С авторами можно связаться по адресу: mail@avrorasystems.com, "для Шалыто".

Туккель Н.И., Шалыто А.А.

Система управления дизель-генератором (фрагмент)

Программирование с явным выделением состояний

Программная документация

Санкт-Петербург
2002

Содержание

Введение

Программное обеспечение системы управления дизель-генератором создано на основе подхода, изложенного в статье Шалыто А.А., Туккель Н.И. "SWITCH-технология — автоматный подход к созданию программного обеспечения событийных систем" (www.softcraft.ru).

Разработанная система предназначена для управления двумя дизель-генераторами, функционирующими по одинаковым алгоритмам. Система управления содержит около 50 дискретных входов, 50 аналоговых входов, 50 дискретных выходов, до 20 одновременно активных выдержек времени и 5 видеокадров.

Для отладки разработанной системы был создан простейший программный имитатор объекта управления, также спроектированный с использованием предлагаемой технологии.

Система управления и имитатор являются отдельными программами и функционируют как параллельные процессы. В них могут быть использованы повторяющиеся обозначения, например при нумерации событий или автоматов.

Программы предназначены для функционирования под управлением операционной системы QNX 4.25 и графической оболочки Photon 1.14.

Программное обеспечение системы управления и имитатора построено в соответствии с предложенной авторами структурой событийных программ (рис. 1).

Как следует из этой структуры, программы, разрабатываемые по предлагаемой технологии, состоят из системонезависимой и системозависимой частей. Системонезависимую часть образует система взаимосвязанных автоматов. Системозависимая часть состоит из модулей, реализующих обработчики событий, входные переменные, выходные воздействия и вспомогательные функции, в том числе функцию протоколирования.

Приведем перечень сокращений, которые используются в дальнейшем:

ГО - газоохладитель, газоотвод;
ГПК - главный пусковой клапан;
Д1СЧВ, Д2СЧВ, Д3СЧВ - датчики 1-ой, 2-ой и 3-ей ступеней частоты вращения;
ДВПМ - датчик валопроворотного механизма;
ДГ - дизель-генератор;
ДПКВ - датчик проворота коленвала;
ДППВ - датчик положения предельного выключателя;
ДЧВ - датчик частоты вращения;
МВП - механизм валопроворотный;
МПН - маслопрокачивающий насос;
ОКС - общекорабельная система;
ПРЕД - предельная частота вращения;
РЧВ - рабочая частота вращения, регулятор частоты вращения;
СУ - система управления;
ЧВГЗ - частота вращения открытия наружной захлопки газоотвода;
ЧВНЗ - частота вращения начала заливания;
ЧВО - частота вращения останова.

В настоящей работе не приведены следующие документы, содержащиеся в полной версии программной документации: структурная схема системы управления, отражающая ее приборный состав, и видеокадры операторского интерфейса системы.

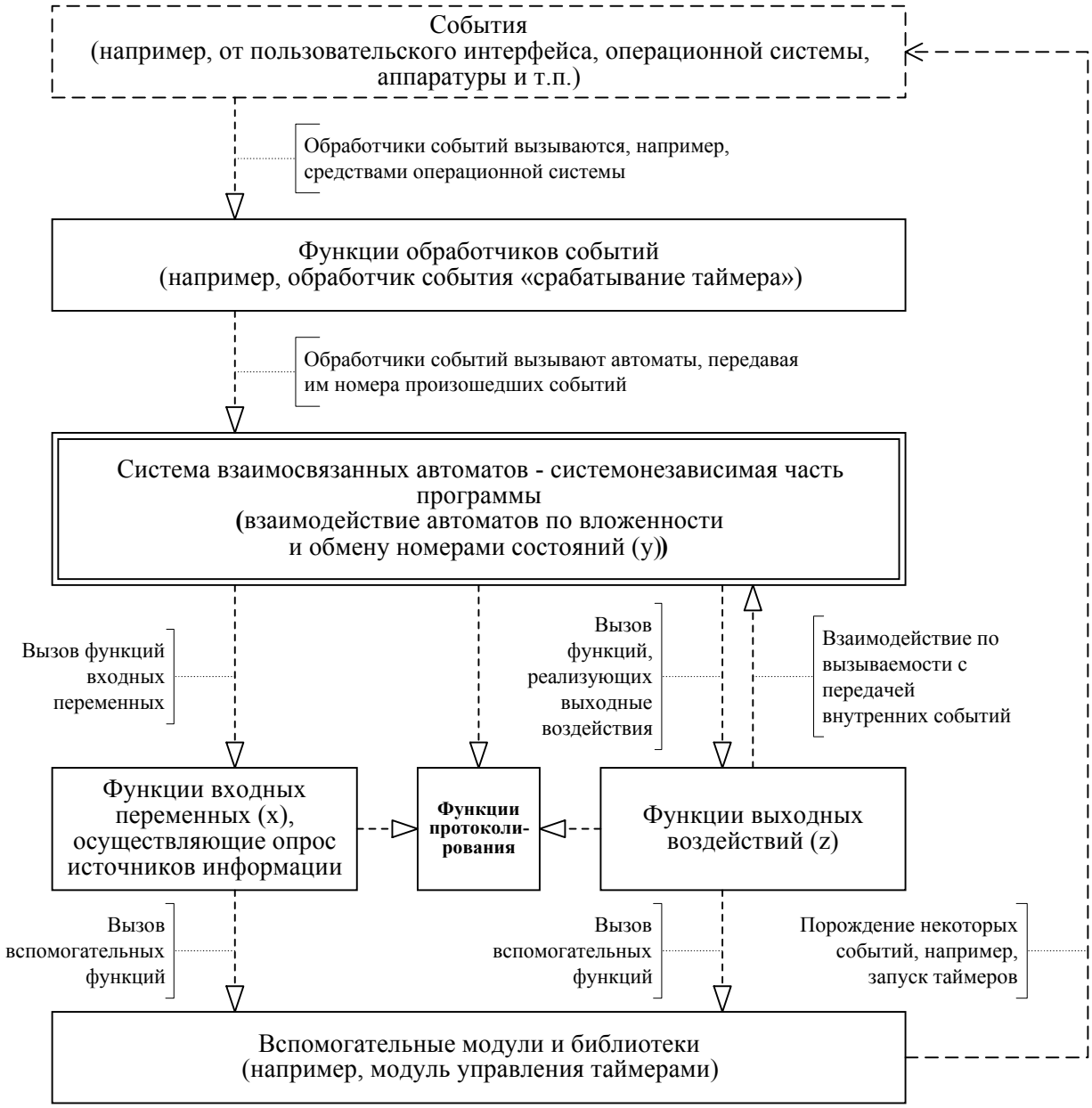

Рис. 1. Предлагаемая структура событийных программ

1. Система управления дизель-генератором

1.1. Структурная схема программного обеспечения

Разработанное программное обеспечение состоит из двух частей: системы управления и имитатора объекта управления (рис. 2).

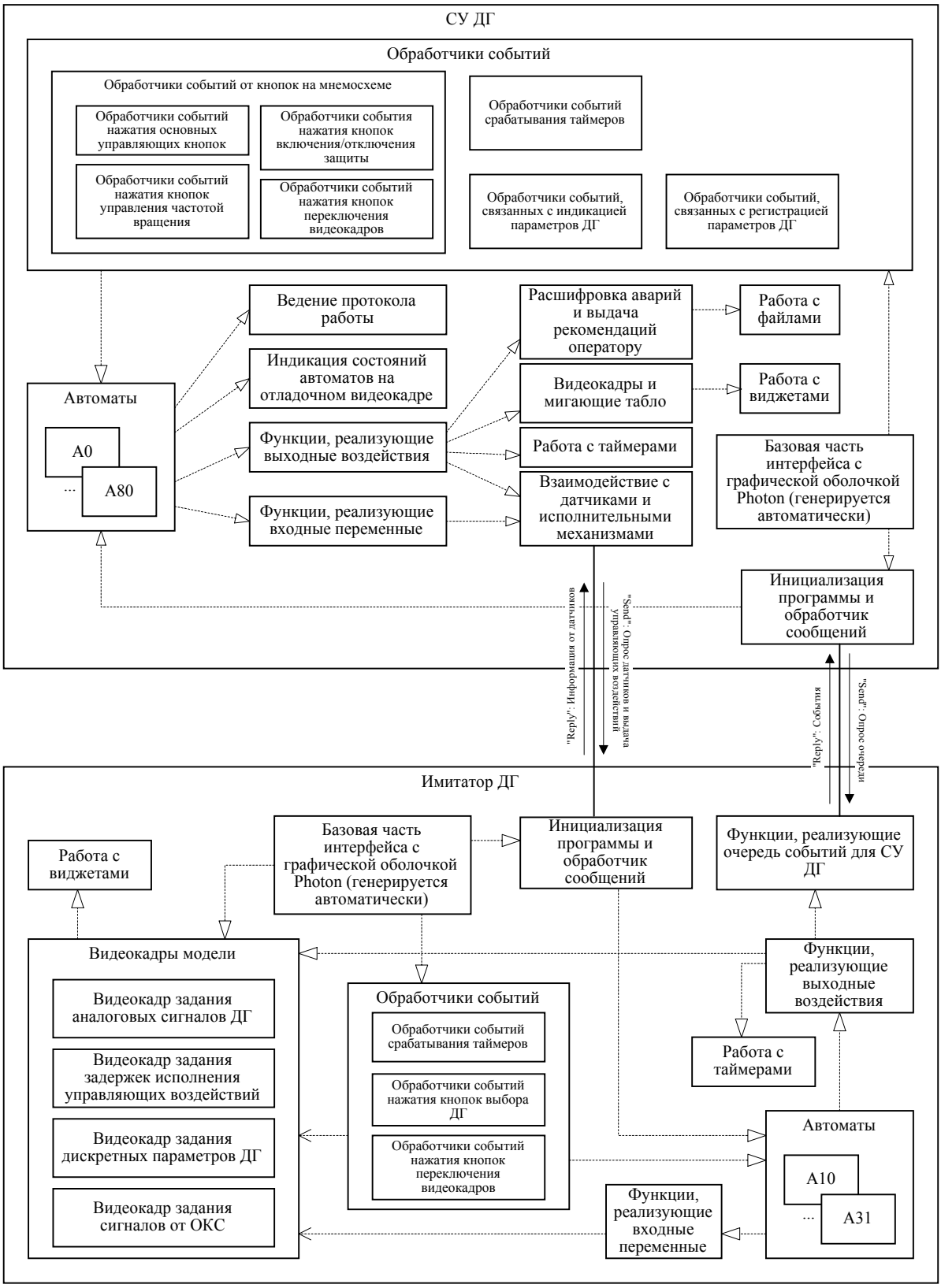

Рис. 2. Структурная схема программного обеспечения

1.2. Перечень событий

Перечислим названия событий (e), на которые реагирует система управления, и их номера.

Внешние от объекта управления

1	Включение СУ (отключение ручного управление)
2	Отключение СУ (включение ручного управления)
10	Нажатие кнопки ПОДГОТОВКА К ПУСКУ
20	Нажатие кнопки ПУСК
30	Нажатие кнопки ОСТАНОВ
40	Нажатие кнопки ЭКСТРЕННЫЙ СТОП
50	Нажатие кнопки КВИТИРОВАНИЕ
60	Нажатие кнопки РАЗБЛОКИРОВКА
110	Появление сигнала МВП НЕ ОТКЛЮЧЕН
120	Появление сигнала ПРЕДЕЛЬНЫЙ ВЫКЛЮЧАТЕЛЬ
121	Исчезновение сигнала ПРЕДЕЛЬНЫЙ ВЫКЛЮЧАТЕЛЬ
130	Появление сигнала ВОДА В ОХЛАДИТЕЛЕ
140	Срабатывание уставки ЧВО (ЧВ < ЧВО)
150	Срабатывание уставки РЧВ (ЧВ > РЧВ)
151	Срабатывание уставки РЧВ (ЧВ < РЧВ)
160	Срабатывание уставки ЧВНЗ (ЧВ < ЧВНЗ)
170	Срабатывание уставки ЧВГЗ
171	Срабатывание уставки ПРЕД
180	Импульс от датчика проворота ДПКВ
190	Срабатывание датчика первой ступени частоты вращения Д1СЧВ
200	Срабатывание датчика второй ступени частоты вращения Д2СЧВ
210	Срабатывание датчика третьей ступени частоты вращения Д3СЧВ
270	Срабатывание уставки Рго
280	Срабатывание уставки Рмп (Рм > Рмп)
290	Снижение давления масла ниже уставки Рмп
600	Нажатие кнопки РАСШИФРОВКА
610	Нажатие кнопки выбора ДГ
620	Нажатие кнопки ПРЕДЫДУЩАЯ АВАРИЯ
630	Нажатие кнопки СЛЕДУЮЩАЯ АВАРИЯ
640	Нажатие кнопки ПОСЛЕДНЯЯ АВАРИЯ
700	Закрытие диалога расшифровки аварий

Внешние от смежных систем

70	Появление сигнала БЛОКИРОВКА ПУСКА из СУ ОКС
80	Снятие сигнала БЛОКИРОВКА ПУСКА из СУ ОКС
90	Появление сигнала ВНЕШНИЙ СТОП из СУ ОКС
100	Появление сигнала НАРУЖНАЯ ЗАХЛОПКА ГО ОТКРЫТА из СУ ОКС

Таймеры

220	Истечение времени проворота
230	Истечение времени пуска
240	Истечение времени на открытие наружной захлопки
250	Истечение времени подачи воды в сепаратор газоотвода
260	Истечение времени прокачки маслом
261	Истечение времени задержки контроля параметров
262	Истечение времени работы насоса охлаждения
263	Истечение времени остановки МПН
264	Истечение времени выдачи сигнала ПОЛНАЯ ОСТАНОВКА ДГ в генераторный щит
265	Истечение времени открытия/закрытия клапана
266	Истечение времени выдачи команды на запуск насоса охлаждения
360	Синхроимпульс для проверки условий переходов
520	Срабатывание таймера регистрации параметров ДГ
530	Срабатывание таймера индикации параметров ДГ

Внутренние

0	Инициализация вложенных автоматов
380	Команда на изменение частоты вращения
390	Команда на останов МПН
400	Команда на выдачу сигнала ПОЛНАЯ ОСТАНОВКА ДГ в генераторный щит
410	Открыть клапан (внутреннее для алгоритма управления клапанами)
420	Закрыть клапан (внутреннее для алгоритма управления клапанами)
500	Изменение периода регистрации параметров ДГ
510	Изменение периода индикации параметров ДГ

1.3. Перечень входных переменных

Перечислим названия входных переменных (X), которые опрашиваются системой управления, и их номера.

Внешние от органов управления

10 Переключатель ДИСТАНЦИОННОЕ-МЕСТНОЕ в положении «Дистанционное»
20 Переключатель ЗАЩИТА ОТКЛЮЧЕНА выключен (защита включена)

Внешние от объекта управления

210 Частота вращения больше уставки ЧВО
220 Частота вращения больше уставки РЧВ
240 Частота вращения меньше уставки ЧВНЗ
250 Частота вращения больше уставки ЧВГЗ
260 Частота вращения больше уставки ПРЕД
280 Первая ступень частоты вращения
290 Вторая ступень частоты вращения
300 Третья ступень частоты вращения
310 Давление газов в газоотводе больше уставки Рго
320 Температура масла меньше Тмм
330 Температура масла больше Тмпр
340 Температура масла больше Тма
350 Давление масла больше уставки Рмп
360 Давление масла меньше уставки Рмпр1
370 Давление масла меньше уставки Рмпр2
380 Давление масла меньше уставки Рмпр3
390 Давление масла меньше уставки Рма1
400 Давление масла меньше уставки Рма2
410 Давление масла меньше уставки Рма3
420 Давление воды меньше уставки Рвпр
430 Температура воды меньше уставки Твм
440 Температура воды больше уставки Твпр
450 Температура воды больше уставки Тва
460 Давление наддува больше уставки Рнпр
470 Температура выхлопных газов выше уставки Твг
480 Давление в отсеке ниже уставки Ротс
490 Давление в отсеке ниже уставки Ротс.а
500 Наличие сигнала ВОДА В ОХЛАДИТЕЛЕ
510 Наличие сигнала МВП НЕ ОТКЛЮЧЕН
520 Наличие сигнала ПРЕДЕЛЬНЫЙ ВЫКЛЮЧАТЕЛЬ
900 Клапан открыт
910 Клапан закрыт

Внешние от смежных систем

700 Наличие сигнала БЛОКИРОВКА ПУСКА из СУ ОКС

Внутренние

800 Два проворота вала
810 Заданная частота вращения выше текущей
820 Заданная частота вращения ниже текущей
901 Открыт клапан подачи воды в газоохладитель

1.4. Перечень выходных воздействий

Перечислим названия выходных воздействий (Z), формируемых системой управления, и их номера.

Внешние — индикация

10 Табло ПОДГОТОВКА К ПУСКУ (0 - отключить, 1 - включить).
Си-функции, реализующие это выходное воздействие называются z10_0() и z10_1() соответственно.
20 Табло ПРОКАЧКА МАСЛОМ (0 - отключить, 1 - включить)
30 Табло ПРОВОРОТ (0 - отключить, 1 - включить)
40 Табло НЕТ ПРОВОРОТА (0 - отключить, 1 - постоянное, 2 - мигающее)
50 Табло ВНЕШНИЙ СТОП (0 - отключить, 1 - постоянное, 2 - мигающее)
60 Табло ЭКСТРЕННЫЙ СТОП (0 - отключить, 1 - постоянное, 2 - мигающее)
70 Табло АВАРИЯ (0 - отключить, 1 - постоянное, 2 - мигающее)
80 Табло ПРЕДЕЛЬНЫЙ ВЫКЛЮЧАТЕЛЬ (0 - отключить, 1 - включить)
90 Табло НАСОС ОХЛАЖДЕНИЯ (0 - отключить, 1 - включить)

100 Табло ПУСК (0 - отключить, 1 - включить)
110 Табло ОСТАНОВКА (0 - отключить, 1 - включить)
120 Табло ПУСК РАЗРЕШЕН (0 - отключить, 1 - включить)
130 Табло НЕТ ПУСКА (0 - отключить, 1 - постоянное, 2 - мигающее)
140 Табло ХОЛОДНОЕ МАСЛО (0 - отключить, 1 - включить)
150 Табло ПЕРЕГРЕВ МАСЛА (0 - отключить, 1 - желтое постоянное, 2 - желтое мигающее, 3 - красное постоянное, 4 - красное мигающее)
160 Табло ДАВЛЕНИЕ МАСЛА (0 - отключить, 1 - желтое постоянное, 2 - желтое мигающее, 3 - красное постоянное, 4 - красное мигающее)
170 Табло ДАВЛЕНИЕ ВОДЫ (0 - отключить, 1 - желтое постоянное, 2 - желтое мигающее)
180 Табло ХОЛОДНАЯ ВОДА (0 - отключить, 1 - включить)
190 Табло ПЕРЕГРЕВ ВОДЫ (0 - отключить, 1 - желтое постоянное, 2 - желтое мигающее, 3 - красное постоянное, 4 - красное мигающее)
200 Табло НАДДУВ ВЫШЕ НОРМЫ (0 - отключить, 1 - желтое постоянное, 2 - желтое мигающее)
210 Табло ТЕМПЕРАТУРА ГАЗОВ ВЫШЕ НОРМЫ (0 - отключить, 1 - желтое постоянное, 2 - желтое мигающее)
220 Табло ДАВЛЕНИЕ В ОТСЕКЕ НИЖЕ НОРМЫ (0 - отключить, 1 - желтое постоянное, 2 - желтое мигающее, 3 - красное постоянное, 4 - красное мигающее)
230 Табло ВОДА В ОХЛАДИТЕЛЕ (0 - отключить, 1 - красное постоянное, 2 - красное мигающее)
280 Табло РАЗНОС (0 - отключить, 1 - красное постоянное, 2 - красное мигающее)
240 Перевести все аварийные табло в постоянное свечение
250 Погасить все аварийные табло
260 Обновить индикацию частоты вращения
270 Индикатор состояния клапана (0 - закрыт, 1 - открыт, 2 - закрывается/открывается, 3 - неисправен)
271 Индикатор состояния наружной захлопки (0 - закрыт, 1 - открыт, 2 - закрывается/открывается, 3 - неисправен)
290 Индикатор состояния ДГ (0 - остановлен, 1 - запущен, 2 - запускается, 3 - останавливается)
700 Обобщенная звуковая сигнализация (0 - отключить, 1 - включить)
1140 Индикация параметров ДГ
1141 Регистрация параметров ДГ
1200 Показать окно расшифровки аварий, установив выбранный ДГ и очистив информационные поля
1210 Показать информацию о последней аварии
1220 Показать информацию о предыдущей аварии
1230 Показать информацию о следующей аварии
1240 Изменить состояние кнопки РАСШИФРОВКА (0 - отключить, 1 - включить)
1300 Добавить сообщение в очередь сообщений об авариях

Внешние — объект управления

400 Магнитный пускатель МПН (0 - отключить, 1 - включить)
411 Команда остановки МПН (0 - снять, 1 - выдать)
420 Электромагнит стопа РЧВ (0 - снять питание, 1 - подать питание)
430 Клапан подачи воздуха к ДГ (0 - закрыть, 1 - открыть)
440 Клапан подачи воздуха низкого давления (0 - закрыть, 1 - открыть)
450 Главный пусковой клапан (0 - закрыть, 1 - открыть)
480 Клапан подачи воды в газоохладитель (0 - закрыть, 1 - открыть)
490 Клапан слива воды из сепаратора газоотвода (0 - закрыть, 1 - открыть)
500 Клапан вентиляции сепаратора топлива (0 - закрыть, 1 - открыть)
510 Клапан подачи топлива от сепаратора топлива (0 - закрыть, 1 - открыть)
570 Электромагнит предельного выключателя (0 - снять питание, 1 - подать питание)
590 Насос охлаждения (0 - остановить, 1 - выдать команду запуск, 2 - снять команду запуска)
610 Двигатель регулятора частоты вращения (0 - остановить, 1 - увеличивать частоту, 2 - уменьшать частоту)

Внешние — смежные системы

710 Выдать сигнал ПОДГОТОВКА К ПУСКУ в СУ ОКС
720 Выдать сигнал ОТКРЫТЬ НАРУЖНУЮ ЗАХЛОПКУ в СУ ОКС
730 Выдать сигнал ЗАКРЫТЬ НАРУЖНУЮ ЗАХЛОПКУ в СУ ОКС
740 Выдать сигнал ПОЛНАЯ ОСТАНОВКА ДГ в СУ ОКС
751 Сигнал ПОЛНАЯ ОСТАНОВКА ДГ в генераторный щит (0 - снять, 1 - выдать)

Таймеры

810 Таймер контроля времени проворота 60 с (0 - сбросить, 1 - запустить)

820 Таймер контроля времени открытия наружной захлопки 5 с (0 - сбросить, 1 - запустить)
830 Таймер контроля времени подачи воды 5 с (0 - сбросить, 1 - запустить)
840 Таймер контроля времени прокачки маслом 60 с (0 - сбросить, 1 - запустить)
850 Таймер контроля времени охлаждения 180 с (0 - сбросить, 1 - запустить)
860 Таймер контроля времени пуска 15 с (0 - сбросить, 1 - запустить)
880 Таймер задержки контроля параметров 10 с (0 - сбросить, 1 - запустить)
890 Таймер контроля времени остановки МПН 4 с (0 - сбросить, 1 - запустить)
900 Таймер контроля времени выдачи сигнала ПОЛНАЯ ОСТАНОВКА ДГ 3 с (0 - сбросить, 1 - запустить)
950 Таймер контроля времени открытия/закрытия клапана (0 - сбросить, 1 - запустить)
1100 Таймер опроса датчика давления в отсеке 0.5 с (0 - сбросить, 1 - запустить)
1110 Таймер периода регистрации параметров ДГ (0 - сбросить, 1 - запустить)
1120 Таймер периода индикации параметров ДГ (0 - сбросить, 1 - запустить)
1130 Таймер контроля времени выдачи команды запуска насоса охлаждения (0 - сбросить, 1 - запустить)

Внутренние

750 Выдача сигнала ПОЛНАЯ ОСТАНОВКА ДГ в генераторный щит
410 Остановка МПН на 4 с
460 Выдать команду на установку частоты вращения 1-ой ступени
800 Счетчик проворотов (0 - сбросить, 1 - увеличить)
1000 Команда на клапан (0 - закрыть, 1 - открыть)

Инициализация

600 Инициализация всех исполнительных механизмов
870 Инициализация всех управляющих автоматов

1.5. Схема взаимодействия автоматов

Как отмечалось выше, системонезависимая часть состоит из системы взаимосвязанных автоматов, взаимодействие которых отражено на схеме (рис. 3), которая напоминает диаграмму классов, создаваемую при объектно-ориентированном проектировании. Отметим, что число автоматов в этой схеме — 31.

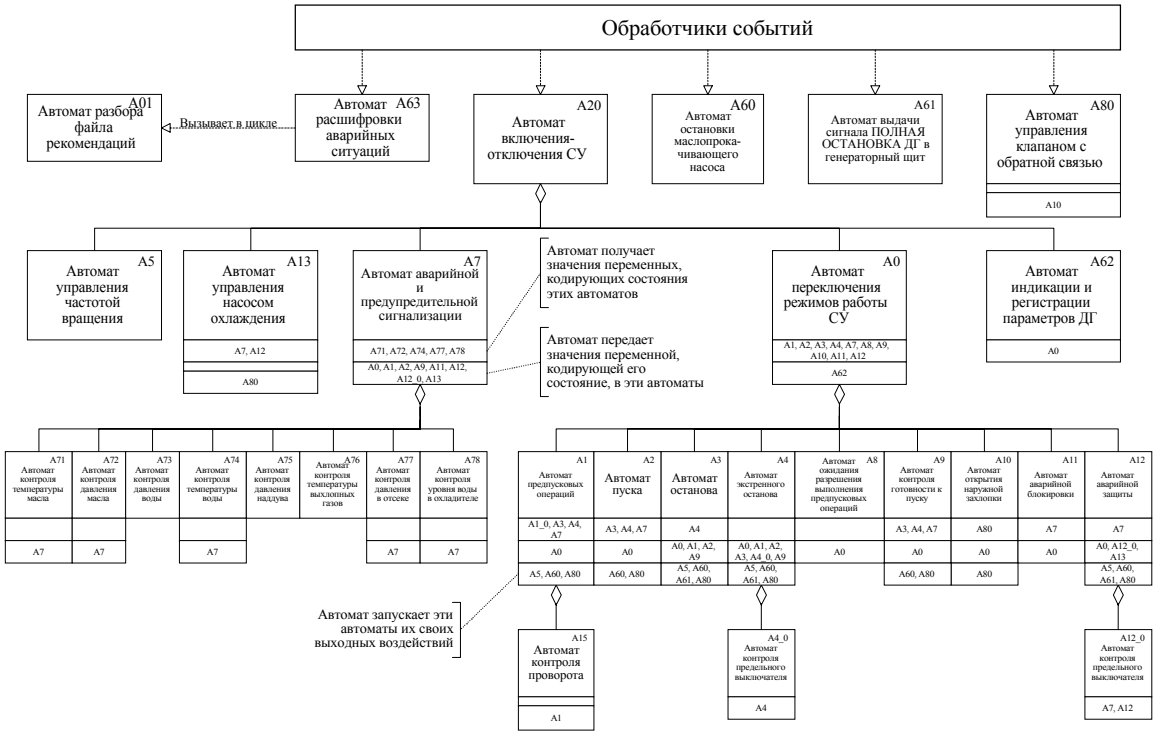

Рис. 3. Схема взаимодействия автоматов

В силу большого количества автоматов в этой схеме, дальнейшее изложение проведем на ее фрагменте, показанном на рис. 4. В этой схеме, также, как и в предыдущей, автоматы взаимодействуют по вложенности, вызываемости и обмену номерами состояний.

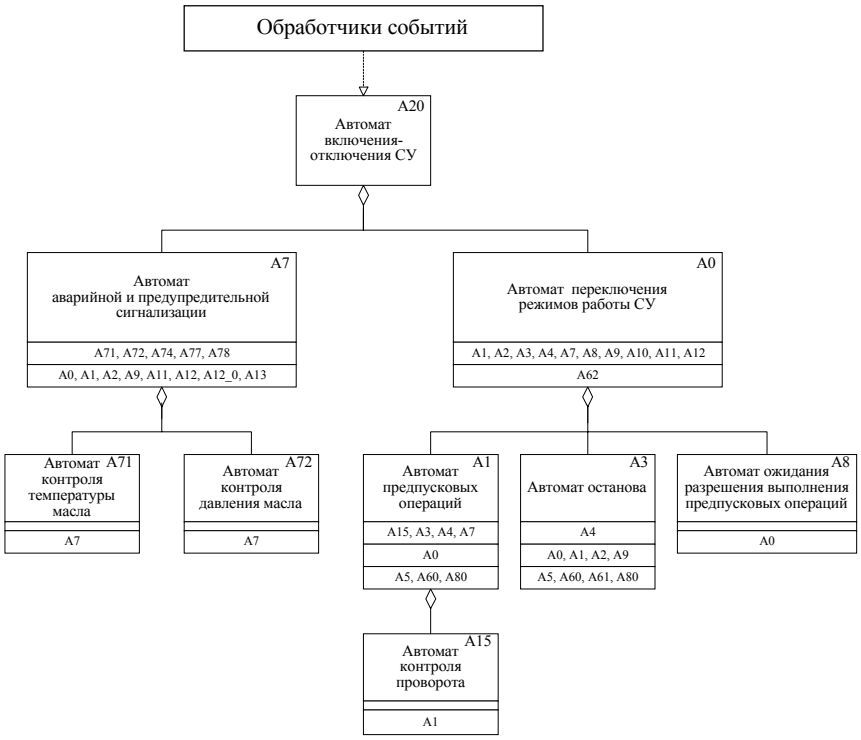

Рис. 4. Фрагмент схемы взаимодействия автоматов

Для каждого из автоматов, приведенных на этой схеме, разработаны четыре документа: словесное описание; схема связей; граф переходов; текст функции, реализующей автомат. При этом для автоматов, реализующих режимы работы, явно описанные в техническом задании, в качестве словесного описания приводятся его соответствующие фрагменты. Выполнение указанными автоматами, построенными "по мотивам" этих описаний, требований технического задания подтверждается протоколами работы системы. Пример одного из таких протоколов приведен в конце раздела 1.

1.6. Автомат включения-отключения системы управления

1.6.1. Словесное описание

Автомат, реализующий данный алгоритм является головным в схеме взаимодействия автоматов. Этот автомат обеспечивает выполнение перечисленных ниже действий и вызов вложенных автоматов.

1. При запуске программы выполняется инициализация всех органов управления.

2. При переводе переключателя "дистанционное/местное" в положении "местное" автомат выполняет инициализацию исполнительных механизмов и переходит в отключенное состояние.

3. При переводе переключателя "дистанционное/местное" в положении "дистанционное" автомат запускает генератор синхроимпульсов (период 0.5 с) и начинает выполнение алгоритма управления.

При этом:

– при нажатии кнопки "Квитирование" все мигающие табло переводятся в постоянное свечение и отключается обобщенная звуковая сигнализация;

– при появлении сигнала "Предельный выключатель" включается соответствующее табло;

– при исчезновении сигнала "Предельный выключатель" выключается соответствующее табло.

Схема связей автомата и граф переходов приведены на рис. 5, 6.

1.6.2. Схема связей

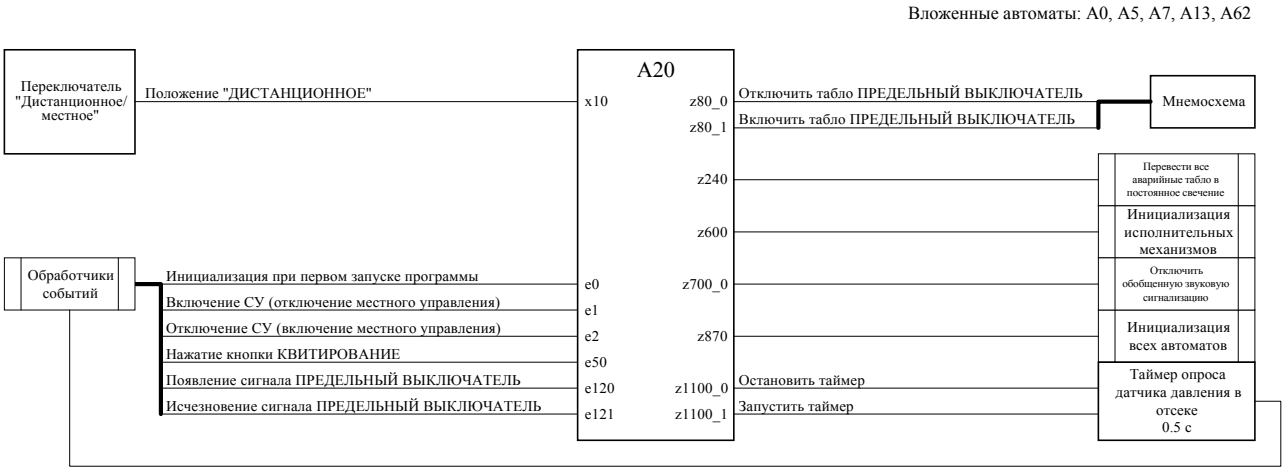

Рис. 5. Схема связей автомата включения-отключения системы управления

1.6.3. Граф переходов

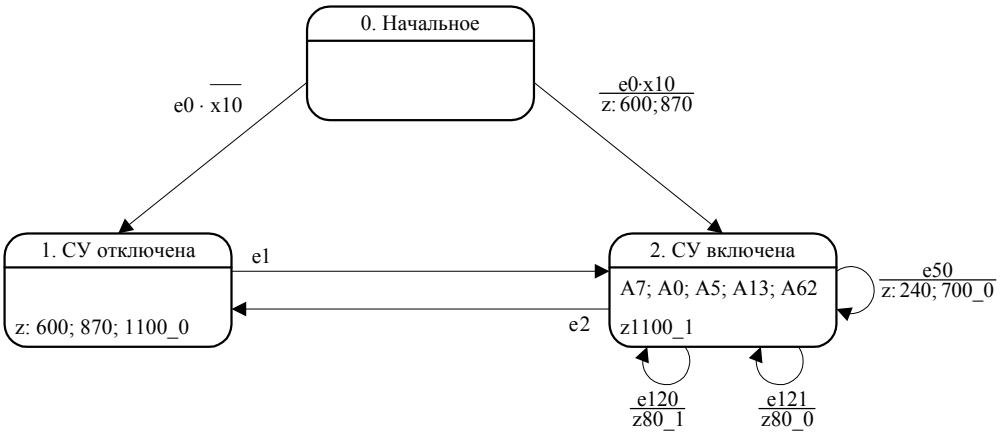

Рис. 6. Граф переходов автомата включения-отключения системы управления

1.6.4. Текст функции, реализующей автомат

```
#include "photon_stuff.h"
#include "dg.h"
#include "log.h"
#include "defines.h"

void A20( int e, dg_t *dg )
{
  int y_old = dg->y20 ;
```

```
  #ifdef HEAD_EVENTS_LOGGING
    log_exec( dg, "A20", y_old, e ) ;
  #endif

  switch( dg->y20 )
  {
    case 0:
      if( e == 0 && !x10(dg) )                          dg->y20 = 1 ;
      else
      if( e == 0 && x10(dg) ) { z600(dg) ; z870(dg) ;  dg->y20 = 2 ; } ;
    break ;

    case 1:
      if( e == 1 )                                      dg->y20 = 2 ;
    break ;

    case 2:
      A7( e, dg ) ; A0( e, dg ) ; A5( e, dg ) ; A13( e, dg ) ; A62( e, dg ) ;
      if( e == 2 )                                      dg->y20 = 1 ;
      else
      if( e == 50 )  { z240(dg) ; z700_0(dg) ; }
      else
      if( e == 121 ) { z80_0(dg) ; }
      else
      if( e == 120 ) { z80_1(dg) ; } ;
    break ;

    default:
      #ifdef GRAPH_ERRORS_LOGGING
        log_write( LOG_GRAPH_ERROR, dg->number,
                   "ERROR IN A20: unknown state number!", 0 ) ;
      #endif
    break ;
  } ;

  if( y_old == dg->y20 ) goto A20_end ;

  {
    #ifdef GRAPH_TRANS_LOGGING
      log_trans( dg, "A20", y_old, dg->y20 ) ;
    #endif

    #ifdef DEBUG_FRAME
      update_debug() ;
    #endif
  } ;

  switch( dg->y20 )
  {
    case 1:
      z600(dg) ; z870(dg) ; z1100_0(dg) ;
    break ;

    case 2:
      A7( 0, dg ) ; A0( 0, dg ) ; A5( 0, dg ) ; A13( 0, dg ) ; A62( 0, dg ) ;
      z1100_1(dg) ;
    break ;

  } ;

  A20_end:
  #ifdef HEAD_ENDS_LOGGING
    log_end( dg, "A20", dg->y20, e ) ;
  #endif
  ;
} ;
```

1.7. Автомат переключения режимов работы системы управления

1.7.1. Словесное описание

Автомат переключения режимов работы реализует основной алгоритм управления. Автоматы, вложенные в его состояния,

обеспечивают детализацию алгоритма работы в соответствующих режимах. Перечислим эти режимы.

1. Дизель остановлен. В рассматриваемом режиме осуществляется проверка условий начала выполнения алгоритма предпусковых операций. Если при нахождении системы управления в этом режиме дизель-генератор был запущен в обход ее, то система отреагирует на это переходом в установившийся режим. Срабатывание одного из алгоритмов аварийной и предупредительной сигнализации в этом режиме приводит к переходу системы в режим аварийной блокировки.

2. Предпусковые операции. В рассматриваемом режиме выполняется алгоритм предпусковых операций.

3. К пуску готов. Режим соответствует удачному завершению выполнения алгоритма предпусковых операций. В этом режиме система проверяет сохранились ли условия, разрешающие пуск. При нажатии оператором кнопки "Пуск" начинается выполнение алгоритма пуска.

4. Выполняется пуск. В рассматриваемом режиме выполняется алгоритм пуска, при успешном осуществлении которого система переходит в установившийся режим.

5. Установившийся режим. В этом режиме при необходимости выполняется алгоритм открытия наружной захлопки, реализуемый автоматом А10. Выход из этого режима происходит в начале выполнения одного из видов останова, при срабатывании алгоритма аварийной защиты или при останове дизель-генератора в обход системы управления.

6. Останов. В рассматриваемом режиме выполняется алгоритм останова. Указанный алгоритм может быть прерван в случае экстренного останова.

7. Экстренный останов. В рассматриваемом режиме выполняется алгоритм экстренного останова. После завершения этого алгоритма система переходит в режим аварийной блокировки.

8. Аварийная блокировка. В этом режиме система ждет пока оператор подтвердит свою реакцию на аварию нажатием кнопки "Квитирование" и, устранив аварию, нажмет кнопку "Разблокировка".

9. Аварийная защита. Этот режим аналогичен режиму экстренного останова.

Схема связей автомата и граф переходов приведены на рис. 7, 8.

1.7.2. Схема связей

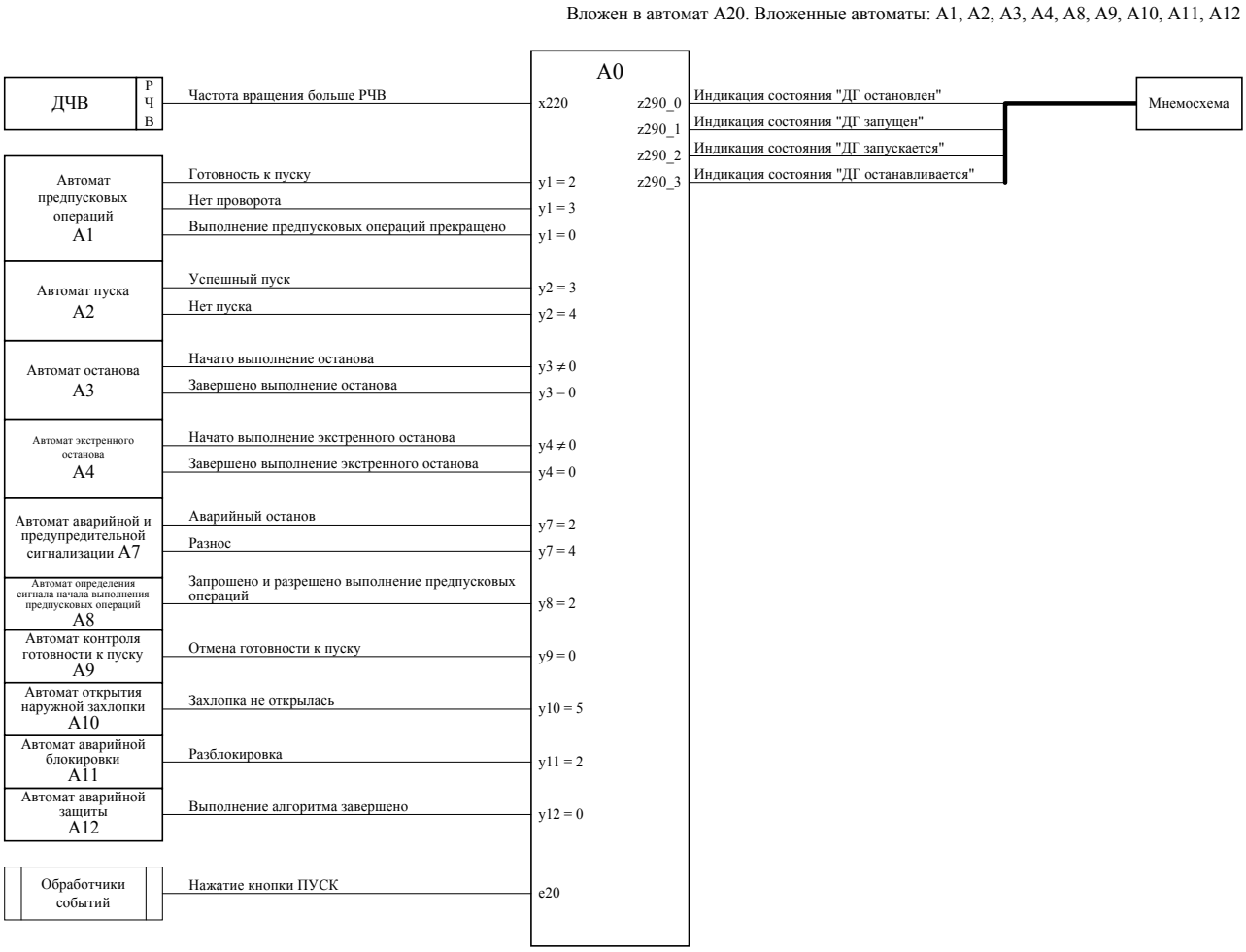

Рис. 7. Схема связей автомата переключения режимов работы системы управления

1.7.3. Граф переходов

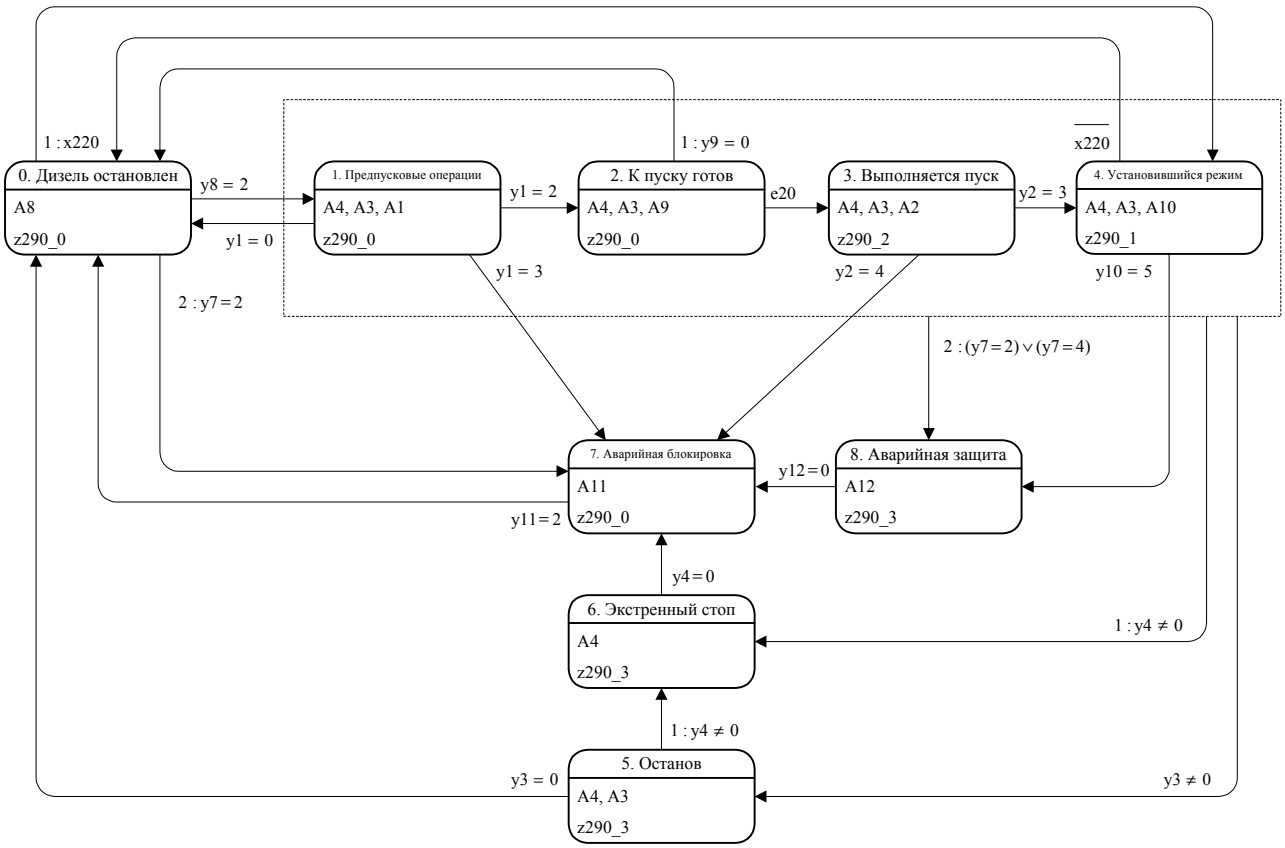

Рис. 8. Граф переходов автомата переключения режимов работы системы управления

1.7.4. Текст функции, реализующей автомат

```
#include "photon_stuff.h"
#include "dg.h"
#include "log.h"
#include "defines.h"

void A0( int e, dg_t *dg )
{
  int y_old = dg->y0 ;

  #ifdef A0_BEGIN_LOGGING
    log_begin( dg, "A0", y_old, e ) ;
  #endif

  switch( dg->y0 )
  {
    case 0:
      A8( e, dg ) ;
      if( x220(dg) )                    dg->y0 = 4 ;
      else
      if( dg->y7 == 2 )                 dg->y0 = 7 ;
      else
      if( dg->y8 == 2 )                 dg->y0 = 1 ;
    break ;

    case 1:
      A4( e, dg ) ; A3( e, dg ) ; A1( e, dg ) ;
      if( dg->y4 != 0 )                 dg->y0 = 6 ;
      else
      if( dg->y7 == 2 || dg->y7 == 4 )  dg->y0 = 8 ;
      else
      if( dg->y3 != 0 )                 dg->y0 = 5 ;
      else
      if( dg->y1 == 0 )                 dg->y0 = 0 ;
```

```
    else
    if( dg->y1 == 3 )                dg->y0 = 7 ;
    else
    if( dg->y1 == 2 )                dg->y0 = 2 ;
  break ;

  case 2:
    A4( e, dg ) ; A3( e, dg ) ; A9( e, dg ) ;
    if( dg->y4 != 0 )                dg->y0 = 6 ;
    else
    if( dg->y7 == 2 || dg->y7 == 4 )  dg->y0 = 8 ;
    else
    if( dg->y3 != 0 )                dg->y0 = 5 ;
    else
    if( dg->y9 == 0 )                dg->y0 = 0 ;
    else
    if( e == 20 )                    dg->y0 = 3 ;
  break ;

  case 3:
    A4( e, dg ) ; A3( e, dg ) ; A2( e, dg ) ;
    if( dg->y4 != 0 )                dg->y0 = 6 ;
    else
    if( dg->y7 == 2 || dg->y7 == 4 )  dg->y0 = 8 ;
    else
    if( dg->y3 != 0 )                dg->y0 = 5 ;
    else
    if( dg->y2 == 4 )                dg->y0 = 7 ;
    else
    if( dg->y2 == 3 )                dg->y0 = 4 ;
  break ;

  case 4:
    A4( e, dg ) ; A3( e, dg ) ; A10( e, dg ) ;
    if( dg->y4 != 0 )                dg->y0 = 6 ;
    else
    if( dg->y7 == 2 || dg->y7 == 4 )  dg->y0 = 8 ;
    else
    if( dg->y3 != 0 )                dg->y0 = 5 ;
    else
    if( dg->y10 == 5 )               dg->y0 = 8 ;
    else
    if( !x220(dg) )                  dg->y0 = 0 ;
  break ;

  case 5:
    A4( e, dg ) ; A3( e, dg ) ;
    if( dg->y4 != 0 )                dg->y0 = 6 ;
    else
    if( dg->y3 == 0 )                dg->y0 = 0 ;
  break ;

  case 6:
    A4( e, dg ) ;
    if( dg->y4 == 0 )                dg->y0 = 7 ;
  break ;

  case 7:
    A11( e, dg ) ;
    if( dg->y11 == 2 )               dg->y0 = 0 ;
  break ;

  case 8:
    A12( e, dg ) ;
    if( dg->y12 == 0 )               dg->y0 = 7 ;
  break ;

  default:
    #ifdef A0_ERRORS_LOGGING
      log_write( LOG_GRAPH_ERROR, dg->number,
                 "Ошибка в автомате A0: неизвестный номер состояния!", 0 ) ;
    #endif
} ;
```

```
  if( y_old == dg->y0 ) goto A0_end ;

  {
    #ifdef A0_TRANS_LOGGING
      log_trans( "A0", y_old, dg->y0 ) ;
    #endif

    #ifdef DEBUG_FRAME
      update_debug() ;
    #endif
  } ;

  switch( dg->y0 )
  {
    case 0:
      A8( 0, dg ) ;
      z290_0(dg) ;
    break ;

    case 1:
      A4( 0, dg ) ; A3( 0, dg ) ; A1( 0, dg ) ;
      z290_0(dg) ;
    break ;

    case 2:
      A4( 0, dg ) ; A3( 0, dg ) ; A9( 0, dg ) ;
      z290_0(dg) ;
    break ;

    case 3:
      A4( 0, dg ) ; A3( 0, dg ) ; A2( 0, dg ) ;
      z290_2(dg) ;
    break ;

    case 4:
      A4( 0, dg ) ; A3( 0, dg ) ; A10( 0, dg ) ;
      z290_1(dg) ;
    break ;

    case 5:
      A4( 0, dg ) ; A3( 0, dg ) ;
      z290_3(dg) ;
    break ;

    case 6:
      A4( 0, dg ) ;
      z290_3(dg) ;
    break ;

    case 7:
      A11( 0, dg ) ;
      z290_0(dg) ;
    break ;

    case 8:
      A12( 0, dg ) ;
      z290_3(dg) ;
    break ;
  } ;

  A0_end: ;
  #ifdef A0_END_LOGGING
    log_end( dg, "A0", dg->y0, e ) ;
  #endif
} ;
```

1.8. Автомат предпусковых операций

1.8.1. Словесное описание

1. Перечислим сигналы, блокирующие проведение предпусковых операций.

1.1. Обобщенный сигнал "Блокировка пуска" из СУ ОКС. При наличии этого сигнала выдается сигнал "Подготовка пуска" в указанную систему.

1.2. Сигнал "Механизм валопроворотный не отключен".

1.3. Сигнал "Предельный выключатель".

1.4. Дизель не остановлен — частота вращения превышает рабочую частоту.

1.5. Команда на выполнение любого вида останова.

1.6. Наличие сигнала "Вода в охладителе" (аварийный уровень).

2. При наличии хотя бы одного из сигналов, указанных в п.п.1.2...1.6, система управления не должна принимать управляющую команду подготовки к пуску.

3. При нажатии оператором кнопки "Подготовка к пуску" выдается сигнал в СУ ОКС. После снятия сигнала "Блокировка пуска" из СУ ОКС и при отсутствии блокирующих сигналов по п.п. 1.2...1.6, система должна выполнить перечисленные ниже действия.

3.1. Запомнить команду на подготовку к пуску.

3.2. Включить табло "Подготовка к пуску".

3.3. Выдать команду на включение маслопрокачивающего насоса, замкнув контакт в цепи его магнитного пускателя.

3.4. Включить табло "Прокачка маслом" при замыкании контакта пускателя маслопрокачивающего насоса.

3.5. Подать питание на электромагнит стопа регулятора частоты вращения.

3.6. Подать команду на открытие клапана подачи воздуха к дизель-генератору.

3.7. Подать команду на открытие клапана воздуха низкого давления.

3.8. Включить табло "Проворот".

3.9. Включить контроль времени проворота (60 с).

3.10. Выдать команду в регулятор частоты вращения на установку первой ступени частоты, и через 4 с снять эту команду.

4. После получения 3-х импульсов от датчика проворота коленчатого вала, до истечения времени по п. 3.9, система управления должна выполнить перечисленные ниже действия.

4.1. Снять питание с электромагнита клапана воздуха низкого давления.

4.2. Отключить табло "Проворот".

4.3. Отключить контроль времени проворота (60 с).

5. При превышении давлением масла предпускового давления и отработки команды на уменьшение заданной частоты вращения до первой ступени, система управления должна выполнить перечисленные ниже действия.

5.1. Снять команду на включение маслопрокачивающего насоса.

5.2. Снять команды на подготовку к пуску, перечисленные в п.3.

5.3. Отключить табло "Подготовка к пуску".

5.4. Включить табло "Пуск разрешен".

5.5. Включить память завершения предпусковых операций и дать разрешение на выполнение пуска.

6. В случае снижения давления масла ниже предпускового давления или увеличения затяжки пружины регулятора частоты вращения до третьей позиции или появления блокирующих сигналов по п. 1.1—1.3, 1.5, 1.6 система управления должна выполнить перечисленные ниже действия.

6.1. Отключить память завершения предпусковых операций.

6.2. Отключить табло "Пуск разрешен".

6.3. Разомкнуть контакт остановки маслопрокачивающего насоса и через 4 с вновь замкнуть его.

6.4. Снять питание с электромагнита остановки регулятора частоты вращения.

6.5. Отключить табло "Прокачка маслом" при размыкании контакта пускателя маслопрокачивающего насоса.

6.6. Снять команду на открытие клапана подачи воздуха к дизель-генератору.

7. Если по истечении времени (п. 3.9) от датчика проворота коленвала не поступило 3-х импульсов, то система управления должна выполнить перечисленные ниже действия.

7.1. Включить аварийное табло "Нет проворота".

7.2. Включить обобщенный сигнал звуковой сигнализации.

7.3. Отключить табло "Подготовка к пуску".

7.4. Снять питание с электромагнита клапана пускового воздуха низкого давления.

7.5. Отключить контроль времени проворота (60 с).

7.6. Снять команду на включение маслопрокачивающего насоса.

7.7. Разомкнуть контакт остановки маслопрокачивающего насоса и через 4 с вновь замкнуть этот контакт.

7.8. Отключить табло "Прокачка маслом" при размыкании контакта пускателя маслопрокачивающего насоса.

7.9. Отключить память команды на подготовку к пуску.

7.10. Снять команду на открытие клапана подачи воздуха к дизель-генератору.

8. Отключение обобщенной звуковой сигнализации и перевод аварийного табло "Нет проворота" в постоянное свечение производится нажатием кнопки "Квитирование".

9. Разблокировка системы управления и отключение табло "Нет проворота" производится нажатием кнопки "Разблокировка".

Схема связей автомата и граф переходов приведены на рис. 9, 10.

1.8.2. Схема связей

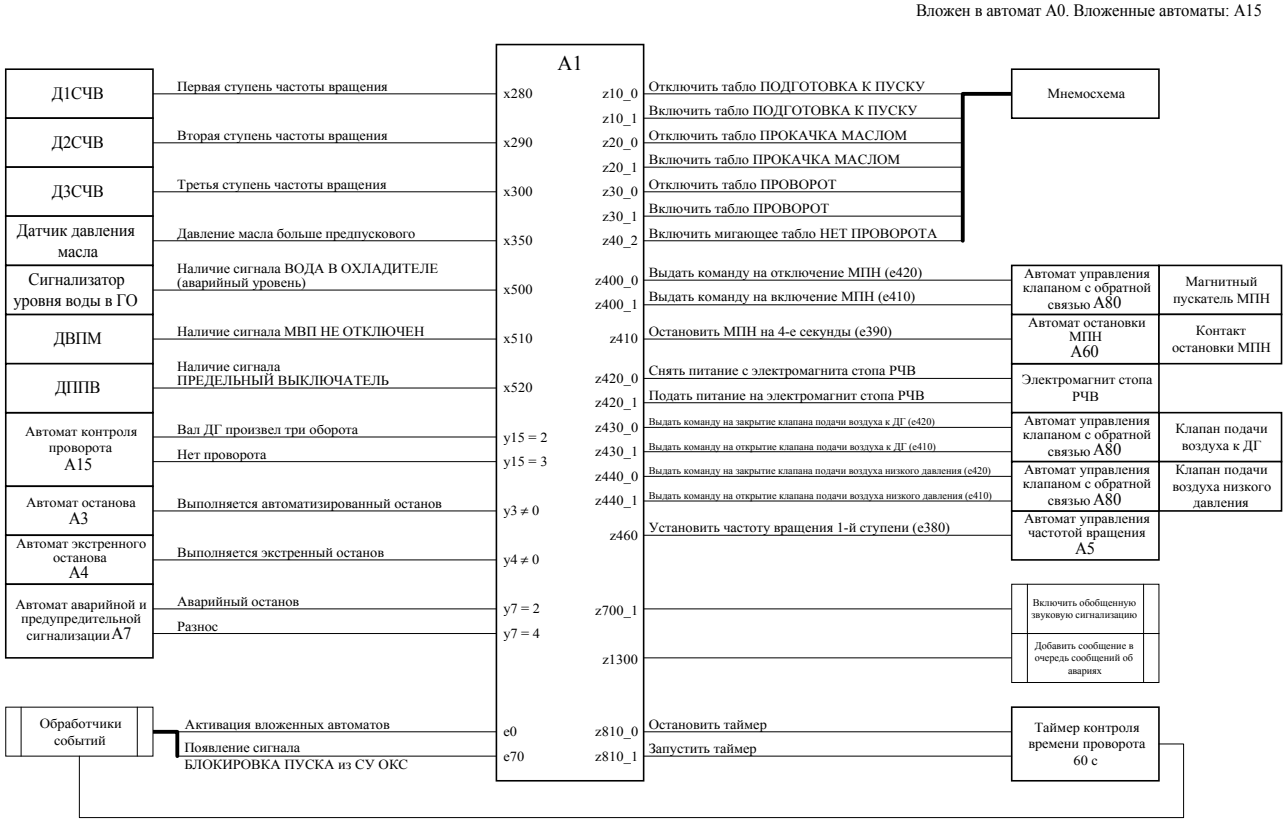

Рис. 9. Схема связей автомата предпусковых операций

1.8.3. Граф переходов

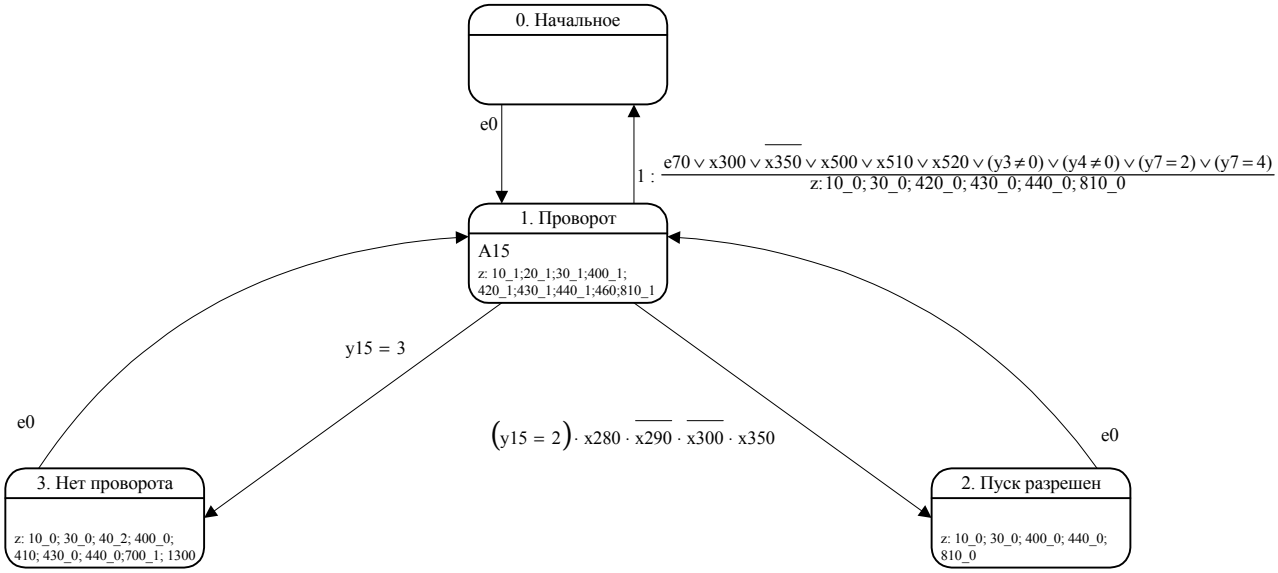

Рис. 10. Граф переходов автомата предпусковых операций

1.8.4. Текст функции, реализующей автомат

```
#include "photon_stuff.h"
#include "dg.h"
#include "log.h"
#include "defines.h"

void A1( int e, dg_t *dg )
{
  int y_old = dg->y1 ;

  #ifdef GRAPH_EVENTS_LOGGING
    log_exec( dg, "A1", y_old, e ) ;
  #endif

  switch( dg->y1 )
  {
    case 0:
      if( e == 0 )
                                                dg->y1 = 1 ;
    break ;

    case 1:
      A1_0( e, dg ) ;
      if(    e == 70 || x300(dg) || !x350(dg) || x500(dg) || x510(dg) || x520(dg)
          || dg->y3 != 0 || dg->y4 != 0 || dg->y7 == 2 || dg->y7 == 4 )
      { z10_0(dg) ; z30_0(dg) ; z420_0(dg) ; z430_0(dg) ; z440_0(dg) ; z810_0(dg) ;
                                                dg->y1 = 0 ; }
      else
      if( dg->y1_0 == 3 )
                                                dg->y1 = 3 ;
      else
      if( dg->y1_0 == 2 && x350(dg) && x280(dg) && !x290(dg) && !x300(dg) )
                                                dg->y1 = 2 ;
    break ;

    case 2:
      if( e == 0 )
                                                dg->y1 = 1 ;
    break ;

    case 3:
      if( e == 0 )
                                                dg->y1 = 1 ;
    break ;
```

```
    default:
      #ifdef GRAPH_ERRORS_LOGGING
        log_write( LOG_GRAPH_ERROR, dg->number,
                   "ERROR IN A1: unknown state number!", 0 ) ;
      #endif
    break ;
  } ;

  if( y_old == dg->y1 ) goto A1_end ;

  {
    #ifdef GRAPH_TRANS_LOGGING
      log_trans( dg, "A1", y_old, dg->y1 ) ;
    #endif

    #ifdef DEBUG_FRAME
      update_debug() ;
    #endif
  } ;

  switch( dg->y1 )
  {
    case 1:
      A1_0( 0, dg ) ;
      z10_1(dg) ; z20_1(dg) ; z30_1(dg) ; z400_1(dg) ; z420_1(dg) ;
      z430_1(dg) ; z440_1(dg) ; z460(dg) ; z810_1(dg) ;
    break ;

    case 2:
      z10_0(dg) ; z30_0(dg) ; z400_0(dg) ; z440_0(dg) ; z810_0(dg) ;
    break ;

    case 3:
      z10_0(dg) ; z30_0(dg) ; z40_2(dg) ; z400_0(dg) ; z430_0(dg) ;
      z440_0(dg) ; z410(dg) ;
      z700_1(dg) ; z1300(dg, MSG_NO_TURN) ;
    break ;
  } ;

  A1_end:
  #ifdef GRAPH_ENDS_LOGGING
    log_end( dg, "A1", dg->y1, e ) ;
  #endif
  ;
} ;
```

1.9. Автомат контроля проворота

1.9.1. Словесное описание

Автомат контроля проворота реализует часть алгоритма предпусковых операций, который описан в предыдущем подразделе (например, в п.п. 4, 7).

Обратим внимание, что в качестве объекта управления этого автомата выступает не "физический" объект, а вычислительное устройство, реализуемое программно.

Схема связей автомата и граф переходов приведены на рис. 11, 12.

1.9.2. Схема связей

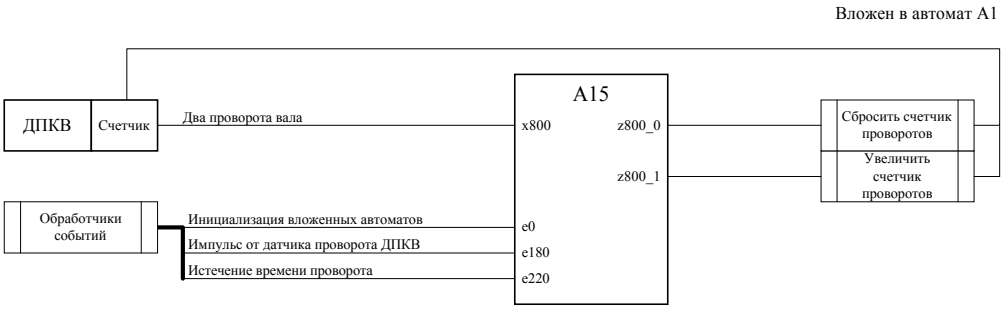

Рис. 11. Схема связей автомата контроля проворота

1.9.3. Граф переходов

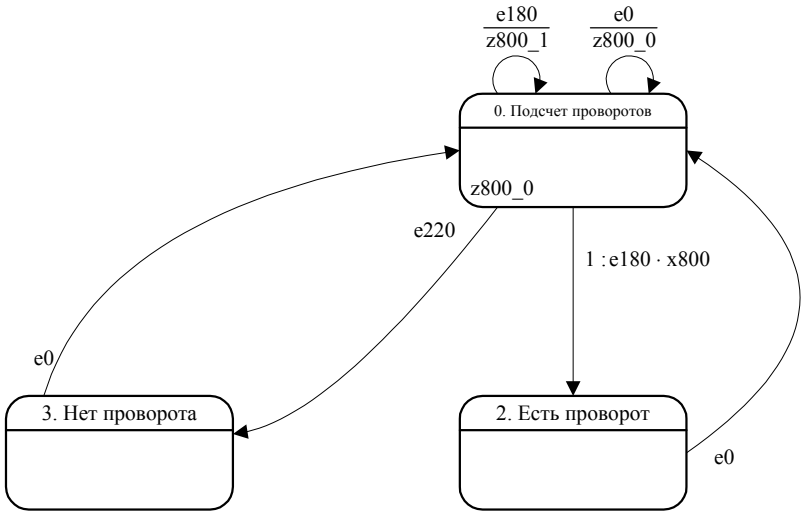

Рис. 12. Граф переходов автомата контроля проворота

1.9.4. Текст функции, реализующей автомат

```
#include "photon_stuff.h"
#include "dg.h"
#include "log.h"
#include "defines.h"

void A15( int e, dg_t *dg )
{
  int y_old = dg->y15 ;

  #ifdef GRAPH_EVENTS_LOGGING
    log_exec( dg, "A15", y_old, e ) ;
  #endif

  switch( dg->y15 )
  {
    case 0:
      if( e == 180 && x800(dg) )            dg->y15 = 2 ;
      else
      if( e == 220 )                        dg->y15 = 3 ;
      else
      if( e == 0 )   {     z800_0(dg) ; }
      else
      if( e == 180 ) {     z800_1(dg) ; } ;
    break ;

    case 2:
      if( e == 0 )                          dg->y15 = 0 ;
    break ;
```

```
    case 3:
      if( e == 0 )                          dg->y15 = 0 ;
    break ;

    default:
      #ifdef GRAPH_ERRORS_LOGGING
        log_write( LOG_GRAPH_ERROR, dg->number,
                   "ERROR IN A15: unknown state number!", 0 ) ;
      #endif
    break ;
  } ;

  if( y_old == dg->y15 ) goto A15_end ;

  {
    #ifdef GRAPH_TRANS_LOGGING
      log_trans( dg, "A15", y_old, dg->y15 ) ;
    #endif

    #ifdef DEBUG_FRAME
      update_debug() ;
    #endif
  } ;

  switch( dg->y15 )
  {
    case 0:
      z800_0(dg) ;
    break ;
  } ;

  A15_end:
  #ifdef GRAPH_ENDS_LOGGING
    log_end( dg, "A15", dg->y15, e ) ;
  #endif
  ;
} ;
```

1.10. Автомат останова

1.10.1. Словесное описание

1. При нажатии оператором кнопки "Останов" система управления должна выполнить перечисленные ниже действия.

1.1. Отключить память команды на подготовку к пуску и табло "Подготовка к пуску", память готовности к пуску и табло "Пуск разрешен", память команды пуск и табло "Пуск", если соответствующие операции в момент подачи команды выполнялись.

1.2. Включить память команды останов.

1.3. Включить табло "Останов".

1.4. Подать питание на электромагнит остановки регулятора частоты вращения.

1.5. Выдать команду на уменьшение затяжки пружины регулятора частоты вращения до величины соответствующей первой позиции и через 4 с снять эту команду.

2. При уменьшении частоты вращения ниже частоты "начала заливания" система управления должна выдать в СУ ОКС сигнал "Закрыть наружную захлопку".

3. При уменьшении частоты вращения ниже рабочей частоты система управления должна выполнить перечисленные ниже действия.

3.1. Выдать команду на включение маслопрокачивающего насоса.

3.2. Включить отсчет времени прокачки маслом (60 с).

3.3. Включить табло "Прокачка маслом".

4. При уменьшении частоты вращения ниже частоты вращения останова система управления должна выполнить перечисленные ниже действия.

4.1. Выдать в СУ ОКС сигнал "Полная остановка ДГ".

4.2. Выдать в генераторный щит сигнал "Полная остановка ДГ" длительностью 3 с.

4.3. Выдать команду на закрытие клапана подачи воды в газоохладитель и на закрытие клапана слива воды из сепаратора газоотвода.

5. Через установленное время (п. 3.2) после запуска маслопрокачивающего насоса система управления должна выполнить перечисленные ниже действия.

5.1. Снять команду на включение маслопрокачивающего насоса.

5.2. Разомкнуть контакт остановки маслопрокачивающего насоса и через 4 с вновь замкнуть этот контакт.

5.3. Отключить табло "Прокачка маслом" при размыкании контакта пускателя маслопрокачивающего насоса.

5.4. Снять питание с электромагнита остановки регулятора частоты вращения.

5.5. Отключить память команды останов.

5.6. Отключить табло "Останов".

Схема связей автомата и граф переходов приведены на рис. 13, 14.

1.10.2. Схема связей

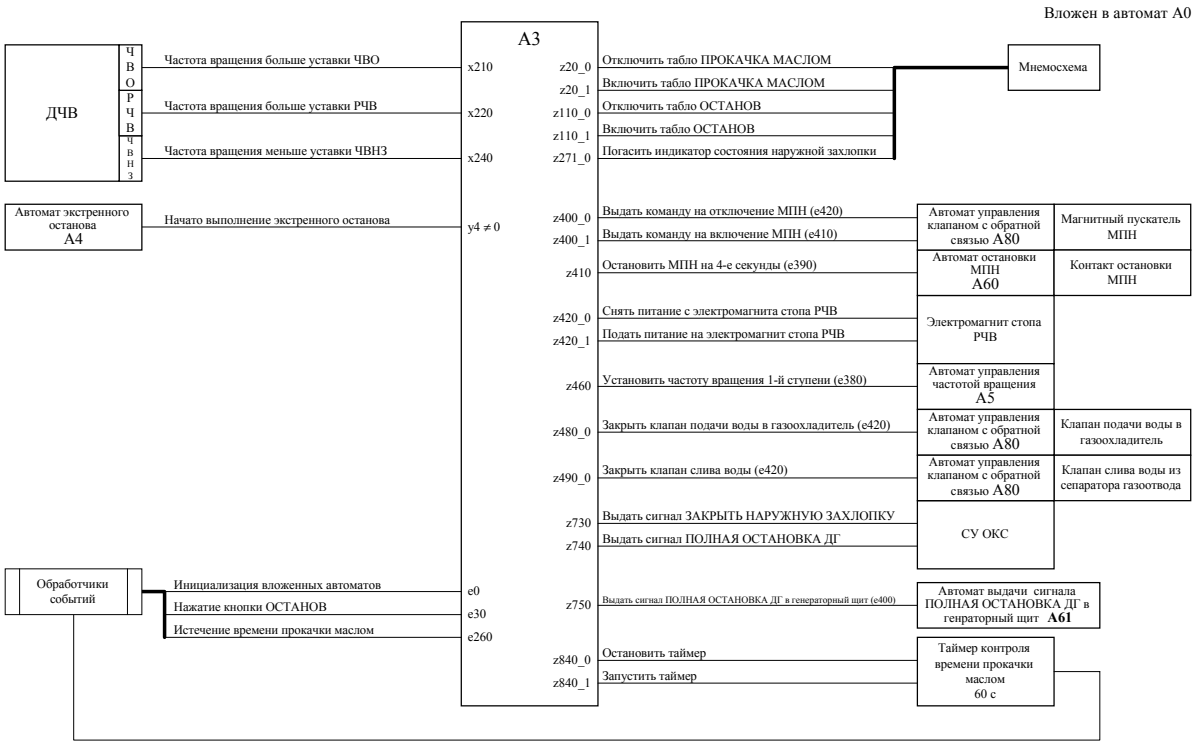

Рис. 13. Схема связей автомата останова

1.10.3. Граф переходов

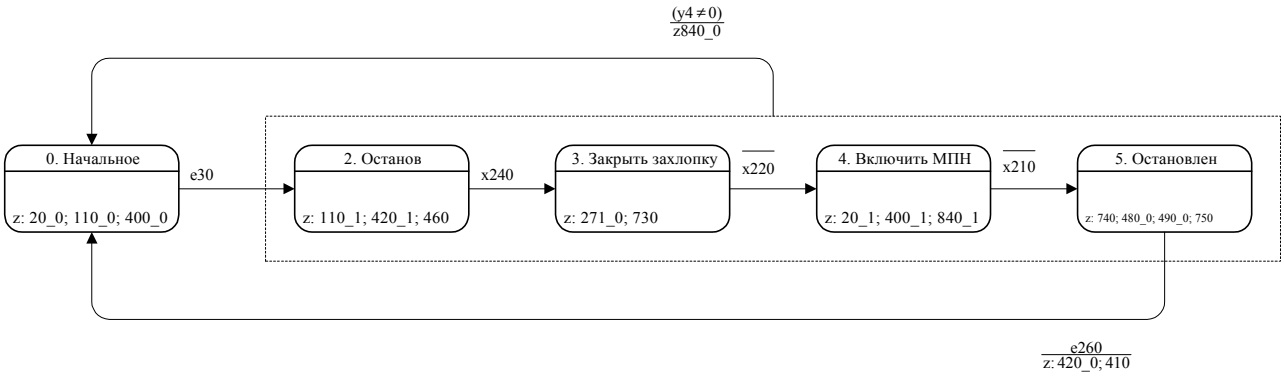

Рис. 14. Граф переходов автомата останова

1.10.4. Текст функции, реализующей автомат

```
#include "photon_stuff.h"
#include "dg.h"
#include "log.h"
#include "defines.h"

void A3( int e, dg_t *dg )
{
  int y_old = dg->y3 ;

  #ifdef GRAPH_EVENTS_LOGGING
    log_exec( dg, "A3", y_old, e ) ;
  #endif

  switch( dg->y3 )
  {
    case 0:
      if( e == 30 )                              dg->y3 = 2 ;
    break ;

    case 2:
      if( dg->y4 != 0 ) { z840_0(dg) ;           dg->y3 = 0 ; }
      else
      if( x240(dg) )                             dg->y3 = 3 ;
    break ;

    case 3:
      if( dg->y4 != 0 ) { z840_0(dg) ;           dg->y3 = 0 ; }
      else
      if( !x220(dg) )                            dg->y3 = 4 ;
    break ;

    case 4:
      if( dg->y4 != 0 ) { z840_0(dg) ;           dg->y3 = 0 ; }
      else
      if( !x210(dg) )                            dg->y3 = 5 ;
    break ;

    case 5:
      if( dg->y4 != 0 ) { z840_0(dg) ;           dg->y3 = 0 ; }
      else
      if( e == 260 ) { z410(dg) ; z420_0(dg) ; dg->y3 = 0 ; }
    break ;

    default:
      #ifdef GRAPH_ERRORS_LOGGING
        log_write( LOG_GRAPH_ERROR, dg->number,
                   "ERROR IN A3: unknown state number!", 0 ) ;
      #endif
    break ;
  } ;
```

```
  if( y_old == dg->y3 ) goto A3_end ;

  {
    #ifdef GRAPH_TRANS_LOGGING
      log_trans( dg, "A3", y_old, dg->y3 ) ;
    #endif

    #ifdef DEBUG_FRAME
      update_debug() ;
    #endif
  } ;

  switch( dg->y3 )
  {
    case 0:
      z20_0(dg) ; z110_0(dg) ; z400_0(dg) ;
    break ;

    case 2:
      z110_1(dg) ; z420_1(dg) ; z460(dg) ;
    break ;

    case 3:
      z271_0(dg) ; z730(dg) ;
    break ;

    case 4:
      z20_1(dg) ; z400_1(dg) ; z840_1(dg) ;
    break ;

    case 5:
      z480_0(dg) ; z490_0(dg) ; z740(dg) ; z750(dg) ;
    break ;
  } ;

  A3_end:
  #ifdef GRAPH_ENDS_LOGGING
    log_end( dg, "A3", dg->y3, e ) ;
  #endif
  ;
} ;
```

1.11. Автомат ожидания разрешения выполнения предпусковых операций

1.11.1. Словесное описание

Автомат реализует часть алгоритма предпусковых операций, описанного в подразделе 1.5. При нажатии кнопки "Подготовка к пуску" автомат проверяет значения сигналов, запрещающих выполнение алгоритма предпусковых операций и, при необходимости, выдает в СУ ОКС сигнал "Подготовка к пуску".

Схема связей автомата и граф переходов приведены на рис. 15, 16.

1.11.2. Схема связей

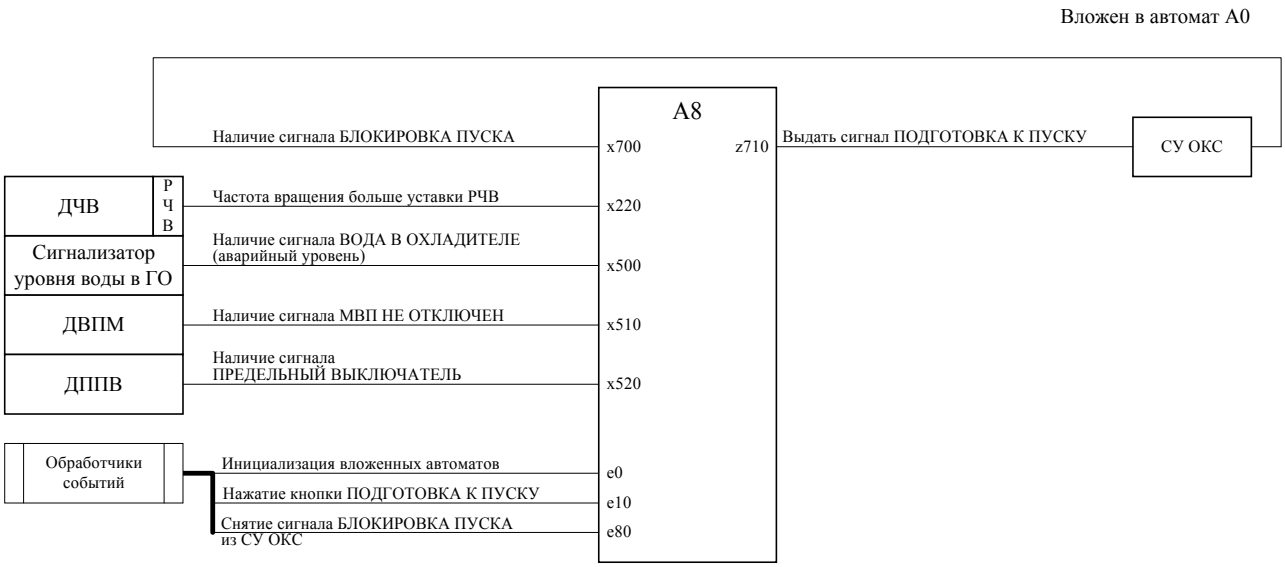

Рис. 15. Схема связей автомата ожидания разрешения выполнения предпусковых операций

1.11.3. Граф переходов

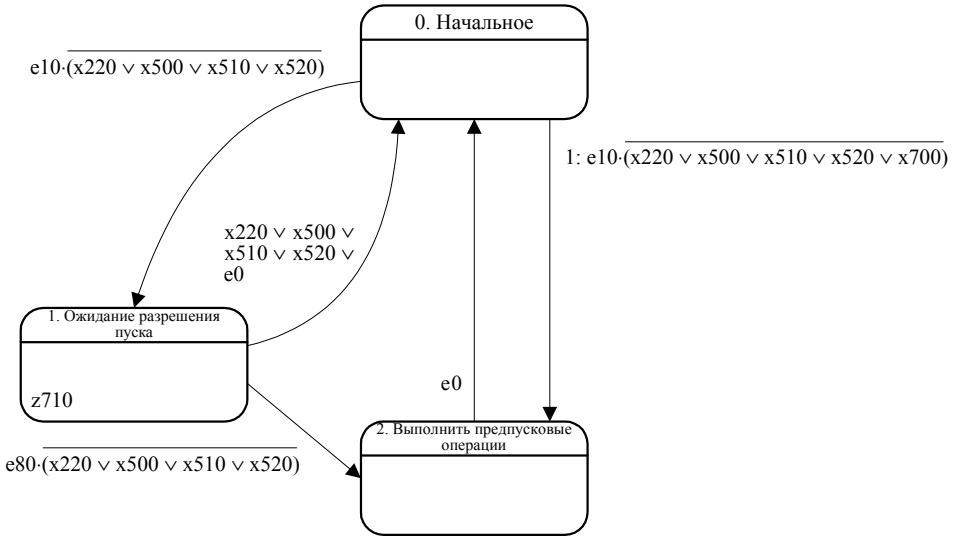

Рис. 16. Граф переходов автомата ожидания разрешения выполнения предпусковых операций

1.11.4. Текст функции, реализующей автомат

```
#include "photon_stuff.h"
#include "dg.h"
#include "log.h"
#include "defines.h"

void A8( int e, dg_t *dg )
{
  int y_old = dg->y8 ;

  #ifdef GRAPH_EVENTS_LOGGING
    log_exec( dg, "A8", y_old, e ) ;
  #endif
```

```
  switch( dg->y8 )
  {
    case 0:
      if( e == 10 && !( x220(dg) || x500(dg) || x510(dg) || x520(dg) || x700(dg)))
                                                dg->y8 = 2 ;
      else
      if( e == 10 && !( x220(dg) || x500(dg) || x510(dg) || x520(dg) ))
                                                dg->y8 = 1 ;
    break ;

    case 1:
      if( x220(dg) || x500(dg) || x510(dg) || x520(dg) || e == 0 )
                                                dg->y8 = 0 ;
      else
      if( e == 80 && !( x220(dg) || x500(dg) || x510(dg) || x520(dg)))
                                                dg->y8 = 2 ;
    break ;

    case 2:
      if( e == 0 )
                                                dg->y8 = 0 ;
    break ;

    default:
      #ifdef GRAPH_ERRORS_LOGGING
        log_write( LOG_GRAPH_ERROR, dg->number,
                   "ERROR IN A8: unknown state number!", 0 ) ;
      #endif
    break ;
  } ;

  if( y_old == dg->y8 ) goto A8_end ;

  {
    #ifdef GRAPH_TRANS_LOGGING
      log_trans( dg, "A8", y_old, dg->y8 ) ;
    #endif

    #ifdef DEBUG_FRAME
      update_debug() ;
    #endif
  } ;

  switch( dg->y8 )
  {
    case 1:
      z710(dg) ;
    break ;
  } ;

  A8_end:
  #ifdef GRAPH_ENDS_LOGGING
    log_end( dg, "A8", dg->y8, e ) ;
  #endif
  ;
} ;
```

1.12. Автомат аварийной и предупредительной сигнализации

1.12.1. Словесное описание

Автомат обобщает работу автоматов контроля параметров дизель-генратора. Выделены четыре режима работы алгоритма аварийной и предупредительной сигнализации.

1. Рабочий режим. Все параметры в допуске.

2. Аварийный останов. Аварийное отклонение как минимум одного из контролируемых параметров.

3. Авария без останова. Аварийное отклонение температуры масла или температуры воды при отключенной защите по этим параметрам.

4. Разнос. Частота вращения дизель-генератора превысила предельно допустимую.

Схема связей автомата и граф переходов приведены на рис. 17, 18.

1.12.2. Схема связей

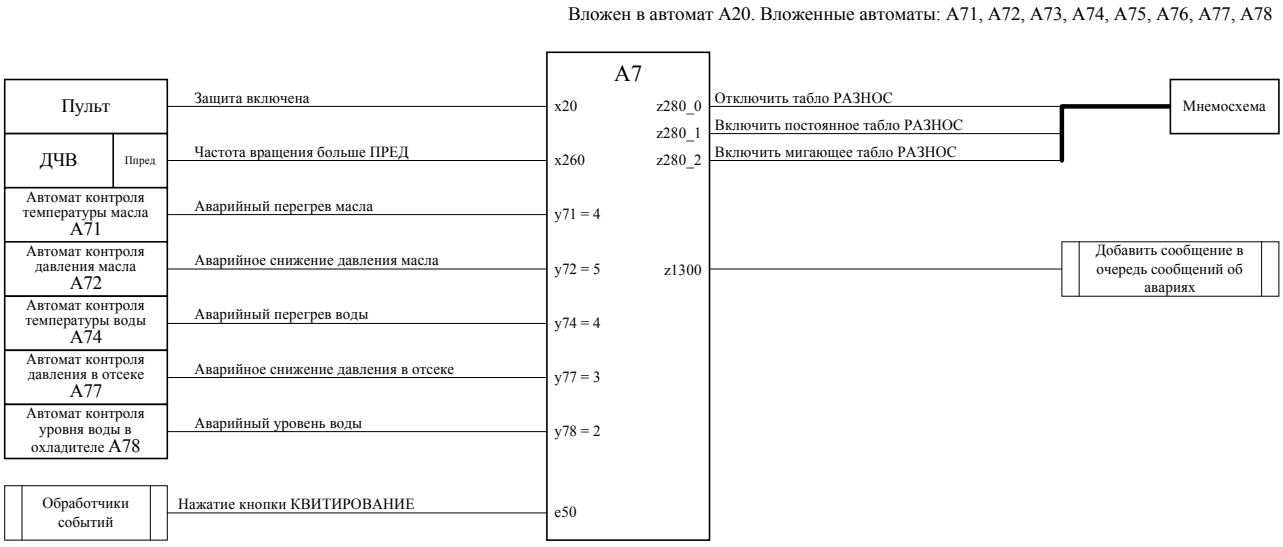

Рис. 17. Схема связей автомата аварийной и предупредительной сигнализации

1.12.3. Граф переходов

$$C7\_1 = x20 \cdot \big((y71 = 4) \vee (y74 = 4)\big) \vee (y72 = 5) \vee (y77 = 3) \vee (y78 = 2)$$
$$C7\_2 = \overline{x20} \cdot \big((y71 = 4) \vee (y74 = 4)\big)$$

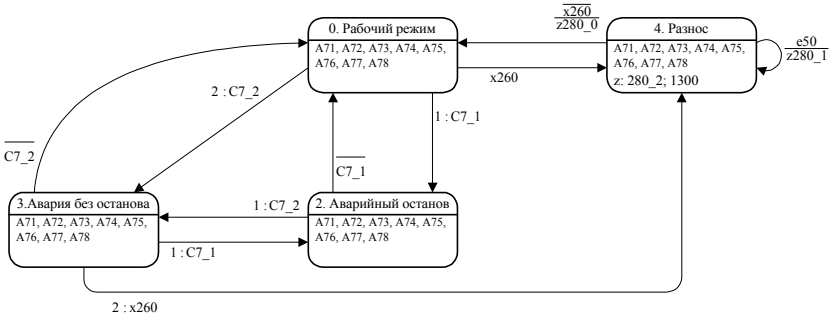

Рис. 18. Граф переходов автомата аварийной и предупредительной сигнализации

1.12.4. Текст функции, реализующей автомат

```
#include "photon_stuff.h"
#include "dg.h"
#include "log.h"
#include "defines.h"

int C7_1( dg_t *dg )
{
  return    ( x20(dg) && ( dg->y71 == 4 || dg->y74 == 4 ))
         || dg->y72 == 5 || dg->y77 == 3 || dg->y78 == 2 ;
} ;

int C7_2( dg_t *dg )
{
  return !x20(dg) && ( dg->y71 == 4 || dg->y74 == 4 ) ;
} ;
```

```
void A7( int e, dg_t *dg )
{
  int y_old = dg->y7 ;

  #ifdef GRAPH_EVENTS_LOGGING
    log_exec( dg, "A7", y_old, e ) ;
  #endif

  switch( dg->y7 )
  {
    case 0:
      A71(e,dg) ; A72(e,dg) ; A73(e,dg) ; A74(e,dg) ; A75(e,dg) ;
      A76(e,dg) ; A77(e,dg) ; A78(e,dg) ;
      if( C7_1(dg) )                dg->y7 = 2 ;
      else
      if( C7_2(dg) )                dg->y7 = 3 ;
      else
      if( x260(dg) )                dg->y7 = 4 ;
    break ;

    case 2:
      A71(e,dg) ; A72(e,dg) ; A73(e,dg) ; A74(e,dg) ; A75(e,dg) ;
      A76(e,dg) ; A77(e,dg) ; A78(e,dg) ;
      if( C7_2(dg) )                dg->y7 = 3 ;
      else
      if( !C7_1(dg) )               dg->y7 = 0 ;
    break ;

    case 3:
      A71(e,dg) ; A72(e,dg) ; A73(e,dg) ; A74(e,dg) ; A75(e,dg) ;
      A76(e,dg) ; A77(e,dg) ; A78(e,dg) ;
      if( C7_1(dg) )                dg->y7 = 2 ;
      else
      if( x260(dg) )                dg->y7 = 4 ;
      else
      if( !C7_2(dg) )               dg->y7 = 0 ;
    break ;

    case 4:
      A71(e,dg) ; A72(e,dg) ; A73(e,dg) ; A74(e,dg) ; A75(e,dg) ;
      A76(e,dg) ; A77(e,dg) ; A78(e,dg) ;
      if( !x260(dg) ) { z280_0(dg) ; dg->y7 = 0 ; }
      else
      if( e == 50 ) { z280_1(dg) ; }
    break ;

    default:
      #ifdef GRAPH_ERRORS_LOGGING
        log_write( LOG_GRAPH_ERROR, dg->number,
                   "ERROR IN A7: unknown state number!", 0 ) ;
      #endif
    break ;
  } ;

  if( y_old == dg->y7 ) goto A7_end ;

  {
    #ifdef GRAPH_TRANS_LOGGING
      log_trans( dg, "A7", y_old, dg->y7 ) ;
    #endif

    #ifdef DEBUG_FRAME
      update_debug() ;
    #endif
  } ;

  switch( dg->y7 )
  {
    case 0:
      A71(0,dg) ; A72(0,dg) ; A73(0,dg) ; A74(0,dg) ; A75(0,dg) ;
      A76(0,dg) ; A77(0,dg) ; A78(0,dg) ;
    break ;

    case 2:
      A71(0,dg) ; A72(0,dg) ; A73(0,dg) ; A74(0,dg) ; A75(0,dg) ;
      A76(0,dg) ; A77(0,dg) ; A78(0,dg) ;
    break ;
```

```
    case 3:
      A71(0,dg) ; A72(0,dg) ; A73(0,dg) ; A74(0,dg) ; A75(0,dg) ;
      A76(0,dg) ; A77(0,dg) ; A78(0,dg) ;
    break ;

    case 4:
      A71(0,dg) ; A72(0,dg) ; A73(0,dg) ; A74(0,dg) ; A75(0,dg) ;
      A76(0,dg) ; A77(0,dg) ; A78(0,dg) ;
      z280_2(dg) ; z1300(dg, MSG_RAZNOS_FAIL) ;
    break ;
  } ;

  A7_end:
  #ifdef GRAPH_ENDS_LOGGING
    log_end( dg, "A7", dg->y7, e ) ;
  #endif
  ;
} ;
```

1.13. Автомат контроля температуры масла

1.13.1. Словесное описание

Автомат производит контроль температуры масла, определяет и выполняет индикацию следующих ситуаций:

– низкая температура. Температура масла ниже уставки Тмм;

– предупредительный перегрев. Температура масла выше уставки Тмпр;

– аварийный перегрев. Температура масла выше уставки Тма.

Схема связей автомата и граф переходов приведены на рис. 19, 20.

1.13.2. Схема связей

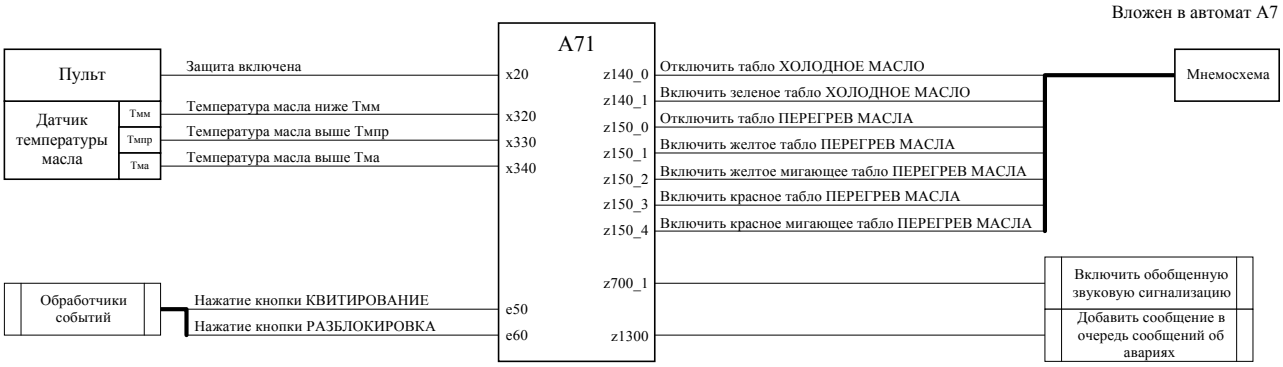

Рис. 19. Схема связей автомата контроля температуры масла

1.13.3. Граф переходов

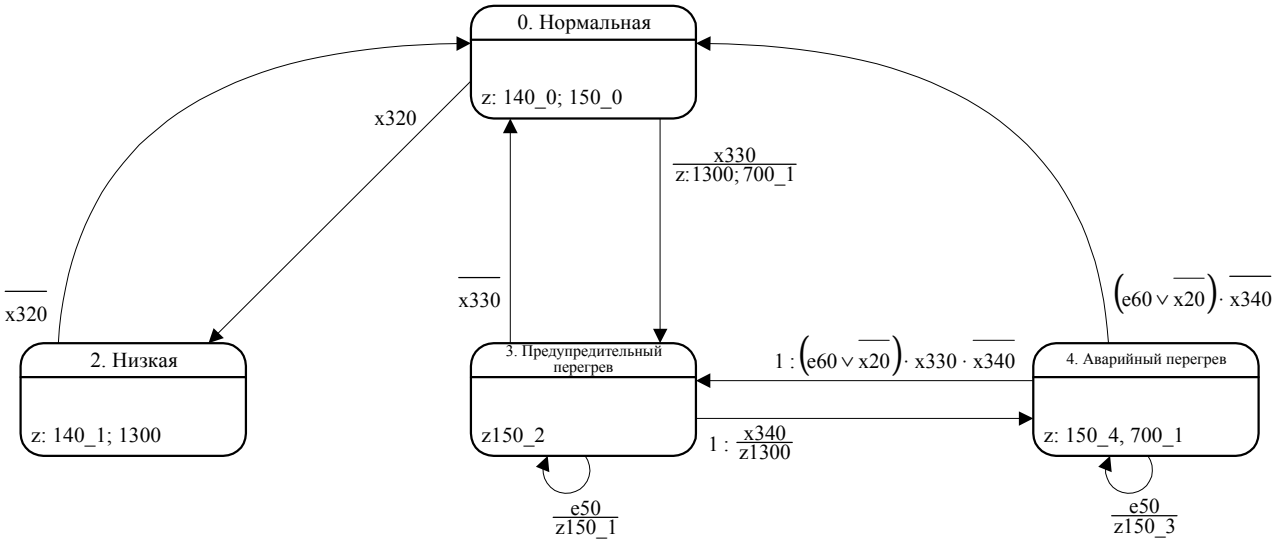

Рис. 20. Граф переходов автомата контроля температуры масла

1.13.4. Текст функции, реализующей автомат

```
#include "photon_stuff.h"
#include "dg.h"
#include "log.h"
#include "defines.h"

void A71( int e, dg_t *dg )
{
  int y_old = dg->y71 ;

  #ifdef GRAPH_EVENTS_LOGGING
    log_exec( dg, "A71", y_old, e ) ;
  #endif

  switch( dg->y71 )
  {
    case 0:
      if( x320(dg) )                                    dg->y71 = 2 ;
      else
      if( x330(dg) ) { z1300(dg, MSG_OIL_TEMP_HIGH_ALARM) ; z700_1(dg) ;
                                                        dg->y71 = 3 ; }
    break ;

    case 2:
      if( !x320(dg) )                                   dg->y71 = 0 ;
    break ;

    case 3:
      if( x340(dg) ) { z1300(dg, MSG_OIL_TEMP_HIGH_FAIL) ;
                                                        dg->y71 = 4 ; }
      else
      if( !x330(dg) )                                   dg->y71 = 0 ;
      else
      if( e == 50 ) { z150_1(dg) ; }
    break ;

    case 4:
      if( ( e == 60 || !x20(dg) ) && x330(dg) && !x340(dg))
                                                        dg->y71 = 3 ;
      else
      if( ( e == 60 || !x20(dg) ) && !x340(dg) )
                                                        dg->y71 = 0 ;
      else
      if( e == 50 ) { z150_3(dg) ; }
    break ;
```

```
    default:
      #ifdef GRAPH_ERRORS_LOGGING
        log_write( LOG_GRAPH_ERROR, dg->number,
                   "ERROR IN A71: unknown state number!", 0 ) ;
      #endif
    break ;
  } ;

  if( y_old == dg->y71 ) goto A71_end ;

  {
    #ifdef GRAPH_TRANS_LOGGING
      log_trans( dg, "A71", y_old, dg->y71 ) ;
    #endif

    #ifdef DEBUG_FRAME
      update_debug() ;
    #endif
  } ;

  switch( dg->y71 )
  {
    case 0:
      z140_0(dg) ; z150_0(dg) ;
    break ;

    case 2:
      z140_1(dg) ; z1300(dg, MSG_OIL_TEMP_LOW_ALARM) ;
    break ;

    case 3:
      z150_2(dg) ;
    break ;

    case 4:
      z150_4(dg) ; z700_1(dg) ;
    break ;
  } ;

  A71_end:
  #ifdef GRAPH_ENDS_LOGGING
    log_end( dg, "A71", dg->y71, e ) ;
  #endif
  ;
} ;
```

1.14. Автомат контроля давления масла

1.14.1. Словесное описание

Автомат производит контроль давления масла, определяет и выполняет индикацию следующих ситуаций:

– предупредительное снижение. Давление масла ниже уставки Рмпрi, где i - текущая ступень частоты вращения;

– аварийное снижение. Давление масла ниже уставки Рмаi.

Контроль давления масла включается через 10 с после запуска дизель-генератора.

Схема связей автомата и граф переходов приведены на рис. 21, 22.

1.14.2. Схема связей

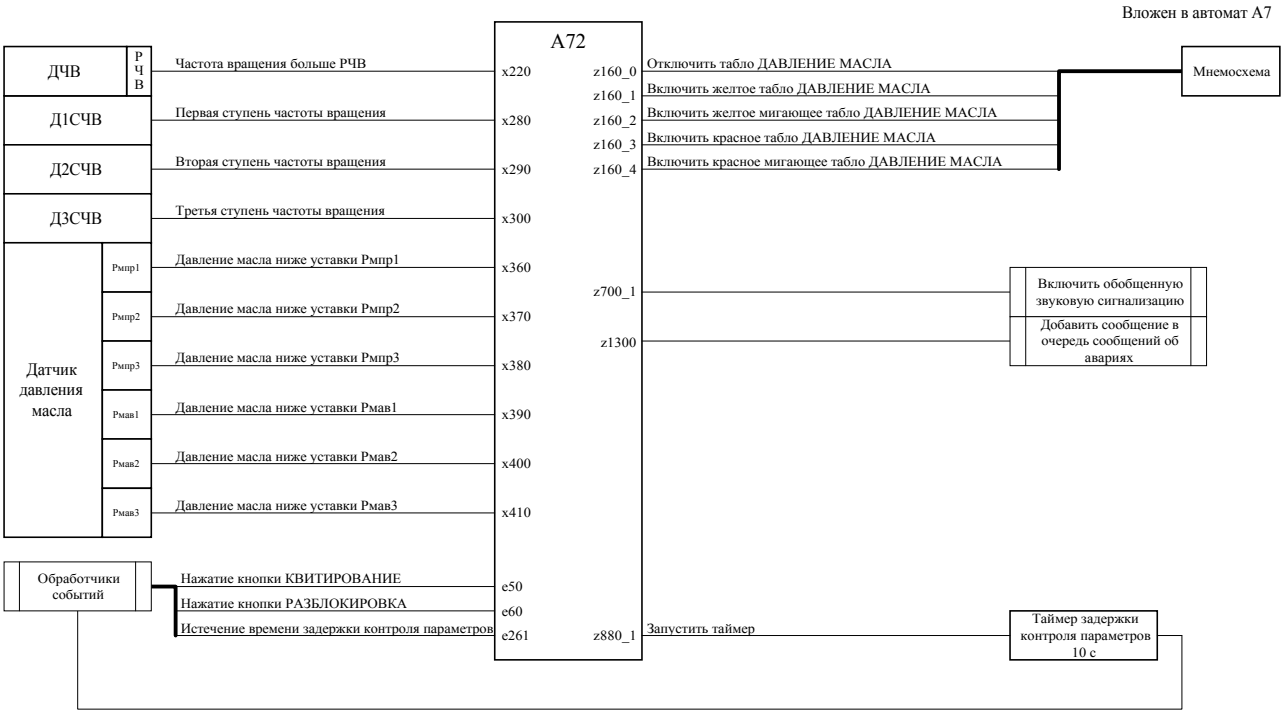

Рис. 21. Схема связей автомата контроля давления масла

1.14.3. Граф переходов

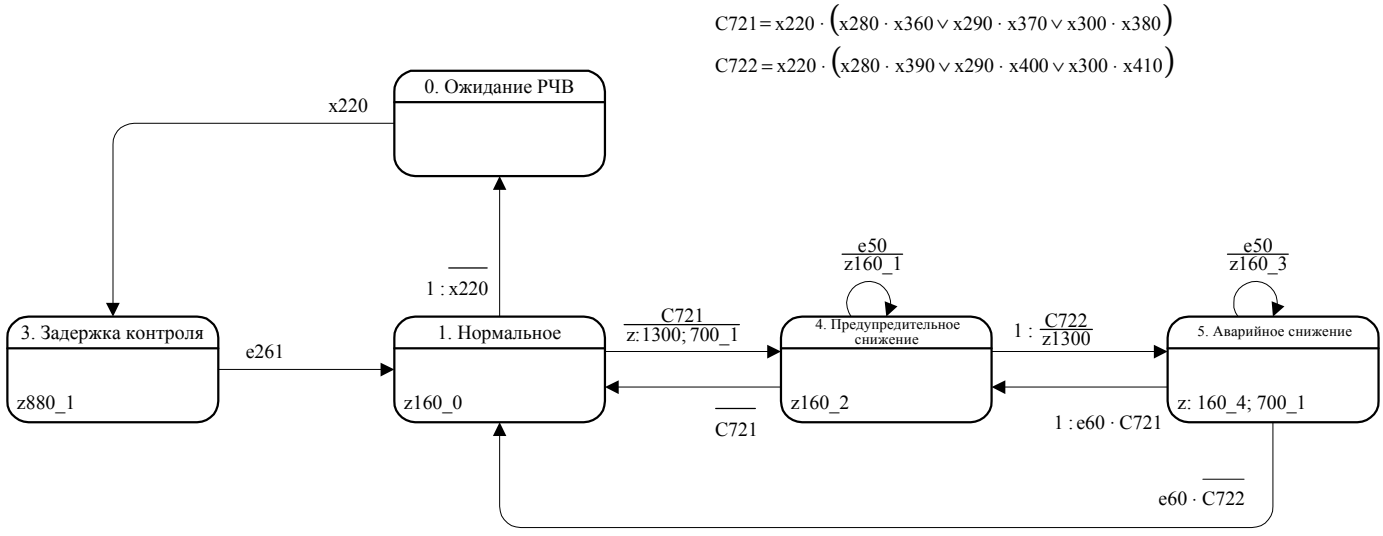

Рис. 22. Граф переходов автомата контроля давления масла

1.14.4. Текст функции, реализующей автомат

```
#include "photon_stuff.h"
#include "dg.h"
#include "log.h"
#include "defines.h"

int C721( dg_t *dg )
{
  return   x220(dg) && (     ( x280(dg) && x360(dg) )
                          || ( x290(dg) && x370(dg) )
                          || ( x300(dg) && x380(dg) )
                        ) ;
} ;
```

```
int C722( dg_t *dg )
{
  return   x220(dg) && (    ( x280(dg) && x390(dg) )
                         || ( x290(dg) && x400(dg) )
                         || ( x300(dg) && x410(dg) )
                       ) ;
} ;

void A72( int e, dg_t *dg )
{
  int y_old = dg->y72 ;

  #ifdef GRAPH_EVENTS_LOGGING
    log_exec( dg, "A72", y_old, e ) ;
  #endif

  switch( dg->y72 )
  {
    case 0:
      if( x220(dg) )                            dg->y72 = 3 ;
    break ;

    case 1:
      if( !x220(dg) )                           dg->y72 = 0 ;
      else
      if( C721(dg) ) { z1300(dg, MSG_OIL_PRESS_ALARM) ; z700_1(dg) ;
                                                dg->y72 = 4 ; }
    break ;

    case 3:
      if( e == 261 )                            dg->y72 = 1 ;
    break ;

    case 4:
      if( C722(dg) ) { z1300(dg, MSG_OIL_PRESS_FAIL) ;
                                                dg->y72 = 5 ; }
      else
      if( !C721(dg) )                           dg->y72 = 1 ;
      else
      if( e == 50 ) { z160_1(dg) ; }
    break ;

    case 5:
      if( e == 60 && C721(dg) )                 dg->y72 = 4 ;
      else
      if( e == 60 && !C722(dg) )                dg->y72 = 1 ;
      else
      if( e == 50 ) { z160_3(dg) ; }
    break ;

    default:
      #ifdef GRAPH_ERRORS_LOGGING
        log_write( LOG_GRAPH_ERROR, dg->number,
                   "ERROR IN A72: unknown state number!", 0 ) ;
      #endif
    break ;
  } ;

  if( y_old == dg->y72 ) goto A72_end ;

  {
    #ifdef GRAPH_TRANS_LOGGING
      log_trans( dg, "A72", y_old, dg->y72 ) ;
    #endif

    #ifdef DEBUG_FRAME
      update_debug() ;
    #endif
  } ;

  switch( dg->y72 )
  {
    case 1:
      z160_0(dg) ;
    break ;
```

```
    case 3:
      z880_1(dg) ;
    break ;

    case 4:
      z160_2(dg) ;
    break ;

    case 5:
      z160_4(dg) ; z700_1(dg) ;
    break ;
  } ;

  A72_end:
  #ifdef GRAPH_ENDS_LOGGING
    log_end( dg, "A72", dg->y72, e ) ;
  #endif
  ;
} ;
```

1.15. Обработчики событий

В качестве примера приведем несколько обработчиков событий, вызывающих автоматы.

```
int e10( PtWidget_t *widget, ApInfo_t *apinfo, PtCallbackInfo_t *cbinfo )
{
  if( widget == ABW_dg1_prepare_btn )
    A20( 10, &dgs[DG1] ) ;
  else if( widget == ABW_dg2_prepare_btn )
    A20( 10, &dgs[DG2] ) ;

  return( Pt_CONTINUE ) ;
} ;

int e220( void *data, pid_t pid, void *msg, size_t msg_size )
{
  dg_t  *dg = (dg_t*)data ;

  A20( 220, dg ) ;

  return Pt_END ;
} ;

int e266( void *data, pid_t pid, void *msg, size_t msg_size )
{
  dg_t  *dg = (dg_t*)data ;

  A13( 266, dg ) ;

  return Pt_END ;
} ;
```

1.16. Входные переменные

В качестве примера приведем реализацию нескольких входных переменных.

```
int x10( dg_t *dg )
{
  int result = 0 ;

  #ifdef IMITATOR_MODE
  {
    message_t     msg, rmsg ;

    if( imitator_pid == -1 )
      result = 0 ;
    else
    {
      msg.type = X10_MSG ;
      msg.dg_num = dg->number - 1 ;
```

```
      Send( imitator_pid, &msg, &rmsg, sizeof(msg), sizeof(rmsg) ) ;

      result = rmsg.data ;
    }
  } ;
  #endif

  #ifdef INPUTS_LOGGING
    log_input( dg,
    "x10 - дистанционное управление включено -", result ) ;
  #endif

  return result ;
} ;

int x20( dg_t *dg )
{
  int result = 0 ;

  result = dg->number == 2 ? is_wgt_set(ABW_dg2_protection_btn) :
                             is_wgt_set(ABW_dg1_protection_btn) ;

  #ifdef INPUTS_LOGGING
    log_input( dg, "x20 - защита включена -", result ) ;
  #endif

  return result ;
} ;

int x320( dg_t *dg )
{
  int result = 0 ;

  result = get_dg_oil_temp(dg) < Tmm ;

  #ifdef INPUTS_LOGGING
    log_input( dg, "x320 - температура масла меньше Тмм -", result ) ;
  #endif

  return result ;
} ;
```

1.17. Выходные воздействия

В качестве примера приведем реализацию нескольких выходных воздействий.

```
void z10_0( dg_t *dg )
{
  #ifdef ACTIONS_LOGGING
    log_write( LOG_ACTION, dg->number,
               "z10_0. Отключить табло ПОДГОТОВКА К ПУСКУ.", 0 ) ;
  #endif

  tab_off( dg->prepare_tab ) ;
} ;

void z10_1( dg_t *dg )
{
  #ifdef ACTIONS_LOGGING
    log_write( LOG_ACTION, dg->number,
               "z10_1. Включить табло ПОДГОТОВКА К ПУСКУ.", 0 ) ;
  #endif

  tab_on( dg->prepare_tab, on_tab_color ) ;
} ;

void z810_0( dg_t *dg )
{
  #ifdef ACTIONS_LOGGING
    log_write( LOG_ACTION, dg->number,
               "z810_0. Сбросить таймер контроля проворота.", 0 ) ;
  #endif

  disarm_timer( dg->e220_timer_id ) ;
  dg->e220_timer_id = -1 ;
} ;
```

```
void z810_1( dg_t *dg )
{
  #ifdef ACTIONS_LOGGING
    log_write( LOG_ACTION, dg->number,
               "z810_1. Запустить таймер контроля проворота.", 0 ) ;
  #endif

  dg->e220_timer_id = arm_timer( turn_timer_delay, 0, 0, 0, e220, dg ) ;
} ;
```

1.18. Вспомогательные модули

Вспомогательные модули в зависимости от характера реализуемого ими алгоритма могут как содержать автоматы, так и не содержать их. В качестве примера приведем реализацию вспомогательных функций из модуля управления виджетами.

```
//======
// Отключить табло.
//
void tab_off( PtWidget_t *wgt )
{
  PtArg_t       args[5] ;
  int           n = 0 ;

  if( !wgt ) return ;

  PtSetArg( &args[n++], Pt_ARG_FLAGS, Pt_GHOST, Pt_GHOST ) ;
  PtSetArg( &args[n++], Pt_ARG_FILL_COLOR, Pg_GREY, 0 ) ;
  PtSetArg( &args[n++], Pt_ARG_COLOR, PgRGB(144,144,144), 0 ) ;

  PtSetResources( wgt, n, args ) ;
} ;

//======
// Включить табло.
//
void tab_on( PtWidget_t *wgt, PgColor_t color )
{
  PtArg_t       args[5] ;
  int           n = 0 ;

  if( !wgt ) return ;

  PtSetArg( &args[n++], Pt_ARG_FLAGS, ~Pt_GHOST, Pt_GHOST ) ;
  PtSetArg( &args[n++], Pt_ARG_FILL_COLOR, color, 0 ) ;
  PtSetArg( &args[n++], Pt_ARG_COLOR, Pg_BLACK, 0 ) ;

  PtSetResources( wgt, n, args ) ;
} ;
```

1.19. Протоколы функционирования системы управления

Приведем примеры протоколов работы для системы управления в целом при обработке нажатия кнопки "Подготовка к пуску" (событие **e10**), полученный автоматически с выделенными вручную этапами.

Диагностирующий (полный) протокол имеет следующий вид.

Нажатие кнопки "Подготовка к пуску":

```
11:34:02.507{ DG1: A20: в состоянии 2 запущен с событием e10
11:34:02.507{ DG1: A7: в состоянии 0 запущен с событием e10
11:34:02.507{ DG1: A71: в состоянии 0 запущен с событием e10
11:34:02.507> DG1: x320 - температура масла меньше Тмм - вернул 0
11:34:02.507> DG1: x330 - температура масла больше Тмпр - вернул 0
11:34:02.507} DG1: A71: завершил обработку события e10 в состоянии 0
11:34:02.507{ DG1: A72: в состоянии 0 запущен с событием e10
11:34:02.507> DG1: x220 - частота вращения больше РЧВ - вернул 0
11:34:02.507} DG1: A72: завершил обработку события e10 в состоянии 0
11:34:02.507{ DG1: A73: в состоянии 0 запущен с событием e10
```

```
11:34:02.507> DG1: x220 - частота вращения больше РЧВ - вернул 0
11:34:02.507} DG1: A73: завершил обработку события e10 в состоянии 0
11:34:02.507{ DG1: A74: в состоянии 0 запущен с событием e10
11:34:02.507> DG1: x430 - температура воды меньше Твм - вернул 0
11:34:02.507> DG1: x440 - температура воды больше Твпр - вернул 0
11:34:02.507} DG1: A74: завершил обработку события e10 в состоянии 0
11:34:02.507{ DG1: A75: в состоянии 0 запущен с событием e10
11:34:02.507> DG1: x460 - давление наддува больше Рнпр - вернул 0
11:34:02.507} DG1: A75: завершил обработку события e10 в состоянии 0
11:34:02.507{ DG1: A76: в состоянии 0 запущен с событием e10
11:34:02.507> DG1: x470 - температура газов больше Твг - вернул 0
11:34:02.507} DG1: A76: завершил обработку события e10 в состоянии 0
11:34:02.507{ DG1: A77: в состоянии 0 запущен с событием e10
11:34:02.507> DG1: x480 - давление в отсеке ниже Ротс - вернул 0

11:34:02.507} DG1: A77: завершил обработку события e10 в состоянии 0
11:34:02.507{ DG1: A78: в состоянии 0 запущен с событием e10
11:34:02.507> DG1: x500 - сигнал ВОДА В ОХЛАДИТЕЛЕ - вернул 0
11:34:02.507} DG1: A78: завершил обработку события e10 в состоянии 0
11:34:02.507> DG1: x20 - защита включена - вернул 1
11:34:02.507> DG1: x20 - защита включена - вернул 1
11:34:02.507> DG1: x260 - частота вращения больше ППРЕД - вернул 0
11:34:02.507} DG1: A7: завершил обработку события e10 в состоянии 0
11:34:02.507{ DG1: A0: в состоянии 0 запущен с событием e10
11:34:02.507{ DG1: A8: в состоянии 0 запущен с событием e10
11:34:02.507> DG1: x220 - частота вращения больше РЧВ - вернул 0
11:34:02.507> DG1: x500 - сигнал ВОДА В ОХЛАДИТЕЛЕ - вернул 0
11:34:02.507> DG1: x510 - сигнал МВП НЕ ОТКЛЮЧЕН - вернул 0
11:34:02.507> DG1: x520 - сигнал ПРЕДЕЛЬНЫЙ ВЫКЛЮЧАТЕЛЬ - вернул 0
11:34:02.507> DG1: x700 - сигнал БЛОКИРОВКА ПУСКА из СУ ОКС - вернул 0
11:34:02.507T DG1: A8: перешел из состояния 0 в состояние 2
11:34:02.507} DG1: A8: завершил обработку события e10 в состоянии 2
11:34:02.507T DG1: A0: перешел из состояния 0 в состояние 1
11:34:02.507{ DG1: A4: в состоянии 0 запущен с событием e0
11:34:02.507{ DG1: A4_0: в состоянии 0 запущен с событием e0
11:34:02.507} DG1: A4_0: завершил обработку события e0 в состоянии 0
11:34:02.507} DG1: A4: завершил обработку события e0 в состоянии 0
11:34:02.507{ DG1: A3: в состоянии 0 запущен с событием e0
11:34:02.507} DG1: A3: завершил обработку события e0 в состоянии 0
11:34:02.507{ DG1: A1: в состоянии 0 запущен с событием e0
11:34:02.507T DG1: A1: перешел из состояния 0 в состояние 1
11:34:02.507{ DG1: A1_0: в состоянии 0 запущен с событием e0
11:34:02.507* DG1: z800_0. Сбросить счетчик проворотов.
11:34:02.507} DG1: A1_0: завершил обработку события e0 в состоянии 0
11:34:02.507* DG1: z10_1. Включить табло ПОДГОТОВКА К ПУСКУ.
11:34:02.507* DG1: z20_1. Включить табло ПРОКАЧКА МАСЛОМ.
11:34:02.507* DG1: z30_1. Включить табло ПРОВОРОТ.
```

Запустить маслопрокачивающий насос:

```
11:34:02.507* DG1: z400_1. Включить магнитный пускатель МПН.
11:34:02.507{ DG1: A80( z400 ): в состоянии 0 запущен с событием e410
11:34:02.507> DG1: x900( z400 ) - клапан открыт - вернул 0
11:34:02.507T DG1: A80( z400 ): перешел из состояния 0 в состояние 3
11:34:02.507* DG1: z270_2( z400 ). Индикация состояния клапана - закрывается/открывается.
11:34:02.507* DG1: z950_1( z400 ). Запустить таймер контроля срабатывания.
11:34:02.507* DG1: z1000_1( z400 ). Открыть/запустить.
11:34:02.507} DG1: A80( z400 ): завершил обработку события e410 в состоянии 3

11:34:02.507* DG1: z420_1. Подать питание на электромагнит стопа РЧВ.
```

Открыть клапан подачи воздуха к дизель-генератору:

```
11:34:02.507* DG1: z430_1. Открыть клапан подачи воздуха к ДГ.
11:34:02.507{ DG1: A80( z430 ): в состоянии 0 запущен с событием e410
11:34:02.507> DG1: x900( z430 ) - клапан открыт - вернул 0
11:34:02.507T DG1: A80( z430 ): перешел из состояния 0 в состояние 3
11:34:02.507* DG1: z270_2( z430 ). Индикация состояния клапана - закрывается/открывается.
11:34:02.507* DG1: z950_1( z430 ). Запустить таймер контроля срабатывания.
11:34:02.507* DG1: z1000_1( z430 ). Открыть/запустить.
11:34:02.507} DG1: A80( z430 ): завершил обработку события e410 в состоянии 3
```

Открыть клапан подачи воздуха низкого давления:

```
11:34:02.507* DG1: z440_1. Открыть клапан подачи воздуха низкого давления.
11:34:02.507{ DG1: A80( z440 ): в состоянии 0 запущен с событием e410
11:34:02.507> DG1: x900( z440 ) - клапан открыт - вернул 0
```

```
11:34:02.507T DG1: A80( z440 ): перешел из состояния 0 в состояние 3
11:34:02.507* DG1: z270_2( z440 ). Индикация состояния клапана - закрывается/открывается.
11:34:02.507* DG1: z950_1( z440 ). Запустить таймер контроля срабатывания.
11:34:02.507* DG1: z1000_1( z440 ). Открыть/запустить.
11:34:02.507} DG1: A80( z440 ): завершил обработку события e410 в состоянии 3
```

Выдать команду на установку частоты вращения первой ступени:

```
11:34:02.507* DG1: z460. Выдать команду на установку частоты вращения первой ступени.
11:34:02.507{ DG1: A5: в состоянии 0 запущен с событием e380
11:34:02.507> DG1: x810 - заданная частота вращения выше установленной - вернул 1
11:34:02.507T DG1: A5: перешел из состояния 0 в состояние 2
11:34:02.507* DG1: z610_1. Увеличить установленную частоту РЧВ.
11:34:02.507} DG1: A5: завершил обработку события e380 в состоянии 2

11:34:02.507* DG1: z810_1. Запустить таймер контроля проворота.
11:34:02.507} DG1: A1: завершил обработку события e0 в состоянии 1
11:34:02.507* DG1: z290_0. Индикация состояния ДГ - остановлен.
11:34:02.507} DG1: A0: завершил обработку события e10 в состоянии 1
11:34:02.507{ DG1: A5: в состоянии 2 запущен с событием e10
11:34:02.507> DG1: x810 - заданная частота вращения выше установленной - вернул 1
11:34:02.517> DG1: x820 - заданная частота вращения ниже установленной - вернул 0
11:34:02.517} DG1: A5: завершил обработку события e10 в состоянии 2
11:34:02.517{ DG1: A13: в состоянии 0 запущен с событием e10
11:34:02.517} DG1: A13: завершил обработку события e10 в состоянии 0
11:34:02.517{ DG1: A62: в состоянии 0 запущен с событием e10
11:34:02.517T DG1: A62: перешел из состояния 0 в состояние 1
11:34:02.517* DG1: z1110_1. Запустить таймер регистрации параметров.
11:34:02.517* DG1: z1120_1. Запустить таймер индикации параметров.
11:34:02.517} DG1: A62: завершил обработку события e10 в состоянии 1
11:34:02.517} DG1: A20: завершил обработку события e10 в состоянии 2
```

Ниже приводится проверяющий (короткий) протокол обработки нажатия кнопки "Подготовка к пуску" (событие **e10**).

```
11:41:06.188{ DG1: A20: в состоянии 2 запущен с событием e10
11:41:06.188* DG1: z800_0. Сбросить счетчик проворотов.
11:41:06.188* DG1: z10_1. Включить табло ПОДГОТОВКА К ПУСКУ.
11:41:06.188* DG1: z20_1. Включить табло ПРОКАЧКА МАСЛОМ.
11:41:06.188* DG1: z30_1. Включить табло ПРОВОРОТ.
11:41:06.188* DG1: z400_1. Включить магнитный пускатель МПН.
11:41:06.188* DG1: z270_2( z400 ). Индикация состояния клапана - закрывается/открывается.
11:41:06.188* DG1: z950_1( z400 ). Запустить таймер контроля срабатывания.
11:41:06.188* DG1: z1000_1( z400 ). Открыть/запустить.
11:41:06.188* DG1: z420_1. Подать питание на электромагнит стопа РЧВ.
11:41:06.188* DG1: z430_1. Открыть клапан подачи воздуха к ДГ.
11:41:06.188* DG1: z270_2( z430 ). Индикация состояния клапана - закрывается/открывается.
11:41:06.188* DG1: z950_1( z430 ). Запустить таймер контроля срабатывания.
11:41:06.188* DG1: z1000_1( z430 ). Открыть/запустить.
11:41:06.188* DG1: z440_1. Открыть клапан подачи воздуха низкого давления.
11:41:06.188* DG1: z270_2( z440 ). Индикация состояния клапана - закрывается/открывается.
11:41:06.188* DG1: z950_1( z440 ). Запустить таймер контроля срабатывания.
11:41:06.188* DG1: z1000_1( z440 ). Открыть/запустить.
11:41:06.188* DG1: z460. Выдать команду на установку частоты вращения первой ступени.
11:41:06.188* DG1: z610_1. Увеличить установленную частоту РЧВ.
11:41:06.188* DG1: z810_1. Запустить таймер контроля проворота.
11:41:06.188* DG1: z290_0. Индикация состояния ДГ - остановлен.
11:41:06.188* DG1: z1110_1. Запустить таймер регистрации параметров.
11:41:06.188* DG1: z1120_1. Запустить таймер индикации параметров.
11:41:06.188} DG1: A20: завершил обработку события e10 в состоянии 2
```

Для иллюстрации работы вложенных автоматов приведем структуру рассмотренного полного протокола.

```
1. A20          e10 в 2            В состояние 2 автомата A20 вложены
                                   A7,A0,A5,A13,A62
2. A7           e10 в 0            Вложен в состояние 2 автомата A20
3. A71          e10 в 0            Вложен в состояние 0 автомата A7
   x320==0, x330==0
   A71          Завершил обработку e10 в состоянии 0
4. A72          e10 в 0            Вложен в состояние 0 автомата A7
   x220==0
   A72          Завершил обработку e10 в состоянии 0
```

```
5.  A73          e10 в 0            Вложен в состояние 0 автомата A7
    x220==0
    A73          Завершил обработку e10 в состоянии 0

6.  A74          e10 в 0            Вложен в состояние 0 автомата A7
    x430==0, x440==0
    A74          Завершил обработку e10 в состоянии 0

7.  A75          e10 в 0            Вложен в состояние 0 автомата A7
    x460==0
    A75          Завершил обработку e10 в состоянии 0

8.  A76          e10 в 0            Вложен в состояние 0 автомата A7
    x470==0
    A76          Завершил обработку e10 в состоянии 0

9.  A77          e10 в 0            Вложен в состояние 0 автомата A7
    x480==0
    A77          Завершил обработку e10 в состоянии 0

10. A78          e10 в 0            Вложен в состояние 0 автомата A7
    x500==0
    A78          Завершил обработку e10 в состоянии 0
    x20==1, x260==0
    A7           Завершил обработку e10 в состоянии 0

11. A0           e10 в 0            Вложен в состояние 2 автомата A20
12. A8           e10 в 0            Вложен в состояние 0 автомата A0
    x220==0, x500==0, x510==0, x520==0, x700==0
    A8           0 → 2
    A8           Завершил обработку e10 в состоянии 2
    A0           0 → 2              В состояние 2 автомата A0 вложены
                                    A4, A3, A1
13. A4           e0 в 0             Вложен в состояние 2 автомата A0
14. A4_0         e0 в 0             Вложен в состояние 0 автомата A4
    A4_0         Завершил обработку e0 в состоянии 0
    A4           Завершил обработку e0 в состоянии 0

15. A3           e0 в 0             Вложен в состояние 2 автомата A0
    A3           Завершил обработку e0 в состоянии 0

16. A1           e0 в 0             Вложен в состояние 2 автомата A0
    A1           0 → 1              В состояние 1 автомата A1 вложен A1_0

17. A1_0         e0 в 0
    z800_0                          На петле в состоянии 0
    A1_0         Завершил обработку e0 в состоянии 0

    z10_1, z20_1, z30_3             В состояние 1 автомата A1

    z400_1                          В состояние 1 автомата A1
18. A80(z400)    e410 в 0
    x900(z400)==0
    A80(z400)    0 → 3
    z270_2(z400), z950_1(z400), z1000_1(z400)
    A80(z400)    Завершил обработку e410 в состоянии 3

    z420_1                          В состояние 1 автомата A1

    z430_1                          В состояние 1 автомата A1
19. A80(z430)    e410 в 0
    x900(z430)==0
    A80(z430)    0 → 3
    z270_2(z430), z950_1(z430), z1000_1(z430)
    A80(z430)    Завершил обработку e410 в состоянии 3

    z440_1                          В состояние 1 автомата A1
20. A80(z440)    e410 в 0
    x900(z440)==0
    A80(z440)    0 → 3
    z270_2(z440), z950_1(z440), z1000_1(z440)
    A80(z440)    Завершил обработку e410 в состоянии 3

    z460                            В состояние 1 автомата A1
21. A5           e380 в 0           Вызван из z460
    x810==1
    A5           0 → 2
    z610_1
    A5           Завершил обработку e380 в состоянии 2
```

```
    z810_1                          В состояние 1 автомата A1
    A1          Завершил обработку e0 в состоянии 1
    z290_0                          В состояние 2 автомата A0
    A0          Завершил обработку e0 в состоянии 2
22. A5          e10 в 2             Вложен в состояние 2 автомата A20
    x810==1, x820==0
    A5          Завершил обработку e10 в состоянии 2
23. A13         e10 в 0             Вложен в состояние 2 автомата A20
    A13         Завершил обработку e10 в состоянии 0
24. A62         e10 в 0             Вложен в состояние 2 автомата A20
    A62         0 → 1
    z1110_1, z1120_1, z1140
    A62         Завершил обработку e10 в состоянии 1
    A20         Завершил обработку e10 в состоянии 2
```

2. Программный имитатор дизель-генератора

Покажем, что предлагаемый подход применим не только для разработки программного обеспечения систем управления, но и для создания имитатора объекта управления, необходимого для автономной отладки указанной системы.

Так как среди четырех разработанных автоматов между собой взаимодействуют только автоматы A30 и A31 (по вложенности), то схема взаимодействия автоматов не приводится.

Обратим внимание, что некоторые из реализующих имитатор автоматов осуществляют управление аналоговыми параметрами. Среди автоматов, рассмотренных ниже, автомат A20 выполняет имитацию изменения затяжки пружины регулятора частоты вращения, а автомат A31 имитирует изменение частоты вращения дизель-генератора. Это возможно благодаря тому, что входные переменные и выходные воздействия автоматов реализуются функциями, которые могут выполнять произвольные действия.

2.1. Перечень событий

Перечислим названия событий (e), на которые реагирует имитатор, и их номера.

```
0       Инициализация вложенных автоматов (внутреннее событие)
10      Открыть клапан
20      Закрыть клапан
30      Истечение времени исполнения команды

11      Остановить регулятор частоты вращения
21      Увеличить затяжку пружины регулятора частоты вращения
31      Уменьшить затяжку пружины регулятора частоты вращения
41      Достигнуто минимальное значение затяжки пружины регулятора частоты вращения
51      Достигнуто максимальное значение -"-
61      Сигнал от таймера синхроимпульсов модели регулятора частоты вращения

70      Открытие клапана низкого давления
71      Закрытие клапана низкого давления
72      Закрытие клапана подачи воздуха к ДГ
73      Закрытие главного пускового клапана
80      Частота вращения больше уставки ЧВО
81      Частота вращения меньше уставки ЧВО
90      Сигнал от таймера синхроимпульсов модели проворота
100     Сигнал от таймера имитации задержки пуска
110     Сигнал от таймера синхроимпульсов модели дизеля
200     Обобщенная команда останова дизеля
```

2.2. Перечень входных переменных

Перечислим названия входных переменных (X), которые опрашиваются имитатором, и их номера.

```
10      Клапан подачи воздуха к ДГ открыт
20      Клапан подачи воздуха низкого давления к ДГ открыт
30      Главный пусковой клапан открыт
40      Текущая частота вращения больше установленной на регуляторе
50      Текущая частота вращения меньше установленной на регуляторе
```

2.3. Перечень выходных воздействий

Перечислим названия выходных воздействий (Z), формируемых имитатором, и их номера.

```
10      Таймер задержки выполнения команды исполнительным устройством
        (0 - сбросить, 1 - запустить)
20      Таймер синхроимпульсов модели регулятора частоты вращения
        (0 - сбросить, 1 - запустить)
30      Изменить затяжку пружины регулятора частоты вращения
        (0 - уменьшить, 1 - увеличить)
50      Таймер синхроимпульсов модели проворота (0 - сбросить, 1 - запустить)
60      Выдать в СУ ДГ сигнал от датчика проворота коленчатого вала
70      Таймер имитации задержки пуска (0 - сбросить, 1 - запустить)
80      Установить частоту вращения дизеля на отметку частота вращения
        останова плюс 10
90      Таймер синхроимпульсов модели дизеля (0 - сбросить, 1 - запустить)
100     Изменить частоту вращения дизеля (0 - уменьшить, 1 - увеличить,
        2 - резко уменьшить)
```

2.4. Автомат имитации исполнительного устройства

2.4.1. Словесное описание

Автомат предназначен для имитации клапана (или любого аналогичного по логике работы исполнительного устройства, например, электродвигателя, используемого в рассматриваемой системе), снабженного сигнализаторами открытого и закрытого положений.

При построении автомата учитываются следующие особенности моделируемого устройства.

1. При поступлении команды открытия или закрытия клапана включается задержка времени исполнения команды, по истечении которой имитируется срабатывание сигнализатора соответствующего положения.

2. Если до истечения времени задержки исполнения команды поступает команда перевода клапана в исходное положение, задержка включается заново и выполняется отработка новой команды.

Схема связей автомата и граф переходов приведены на рис. 23, 24.

2.4.2. Схема связей

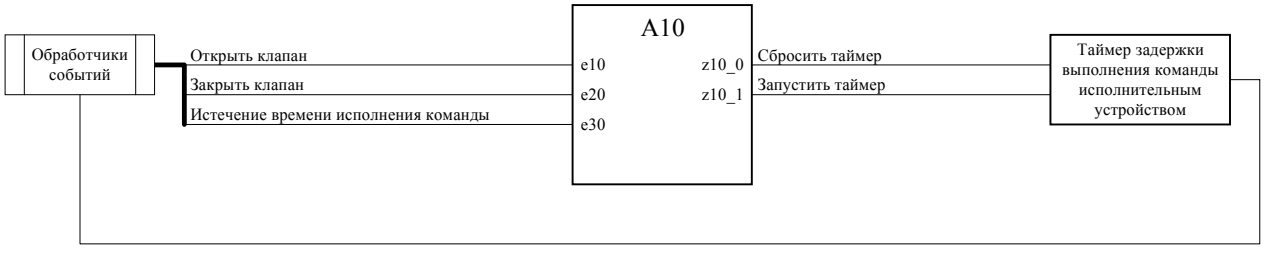

Рис. 23. Схема связей автомата имитации исполнительного устройства

2.4.3. Граф переходов

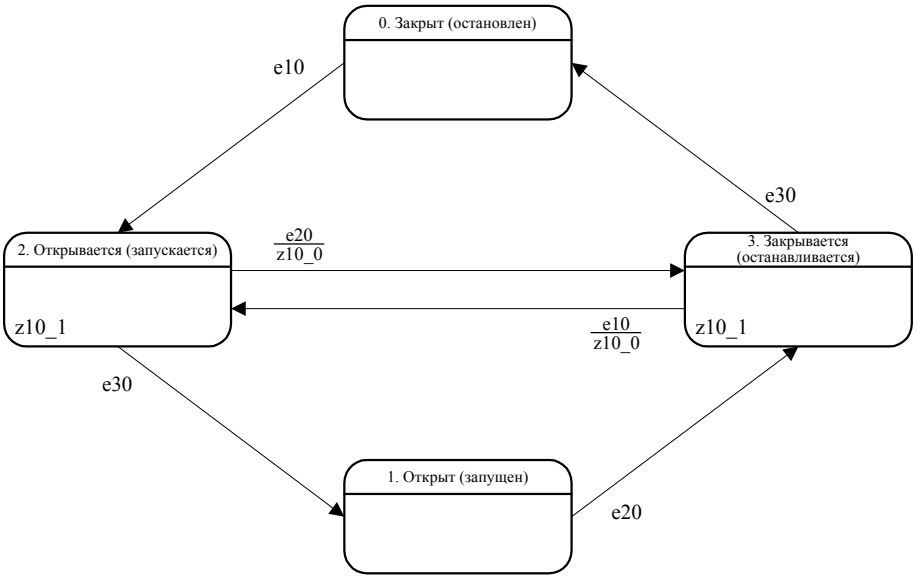

Рис. 24. Граф переходов автомата имитации исполнительного устройства

2.4.4. Текст функции, реализующей автомат

```
#include "photon_stuff.h"
#include "dg.h"

void A10( int e, dg_t *dg, valve_t *v )
{
  int y_old = v->y ;

  switch( v->y )
  {
    case 0:
      if( e == 10 )                 v->y = 2 ;
    break ;

    case 1:
      if( e == 20 )                 v->y = 3 ;
    break ;

    case 2:
      if( e == 30 )                 v->y = 1 ;
      else
      if( e == 20 ) { z10_0(dg,v) ; v->y = 3 ; } ;
    break ;

    case 3:
      if( e == 30 )                 v->y = 0 ;
      else
      if( e == 10 ) { z10_0(dg,v) ; v->y = 2 ; } ;
    break ;
  } ;

  if( y_old == v->y ) return ;

  switch( v->y )
  {
    case 2:
      z10_1(dg,v) ;
    break ;

    case 3:
      z10_1(dg,v) ;
    break ;
  } ;
} ;
```

2.5. Автомат имитации регулятора частоты вращения

2.5.1. Словесное описание

Автомат предназначен для имитации регулятора частоты вращения. При этом автомат обрабатывает команды на уменьшение или увеличение затяжки пружины регулятора, имитируя срабатывание сигнализаторов крайних положений регулятора.

Схема связей автомата и граф переходов приведены на рис. 25, 26.

2.5.2. Схема связей

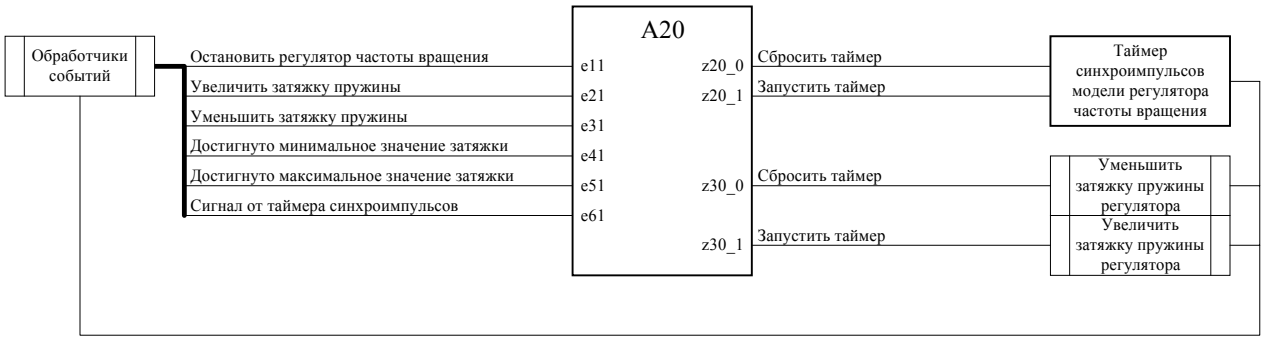

Рис. 25. Схема связей автомата имитации регулятора частоты вращения

2.5.3. Граф переходов

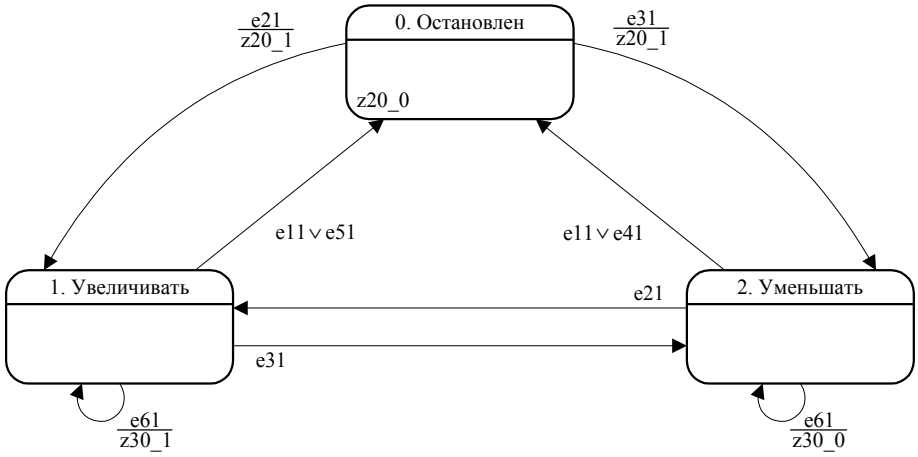

Рис. 26. Граф переходов автомата имитации регулятора частоты вращения

2.5.4. Текст функции, реализующей автомат

```
#include "photon_stuff.h"
#include "dg.h"

void A20( int e, dg_t *dg )
{
  int y_old = dg->y20 ;

  switch( dg->y20 )
  {
    case 0:
      if( e == 21 ) { z20_1(dg) ;    dg->y20 = 1 ; }
      else
      if( e == 31 ) { z20_1(dg) ;    dg->y20 = 2 ; }
    break ;
```

```
    case 1:
      if( e == 11 || e == 51 )       dg->y20 = 0 ;
      else
      if( e == 31 )                  dg->y20 = 2 ;
      else
      if( e == 61 ) { z30_1(dg) ; } ;
    break ;

    case 2:
      if( e == 11 || e == 41 )       dg->y20 = 0 ;
      else
      if( e == 21 )                  dg->y20 = 1 ;
      else
      if( e == 61 ) { z30_0(dg) ; } ;
    break ;
  } ;

  if( y_old == dg->y20 ) return ;

  switch( dg->y20 )
  {
    case 0:
      z20_0(dg) ;
    break ;
  } ;
} ;
```

2.6. Автомат имитации дизель-генератора

2.6.1. Словесное описание

Автомат предназначен для имитации основных состояний дизель-генератора. В нем выделены состояния, перечисленные ниже.

1. Остановлен.

2. Проворот. Имитатор дизель-генератора переходит в это состояние при срабатывании имитаторов сигнализаторов открытых положений клапанов подачи воздуха и подачи воздуха низкого давления. В этом состоянии с заданной частотой имитируется выдача сигналов от датчика проворота коленвала.

3. Запуск. Имитатор дизель-генератора переходит в это состояние при срабатывании имитаторов сигнализаторов открытого положения клапана подачи воздуха и главного пускового клапана. При этом запускается задержка времени пуска, по истечении которой имитируется запуск дизель-генератора.

4. Работа. В этом состоянии выполняется алгоритм имитации изменения частоты вращения, описанный в следующем подразделе.

Схема связей автомата и граф переходов приведены на рис. 27, 28.

2.6.2. Схема связей

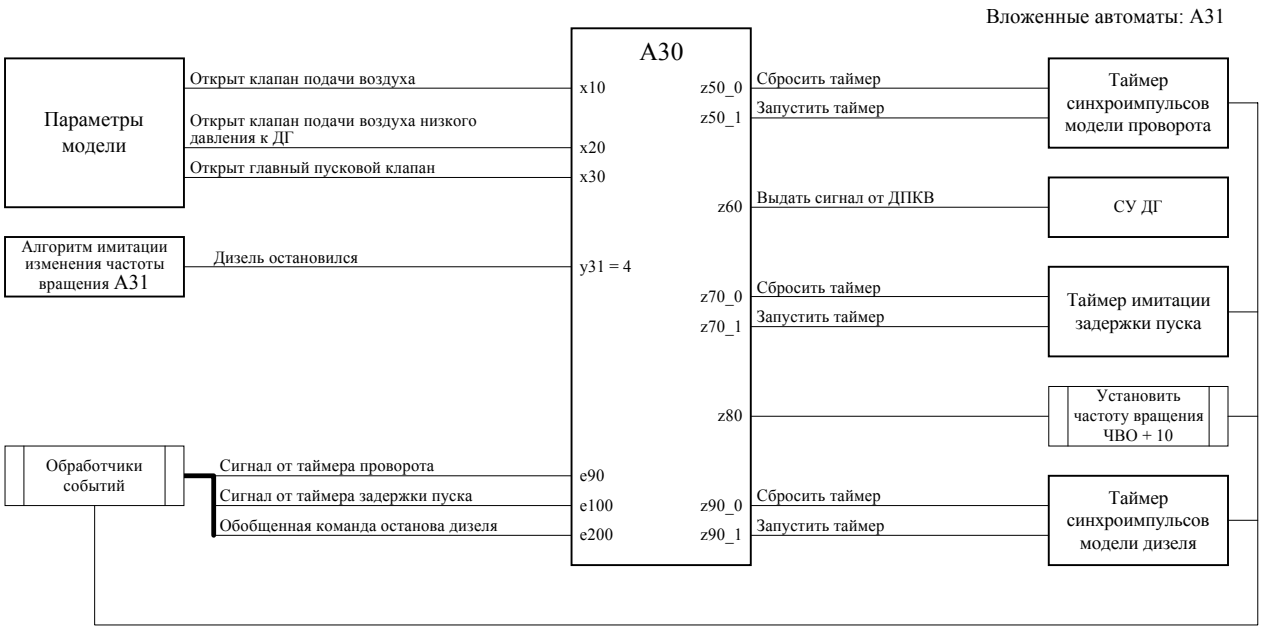

Рис. 27. Схема связей автомата имитации дизель-генератора

2.6.3. Граф переходов

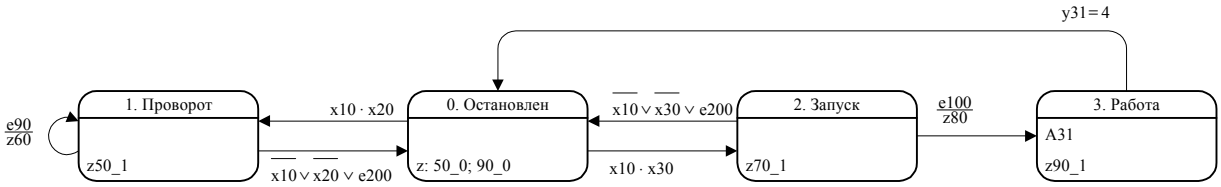

Рис. 28. Граф переходов автомата имитации дизель-генератора

2.6.4. Текст функции, реализующей автомат

```
#include "photon_stuff.h"
#include "dg.h"

void A30( int e, dg_t *dg )
{
  int y_old = dg->y30 ;

  switch( dg->y30 )
  {
    case 0:
      if( x10(dg) && x20(dg) )                dg->y30 = 1 ;
      else
      if( x10(dg) && x30(dg) )                dg->y30 = 2 ;
    break ;

    case 1:
      if( !x10(dg) || !x20(dg) || e == 200 )  dg->y30 = 0 ;
      else
      if( e == 90 ) { z60(dg) ; } ;
    break ;

    case 2:
      if( !x10(dg) || !x30(dg) || e == 200 )  dg->y30 = 0 ;
      else
      if( e == 100 ) { z80(dg) ;              dg->y30 = 3 ; } ;
    break ;
```

```
    case 3:
      A31( e, dg ) ;
      if( dg->y31 == 4 )                    dg->y30 = 0 ;
    break ;
  } ;

  if( y_old == dg->y30 ) return ;

  switch( dg->y30 )
  {
    case 0:
      z50_0(dg) ; z90_0(dg) ;
    break ;

    case 1:
      z50_1(dg) ;
    break ;

    case 2:
      z70_1(dg) ;
    break ;

    case 3:
      A31(0,dg) ;
      z90_1(dg) ;
    break ;
  } ;
} ;
```

2.7. Автомат имитации изменения частоты вращения

2.7.1. Словесное описание

Автомат предназначен для имитации изменения частоты вращения дизель-генератора.

Если текущая частота вращения дизель-генератора отличается от установленной на регуляторе, то имитируется плавное изменение частоты вращения.

При поступлении от системы управления команды на останов дизель-генератора (этой командой считается команда на закрытие наружной захлопки газоотвода), выполняется быстрое уменьшение его частоты вращения до нуля.

Схема связей автомата и граф переходов приведены на рис. 29, 30.

2.7.2. Схема связей

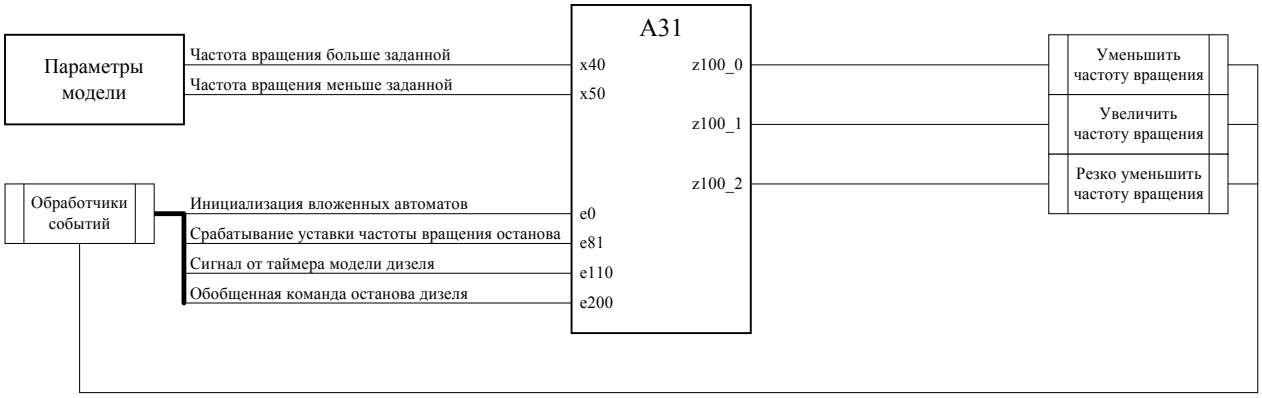

Рис. 29. Схема связей автомата имитации изменения частоты вращения

2.7.3. Граф переходов

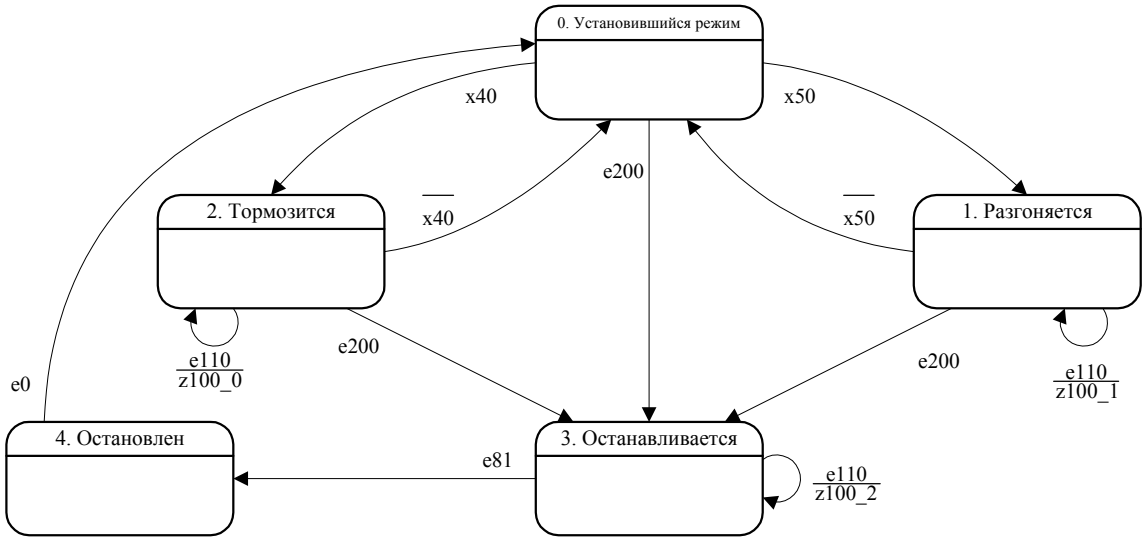

Рис. 30. Граф переходов автомата имитации изменения частоты вращения

2.7.4. Текст функции, реализующей автомат

```
#include "photon_stuff.h"
#include "dg.h"

void A31( int e, dg_t *dg )
{
  int y_old = dg->y31 ;

  switch( dg->y31 )
  {
    case 0:
      if( x40(dg) )           dg->y31 = 2 ;
      else
      if( x50(dg) )           dg->y31 = 1 ;
      else
      if( e == 200 )          dg->y31 = 3 ;
    break ;

    case 1:
      if( !x50(dg) )          dg->y31 = 0 ;
      else
      if( e == 110 ) { z100_1(dg) ; }
      else
      if( e == 200 )          dg->y31 = 3 ;
    break ;

    case 2:
      if( !x40(dg) )          dg->y31 = 0 ;
      else
      if( e == 110 ) { z100_0(dg) ; }
      else
      if( e == 200 )          dg->y31 = 3 ;
    break ;

    case 3:
      if( e == 81 )           dg->y31 = 4 ;
      else
      if( e == 110 ) { z100_2(dg) ; } ;
    break ;

    case 4:
      if( e == 0 )            dg->y31 = 0 ;
    break ;
  } ;
} ;
```

2.8. Обработчики событий

В качестве примера приведем несколько обработчиков событий, вызывающих автоматы.

```
int e30( void *data, pid_t pid, void *msg, size_t msg_size )
{
  valve_t       *v = (valve_t*)data ;
  dg_t          *dg = (char*)v > (char*)&dgs[DG2] ? &dgs[DG2] : &dgs[DG1] ;

  A10( 30, dg, v ) ;

  return Pt_END ;
} ;

int e61( void *data, pid_t pid, void *msg, size_t msg_size )
{
  dg_t          *dg = (dg_t*)data ;

  A20( 61, dg ) ;

  return Pt_CONTINUE ;
} ;
```

2.9. Входные переменные

В качестве примера приведем реализацию нескольких входных переменных.

```
int x10( dg_t * dg )
{
  return dg->air_valve.y == 1 ;
} ;

int x20( dg_t * dg )
{
  return dg->low_air_valve.y == 1 ;
} ;
```

2.10. Выходные воздействия

В качестве примера приведем реализацию нескольких выходных воздействий.

```
void z10_0( dg_t *dg, valve_t *v )
{
  disarm_timer( v->timer_id ) ;
  v->timer_id = -1 ;
} ;

void z10_1( dg_t *dg, valve_t *v )
{
  v->timer_id = arm_timer( v->timeout, 0, 0, 0, e30, v ) ;
} ;

void z80( dg_t *dg )
{
  set_freq( dg, 10 ) ;
} ;
```

2.11. Вспомогательные модули

В качестве примера приведем модуль, реализующий очередь событий.

```
#include "photon_stuff.h"
#include "dg.h"
#include "events_queue.h"

enum                    { QUEUE_SIZE = 1000 } ; // Максимальный размер очереди.
```

```
event_in_queue_t        eq[QUEUE_SIZE] ;        // Массив для хранения очереди.
uint32_t                readpos,                // Индекс последнего считанного.
                        writepos ;              // Индекс последнего записанного.

//======
// Увеличить параметр на 1,
// обнулив его в случае если он превысил
// максимальную длину очереди.
//
uint32_t eq_inc( uint32_t value )
{
  value++ ;
  if( value >= QUEUE_SIZE )
  {
    value = 0 ;
  } ;

  return value ;
} ;

//======
// Записать в очередь новый элемент.
//
void eq_write( uint32_t dg_index, uint32_t e )
{
  if( eq_inc( writepos ) == readpos )
  {
    // Переполнение очереди.

    if( writepos == 0 )
      writepos = QUEUE_SIZE - 1 ;
    else
      writepos-- ;

    return ;
  } ;

  writepos = eq_inc( writepos ) ;
  eq[writepos].dg_num = dg_index ;
  eq[writepos].e = e ;
} ;

//======
// Считать очередной элемент.
//
event_in_queue_t *eq_read()
{
  if( readpos == writepos ) return NULL ;  // Очередь пуста.

  readpos = eq_inc( readpos ) ;

  return &eq[readpos] ;
} ;
```

Заключение

Рассмотрение представленной выше документации свидетельствует о том, что разработанные программы реализуют весьма сложные алгоритмы управления. Их непросто понять, даже используя предложенную технологию, включающую различные схемы, диаграммы и протоколы. Авторам остается только догадываться, насколько возросла бы сложность программирования данной задачи и последующего понимания построенных программ при применении традиционного подхода, в котором для реализации логики используются флаги.

По-нашему мнению, для задач рассматриваемого класса использование объектно-ориентированного подхода не решило бы проблему понимания построенных программ.

Отметим также, что при столь сложной логике, как в рассмотренном примере, применение диаграмм взаимодействий и диаграмм состояний из UML практически невозможно.