Подробный алгоритм функций LACH_CONTROL() и LACH_CONTROL_DU()

Общее назначение функций

Функции `LACH_CONTROL()` и `LACH_CONTROL_DU()` выполняют управление реле системы через импульсное воздействие на аппаратные выходы. Они реализуют логику включения/отключения реле с защитой от одновременного нажатия кнопок и формированием управляющих импульсов.

Различия между функциями

LACH_CONTROL() - Локальный режим управления

- **Назначение**: Управление реле в локальном режиме (flags.F_DU == 0)
- **Особенности**:
  - Проверяет физическое состояние кнопок через RD4
  - Защита от одновременного нажатия обеих кнопок
  - Полная обработка состояний кнопок

LACH_CONTROL_DU() - Дистанционный режим управления

- **Назначение**: Управление реле в дистанционном режиме (flags.F_DU == 1)
- **Особенности**:
  - НЕ проверяет физическое состояние кнопок
  - НЕ имеет защиты от одновременного нажатия
  - Работает только с программными флагами

Карта используемых сигналов

Входные сигналы (RD4) - только LACH_CONTROL()

```
Бит 5: FS_BKL  - физическая кнопка "Включить реле" (инверсная логика)
Бит 6: FS_OTKL - физическая кнопка "Отключить реле" (инверсная логика)
```

Программные флаги управления

```
flags.F_START1  - команда включения реле
flags.F_START2  - команда отключения реле
flags.FI_BKL    - индикация включенного реле
flags.FI_OTKL   - индикация отключенного реле
flags.FS_BKL    - состояние кнопки включения
flags.FS_OTKL   - состояние кнопки отключения
```

Выходные сигналы (WR2)

```
Бит 5: F_BKL    - импульс включения реле (инверсная логика)
Бит 6: F_OTKL   - импульс отключения реле (инверсная логика)
```

Подробный алгоритм LACH_CONTROL()

Шаг 1: Чтение состояния физических кнопок

```
unsigned char buffer;
buffer = perform_actionR(RD4);
flags.FS_BKL = (buffer & (1 << 5)) ? 0 : 1;   // Инверсная логика: 0=нажата, 1=отпущена
flags.FS_OTKL = (buffer & (1 << 6)) ? 0 : 1;  // Инверсная логика: 0=нажата, 1=отпущена
```

Назначение: Определение текущего состояния физических кнопок управления реле.

Шаг 2: Защита от одновременного нажатия кнопок

```
if (flags.F_START1 && flags.F_START2)    // Обе команды активны одновременно
{
    flags.FS_BKL = 0;                     // Сброс состояния кнопок
    flags.FS_OTKL = 0;
    write_UART2("RELE both buttons pressed \\r\\n");  // Сообщение об ошибке
    delay_ms(400);                        // Задержка для предотвращения повторных срабатываний
    return;                               // Выход без выполнения команд
}
```

Назначение: Предотвращение конфликтной ситуации при одновременном получении команд включения и отключения реле.

Шаг 3: Обработка команды включения реле (F_START1)

3.1 Проверка активности команды

```
if (flags.F_START1)  // Если активна команда включения
```

3.2 Установка состояний индикации

```
buffer = 0x60;        // Базовое значение: 0110 0000 (биты 5,6 = 1, реле в покое)
flags.FI_BKL = 1;     // Включить индикацию "реле включено"
flags.FI_OTKL = 0;    // Отключить индикацию "реле отключено"
```

3.3 Формирование управляющего буфера

```
update_relay(&buffer);              // Добавление состояний питания
TOGGLE_BIT(buffer, F_BKL_MASK);    // Инверсия бита 5 (0x20) для активации импульса
```

3.4 Генерация импульса включения

```
perform_actionW(WR2, buffer);       // Запись буфера с активным импульсом
delay_ms(20);                       // Удержание импульса 20 мс
```

3.5 Завершение импульса и сброс флагов

```
flags.FS_BKL = 0;                   // Сброс состояния кнопки
flags.F_START1 = 0;                 // Сброс команды включения
update_relay(&buffer);              // Обновление состояний питания
TOGGLE_BIT(buffer, F_BKL_MASK);    // Возврат бита 5 в исходное состояние
perform_actionW(WR2, buffer);       // Запись буфера с неактивным импульсом
```

Шаг 4: Обработка команды отключения реле (F_START2)

4.1 Проверка активности команды

```
if (flags.F_START2)  // Если активна команда отключения
```

4.2 Установка состояний индикации

```
buffer = 0x60;        // Базовое значение
flags.FI_BKL = 0;     // Отключить индикацию "реле включено"
flags.FI_OTKL = 1;    // Включить индикацию "реле отключено"
```

4.3 Формирование управляющего буфера

```
update_relay(&buffer);              // Добавление состояний питания
TOGGLE_BIT(buffer, F_OTKL_MASK);   // Инверсия бита 6 (0x40) для активации импульса
```

4.4 Генерация импульса отключения

```
perform_actionW(WR2, buffer);       // Запись буфера с активным импульсом
delay_ms(20);                       // Удержание импульса 20 мс
```

4.5 Завершение импульса и сброс флагов

```
flags.FS_OTKL = 0;                  // Сброс состояния кнопки
flags.F_START2 = 0;                 // Сброс команды отключения
update_relay(&buffer);              // Обновление состояний питания
TOGGLE_BIT(buffer, F_OTKL_MASK);   // Возврат бита 6 в исходное состояние
perform_actionW(WR2, buffer);       // Запись буфера с неактивным импульсом
```

Подробный алгоритм LACH_CONTROL_DU()

Отличия от LACH_CONTROL():

1. **НЕТ чтения физических кнопок** - пропущен Шаг 1
2. **НЕТ защиты от одновременного нажатия** - пропущен Шаг 2
3. **Идентичная логика импульсов** - Шаги 3-4 аналогичны

Упрощенный алгоритм:

Обработка команды включения (F_START1)

```
if (flags.F_START1)
{
    buffer = 0x60;
    flags.FI_BKL = 1;               // Включить индикацию
    flags.FI_OTKL = 0;              // Отключить индикацию

    update_relay(&buffer);
    TOGGLE_BIT(buffer, F_BKL_MASK); // Активация импульса
    perform_actionW(WR2, buffer);

    delay_ms(20);                   // Удержание импульса

    flags.FS_BKL = 0;               // Сброс флагов
    flags.F_START1 = 0;
    update_relay(&buffer);
    TOGGLE_BIT(buffer, F_BKL_MASK); // Деактивация импульса
    perform_actionW(WR2, buffer);
}
```

Обработка команды отключения (F_START2)

```
if (flags.F_START2)
{
    // Аналогично команде включения, но с битом 6 и другими флагами
}
```

Детальный алгоритм функции update_relay()

Назначение

Формирование базового состояния регистра WR2 с учетом состояний питания системы.

Пошаговое выполнение

```
void update_relay(unsigned char *buffer)
{
    // Управление питанием PDP (бит 0)
    UPDATE_BUFFER(*buffer, flags.FPIT_PDP != 1, PITPDP_MASK, PITPDP_OFF_MASK);

    // Управление питанием CKP (бит 1)
    UPDATE_BUFFER(*buffer, flags.FPIT_CKP != 1, PITCKP_MASK, PITCKP_OFF_MASK);

    // Управление питанием блока 6.7 PDP (бит 2)
    UPDATE_BUFFER(*buffer, flags.F6_7_PDP == 0, PITPDP_6_MASK, PITPDP_7_MASK);

    // Индикация питания PDP (бит 7, инверсная логика)
    UPDATE_BUFFER_AND_OR(*buffer, flags.FPIT_PDP != 0, I_PITPDP_MASK, I_PITPDP_OFF_MASK);
}
```

Логика работы импульсов

Принцип импульсного управления

1. **Исходное состояние**: Биты 5,6 = 1 (реле в покое, инверсная логика)
2. **Активация**: TOGGLE_BIT инвертирует нужный бит (1→0, активация)
3. **Удержание**: 20 мс активного состояния
4. **Деактивация**: TOGGLE_BIT возвращает бит в исходное состояние (0→1, покой)

Временные диаграммы

```
Включение реле (бит 5):
Время:     0ms    20ms   40ms
Бит 5:     1  ->  0  ->  1
Импульс:          _______
           ‾‾‾‾‾‾‾       ‾‾‾‾‾‾‾
           покой  актив  покой

Отключение реле (бит 6):
Время:     0ms    20ms   40ms
Бит 6:     1  ->  0  ->  1
Импульс:          _______
           ‾‾‾‾‾‾‾       ‾‾‾‾‾‾‾
           покой  актив  покой
```

Состояния системы

Флаги управления

- **F_START1** - команда включения (устанавливается извне, сбрасывается функцией)
- **F_START2** - команда отключения (устанавливается извне, сбрасывается функцией)

Флаги состояния

- **FS_BKL** - физическое состояние кнопки включения
- **FS_OTKL** - физическое состояние кнопки отключения

Флаги индикации

- **FI_BKL** - показывать индикацию "реле включено"
- **FI_OTKL** - показывать индикацию "реле отключено"

Условия вызова функций

LACH_CONTROL()

- Вызывается в **локальном режиме** (flags.F_DU == 0)
- Из основного цикла программы при обработке кнопок
- Периодичность: при изменении состояния кнопок

LACH_CONTROL_DU()

- Вызывается в **дистанционном режиме** (flags.F_DU == 1)
- Из функции `init_DU()` при переходе в дистанционный режим
- Из команд, полученных по SPI

Критические аспекты

1. Временные характеристики

- **Длительность импульса**: 20 мс (критично для срабатывания реле)
- **Блокировка других операций**: Во время выполнения импульса

2. Защита от ошибок

- **Одновременное нажатие**: только в LACH_CONTROL()
- **Инверсная логика**: правильная интерпретация сигналов
- **Сброс флагов**: предотвращение повторных срабатываний

3. Синхронизация

- **Атомарность операций**: импульс выполняется полностью
- **Согласованность флагов**: индикация соответствует команде
- **Блокировка WRITE_DATA()**: WR2 не перезаписывается во время импульса

4. Различия режимов

- **Локальный**: полная проверка и защита
- **Дистанционный**: упрощенная логика без физических кнопок

Взаимодействие с другими функциями

Входные данные:

- Физические кнопки (RD4) - только LACH_CONTROL()
- Программные команды (F_START1, F_START2)
- Состояние питания (flags.FPIT_*)

Выходные данные:

- Импульсы управления реле (WR2)
- Флаги индикации (FI_BKL, FI_OTKL)
- Сообщения отладки (UART2)

Блокируемые функции:

- `WRITE_DATA()` - проверяет (F_START1 || F_START2) перед записью WR2

Функции обеспечивают надежное импульсное управление реле с защитой от ошибочных состояний и адаптацией к различным режимам работы системы.