Использование конвертора Visio2SWITCH

(черновик для обсуждения)

Содержание

Введение

Конвертор Visio2SWITCH предназначен для автоматической генерации C/C++ кода по автоматным графам, нарисованных в Visio в соответствии с требованиями SWITCH-технологии. Посетите на http://www.softcraft.ru/ раздел Автоматы для подробного ознакомления со SWITCH.

Поставка

В архив входят три файла: данный документ, шаблон Visio SWITCH.vss для рисования графов и документ Visio Example.vsd с примером реализации автоматного графа.

Хард & Софт

Для работы необходима Visio с версией 4.0 и выше. Разработка проводилась на Visio 2000 SR1 (6.0.2072).

Поддержка

E-Mail: stardrive@mtu-net.ru

Site: http://www.geocities.com/goloveshin

Благодарности

Я хотел бы поблагодарить Никиту Туккеля (cynical@mail.ru) за ответы на многочисленные вопросы по SWITCH, сайт http://www.softcraft.ru/ за его существование.

Литература

1. Шалыто А.А. SWITCH-технология. Алгоритмизация и программирование задач логического управления. СПб.: Наука, 1998.
2. Шалыто А.А., Туккель Н.И. SWITCH-технология — автоматный подход к созданию программного обеспечения "реактивных" систем //Промышленные АСУ и контроллеры. 2000. №10.
3. Шалыто А.А. Алгоритмизация и программирование для систем логического управления и реактивных систем //Автоматика и телемеханика. 2001. №1.
4. Шалыто А.А., Туккель Н.И. Программирование с явным выделением состояний //Мир ПК. №8, 9.

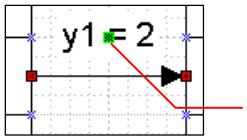
Некоторые операции в Visio

Работа с текстом

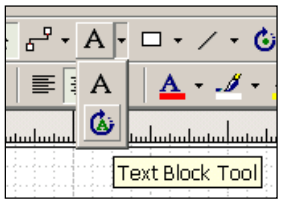
Редактирование текста возможно двумя способами:
1. двойной щелчок левой кнопкой мыши на объекте.
2. нажатие F2, если объект выделен.

Редактирование текста возможно при условии, что объект разрешает данную операцию.

Перемещение текста возможно двумя способами:
1. Перемещение специального контроллера текста, если он имеется.

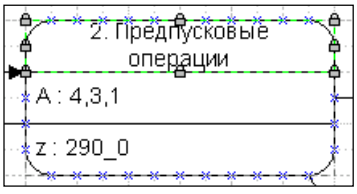


2. Использование Text Block Tool.



Выделение вложенных объектов

Многие (а точнее - большинство) объектов Visio представляют собой группу (group) - объект, состоящий из более простых объектов, которые также могут быть группами. Часто возникает необходимость редактировать свойства произвольно выбранного вложенного в группу объекта. Для этого его необходимо выделить. Рассмотрим пример. Допустим, что мы хотим выделить объект, содержащий имя состояния ‘Предпусковые операции’. Наведем на него курсор и щелкнем левой кнопкой мыши. Произойдет выделение самого внешнего объекта, в котором находится наш объект – группы. Щелкнем еще раз. Произойдет выделение следующего по иерархии вложенности объекта – состояния. Еще один щелчок - и желаемый объект выделен. Теперь можно нажать F2 и произвести редактирование текста объекта.



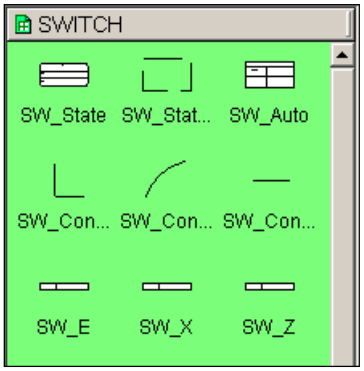
Необходимо заметить, что некоторые объекты могут запрещать выделение себя пользователем, а также изменять порядок выделения (сначала выделяется вложенный объект, а только потом группа).

Шаблон SWITCH

Шаблон SWITCH.vss (Visio Stencil) содержит объекты (masters), с помощью которых пользователь конструирует автоматные графы.

Размещение шаблона SWITCH

Для использования шаблона необходимо сообщить Visio о его существовании. Самый простой способ – поместить шаблон в одну из поддиректорий **Solutions**. Скопируйте его в **{VisioDir}/Solutions/Visio Extras**. Размещение шаблона в **Solutions** гарантирует его автоматическое нахождение Visio при открытии документа, содержащего объекты из данного шаблона. Если пользователь создал/открыл документ, но шаблон не отобразился на экране, то его можно открыть, следуя по цепочке File -> Stencils -> Visio Extras -> SWITCH. На рисунке приведено содержимое шаблона. Рассмотрим подробно каждый из его элементов.



SW_Auto – данные автомата

Назначение: определяет параметры автомата.

Автомат	
Имя:	0
Название:	Дизель

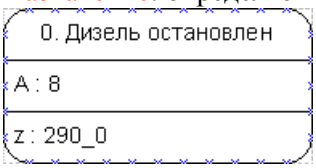
Имя: имя автомата.
Формат: любая комбинация из цифр [0-9], латинских букв верхнего и нижнего регистра (кроме [e,x,y,z]) и подчеркивания [_].
В данном примере имя определено как [0] и, следовательно, автоматная функция получит имя [A0], а переменная состояния - [y0].

Название: ассоциируемая с именем автомата строка.
Формат: произвольная символьная строка.

Если данный объект присутствует на странице, то считается, что на странице определен автоматный граф и конвертор будет обрабатывать состояния, группы состояний и дуги. Если SW_Auto отсутствует, то вышеперечисленные объекты игнорируются, но объекты SW_E[X,Z] обрабатываются, что позволяет использовать под комментарии целые страницы.

SW_State – состояние

Назначение: определяет параметры одного состояния автомата.



Имя и название: номер и название состояния.
Формат: {число}[.]{произвольная символьная строка}.

Вложенные автоматы: набор вложенных автоматов.
Формат: [A:]{имена автоматов через запятую}. Пример: A: 8, 3, 4

Действия: набор действий.
Формат: [z:]{имена действий через запятую}. Пример: z: 290_0, 110, 10

SW_StateGroup – группа состояний

Назначение: определяет группу состояний.

Объекты состояние и группа состояний настроены так, что добавление состояния в группу происходит автоматически при перетаскивании состояния в группу. Аналогично, вынесение состояния за пределы группы удаляет состояние из группы. Для создания группы пользователь должен сначала поместить на страницу группу состояний и только потом помещать в нее состояния. Возможны вложенные группы состояний.

Text: конвертором никак не интерпретируется и может использоваться для комментариев (например, описание особенностей данной группы состояний).

SW_E, SW_X, SW_Z – комментарии

Назначение: определяет комментарий к имени события, входной переменной и действия.

e20	Нажатие кнопки ПУСК

Имя: имя события, входной переменной или действия.
Формат: имя должно быть полным, т.е. с префиксами: x220_0, z290_0, e10.

Комментарий: комментарий к имени.
Формат: произвольная символьная строка.

Для удобства чтения пользователем автоматных графов допускается многократное дублирование комментариев для одного имени на разных страницах: если два автомата используют e20, то на их страницах можно разместить вышеприведенный (рисунок) комментарий.

SW_Connector1(2,3) – дуги

Назначение: задают соединения между состояниями - дуги.

Всего предлагается три соединителя, которые отличаются (с точки зрения пользователя) только стилем отрисовки: 1 – набор прямых, 2 – дуга окружности, 3 – прямая. Свойства отрисовки – толщина линий, цвет, стрелки на концах и т.п. – доступны для редактирования (которого лучше избегать для унификации изображений графов).

Text: описание параметров перехода.
Формат: {число}[:]{условие перехода}[/ z:]{имена действий через запятую}. Обязательно только {условие перехода}, остальное может быть опущено.

{число}	Приоритет перехода: [1,2,3,…]. (1 - высший)
[/ z:]{имена действий}	Пример: [/ z: 290_0, 110, 10].
{условие перехода}	Логическое выражение с участием имен событий, переменных состояний автоматов, входных переменных. Пример: e10 x10 + x220_0 (y10_0 = 5)
Имя события	[e] или [E], за которым следует любая комбинация из цифр [0-9]. Пример: [e10].
Имя переменной состояния	[y] или [Y], за которым следует имя автомата. Пример: [y20].
Имя входной переменной	[x] или [X], за которым следует любая комбинация из цифр [0-9], латинских букв верхнего и нижнего регистра (кроме [e,x,y,z]) и подчеркивания [_]. Пример: [x220_0].

В {условие перехода} доступны следующие операторы:

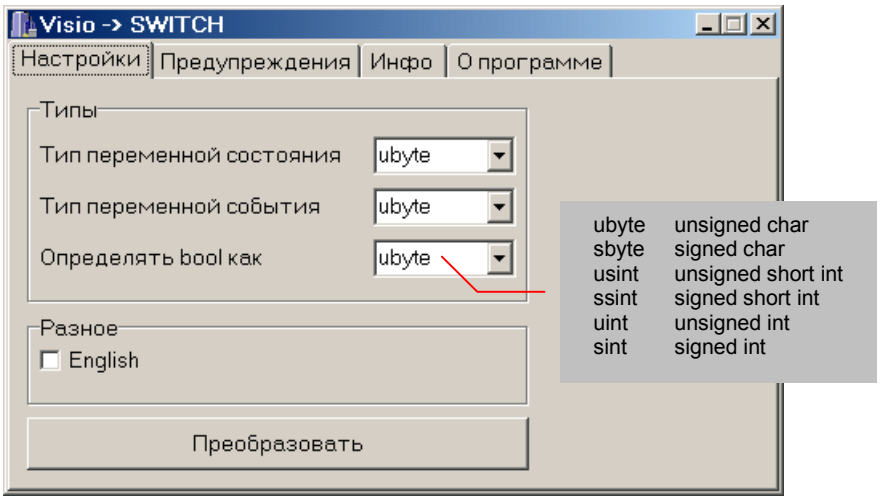
Требуемый оператор	Варианты через запятую	Примеры
AND	*, &&, (ничего)	x10*e20, x10 && e20, x10e20
OR	+, ||	x10 + e20, x10 || e20
NOT	!	!x220_0
EQUAL	=, ==	y20 = 1, y20 == 2
NOT EQUAL	!=, <>	y20 != 1, y20 <> 2

Конвертор

Текущие плохизмы:

- мало тестировался
- преобразование не выделено в отдельную нить
- только C-код на выходе
- плохое быстродействие (в стадии работы с Visio)

Интерфейс



Соглашения

Имеют место следующие соглашения:

1. Под проектом SWITCH понимается множество автоматных функций, реализующих системонезависимую часть и множество “заглушек”, реализующих системозависимую часть.
2. Документ Visio ставиться в соответствие проекту SWITCH.
3. Имя документа ставиться в соответствие имени проекта.
4. В директории конвертора создается директория с именем проекта для размещения файлов проекта.
5. Документ Visio может состоять из одной и более страниц.
6. Имя страницы конвертором не используется и может быть произвольным.
7. Страница ставиться в соответствие автоматному графу, но только в случае присутствия объекта SW_Auto.
8. Конвертор обрабатывает только активный документ, т.е. ‘видимый’ в текущий момент. Другие открытые документы игнорируются.
9. Пользователь должен самостоятельно запустить Visio и открыть нужный документ для преобразования.

Файлы

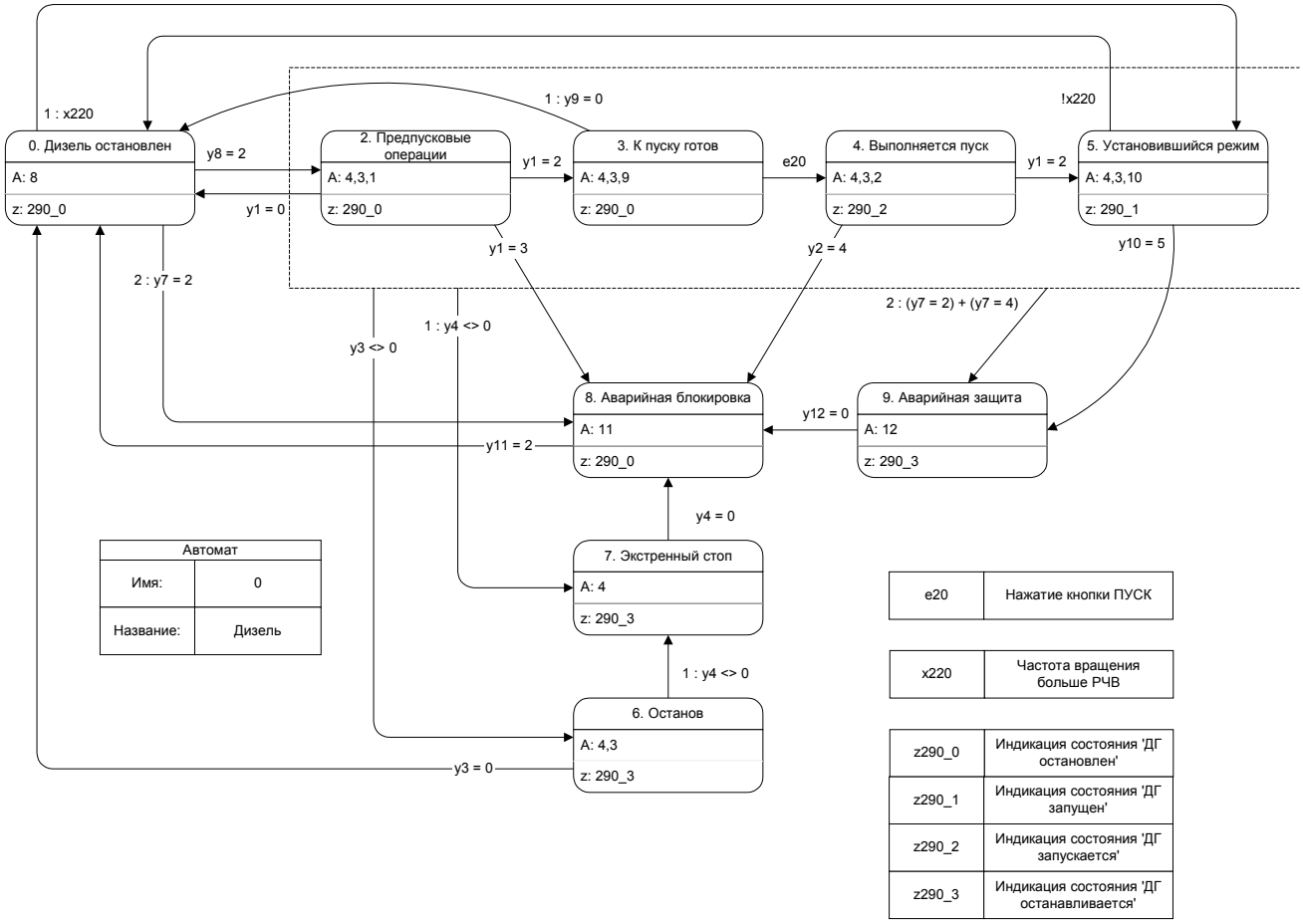
Реализацию конвертором нижеприведенных файлов для Example.vsd смотрите в Приложении.

Файл	Описание	Обновляется
types.h	определение типов	всегда
log.h	отладочная информация	всегда
log.cpp	реализация отладочных функций как ‘оберток’ функций пользователя, которые непосредственно осуществляют отладочный вывод	всегда
log_user.cpp	пользователь реализует функции отладочного вывода	log_user.cpp отсутствует
x.cpp	реализация функций входных переменных как ‘оберток’ функций пользователя	всегда
x_user.cpp	пользователь реализует функции входных переменных	x_user.cpp отсутствует
z.cpp	реализация функций действий как ‘оберток’ функций пользователя	всегда
z_user.cpp	пользователь реализует функции действий	z_user.cpp отсутствует
common.h	объявление автоматных функций, переменных состояний автоматов	всегда
common.cpp	реализация автоматных функций	всегда

Приложение

Автоматный граф из Example.vsd и все сгенерированные для него файлы.

Автоматный граф



Автомат	
Имя:	0
Название:	Дизель

e20	Нажатие кнопки ПУСК

x220	Частота вращения больше РЧВ

z290_0	Индикация состояния 'ДГ остановлен'
z290_1	Индикация состояния 'ДГ запущен'
z290_2	Индикация состояния 'ДГ запускается'
z290_3	Индикация состояния 'ДГ останавливается'

types.h

```
//--- this file is machine generated ---

#ifndef TypesH
#define TypesH

typedef unsigned char ubyte;
typedef signed char sbyte;
typedef unsigned short int usint;
typedef signed short int ssint;
typedef unsigned int uint;
typedef signed int sint;
typedef unsigned long ulong;
typedef signed long slong;

#endif
```

common.h

```
//--- this file is machine generated ---

#ifndef CommonH
#define CommonH

#include "types.h"

typedef struct{
  ubyte y0; // Дизель
  ubyte y1;
  ubyte y10;
  ubyte y11;
  ubyte y12;
  ubyte y2;
  ubyte y3;
  ubyte y4;
  ubyte y7;
  ubyte y8;
  ubyte y9;
} common_t;

extern common_t cm;

// Автоматы A
void A0( ubyte e );
void A1( ubyte e );
void A10( ubyte e );
void A11( ubyte e );
void A12( ubyte e );
void A2( ubyte e );
void A3( ubyte e );
void A4( ubyte e );
void A7( ubyte e );
void A8( ubyte e );
void A9( ubyte e );

// Переменные X
ubyte x220(void); // Частота вращения больше РЧВ

// Действия Z
void z290_0(void); // Индикация состояния 'ДГ остановлен'
void z290_1(void); // Индикация состояния 'ДГ запущен'
void z290_2(void); // Индикация состояния 'ДГ запускается'
void z290_3(void); // Индикация состояния 'ДГ останавливается'

// События E
// e0 - _инициализация_
// e20 - Нажатие кнопки ПУСК

#endif
```

common.cpp

```
//--- this file is machine generated ---

#include "common.h"
#include "log.h"

common_t cm;

//-----------------------------------------------------------
// A0 - Дизель
//-----------------------------------------------------------
void A0( ubyte e )
{
 ubyte y_old = cm.y0;

 #ifdef A0_BEGIN_LOGGING
  log_a_begin(0, y_old, e);
 #endif

 switch( cm.y0 )
 {
  case 0:// Дизель остановлен
   A8(e);
   if(x220())
   {
                        cm.y0 = 5;
   }
   else
   if(cm.y7 == 2)
   {
                        cm.y0 = 8;
   }
   else
   if(cm.y8 == 2)
   {
                        cm.y0 = 2;
   }
  break;

  case 2:// Предпусковые операции
   A4(e); A3(e); A1(e);
   if(cm.y4 != 0)
   {
                        cm.y0 = 7;
   }
   else
   if((cm.y7 == 2) || (cm.y7 == 4))
   {
                        cm.y0 = 9;
   }
   else
   if(cm.y3 != 0)
   {
                        cm.y0 = 6;
   }
   else
   if(cm.y1 == 3)
   {
                        cm.y0 = 8;
   }
   else
   if(cm.y1 == 2)
   {
                        cm.y0 = 3;
   }
   else
   if(cm.y1 == 0)
   {
                        cm.y0 = 0;
   }
  break;

  case 3:// К пуску готов
   A4(e); A3(e); A9(e);
   if(cm.y4 != 0)
   {
                        cm.y0 = 7;
   }
   else
```

```
 if((cm.y7 == 2) || (cm.y7 == 4))
 {
                        cm.y0 = 9;
 }
 else
 if(cm.y3 != 0)
 {
                        cm.y0 = 6;
 }
 else
 if(cm.y9 == 0)
 {
                        cm.y0 = 0;
 }
 else
 if((e == 20))
 {
                        cm.y0 = 4;
 }
break;

case 4:// Выполняется пуск
 A4(e); A3(e); A2(e);
 if(cm.y4 != 0)
 {
                        cm.y0 = 7;
 }
 else
 if((cm.y7 == 2) || (cm.y7 == 4))
 {
                        cm.y0 = 9;
 }
 else
 if(cm.y3 != 0)
 {
                        cm.y0 = 6;
 }
 else
 if(cm.y2 == 4)
 {
                        cm.y0 = 8;
 }
 else
 if(cm.y1 == 2)
 {
                        cm.y0 = 5;
 }
break;

case 5:// Установившийся режим
 A4(e); A3(e); A10(e);
 if(cm.y4 != 0)
 {
                        cm.y0 = 7;
 }
 else
 if((cm.y7 == 2) || (cm.y7 == 4))
 {
                        cm.y0 = 9;
 }
 else
 if(cm.y3 != 0)
 {
                        cm.y0 = 6;
 }
 else
 if(cm.y10 == 5)
 {
                        cm.y0 = 9;
 }
 else
 if(!x220())
 {
                        cm.y0 = 0;
 }
break;

case 6:// Останов
 A4(e); A3(e);
 if(cm.y4 != 0)
 {
```

```
                                cm.y0 = 7;
  }
  else
  if(cm.y3 == 0)
  {
                                cm.y0 = 0;
  }
 break;

 case 7:// Экстренный стоп
  A4(e);
  if(cm.y4 == 0)
  {
                                cm.y0 = 8;
  }
 break;

 case 8:// Аварийная блокировка
  A11(e);
  if(cm.y11 == 2)
  {
                                cm.y0 = 0;
  }
 break;

 case 9:// Аварийная защита
  A12(e);
  if(cm.y12 == 0)
  {
                                cm.y0 = 8;
  }
 break;

 default:
  #ifdef A0_ERRORS_LOGGING
    log_write(LOG_GRAPH_ERROR, "Ошибка в автомате A0: неизвестный номер состояния!");
  #else
    ;
  #endif
}

if( y_old == cm.y0 ) goto A0_end;

#ifdef A0_TRANS_LOGGING
 log_a_trans(0, y_old, cm.y0);
#endif

switch( cm.y0 )
{
 case 0:// Дизель остановлен
  A8(0);
  z290_0();
 break;

 case 2:// Предпусковые операции
  A4(0); A3(0); A1(0);
  z290_0();
 break;

 case 3:// К пуску готов
  A4(0); A3(0); A9(0);
  z290_0();
 break;

 case 4:// Выполняется пуск
  A4(0); A3(0); A2(0);
  z290_2();
 break;

 case 5:// Установившийся режим
  A4(0); A3(0); A10(0);
  z290_1();
 break;

 case 6:// Останов
  A4(0); A3(0);
  z290_3();
 break;

 case 7:// Экстренный стоп
  A4(0);
```

```
      z290_3();
    break;

    case 8:// Аварийная блокировка
      A11(0);
      z290_0();
    break;

    case 9:// Аварийная защита
      A12(0);
      z290_3();
    break;

  }

  A0_end: ;
  #ifdef A0_END_LOGGING
    log_a_end(0, cm.y0, e);
  #endif
}

void A1( ubyte e ){}
void A10( ubyte e ){}
void A11( ubyte e ){}
void A12( ubyte e ){}
void A2( ubyte e ){}
void A3( ubyte e ){}
void A4( ubyte e ){}
void A7( ubyte e ){}
void A8( ubyte e ){}
void A9( ubyte e ){}
```

log.h

```
//--- this file is machine generated ---

#ifndef LogH
#define LogH

#include "types.h"

#define SWITCH_LOGGING

#ifdef SWITCH_LOGGING
  #define Z_LOGGING
  #define X_LOGGING
  #define A_BEGINS_LOGGING
  #define A_TRANS_LOGGING
  #define A_ENDS_LOGGING
  #define A_ERRORS_LOGGING

  enum{
    LOG_Z = '*',
    LOG_X = '>',
    LOG_GRAPH_BEGIN = '{',
    LOG_GRAPH_TRANS = 'T',
    LOG_GRAPH_END = '}',
    LOG_GRAPH_ERROR = 'E'
  };

  void log_a_begin(ubyte a, ubyte y, ubyte e);
  void log_a_trans(ubyte a, ubyte yo, ubyte yn);
  void log_a_end(ubyte a, ubyte y, ubyte e);
  void log_x(ubyte x, ubyte res);
  void log_z(ubyte z);
  void log_write(char, const char* str);
#endif

#ifdef A_BEGINS_LOGGING
  #define A0_BEGIN_LOGGING
  #define A1_BEGIN_LOGGING
  #define A10_BEGIN_LOGGING
  #define A11_BEGIN_LOGGING
  #define A12_BEGIN_LOGGING
  #define A2_BEGIN_LOGGING
  #define A3_BEGIN_LOGGING
  #define A4_BEGIN_LOGGING
  #define A7_BEGIN_LOGGING
  #define A8_BEGIN_LOGGING
  #define A9_BEGIN_LOGGING
#endif

#ifdef A_TRANS_LOGGING
  #define A0_TRANS_LOGGING
  #define A1_TRANS_LOGGING
  #define A10_TRANS_LOGGING
  #define A11_TRANS_LOGGING
  #define A12_TRANS_LOGGING
  #define A2_TRANS_LOGGING
  #define A3_TRANS_LOGGING
  #define A4_TRANS_LOGGING
  #define A7_TRANS_LOGGING
  #define A8_TRANS_LOGGING
  #define A9_TRANS_LOGGING
#endif

#ifdef A_ENDS_LOGGING
  #define A0_END_LOGGING
  #define A1_END_LOGGING
  #define A10_END_LOGGING
  #define A11_END_LOGGING
  #define A12_END_LOGGING
  #define A2_END_LOGGING
  #define A3_END_LOGGING
  #define A4_END_LOGGING
  #define A7_END_LOGGING
  #define A8_END_LOGGING
  #define A9_END_LOGGING
#endif

#ifdef A_ERRORS_LOGGING
  #define A0_ERRORS_LOGGING
```

```
  #define A1_ERRORS_LOGGING
  #define A10_ERRORS_LOGGING
  #define A11_ERRORS_LOGGING
  #define A12_ERRORS_LOGGING
  #define A2_ERRORS_LOGGING
  #define A3_ERRORS_LOGGING
  #define A4_ERRORS_LOGGING
  #define A7_ERRORS_LOGGING
  #define A8_ERRORS_LOGGING
  #define A9_ERRORS_LOGGING
#endif

#endif
```

log.cpp

```
//--- this file is machine generated ---

#include "log.h"

#ifdef SWITCH_LOGGING

typedef struct{
  ubyte       dig;
  const char* n;
  const char* n_name;
} int_str2_t;

typedef struct{
  const char* n;
  const char* n_name;
} str2_t;

typedef struct{
  const char* n;
  const char* n_name;
  str2_t* str2;
} str3_t;

int_str2_t e_str2[2] =
{
  {  0, "e0", "_инициализация_" },
  {  20, "e20", "Нажатие кнопки ПУСК" }
};

str2_t x_str2[1] =
{
  { "x220", "Частота вращения больше РЧВ" }
};

str2_t z_str2[4] =
{
  { "z290_0", "Индикация состояния 'ДГ остановлен'" },
  { "z290_1", "Индикация состояния 'ДГ запущен'" },
  { "z290_2", "Индикация состояния 'ДГ запускается'" },
  { "z290_3", "Индикация состояния 'ДГ останавливается'" }
};

str2_t a0_str2[10] =
{
  { "0", "Дизель остановлен" },
  { "?", "" },
  { "2", "Предпусковые операции" },
  { "3", "К пуску готов" },
  { "4", "Выполняется пуск" },
  { "5", "Установившийся режим" },
  { "6", "Останов" },
  { "7", "Экстренный стоп" },
  { "8", "Аварийная блокировка" },
  { "9", "Аварийная защита" }
};

str3_t A_str3[1] =
{
  { "A0", "Дизель", a0_str2 }
};

//---------------------------------------------------------------------------
void e_find(ubyte e, const char** n, const char** n_name)
{
  static const char* nothing = "нет такого!";
  *n = nothing;
  *n_name = nothing;
  for(uint i = 0; i < 2; i++)
    if(e_str2[i].dig == e){
      *n = e_str2[i].n; *n_name = e_str2[i].n_name; return;
    }
}
//---------------------------------------------------------------------------

//---------------------------------------------------------------------------
void log_a_begin_user(const char* a, const char* a_name, const char* y, const char* y_name, const char* e, const
char* e_name);
void log_a_begin(ubyte a, ubyte y, ubyte e)
```

```
{
 const char *e_n, *e_n_name;
 e_find(e, &e_n, &e_n_name);
 log_a_begin_user(A_str3[a].n, A_str3[a].n_name, A_str3[a].str2[y].n, A_str3[a].str2[y].n_name, e_n, e_n_name);
}
//---------------------------------------------------------------------------
void log_a_trans_user(const char* a, const char* a_name, const char* yo, const char* yo_name, const char* yn, const char* yn_name);
void log_a_trans(ubyte a, ubyte yo, ubyte yn)
{
 log_a_trans_user(A_str3[a].n, A_str3[a].n_name, A_str3[a].str2[yo].n, A_str3[a].str2[yo].n_name, A_str3[a].str2[yn].n, A_str3[a].str2[yn].n_name);
}
//---------------------------------------------------------------------------
void log_a_end_user(const char* a, const char* a_name, const char* y, const char* y_name, const char* e, const char* e_name);
void log_a_end(ubyte a, ubyte y, ubyte e)
{
 const char *e_n, *e_n_name;
 e_find(e, &e_n, &e_n_name);
 log_a_end_user(A_str3[a].n, A_str3[a].n_name, A_str3[a].str2[y].n, A_str3[a].str2[y].n_name, e_n, e_n_name);
}
//---------------------------------------------------------------------------
void log_x_user(const char* x, const char* x_name, ubyte res);
void log_x(ubyte x, ubyte res)
{
 log_x_user(x_str2[x].n, x_str2[x].n_name, res);
}
//---------------------------------------------------------------------------
void log_z_user(const char* z, const char* z_name);
void log_z(ubyte z)
{
 log_z_user(z_str2[z].n, z_str2[z].n_name);
}
//---------------------------------------------------------------------------
void log_write_user(char ch, const char* str);
void log_write(char ch, const char* str)
{
 log_write_user(ch, str);
}
//---------------------------------------------------------------------------

#endif
```

log_user.cpp

Представлен пример реализации вывода через std::ofstream.

```
#include "log.h"

#ifdef SWITCH_LOGGING

#include <fstream>
using namespace std;

ofstream out("test.txt");

//---------------------------------------------------------------------------
void log_a_begin_user(const char* a, const char* a_name, const char* y, const char* y_name, const char* e, const char* e_name)
{
	out << (char)LOG_GRAPH_BEGIN << "  " << a << "("<< a_name << "): в состоянии " << y << "(" << y_name << ") запущен с событием " << e << "(" << e_name << ")" << endl;
}
//---------------------------------------------------------------------------
void log_a_trans_user(const char* a, const char* a_name, const char* yo, const char* yo_name, const char* yn, const char* yn_name)
{
	out << (char)LOG_GRAPH_TRANS << "  " << a << "("<< a_name << "): перешел из состояния " << yo << "(" << yo_name << ") в состояние " << yn << "(" << yn_name << ")" << endl;
}
//---------------------------------------------------------------------------
void log_a_end_user(const char* a, const char* a_name, const char* y, const char* y_name, const char* e, const char* e_name)
{
	out << (char)LOG_GRAPH_END << "  " << a << "("<< a_name << "): завершил обработку события " << e << "(" << e_name << ") в состоянии " << y << "(" << y_name << ")" << endl;
}
//---------------------------------------------------------------------------
void log_x_user(const char* x, const char* x_name, ubyte res)
{
	out << (char)LOG_X << "  " << x << " - " << x_name << " - вернул " << (bool)res << endl;
}
//---------------------------------------------------------------------------
void log_z_user(const char* z, const char* z_name)
{
	out << (char)LOG_Z << "  " << z << ". " << z_name << endl;
}
//---------------------------------------------------------------------------
void log_write_user(char ch, const char* str)
{
	out << ch << " " << str << endl;
}
//---------------------------------------------------------------------------

#endif
```

x.cpp

```
//--- this file is machine generated ---

#include "common.h"
#include "log.h"

//---------------------------------------------------------------------------
ubyte x220_user(void);
ubyte x220(void)
{
  ubyte b = x220_user();

  #ifdef X_LOGGING
    log_x(0, b);
  #endif

  return b;
}
//---------------------------------------------------------------------------
```

x_user.cpp

```
#include "common.h"

//---------------------------------------------------------------------------
ubyte x220_user(void)
{ // Частота вращения больше РЧВ

  //---- Расположите Ваш код здесь ----

  return 1;
}
//---------------------------------------------------------------------------
```

z.cpp

```
//--- this file is machine generated ---

#include "common.h"
#include "log.h"

//---------------------------------------------------------------------------
void z290_0_user(void);
void z290_0(void)
{
 z290_0_user();

 #ifdef Z_LOGGING
   log_z(0);
 #endif
}
//---------------------------------------------------------------------------
void z290_1_user(void);
void z290_1(void)
{
 z290_1_user();

 #ifdef Z_LOGGING
   log_z(1);
 #endif
}
//---------------------------------------------------------------------------
void z290_2_user(void);
void z290_2(void)
{
 z290_2_user();

 #ifdef Z_LOGGING
   log_z(2);
 #endif
}
//---------------------------------------------------------------------------
void z290_3_user(void);
void z290_3(void)
{
 z290_3_user();

 #ifdef Z_LOGGING
   log_z(3);
 #endif
}
//---------------------------------------------------------------------------
```

z_user.cpp

```
#include "common.h"

//---------------------------------------------------------------------------
void z290_0_user(void)
{ // Индикация состояния 'ДГ остановлен'

  //---- Расположите Ваш код здесь ----
}
//---------------------------------------------------------------------------
void z290_1_user(void)
{ // Индикация состояния 'ДГ запущен'

  //---- Расположите Ваш код здесь ----
}
//---------------------------------------------------------------------------
void z290_2_user(void)
{ // Индикация состояния 'ДГ запускается'

  //---- Расположите Ваш код здесь ----
}
//---------------------------------------------------------------------------
void z290_3_user(void)
{ // Индикация состояния 'ДГ останавливается'

  //---- Расположите Ваш код здесь ----
}
//---------------------------------------------------------------------------
```