 20.7604 | 1.70004 | 1.79653 | 4.05177 |
| 1.07714 | 20.3526 | 20.4163 | 20.8055 | 1.85527 | 1.95657 | 4.20103 |
| 0.960002 | 20.3787 | 20.4461 | 20.849 | 1.95665 | 2.06122 | 4.39098 |
| 0.855602 | 20.4217 | 20.4892 | 20.8947 | 2.09968 | 2.18911 | 4.56566 |
| 0.762556 | 20.4606 | 20.534 | 20.9453 | 2.22069 | 2.32579 | 4.76015 |
| 0.679629 | 20.523 | 20.5908 | 21.0013 | 2.40207 | 2.52493 | 4.99063 |
| 0.60572 | 20.6253 | 20.697 | 21.0559 | 2.76056 | 2.88823 | 5.26117 |
| 0.539849 | 20.6995 | 20.7795 | 21.1234 | 2.96068 | 3.1181 | 5.48537 |
| 0.481141 | 20.768 | 20.8574 | 21.1874 | 3.15373 | 3.29702 | 5.76235 |
| 0.428817 | 20.8904 | 20.9813 | 21.2659 | 3.56892 | 3.70349 | 6.12683 |
| 0.382184 | 21.0134 | 21.1242 | 21.3365 | 3.99167 | 4.10116 | 6.41307 |
| 0.340622 | 21.1466 | 21.2597 | 21.423 | 4.29285 | 4.42276 | 6.76072 |
| 0.303579 | 21.2868 | 21.4092 | 21.5233 | 4.71253 | 4.79954 | 7.15292 |
| 0.270566 | 21.44 | 21.5686 | 21.6116 | 5.13457 | 5.2575 | 7.53262 |
| 0.241142 | 21.6117 | 21.748 | 21.7117 | 5.67949 | 5.81147 | 8.01736 |
| 0.214918 | 21.8003 | 21.9721 | 21.8619 | 6.22165 | 6.36811 | 8.53165 |
| 0.191546 | 22.0184 | 22.238 | 21.9731 | 6.92668 | 7.07514 | 9.09269 |
| 0.170715 | 22.2756 | 22.5018 | 22.121 | 7.67816 | 7.96051 | 9.79948 |
| 0.15215 | 22.5915 | 22.8661 | 22.2798 | 8.65635 | 8.82155 | 10.406 |
| 0.135604 | 22.9828 | 23.2734 | 22.4682 | 9.87042 | 10.0184 | lis.68 |
| 0.120857 | 23.5043 | 23.797 | 22.6715 | 11.154 | 11.315 | 12.1753 |
| 0.107714 | 24.1941 | 24.5756 | 22.9851 | 13.0461 | 13.3065 | 13.3434 |

Tabela 3: Przykładowa tablica charakterystyki przekładnika.

## 4.5 Plik HEAD.TXT

Ten plik zawiera liczbę identyfikującą ostatnio(obecnie) wykorzystywane pliki i przyjmuje wartości $x$ od 1 do 999. Do tych plików zalicza się:

- xLogs.bin - zawiera dane binarne zapisywane co $10s$ działania kompensatora (uruchomione sterowanie). Maksymalna długość pliku to jeden dzień, następnie wartość w head.txt jest inkrementowana.

```
struct FatFS_time_struct
{
  Uint32 second_2:5;
  Uint32 minute:6;
  Uint32 hour:5;
  Uint32 day:5;
  Uint32 month:4;
  Uint32 year:7;
};
```

```
float temp_array[17];
temp_array[0] = *(float *)&FatFS_time;
temp_array[1] = Grid_filter.U_grid_1h.a;
temp_array[2] = Grid_filter.U_grid_1h.b;
temp_array[3] = Grid_filter.U_grid_1h.c;
temp_array[4] = Grid_filter.Q_load_1h.a;
temp_array[5] = Grid_filter.Q_load_1h.b;
temp_array[6] = Grid_filter.Q_load_1h.c;
temp_array[7] = Grid_filter.P_load_1h.a;
temp_array[8] = Grid_filter.P_load_1h.b;
temp_array[9] = Grid_filter.P_load_1h.c;

temp_array[10] = Grid_filter.Q_conv_1h.a;
temp_array[11] = Grid_filter.Q_conv_1h.b;
temp_array[12] = Grid_filter.Q_conv_1h.c;
temp_array[13] = Grid_filter.P_conv_1h.a;
temp_array[14] = Grid_filter.P_conv_1h.b;
temp_array[15] = Grid_filter.P_conv_1h.c;
temp_array[16] = fmaxf(Meas.Temp1, fmaxf(Meas.Temp2, Meas.Temp3));
```

- xError.txt - zawiera tekstowy opis jak długo pracował przekształtnik i ostatnie alarmy. Przykład:

  The converter has worked for:
  0:9:22:10 days/hours/minutes/seconds

  List of snapshot errors:
  I_comp_c_SDL
  TZ_SD_COMP
  TZ

  List of all errors:
  I_comp_c_SDL
  TZ_SD_COMP
  TZ

- xScope.bin - zawiera binarny zrzut oscyloskopu zatrzymany na zdarzeniu, które zatrzymało pracę przekształtnika. Domyślnie punkt zatrzynia jest w 800 próbce.

```
//dane poszczegolnych kanalow
Scope.data_in[0] = &Meas.U_grid.a;
Scope.data_in[1] = &Meas.U_grid.b;
Scope.data_in[2] = &Meas.U_grid.c;
Scope.data_in[3] = &Meas.I_grid_avg.a;
Scope.data_in[4] = &Meas.I_grid_avg.b;
Scope.data_in[5] = &Meas.I_grid_avg.c;
Scope.data_in[6] = &Meas.I_conv_avg.a;
Scope.data_in[7] = &Meas.I_conv_avg.b;
Scope.data_in[8] = &Meas.I_conv_avg.c;
Scope.data_in[9] = &Meas.I_conv_avg.n;
Scope.data_in[10] = &Meas.U_dc_avg;
//tablica z danymi zapisywana na karte SD
float data[11][1250];
```

# 5 Proces kalibracji

Każdy kontakt z płytą przekształtnika musi być poprzedzony dotknięciem obudowy, a następnie potencjału GND płyty. Ma to na celu wyrównanie w bezpieczny sposób potencjałów między przekształtnikiem, a osobą dotykającą. Procedura kalibracji przebiega według punktów:

1. Skonfiguruj lutowaną zworką odpowiedni pomiar prądu – przewlekany/przekładnik.
2. Podepnij termistor(y).
3. Przygotuj okablowanie służące do kalibracji:
   (a) Zewrzyj DC-link przekształtnika;
   (b) Połącz w szereg wszystkie pomiary prądu, tak aby przez wszystkie przepływał ten sam prąd – kierunek nie ma znaczenia;
   (c) Ujemny zacisk zasilacza laboratoryjnego podepnij do terminala N, a następnie kabelkiem z N do szeregowo połączonych pomiarów prądu. Dodatni zacisk zasilacza podepnij na drugim końcu szeregu.
4. Podepnij programator JTAG najpierw od strony PCB, a następnie USB do komputera. Taka kolejność zapewnia większe bezpieczeństwo w przypadku wystąpienia ESD.
5. Przygotuj kartę SD z plikami podanymi poniżej, a następnie umieść ją w slocie na płycie przekształtnika.
   (a) cpu01.hex;
   (b) cpu02.hex;
   (c) harmon.csv;
   (d) settings.csv – może zawierać przygotowane ustawienia do testów mocy;
   (e) CT_CHAR.CSV, jeśli wykorzystywany jest przekładnik.
6. Włącz zasilanie $24V$ – wentylatory zaczną pracować, jeśli obecne.
7. Wgraj kod bootloadera za pomocą programu UniFlash.
8. Bootloader załaduje właściwy program z karty SD i po kilku sekundach zacznie migać zielona dioda, a wentylatory się wyłączą.
9. Sprawdź czy zapalone są diody przy transformatorkach, a także czy równomiernie się świecą.

10. Przystąp do kalibracji poprzez przełączenie głównego przycisku na pozycję ON. Kolejne przełączenia będą sygnalizować procesowi gotowość ustawień. Ilość mrugnięć pomarańczowej diody sygnalizuje etap w którym obecnie znajduje się kalibracja. Czerwona dioda oznacza niepowodzenie danego etapu kalibracji, który można powtórzyć ponownie przełączając główny przełącznik. W przypadku powtarzającego się niepowodzenia może to oznaczać niedziałające pomiary w danym urządzeniu.

    (a) Pierwszy etap kalibracji ma na celu usunięcie przesunięć zera i błędów wzmocnienia pomiarów, a zarazem jest to test ich działania. W tym punkcie zostaną usunięte przesunięcia zera na pomiarach – zasilacz laboratoryjny musi być wyłączony. Zapali się na pomarańczowo dioda sygnalizująca trwający proces.

    (b) Uruchom zasilacz na dokładne $5A$ i zmień pozycję głównego przełącznika. Zostanie skalibrowane wzmocnienie każdego pomiaru prądu.

    (c) Wyłącz zasilacz laboratoryjny. Odepnij kabelek łączący terminal N z początkiem szeregu pomiarów prądu i uruchom zasilacz na napięcie $30V$. Po zmianie pozycji głównego przełącznika zostanie skalibrowane wzmocnienie pomiarów napięcia sieci.

    (d) Wyłącz zasilacz laboratoryjny i usuń zwarcie na pomiarze napięcia DC-link. Podepnij zasilacz równolegle do DC-link i uruchom na napięciu $30V$. Po zmianie pozycji głównego przełącznika nastąpi kalibracja wzmocnienia pomiaru napięcia DC-link.

11. Po przejściu powyższych etapów dane kalibracji zostaną zapisane na karcie SD i zielona dioda będzie migać. Przy poprawności działania pomiarów inne błędy mogą zostać wychwycone przez sam przekształtnik podczas testów mocy i zasygnalizowane flagą. Zaleca się uruchomienie przekształtnika z niższego napięcia sieci np. $30Vrms$, aby następnie zrobić test z $230Vrms$.

# 6 Problemy ze sterowaniem członami pasywnymi

Ten rozdział opisuje obecny algorytm sterowania blokami, jego wady i możliwe rozwiązanie.

## 6.1 Algorytm sterowania blokami w LRM001

1. Definicje.
   - P - aktualna moc czynna
   - Q - aktualna moc bierna
   - cos_cfg - ustawiona wartość cosinusa
   - cos - aktualna wartość cosinusa
   - tg_cfg - wartość tangensa wyliczona z cos_cfg. Do wykonywania obliczeń.
   - Qblok(min) - najmniejsza wartość bloku regulacyjnego (moduł liczby).
   - Moc Q jest dodatnia dla obciążenia indukcyjnego i ujemna dla pojemnościowego.
   - Moc Qblok - dodatnia dla bloku indukcyjnego i ujemna dla bloku pojemnościowego.
   - Strefa nieczułości dla bloków pojemnościowych:
     - załączenie bloku, gdy moc bierna jest przekroczona o wartość |Qblok(min)/2|
     - wyłączenie bloku, gdy moc bierna jest mniejsza od zdefiniowanej o wartość |Qblok(min)|
   - Timery:
     - TimerAlarmL, TimerAlarmC - timery alarmów niedokompensowania/przekompensowania.
     - TimerOn - timer załączenia bloków pojemnościowych/indukcyjnych.
     - TimerOff - timer wyłączenia bloków pojemnościowych/indukcyjnych.
2. Wstępna analiza.

   Analiza odbywa się co 100ms po uśrednieniu pomiarów z 5 okresów. W zależności od znaku mocy biernej uruchamiane jest sterowanie blokami pojemnościowymi lub indukcyjnymi. Nie ma miejsca doregulowywanie blokami indukcyjnymi w przypadku obciążenia indukcyjnego i załączonych blokach pojemnościowych.
3. Sterowanie - moc bierna indukcyjna.
   - skasowanie TimerAlarmC
   - wyliczenie dopuszczalnej mocy biernej na podstawie aktualnej mocy czynnej: $Qcfg = P * tg\_cfg$
   - porównanie wartości cos z cos_cfg

     $cos < cos\_cfg$ (wymagane załączenie bloku)
     - sprawdzenie, czy nie są przypadkiem załączone bloki indukcyjne.
       Jeśli tak, wyłączenie zgodnie z TimerOff
       return
     - wyliczenie różnicy mocy:
       $Qdelta = Q - Qcfg$
     - gdy |Qdelta| < |Qblok(min)/2| - reset TimerAlarmL, TimerOn, TimerOff.
       return (strefa nieczułości)
     - tutaj odliczamy TimerAlarmL
     - odliczanie/zazbrojenie TimerOn. Gdy czas nie upłynął - return.

- wyznaczenie nowego zestawu bloków do załączenia.
  Wyznaczenie polega na wirtualnym wyłączeniu załączonych bloków (wyznaczenie aktualnej mocy biernej bez załączonych bloków) i przeliczeniu nowego zestawu bloków. Następnie wykonywane jest załączenie wybranych bloków.

$cos > cos\_cfg$ (możliwe odłączenie bloku)
- skasowanie TimerAlarmL
  Jeśli tak, wyłączenie zgodnie z TimerOff
  return
- sprawdzenie, czy są załączone jakieś bloki pojemnościowe. Gdy nie - można załączyć bloki indukcyjne (o ile są) aby przesunąć się w kierunku zadanego cos_cfg. Jest to wymagane w przypadku kompensacji tylko obciążenia pojemnościowego. Załączenie zgodnie z TimerOn.
  return
- sprawdzenie czy można wyłączyć któryś z bloków:
  $Qdelta = Q - Qcfg$
- gdy |Qdelta| < |Qblok(min)| - reset TimerOn i TimerOff.
  return (strefa nieczułości)
- odliczanie/zazbrojenie TimerOff. Gdy czas nie upłynął - return.
- wyznaczenie nowego zestawu bloków do załączenia. W tym wypadku Qcfg jest zmniejszona o wartość |Qblok(min)/2| co pozwala lepiej doregulować cos.
  $cos = cos\_cfg$ (chyba niemożliwe)
- kasowanie TimerOn, TimerOff, TimerAlarmL.

4. Sterowanie - moc bierna pojemnościowa.

   Analogicznie jak powyżej. Zamiana bloków pojemnościowych i indukcyjnych oraz TimerAlarmC i TimerAlarmL.

## 6.2 Analiza problemów algorytmu sterowania blokami w LRM001

Równoległa praca przekształtnika i bloków pasywnych wymaga zmiany założeń i celów osiąganych przez algorytm:

1. Podstawowym celem bloków pasywnych jest zapewnienie pracy przekształtnika w jego możliwym zakresie (bez wchodzenia w limity kompensacji/symetryzacji).

2. Dodatkowo, należy dążyć do jak najmniejszego obciążenia przekształtnika (większość mocy biernej kompensowana przez człony pasywne).

   Wypadkowy prąd w przewodzie neutralnym może osiągać wartości większe niż w przewodach fazowych. Przy kompensacji tylko mocy biernej, moc przekształtnika może zostać ograniczona dwukrotnie przy dużej asymetrii. Biorąc pod uwagę też symetryzację mocy czynnej, prąd w przewodzie neutralnym może osiągnąć wartości 3 razy większe niż prądy fazowe. Prąd przewodu neutralnego przekształtnika przepływa przez kondensatory elektrolityczne, i podgrzewając je pogarsza żywotność. Jest on najbardziej istotny do ograniczenia, a następnie dopiero prądy przewodów fazowych.

   Takie podejście oznacza, że równoległe załączanie członów pojemnościowych i indukcyjnych może poszerzyć zakres pracy przekształtnika.

3. Straty przekształtnika rosną kwadratowo względem prądów przekształtnika. Oznacza to, że praca przy 50% prądu znamionowego generuje tylko 25% strat pracy znamionowej. Wpływ na żywotność przekształtnika przy pracy poniżej tego progu jest znikomy i można

wprowadzić strefę nieczułości algorytmu załączania bloków pasywnych. Takie podejście zmniejszy zużycie styczników.

4. Algorytm powinien działać poprawnie także w przypadku kompensacji do wartości zadanej innej niż $0var$. Celem wciąż jest odciążenie przekształtnika.

Obecny algorytm dopuszcza tylko operację załączenia/wyłączenia w jednym cyklu. Jednak mogą nastąpić sytuacje, gdzie stan wszystkich styczników ulegnie zmianie. Przykładem jest kompensacja $5kvar$ z użyciem członów o mocach 5, 3, $3kvar$. Wedle obecnego algorytmu (lub mojego zrozumienia) zostanie załączony człon 5kvar - sytuacja poprawna. Następnie moc bierna do kompensacji wzrasta do $6kvar$ - nic się nie stanie, podczas gdy można załączyć dwa człony $3kvar$ przy wyłączonym $5kvar$.

Proponuję rozważenie opcji kompensacji, która wynikła z burzy mózgów w laboratorium. Należy utworzyć listę wszystkich konfiguracji członów (powiedzmy, że na razie tylko indukcyjnych) wraz z ich wypadkową indukcyjnością. Wykonywanie obliczeń na indukcyjnościach uniezależnia nas od zmian napięcia sieci. Od tych indukcyjności należy odjąć zadaną indukcyjność (potrzebną do skompensowania), a następnie z użyciem funkcji qsort(wbudowana w język C) posortować względem wartości absolutnych. Na pierwszej pozycji listy uzyskamy najlepszy wariant członów.

Na bazie otrzymanej listy można wprowadzić dodatkowe funkcjonalności optymalizujące. Z listy możemy wykreślić pozycje, które są obecnie niemożliwe do wykorzystania (np. nie można jeszcze danego członu załączyć). Wciąż uzyskamy rozwiązanie, które jest najbliżej zadanej wartości. Kolejnym krokiem może być wykreślenie pozycji, które wymagają przełączenia więcej niż np. jednego bloku - ograniczamy liczbę przełączeń. W zależności od uchybu mocy lub od całki uchybu mocy, liczbę możliwych przełączeń można modyfikować - zapewniając przełączenia tylko w momentach, w których to jest konieczne.

Zapisanie algorytmu w ten sposób naszym zdaniem umożliwia łatwe zaimplementowanie wielu innych strategii sterowania blokami pasywnymi.