Заголовок

1. МОЇ ПРОПОЗИЦІЇ ДО ОРГАНІЗАЦІЇ ВИКОРИСТАННЯ ПАМ’ЯТІ ДЛЯ МОДУЛЬНОГО ПРИЛАДУ.....2

2.ТАБЛИЦЯ КОНФІГУРАЦІЇ 4

3.ФОРМАТ ПОЛЯ PARAM ДЛЯ ВИЗНАЧЕННЯ, ЯКИЙ СИГНАЛ ЗАВОДИТЬСЯ НА ВХІД(ВХОДИ) ФУНКЦІОНАЛЬНОГО БЛОКУ 5

4. КОНСТАНТИ І МАСИВ ВКАЗІВНИКІВ НА НАЛАШТУВАНЬ У ДИНАМІЧНІЙ ПАМ’ЯТІ.....6

5. МАКРОСИ 8

6. ТАБЛИЦЯ НАЛАШТУВАНЬ, РОЗМІРИ ЯКИХ ВІДОМО НА МОМЕНТ КОМПІЛЯЦІЇ 9

7. СТРУКТУРИ ДЛЯ ДИСКРЕТНИХ ВХОДІВ.....11

8.СТРУКТУРИ ДЛЯ ДИСКРЕТНОГО ВИХОДУ І СВІТЛОІНДИКАТОРА 12

9.СТРУКТУРИ ДЛЯ ЕЛЕМЕНТУ СТАНДАРТНОЇ ЛОГІКИ «І» 14

10. СТРУКТУРИ ДЛЯ ЕЛЕМЕНТУ СТАНДАРТНОЇ ЛОГІКИ «АБО».....15

11. СТРУКТУРИ ДЛЯ ЕЛЕМЕНТУ СТАНДАРТНОЇ ЛОГІКИ «ВИКЛ.АБО» 16

12. СТРУКТУРИ ДЛЯ ЕЛЕМЕНТУ СТАНДАРТНОЇ ЛОГІКИ «НЕ» 17

13. СТРУКТУРИ ДЛЯ ЕЛЕМЕНТУ СТАНДАРТНОЇ ЛОГІКИ «БФ-ТАЙМЕР» 18

14. СТРУКТУРИ ДЛЯ ЕЛЕМЕНТУ СТАНДАРТНОЇ ЛОГІКИ «ТРИҐЕР» 20

15. СТРУКТУРИ ДЛЯ ЕЛЕМЕНТУ ФУНКЦІОНАЛЬНОГО БЛОКУ «ГЕНЕРАТОР ПЕРІОДИЧНИХ СИГНАЛІВ» 21

16. СТРУКТУРИ ДЛЯ ЕЛЕМЕНТУ ФУНКЦІОНАЛЬНОГО БЛОКУ «СВІТЛОЗВУКОВА СИГНАЛІЗАЦІЯ» 22

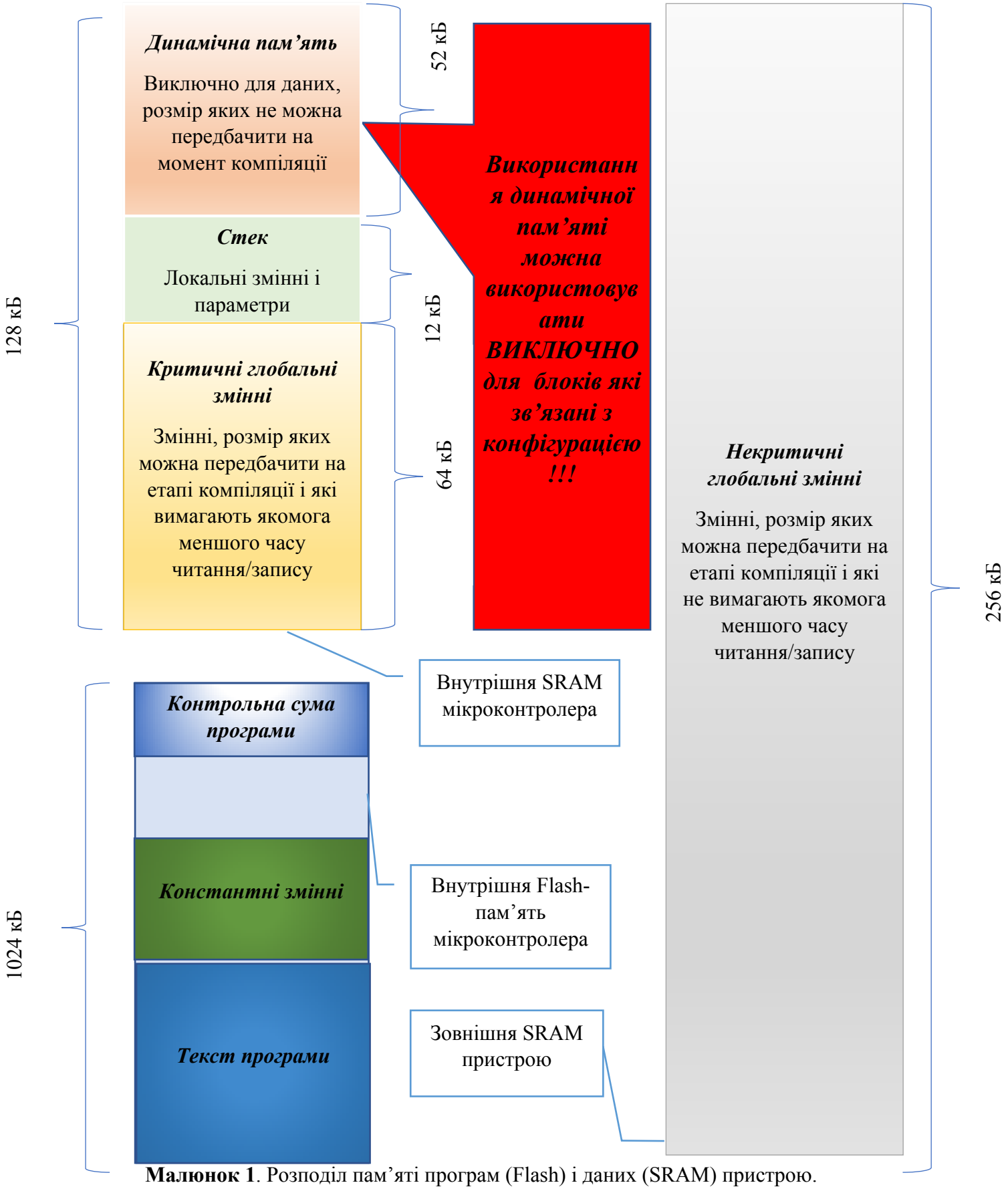
17. СТРУКТУРИ ДЛЯ ЕЛЕМЕНТУ ФУНКЦІОНАЛЬНОГО БЛОКУ «ШИНКИ ГРУПОВОЇ СИГНАЛІЗАЦІЇ» 24

18.СТРУКТУРИ ДЛЯ ФУНКЦІОНАЛЬНИХ КНОПОК І КОМАНД ТЕЛЕУПРАВЛІННЯ 26

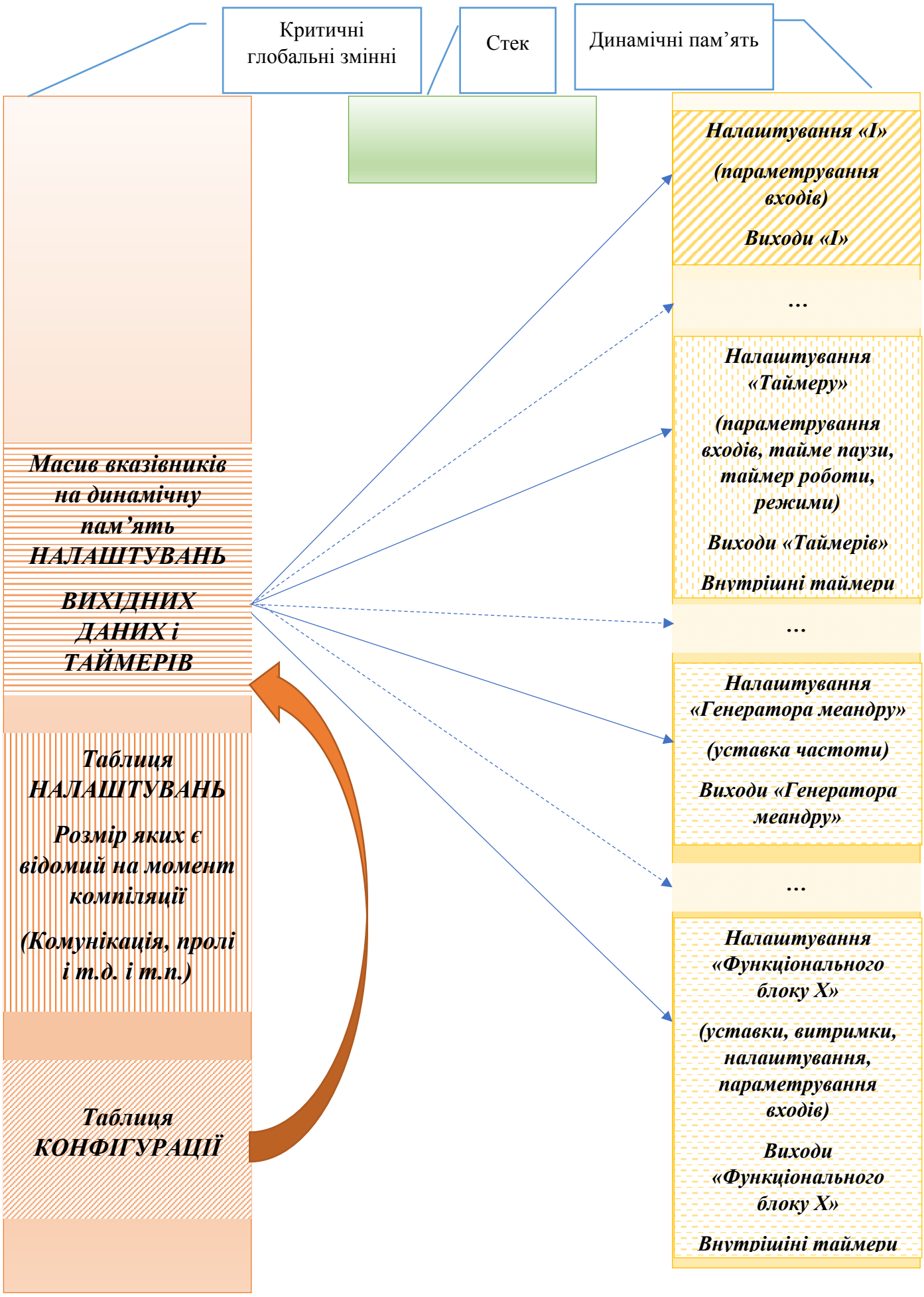
МОЯ ПРОПОЗИЦІЯ (ДИСКУСІЙНА!) ЩОДО АЛГОРИТМУ ОПРАЦЮВАННЯ ЛОГІКИ ВСІХ ФУНКЦІОНАЛЬНИХ БЛОКІВ У КОНКРЕТНИЙ ЧАСОВИЙ ВІДРІЗОК 27

1. Мої пропозиції до організації використання пам’яті для модульного приладу

(з прив’язкою до приладу МРЗС-05Л-ЦС)



Малюнок 1. Розподіл пам’яті програм (Flash) і даних (SRAM) пристрою.



Малюнок 2. Пропозиція по організації конфігурації, налаштувань і вихідних даних логіки пристрою

2.Таблиця конфігурації

*(Об'явлена структура у файлі **type_definition.h**)*

```
typedef struct
{
  uint32_t device_id;                 //Тип пристрою

  uint32_t n_input;                   //Кількість дискретних входів
  uint32_t n_output;                  //Кількість дискретних виходів
  uint32_t n_led;                     //Кількість дискретних світлоіндикаторів
  uint32_t n_button;                  //Кількість ФК

  uint32_t n_alarm;                   //Кількість блоків сигналізацій
  uint32_t n_group_alarm;             //Контроль приростів струмів
  uint32_t n_and;                     //Кількість елементів "І"
  uint32_t n_or;                      //Кількість елементів "АБО"
  uint32_t n_xor;                     //Кількість елементів "Викл.АБО"
  uint32_t n_not;                     //Кількість елементів "НЕ"
  uint32_t n_timer;                    //Кількість таймерів
  uint32_t n_trigger;                 //Кількість триґерів

  uint32_t n_meandes                  //Кількість генераторів меандру

  uint32_t n_tu;                      //Кількість ТУ


  uint8_t time_config[7+1];       //Час останніх змін уставок-витримок-управління
                              //Останній байт масиву сигналізує мітку звідки зміни були проведені
                              //0 - мінімальні параметри
                              //1 - клавіатура
                              //2 - USB
                              //3 - RS-485

} __CONFIG;
```

3.Формат поля PARAM для визначення, який сигнал заводиться на вхід(входи) функціонального блоку

param - 4-байте слово беззнакове слово(uint32_t)

MSB LSB

4-й байт	3-й байт	2-й байт	1-й байт
ID-функціонального блоку (нумерація іде натуральними числами)	Порядковий номер функціонального блоку (нумерація іде натуральними числами)	Порядковий номер виходу функціонального блоку (нумерація іде натуральними числами)	

*(Константи об'явлені у **const_settings.h**)*

```
#define SFIFT_PARAM_ID          24
#define SFIFT_PARAM_N           16
#define SFIFT_PARAM_OUT          0

#define MASKA_PARAM_ID          ((1 << (8*sizeof(uint32_t) - SFIFT_PARAM_ID )) - 1)
#define MASKA_PARAM_N           ((1 << (SFIFT_PARAM_ID     - SFIFT_PARAM_N  )) - 1)
#define MASKA_PARAM_OUT         ((1 << (SFIFT_PARAM_N      - SFIFT_PARAM_OUT)) - 1)
```

4. Константи і масив вказівників на налаштувань у динамічній пам’яті

*(Перечислення об’явлено у **type_definition.h**)*

```
typedef enum _id_fb
{
  _ID_FB_FIRST_ALL = 1,                                                    /*1*/

    _ID_FB_FIRST_FIX = _ID_FB_FIRST_ALL,                                   /*1*/

     ID_FB_CONTROL_BLOCK = _ID_FB_FIRST_FIX,                               /*1*/

    _ID_FB_LAST_FIX,                                                       /*2*/

    _ID_FB_FIRST_VAR = _ID_FB_LAST_FIX,                                    /*2*/

     _ID_FB_FIRST_VAR_NONE_CHANGED = _ID_FB_FIRST_VAR,                     /*2*/

       ID_FB_INPUT = _ID_FB_FIRST_VAR_NONE_CHANGED,                        /*2*/
       ID_FB_OUTPUT,                                                       /*3*/
       ID_FB_LED,                                                          /*4*/
       ID_FB_BUTTON,                                                       /*5*/

     _ID_FB_LAST_VAR_NONE_CHANGED,                                         /*6*/

     _ID_FB_FIRST_VAR_CHANGED = _ID_FB_LAST_VAR_NONE_CHANGED,              /*6*/

       ID_FB_ALARM = _ID_FB_FIRST_VAR_CHANGED,                             /*6*/
       ID_FB_GROUP_ALARM,                                                  /*7*/
       ID_FB_AND,                                                          /*8*/
       ID_FB_OR,                                                           /*9*/
       ID_FB_XOR,                                                          /*10*/
       ID_FB_NOT,                                                          /*11*/

       ID_FB_TIMER,                                                        /*12*/
       ID_FB_TRIGGER,                                                      /*13*/

       ID_FB_MEANDER,                                                      /*14*/
       ID_FB_TU,                                                           /*15*/

     _ID_FB_LAST_VAR_CHANGED,                                              /*16*/

    _ID_FB_LAST_VAR = _ID_FB_LAST_VAR_CHANGED,                             /*16*/

  _ID_FB_LAST_ALL = _ID_FB_LAST_VAR                                        /*16*/
} __id_fb;
```

*(Макроси об’явлені у **const_settings.h**)*

```
#define NUMBER_FIX_BLOCKS       (_ID_FB_LAST_FIX - _ID_FB_FIRST_FIX)

#define NUMBER_VAR_BLOCKS_NONE_CHANGED  (_ID_FB_LAST_VAR_NONE_CHANGED -
_ID_FB_FIRST_VAR_NONE_CHANGED)
#define NUMBER_VAR_BLOCKS_CHANGED       (_ID_FB_LAST_VAR_CHANGED     -
_ID_FB_FIRST_VAR_CHANGED     )
```

#define NUMBER_VAR_BLOCKS (NUMBER_VAR_BLOCKS_NONE_CHANGED + NUMBER_VAR_BLOCKS_CHANGED)
#define NUMBER_ALL_BLOCKS (NUMBER_FIX_BLOCKS + NUMBER_VAR_BLOCKS)

*(Масив об'явлено у **variables_global.h** і **variables_external.h**)*
uintptr_t *spca_of_p_prt [NUMBER_VAR_BLOCKS];

5. Макроси

*(Макроси визначені у **macroses.h**)*

```
#define DIV_TO_HIGHER(_N, _M) ((_N / _M) + ((_N % _M) != 0))
```

6. Таблиця налаштувань, розміри яких відомо на момент компіляції

*(Перечислення об'явлено у **const_settings.h**)*

```
enum _FIX_BLOCK_output_signals
{
 FIX_BLOCK_DEFECT = 0,                  /*Загальна несправність*/
 FIX_BLOCK_AVAR_DEFECT,                 /*Аварійна несправність*/
 FIX_BLOCK_SETTINGS_LOG_WORK,           /*Робота Журналу подій*/
 FIX_BLOCK_SETTINGS_CHANGED,            /*Зміна налаштувань*/

 FIX_BLOCK_SIGNALS_OUT                  /*Кількість вихідних сигналі загального блоку*/
};

enum _FIX_BLOCK_input_signals
{
 FIX_BLOCK_ALARM = 0,                   /*Тривога*/
 FIX_BLOCK_MUTE,                        /*Тиша*/
 FIX_BLOCK_BLOCK,                       /*Блокування*/

 FIX_BLOCK_SIGNALS_IN                   /*Кількість вхідних сигналі загального блоку*/
};
```

*(Масив об'явлено у **variables_global.h** і прототип у **variables_external.h**)*

```
uint8_t fix_block_active_state[DIV_TO_HIGHER(FIX_BLOCK_OUT, 8)]; /*Активні функції*/
uint8_t fix_block_trigger_state[DIV_TO_HIGHER(FIX_BLOCK_OUT, 8)]; /*Спрацьовані функції*/
```

*(Об'явлена структура у файлі **type_definition.h**)*

```
typedef struct

{

 uint32_t param[FIX_BLOCK_SIGNALS_IN];          //Сигнали на входах


 uint32_t password_1;//Пароль для проглядання з меню

 uint32_t password_2;//Пароль для редагування з меню

 uint32_t timeout_deactivation_password_interface_USB; //Час деактивації паролю для редагування з інтерфейсу USB

 uint32_t password_interface_USB; //Пароль для редагування з інтерфейсу USB

 uint32_t timeout_deactivation_password_interface_RS485; //Час деактивації паролю для редагування з інтерфейсу RS485

 uint32_t password_interface_RS485; //Пароль для редагування з інтерфейсу RS485


 uint32_t timeout_idle_new_settings;


 //Комунікація

 uint8_t name_of_cell[MAX_CHAR_IN_NAME_OF_CELL];//І'мя ячейки
```

```
  uint16_t user_register[(M_ADDRESS_LAST_USER_REGISTER_DATA -
M_ADDRESS_FIRST_USER_REGISTER_DATA) + 1]; //Регістри користувача
  uint32_t address; //Адреса
  int32_t baud_RS485;//швидкість обміну
                                          // 0 - 9600
                                          // 1 - 14400
                                          // 2 - 19200
                                          // 3 - 28800
                                          // 4 - 38400
                                          // 5 - 57600
                                          // 6 - 115200
  int32_t pare_bit_RS485; //паритет
                                          // 0 - NONE
                                          // 1 - ODD
                                          // 2 - EVEN
  int32_t number_stop_bit_RS485; //кількість стоп-біт
                                          // 0 - 1 stop-bit
                                          // 1 - 2 stop-bits
  uint32_t time_out_1_RS485; //time-out наступного символу = X/10 символу

  int32_t language;  //мова меню  0= змінна мов не підтримується; 1=RU; 2=UA; 3=EN; 4=KZ; 5=др.

  unsigned char time_setpoints[7+1]; //Час останніх змін уставок-витримок-управління
                                     //Останній байт масиву сигналізує мітку звідки зміни були проведені
                                          //0 - мінімальні параметри
                                          //1 - клавіатура
                                          //2 - USB
                                          //3 - RS-485

} __SETTINGS_FIX;
```

7. Структури для дискретних входів

*(Перечислення об'явлено у **const_settings.h**)*

```
enum _settings_delay_of_INPUT
{
 INPUT_SET_DELAY_DOPUSK = 0,      /* Допуск ДВ */

 INPUT_SET_DELAYS                 /*Кількість налаштовуваних витримок*/

};

enum _work_delay_of_INPUT
{
 INPUT_WORK_DELAY_DOPUSK = 0,  /*Пот.час роботи таймера допуску*/

 INPUT_WORK_DELAYS                /*Кількість робочих таймерів*/
};

enum _INPUT_output_signals
{
 INPUT_OUT = 0,                   /*Вихід*/

 INPUT_SIGNALS_OUT                /*Кількість вихідних сигналів*/

};

enum __index_ctrl_input /*перечислення для управління*/
{
 INDEX_CTRL_INPUT_TYPE_SIGNAL = 0, /*0 – Постійний сигнал; 1 – Змінний сигнал*/

 MAX_INDEX_CTRL_INPUT                  /*Кількість елементів управління*/
};
```

*(Типи об'явлено у **type_definition.h**)*

Структура витримок/таймерів
Структура налаштувань

```
typedef struct
{

 int32_t delay[INPUT_SET_DELAYS];                    /*Таймери*/
 uint32_t control;                                   /*Управління*/

} __settings_for_INPUT;
```

Структура функціонального елементу

```
typedef struct
{
 __settings_for_INPUT settings;                      /*Налаштування*/

 int32_t delay[INPUT_WORK_DELAYS];
 uint8_t active_state[DIV_TO_HIGHER (INPUT_SIGNALS_OUT, 8)];        /*Активні сигнали*/
 uint8_t trigger_state[DIV_TO_HIGHER(INPUT_SIGNALS_OUT, 8)];./*Спрацьовані (тригерні
сигнали)*/
} __LN_INPUT;
```

8.Структури для дискретного виходу і світлоіндикатора

*(Перечислення об'явлено у **const_settings.h**)*

```
enum _OUTPUT_LED_output_signals
{
 OUTPUT_LED_OUT = 0,  /*Логічний вихід реле/св., без врахування імпульсного режиму*/
 OUTPUT_LED_SIGNALS_OUT, /*Кількість вихідних сигналів*/

 OUTPUT_LED_BOARD = OUTPUT_LED_SIGNALS_OUT, /*Вихід реле/св., який іде на фізичне реле/св.*/


 OUTPUT_LED_SIGNALS_OUT_TOTAL /*Загальна кількість виходів*/
};
enum _OUTPUT_LED_nput_signals
{
 OUTPUT_LED_LOGIC_INPUT = 0,       /*Логічний вхід*/
 OUTPUT_LED_RESET,                 /*Вхід сигналу Reset*/

 OUTPUT_LED_SIGNALS_IN      /*Кількість входів, які використовуються пари побудові зв'язків*/

 OUTPUT_LED_MEANDER1 = OUTPUT_LED_SIGNALS_IN/*Вхід основного ГПС*/
 OUTPUT_LED_MEANDER2,              /*Вхід допоміжного ГПС*/

 OUTPUT_LED_SIGNALS_IN_TOTAL /*Загальна кількість входів*/
};

enum __index_ctrl_output_led /*перечислення для управління*/
{
 INDEX_CTRL_OUTPUT_LED_N_T = 0,          /*0 – Нормальний; 1 – Триґерний*/
 INDEX_CTRL_OUTPUT_LED_C_I,              /*0 – Постійний; 1 – Імпульсний*/
 INDEX_CTRL_OUTPUT_LED_SI_EI,    /*0 – Імп. простий; 1 – Імп. розширений*/

/*Решта елементів перечислення не потрібні для системи обробки логіки елементу*/

_MAX_INDEX_CTRL_OUTPUT_LED_BITS_SETTINGS,

 INDEX_CTRL_OUTPUT_LED_MEANDER1 =
_MAX_ROW_CTRL_OUTPUT_LED_BITS_SETTINGS,
 INDEX_CTRL_OUTPUT_LED_MEANDER2,

 MAX_INDEX_CTRL_OUTPUT_LED
};
```

*(Типи об'явлено у **type_definition.h**)*

Структура налаштувань

```
typedef struct
{

 uint32_t control;                                  /*Управління*/
 uint32_t param[OUTPUT_SIGNALS_IN];                 /*Сигнали на входах*/

} __settings_for_OUTPUT_LED;
```

Структура функціонального елементу

```
typedef struct
{
  __settings_for_OUTPUT_LED settings;                                   /*Налаштування*/

  uint8_t active_state[DIV_TO_HIGHER (OUTPUT_SIGNALS_OUT, 8)];        /*Активні сигнали*/
  uint8_t trigger_state[DIV_TO_HIGHER(OUTPUT_SIGNALS_OUT, 8)];/*Спрацьовані (триґерні
сигнали)*/
} __LN_OUTPUT_LED;
```

9.Структури для елементу стандартної логіки «І»

(Перечислення об’явлено у **const_settings.h**)

```
enum _STANDARD_LOGIC_output_signals
{
  STANDARD_LOGIC_OUT = 0,              /*Логічний вихід*/

  STANDARD_LOGIC_SIGNALS_OUT           /*Кількість вихідних сигналів*/

};
```

(Константу об’явлено у **const_settings.h**)

```
#define AND_SIGNALS_IN                 8
```

(Типи об’явлено у **type_definition.h**)

Структура налаштувань

```
typedef struct
{

  uint32_t param[AND_SIGNALS_IN];                    /*Сигнали на входах*/

} __settings_for_AND;
```

Структура функціонального елементу

```
typedef struct
{
  __settings_for_AND settings;                       /*Налаштування*/

  uint8_t active_state[DIV_TO_HIGHER(STANDARD_LOGIC_SIGNALS_OUT, 8)]; /*Активні сигнали*/
  uint8_t trigger_state[DIV_TO_HIGHER(STANDARD_LOGIC_SIGNALS_OUT, 8)]; /*Спрацьовані (тригерні сигнали)*/

} __LN_AND;
```

10. Структури для елементу стандартної логіки «АБО»

(Константу об’явлено у **const_settings.h**)
#**define** OR_SIGNALS_IN 8

(Типи об’явлено у **type_definition.h**)
Структура налаштувань
typedef struct
{

uint32_t param[NUMBER_IN_OR]; /*Сигнали на входах*/

} __settings_for_OR;

Структура функціонального елементу
typedef struct
{

__settings_for_OR settings; /*Налаштування*/

uint8_t active_state[DIV_TO_HIGHER(STANDARD_LOGIC_SIGNALS_OUT, 8)]; /*Активні сигнали*/
uint8_t trigger_state[DIV_TO_HIGHER(STANDARD_LOGIC_SIGNALS_OUT, 8)];/*Спрацьовані (тригерні сигнали)*/

} __LN_OR;

11. Структури для елементу стандартної логіки «Викл.АБО»

(Типи об’явлено у **type_definition.h**)
Структура налаштувань

```
typedef struct
{

  uint32_t param[2];                                    /*Сигнали на входах*/

} __settings_for_XOR;
```

Структура функціонального елементу

```
typedef struct
{
  __settings_for_XOR settings;                          /*Налаштування*/

  uint8_t active_state[DIV_TO_HIGHER(STANDARD_LOGIC_SIGNALS_OUT, 8)]; /*Активні сигнали*/
  uint8_t trigger_state[DIV_TO_HIGHER(STANDARD_LOGIC_SIGNALS_OUT, 8)]; /*Спрацьовані (тригерні сигнали)*/

} __LN_XOR;
```

12. Структури для елементу стандартної логіки «НЕ»

(Типи об’явлено у **type_definition.h**)
Структура налаштувань
typedef struct
{

  uint32_t param[1]; /*Сигнали на входах*/

} __settings_for_NOT;

Структура функціонального елементу
typedef struct
{

  __settings_for_NOT settings; /*Налаштування*/

  uint8_t active_state[DIV_TO_HIGHER(STANDARD_LOGIC_SIGNALS_OUT, 8)]; /*Активні сигнали*/
  uint8_t trigger_state[DIV_TO_HIGHER(STANDARD_LOGIC_SIGNALS_OUT, 8)]; /*Спрацьовані (триґерні сигнали)*/

} __LN_NOT;

13. Структури для елементу стандартної логіки «БФ-Таймер»

(Перечислення об’явлено у **const_settings.h**)

```
enum _settings_delay_of_TIMER
{
 TIMER_SET_DELAY_PAUSE = 0,          /*Таймер паузи*/
 TIMER_SET_DELAY_WORK,               /*Таймер роботи*/

 TIMER_SET_DELAYS                    /*Кількість налаштовуваних витримок*/
};

enum _work_delay_of_TIMER
{
 TIMER_WORK_DELAY_PAUSE_RISE = 0,    /*Пот.час роботи таймера паузи по нарост.фронту*/
 TIMER_WORK_DELAY_PAUSE_FALL,        /*Пот.час роботи таймера паузи по спадн.фронту*/
 TIMER_WORK_DELAY_WORK_IMPULSE_RISE, /*Пот.час роботи таймера роботи для імпульсу
по нарост.фронту*/
 TIMER_WORK_DELAY_WORK_DELAY_RISE, /*Пот.час роботи таймера роботи для затримки по
нарост.фронту*/
 TIMER_WORK_DELAY_WORK_IMPULSE_FALL, /*Пот.час роботи таймера роботи для імпульсу
по спадн.фронту*/

 TIMER_WORK_DELAYS/*Кількість робочих таймерів для вибрано таймера*/
};

enum _TIMER_output_signals
{
 TIMER_OUT_RISE_IMPULSE = 0,         /*Вихід 1*/
 TIMER_OUT_RISE_DELAY,               /*Вихід 2*/
 TIMER_OUT_FALL_IMPULSE,             /*Вихід 3*/

 TIMER_SIGNALS_OUT                   /*Кількість вихідних сигналів*/

};
enum _TIMER_input_signals
{
 TIMER_LOGIC_INPUT = 0,          /*Логічний вхід*/
 TIMER_RESET,                    /*Вхід сигналу Reset*/

 TIMER_SIGNALS_IN                /*Кількість входів*/
};
```

(Типи об’явлено у **type_definition.h**)

Структура витримок/таймерів

Структура налаштувань

```
typedef struct
{

 int32_t set_delay[TIMER_SET_DELAYS];          /*Витримки*/
 uint32_t param [TIMER_SIGNALS_IN];            /*Сигнали на входах*/

} __settings_for_TIMER;
```

Структура функціонального елементу

```
typedef struct
{

  __settings_for_TIMER settings;                                    /*Налаштування*/

  int32_t work_delay[TIMER_WORK_DELAYS];                    /*Таймери*/
  uint8_t active_state[DIV_TO_HIGHER(TIMER_SIGNALS_OUT, 8)]; /*Активні сигнали*/
  uint8_t trigger_state[DIV_TO_HIGHER(TIMER_SIGNALS_OUT, 8)];/*Спрацьовані (тригерні
сигнали)*/

} __LN_TIMER;
```

14. Структури для елементу стандартної логіки «Триґер»

(Перечислення об’явлено у **const_settings.h**)

```
enum _TRIGGER_output_signals
{
  TRIGGER_OUT = 0,                /*Логічний вихід (прямий)*/
  TRIGGER_OUT_INV,                /*Логічний вихід (інверсний)*/

  TRIGGER_SIGNALS_OUT                   /*Кількість вихідних сигналів*/

};
enum _TRIGGER_input_signals
{
  INPUT_TRIGGER_SET = 0,                /*Встановлення прямого виходу триґера*/
  INPUT_TRIGGER_RESET,                  /*Скидання прямого виходу триґерва*/
  INPUT_TRIGGER_D,                      /*D-вхід*/
  INPUT_TRIGGER_C,                      /*Синхро-вхід*/

  TRIGGER_SIGNALS_IN            /*Кількість входів*/

};
```

(Типи об’явлено у **type_definition.h**)

Структура налаштувань

```
typedef struct
{

  uint32_t param[TRIGGER_SIGNALS_IN];                 /*Сигнали на входах*/


} __settings_for_TRIGGER;
```

Структура функціонального елементу

```
typedef struct
{

  __settings_for_TRIGGER settings;                                  /*Налаштування*/

  uint8_t active_state[DIV_TO_HIGHER(TRIGGER_SIGNALS_OUT, 8)];      /*Активні сигнали*/
  uint8_t trigger_state[DIV_TO_HIGHER(TRIGGER_SIGNALS_OUT, 8)];     /*Спрацьовані (триґерні
сигнали)*/

} __LN_TRIGGER;
```

15. Структури для елементу функціонального блоку «Генератор періодичних сигналів»

(Перечислення об’явлено у **const_settings.h**)

```
enum _settings_delay_of_MEANDER
{
  MEANDER_SET_DELAY_PERIOD = 0,

  MEANDER_SET_DELAYS
};

enum _work_delay_of_MEANDER
{
  MEANDER_WORK_DELAY_PERIOD = 0,

  MEANDER_WORK_DELAYS
};

enum _MEANDER_output_signals
{
  MEANDER_OUT = 0,                    /*Логічний вихід*/

  MEANDER_SIGNALS_OUT                 /*Кількість вихідних сигналів*/
};
```

(Типи об’явлено у **type_definition.h**)
Структура витримок/таймерів
Структура налаштувань

```
typedef struct
{

  int32_t delay [MEANDER_SET_DELAYS];          /*Витримки*/

} __settings_for_MEANDER;
```

Структура функціонального елементу

```
typedef struct
{

  __settings_for_MEANDER settings;                    /*Налаштування*/

  int32_t delay [MEANDER_WORK_DELAYS];                          /*Таймери*/
  uint8_t active_state[DIV_TO_HIGHER(MEANDER_SIGNALS_OUT, 8)];  /*Активні сигнали*/
} __LN_MEANDER;
```

16. Структури для елементу функціонального блоку «Світлозвукова сигналізація»

*(Перечислення об'явлено у **const_settings.h**)*

```
enum _settings_delay_of_ALARM
{
 ALARM_SET_DELAY_PERIOD = 0,  /*Витримка СЗС для режиму «На заданий час»*/

 ALARM_SET_DELAYS                /*Кількість витримок*/
};

enum _work_delay_of_ALARM
{
 ALARM_WORK_DELAY_PERIOD = 0, /*Пот.час роботи таймера СЗС у режимі «На заданий час»*/

 ALARM_WORK_DELAYS               /*Кількість робочих таймерів*/
};

enum _ALARM_output_signals
{
 ALARM_OUT_ALARM = 0,            /*Вихід: Тривога*/
 ALARM_OUT_MUTE,                 /* Вихід: Тиша*/
 ALARM_OUT_BLOCK,                /* Вихід: Блокування*/

 ALARM_SIGNALS_OUT               /*Кількість вихідних сигналів*/

};
enum _ALARM_input_signals
{
 ALARM_LOGIC_INPUT = 0,          /* Вхід: Інформаційний сигнал*/
 ALARM_IN_MUTE,                  /* Вхід: Тиша*/
 ALARM_IN_BLOCK,                 /* Вхід: Блокування*/
 ALARM_RESET,                    /* Вхід: Скидання*/

 ALARM_SIGNALS_IN                /*Кількість вхідних сигналів*/
};

enum __index_ctrl_alarm
{
 _MAX_INDEX_CTRL_ALARM_BITS_SETTINGS = 0,

 INDEX_CTRL_ALARM_MODE = _MAX_INDEX_CTRL_ALARM_BITS_SETTINGS,

 MAX_INDEX_CTRL_ALARM
};

enum _ALARM_modes
{
 ALARM_MODE_SIMPLE = 0,          /*Простий*/
 ALARM_MODE_TRIGGER,             /*Тригерний*/
 ALARM_MODE_PERIOD,              /*На заданий час*/

 ALARM_MODES_NUMBER              /*Кількість режимів*/
```

```
};
```

(Масив об'явлено у ***variables_global.h*** *і* ***variables_external.h****)*

```
const uint32_t alarm_ctrl_patten[MAX_INDEX_CTRL_ALARM][2] = {INDEX_CTRL_ALARM_MODE, 2};/*масив (початкове зміщення, кількість біт) для визначення у слові управління (control) структури налаштувань потрібної порції інформації*/
```

(Типи об'явлено у ***type_definition.h****)*

Структура налаштувань

```
typedef struct
{
  int32_t set_delay[ALARM_SET_DELAYS];          /*Витримки*/
  uint32_t control;                             /*Управління*/
  uint32_t param[ALARM_SIGNALS_IN];             /*Сигнали на входах*/

} __settings_for_ALARM;
```

Структура функціонального елементу

```
typedef struct
{
  __settings_for_ALARM settings;                        /*Налаштування*/

  int32_t work_delay[ALARM_WORK_DELAYS];                /*Робочі таймери*/
  uint8_t active_state[DIV_TO_HIGHER(ALARM_SIGNALS_OUT, 8)];    /*Активні сигнали*/
  uint8_t trigger_state[DIV_TO_HIGHER(ALARM_SIGNALS_OUT, 8)];   /*Спрацьовані сигнали*/

} __LN_ALARM;
```

17. Структури для елементу функціонального блоку «Шинки групової сигналізації»

*(Перечислення об'явлено у **const_settings.h**)*

```
enum _settings_pickup_of_GROUP_ALARM
{
 GROUP_ALARM_PICKUP_DELTA_I = 0,          /*ΔI*/

 GROUP_ALARM_PICKUPS                      /*Кількість уставок*/
};

enum _settings_delay_of_GROUP_ALARM
{
 GROUP_ALARM_SET_DELAY_DELAY = 0,         /*T затримки*/

 GROUP_ALARM_SET_DELAYS                   /*Кількість витримок*/
};

enum _work_delay_of_GROUP_ALARM
{
 GROUP_ALARM_WORK_DELAY_DELAY = 0,        /*Поточний час для витримки «T затримки»*/

 GROUP_ALARM_WORK_DELAYS                  /*Кількість таймерів*/
};

enum _GROUP_ALARM_output_signals
{
 GROUP_ALARM_OUT_NNP = 0,          /*Фіксація збільшення кількості спрацювань*/
 GROUP_ALARM_OUT_NNM,              /*Фіксація зменшення кількості спрацювань*/
 GROUP_ALARM_OUT_CC,               /*Наявність спрацювань*/
 GROUP_ALARM_OUT_CE,               /*Не проходить контроль даної ШГС*/
 GROUP_ALARM_OUT_OC,               /*Струм перевищує граничну норму*/

 GROUP_ALARM_SIGNALS_OUT           /*Кількість вихідних сигналів*/

};


enum __index_ctrl_group_alarm
{
 INDEX_CTRL_GROUP_ALARM_STATE = 0,        /*Стан ШГС: 0-Вимк. 1-Ввімк.*/
 INDEX_CTRL_GROUP_ALARM_CTRL_STATE,       /*Стан контролю ШГС: 0-Вимк. 1-Ввімк.*/

_MAX_INDEX_CTRL_GROUP_ALARM_BITS_SETTINGS, /*Кількість налаштувань (дискретних)*/

 INDEX_CTRL_GROUP_ALARM_I = _MAX_INDEX_CTRL_GROUP_ALARM_BITS_SETTINGS,
 /*Вхідний струм*/
 MAX_INDEX_CTRL_GROUP_ALARM        /*Кількість налаштувань (дискретних і аналогових)*/
};
```

*(Масив об'явлено у **variables_global.h** і **variables_external.h**)*

```
const uint32_t group_alarm_analog_ctrl_patten[MAX_INDEX_CTRL_GROUP_ALARM -
_MAX_INDEX_CTRL_GROUP_ALARM_BITS_SETTINGS][2] = {
```

```
 {0, 8}
};
```
/*масив (початкове зміщення, кількість біт) для визначення у слові управління (control) структури налаштувань потрібної порції інформації*/

*(Типи об'явлено у **type_definition.h**)*

Структура налаштувань

```
typedef struct
{
  int32_t pickup[GROUP_ALARM_PICKUPS];                  /*Уставки*/
  int32_t set_delay[GROUP_ALARM_SET_DELAYS];            /*Витримки*/
  uint32_t control;                                     /*Управління*/
  uint32_t analog_input_control;                        /*Управлыння для вхыдних струмыв*/

} __settings_for_GROUP_ALARM;
```

Структура функціонального елементу

```
typedef struct
{
  __settings_for_GROUP_ALARM settings;                  /*Налаштування*/

  int32_t work_delay[GROUP_ALARM_WORK_DELAYS];      /*Робочі таймери*/
  uint8_t active_state[DIV_TO_HIGHER(GROUP_ALARM_SIGNALS_OUT, 8)];      /*Активні функції*/
  uint8_t trigger_state[DIV_TO_HIGHER(GROUP_ALARM_SIGNALS_OUT, 8)]; /*Спрацьовані функції*/
  uint32_t NNC; /*New number of Curcuit*/ /*Кількість кіл*/

} __LN_GROUP_ALARM;
```

18.Структури для Функціональних кнопок і команд Телеуправління

*(Перечислення об'явлено у **const_settings.h**)*

```
enum _BUTTON_TU_output_signals
{
  BUTTON_TU_OUT = 0,                    /*Логічний вихід*/

  BUTTON_TU_SIGNALS_OUT,    /          *Кількість вихідних сигналів*/
};
```

*(Типи об'явлено у **type_definition.h**)*

Структура налаштувань

```
typedef struct
{

  uint32_t control;                                    /*Управління*/
  uint32_t param[OUTPUT_SIGNALS_IN];                   /*Сигнали на входах*/

} __settings_for_OUTPUT_LED;
```

Структура функціонального елементу

```
typedef struct
{
  uint8_t active_state[DIV_TO_HIGHER(BUTTON_TU_SIGNALS_OUT, 8)];
  uint8_t trigger_state[DIV_TO_HIGHER(BUTTON_TU_SIGNALS_OUT, 8)];

} __LN_BUTTON_TU;
```

Моя пропозиція (дискусійна!) щодо алгоритму опрацювання логіки всіх функціональних блоків у конкретний часовий відрізок

Умова завершення опрацьовування всіх функціональних блоків
- опрацьовано всі функціональні блоки
- вхідні сигнали між I-тою ітерацією і (I - 1)-тою ітерацією для всіх функціональних блоків не змінилися
- максимальні кількість ітерацій не перевищує критично встановлене значення

У випадку, якщо кількість ітерацій перевищила максимально-допустиму кількість ітерацій у конкретний часовий відрізок, то встановлюється сигнал «Помилка параметрування внутрішніх зв'язків».

Другою умовою дострокового припинення пошуку стаціонарного стану є випадок, якщо стан для будь-яких J-ої ітерації і K-тої ітерації **ВСІ** стани сигналів повторилися. У такому випадку ітераційний процес також завершується і виставляється сигнал «Помилка параметрування внутрішніх зв'язків»

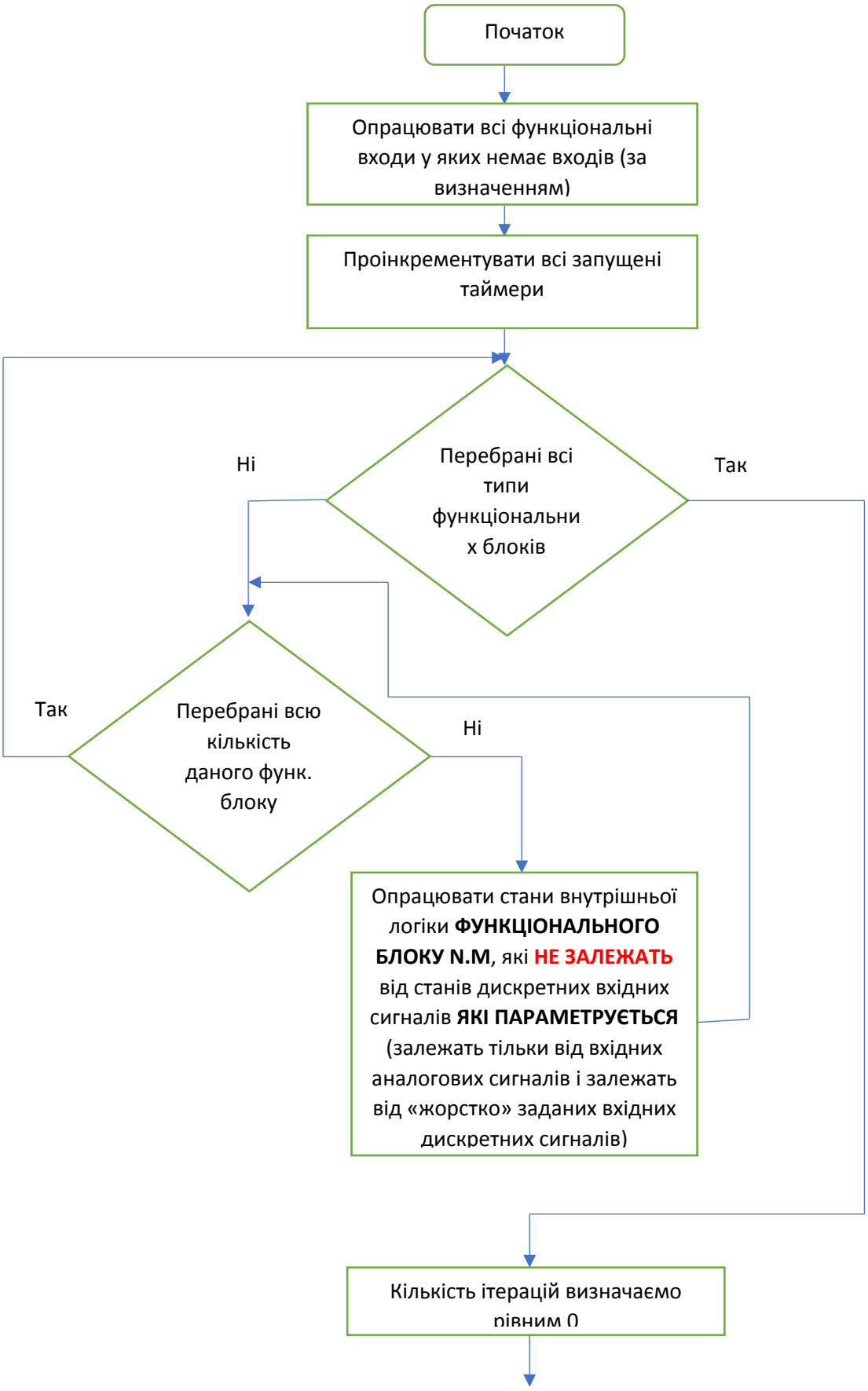
Для реалізації визначення того, що існує, а бо не існує два проміжні стани у перехідному процесі, коли всі сигнали повторюються, а також для фіксації ситуації що встановився стаціонарний стан (між двома останніми ітераціями не відбувалося зміни станів) у ДИНАМІЧНІЙ пам'яті виділяється масив розміром «МАКСИМАЛЬНА КІЛЬКІСТЬ ІТЕРАЦІЙ»х«КІЛЬКІСТЬ 32-БІТНИХ СЛІВ ДЛЯ ВСІХ СИГНАЛІВ ВИБАНОЇ КОНФІГУРАЦІЇ[1]»

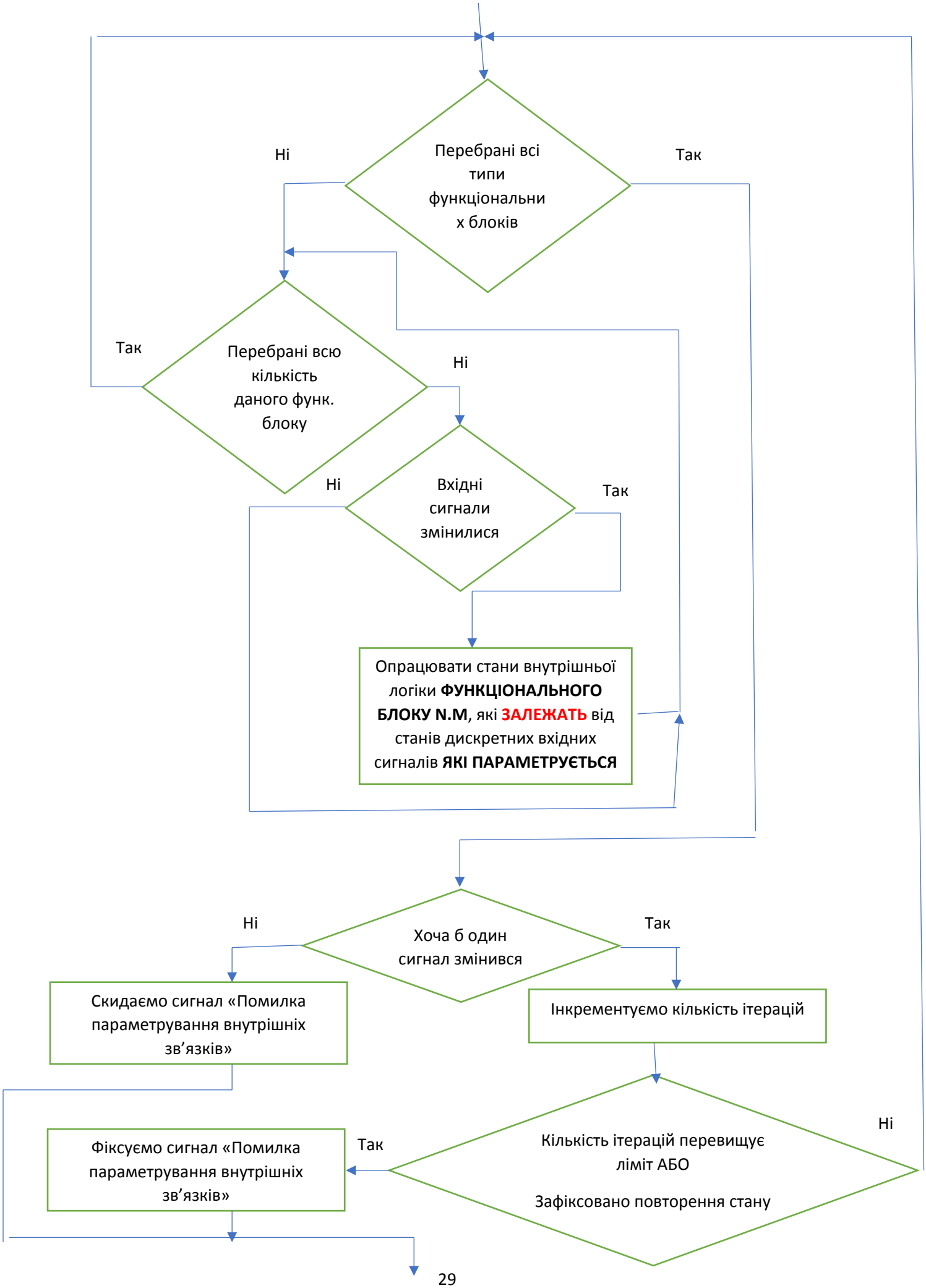
N – Максимальна кількість ітерацій
M – максимальна кількість 32-бітних слів для всіх сигналів вибраної конфігурації

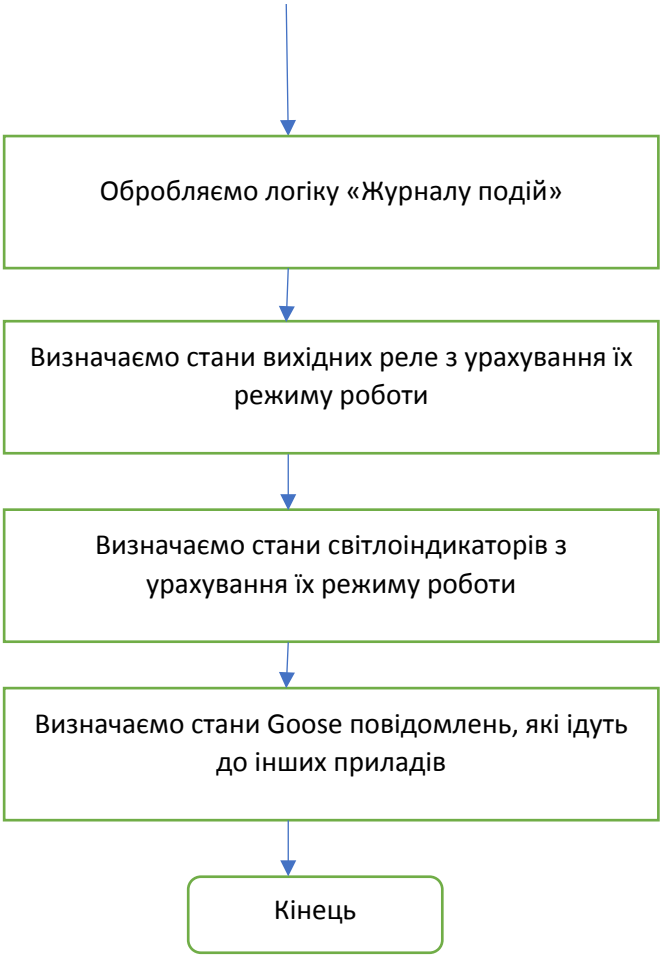
	LSB			HSB
0 ітерація	0 - слово	1 - слово	…	(M – 1) - слово
1 ітерація	0 - слово	1 - слово	…	(M – 1) - слово
		…		
(I - 1) ітерація	0 - слово	1 - слово	…	(M – 1) - слово
I ітерація	0 - слово	1 - слово	…	(M – 1) - слово
(I + 1) ітерація	0 - слово	1 - слово	…	(M – 1) - слово
		…		
N ітерація	0 - слово	1 - слово	…	(M – 1) - слово

[1] Оскільки загальна кількість сигналів залежить від вибраної конфігурації, то цей масив буде мати розмір залежний від вибору конфігурації, а значить буде розміщений у динамічний пам'яті. Вказівник на нього буде встановлюватися при старті системи і змінюватися при зміні конфігурації. Так само у визначеній змінній буде визначено при старті системи і перевизначено після зміни конфігурації величина «кількість 32-бітних слів для всіх сигналів вибраної конфігурації».

Блок схема обробки всіх функціональних блоків




Перебрані всі типи функціональни х блоків
Ні
Так
Перебрані всю кількість даного функ. блоку
Так
Ні
Вхідні сигнали змінилися
Ні
Так
Опрацювати стани внутрішньої логіки **ФУНКЦІОНАЛЬНОГО БЛОКУ N.M**, які **ЗАЛЕЖАТЬ** від станів дискретних вхідних сигналів **ЯКІ ПАРАМЕТРУЄТЬСЯ**
Хоча б один сигнал змінився
Ні
Так
Скидаємо сигнал «Помилка параметрування внутрішніх зв’язків»
Інкрементуємо кількість ітерацій
Фіксуємо сигнал «Помилка параметрування внутрішніх зв’язків»
Так
Кількість ітерацій перевищує ліміт АБО
Зафіксовано повторення стану
Ні


Обробляємо логіку «Журналу подій»
Визначаємо стани вихідних реле з урахування їх режиму роботи
Визначаємо стани світлоіндикаторів з урахування їх режиму роботи
Визначаємо стани Goose повідомлень, які ідуть до інших приладів
Кінець