ИНТЕГРИРОВАННАЯ СРЕДА ДРАКОН. КУРС МОЛОДОГО БОЙЦА

Урок 1.

УВЛЕКАТЕЛЬНОЕ НАЧАЛО

Содержание

§1. ВВЕДЕНИЕ.....3
§2. ПРИ ПЕРВОМ ЗАПУСКЕ ИС ДРАКОН ЕЁ НУЖНО НАСТРОИТЬ.....4
§3. ПРОГРАММА УПРАВЛЕНИЯ ДВЕРНОГО ЗАМКА.....5
§4. НАЧАЛО РАБОТЫ.....6
§5. КОНСТРУИРУЕМ АЛГОРИТМ.....7
§6. ПРИСТУПАЕМ К ПРОГРАММИРОВАНИЮ.....13

§1. ВВЕДЕНИЕ

Интегрированная Среда ДРАКОН (далее ИС ДРАКОН) обладает своеобразным интерфейсом, особенности которого не полностью раскрыты в её справке. Из-за этого первые попытки использования вызывают заметные трудности.

Не волнуйтесь! Это дело поправимое.

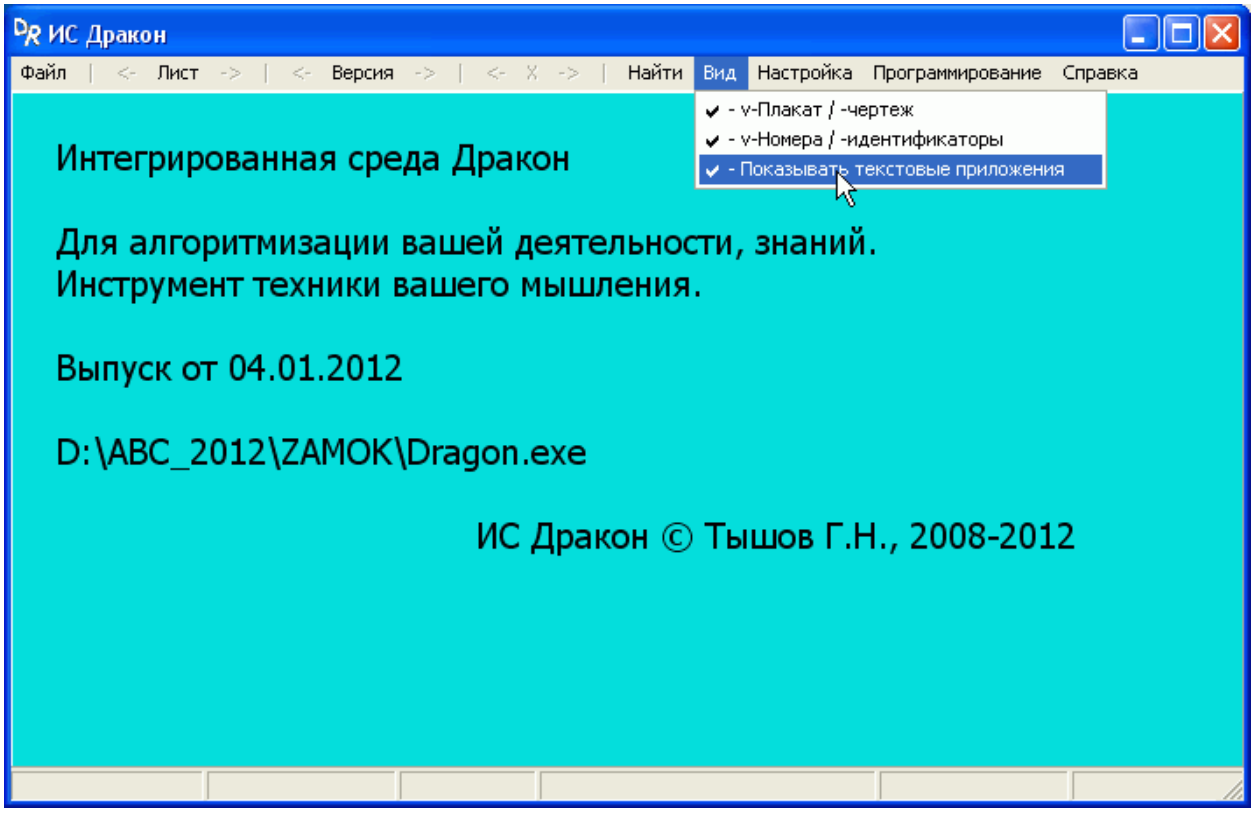
Курс молодого бойца поможет вам совершить первые шаги по созданию дракон-схемы, и генерации исходного текста программы.

Здесь нет описания языка ДРАКОН, анализа его возможностей, достоинств и недостатков. Перед вами простая пошаговая инструкция.

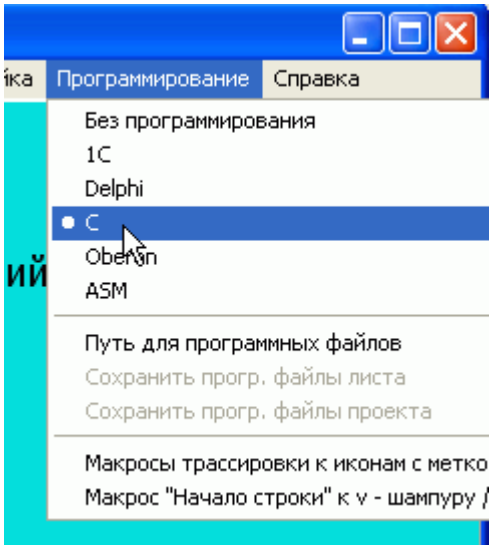
С помощью этой инструкции вы легко убедитесь, что ИС ДРАКОН Геннадия Тышова — это полезный инструмент для *практической* работы. Инструмент, который существенно повышает производительность труда программиста.

§2. ПРИ ПЕРВОМ ЗАПУСКЕ ИС ДРАКОН ЕЁ НУЖНО НАСТРОИТЬ

1) В пункте главного меню «Вид» ставим галочки на всех трёх пунктах:



2) В пункте главного меню «Программирование» ставим метку выбора напротив нужного языка (в данном примере это будет Си).



3) В пункте главного меню «Программирование» нажимаем «Путь для программных файлов».

4) В открывшемся окне выбора файла выбираем любой файл в каталоге с исходными файлами нашей программы, и нажимаем «Открыть».
Внимание. Файл не будет открыт, просто ИС ДРАКОН запомнит каталог, где впоследствии будет сохранять файлы с исходными текстами, сгенерированными из дракон-схем.

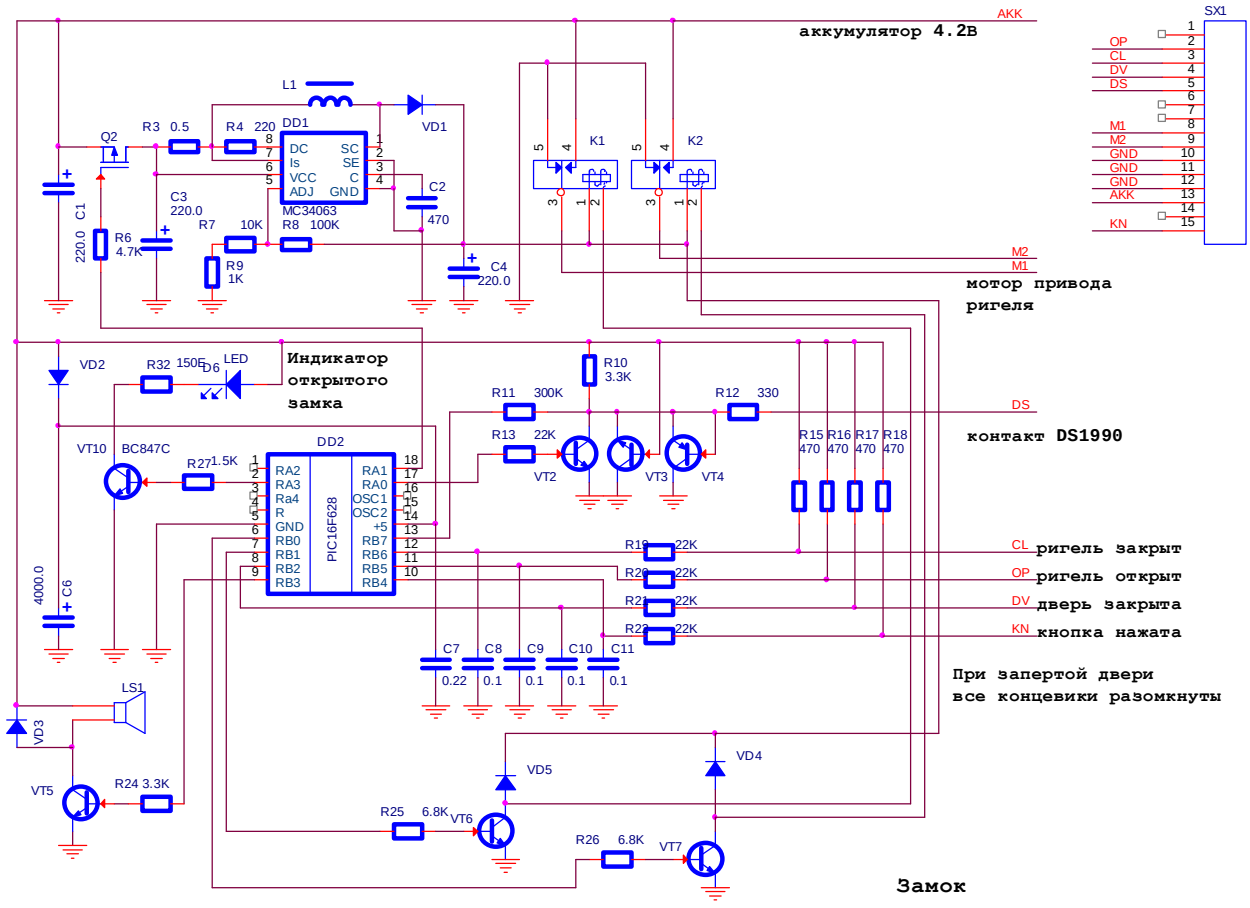
5) Остальные пункты пока не трогаем.

6) Для сохранения настроек не требуется нажимать кнопок типа «ОК», «Сохранить» и т.п.
Все настройки ИС сохраняет в файле «Dragon.ini».

Вот и все. **ИС ДРАКОН готова к работе**.

§3. ПРОГРАММА УПРАВЛЕНИЯ ДВЕРНОГО ЗАМКА

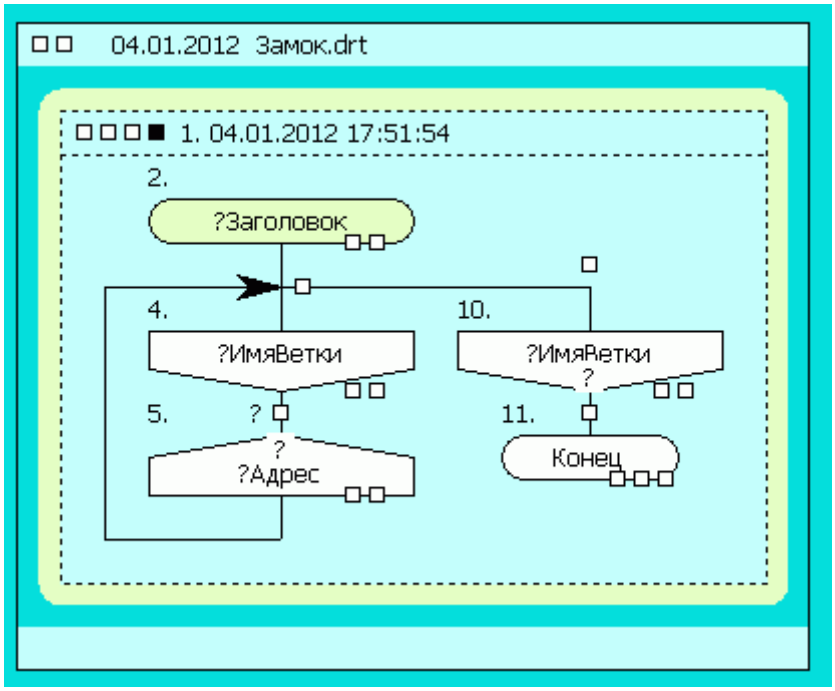
В качестве **первого примера** создадим программу для микропроцессора, управляющего приводом дверного замка.



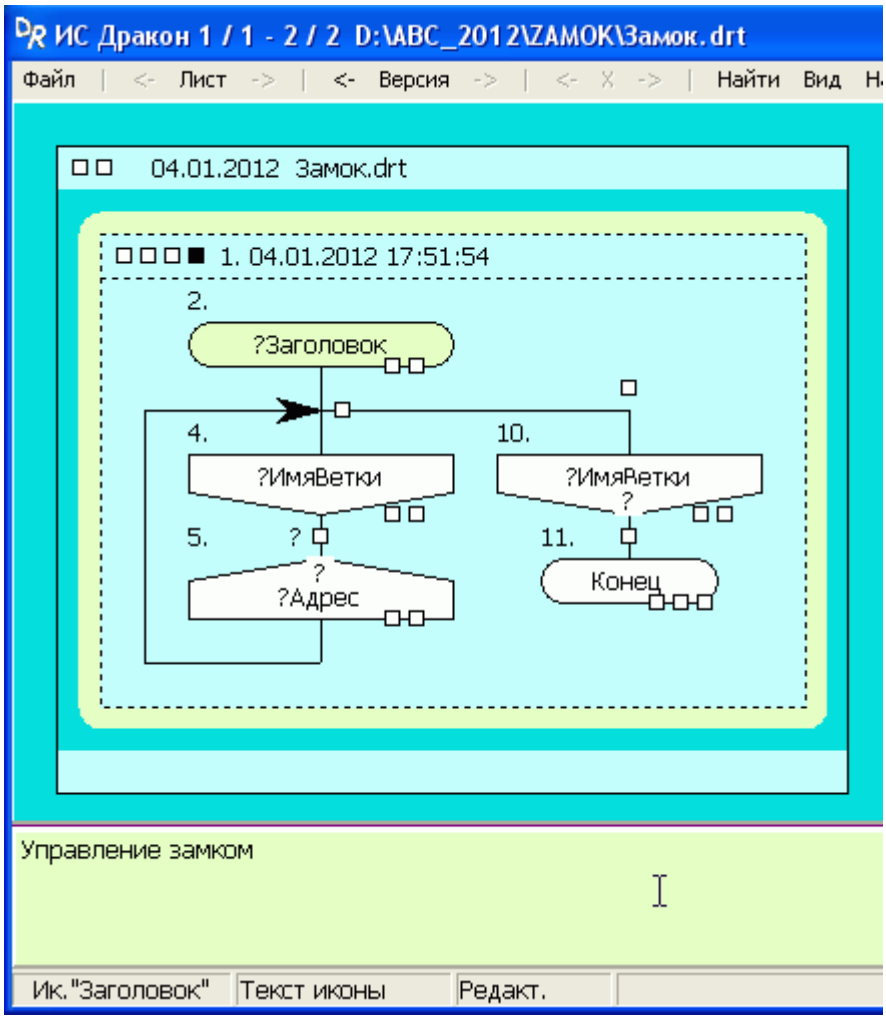
§4. НАЧАЛО РАБОТЫ

7) В пункте главного меню «Файл» выбираем пункт «Новый лист», потом «Сохранить лист как», и сохраняем файл под осмысленным именем, например «Замок».

8) Щёлкаем правой кнопкой мышки в пределах появившегося прямоугольника, и выбираем в контекстном меню «Вставить Силуэт». На листе появляется заготовка схемы «**силуэт**»:



9) ДВОЙНЫМ щелчком по иконе «?Заголовок» открываем текстовое окно, и вносим имя нашей схемы, например – “Управление замком»:

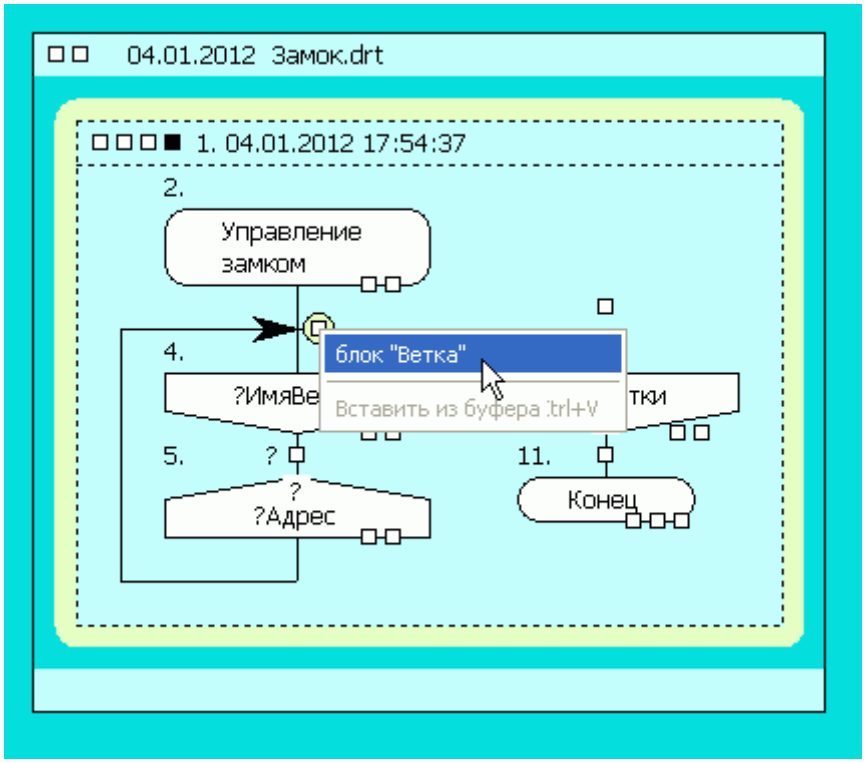


10) Закрываем текстовое окно кнопкой “Esc”, или щелчком мыши за пределами иконы. Заметим сразу, что подобным образом редактируются имена всех икон, кроме адресов переходов (нижний ряд икон).

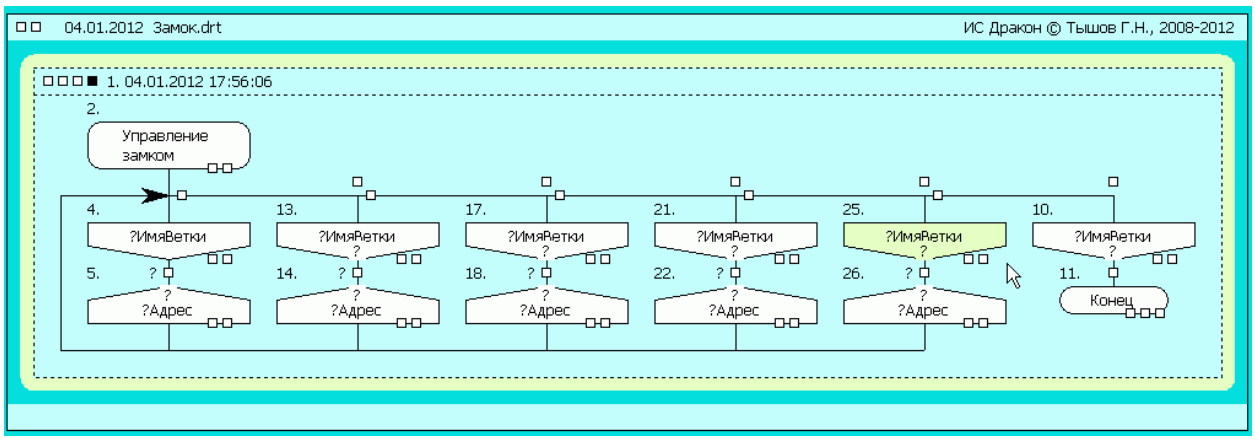
§5. КОНСТРУИРУЕМ АЛГОРИТМ

11) Разбиваем нашу задачу на части, соответствующие состояниям замка: «**Открыт**», «**Закрывается**», «**Закрыт**» «**Открывается**». Каждому состоянию будет соответствовать своя ветка. Сейчас на схеме только 2 ветки, нужно добавить ещё.

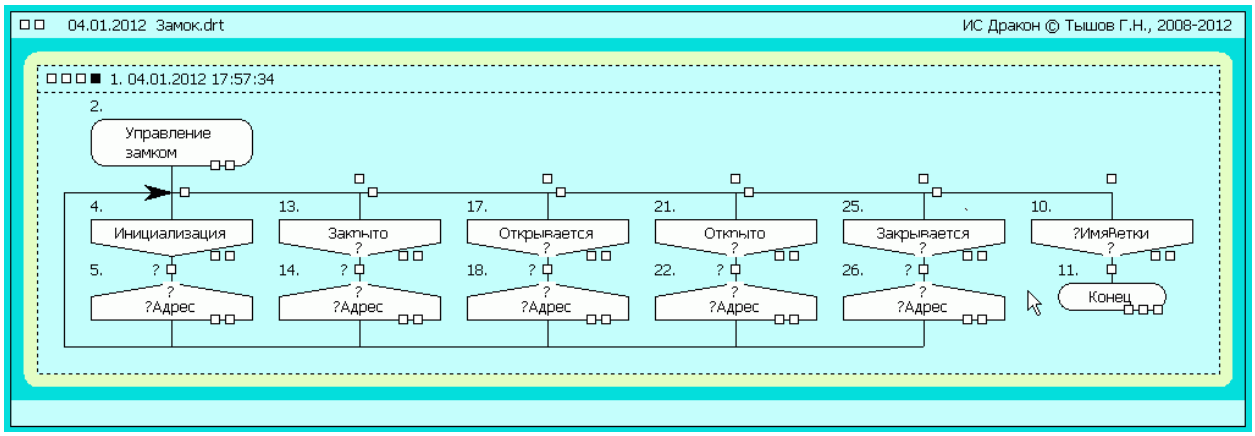
12) На линиях, показывающих связи силуэта, есть белые квадратики. Это «точки ввода», в них можно добавлять новые элементы графики. Щёлкаем правой кнопкой мыши по квадратику под иконой «Управление замком». В открывшемся контекстном меню выбираем пункт «Блок «Ветка»»:



В схеме появляется ещё одна ветка. Добавляем таким образом на схему ещё 4 ветки.
Всего теперь на схеме 6 веток. Первая будет заниматься инициализацией процессора, 4 будут соответствовать нашим состояниям, а шестая, заканчивающаяся иконой «Конец», нам не понадобится. Её мы потом удалим:

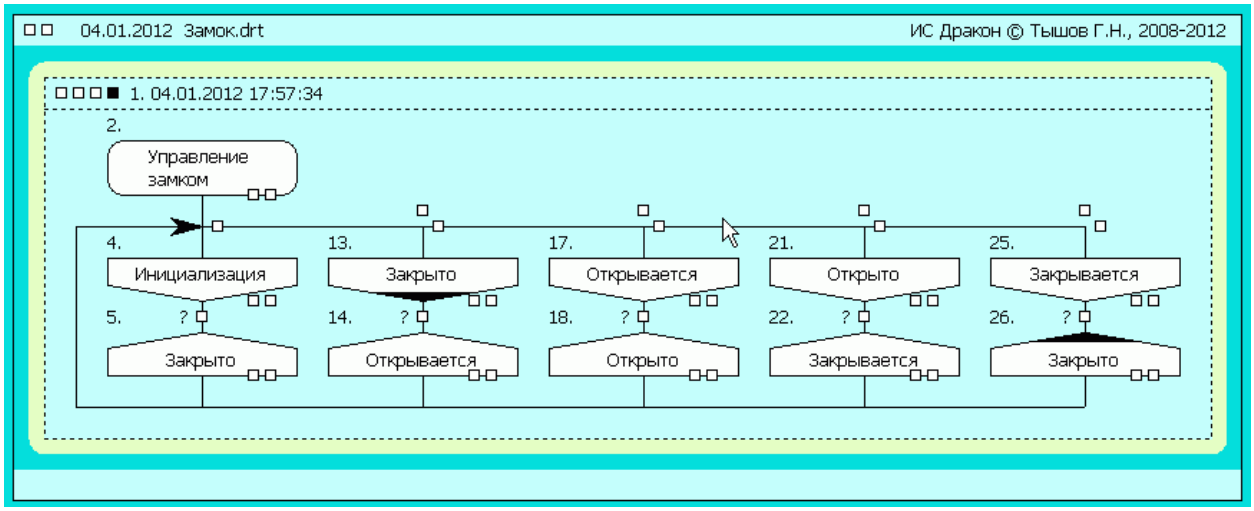


13) Редактируем имена веток. Двойной щелчок по телу иконы «?Имя-Ветки», вводим имя, двойной щелчок по телу следующей иконы и т.д.:



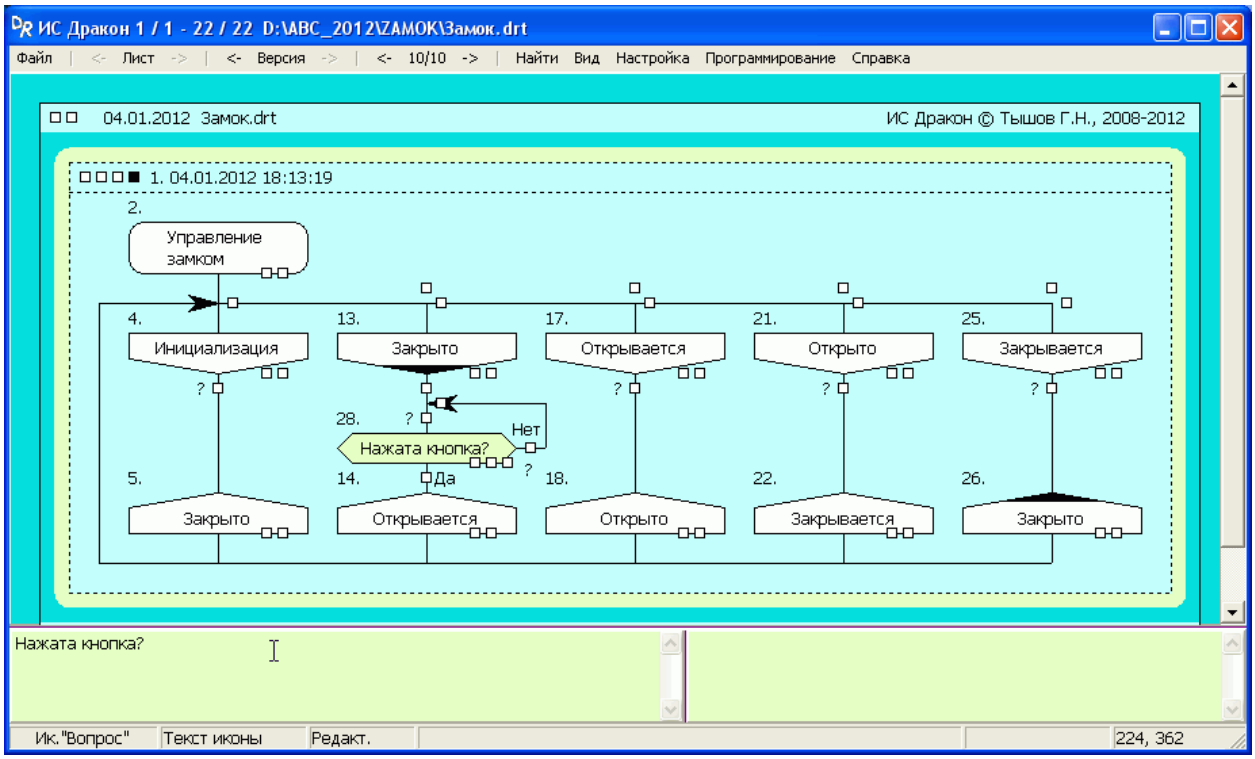
14) Расставляем переходы между ветками. Для этого щелкаем правой кнопкой по телу иконы «?Адрес» в конце ветки «Инициализация», и выбираем в контекстном меню пункт «Для «Адрес» выбрать ветку, <-Х->» , после чего щёлкаем по телу иконы «Закрыто». Повторяем действия для других икон «?Адрес». Из ветки «Закрыто» задаём переход в «Открывается», из «Открывается» - в «Открыто», из «Открыто» - в «Закрывается», из «Закрывается» в «Закрыто».

15) После расстановки переходов можно удалить ненужную ветку с иконой «Конец». Щёлкаем правой кнопкой по телу иконы «?ИмяВетки», и выбираем «Удалить»:

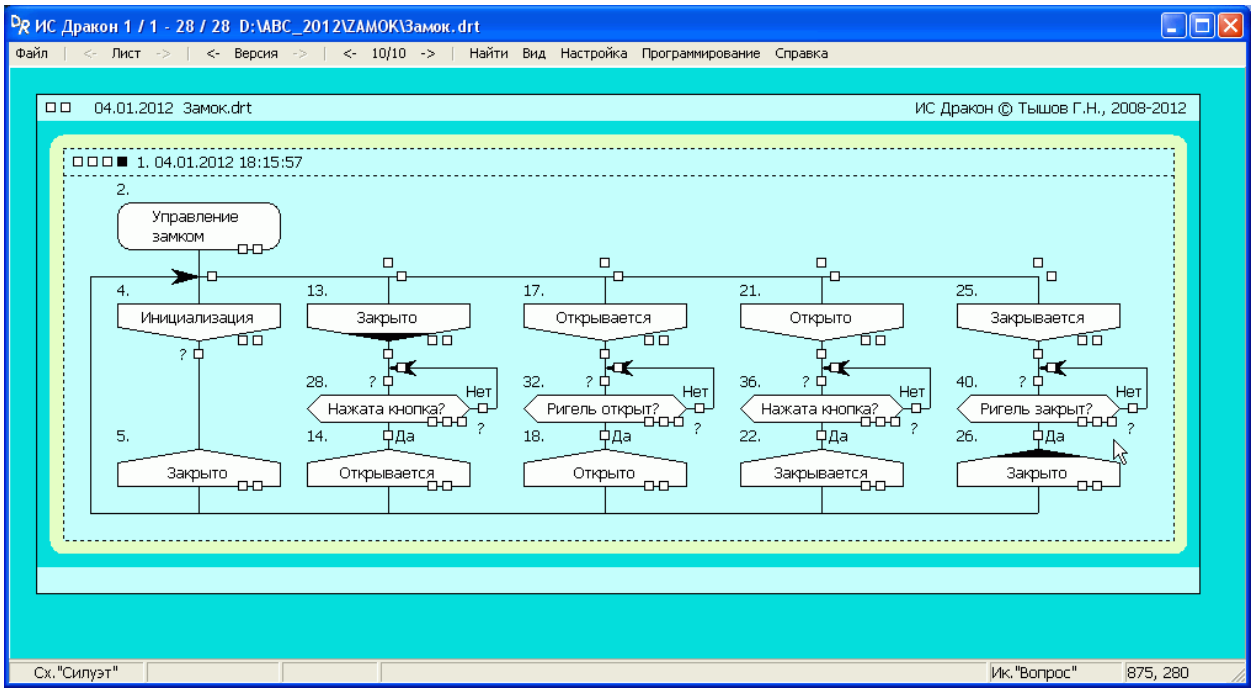


16) Зададим условия, при которых будут происходить переходы.
На ветке «Закрыто», ниже иконы «Закрыто», есть точка ввода (белый квадратик). Щелкаем по нему правой кнопкой, и выбираем в выпавшем меню «блок «Обычный цикл»». Появляется икона «Развилка» с именем «?Вопрос». Двойным щелчком по телу этой иконы открываем текстовое окно, и пишем условие, при котором будет происходить пе-

реход. В данном случае это «Нажата кнопка?». Если ответ на этот вопрос «Да» то будет происходить переход на ветку «Открывается». Если «Нет» - перехода не будет.

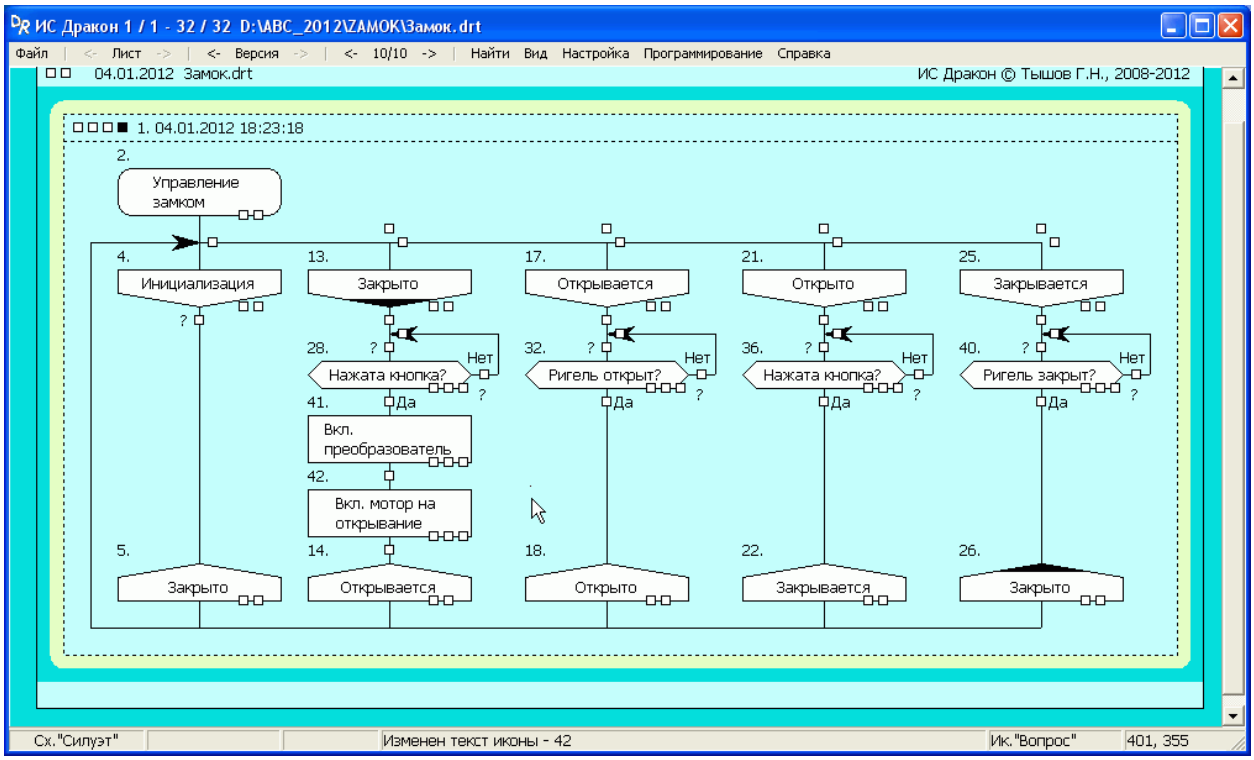


Аналогично расставляем условия переходов для остальных веток:

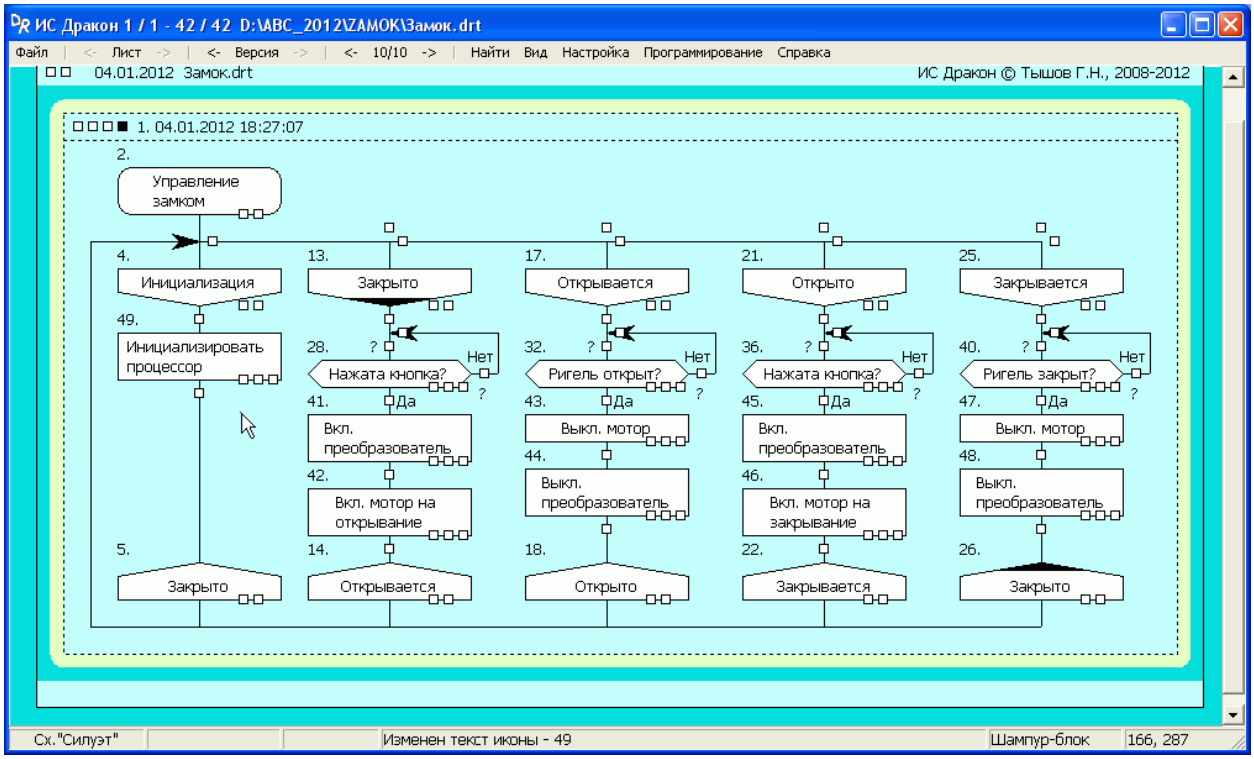


Логика у нас пока упрощённая, учебная. Сейчас мы к этому придираться не будем.

17) Переходим к управлению мотором замка.
На ветке «Закрыто» в точку ввода ниже иконы «Нажата кнопка?» вставляем икону «Действие». Меняем имя иконы на «Вкл. преобразователь».
Ниже иконы «Вкл. преобразователь» вставляем ещё одну икону «Действие», и меняем её имя на «Вкл. мотор на открывание»:

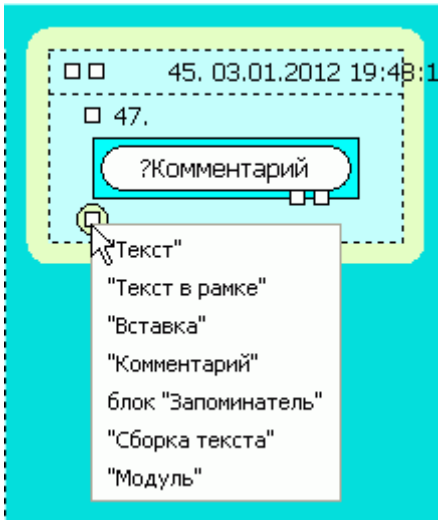


Действуя по аналогии, расставляем остальные иконы «действие». При этом, для облегчения процесса, иконы, или группы икон можно копировать, выделяя их мышкой, и пользуясь Ctr+C / Ctr+V, или контекстным меню. На ветку «Инициализация» вставляем икону с имененм «Инициализировать процессор»:

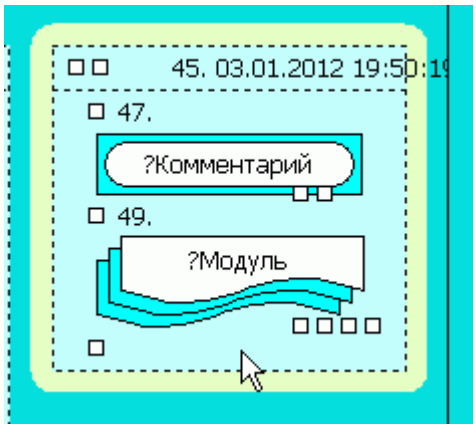


§6. ПРИСТУПАЕМ К ПРОГРАММИРОВАНИЮ

17) Щёлкаем правой кнопкой по полю листа вне схемы «Силуэт», и выбираем в выпавшем меню «вставить «Гном»». Появляется Гном. Щёлкаем правой кнопкой по точке ввода ниже иконы «?Комментарий»:



и выбираем «Модуль». Появляется Модуль:

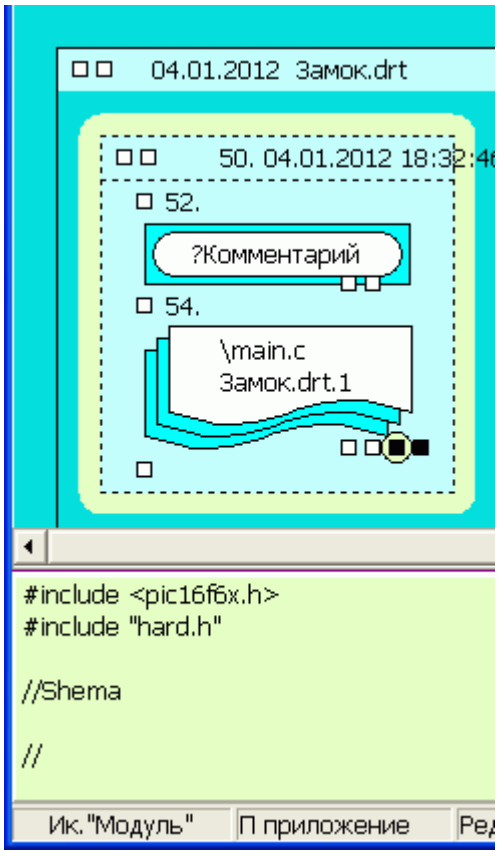


18) Дважды щёлкаем по иконе «Модуль» и вводим в текстовом окне имя файла, в котором будет сохраняться текст программы. Перед именем файла ставим обратный слэш:

`\main.c`

19) Далее щёлкаем по свободному участку схемы «Силуэт» правой кнопкой, и выбираем «Взять ссылку в системный буфер». Снова дважды щёлкаем по иконе «Модуль», в открывшемся текстовом окне устанавливаем курсор в начало второй строки, и нажимаем Ctr+V. Закрываем текстовое окно. Икона «Модуль» должна принять сл. вид:

20) Щёлкаем левой кнопкой по третьему слева кваратику под иконой «Модуль». В открывшемся текстовом окне записываем те строки, которые должны быть в начале нашего программного файла. В этом примере мы запишем директивы включения двух заголовочных файлов. Ниже директив **пишем две «магических» строчки, начинающиеся со знаков комментария:**



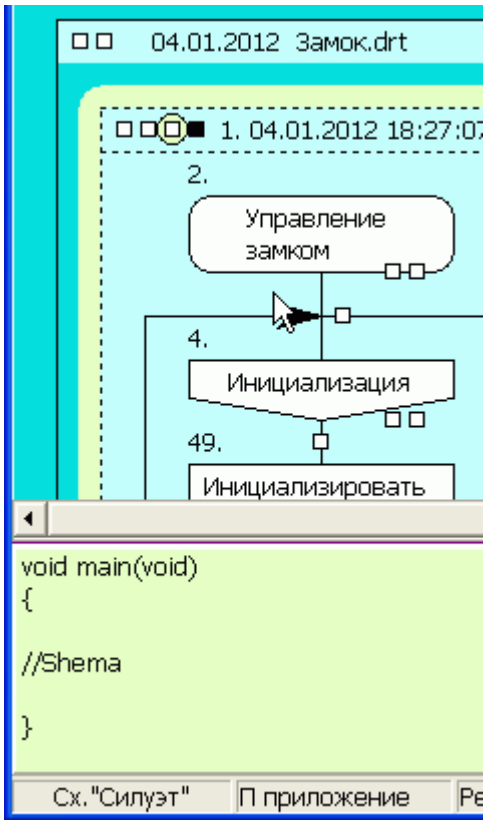
Здесь первый включаемый файл (pic16f6x.h) – файл макроопределений для нашего процессора. Во втором файле (hard.h) объявлены функции для работы с нашим контроллером. В этом уроке мы не будем рассматривать содержание этих функций. Функции очень простые, их имена говорят сами за себя.

Вот содержание файла «hard.h»:

```
#define RIGEL_LOCK    1
#define RIGEL_UNLOCK  2
#define DOOR_CLOSE    1
#define PRESS_BUTTON  1

char get_state_door   ( void );
char get_state_rigel  ( void );
char get_state_button ( void );
void dc_dc_on         ( void );
void dc_dc_off        ( void );
void motor_on_to_open ( void );
void motor_on_to_close( void );
void motor_off        ( void );
void init             ( void );
```

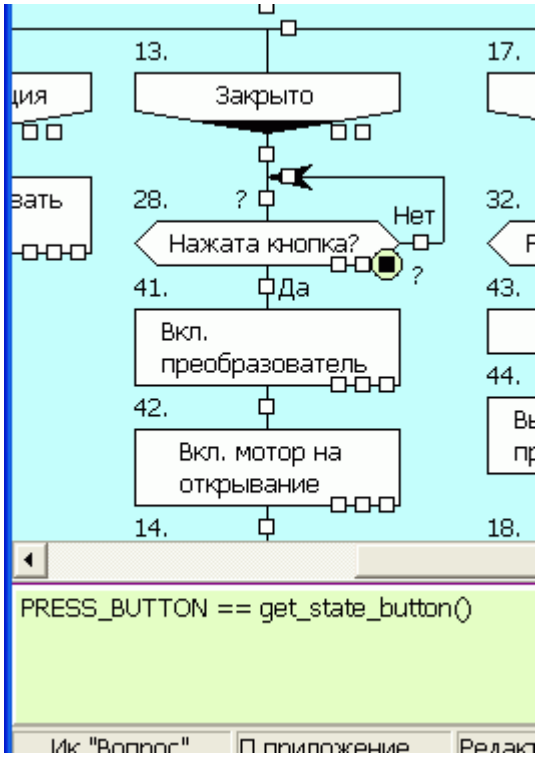
21) В верхнем левом углу листа схемы «Силуэт» тоже есть 4 квадрата. Щёлкаем левой кнопкой мышки по третьему слева квадрату . В открывшемся текстовом окне вводим «скелет» функции, которая впоследствии будет сгенерирована :



22) Приступаем к программированию икон.
Щёлкаем по третьему квадрату иконы «Нажата кнопка?», и вводим логическое выражение, результат которого отвечает на вопрос «Нажата ли кнопка?». В данном случае это следующая строчка:

```
PRESS_BUTTON == get_state_button()
```

В тексте программы эта строчка будет целиком помещена в круглых скобках оператора **if**, поэтому здесь мы **не ставим** точку с запятой после вызова функции:



Далее щёлкаем по третьему квадрату иконы «Вкл. преобразователь», и вписываем вызов соответствующей функции:

```
dc_dc_on();
```

В икону «Вкл. мотор на открывание» записываем:

```
motor_on_to_open();
```

Действуя аналогично, заполняем третьи квадраты всех икон (не забываем о Ctr+C / Ctr+V).

После заполнения третьих квадратов икон всё готово для получения текста программного файла.

23) Нажимаем клавишу **F9**. В заданном при настройке каталоге появляется файл «main.c» с исходным текстом.
В данном примере сгенерировался вот такой файл:

```
==========================================================================
#include <pic16f6x.h>
#include "hard.h"


    /* D:\ABC_2012\ZAMOK\Замок.drt */
    /* 1. ИС Дракон. Трансляция маршрутной части. Изме-
нение 04.01.2012 18:59:28 */
    /* Схема - Процедура */
void main(void)
{


    /* ==================== Шампур */

    /* 2. Заголовок // Управление замком */

    /* 4. ИмяВетки // Инициализация */

    /* 49. Действие // Инициализировать процессор */
init();

    /* 5. Адрес // Закрыто */
goto L28;

    /* ==================== Шампур */

    /* 13. ИмяВетки // Закрыто */

L28:
    /* 28. Вопрос // Нажата кнопка? */
if (!(PRESS_BUTTON == get_state_button())) goto L28;

    /* 41. Действие // Вкл. преобразователь */
dc_dc_on();

    /* 42. Действие // Вкл. мотор на открывание */
motor_on_to_open();

    /* 14. Адрес // Открывается */
goto L32;
```

```
    /* ==================== Шампур */

    /* 17. ИмяВетки // Открывается */

L32:
    /* 32. Вопрос // Ригель открыт? */
if (!(RIGEL_UNLOCK == get_state_rigel())) goto L32;

    /* 43. Действие // Выкл. мотор */
motor_off();

    /* 44. Действие // Выкл. преобразователь */
dc_dc_off();

    /* 18. Адрес // Открыто */
goto L36;

    /* ==================== Шампур */

    /* 21. ИмяВетки // Открыто */

L36:
    /* 36. Вопрос // Нажата кнопка? */
if (!(PRESS_BUTTON == get_state_button())) goto L36;

    /* 45. Действие // Вкл. преобразователь */
dc_dc_on();

    /* 46. Действие // Вкл. мотор на закрывание */
motor_on_to_close();

    /* 22. Адрес // Закрывается */
goto L40;

    /* ==================== Шампур */

    /* 25. ИмяВетки // Закрывается */

L40:
    /* 40. Вопрос // Ригель закрыт? */
if (!(RIGEL_LOCK == get_state_rigel();)) goto L40;

    /* 47. Действие // Выкл. мотор */
motor_off();
```

```
    /* 48. Действие // Выкл. преобразователь */
dc_dc_off();

    /* 26. Адрес // Закрыто */
goto L28;

}

//
```