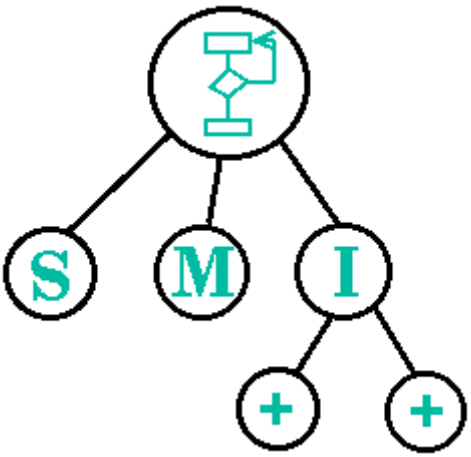
SMI – технология создания распределенных Систем Управления на базе Конечных Автоматов



SMI++ - State Manager Interface
B.Franek, Rutherford Appleton Laboratory
C.Gaspar, EP Department, CERN

Аннотация

SMI++ (*State Manager Interface*) - это программная технология для разработки и реализации распределенных систем управления. Она основана на концепции **Менеджера Конечных Автоматов** (*State Manager*), первоначально разработанного в эксперименте **DELPHI** в **CERN** [1]. В этой концепции реальная физическая система описывается с помощью логических объектов, ведущих себя как Конечные Автоматы (**KA**) или логические машины с конечным числом состояний **FSM** (*Finite State Machines*). Оригинальная идея и реализация созданы благодаря сотрудникам **CERN** (*M.Jonker, B.Franek, A.Vascotto and P. Vande Vyvre*).

Объекты **SMI++** могут работать в распределенной сети на широком наборе программно-аппаратных платформ, всё сетевое взаимодействие прозрачно для прикладного программиста и основано на сетевой технологии **DIM** [2,3]. Это руководство предоставляет обзорное описание функционала **SMI++** и его использования.

С точки зрения прикладного программиста, **SMI** представляет собой набор динамических библиотек (**DLL**) с интерфейсами для разных языков программирования (**C**, **C++**, **Pascal**, …), а также набор служебных программ (**SmiSM**, **SmiTrans**, **SmiGen**, …), распространяемых по лицензии **GPL**.

Система **SMI++** написана на **C/C++**, доступна в исходных кодах, работает под **Windows**, **Linux**, **Unix**. При необходимости она может быть перенесена и на другие платформы. На сетевом уровне **SMI++** построена на технологии **DIM**, которая базируется на сокетах **TCP/IP**.

Система **SMI++** доступна на сайте http://smi.web.cern.ch

Список сокращений

Сокращение	Расшифровка
CERN	Европейский центр ядерных исследований
КА	Конечный Автомат
FSM	Finite State Machine, Конечный Автомат
SM	State Manager, Менеджер Состояний
SMI	State Manager Interface, интерфейс **SM**
SMI++	Программная реализация **SMI** [4,5,6,7,8,9]
DIM	Distributed Information Manager [2,3]
SCADA **СКАДА**	System Control And Data Acquisition Система сбора данных, контроля и управления

Оглавление

Введение

Проблема управления физическими установками, включающими в себя системы сбора данных и системы технологического управления является сложной и комплексной, в первую очередь из-за сложности требуемых операций управления, включающих смесь аппаратных и программных средств.

В технологии **SMI**, решение задач управления достигается на базе Менеджера Состояний (*State Manager*) (**SM**): управление экспериментом имитируется **Конечным Автоматом** (**КА** или **FSM** = *finite state machine*), который представляет логическую модель поведения реального оборудования, которым надо управлять.

Помимо так называемых процедур управления прогоном (*run control*), в эксперименте существует множество других процедур, требующих организации и синхронизации действий, выполняемых независимыми процессами и устройствами, таких как калибровка, управление аппаратурой, первоначальная настройка и т.д. Подход **SM** может также использовать во всех этих случаях.

В эксперименте с ограниченными требованиями можно рассмотреть возможность использования более простых методов управления запуском. Монолитная управляющая программа вполне может справиться с этой задачей, избегая периода обучения, необходимого для правильного использования **SM**. Тем не менее преимущества использования **SM** вполне могут окупиться в долгосрочной перспективе даже для небольшого эксперимента. Аспекты **SM**, которые следует принимать во внимание, - это ее гибкость для адаптации к различным требованиям, присущая модульность модели, принятой для описания эксперимента, тот факт, что описание эксперимента легко модифицируется, функция самодокументирования.

1 Обзор Менеджера Состояний (SM)

Менеджер состояний (*State Manager* - **SM**) по сути представляет собой компьютерную систему управления экспериментом. Эксперимент (набор аппаратных устройств и программ) рассматривается **SM** исключительно через компьютерные процессы, предназначенные для одной конкретной деятельности, называемые ассоциированными (связанными) процессами (*associated process*), именуемыми **прокси** (*proxy - представитель*) (рис. 1).

Основным действием, с помощью которого **SM** может управлять действиями в эксперименте, является обмен сообщениями со связанными процессами прокси. Это ограничение позволяет **SM** иметь единый интерфейс с внешним миром: конкретные механизмы управления выполняются прокси процессами.

SM может управляться командными сообщениями, созданными другими процессами, называемыми процессами управления. Использование интерактивного процесса управления является обычным способом взаимодействия оператора с **SM**.

SM включает в себя модель различных **действий** (*action*), которые необходимо организовать, что по сути является описанием эксперимента. Таким образом, задача создания системы управления эквивалентна проблеме хорошего описания эксперимента с точки зрения контролируемых **объектов** (*object*) и процедур, которые с ними

воздействуют. Программа **SM** является реализацией модели, определенной описанием эксперимента. Это описание, сделанное физиком в терминах языка **SM**, обрабатывается инструментами **SMI** для создания **State Manager** и связанных с ним прокси.

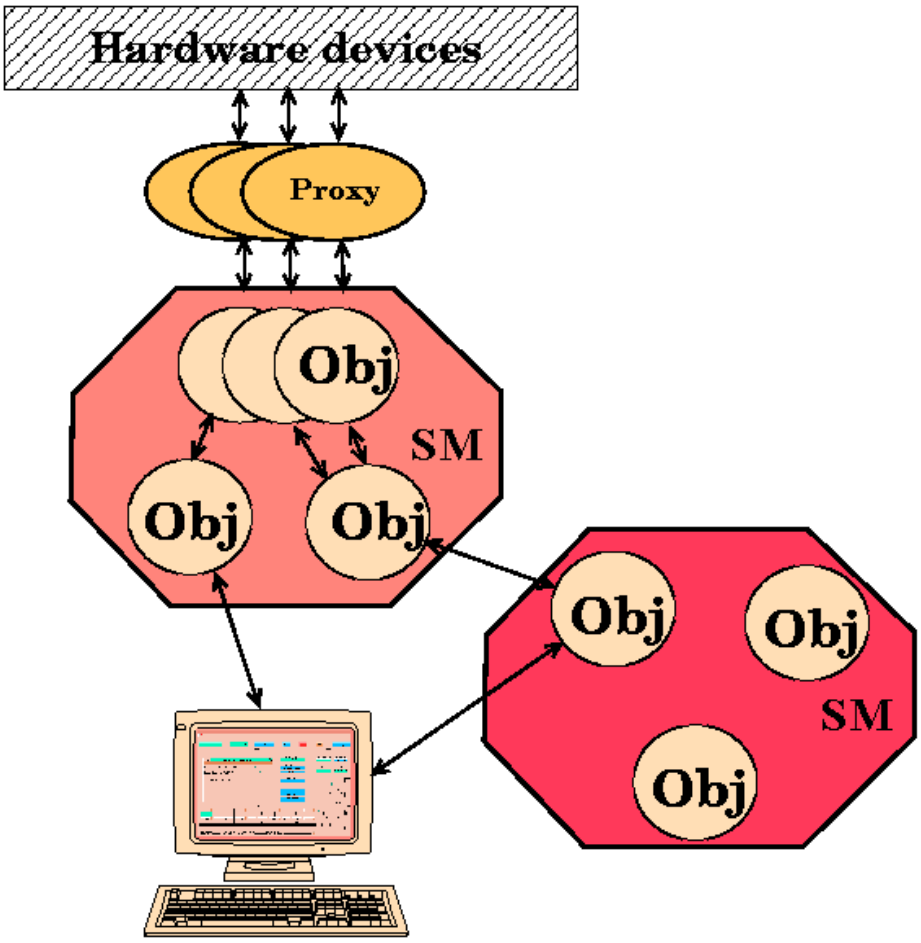


Рисунок 1. Среда окружения **SMI**

1.1 Модель объектов/сообщений

Подход, принятый для моделирования эксперимента, основан на объектно-ориентированной декомпозиции. Эксперимент описывается с точки зрения объектов: **объект** (*object*) может напрямую соответствовать конкретному объекту в эксперименте (например, «газовый клапан 12»); он может в равной степени представлять собой логическую подсистему или любую абстракцию, используемую при описании эксперимента, при условии, что он может быть идентифицирован существительным (например, «прогон» **RUN**, «триггер» **TRIGGER**, «центральный детектор» **CENTRAL_DETECTOR**).

По соглашению, имена **SMI** содержат латинские буквы, цифры и знак подчеркивания, и начинаются с буквы или знака подчеркивания. Имена не чувствительны к регистру символов. Эти соглашения можно описать шаблоном регулярного выражения **/[a-zA-Z_]+[a-zA-Z_0-9]*/i**.

Объектная модель близко соответствует нашей собственной модели реальности; поэтому эксперимент можно описать с помощью тех же понятий и терминов, которые мы используем, думая о нем.

Главным атрибутом объекта является его **состояние** (*state*), единственное, которое видно другим объектам. Состояние — это переменная, которая может принимать любое значение из списка значений, перечисленных разработчиками. Обычной практикой является определение значений, соответствующих прилагательным, которые могут уточнять имя объекта (например, объект «запуск» может находиться в одном из состояний «активен», «неактивен» или «приостановлен» - **active**, **dormant**, **paused**). Объект воздействует на другие объекты, посылая им командные сообщения. Коммуникационная сетка, связывающая объекты, несет в себе глобальный поток управления в системе. Опять же, эта модель взаимодействия между объектами отражает нашу абстракцию реальности. Это позволяет одновременно выполнять операции и развивать множество потоков управления.

1.2 Поведение объектов

Объекты (*object*) ведут себя как конечные автоматы. Изменение состояния (*state*) вызывается получением команды. После принятия команда запускает выполнение действия (*action*); это в конечном итоге прекращается, когда объект достигает нового устойчивого состояния. Действие определяется глаголом, применимым к имени объекта (например, объект «запуск» (RUN) может выполнять любое из действий «запустить» ***start***, «приостановить» ***pause***, «продолжить» ***continue*** и «остановить» ***stop***). Соответствующая команда определяется тем же глаголом, который можно рассматривать как используемый в повелительном наклонении.

Действие состоит из последовательности операций, заданной на языке **SML** (*state manager language*) списком инструкций. По сути, они бывают двух типов: **DO** и **IF**. **DO** — инструкция, посылающая команду другому объекту; **IF** — это инструкция, которая оценивает логическую функцию состояний других объектов и делает условный переход в зависимости от результата. Пока объект выполняет действие, его состояние равно, и никакие дальнейшие команды не принимаются до тех пор, пока действие не будет завершено. Конечное состояние, достигаемое после выполнения действия, может зависеть от результатов проверки состояния других объектов.

Команда — не единственный способ инициировать действие: его может спровоцировать изменение состояния другого объекта; этот тип зависимости определяется в языке **SML** предложением **WHEN**.

1.3 Взаимодействие с внешним миром

SM взаимодействует с другими процессами посредством обмена сообщениями. Процесс управления может послать сообщение любому объекту; **SM** отправляет команду и инициирует действие.

Прокси (*proxy*) видны **SM** через объект особого (ассоциированного или связанного) типа в описании эксперимента, помеченный соответствующим атрибутом (*associated*). Они отличаются тем, что не могут самостоятельно выполнять никаких действий **SML**: команда, полученная ассоциированным объектом, отправляется соответствующему прокси для непосредственного выполнения. И наоборот, изменение

состояния, происходящее в объекте, контролируемом прокси, точно отражается связанным с ним объектом. Ассоциированные объекты и связанные с ними прокси являются посредниками между реальным миром (аппаратурой) и логической моделью **SM**. Ассоциированные объекты (***associated object***) внутри **SM** являются агентами, позволяющими управлять внешним миром. Связь между **SM** и другими процессами использует базовый пакет **DIM** [2,3]; поэтому естественным образом поддерживается распределенная гетерогенная многомашинная конфигурация. В такой среде на одной из машин работает один процесс **SM**, а другие процессы могут выполняться на любой другой машине в конфигурации. Базовый коммуникационный пакет использует соглашения об именах для правильной адресации удаленных объектов.

Независимо от принятой физической конфигурации (одномашинная или многомашинная) в системе можно запустить более одного **SM**. Это достигается разделением системы на домены, как описано в следующем разделе.

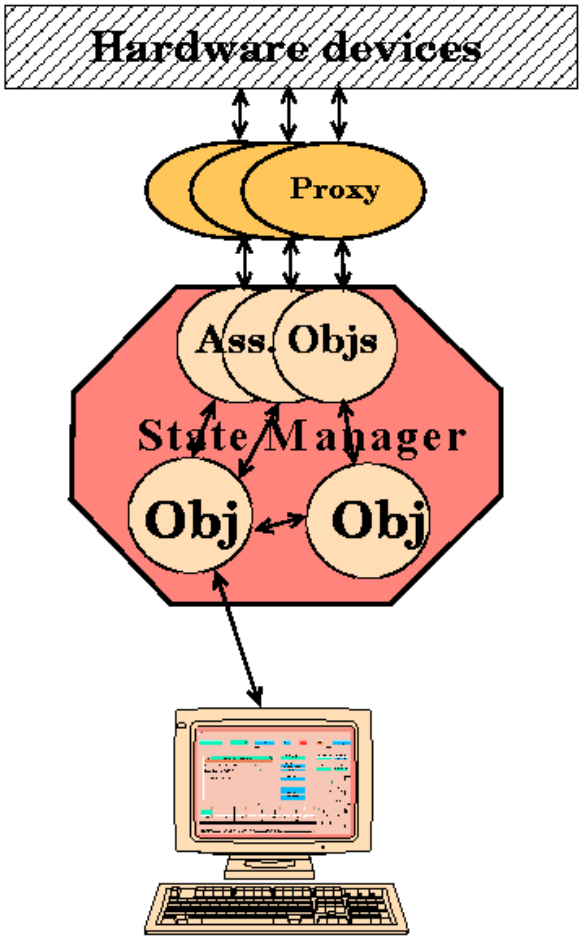


Рисунок 2. Взаимодействие **SMI** с внешним миром.

1.4 Домены менеджера состояний SM

Домен (*domain*) определяет область видимости имен связанных объектов. Поскольку адресация сообщений между **SM** и другими процессами основана на именах объектов, домен устанавливает границы пространства управления **SM**. Это достигается путем присвоения имени каждому домену. Задачей коммуникационного пакета является использование доменного имени и обращение к правильному объекту, указанному парой домен-объект. Доменная структура накладывается совершенно прозрачно; описание эксперимента не содержит упоминания домена, и связанные процессы не нуждаются в каком-либо специальном коде для его указания.

Концепция домена может быть применена, например, к случаю двух **SM**, работающих на одной машине; один может тестироваться, а другой используется для управления рабочим процессом. Разделение системы позволяет избежать помех, когда **SM** обращаются к различным прокси-серверам с одинаковыми именами.

По соглашениям **SMI**, пути (полные имена) объектов с учетом доменов выглядят как **DOMAIN::OBJECT**, например, **EGP::LOGGER** – объект **LOGGER** в домене **EGP**.

1.5 Иерархия доменов

Ссылки могут быть установлены между отдельными доменами, чтобы обеспечить определенную степень видимости между **SM** (рис. 3).

Этот тип отношений может быть только иерархическим; ведомый **SM** не знает, кем он управляется, в то время как ведущий **SM** должен знать, каким ведомым **SM** он управляет, поскольку он может управлять более чем одним одновременно. Ссылки могут быть установлены только в определенных точках подчиненных **SM**. Эти точки являются объектами, помеченными атрибутом **visible**: они позволяют ведомому **SM** быть видимым другим **SM** как связанный процесс. Следовательно, главный **SM** может объявить ассоциированный объект (в своем собственном домене), который будет имитировать поведение видимого объекта в другом домене. Все команды, направленные на связанный объект, будут отправлены на видимый объект, и изменения состояния видимого объекта будут воспроизведены ассоциированным объектом. Домен необходимо указывать при обращении к видимому объекту, поскольку в разных доменах могут находиться видимые объекты с одинаковым именем; это достигается путем префикса имени объекта с именем домена, которые разделяются двойным двоеточием (например, **2::A** означает объект **A** в домене **2**).

Примером применения этих функций является эксперимент, состоящий из подсистем (например, детекторов), которые можно запускать независимо: ими должны управлять отдельные **SM**. В конце этапа настройки две подсистемы будут интегрированы, и эксперимент будет запущен как единое целое. Нет необходимости модифицировать исходные **SM**: ими может управлять третий.

В терминах языка **SML** взаимодействие доменов происходит примерно так.

В файле **domain2.sml** содержится описание объекта **A**:

```
object:  A
   state:  ON
      ...
```

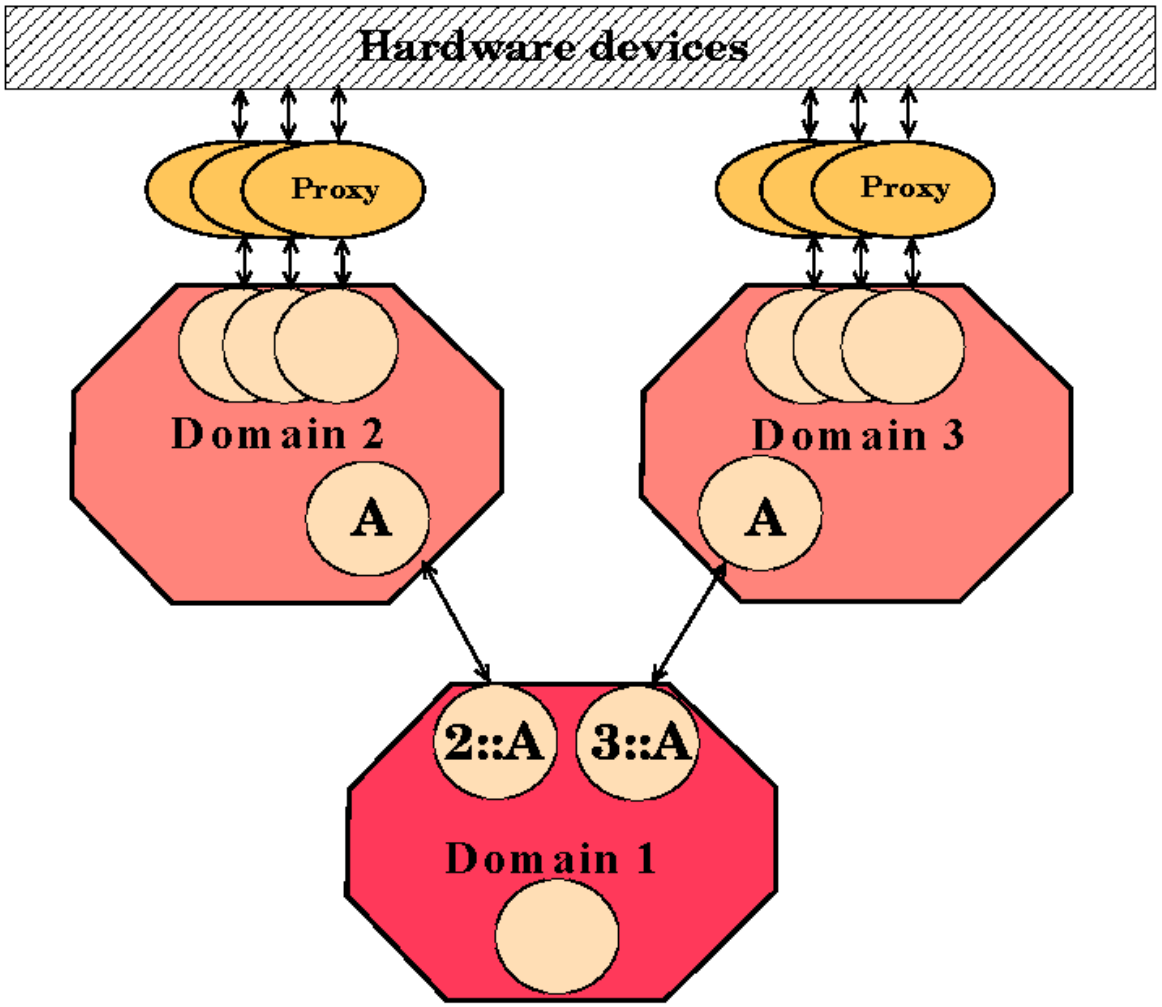


Рисунок 3. Иерархия доменов **SMI**.

Аналогичное описание содержится в файле **domain3.sml**. Описание объектов является содержательным, т. е. содержит реализацию в виде кода (инструкций **SML**).

В файле **domain1.sml** домена верхнего уровня содержится описание ассоциированных объектов в виде:

```
object:  2::A /associated
   state:  ON
      ...
object:  3::A /associated
   state:  ON
      ...
```

Ассоциированные объекты имеют имена **2::A** и **3::A** специального вида, содержащего в себе ссылку на домен. Они должны иметь такой же набор состояний и действий, как исходные объекты, но не должны иметь кода реализации, т. к. их логику реализуют объекты, с которыми они ассоциированы. Таким образом, логический объект (**A** в домене **2**) может иметь «аватара» (заместителя) в другом домене (**2::A** в домене **1**), который представляет поведение ведомого домена (подсистемы) в ведущем домене (системе). При этом в коде **domain1.sml** ссылка на прокси объект **2::A** повлечет за собой обращение к объекту **A** из домена **2** (через внутренние средства **SMI**).

1.6 Проектирование системы управления (обзор)

Применение концепций **SM** к эксперименту должно состоять из ряда шагов, в ходе которых проблема управления постепенно сужается до управляемого уровня сложности.

Прежде всего, следует указать, в каких широких областях необходима независимая управляющая деятельность (независимая означает, что области могут управляться независимо, но взаимодействие между различными областями не исключается); эти области в конечном итоге станут доменами **SM**. Примерами этих областей в эксперименте являются «управление прогоном» (*run control*) или «управление детектором» (*detector control*); если аппаратура состоит из нескольких детекторов, каждый из них может принадлежать независимому домену **SM**; особенно сложная деятельность может быть разделена на отдельные домены; согласованное выполнение всего эксперимента может потребовать перекомпоновки задач с помощью иерархической структуры, охватывающей несколько доменов. Существует широкая свобода выбора, но важно, чтобы каждая область была тесно связана с объектами или группами действий, которые имеют значение в контексте эксперимента. Затем для каждого домена **SM** следует определить, какие элементы являются управляемыми: это реальные объекты, которыми можно управлять с помощью команд и которые способны принимать состояние; для каждого из них должна быть предоставлена сервисная программа, которая будет управлять всей деятельностью, связанной с этим объектом, в соответствии с инструкциями, полученными **SM** (программа будет связана с объектом в **SM**). Экспериментаторам может оказаться полезным определить другие объекты, связанные с логическими объектами (называемые в данном руководстве логическими объектами, а не ассоциированными объектами), чтобы представить различные режимы проведения эксперимента (такие как «сбор данных», «калибровка» и т.д.). и т.д.) или наборы более простых объектов (таких как «центральный детектор», который, например, может быть «готов», когда готовы все компоненты реального центрального детектора).

Модель поведения данной области должна быть сформулирована в виде программы **SM** (обычный текстовый файл, который можно написать в любом текстовом редакторе). Существует простой язык **SML** (*state manager language*), который содержит все примитивы, подходящие для этой цели (см. пример на рис. 4).

Программа **SM** описывает предварительно определенные объекты, состояния, которые они могут принимать, и команды, которые они принимают в каждом состоянии. Затем действия, генерируемые каждой командой, должны быть описаны: они состоят из последовательности инструкций, каждая инструкция является исполняемым оператором языка описания эксперимента. Основными типами инструкций являются **DO** и **IF**. **DO** – инструкция, посылающая команду другому объекту и, следовательно, позволяющая управлять процессом иерархически зависимых объектов; **IF** – это инструкция, которая допускает условное ветвление по состоянию других объектов. Конечно, в языке разрешены логические выражения, относящиеся к состоянию многих объектов. Язык также содержит оператор **WHEN**, который позволяет **SM** инициировать действие при незапрошенном изменении состояния объекта. Сервисные программы также должны быть подготовлены (на

других языках программирования): они являются посредниками (*proxy*) в терминологии, используемой в данном руководстве. Эти программы работают на аппарате при получении командных сообщений. Их поведение должно быть представлено конечным автоматом. Генерации кода, реализующего конечный автомат, помогает набор инструментов **SMI**; с этой целью поведение связанного объекта должно быть полностью описано сверх того, что обычно требуется **SM**. Функциональная часть сервисной программы остается за программистом, связь с программой **SM** обеспечивается набором библиотечных подпрограмм (см. Приложение 2,3).

```
object : RUN
    state : IDLE
        action : START
            do BEGIN EVT_BUILDER
            if (EVT_BUILDER in_state READING) then
                do ACTIVATE DATA_LOGGER
                move_to ACTIVE
            else
                move_to ERROR
            endif
    state : ERROR
        action : RESET
            do RESET EVT_BUILDER
            move_to IDLE
    state : ACTIVE
    when (DATA_LOGGER in_state END_OF_TAPE) do STOP
        action : STOP
            do END EVT_BUILDER
            do DEACTIVATE DATA_LOGGER
            move_to IDLE

object : DATE_LOGGER/associated
    state : AVAILABLE
        action : ACTIVATE
    state : WRITING
        action: DEACTIVATE
    state : END_OF_TAPE
        action : DEACTIVATE

object : EVT_BUILDER/associated
    state : IDLE
        action : BEGIN
    state : READING
        action : END
    state : ERROR
        action : reset
```

Рисунок 4. Пример программы SML.

Этап реализации состоит из генерации **SM** (одного или нескольких) и всех связанных процессов. Описание **SM** на языке **SML** преобразуется в промежуточный объектный формат (***.sobj**), который затем считывается во время выполнения общим механизмом **SM**, который

создает экземпляры всех объектов **SMI** и управляет моделью **SM**. Скелеты (прототипы) прокси (*proxy skeleton*) также генерируются из описания **SM**.

Система времени выполнения состоит из одного серверного процесса **SM** (программа **smiSM.exe**) на каждый домен **SM** и всех его прокси. Порядок активации не имеет значения, поскольку механизмы, с помощью которых **SM** и его партнерские процессы узнают друг о друге, не зависят от того, кто запустился первым.

2 Описание языка SML

Объектная модель эксперимента описывается с помощью **State Manager Language** (**SML**). Этот язык позволяет подробно описывать объекты, такие как их состояния, действия и связанные с ними условия. Основными характеристиками этого языка являются:

- **Логика конечного состояния**
  Объекты (*object*) описываются как **Конечные Автоматы** (*Finite State Machine* - **FSM**). Единственным атрибутом объекта **FSM** является его **состояние** (*state*). Команды, отправленные объекту, вызывают **действия** (*action*), которые могут привести к изменению его состояния.
- **Последовательность действий**
  Действие над абстрактным объектом задается последовательностью инструкций, в основном состоящей из команд, посылаемых другим объектам, и логических проверок состояний других объектов. Действия над конкретными объектами отправляются в виде сообщений процессам управления драйверами.
- **Асинхронный режим**
  Несколько действий могут выполняться параллельно: команда, отправленная объектом A объекту B, не приостанавливает последовательность инструкций объекта A. Только проверка объектом A состояния объекта B приостанавливает последовательность инструкций объекта A, если объект B все еще находится в переходном состоянии.
- **AI-подобные правила**
  Каждый объект может задавать логические условия на основе состояний других объектов. Когда они удовлетворены, они вызовут действие на локальном объекте. Это обеспечивает механизм ответа объекта на незапрошенные изменения состояния его среды.

2.1 Алфавит и идентификаторы SML

Для построения выражений **SML** используются символы латинского алфавита [**a-zA-Z**], цифры [**0-9**], знак подчеркивания [**_**], а также символы [**:{},"**]. Символы [**!#**] используются в качестве комментариев. Идентификаторы доменов, классов, объектов, состояний, действий и других элементов или сущностей (***entity***) содержат латинские буквы, цифры и знак подчеркивания, и начинаются буквы или знака подчеркивания (но не с цифры), удовлетворяя шаблону регулярного выражения **/[a-zA-Z_]+[a-zA-Z_0-9]/i**.

2.2 Комментарии и расширенный синтаксис SML

Комментарии в языке **SML** отделяются символами **!** или **#**. Однако, поскольку файлы **SML** используются как единый источник кода для исполнения сервера и клиента, обслуживающего графический интерфейс **GUI**, в комментариях можно хранить дополнительные параметры объектов и состояний, которые рассматриваются как комментарии сервером, но обрабатываются клиентским кодом **GUI**. Эти соглашения образуют расширенный синтаксис **SML**.

В расширенном синтаксисе **SML** безусловным комментарием является только текст после символа **#**. В то же время комментарий, отделенный символом **!**, может служить для задания дополнительных параметров (опций, атрибутов) клиента, в виде выражения:

```
!name: value
```

для параметра с именем **name** и значением **value**.

Например:

```
object: LOGGER /associated
    state: OFF  !color: Silver
                # задает серый (silver) цвет (color) состояния
```

Расширенный синтаксис позволяет расширять возможности базового языка **SML**, не нарушая его правил, и в то же время задавать необходимые для клиентской части параметры. Совмещение кода сервера и клиента **SML** в одном файле имеет большие преимущества с точки зрения сохранения непротиворечивости и целостности проекта, т. к. при разделении проекта на несколько разных файлов возникает проблема синхронизации версий, которая может привести к дополнительным ошибкам (несогласованности кода сервера и клиента). Единый источник кода устраняет эту проблему.

В настоящее время в коде **SML** применяются следующие опции, служащие для организации графического интерфейса (**GUI**):

```
#############################################################################
#  !visible: v      - Флаг видимости действий (v=0/1) в меню.
#  !color: c        - Цвет (c) для отображения состояния.
#  !panel: p        - Имя панели (p), т. е. окна мнемосхемы данного объекта.
#  !title: t        - Заголовок (t) для отображения имени или состояния.
#  !busy_title: t   - Заголовок (t) для отображения состояния BUSY.
#  !busy_color: c   - Цвет (c)      для отображения состояния BUSY.
#  !confirm: c      - Флаг (c=0/1) подтверждения для выполнения действия.
#############################################################################
```

2.3 Декларативные выражения SML

- **Class**
- **Object**
- **ObjectSet**
- **Action**
- **Function**
- **State**

Декларативные выражения отличаются тем, что они создают элементы структуры **SM** (объекты, наборы объектов, состояния, действия) для последующего использования во время работы Менеджера Конечных Автоматов - **SM**.

2.3.1 Объявление CLASS

Формат объявления:

```
class: class-name [/associated]
```

Это объявление используется для описания прототипа объекта - группы однотипных объектов. Описание группы начинается с объявления класса, в котором также указываются некоторые характеристики объектов. Сразу же должны следовать все выражения, касающиеся группы объектов. Для ассоциированного класса (*associated*) применяются те же соображения, что и для ассоциированного объекта (см. объявление **OBJECT**). Например:

```
class: TOP_cooling_top_CLASS
class: phs_col_dp_CLASS /associated
```

После объявления класса за ним могут быть объявлены параметры, см. раздел 2.3.2.1 «Параметры объекта».

В расширенном синтаксисе после объявления класса можно задавать дополнительные параметры (для клиентской программы обслуживания графического интерфейса **GUI**) в комментариях с помощью выражений вида **!name: value**, как описано в разделе 2.2 .

В целом описание класса аналогично описанию объекта, т. к. класс является фактически описанием группы похожих объектов.

Классы являются **абстрактными** элементами **SML**. Это значит, что экземпляры классов не создаются и не используются при работе сервера **SM**. Классы нужны только для объявления объектов с помощью конструкции **is_of_class**. Фактически классы – это просто способ избежать многократного повторения описаний похожих объектов. Например, если имеется несколько однотипных источников бесперебойного питания (**UPS**), то можно описать класс **UPS_CLASS** и затем объявить несколько объектов этого класса:

```
class: UPS_CLASS
   … описание класса UPS_CLASS …
object: UPS1 is_of_class UPS_CLASS
object: UPS2 is_of_class UPS_CLASS
…
```

Без использования класса в этом примере каждый объект **UPS** надо было бы описывать отдельно, что привело бы к дублированию кода.

2.3.2 Объявление OBJECT

Описание объекта начинается с объявления **OBJECT**, которое, кроме его имени, указывает также некоторые другие характеристики объекта. Все выражения, относящиеся к одному объекту, должны следовать непосредственно за его объявлением.

Формат объявления:

```
object: object-name [is_of_class class-name] [/associated]
```

Связанный (*associated*) объект не выполняет сам действие, которое он получает. Он передает команду связанному с ним прокси процессу и обновляет свое состояние в соответствии с изменением состояния связанного процесса. Поэтому код **SML** ассоциированного объекта должен содержать только список состояний и для каждого из них список действий, принимаемых в каждом состоянии. Связанный процесс может принадлежать домену, отличному от домена самого **SM**: в этом случае домен должен быть явно указан как префикс имени объекта, отделенный от него двойным двоеточием (например, **TPC::HV** указывает на объект **HV** в домене **TPC**).

При использовании конструкции **is_of_class** отдельное описание объявляемого объекта не требуется, т. к. он уже фактически был описан при объявлении класса. Объект указанного класса наследует всё описание (параметры, состояния, действия) из класса, который указан в конструкции **is_of_class**.

2.3.2.1 Параметры объекта

Оба типа объекта (абстрактный и ассоциированный) могут иметь связанные с ним параметры. Одним из наиболее важных их применений является передача значений между миром **SMI** и соответствующими прокси - серверами. Когда у объекта есть параметры, за объявлением объекта должно следовать сразу:

```
Parameters: parameters-declaration
```

Подробнее об объявлении параметров см. раздел 2.5.3 .
Например:

```
object: RUN
    parameters: int RUN_NUMBER = 1,
                float ENERGY = 1.0,
                string RUN_MODE = "DEMO"
```

Параметры могут иметь тип **int**, **float**, **string**. При объявлении можно указывать значения по умолчанию. Параметры разделяются запятыми, при этом допускается после запятой делать перенос строки.

В расширенном синтаксисе после объявления объекта можно задавать дополнительные параметры (для клиентской программы обслуживания графического интерфейса **GUI**) в комментариях с помощью выражений вида **!name: value**, как описано в разделе 2.2 .

2.3.3 Объявление ObjectSet

Можно группировать объекты в группы (наборы) и затем выполнять действия и тесты над всеми членами набора.

Формат объявления:

1) Простой ObjectSet

ObjectSet: *set-name* [{obj$_1$,obj$_2$,...obj$_n$}]

где необязательная конструкция **{obj$_1$,obj$_2$,…,obj$_n$}** инициализирует набор перечисленными объектами.

Объект может быть динамически вставлен (удален) в объявленный набор. Это достигается выполнением следующих 3 инструкций:

insert *object-name* in *set-name*

remove *object-name* from *set-name*

removeAll from *set-name*

2) Объединенный набор union

Можно определить набор объектов, который является объединением других наборов объектов. Это достигается с помощью ключевого слова «**union**» в объявлении набора объектов:

ObjectSet: *set-name* union {*set-name$_1$*,*set-name$_2$*,…,*set$_n$*}

Пара замечаний:

1. На данный момент Union Set не может быть членом другого Union Set. Объявление объединенного набора является статическим в отношении его состава с точки зрения наборов объектов. Однако он динамичен в том, что касается его состава с точки зрения Объектов. См. примечание 2.
2. Вставка/удаление объектов непосредственно в/из Union Set не разрешена, так как это не имеет особого смысла. Это связано с тем, что объект может принадлежать более чем одному простому набору объектов в объединении. Результаты операций вставки/удаления, выполняемых непосредственно над содержащимися в них простыми наборами объектов, точно отражаются в содержимом объектов объединенного набора. В частности, когда объект O удаляется/вставляется из/в простой набор объектов S, он затем удаляется/вставляется из/в объединенный набор U в зависимости от того, принадлежит ли O также какому-либо другому набору объектов в объединении. Кроме того, аналогичным образом обрабатываются все остальные наборы соединений, которые содержат набор объектов S.

Использование в простых условиях

```
( any_in set-name in_state {state-list} )

( all_in set-name in_state {state-list} )

( set-name empty )
```

Примечания:

1. В приведенном выше также допустимы значения **not_in_state** и **not_empty**.
2. Эти простые условия могут быть частью любого сложного условия.

Использование в инструкции DO

```
do action-name all_in set-name
```

Например:

```
do SWITCH_ON all_in POWER_SUPPLIES
```

См. также примеры из Приложения 5,6,7,8.

2.3.4 Объявление Action

Действие (*action*) – это именованная группа инструкций, предназначенных для последовательного выполнения и относящихся к определенному состоянию объекта. Действие вызывается снаружи объекта инструкциями **DO** в других объектах или внешними командами. Эти команды помещаются в очередь действий объекта и выполняются Планировщиком в установленном порядке. Объект выполнит действие только тогда, когда он находится в соответствующем состоянии.

Действия для конкретного состояния объявляются сразу после объявления состояния. (см. ниже). Объявление действия начинается с ключевого слова **Action**, за которым следует двоеточие. Затем следует название действия. Это должно быть уникальным среди действий состояния.

Действие может принимать параметры, заключаемые в скобки (…). Когда у действия есть параметры, последует объявление параметров. Подробнее об объявлении параметров см. раздел 2.5.3 .

Для логического объекта за объявлением **ACTION** должны следовать инструкции (исполняемые операторы), которые объект выполняет при получении команды. С другой стороны, связанный (*associated*) объект не выполняет никаких действий, а отправляет команду в виде строки сообщения связанному с ним прокси процессу; поэтому после объявления **ACTION** у связанного объекта нет исполняемых операторов.

Синтаксис объявления Action:

```
Action: action-name [( parameters-declaration )]

   [SML Instructions]
```

2.3.4.1 Расположение объявлений Action

Объявления **Action** (свой список для каждого состояния объекта) следуют после объявления объекта, объявления его параметров и объявления состояния:

```
Object: obj-name
Parameters: ….
            ….
State: state1
    Action: action1
                …
    Action: actioni
                …
State: statei
    Action: action1
                …
    Action: actioni
                …
```

См. также примеры из Приложения 5,6,7,8.

2.3.5 Объявление Function

Функция – это именованная группа инструкций, предназначенная для последовательного выполнения и принадлежащая объекту. В этом отношении это точно так же, как Действие (*action*). Отличие состоит в том, что Действие принадлежит определенному состоянию объекта, а Функция принадлежит объекту в целом. Еще одно отличие заключается в их вызове. Действие вызывается извне объекта инструкциями **DO** в других объектах или внешними командами путем помещения действия в очередь действий объекта, в то время как, с другой стороны, Функция вызывается из других действий того же объекта командой **CALL** и выполняется немедленно как часть этих действий.

Функции объявляются сразу после объявления параметров объекта. (см. ниже). Объявление функции начинается с ключевого слова **Function**, за которым следует двоеточие. Затем следует имя функции. Это должно быть уникальным среди функций объекта.

Функция может принимать параметры так же, как и действие. Когда у функции есть параметры, последует объявление параметра. Подробнее об объявлении параметров см. Параметры в **SMI**.

Формат объявления Function:

```
Function: function-name [( parameters-declaration )]

[SML Instructions]
```

2.3.5.1 Расположение объявлений Function

Объявления **Function** (свой список для каждого объекта) следуют после объявления объекта, объявления его параметров и до объявления состояний:

```
Object: obj-name
Parameters: …
   Function: fun1
      …
   Function: funi
      …
State: state1
   …
```

2.3.6 Объявление State

Этот оператор используется для объявления одного из возможных состояний, которые может принимать объект. За объявлением состояния могут следовать все операторы **WHEN** и **ACTION**, применимые к этому состоянию.

Формат объявления **State**:

State: ***state-name*** **[/initial_state] [/dead_state]**

где:

- **initial_state** (начальное состояние) является квалификатором состояния, семантика которого зависит от того, связан объект или нет. В случае логических объектов это указывает, что объект должен быть переведен в данное состояние при запуске программы **SM**. В случае ассоциированных объектов помеченное состояние — это состояние, которое изначально принимает объект, когда с ним связывается процесс. При запуске **SM** объект также может принимать это состояние, если связанный с ним процесс запущен и сам еще не указал состояние. По умолчанию начальное состояние объявляется первым.
- **dead_state** (мертвое состояние) является квалификатором состояния, который применяется только к связанным объектам. Он определяет состояние, которое должно быть присвоено объекту, когда связанный с ним процесс прерывается или вообще не выполняется. Поведение объекта также зависит от того, есть ли состояние, помеченное этим квалификатором:
  - **состояние не отмечено** - когда связанный процесс прерывается, объект переходит в зависшее состояние, запрещающее любые дальнейшие операции. Все команды, поступающие к этому объекту (включая операторы **IF**, проверяющие его состояние), будут помещены в очередь (исходные действия будут приостановлены) до тех пор, пока связанный процесс не возобновит операции. Нет никаких средств, с помощью которых другие объекты могли бы проверить это условие.
  - **одно из состояний отмечено** - тогда объект остается активным, даже если связанный с ним процесс не запущен. Полученные команды принимаются и отбрасываются (поскольку нет активного партнера); операторы **IF** работают как обычно, а именно любой объект может проверить это состояние.

В расширенном синтаксисе после объявления **state** допускается указывать дополнительные параметры для клиента **GUI**, наиболее часто используемым из которых является цвет (*color*), например:

```
object: LOGGER /associated
    state: DEAD /dead_state                  !color: DarkGray
    state: NOT_LOGGING                       !color: Aqua
        action: LOG
    state: LOGGING                           !color: Lime
        …
```

См. также примеры из Приложения 5,6,7,8.

2.4 Исполняемые выражения

Исполняемые выражения отличаются от декларативных тем, что они не создают структуры данных для будущего использования, а выполняют действия над объектами непосредственно.

2.4.1 Инструкция DO

Эта инструкция позволяет объекту инициировать действия над другим объектом (целевым объектом) или даже над самим собой. Он просто помещает команду для выполнения действия в очередь действий цели. Затем это выполняется планировщиком в установленном порядке.

Синтаксис вызова:

```
do action-name [( parameters-declaration )] target-object-spec
```

где

action-name
имя одного из действий, которое должно быть выполнено в целевом объекте

parameters-declaration
действие в целевом объекте может ожидать параметры, указанные в его объявлении. Значения этих параметров приведены здесь. Затем они вместе с командой доставляются к целевому объекту. Подробнее см. раздел 2.5.3 .

target-object-spec
спецификация целевого объекта, является одним из следующих:
или *object-name* (имя объекта **SMI**)
или **all_in** *objectset-name* (все в наборе объектов **SMI**)
это выражение отправит команду всем объектам в наборе.

NB: Для более расширенного использования выражения **DO** для экспертов см. также раздел 2.5.5 .

Пример 1

```
Action: START_RUN (new_run_number)
      …
      do START (run_number=new_run_number)  RUN_CONTROL
```

Пример 2

```
Action: START
      …
      do SWITCH_ON (run_number=78)  all_in  POWER_SUPPLIES
```

2.4.2 Инструкция CALL

Эта инструкция вызывает выполнение именованного набора инструкций - **Function** (см. раздел 2.3.5). Иными словами: когда исполняемое действие достигает инструкции **CALL**, оно продолжает выполнение с инструкциями, указанными в объявлении функции, а когда функция достигает своего конца, действие продолжает выполнять инструкцию, следующую за инструкцией **CALL**.

Синтаксис вызова:

```
call function-name [( parameters-declaration )]
```

где

function-name
— это имя одной из функций, объявленных в текущем объекте.

parameters-declaration
функция может ожидать параметры, как указано в ее объявлении. Значения этих параметров приведены здесь. Затем они соответствующим образом используются при выполнении функции. Подробнее см. в разделе 2.5.3 .

Пример:

```
Action : START_RUN (new_run_number)
            …
      call PREPARE_FOR_RUN (run_number = 5)
            …
```

2.4.3 Инструкция MOVE_TO и TERMINATE_ACTION

Эта инструкция используется в двух различных ситуациях:

- **как одна из инструкций ДЕЙСТВИЯ**

  Формат:

  ```
  move_to state-name
  ```

  Эта инструкция завершит выполнение текущего действия, и объект примет новое состояние ***state-name***.

  NB: более расширенное использование для экспертов см. в разделе 2.5.5 .

  NB: Старая версия этой инструкции
  (terminate_action / state = *state-name*)
  все еще может использоваться.

- **в выражении WHEN**
  Формат:

```
when (condition) move_to state-name
```

  когда условие ***condition*** становится истинным, объект переходит из текущего состояния в состояние ***state-name***. (Для этого не нужно изобретать специальное действие).

Таким образом, инструкция **MOVE_TO** (или её несколько устаревший аналог **TERMINATE_ACTION**) является основным способом перевода объектов в другие состояния.

Инструкция **MOVE_TO** (или **TERMINATE_ACTION**) является терминальной, т. е. завершающей текущее выполняемое действие (**action**). Это значит, что инструкции, следующие после **MOVE_TO** (если они есть) выполняться не будут, т. к. обработка действия завершается.

2.4.4 Инструкция IF

Оператор условно выполняет блок инструкций. Кроме того, он предоставляет механизм блокировки, предотвращающий дальнейшее выполнение действия, из которого он вызывается, до тех пор, пока все объекты, на которые ссылается **IF**, не достигнут стабильного состояния.

Синтаксис:

```
IF (condition) THEN

   instructions

ELSE IF (condition) THEN

   instructions

ELSE

   instructions

END IF
```

где

condition – условия, как описано в разделе 2.5.1 ,

instructions - это одна или несколько инструкций **SML**.

Пример:

```
if ( TAPE1 in_state END_OF_TAPE or ALARM in_state ON)
      and ( RUN not_in_state DORMANT)  then
   do STOP RUN
end if
```

Предложения **ELSE IF** и **ELSE** являются необязательными. Вы можете иметь сколько угодно предложений **ELSE IF** в инструкции **IF**, но ни одно предложение **ELSE IF** не может стоять после предложения **ELSE**.

Инструкции **IF** могут быть вложены друг в друга. Это означает, что инструкции могут содержать другие инструкции **IF**.

Грубо говоря, инструкция ведет себя обычным образом, как и в большинстве языков программирования, а именно:

Когда встречается инструкция **IF**, проверяется условие, следующее за предложением **IF**. Если условие **TRUE**, выполняются инструкции, следующие за **THEN**. Если условие **FALSE**, каждый **ELSE IF** (если есть) обрабатывается один за другим в том порядке, в котором они указаны. Когда условие **TRUE** найдено, выполняются инструкции, следующие непосредственно за предложением **THEN** связанного с ним **ELSE IF**. Если ни одно из условий не оценивается как **TRUE** или если нет предложений **ELSE IF**, выполняются инструкции, следующие за **ELSE**. После выполнения инструкций, следующих за **THEN** или **ELSE**, выполнение продолжается с инструкции, следующей за **END IF**.

В **SMI++** это сложнее из-за вероятной ситуации невозможности в данный момент оценить условия. Это связано с тем, что некоторые объекты, от которых зависят условия, либо выполняют действие, либо ожидают выполнения действий для некоторых из этих объектов (в **SMI** это промежуточное состояние неопределенности или ожидания обозначается именем **&Busy** - «занят»). Дополнительная сложность возникает также из-за того, что во время выполнения инструкции **IF** могло измениться содержимое некоторых наборов объектов, от которых зависят условия.

Обработка инструкции IF

Сначала несколько определений:

- **REF-OBJECT** - объект, на который ссылается данное условие, т. е. от которого зависит условие. Ссылка может быть как прямой, так и косвенной через членство в наборе объектов.
- **REF-OBJECT-SET** – набор объектов, на который ссылаются в данном условии.
- **LOCKED-OBJECT** - бездействующий (заблокированный) объект, который не может выполнять действия.
- **FROZEN-OBJECT-SET** – набор объектов, который был «заморожен». Чтобы сделать объект замороженным, нужно сделать копию его содержимого в данном экземпляре. Впоследствии, даже если реальное содержание изменится, используется копия.

Когда **State Manager** (далее **SM**) начинает обработку **IF**, он пытается заблокировать **REF-OBJECT** во всех состояниях **IF**. Блокировка возможна только в том случае, когда «запираемый» объект бездействует и нет действий, ожидающих выполнения этого объекта. Если блокировка невозможна, выдается запрос блокировки для этого объекта. (Этот запрос на блокировку в конечном итоге заблокирует объект после выполнения ожидающих действий).

Теперь есть две возможности:

1) Есть некоторые **REF-OBJECT**, которые сейчас ждут блокировки. В этом случае все **REF-OBJECT-SET** замораживаются, а обработка **IF** приостанавливается и, следовательно, приостанавливается выполнение действия, содержащего **IF**. **State Manager** продолжит заниматься своими другими делами. (В конце концов он вернется, чтобы продолжить выполнение приостановленного **IF** и его родительского действия, когда получит сигнал о том, что все **REF-OBJECT IF** были заблокированы).
2) Все **REF-OBJECT** заблокированы.

В этом случае условия оцениваются одно за другим. Когда условие **TRUE** не найдено, инструкция **IF** завершается, и **SM** продолжает выполнение следующей инструкции после **END IF**. (Если блок не заканчивается инструкцией **MOVE_TO**). Если условие **TRUE** найдено, то **REF-OBJECT-SET** размораживаются и выполняется блок инструкций, следующий за соответствующим **ELSE IF**. Это может привести к следующим возможным результатам:

1) Выполнение блока приостанавливается из-за приостановки одной из его инструкций.
   В этом случае обработка **IF** приостанавливается, как и выполнение действия, содержащего **IF**. **State Manager** продолжит заниматься другими своими делами. В конечном итоге он вернется, чтобы продолжить выполнение приостановленного **IF** и его родительского действия, когда получит сигнал о том, что инструкция, вызвавшая приостановку, готова к продолжению.
2) Выполнение блока завершено.
   В этом случае блокировки со всех заблокированных объектов снимаются, и **SM** продолжает выполнение инструкции после **END IF**. (Если только блок не заканчивается инструкцией **MOVE_TO**).

NB: Это поведение можно изменить, используя опцию **Unlocked IFs** (см. опции **smiSM** в Приложении 4). Если установлено значение 1, **REF-OBJECT** разблокируются после оценки условий.

2.4.5 Инструкция WAIT

Эта инструкция будет ждать, пока указанные объекты перейдут в стабильное состояние, а затем завершится.

Синтаксис:

wait (*elem*$_1$,*elem*$_2$,…,*elem*$_n$)

где элемент *elem*$_i$ - одно из следующего:

- ***object-name*** - имя объекта,

или

- **all_in *set-name*** – элементы из набора объектов.

NB: более расширенное использование для экспертов см. в разделе 2.5.5 «Переменные элементы в SML».

Пример:

wait (*A*, all_in *B*, *C*)

где ***A*** и ***C*** – имена объектов, а ***B*** – имя набора объектов.

Описание обработки WAIT

Далее мы рассматриваем объект, находящийся в любом из двух типов состояний:

1. **available** (доступный) - объект не занят (т. е. не выполняет действие), и в его внутренней очереди действий нет никаких действий.
2. **not-available** (недоступный) – объект либо занят, либо в его внутренней очереди действий стоят действия.

Инструкция ожидания является «синхронизирующей» инструкцией. Он принимает в качестве параметров список объектов и/или список наборов объектов и будет вести себя следующим образом:

- если доступны все перечисленные объекты, а также объекты, включенные в перечисленные наборы объектов (см. выше), оно завершится, и выполнение текущего действия продолжится с инструкцией, следующей за инструкцией ожидания.
  или
- если какой-либо из перечисленных объектов или любой из объектов, включенных в перечисленные наборы объектов, недоступен (см. выше), он будет
  - «блокировать» все доступные объекты. Если объект «заблокирован», он не начнет выполнение каких-либо действий.
  - для недоступных объектов он поставит в очередь «действие блокировки» в их внутренних очередях действий. Это действие в конечном итоге будет выполнено и будет иметь эффект «блокировки» его объекта.
  - приостановить его (ожидание) выполнение и, следовательно, выполнение действия, выполняющего команду ожидания.

Выполнение инструкции ожидания возобновляется, как только все объекты будут заблокированы. В этот момент все объекты снова освобождаются («разблокируются»), и инструкция ожидания завершается.

2.4.6 Инструкция WHEN

Выражение **WHEN** позволяет объекту реагировать на незапрошенные изменения состояния других объектов. Такие изменения могут происходить в результате изменения внешних условий, передаваемые через прокси драйверы, связывающие **SM** с внешним миром.

Есть два вида **WHEN**. Первый тип – это те, которые объявляются после объявления состояния (иногда называемые «**state WHEN**»), а второй тип – это те, которые указаны в инструкции **WAIT_FOR** (иногда называемые **WF WHEN**, см. раздел 2.4.12). Оба этих типа позволяют объекту реагировать на незапрошенные изменения состояния других объектов, и даже имеют одинаковый синтаксис. Однако между ними есть некоторые различия.

В этом разделе мы будем говорить только о «**state WHEN**», но будем называть их просто **WHEN**.

Выражения **WHEN** объявляется (появляются) сразу после объявления состояния **State**. **WHEN** относится только к состоянию, в котором оно объявлено. Грубо говоря, пока объект находится в этом состоянии и одно или несколько условий в объявленных **WHEN** становятся истинными, объект инициирует выполнение ответа, указанного в первом **WHEN**, чье условие истинно. См. ниже для получения более подробной информации об обращении.

Синтаксис:

```
when (condition) response
```

где

```
condition (условие) – это
    некоторое условное выражение, см. раздел  2.5.1
response (отклик, ответ) – это
   либо
      do action-name
   либо
      move_to state-name
   либо
      stay_in_state
```

NB:

- ***action-name*** – это имя действия, которое должно быть выполнено, когда условие (***condition***) имеет значение **TRUE**. Он должен быть объявлен где-то в том же состоянии, что и **WHEN**.
- ***state-name*** – это имя состояния, в которое перейдет объект, когда условие (***condition***) истинно.

Пример:

Давайте представим гипотетический сценарий: у нас есть объект с именем **RUN**, и одно из его состояний называется **RUNNING**. Когда объект **RUN** находится в этом состоянии, важно убедиться, что ни один из объектов в наборе объектов **SUBSYSTEMS** не переходит в состояние **ERROR**. Мы можем добиться этого, объявив предложение **WHEN** в состоянии **RUNNING** (см. ниже). Это приведет к тому, что как только любой из объектов в наборе объектов **SUBSYSTEMS** перейдет в состояние **ERROR**, будет запланировано выполнение действия **RECOVER**. Затем действие обеспечит правильное восстановление.

```
object: RUN
        …

   state: RUNNING

      when ( any_in SUBSYSTEMS in_state ERROR ) do RECOVER
         …
      action:  RECOVER
         …
```

Обработка последовательности WHEN

В каждом состоянии объекта **SMI** мы можем объявить ряд **WHEN** (так называемая последовательность **WHEN**). Таким образом, каждое **WHEN** идентифицируется своим объектом, своим состоянием и позицией в последовательности **WHEN**. Когда объект находится в одном из своих возможных состояний, **WHEN**, объявленные в этом состоянии, будут реагировать на события.

События (**Event**)

Что касается **WHEN**, события бывают двух видов:

1. после перехода объект в домене достигает одного из своих состояний или
2. содержимое набора объектов в домене изменяется.

В дальнейшем объект, вызвавший событие, будем называть **Oev**, а в случае набора объектов **OSev**.

NB: изменение значения параметра не является событием. Поэтому простое условие типа 4 следует рассматривать только как дополнительное условие, которое само по себе не вынудит своего **WHEN** реагировать на событие.

Клиент WHEN

WHEN называется клиентом **Oev** или **OSev**, когда его объект находится в состоянии, в котором было объявлено **WHEN**, и связанное с ним условие зависит от **Oev** или **Osev**. **WHEN** считается клиентом **Oev**, даже если эта зависимость является только косвенной через **OSev**.

Клиентский объект (Object)

Объект называется клиентским объектом, когда он находится в одном из своих состояний и присутствует клиент **WHEN**.

Каждый раз, когда возникает событие **Oev**, диспетчер состояний (**SM**) обрабатывает последовательность **WHEN** объекта **Oev**, который объявлен в состоянии, которого только что достиг **Oev**. В случае событий **Oev** или **OSev SM** затем берет клиентские объекты один за другим и для каждого выполняет последовательность **WHEN**, объявленную в ее текущем состоянии. Эти **WHEN** выполняются в том порядке, в котором они были объявлены. Первый **WHEN**, чье условие оценивается как **TRUE**, вызывает выполнение своего ответа и прекращение обработки последовательности **WHEN**.

NB: Эта процедура на практике более сложна, но для понимания того, как она работает, этого описания достаточно.

Обработка индивидуального WHEN

Индивидуальное **WHEN** обрабатывается как часть последовательности **WHEN** следующим образом:

1. если какой-либо из объектов, от состояния которых зависит условие **WHEN**, либо находится в переходном состоянии, либо имеет ожидающие действия, связанное условие объявляется невыполнимым, и поэтому обработка **WHEN** завершается. В этом случае **SM** продолжит обработку следующего **WHEN** в последовательности **WHEN**.
2. если связанное условие поддается оценке и имеет значение **FALSE**, обработка **WHEN** также завершается, и **SM** будет вести себя, как указано выше.
3. если связанное условие поддается оценке и имеет значение **TRUE**, выполняется связанный ответ.
   В этом случае **SM** также прерывает и прекращает обработку последовательности **WHEN**. Это означает, что даже если некоторые из **WHEN**, следующие в последовательности, имеют условия, которые оцениваются как **TRUE**, они не будут иметь никакого эффекта.

2.4.7 Инструкция INSERT, REMOVE и REMOVE_ALL

Используя эти инструкции, пользователь может динамически управлять содержимым наборов объектов. То есть пользователь может вставлять или удалять объекты из наборов объектов.

Синтаксис:

```
insert object-identifier in set-name
remove object-identifier from set-name
remove_all from set-name
```

где

object-identifier – это

- *object-name*

  или

- *$(parameter-name)*

  где *parameter-name* - имя параметра действия, содержащего инструкцию вставки/удаления.

Примеры:

```
insert HV-78 in POWER-SUPPLIES

   эта инструкция вставит объект с именем HV-78
   в набор объектов с именем POWER-SUPPLIES

remove HV-45 from POWER-SUPPLIES

   эта инструкция удалит объект с именем HV-45
   из набора объектов с именем POWER-SUPPLIES

remove_all from POWER-SUPPLIES

   сделает набор объектов с именем POWER-SUPPLIES пустым

action: CONFIGURE (…, HV-NAME, …)
   …
insert $(HV-NAME) in POWER-SUPPLIES

   эта инструкция вставит объект, имя которого берется
   из параметра действия HV-NAME, в набор объектов
   с именем POWER-SUPPLIES
```

2.4.8 Инструкция CREATE_OBJECT

Эта инструкция создает объект **SMI** указанного класса.
Синтаксис:

create_object *object-identifier* of_class *class-name*

где

object-identifier – это ***object-name*** (имя объекта)
или
$(parameter-name) – где ***parameter-name*** – это имя параметра действия, содержащего инструкцию **create_object**.

Пример:

Как и любая другая инструкция, эта инструкция будет включена в действие. Назовем это действие произвольно **CONFIGURE**. Так же назначим этому действию параметр и так же условно назовем его **NEW_POWER_SUPPLY**. Предположим также, что наш домен **SMI** содержит класс объектов с именем **POWER_SUPLY**. В один из объектов домена мы могли бы добавить действие **CONFIGURE** следующим образом:

```
action: CONFIGURE (NEW_POWER_SUPPLY)
   …
   create_object $(NEW_POWER_SUPPLY) of_class POWER_SUPPLY
   …
```

Действие **CONFIGURE**, скорее всего, вызванное из **GUI**, примет значение параметра **NEW_POWER_SUPPLY** (назовем его произвольно **HV109**) и будет создан новый объект с именем **HV109** вышеуказанного класса.

2.4.9 Инструкция DESTROY_OBJECT

Эта инструкция полностью уничтожает указанный объект **SMI**. Он предназначен для удаления объектов, ранее созданных инструкцией **CREATE_OBJECT**, когда это становится желательным.
Синтаксис:

destroy_object *object-identifier*

где

object-identifier – это ***object-name*** (имя объекта)
или
$(parameter-name) – где ***parameter-name*** – это имя параметра действия, содержащего инструкцию **destroy_object**.

2.4.10 Инструкция SET

Эта инструкция присваивает (устанавливает) значение параметру, объявленному в текущем объекте. Если с назначением что-то пойдет не так, инструкция будет проигнорирована, а в файле журнала будет напечатано предупреждающее сообщение.

Синтаксис:

либо

SET ***object-parameter-name*** **=** ***item1***

либо

SET ***object-parameter-name*** **=** ***item1*** **arithmetic-operator** ***item2***

где

object-parameter-name - имя одного из параметров, объявленных для текущего объекта.
item1, ***item2*** - являются **indiValues**, см. раздел 2.5.4
arithmetic-operator – это один из следующих операторов:

+	сложение	item1 + item2
-	вычитание	item1 - item2
*	умножение	item1 * item2
/	деление (частное)	item1 / item2
%	остаток деления	item1 / item2

Предостережения

- Для строк разрешен только оператор **+**, который подразумевает конкатенацию.
- Оператор **%** разрешен только для целых чисел.

Как правило, указанные выше операции выполняются только между элементами, представляющими значения одного типа. Если элементы не представляют значения одного и того же типа, этого можно добиться путем приведения одного из них или обоих типов. См. обсуждение ниже.

Приведение типа элемента и смешивание типов

Правая часть инструкции SET

Это идентично обработке простого условия типа 4, см. раздел 2.5.1 «**Обработка условий**». Когда на этапе выполнения (стадия **State Manager**) фактическое строковое значение операнда, которое должно быть приведено к целому числу, не может быть интерпретировано как целое, выводится предупреждающее сообщение. в файле журнала **State Manager**, а инструкция **SET** не выполняется.

Сравнение левой и правой сторон инструкции SET

Результат операции должен быть того же типа, что и левая сторона. Если это не так, то это может быть достигнуто путем приведения типа правой стороны инструкции **SET.** Например, в выражении

SET A = (string)(B + C)

Левая сторона не может быть приведена к типу!

Если пользователь использовал приведение типа, то **SMI Translator** будет настаивать на том, чтобы результат правой стороны был таким же, как результат левой. Если пользователь не использовал приведение, то **SMI Translator** попытается сделать это следующим образом:

1. когда левая сторона представляет строку **string**, правая сторона преобразуется в строку **string** независимо от ее типа,
2. когда левая часть представляет собой целое число **int**, правая часть приводится к целому числу **int** независимо от его типа, но см. примечание ниже,
3. когда левая часть представляет значение с плавающей запятой **float**, а правая часть – целое число **int**, правая часть преобразуется в число с плавающей запятой **float**,
4. когда левая сторона представляет значение **float** с плавающей запятой, а правая – строку **string**, это не разрешено и приведет к сбою на этапе транслятора.

Примечание.
В случае 2, когда на этапе выполнения (этап диспетчера состояний) фактическое строковое значение не может быть интерпретировано как целое число, тогда в файле журнала **State Manager** печатается предупреждающее сообщение, а инструкция **SET** не выполняется.

2.4.11 Инструкция SLEEP

Эта инструкция приведет к приостановке текущего действия. Действие впоследствии возобновится через указанный период времени в секундах.
Синтаксис:

```
sleep time-period
```

где период времени – целое число (секунд), это косвенное значение **IndiValue**, которое может быть простым целым числом или более сложным, как описано в разделе 2.5.4 «Косвенные значения - IndiValue».

Пример:

```
sleep 5

Эта инструкция приостановит текущее действие на 5 секунд.
```

2.4.12 Инструкция WAIT_FOR

Иногда необходимо дождаться возникновения конкретной ситуации, прежде чем продолжить выполнение. Обычно это решалось созданием дополнительного состояния, переходом в него и последующим ожиданием возникновения этой ситуации. См. пример ниже.

```
Action: X
   …
   move_to WAITING_FOR_SOMETHING
   ...
State: WAITING_FOR_SOMETHING
   when(condition) move_to S
   …
   when(condition) do Y
Action: Y
   …
```

К сожалению, это приводит к увеличению количества дополнительных состояний и действий, причем не только в текущем объекте, но и в объектах, учитывающих текущее состояние объекта. Инструкция **WAIT_FOR** упрощает эту ситуацию.

Синтаксис:

```
WAIT_FOR

   when (condition₁) response₁

   …

   when (conditionₙ) responseₙ

END_WAIT_FOR
```

где

response$_i$ – это ответ на условие, являющееся ИСТИННЫМ, и является одним из следующих:

- **move_to *state-name***
  Прервет выполнение **WAIT_FOR** и поместит текущий объект в состояние ***state-name***
- **continue**
  Прервет выполнение **WAIT_FOR**, и выполнение продолжится с инструкции, следующей за **END_WAIT_FOR**.

В начале выполнения конструкции **WAIT_FOR** условия **WHEN** выполняются одно за другим в заданном порядке. Первый, дающий ответ **TRUE**, вызовет выполнение своего ответа и, следовательно, завершит конструкцию **WAIT_FOR**. Если ни одно из условий не истинно, то **WAIT_FOR**, текущее действие и текущий объект приостанавливаются.

Обработка State Manager

Когда во время выполнения действия достигается конструкция **WAIT_FOR** и ни одно из условий **WHEN** не имеет значения **TRUE**, выполнение конструкции и, следовательно, текущее действие приостанавливается аналогичным (но не идентичным) способом, что и инструкция **IF**. Позже, когда планировщик (*scheduler*) обнаруживает ситуацию, в которой одно из условий приостановленного **WHEN** может дать положительный ответ, он просит приостановленный объект оценить условия приостановленного **WHEN**, и если одно из них дает положительный ответ, он возобновляет выполнение приостановленного объекта, соответствующий ответ выполняется, и **WAIT_FOR** завершается.

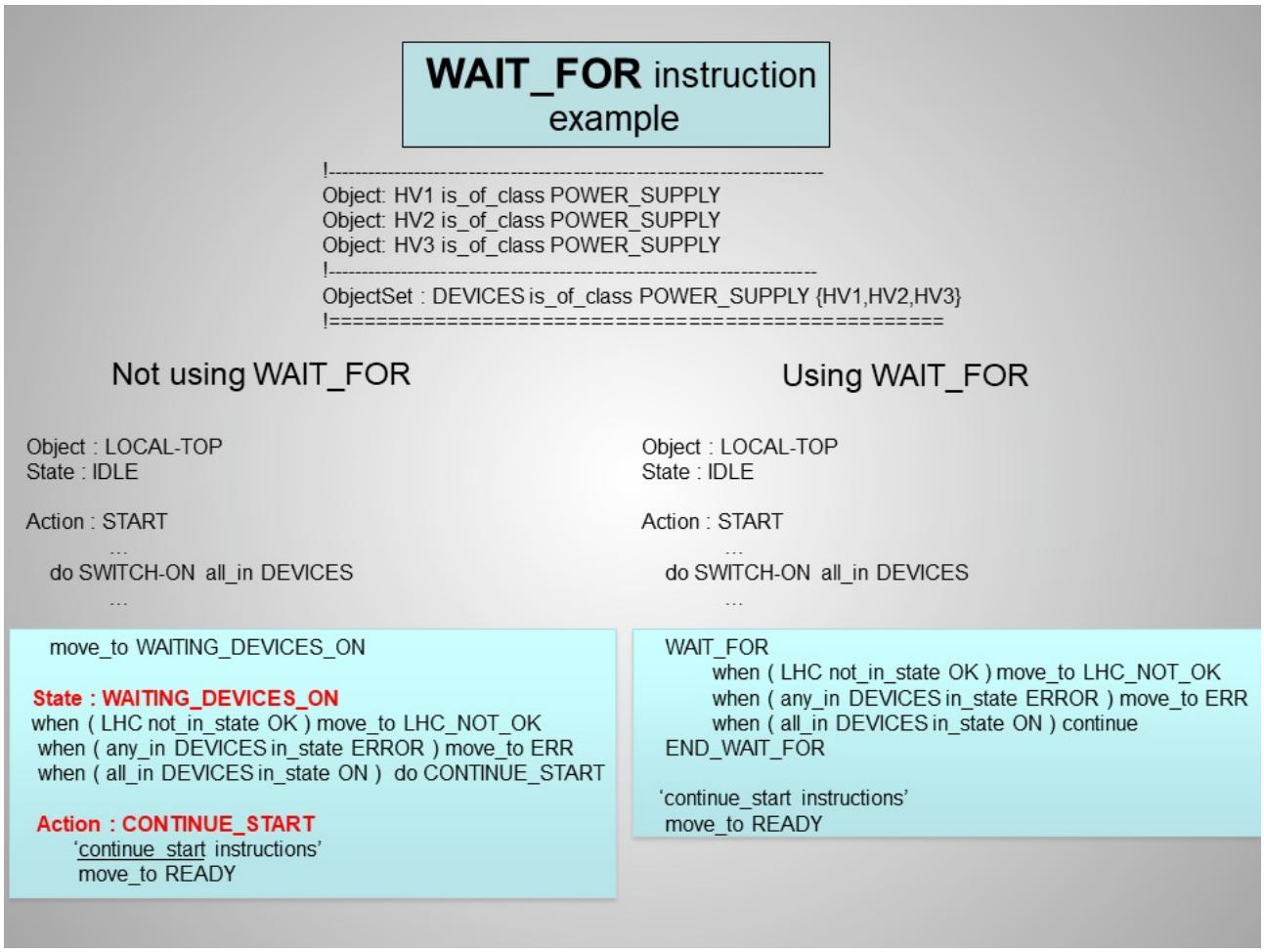


Рисунок 5. Пример использования **WAIT_FOR**. Обратите внимание на наличие дополнительного состояния **WAITING_DEVICES_ON** и действия **CONTINUE_START** в случае, когда **WAIT_FOR** не используется.

2.4.13 Инструкция REPORT

Эта инструкция позволяет пользователю при выполнении действия отправлять сообщения агенту, находящемуся за пределами мира **SMI**. Затем этот агент может делать то, что ему нравится, например отображать эти сообщения на какой-то панели **REPORT**. Различные участники процесса и их взаимодействие показаны на рис. 6 ниже.

Синтаксис:

```
REPORT( severity , msg )
```

где

severity (серьезность)
это уровень серьезности или строгости сообщения, один из следующих: **INFO**, **WARNING**, **ERROR**, **FATAL**

msg (сообщение)
имеет формат $\mathbf{indiVal_1+indiVal_2+…+indiVal_n}$
где $\mathbf{indiVal_i}$ – это ссылка на фактическое передаваемое значение. Подробно описано в разделе 2.5.4 «Косвенные значения - IndiValue».

Пример:

```
Object: O
  …
  Action: A(runno)
  …
  REPORT(INFO,"Object:"+_OBJECT_+"is reporting Run number:"+runno)
  …
```

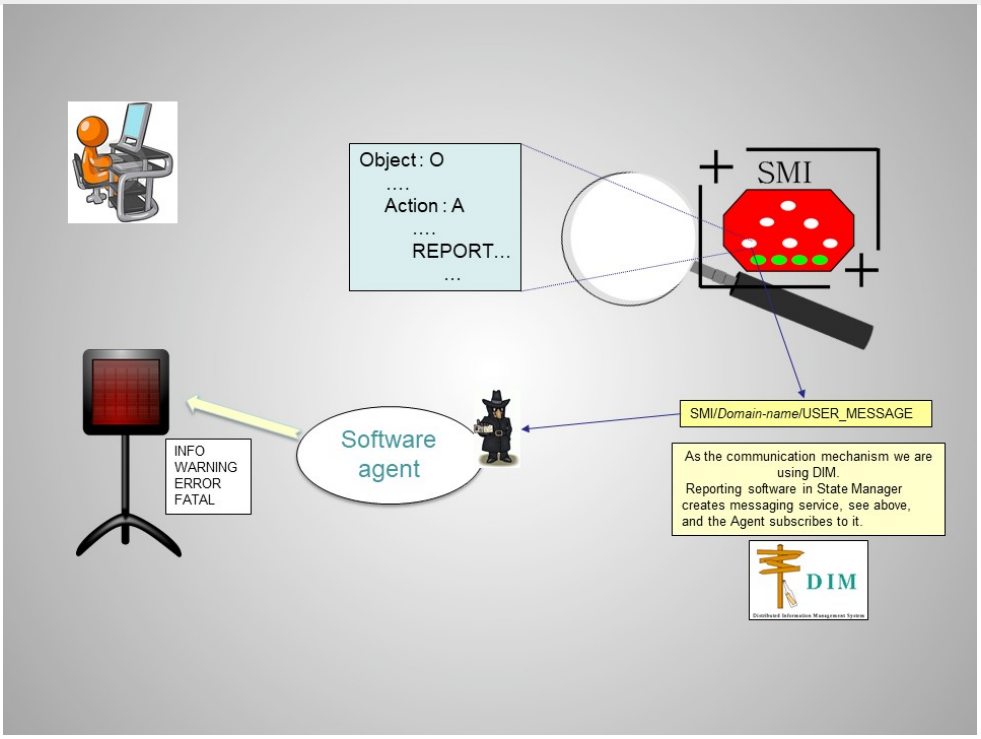


Рисунок 6. Различные участники процесса и их взаимодействие.

2.4.14 Инструкция FOR

ИНСТРУКЦИЯ В ПРОЦЕССЕ РАЗРАБОТКИ

Эта инструкция последовательно выполняет набор инструкций для каждого члена набора объектов.

Синтаксис:

```
for mem in objectSet

   For_Instruction_Block (FIB)

end_for
```

где

```
objectSet - имя набора объектов SMI
FIB - инструкции, которые должны быть выполнены
      для каждого члена objectSet
mem - это произвольное имя, представляющее объекты в наборе
      объектов. Единственным ограничением является то, что оно
      не должно конфликтовать ни с каким другим именем объекта
      в FIB.
```

Пример:

```
for  X  in  POWER_SUPPLIES
   do START X
   wait(...,X,...)
end_for
```

Блок инструкций FOR (FIB)

О **FIB** можно думать, как если бы это была функция с одним аргументом *mem*. Затем этот аргумент используется инструкциями внутри **FIB**. Ниже приведены инструкции **SMI**, которые поддерживаются в **FIB**. Для каждой инструкции описано ее использование в **FIB**.

Инструкция DO

Только объект, на который воздействуют, может быть заменен аргументом *mem*. Как и в приведенном выше примере.

Инструкция WAIT

Любой из объектов в скобках **WAIT** может быть заменен аргументом *mem*. Как и в приведенном выше примере. Продолжается работа по разрешению использования других инструкций в **FIB**.

2.5 Дополнительные правила и соглашения

2.5.1 Обработка условий

Условие в **SMI++** – это логическое выражение, состоящее из комбинации логических операторов {**and**, **or**, **not**}, скобок { **(** , **)** } и логических операторов.

В **SMI++** эти логические операторы называются простыми условиями (SC – *simple conditions*) и бывают следующих 4 типов:

type1	('*object-name*' **in_state** '*state-name*')
type2	(**all_in** '*object-name*' **in_state** '*state-name*') или (**any_in** '*objectset-name*' **in_state** '*state-name*')
type3	('*object-name*' **is_empty**)
type4	('*item1*' оператор отношения '*item2*') NB: этот тип используется для проверки значений параметров. Подробнее см. ниже.

NB: более продвинутое использование для экспертов '*object-name*' и '*objectset-name*' в типах 1, 2 и 3, см. раздел 2.5.5 «Переменные элементы в SML».

NB: в приведенном выше также может быть **not_in_state** вместо **in_state** и **not_empty** вместо **empty**

2.5.1.1 Вычисление условий

Условие транслируется с помощью SMI Translator следующим образом: извлекаются все простые условия (**SC**) и генерируется ряд элементарных логических операций или условных инструкций (**CI** – *condition instruction*), которые могут быть бинарными или унарными. Во время выполнения сначала вычисляются все **SC**, а затем вычисляется серия **CI** от начала до конца, при этом результатом последнего **CI** является общий желаемый ответ. Возможны 3 формы этих **CI**:

- *operand1* **and** *operand2*
- *operand1* **or** *operand2*
- **not** *operand1*

где *operand1*(*operand2*) является либо результатом оценки одного из **SC**, либо одного из предыдущих **CI**.

Пример:

```
( (A in_state ERROR) or (B in_state ERROR) ) and ( LHC in_state PHYSICS))
```

Извлекаются следующие простые условия:

```
SC1:(A in_state ERROR); SC2:(B in_state ERROR); SC3: (LHC in_state PHYSICS);
```

и ряд сгенерированных **CI**:

```
CI1 : результат SC1 or результат SC2

CI2: результат CI1 and результат SC3
```

В «нормальных» обстоятельствах результат оценки **SC**, а также **CI** либо ИСТИНА (**TRUE**), либо ЛОЖЬ (**FALSE**).

Сложность возникает, когда одно или несколько простых условий типа 2 относятся к пустому множеству. Мы приняли довольно нестандартный способ решения этой ситуации:

В дополнение к двум значениям **TRUE** и **FALSE** мы ввели еще одно значение, которое мы называем **GHOST** (призраком) и определяем его следующим образом:

Когда простое условие типа 2 относится к пустому множеству, его значение по определению является ***GHOST***.

Для вычисления **CI** мы приняли следующие правила:

- '*normal-value*' or GHOST = '*normal-value*'
- GHOST or '*normal-value*' = '*normal-value*'
- GHOST or GHOST = GHOST
- '*normal-value*' and GHOST = '*normal-value*'
- GHOST and '*normal-value*' = '*normal-value*'
- GHOST and GHOST = GHOST
- not GHOST = GHOST

В большинстве распространенных ситуаций это работает так, как если бы не было простых условий, относящихся к пустым множествам. Это, наверное, то, что хотелось бы сделать. Проблема возникает в то случаем, когда общий результат – **GHOST**. В этом случае мы заменяем этот результат на **FALSE**. В случае инструкции это именно то, что требуется. В случае инструкций **if** это не идеально, потому что это приведет к выполнению части **else if** (если таковая имеется) без какого-либо обоснования. Очевидно, что лучше было бы вообще игнорировать инструкцию **if**.

2.5.1.2 Простое условие Type4

Формат:

(*item1* оператор отношения *item2*)

где:

item1, *item2* - являются косвенными значениями (**indiValues** - *indirect values*), как описано в разделе 2.5.4 .

оператор отношения является одним из следующих:

- < меньше чем
- > больше, чем
- <= меньше или равно
- >= больше или равно
- == равно
- <> не равно

Как правило, сравнение выполняется только между элементами, представляющими значения одного типа. Если элементы не представляют значения одного и того же типа, то это может быть достигнуто путем приведения типов одного из них или обоих, см. ниже. См. также примеры из Приложения 5,6,7,8.

2.5.1.3 Приведение и смешение типов

Формат приведения типов:

```
(cast-indicator)item
```

где
item - как описано выше
cast-indicator является одним из следующих:
string, **int**, **float** (строка, целое число, число с плавающей запятой)
пример:

```
(int)RUN_TYPE
```

В этом примере RUN_TYPE — это имя параметра, который был объявлен как строка, и таким образом его значение временно преобразуется в целое число. И поэтому его можно сравнивать с другими целочисленными значениями.

2.5.1.4 Ограничения

Есть два основных ограничения, которые накладываются уже на этапе перевода:
1) **item**, представляет постоянное значение, например: **(float)3**
2) Преобразование **item**, представляющего **string**, в значение **float**
Кроме того, на этапе выполнения **SM**:
3) Преобразование элемента, представляющего строковое значение (**string**), в целое число (**int**), когда строка не может быть интерпретирована как таковая.

Когда используется приведение типов, **SMI Translator** настаивает на том, чтобы оба элемента (после приведения) были одного типа.

Когда приведение типов не используется и элементы представляют значения разных типов, **SMI Translator** попытается выполнить собственное приведение типов и в процессе выдаст предупреждения (и, возможно, ошибки):

- Если один из элементов представляет собой строковое значение, а другой представляет значение с плавающей запятой, то **SMI Translator** выдаст фатальную ошибку.
- Если один из элементов представляет целочисленное значение, а другой представляет значение с плавающей запятой, то элемент, представляющий целочисленное значение, преобразуется в число с плавающей запятой.
- Если один из элементов представляет целочисленное значение, а другой представляет строковое значение, то элемент, представляющий строковое значение, имеет тип, приведенный к целому числу. Позже, когда State Manager работает и фактическое строковое значение не может быть интерпретировано как таковое, условие оценивается как GHOST, и State Manager создает предупреждающее сообщение. В этом случае такое простое условие обрабатывается так же, как и условие второго типа, относящееся к пустому множеству. См. выше.

2.5.2 Элементы языка с точки зрения транслятора

SMI Translator (**smiTrans**), основная программа, которая обрабатывает и анализирует серверный код **SML**, распознает несколько различных типов элементов кода **SML**. Большинство этих элементов затем назначаются различным классам **C++**. На рисунке 7 ниже показана иерархическая структура наиболее важных классов:

Примечание. Иерархия связана с владельцем. Мы говорим, что класс является владельцем другого класса, если он либо содержит этот класс, либо хранит указатель на него.

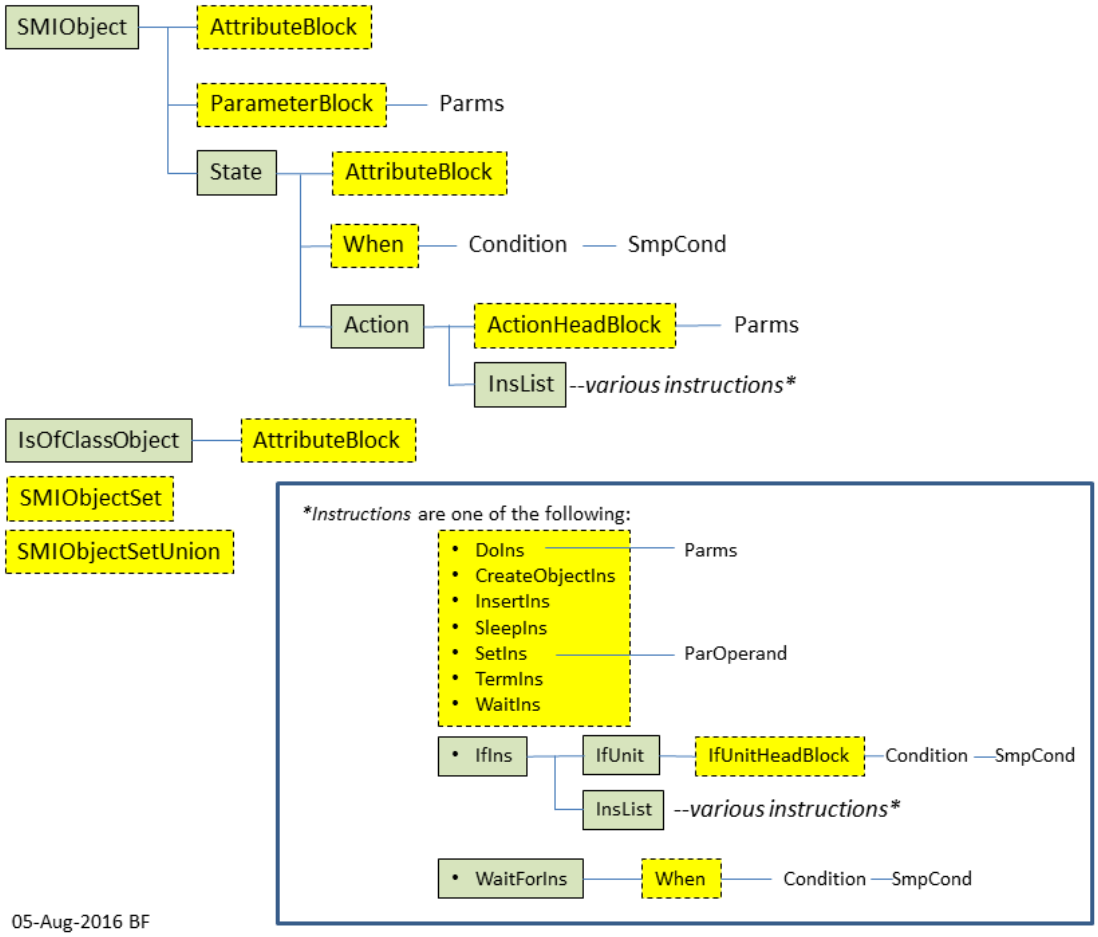


Рисунок 7. Иерархическая структура классов транслятора **SMI**.

Есть 3 разные группы этих классов:

1. Фрагмент кода **SML** (иногда называемый блоком или модулем), состоящий из нескольких полных строк. Первая строка модуля обычно начинается с ключевого слова (например, «**object:**» или «**when**»). Транслятор имеет различные стратегии распознавания последней строки блока. Все эти классы наследуются от класса **SMLUnit**. Внутри блока могут быть дополнительные блоки кода **SML** на более низком уровне. Поскольку все модули создаются в куче, это означает, что модуль должен хранить список указателей на модули, принадлежащие ему на более низком уровне. (Например, модуль «**State**» хранит указатель на свои модули «**When**» и «**Action**».) Эти указатели хранятся в локальных

данных класса **SMLUnit**, от которых наследуются модули. Условно эти единицы можно разделить на две подгруппы:

1. Элементы, которые относятся к другим элементам, таким как «**SMIObject**». На рисунке они обозначены зеленым цветом и обведены сплошной рамкой.
2. Элементы, не относящиеся к другим элементам. На рисунке они обозначены желтым цветом и обведены пунктирной рамкой. NB: блок WHEN является исключением, потому что, хотя он может ссылаться на действие, он не хранит указатель в обычном месте (т.е. локальные данные класса **SMLUnit**, от которого он наследуется).

2. Элементы кода **SML**, не принадлежащие к группе 1, такие как, например, «**Условие**». На рисунке они показаны обычным текстом.

2.5.3 Параметры в SMI

Вначале в **SMI** были только объекты, состояния и действия. По мере развития проекта **SMI** стало очевидно, что было бы полезно передавать значения друг другу. Примером может служить номер запуска, распространяемый мастер-объектом на все его подчиненные и т. д. Так родились параметры. В настоящее время можно встретить их в двух различных категориях. Места, где они объявлены, и места, где они используются.

2.5.3.1 Объявления параметров

Существует два типа параметров: параметры объекта, которые объявляются как часть объявления объекта, и параметры действия, которые объявляются как часть объявления действия.

Объявление объекта object:

Сразу после объявления объекта может следовать объявление параметров, определяющее параметры, связанные с этим объектом:

Object: *object-name*

Parameters: pd_1 [,pd_2, … ,pd_n]

где

pd_i является объявлением параметра в форме:

[type] name [= *default-value*]

где

- **type** - является определением типа и может быть либо **string**, либо **int**, либо **float**. Если нет, то предполагается, тип **string**.
- **name** - это имя параметра
- **default-value** - является значением параметра по умолчанию и должно быть константой соответствующего типа. Если тип **string**, значение должно быть заключено в двойные кавычки.

При объявлении параметры разделяются запятыми.

Объявление действия action:
Действие может иметь связанные с ним параметры, объявленные следующим образом:

Action: ***action-name*** **(** pd_1 **[,**pd_2**, … ,** pd_n**])**

где pd_i имеет тот же смысл, что и выше. Если значение по умолчанию не задано, тогда, когда действие вызывается из графического интерфейса, графический интерфейс предложит пользователю ввести значение. Когда действие получено как команда от другого объекта, то **State Manager** ожидает, что значение будет предоставлено.

При объявлении, содержащем много параметров, возможен перенос строки после запятой.

Пример объявления параметров:

```
object: RUN
    parameters: int NUMBER_T = 0,
                int NUMBER_P = 0,
                RUN_MODE = "DEMO"
    state: STOPPED
        action: START_RUN

object: EVT_BUILDER /associated
    parameters: int NUMBER_T, int NUMBER_P
    state: DEAD /dead_state                    !color: DarkGray
    state: READY                               !color: Aqua
        action: START(TYPE, int NR)
```

См. также примеры из Приложения 5,6,7,8.

2.5.3.2 Использование параметров

Инструкция CREATE_OBJECT

create_object $(*action-parameter-name*) of_class *class-name*

где ***action-parameter-name*** имя одного из параметров действия, объявленных в текущем действии.

Инструкции INSERT, REMOVE

insert $(*action-parameter-name*) in *set-name*

или

remove $(*action-parameter-name*) from *set-name*

где ***action-parameter-name*** имя одного из параметров действия, объявленных в текущем действии.

Инструкция DO

```
do action-name(pd1 [,pd2, ... ,pdn]) target-object-spec
```

где

```
pdi
   имеет формат pnmi = indiVali

где

   pnmi      имя одного из параметров, объявленных в действии
             action-name целевого объекта
   indiVali  является ссылкой на фактическое значение,
   которое должно быть передано. Подробно описано в  2.5.4 .
```

Инструкция CALL

```
CALL function-name(pd1[,pd2, ... ,pdn])
```

где
function-name имя одной из функций, объявленных в текущем объекте
pdi имеет тот же формат, что и инструкция **DO**, но pnm_i – это имя одного из параметров, объявленных в объявлении функции.

Инструкция REPORT

```
report (severity,msg)
```

где
severity - одно из: **INFO**, **WARNING**, **ERROR**, **FATAL**
msg - сообщение для посылки в форме:
$indiVal_1$+$indiVal_2$+...+$indiVal_n$

Инструкция SET

```
set object-parameter-name = indiVal1
```

или

```
set object-parameter-name = indiVal1 arithmetic-operator indiVal2
```

где

object-parameter-name - имя одного из объявленных параметров текущего объекта
indiVal1(2) - косвенные значения (**Indirect Values**). См. **2.5.4** .

Простые условия Simple Condition Type4

```
   (indiVal1 arithmetic-operator indiVal2)
```

где

indival1(2) – косвенные значения **Indirect Values**. См. **2.5.4** .

Предопределенные зарезервированные ключевые слова

Они используются для специальных значений параметров и операндов. На данный момент их всего 4:

```
   _DOMAIN_    _OBJECT_    _STATE_    _ACTION_
```

Примеры их использования:

```
Action: A(runno)
   …
   do B(run=runno, fromobj = _OBJECT_)
   …
   if ( (C in_state READY) and
      (_ACTION_ == "START")) then
      …
   end if
   …
```

Но они не допускаются в декларативных заявлениях:

```
      THIS IS NOT ALLOWED:
      Object: O
      Parameters: P1, …, Pi=_DOMAIN_,...)

   or

      Action: A(P1, …, Pi=_OBJECT_, … )
```

2.5.4 Косвенные значения - IndiValue

IndiValue - это сокращение от (*indirect value*) - «косвенное значение». Он представляет собой простое постоянное значение базового типа, то есть целое число (**int**), число с плавающей запятой (**float**) или строку (**string**). **IndiValue** может быть либо непосредственно таким значением, либо его форма дает указание, как это прямое (фактическое) значение получается.

Мы находим его использование в инструкциях:

Инструкция CALL:

```
   call function-name (…, PAR=indiValue, …)
```

```
Инструкция DO:
   do action-name (… , PAR=indiValue, …) object-name

Инструкция REPORT:
   report (WARNING, … + indiValue + … )

Инструкция SET:
   set PAR = indiValue1 + indiValue2

Простое условие Simple Condition Type4
   ( indiValue1 == indiValue2 )
```

Возможные формы **indiValue**:

INT
простая целочисленная константа, например **5**

FLOAT
простая константа с плавающей запятой, например, **5.1**

STRING
простая строка, например, **"ABC"**

NAME
или имя локального параметра
или зарезервированное имя: **_DOMAIN_** , **_OBJECT_** , **_STATE_** или **_ACTION_**

COMPNAME
Он имеет форму **obj-name.item**,

где:

obj-name - это имя любого объекта, объявленного в домене, а также
item – является
либо именем параметра (одного из параметров, объявленных в объекте **obj-name**)
либо зарезервированным именем **_STATE_** или **_ACTION_** - в этом случае текущее (фактическое) значение **indiValue** оценивается как текущее состояние или действие объекта **obj-name**.

2.5.5 Переменные элементы в SML (VarElement)

Первоначально в инструкциях **SML** использовались элементы (объекты, наборы объектов, состояния) с простыми, явно заданными именами. Затем это правило было расширено и разрешены имена элементов, передаваемые в значениях параметров действий. Это – переменные элементы (*variable element* или *VarElement*). Эта концепция **VarElement** введена для представления простого имени объектов, наборов или состояний. Возможные формы переменных элементов:

Простое имя

Пример: **RUN_CONTROL**

или **косвенная ссылка**

$(*parameter-name*)

где
parameter-name имя одного из параметров действия;
значение параметра принимается в качестве имени элемента.

Пример: **$(objNm)**

они используются в следующих инструкциях:

Инструкция **DO**
do *action-name* (*Parameters*) ***object-name***
или
do *action-name* (*Parameters*) **all_in** ***object-set-name***

Инструкция **MOVE_TO**
move_to ***state_name***

Инструкция **INSERT**, **REMOVE** и **REMOVE_ALL**
insert ***object-name*** **in** ***object-set-name***
remove ***object-name*** **from** ***object-set-name***
remove_all from ***object-set-name***

Инструкция **CREATE_OBJECT**
create_object ***object-name*** **of_class** ***class-name***

Инструкция **DESTROY_OBJECT**
destroy_object ***object-name***

Инструкция **WAIT**
wait (…,***object-name***,…,**all_in** ***object-set-name***,…)

Простые условия
Тип 1
(***object-name*** **in_state** ***state-list*** **)**
Тип 2
(any_in ***object-set-name*** **in_state** ***state-list*** **)**
или
(all_in ***object-set-name*** **in_state** ***state-list*** **)**
Тип 3
(***object-set-name*** **is_empty)**

NB: в инструкциях, описанных выше, можно использовать
not_in_state вместо **in_state** и **not_empty** вместо **is_empty**

IndiValue
В случае если **IndiValue** имеет форму **COMPNAME** (см. 2.5.4)
т.е.
***object-name*.item**
где **item** – зарезервированное имя или имя параметра объекта.

2.5.6 Формальный синтаксис SML

Синтаксис описан в форме **BNF** со следующим формализмом.

Терминальные символы – основные символы, из которых формируется программа: зарезервированные слова, операторы,… Они записаны в верхнем регистре. Следующие знаки препинания используются как разделители в языке **SML**: двоеточие ":", двойное двоеточие "::", запятая ",", скобки "**()**", знак равенства "**=**" и двойные кавычки """. Не-терминальные выражения – это синтаксические переменные, определяющие наборы строк. Они задают иерархические структуры языка, которые нужны для трансляции. Для них используется запись в виде слов в нижнем регистре между знаками "**<**" и "**>**".

Каждое производящее правило состоит из не-терминального символа, последующего "::=" с последующей строкой терминалов и не-терминалов. Синтаксис программы корректен, если он может быть представлен с помощью производящих правил, начиная с не-терминального символа: "<program>". Круглые скобки "{}*" представляют ноль или более пунктов, заключенных в скобках, а скобки "{}+" - по крайней мере один пункт. Квадратные скобки "[]" используются для необязательных терминалов и не-терминалов в производящих правилах.

Вертикальная линия "|" используется для выбора между разными терминальными и не-терминальными выражениями.

```
Definition of a SMI program
<program> ::= { <class def> | <object_def> }+

Definition of a class
<class def> ::= <class decl> { <logical object descr> | <associated class spec> }
<associated class spec> ::= { /ASSOCIATED <associated object descr> }

Definition of an object
<object def> ::= <internal object def> | <external object def>
<internal object def> ::= <internal object decl> { <logical object spec> | <associated object spec> }
<external object def> ::= <external object decl> | <associated object spec>
<logical object spec> ::= { <reference to a class> | <logical object descr> }
<associated object spec> ::= <reference to a class> | { /ASSOCIATED <associated object descr> }
<reference to a class> ::= IS_OF_CLASS <class identifier>
Description of an object
<logical object descr> ::= { <logical state def> }+
<associated object descr> ::= {<associated subobject def> }+ | {<associated state def> }+


Definition of a sub-object
<associated subobject def ::= <subobject decl> { <associated state def>}+


Definition of a state
<logical state def> ::= <state decl> [/INITIAL_STATE] { <when instructions> }* { <logical action def>}+
<associated state def> ::= <state decl> [/DEAD_STATE] { <action_decl> }*


Definition of an action
```

```
<logical action def ::= <action decl> { <standard instruction> }*



Definition of an instruction

<when instruction> ::= WHEN <condition> DO <action identifier>

<standard instruction> ::= <do instruction> | IF <condition> THEN { <standard instruction> }* [ ELSE { <standard
instruction> }* ] ENDIF

<do instruction> ::= DO <action identifier> [ ( {<parameter>}+ {, <parameter> }* ) ] <object reference>



Logical expression

<condition> ::= {<factor> { { AND <factor> } | { OR <factor> } }*

<factor> ::= [NOT] <term>

<term> ::= { <object reference> [ . <subobject identifier> ] { IN_STATE | NOT_IN_STATE } <state list> } |
{ ( <condition> ) }



Declarations

<class decl> ::= CLASS: <class identifier>

<internal object decl> ::= OBJECT: <object identifier>

<external object decl> ::= OBJECT: <domain identifier> :: <object identifier>

<subobject decl> ::= SUBOBJECT: <subobject identifier>

<state decl> ::= STATE: <state identifier>

<action decl> ::= ACTION: <action identifier> [ ( {<parameter decl>}+ {, <parameter decl> }* ) ]

<parameter decl> ::= <formal parameter identifier> [ = <string literal> ]



Syntactic elements

<state list> ::= <state identifier> | { <left curly bracket> <state identifier> { , <state_identifier> }* <right
curly bracket> }

<parameter> ::= <parameter identifier> = { <formal parameter identifier> | <stringliteral> }+



<object reference> ::= [<domain identifier> :: ] <object identifier>

<identifier> ::= <alpha> {[ _ ] | <alphanumeric> }*

<string literal ::= " {all characters except " }* "

<integer> ::= <digit> {<digit>}*

<alphanumeric> ::= <digit> | <alpha>

<digit> ::= 0|1|2|3|4|5|6|7|8|9

<alpha> ::= A| B| C| D| E| F| G| H| I| J| K| L| M| N| O| P| Q| R| S| T| U| V| W| X| Y| Z| a| b| c| d| e| f| g| h|
i| j| k| l| m| n| o| p| q| r| s| t| u| v| w| x| y| z

<left curly bracket> ::= {

<right curly bracket> ::= }

<comment> ::= ! {all characters }*
```

3 Инструменты (служебные программы) SMI++

В пакете программ **SMI** доступен набор инструментов и служебных программ для разработки, исполнения, внедрения и тестирования систем управления, как показано на рисунке 8.

Файл кода на языке **SML** служит единым источником кода для всех компонентов систему управления: для транслятора (*SML translator*) и системы выполнения сервера **SM** (*Logic Engine*); для реализации прокси драйверов; для клиентского кода графического интерфейса (**SMI GUI**).

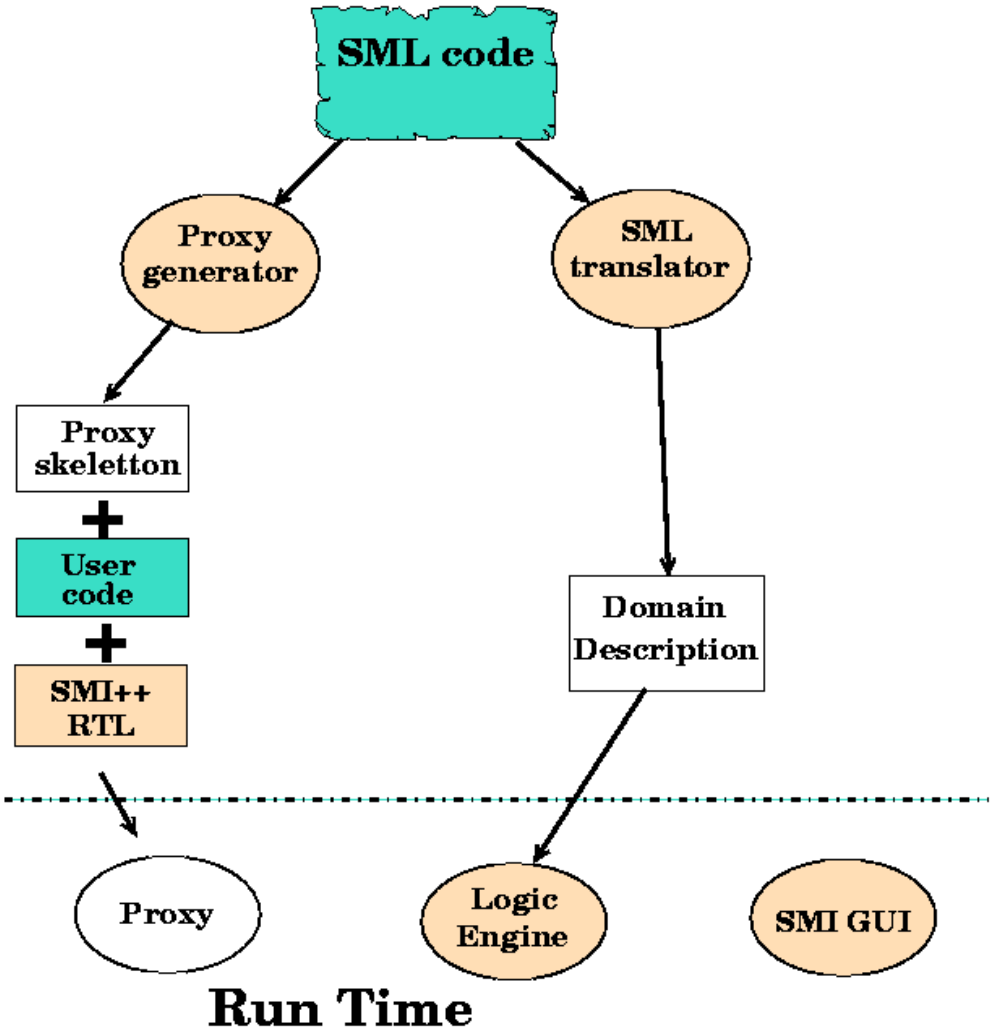


Рисунок 8. Набор инструментов **SMI**.

3.1 Генерация и исполнение Менеджера Состояний SM

Описание конкретного домена на языке **SML** обычно хранится в файле с расширением **.SML**. Каждый файл **.SML** соответствует домену. Транслятор (*SMI Translator*) должен быть запущен для этого файла, чтобы создать объектный файл **.SOBJ**, который затем будет использоваться во время выполнения общим механизмом **SM** (логический механизм на рисунке 8). Задействованы следующие инструменты.

3.1.1 Транслятор smiTrans

Транслятор **smiTrans** генерирует объектный файл для исполнения **SM**.

```
smiTrans.exe  file_name.SML
```

Программа **smiTrans.exe** принимает в качестве аргумента имя **SML** файла ***file_name.SML*** и создает объектный файл ***file_name.SOBJ***.

3.1.2 Сервер smiSM

Сервер **smiSM** выполняет код объектного файла **.SOBJ**, созданного транслятором **smiTrans**, с заданным именем домена.

```
smiSM.exe  domain_name  file_name.SOBJ
```

Программа **smiSM.exe** запускает процесс Менеджера Состояний **SM** с именем домена ***domain_name*** и исполняет код объектного файла ***file_name.SOBJ***. Например:

```
echo Транслировать и запустить
echo run_control.SML в домене  EGP

smiTrans.exe  run_control.SML
smiSM.exe  EGP  run_control.SOBJ
```

На поведение сервера **SM** могут влиять различные опции - параметры времени выполнения, передаваемые в аргументах командной строки. Подробнее см. в Приложении 4.

Сервер **smiSM** также использует переменные окружения, используемые пакетом **DIM**, например, **DIM_DNS_NODE**, см. документацию **DIM** [2,3].

3.2 Общий пользовательский интерфейс SMI - smiGUI

В набор инструментов входит общий (универсальный) настраиваемый пользовательский интерфейс для визуализации, тестирования и отладки определенного домена. Этот интерфейс позволяет изображать состояния объектов в нужном домене, а также отправлять им команды. Набор отображаемых объектов, их положение и цвет состояний настраиваются. Графический интерфейс запускается командой:

```
smiGUI
```

Программа **smiGUI** написана на языке **TCL** (исполняемый файл), поэтому для её работы нужна установка исполняемых библиотек **TCL**.

Работа **smiGUI** осуществляется через вызов служебных утилит, описанных в разделе 3.6 , с перехватом и анализом их консольного вывода. Проверка доступности **DNS** и нужного домена выполняется командами **dnsRunning** и **domainExists**. Список доменов получается вызовом **getDomains**, а список объектов в домене - вызовом **getDomainObjects**. Программы **monObjects** и **tellMonObjects** обеспечивают мониторирование состояния объектов и посылку им команд для изменения их состояния.

Пример работы **smiGUI** показан на рис. 9.

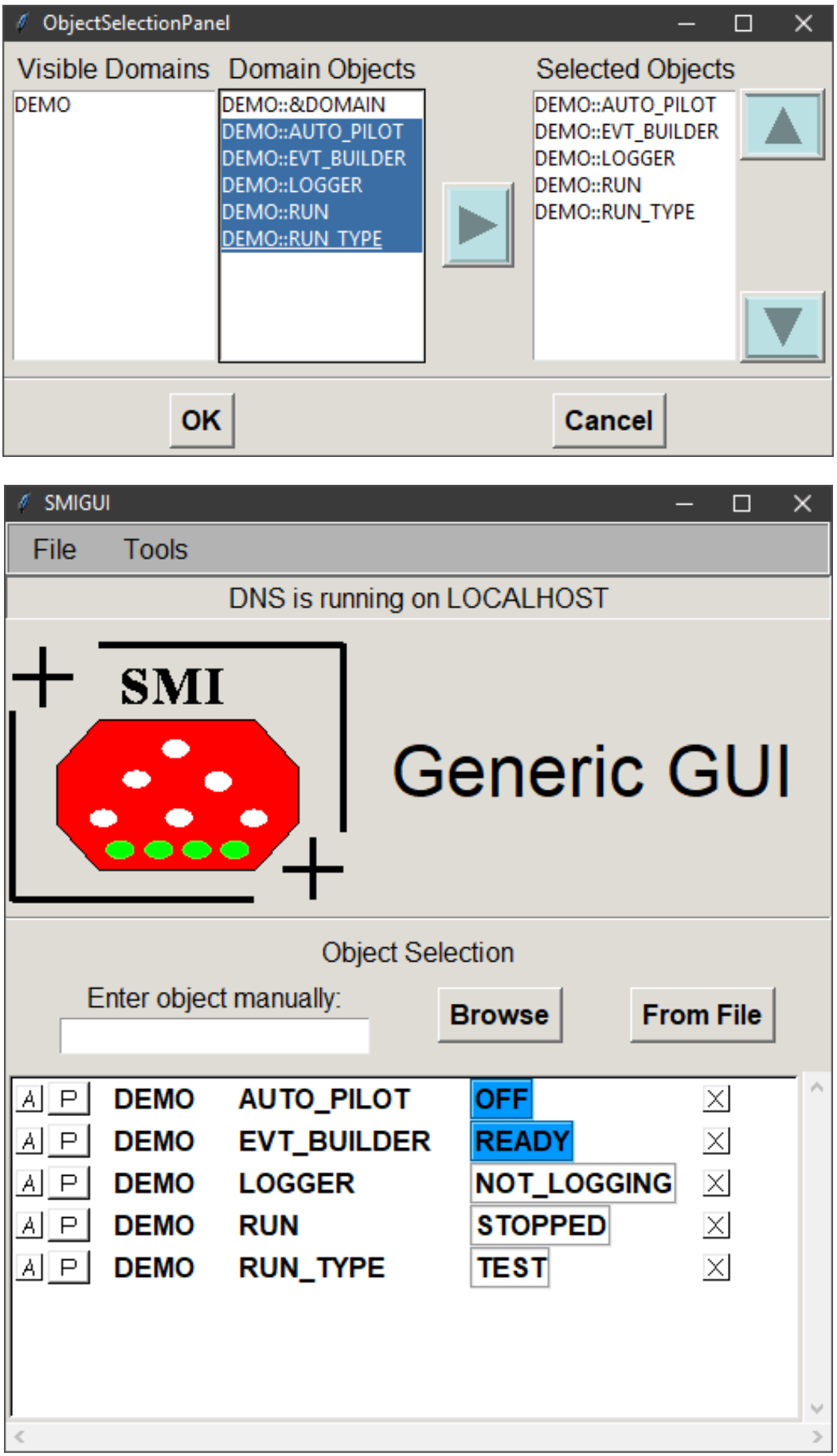


Рисунок 9. Общий вид программы **smiGUI**.

3.3 Программа smi_send_command

Эта команда позволяет пользователю отправлять команды любому объекту **SMI** в пространстве имен **DNS** (см. ниже).

Использование:

```
smi_send_command <objectName> <commandString> [-options] [<proxy>]
```

где

<*objectName*> - полное имя объекта (**DOMAIN::OBJECT**) объекта **SMI**, получающего команду

<*commandString*> - это, в простейшем случае, имя действия, которое должно быть отправлено объекту

<*proxy*> - может быть любым печатным символом (кроме знака минус) и указывает, что команда должна быть отправлена напрямую прокси (минуя любой менеджер состояний).

Доступны следующие опции команды **smi_send_command**:

Опция	Значение1	Значение2	Комментарий	Пример
-dns	<dnsNode>[:<dnsPort>]		позволяет изменить имя DNS узла и номер порта	-dns lxplus001.cern.ch:007
-pi	<parameterName>	<integerValue>	добавит целочисленный параметр к строке команды	-pi RUNNUMBER 269
-pf	<parameterName>	<floatingPointValue>	добавляет параметр с плавающей запятой в командную строку	-pf TEMP 35.67
-ps	<parameterName>	<stringConstant>	добавит строковый параметр к командной строке	-ps FILE '/bin/program.exe'
-dbg			командная строка вместо отправки объекту просто выводится на стандартный вывод. Полезно для отладки.	

Примечания к <stringConstant>:

- строка может содержать любой символ **ASCII** с исключением '**null**', поскольку большое количество специальных символов (таких как **>** или **?**) имеет особое семантическое значение для оболочки, строка должна быть заключена в пару одинарных кавычек, если только она не содержит только буквенно-цифровые символы, и в этом случае кавычки могут быть опущены.
- следующие символы требуют особой обработки, даже если они заключены в кавычки:
  - знак **!** должен предваряться обратной косой чертой; например '**123\!abc**', либо (еще лучше) представляться кодом **\x21**, например, '**123\x21abc**'.
  - двойная кавычка " должна предваряться обратной косой чертой; например '**123\"abc**', либо (еще лучше) представляться кодом **\x22**, например, '**123\x22abc**'.
  - знак кавычки ' должен быть представлен как **\x27**; например '**123\x27abc**'.
  - знак **** должен быть представлен как **** например '**123\\abc**'
- все непечатаемые символы (от **0** до **1F** (hex) и **7F**) должны быть «экранированы» с использованием символа возврата, за которым следует восьмеричный или шестнадцатеричный код **ASCII** символа. Например, **DLE** представлен либо **\020**, либо **\x10**. Рекомендуется использовать шестнадцатеричный код.

Пример:
Предположим, что в домене **EXPERIMENT** выполняется фиктивный **State Manager**. Далее предположим, что внутри домена есть объект **CONTROL**. И, наконец, предположим, что в своем текущем состоянии объект принимает действие **START_NEW_RUN** с двумя параметрами: **RUN_NUMBER** (целое число) и **RUN_TYPE** (строковая константа). Чтобы заставить этот объект выполнить действие, выполняется следующая команда:

```
smi_send_command EXPERIMENT::CONTROL START_NEW_RUN
        -pi RUN_NUMBER 1234 -ps RUN_TYPE 'COSMICS'
```

NB: Поскольку параметр **-dns** отсутствует в командной строке, команда предполагает, что сервер имен **DIM** работает на узле, заданном значением переменной среды **DIM_DNS_NODE**.

3.4 Программа smiGen

Программа **smiGen** генерирует из исходного файла **.SML** на языке **SML** шаблоны кода («скелеты» - *skeleton*) для реализации программ прокси драйверов для связанных (ассоциированных) объектов, см. рисунок 8.
Формат вызова:

```
smiGen [options] file

DESCRIPTION:
        smiGen Generates Proxy skelettons from the SML code ".SMI"
        and include files containing SMI object headers
        Possible Options:
                -p      Generate Proxies
                -c      Generate Proxies in "C"
                -C      Generate Proxies in "C++" (default)
                -m      Generate a makefile for the proxies
                -i      Generate include files for SMI objects
```

В качестве исходного файла передается файл **.SML** с описанием Конечных Автоматов на языке **SML**.
Шаблоны (скелеты) прокси программ генерируются на языках **C** или **C++** (в зависимости от опций). Имена генерируемых файлов для прокси связаны с названиями соответствующих связанных (ассоциированных) объектов. Программа **smiGen** генерирует свой файл кода для каждого ассоциированного объекта. Если в **.SML** файле находится несколько ассоциированных объектов, генерируется несколько файлов кода (с расширениями **.c** или **.cxx**).
Созданные программой **smiGen** шаблоны программ, хотя они и не являются еще готовыми программами управления, облегчают создание прокси драйверов, т. к. они содержат в себе уже готовые конструкции для реализации прокси, которые надо лишь заполнить содержательным кодом, реализующим логику работы прокси.
Пример (см. Приложение 5,9):

```
[c:\smi\bin]# smiGen.exe -p -c run_con.sml
Generating : logger_skel.c
Generating : evt_builder_skel.c
```

3.5 Клиентские библиотеки для разработчиков

Для разработки клиентских программ - как прокси драйверов для реализации связанных объектов **SMI**, так и программ графического интерфейса пользователя – в состав пакета **SMI** входят динамические библиотеки (**smirtl.dll** и **smiuirtl.dll**), а также заголовочные файлы и документация к ним.

3.5.1 Библиотека SMIRTL.DLL для создания SMI Proxy

Для создания прокси – объектов **SMI** (см. раздел 1.3) в наборе программных библиотек **SMI** предоставляется динамическая библиотека **smirtl.dll**. Она содержит все необходимые функции для организации взаимодействия между сервером **SM** и процессом драйвера, реализующего прокси (связанный объект **SMI**), и позволяет создавать прокси программы **SMI** практически на любом языке программирования, поддерживающем динамические библиотеки. К динамической библиотеке прилагаются заголовочные файлы и документация.

Заголовочный файл **smiuirtl.pas** для подключения библиотеки **smiuirtl.dll** на языке **Object Pascal** приведен в Приложении 2.

3.5.2 Библиотека SMIUIRTL.DLL для создания собственного пользовательского интерфейса

Общий пользовательский интерфейс **smiGUI**, предоставляемый в наборе инструментов, очень удобен для использования на этапе внедрения и тестирования системы управления, но не рекомендуется в качестве рабочего пользовательского интерфейса для системы управления. Обычно в окончательном пользовательском интерфейсе пользователь хотел бы объединить объекты из разных доменов, а также информацию о количествах, таких как **RUN_NUMBER** и **TRIGGER_RATE**, которые недоступны в виде состояний объекта **SMI**. Набор инструментов **SMI** содержит библиотеку, обеспечивающую легкий доступ к информации об объектах, состояниях и доступных или выполняемых в данный момент действиях. Чтобы получить доступ к этим функциям, ваш пользовательский интерфейс должен быть связан с библиотекой:

```
SMIUIRTL.DLL
```

Это библиотека времени выполнения пользовательского интерфейса **SMI**. Она также позволяет другим процессам (не пользовательским интерфейсам) отправлять команды объектам **SMI** и/или запрашивать состояние определенного объекта в системе.

Заголовочный файл **smiuirtl.pas** для подключения библиотеки **smiuirtl.dll** на языке **Object Pascal** приведен в Приложении 3.

3.6 Служебные утилиты SMI

В состав пакета **SMI** входит ряд служебных (консольных) утилит, которые помогают выполнять поддержку, настройку, обслуживание и анализ работы распределенных систем управления на базе **SMI**.

3.6.1 Утилита dnsRunning

Утилита **dnsRunning** проверяет, доступен ли сервер имен **DIM DNS**.
Например:

```
[C:\smi\bin]# set DIM_DNS_NODE=localhost
[C:\smi\bin]# dnsRunning.exe
YES
```

При работе утилиты используется переменная окружения **DIM_DNS_NODE**, поэтому перед вызовом её надо задать (если она еще не задана).

3.6.2 Утилита domainExists

Утилита **domainExists** проверяет, доступен ли (работает ли) домен **SMI** с заданным в аргументе именем.
Например:

```
[C:\smi\bin]# domainExists.exe DEMO
setenv SMI_DOMAIN_EXISTS yes

[C:\smi\bin]# domainExists.exe TEMP
setenv SMI_DOMAIN_EXISTS no
```

При работе утилиты используется переменная окружения **DIM_DNS_NODE**, поэтому перед вызовом её надо задать (если она еще не задана).

3.6.3 Утилита getDomains

Утилита **getDomains** выводит список доменов **SMI**, работающих в области имен **DIM DNS**. После имени домена и слеша «**/**» указывается имя сервера, который обслуживает этот домен.
Например:

```
[C:\smi\bin]# getDomains.exe
DEMO/ak-w10x32-vm
```

При работе утилиты используется переменная окружения **DIM_DNS_NODE**, поэтому перед вызовом её надо задать (если она еще не задана).

3.6.4 Утилита getDomainObjects

Утилита **getDomainObjects** выводит список объектов **SMI** в домене, имя которого задано в аргументе.
Например:

```
[C:\smi\bin]# getDomainObjects.exe DEMO
DEMO::AUTO_PILOT
DEMO::RUN_TYPE
DEMO::RUN
DEMO::LOGGER
DEMO::EVT_BUILDER
```

При работе утилиты используется переменная окружения **DIM_DNS_NODE**, поэтому перед вызовом её надо задать (если она еще не задана).

3.6.5 Утилита monObjectState

Программа **monObjectState** служит для отображения и мониторинга текущего состояния заданного объекта **SMI**.

```
[C:\smi\bin]# start monObjectState.exe DEMO::LOGGER

Консоль monObjectState:
DEMO::LOGGER in state LOGGING
DEMO::LOGGER Busy
DEMO::LOGGER in state NOT_LOGGING
DEMO::LOGGER Busy
DEMO::LOGGER in state LOGGING
DEMO::LOGGER Busy
DEMO::LOGGER in state WRITING
DEMO::LOGGER Busy
DEMO::LOGGER in state LOGGING
```

Монитор **monObjectState** позволяет, например, делать обработку событий по изменению состояний конкретных объектов **SMI**. Однако монитор **monObjectState** позволяет наблюдать за изменением только одного (заданного в аргументе) объекта **SMI**. Для наблюдения за группами объектов служит монитор **monObjects**, используемый совместно с командой **tellMonObjects**.

При работе утилиты используется переменная окружения **DIM_DNS_NODE**, поэтому перед вызовом её надо задать (если она еще не задана).

3.6.6 Утилита monObjects

Программа **monObjects** осуществляет мониторинг состояний объектов **SMI** с выводом результатов в консоль. При запуске программы в аргументе указывается номер процесса **GUI**, с которым будет связан монитор (в принципе, это может быть любой уникальный номер). Монитор обычно запускается из программы **GUI**, например, из **smiGUI**, см. раздел 3.2 . Предполагается, что консольный вывод монитора перехватывается программой **GUI** для анализа и отображения. Регистрация объектов для мониторинга производится утилитой **tellMonObjects**. После регистрации объектов монитор начинает выводить в консоль записи о каждом обновлении с указанием обновляемого объекта, его состояния, доступных действий.

Например:

```
[C:\smi\bin]# start monObjects.exe 5776

[C:\smi\bin]# tellMonObjects.exe 5776 register/DEMO::LOGGER

Консоль monObjects:

Mar 05 09:42:17/*object/DEMO::LOGGER/*state/NOT_LOGGING/*end_state/*action/LOG/*end_action
Mar 05 09:44:02/*object/DEMO::LOGGER/*state/&Busy/*end_state
Mar 05 09:44:02/*object/DEMO::LOGGER/*state/LOGGING/*end_state/*action/NOLOG/*end_action/*action/X_OPEN_FILE/*end_action
…
```

При перехвате программа **GUI** или другая программа, запустившая монитор, может декодировать консольный вывод и отображать состояния объектов в графическом виде или выполнять другие нужные действия при изменении состояний объектов **SMI**.

При работе утилиты используется переменная окружения **DIM_DNS_NODE**, поэтому перед вызовом её надо задать (если она еще не задана).

3.6.7 Утилита tellMonObjects

Программа **tellMonObjects** служит для посылки команд программе монитора **monObjects** для регистрации объектов и их последующего отображения и мониторинга.

Например:

```
[C:\smi\bin]# tellMonObjects.exe 5776 register/DEMO::LOGGER
```

В этом примере утилита **tellMonObjects** посылает процессу монитора **monObjects** команду регистрации объекта **DEMO::LOGGER**. После регистрации процесс монитора начинает обновление данных для заданного объекта с выводом обновлений в консоль. Номер процесса **GUI** (5776) должен совпадать с номером, с которым был вызван **monObjects**. Список объектов для регистрации в программе монитора берется, например, из вывода команды **getDomainObjects**.

Программа **tellMonObjects** может (кроме регистрации) передавать и другие команды монитору **monObjects**.

Например:

```
[C:\smi\bin]# tellMonObjects.exe 5776 register/DEMO::LOGGER
[C:\smi\bin]# tellMonObjects.exe 5776 remove/DEMO::LOGGER
[C:\smi\bin]# tellMonObjects.exe 5776 command/DEMO::LOGGER/LOG
[C:\smi\bin]# tellMonObjects.exe 5776 exit/

register/     - зарегистрировать объект в списке мониторинга
remove/       - удалить указанный объект из списка мониторинга
command/      - послать объекту команду выполнения указанного действия
exit/         - выход из программы монитора
```

При работе утилиты используется переменная окружения **DIM_DNS_NODE**, поэтому перед вызовом её надо задать (если она еще не задана).

4 Общий подход к реализации SMI GUI

В этом разделе кратко описывается общий подход к организации графического интерфейса пользователя **GUI** на основе Менеджера Состояний **SM** и технологии **SMI**, который **может** использоваться при создании распределенных систем управления с использованием пакета **CRW-DAQ** [10,11] в качестве супервизора или **SCADA** системы.

4.1 Общая схема взаимодействия компонентов SMI

Общая схема взаимодействия компонентов **SMI/FSM** приведена на рисунке 10.

Файл ***.SML** служит единым источником кода **FSM** для серверной части (реализующей логику управления) и клиентской части (реализующей прокси драйверы и графический интерфейс пользователя **GUI**).

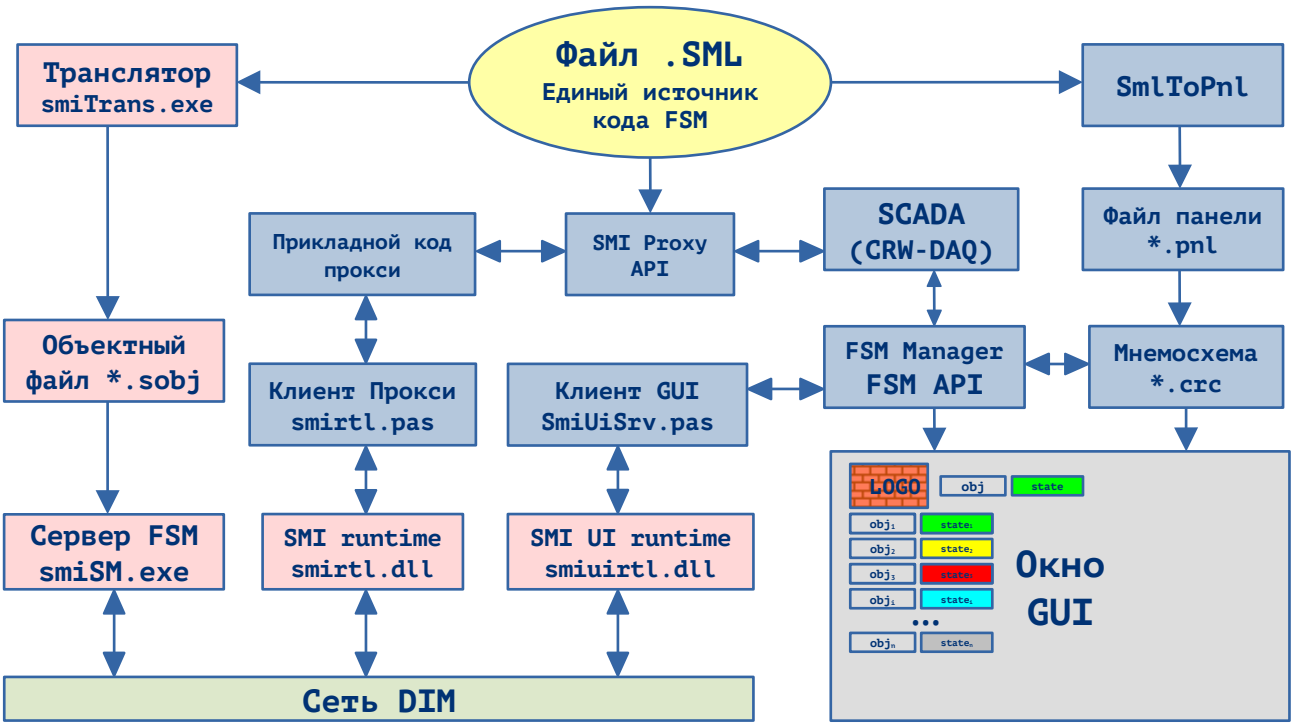


Рисунок 10. Общая схема взаимодействия компонентов **SMI/FSM**.

Серверная часть, отвечающая за логику, работает так, как описано в разделе 3.1 . Исходный код Конечного Автомата **.SML** с помощью **smiTrans** транслируется в объектный код **.SOBJ**, и затем исполняется сервером **smiSM**. Взаимодействие с другими компонентами системы идет через сеть **DIM**. Трансляция кода, запуск, остановка и контроль за работой сервера **smiSM** осуществляется с помощью супервизора **&SmiSrv**, работающего в пакете **CRW-DAQ** (на схеме не показан).

Клиентская часть, отвечающая за работу прокси драйверов, взаимодействует с сервером **smiSM** через сеть **DIM**, библиотеку **smirtl.dll** и клиентскую библиотеку (**smirtl.pas**). Со стороны пакета **CRW-DAQ** прикладной код, реализующий логику прокси, взаимодействует через библиотеку **SMI Proxy API**. Прикладной код прокси создается на языке **DaqPascal** в рамках пакета **CRW-DAQ**.

Клиентская часть, отвечающая за **GUI** (графический интерфейс пользователя), взаимодействует с сервером **smiSM** через сеть **DIM**, библиотеку **smiuirtl.dll** и клиентскую библиотеку (**smiuirtl.pas**). Со стороны пакета **CRW-DAQ** прикладной код, реализующий работу **GUI**, взаимодействует через библиотеку **FSM API**. Прикладной код **GUI** состоит из двух частей – программного кода на языке **DaqPascal**, реализующего взаимодействие с пользователем, и описания графического окна, которое генерируется в виде файла панели (**.pnl**) из файла **.SML** помощью утилиты **SmlToPnl**. Файл панели (**.pnl**) является частью мнемосхемы (файла **.crc**) графического окна, представляющего Конечный Автомат **FSM** на уровне пользователя.

Файл **.SML** служит единым источником кода для сервера **smiSM** (выполняющую логику управления), клиентского кода прокси драйверов (обеспечивающих взаимодействие с аппаратурой) и клиентского кода **GUI** (обеспечивающего взаимодействие с пользователем). Этот факт гарантирует (при определенной дисциплине работы) внутреннюю согласованность и непротиворечивость системы, т. к. при изменении кода **SML** файла соответствующие изменения в серверной и клиентской части делаются автоматически.

4.2 Общий принцип построения GUI

Обобщенный пользовательский интерфейс **GUI** на базе Конечных Автоматов **SMI/FSM** показан на рисунке 11.

Объект управления, как и его подсистемы, при этом подходе моделируется логическими объектами **SMI/FSM**. Поэтому в данном контексте объект управления отождествляется с Конечным Автоматом **FSM**. Графический интерфейс фактически отображает состояние Конечных Автоматов, с учетом их иерархии.

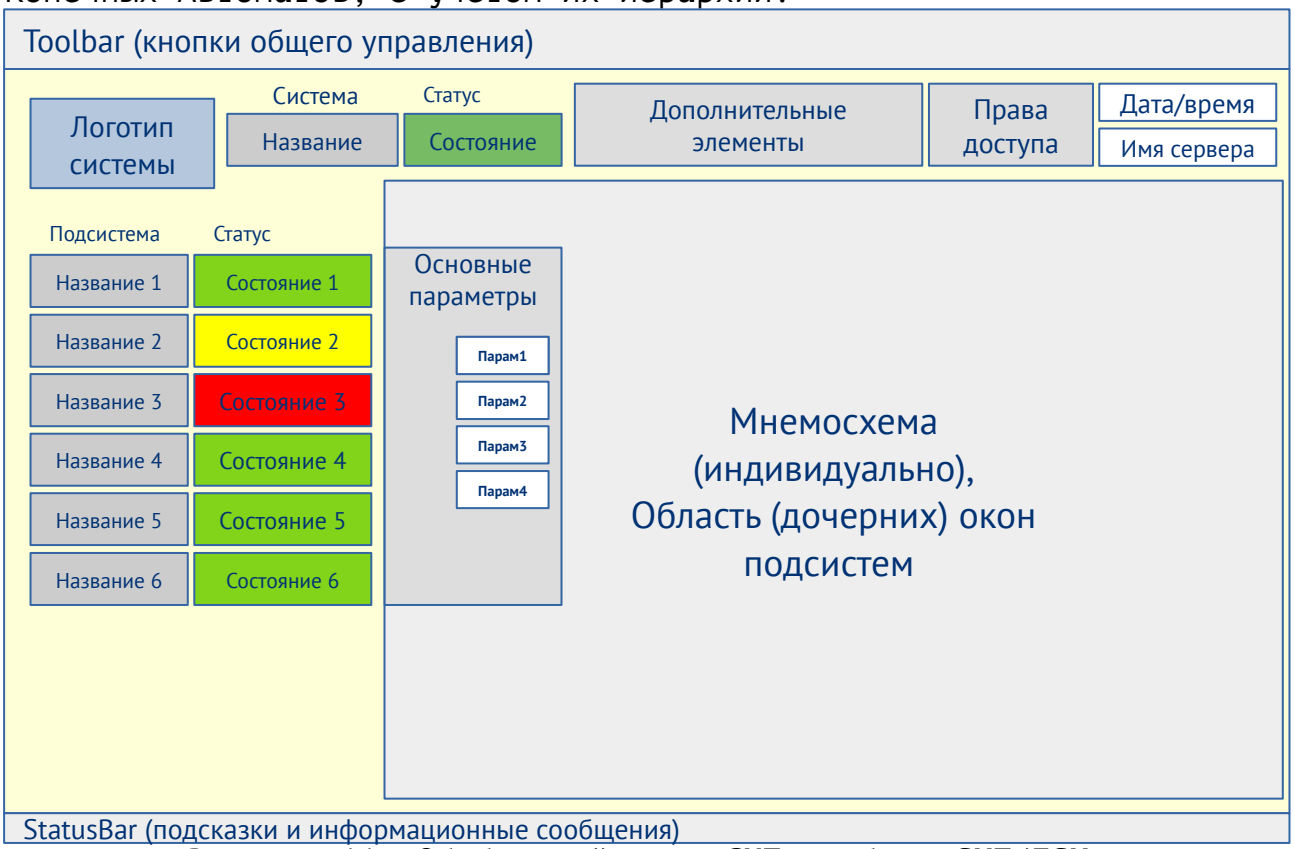


Рисунок 11. Обобщенный вид **GUI** на базе **SMI/FSM**.

В верхней части экрана находится **Toolbar** с кнопками общего управления (например, кнопки завершения работы или вызова диалога параметров настройки).

В нижней части экрана находится **StatusBar** - область подсказок и справочной информации.

В верхнем левом углу находится **Логотип Системы** – условное изображение управляемой физической установки или её подсистемы. При нажатии на логотип должно появляться окно справки, описывающей эту физическую установку.

Рядом, слева от логотипа, находится пара полей – **Система** и её **Состояние**. Поле **Система** содержит название управляемой системы, а поле **Состояние** – отображает её общее состояние в виде надписи на цветном поле (надпись и цвет зависят от состояния системы). При нажатии на кнопку **Состояние** прямо под ней должно появляться меню выбора **Действий**, доступных в этом состоянии, примерно так, как показано на рисунке 12.

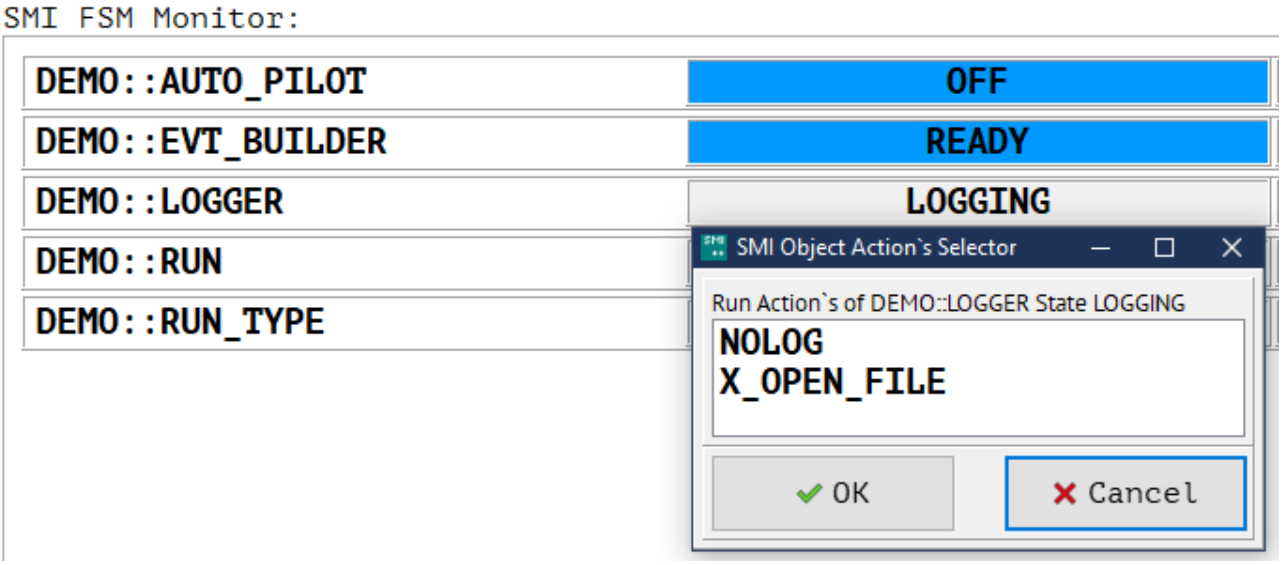


Рисунок 12. Пример меню выбора Действий (**NOLOG**, **X_OPEN_FILE**) при нажатии на Состояние **LOGGING** объекта **DEMO::LOGGER**.

При выборе нужного действия в меню и подтверждения (**OK**) должно быть выполнено действие, указанное в соответствующем пункте меню.

Слева от кнопки состояния системы могут располагаться: дополнительные элементы управления, поля для отображения или задания Прав Доступа, поля Даты/Времени и Имени сервера.

Под логотипом, в левой части экрана, располагаются кнопки подсистем; каждая подсистема представлена в виде пары кнопок **Подсистема** и **Состояние**. Кнопка **Подсистема** отображает имя подсистемы, а при нажатии – открывает дочернее окно подсистемы в области дочерних окон (в центре и нижней правой части экрана). Кнопка **Состояние** отображает (надписью и цветом) текущее состояние логического объекта **SMI/FSM**, представляющего эту подсистему. При нажатии на кнопку состояния должно появляться меню действий, доступных в этом состоянии, аналогичное показанному на рисунке 12.

Дочерние окна подсистем устроены аналогично, при этом каждая подсистема рассматривается как объект **SMI/FSM**, который может содержать свои дочерние подсистемы.

Область мнемосхемы и дочерних окон (в центре и нижней правой части экрана) в целом не регламентируется. Она может содержать изображение установки или управляемой подсистемы в виде мнемосхемы или таблицы параметров - выбор за прикладным программистом.

Описанный принцип построения **GUI** не является строго обязательным, интерфейс можно строить и по-другому. Однако описанный способ организации интерфейса достаточно разумен, удобен и прошел проверку практикой, поэтому его можно рассматривать как рекомендованный для широкого использования разработчиками.

Заключение

Технология Менеджера Состояний **SMI++**, базирующаяся на сетевой технологии **DIM**, хорошо приспособлена для создания распределенных измерительных систем и систем управления на базе Конечных Автоматов в смешанной среде, состоящей из разных компьютеров, работающих на разных аппаратных и программных платформах.

Хорошая переносимость и поддержка большого числа операционных систем и языков программирования позволяет легко увязывать интересы участников и организовывать работу больших экспериментов или установок, в которых разные группы разработчиков могут программировать на разных языках и использовать разные аппаратные и программные платформы.

Открытый исходный код и свободная лицензия (**GPL**) позволяет использовать **SMI** и **DIM** в самых разных задачах, не накладывая серьезных ограничений на её использование.

Наличие готовых универсальных утилит для наблюдения и отладки позволяет сделать разработку распределенных систем управления на основе **SMI** и **DIM** удобной и высокоэффективной.

Технологии **SMI++** и **DIM** хорошо апробированы и обкатаны в ряде крупных экспериментов, проводящихся в **CERN**, включая эксперименты **ALICE** [12,13,14,15,16], **ATLAS** [17,18], **LHCb** [19], **NA62** [20], **CMS** [21], **L3** [22], **CLOUD** [23].

Перечисленные достоинства позволяют автору рекомендовать **SMI++** как одну из лучших технологий для решения задач автоматизации измерительных систем и систем управления для физических установок на базе Конечных Автоматов.

Список использованных источников

1. http://cern.ch
2. http://dim.web.cern.ch/dim/
3. C.Gaspar, M.Dönszelmann, Ph.Charpentier. DIM, a portable, light weight package for information publishing, data transfer and inter-process communication. Computer Physics Communications, Volume 140, Issues 1-2, 15 October 2001, Pages 102-109.
4. https://smi.web.cern.ch/ - SMI++ home web site.
5. B. Franek and C. Gaspar, SMI++ – Object Oriented Framework for Designing and Implementing Distributed Control Systems, IEEE Trans. Nucl. Sci. 51 (2004) 513.
6. C. Gaspar and B. Franek, Tools for the automation of large distributed control systems, IEEE Trans. Nucl. Sci. 53 (2006) 974.
7. SMI++ Object-Oriented Framework for Designing and Implementing Distributed Control Systems (pdf) Presented at: IEEE Rome 2004 and published in IEEE TRANSACTIONS ON NUCLEAR SCIENCE, VOL 52, NO. 4, AUGUST 2005.
8. SMI++ - Object Oriented Framework for Designing Control Systems for HEP Experiments (pdf) Presented at: CHEP 97 - International Conference on Computing for High Energy Physics (Berlin, Germany, Apr 7-11 1997).
9. SMI++ Object Oriented framework for designing Distributed Control Systems (pdf) Presented at: Xth IEEE Real Time Conference 97 (Beaune, France, Sep 22-26 1997)
10. А.В.Курякин, Ю.И.Виноградов. Программа для автоматизации физических измерений и экспериментальных установок (CRW-DAQ). // Свидетельство РФ об официальной регистрации программы для ЭВМ № 2006612848 от 10.08.2006 г. Сайт программы: www.crw-daq.ru
11. А.В. Курякин, Ю.И. Виноградов. Программное обеспечение автоматизированных измерительных систем в области тритиевых технологий. // ВАНТ, серия «Термоядерный синтез», 2008 г., выпуск 2, стр.80-90.
12. The ALICE Collaboration (K. Aamodt et al). The ALICE experiment at the CERN LHC. 2008 JINST 3 S08002.
13. A.Kurepin, A.Augustinus et al. ALICE DCS preparation for RUN 3. Proceedings of the VIII International Conference "Distributed Computing and Grid-technologies in Science andEducation" (GRID 2018), Dubna, Moscow region, Russia, September 10 - 14, 2018, p.65-69.
14. ALICE Collaboration (S. Acharya et al). The ALICE Transition Radiation Detector: Construction, operation, and performance. Nuclear Inst. and Methods in Physics Research, A 881 (2018) 88-127.

15. А.Н.Курепин. Автоматизированная система управления и контроля стартового детектора времяпролетной системы эксперимента ALICE на Большом Адронном Коллайдере. Диссертация на соискание ученой степени кандидата физико-математических наук. Москва 2014.

16. А.В.Курякин, Ю.И.Виноградов, Н.В.Завьялов и др. Опыт длительной эксплуатации автоматизированной системы охлаждения электромагнитного калориметра PHOS в эксперименте ALICE. Ядерная физика и инжиниринг, 2012, том 3, № 6, с. 540-551.

17. A. Barriuso Poy et al. The detector control system of the ATLAS experiment. 2008 JINST 3 P05006.

18. K. Lantzsch et al. The ATLAS Detector Control System. International Conference on Computing in High Energy and Nuclear Physics 2012 (CHEP2012). Journal of Physics: Conference Series 396 (2012) 012028.

19. The LHCb Collaboration (D. Esperante, P. Rodríguez et al). LHCb silicon tracker DAQ and DCS online systems. 2009 16th IEEE-NPSS Real Time Conference - RT'09.

20. E. Cortina Gil et al. The beam and detector of the NA62 experiment at CERN. 2017 JINST 12 P05025.

21. William Badgett, Laura Borrello et al. Web Based Monitoring in the CMS Experiment at CERN.

22. Bert Petersen. The cosmic ray induced muon spectrum measured with the L3 detector. Radboud University Nijmegen 2002.

23. K. Weber et al. Data Acquisition System of the CLOUD Experiment at CERN. IEEE transactions on instrumentation and measurement,vol.70,2021.

24. B. Copy, E. Mandilara, I. Prieto Barreiro, F. Varela Rodriguez. Monitoring of CERN’s data interchange protocol (DIP) system. 16th Int. Conf. on Accelerator and Large Experimental Control Systems - ICALEPCS2017, Barcelona, Spain. doi:10.18429/JACoW-ICALEPCS2017-THPHA162, p.1797-1800.

25. M. Gonzalez-Berges. The Joint COntrols Project Framework. Computing in High Energy and Nuclear Physics, March 24-28 2003, La Jolla, California.

Приложение 1. Состав архива SMI.

```
smi\.setup
smi\History-beg-to-2019-05-17
smi\History.txt
smi\makefile
smi\makefile_common
smi\makefile_generator
smi\makefile_gui
smi\makefile_preprocessor
smi\makefile_rtl
smi\makefile_stateManager
smi\makefile_tclTkGUI
smi\makefile_translator
smi\makefile_utilities
smi\makeSlacLinks
smi\README
smi\ReleaseNotesv50r1.txt
smi\ReleaseNotesv51r1.txt
smi\ReleaseNotesv51r2.txt
smi\ReleaseNotesv51r3.txt
smi\ReleaseNotesv52r1.txt
smi\removeCVS
smi\slacMake
smi\taggingProcedure
smi\updateHistory
smi\bin\A.gif
smi\bin\dim.dll
smi\bin\dnsRunning.exe
smi\bin\domainExists.exe
smi\bin\downArrow.gif
smi\bin\getDomainObjects.exe
smi\bin\getDomains.exe
smi\bin\Microsoft.VC80.CRT.manifest
smi\bin\monObjects.exe
smi\bin\monObjectState.exe
smi\bin\msvcm80.dll
smi\bin\msvcp80.dll
smi\bin\msvcr80.dll
smi\bin\P.gif
smi\bin\rightArrow.gif
smi\bin\smiGen.exe
smi\bin\smiGUI.tcl
smi\bin\smirtl.bsc
smi\bin\smirtl.dll
smi\bin\smirtl.exp
smi\bin\smirtl.lib
smi\bin\smiSM.exe
smi\bin\smiTrans.exe
smi\bin\smiuirtl.bsc
smi\bin\smiuirtl.dll
smi\bin\smiuirtl.exp
smi\bin\smiuirtl.lib
smi\bin\SMIXXLogo.gif
smi\bin\smi_send_command.exe
smi\bin\tclTkGUI-Builder.exe
smi\bin\tellMonObjects.exe
smi\bin\upArrow.gif
smi\bin\X.gif
smi\smixx\parameters.h
smi\smixx\smirtl.h
smi\smixx\smirtl.hxx
smi\smixx\smirtl_core.hxx
smi\smixx\smiuirtl.h
smi\smixx\smiuirtl.hxx
smi\smixx\smiuirtl_core.hxx
smi\smixx\smixx_common.hxx
smi\smixx\smixx_parstring_util.h
smi\smixx\version.h
smi\src\commonSource\argname.cxx
smi\src\commonSource\argname.hxx
smi\src\commonSource\errorwarning.cxx
smi\src\commonSource\errorwarning.hxx
smi\src\commonSource\examination_stage.cxx
smi\src\commonSource\examination_stage.hxx
smi\src\commonSource\History
smi\src\commonSource\name.cxx
smi\src\commonSource\name.hxx
```

```
smi\src\commonSource\namelist.cxx
smi\src\commonSource\namelist.hxx
smi\src\commonSource\namevector.cxx
smi\src\commonSource\namevector.hxx
smi\src\commonSource\nmdptnr.cxx
smi\src\commonSource\nmdptnr.hxx
smi\src\commonSource\nmdptnrlist.cxx
smi\src\commonSource\nmdptnrlist.hxx
smi\src\commonSource\nmdptnrvector.cxx
smi\src\commonSource\nmdptnrvector.hxx
smi\src\commonSource\param.cxx
smi\src\commonSource\param.hxx
smi\src\commonSource\parammanager.cxx
smi\src\commonSource\parammanager.hxx
smi\src\commonSource\parmsbase.cxx
smi\src\commonSource\parmsbase.hxx
smi\src\commonSource\paroperand.cxx
smi\src\commonSource\paroperand.hxx
smi\src\commonSource\ptrvector.cxx
smi\src\commonSource\ptrvector.hxx
smi\src\commonSource\README
smi\src\commonSource\registrar.cxx
smi\src\commonSource\registrar.hxx
smi\src\commonSource\reservednames.cxx
smi\src\commonSource\reservednames.hxx
smi\src\commonSource\smixx_parstring_util.c
smi\src\commonSource\smlline.cxx
smi\src\commonSource\smlline.hxx
smi\src\commonSource\smllinevector.cxx
smi\src\commonSource\smllinevector.hxx
smi\src\commonSource\typedefs.hxx
smi\src\commonSource\utilities.cxx
smi\src\commonSource\utilities.hxx
smi\src\examples\evt_builder.vcproj
smi\src\examples\evt_builder.vcxproj
smi\src\examples\evt_builder_cpp.vcproj
smi\src\examples\evt_builder_cpp.vcxproj
smi\src\examples\examples.sln
smi\src\examples\examplesVS10.sln
smi\src\examples\examplesVS8.sln
smi\src\examples\logger.vcproj
smi\src\examples\logger.vcxproj
smi\src\examples\logger_cpp.vcproj
smi\src\examples\logger_cpp.vcxproj
smi\src\examples\monObjectState.cxx
smi\src\examples\testCui.c
smi\src\examples\testCui.vcproj
smi\src\examples\testCui.vcxproj
smi\src\examples\testCXXui.cxx
smi\src\examples\testCXXui.vcproj
smi\src\examples\testCXXui.vcxproj
smi\src\examples\bin\testCui.exe
smi\src\examples\bin\testCXXui.exe
smi\src\examples\run_control\.setup
smi\src\examples\run_control\config_MINI_RUN.dat
smi\src\examples\run_control\evt_builder.c
smi\src\examples\run_control\evt_builder.hxx
smi\src\examples\run_control\evt_builder_cpp.cxx
smi\src\examples\run_control\logger.c
smi\src\examples\run_control\logger.hxx
smi\src\examples\run_control\logger_cpp.cxx
smi\src\examples\run_control\makefile
smi\src\examples\run_control\run_con.sml
smi\src\examples\run_control\run_con.sobj
smi\src\examples\run_control\test_whens.sml
smi\src\examples\run_control\bin\evt_builder.exe
smi\src\examples\run_control\bin\evt_builder_cpp.exe
smi\src\examples\run_control\bin\logger.exe
smi\src\examples\run_control\bin\logger_cpp.exe
smi\src\generator\generator.c
smi\src\generator\README
smi\src\generator\smimake.c
smi\src\gui\dui_colors.h
smi\src\gui\dui_util.c
smi\src\gui\dui_util.h
smi\src\gui\smid.c
smi\src\gui\smid.h
smi\src\preprocessor\block_name.cxx
smi\src\preprocessor\block_name.hxx
smi\src\preprocessor\codeblock.cxx
smi\src\preprocessor\codeblock.hxx
```

```
smi\src\preprocessor\cpreproc.cxx
smi\src\preprocessor\generblock.cxx
smi\src\preprocessor\generblock.hxx
smi\src\preprocessor\History
smi\src\preprocessor\incfile.cxx
smi\src\preprocessor\incfile.hxx
smi\src\preprocessor\incline.cxx
smi\src\preprocessor\incline.hxx
smi\src\preprocessor\macro.cxx
smi\src\preprocessor\macro.hxx
smi\src\preprocessor\main.cxx
smi\src\preprocessor\README
smi\src\preprocessor\version.hxx
smi\src\rtl\History
smi\src\rtl\smirtl.c
smi\src\rtl\smirtlcpp.cxx
smi\src\rtl\smiuirtl.c
smi\src\rtl\smiuirtlcpp.cxx
smi\src\rtl\smi_change_option.c
smi\src\rtl\smi_kill.c
smi\src\rtl\smi_send_command.c
smi\src\stateManager\action.cxx
smi\src\stateManager\action.hxx
smi\src\stateManager\action_return_status.hxx
smi\src\stateManager\alarm.cxx
smi\src\stateManager\alarm.hxx
smi\src\stateManager\callins.cxx
smi\src\stateManager\callins.hxx
smi\src\stateManager\clientobject.cxx
smi\src\stateManager\clientobject.hxx
smi\src\stateManager\clientstate.cxx
smi\src\stateManager\clientstate.hxx
smi\src\stateManager\clientwhens.cxx
smi\src\stateManager\clientwhens.hxx
smi\src\stateManager\commhandler.cxx
smi\src\stateManager\commhandler.hxx
smi\src\stateManager\condition.cxx
smi\src\stateManager\condition.hxx
smi\src\stateManager\createobjectins.cxx
smi\src\stateManager\createobjectins.hxx
smi\src\stateManager\destroyobjectins.cxx
smi\src\stateManager\destroyobjectins.hxx
smi\src\stateManager\diag.cxx
smi\src\stateManager\diag.hxx
smi\src\stateManager\doins.cxx
smi\src\stateManager\doins.hxx
smi\src\stateManager\forins.cxx
smi\src\stateManager\forins.hxx
smi\src\stateManager\function.cxx
smi\src\stateManager\function.hxx
smi\src\stateManager\History
smi\src\stateManager\ifhandler.cxx
smi\src\stateManager\ifhandler.hxx
smi\src\stateManager\ifins.cxx
smi\src\stateManager\ifins.hxx
smi\src\stateManager\initiator.cxx
smi\src\stateManager\insertins.cxx
smi\src\stateManager\insertins.hxx
smi\src\stateManager\inslist.cxx
smi\src\stateManager\inslist.hxx
smi\src\stateManager\instruction.cxx
smi\src\stateManager\instruction.hxx
smi\src\stateManager\instruction_return_status.hxx
smi\src\stateManager\logic_engine.cxx
smi\src\stateManager\msg.cxx
smi\src\stateManager\msg.hxx
smi\src\stateManager\objectregistrar.cxx
smi\src\stateManager\objectregistrar.hxx
smi\src\stateManager\option.cxx
smi\src\stateManager\option.hxx
smi\src\stateManager\options.cxx
smi\src\stateManager\options.hxx
smi\src\stateManager\parameters.hxx
smi\src\stateManager\parms.cxx
smi\src\stateManager\parms.hxx
smi\src\stateManager\queue_name.cxx
smi\src\stateManager\queue_name.hxx
smi\src\stateManager\queue_twonames.cxx
smi\src\stateManager\queue_twonames.hxx
smi\src\stateManager\README
smi\src\stateManager\rename
```

```
smi\src\stateManager\report.cxx
smi\src\stateManager\report.hxx
smi\src\stateManager\reportins.cxx
smi\src\stateManager\reportins.hxx
smi\src\stateManager\reservednames_sm.cxx
smi\src\stateManager\resumehandler.cxx
smi\src\stateManager\resumehandler.hxx
smi\src\stateManager\scheduler.cxx
smi\src\stateManager\scheduler.hxx
smi\src\stateManager\setins.cxx
smi\src\stateManager\setins.hxx
smi\src\stateManager\set_name.cxx
smi\src\stateManager\set_name.hxx
smi\src\stateManager\sleepins.cxx
smi\src\stateManager\sleepins.hxx
smi\src\stateManager\sleepinstimer.cxx
smi\src\stateManager\sleepinstimer.hxx
smi\src\stateManager\smiclass.cxx
smi\src\stateManager\smiclass.hxx
smi\src\stateManager\smifrozenset.cxx
smi\src\stateManager\smifrozenset.hxx
smi\src\stateManager\smiobject.cxx
smi\src\stateManager\smiobject.hxx
smi\src\stateManager\smiobjectset.cxx
smi\src\stateManager\smiobjectset.hxx
smi\src\stateManager\smiobjectsetsimple.cxx
smi\src\stateManager\smiobjectsetsimple.hxx
smi\src\stateManager\smiobjectsetunion.cxx
smi\src\stateManager\smiobjectsetunion.hxx
smi\src\stateManager\smisetcontainer.cxx
smi\src\stateManager\smisetcontainer.hxx
smi\src\stateManager\smisetmember.cxx
smi\src\stateManager\smisetmember.hxx
smi\src\stateManager\smpcond.cxx
smi\src\stateManager\smpcond.hxx
smi\src\stateManager\smpcondtyp1.cxx
smi\src\stateManager\smpcondtyp1.hxx
smi\src\stateManager\smpcondtyp2.cxx
smi\src\stateManager\smpcondtyp2.hxx
smi\src\stateManager\smpcondtyp3.cxx
smi\src\stateManager\smpcondtyp3.hxx
smi\src\stateManager\smpcondtyp4.cxx
smi\src\stateManager\smpcondtyp4.hxx
smi\src\stateManager\state.cxx
smi\src\stateManager\state.hxx
smi\src\stateManager\state_manager.cxx
smi\src\stateManager\swstay_in_state.cxx
smi\src\stateManager\swstay_in_state.hxx
smi\src\stateManager\termins.cxx
smi\src\stateManager\termins.hxx
smi\src\stateManager\timehandling.cxx
smi\src\stateManager\timehandling.hxx
smi\src\stateManager\todo.txt
smi\src\stateManager\twonames.cxx
smi\src\stateManager\twonames.hxx
smi\src\stateManager\ut_sm.cxx
smi\src\stateManager\ut_sm.hxx
smi\src\stateManager\version.hxx
smi\src\stateManager\waitforins.cxx
smi\src\stateManager\waitforins.hxx
smi\src\stateManager\waitins.cxx
smi\src\stateManager\waitins.hxx
smi\src\stateManager\wfwcontinue.cxx
smi\src\stateManager\wfwcontinue.hxx
smi\src\stateManager\wfwmove_to.cxx
smi\src\stateManager\wfwmove_to.hxx
smi\src\stateManager\when.cxx
smi\src\stateManager\when.hxx
smi\src\stateManager\whenresponse.cxx
smi\src\stateManager\whenresponse.hxx
smi\src\tclTkGUI\A.gif
smi\src\tclTkGUI\actionParPanel_display.tcl
smi\src\tclTkGUI\assemble
smi\src\tclTkGUI\canvasObjects.tcl
smi\src\tclTkGUI\convertHEXtoEASCII.tcl
smi\src\tclTkGUI\dnsinfo.tcl
smi\src\tclTkGUI\downArrow.gif
smi\src\tclTkGUI\extract-section
smi\src\tclTkGUI\History
smi\src\tclTkGUI\listBox_create.tcl
smi\src\tclTkGUI\main.tcl
```

```
smi\src\tclTkGUI\mainPanel.tcl
smi\src\tclTkGUI\makeExecutable
smi\src\tclTkGUI\objectParPanel_create.tcl
smi\src\tclTkGUI\objectSelectionPanel.tcl
smi\src\tclTkGUI\objectSetDisplayPanel.tcl
smi\src\tclTkGUI\P.gif
smi\src\tclTkGUI\parameterGroups.tcl
smi\src\tclTkGUI\parameterHelp_show.tcl
smi\src\tclTkGUI\rightArrow.gif
smi\src\tclTkGUI\Scrolled_Listbox.tcl
smi\src\tclTkGUI\selectFromListPanel_show.tcl
smi\src\tclTkGUI\simpleEntry_show.tcl
smi\src\tclTkGUI\SMIXXLogo.gif
smi\src\tclTkGUI\tcl-smi.tcl
smi\src\tclTkGUI\testobjectParPanel.tcl
smi\src\tclTkGUI\testParPanel.tcl
smi\src\tclTkGUI\testSelectFromListPanel.tcl
smi\src\tclTkGUI\textPanel_show.tcl
smi\src\tclTkGUI\topbar.tcl
smi\src\tclTkGUI\upArrow.gif
smi\src\tclTkGUI\X.gif
smi\src\translator\action.cxx
smi\src\translator\action.hxx
smi\src\translator\actionheadblock.cxx
smi\src\translator\actionheadblock.hxx
smi\src\translator\attributeblock.cxx
smi\src\translator\attributeblock.hxx
smi\src\translator\attributes.cxx
smi\src\translator\attributes.hxx
smi\src\translator\boolitem.cxx
smi\src\translator\boolitem.hxx
smi\src\translator\booloperation.cxx
smi\src\translator\booloperation.hxx
smi\src\translator\callins.cxx
smi\src\translator\callins.hxx
smi\src\translator\condition.cxx
smi\src\translator\condition.hxx
smi\src\translator\createobjectins.cxx
smi\src\translator\createobjectins.hxx
smi\src\translator\cxxtranslatorversion.hxx
smi\src\translator\destroyobjectins.cxx
smi\src\translator\destroyobjectins.hxx
smi\src\translator\doins.cxx
smi\src\translator\doins.hxx
smi\src\translator\forins.cxx
smi\src\translator\forins.hxx
smi\src\translator\function.cxx
smi\src\translator\function.hxx
smi\src\translator\getline_test.cxx
smi\src\translator\History
smi\src\translator\ifins.cxx
smi\src\translator\ifins.hxx
smi\src\translator\ifunit.cxx
smi\src\translator\ifunit.hxx
smi\src\translator\ifunitheadblock.cxx
smi\src\translator\ifunitheadblock.hxx
smi\src\translator\insertins.cxx
smi\src\translator\insertins.hxx
smi\src\translator\inslist.cxx
smi\src\translator\inslist.hxx
smi\src\translator\instruction.cxx
smi\src\translator\instruction.hxx
smi\src\translator\isofclassobject.cxx
smi\src\translator\isofclassobject.hxx
smi\src\translator\main.cxx
smi\src\translator\objectregistrar.cxx
smi\src\translator\objectregistrar.hxx
smi\src\translator\opermanager.cxx
smi\src\translator\opermanager.hxx
smi\src\translator\parameterblock.cxx
smi\src\translator\parameterblock.hxx
smi\src\translator\parms.cxx
smi\src\translator\parms.hxx
smi\src\translator\processcommandline.cxx
smi\src\translator\reportins.cxx
smi\src\translator\reportins.hxx
smi\src\translator\setins.cxx
smi\src\translator\setins.hxx
smi\src\translator\sleepins.cxx
smi\src\translator\sleepins.hxx
smi\src\translator\smiobject.cxx
```

```
smi\src\translator\smiobject.hxx
smi\src\translator\smiobjectset.cxx
smi\src\translator\smiobjectset.hxx
smi\src\translator\smiobjectsetunion.cxx
smi\src\translator\smiobjectsetunion.hxx
smi\src\translator\smlsectiontype.hxx
smi\src\translator\smlunit.cxx
smi\src\translator\smlunit.hxx
smi\src\translator\smpcond.cxx
smi\src\translator\smpcond.hxx
smi\src\translator\smpcondtyp1.cxx
smi\src\translator\smpcondtyp1.hxx
smi\src\translator\smpcondtyp2.cxx
smi\src\translator\smpcondtyp2.hxx
smi\src\translator\smpcondtyp3.cxx
smi\src\translator\smpcondtyp3.hxx
smi\src\translator\smpcondtyp4.cxx
smi\src\translator\smpcondtyp4.hxx
smi\src\translator\stack.cxx
smi\src\translator\stack.hxx
smi\src\translator\stackitem.cxx
smi\src\translator\stackitem.hxx
smi\src\translator\state.cxx
smi\src\translator\state.hxx
smi\src\translator\swmove_to.cxx
smi\src\translator\swmove_to.hxx
smi\src\translator\swstay_in_state.cxx
smi\src\translator\swstay_in_state.hxx
smi\src\translator\termins.cxx
smi\src\translator\termins.hxx
smi\src\translator\todo.txt
smi\src\translator\ut_tr.cxx
smi\src\translator\ut_tr.hxx
smi\src\translator\waitforins.cxx
smi\src\translator\waitforins.hxx
smi\src\translator\waitins.cxx
smi\src\translator\waitins.hxx
smi\src\translator\wfwcontinue.cxx
smi\src\translator\wfwcontinue.hxx
smi\src\translator\wfwmove_to.cxx
smi\src\translator\wfwmove_to.hxx
smi\src\translator\when.cxx
smi\src\translator\when.hxx
smi\src\translator\whenresponse.hxx
smi\src\utilities\dnsdebugging.cxx
smi\src\utilities\dnsexists.cxx
smi\src\utilities\dnsRunning.cxx
smi\src\utilities\domainexists.cxx
smi\src\utilities\getDimVersions.cxx
smi\src\utilities\getDomainObjects.cxx
smi\src\utilities\getDomainObjectSets.cxx
smi\src\utilities\getDomains.cxx
smi\src\utilities\getObjectState.cxx
smi\src\utilities\getSetObjects.c
smi\src\utilities\getSmiVersions.cxx
smi\src\utilities\getTimeString.c
smi\src\utilities\History
smi\src\utilities\listdomain.cxx
smi\src\utilities\makeSkeleton
smi\src\utilities\monObjects.cxx
smi\src\utilities\monObjectsServiceName.cxx
smi\src\utilities\monObjectState.cxx
smi\src\utilities\proxyexists.cxx
smi\src\utilities\serverInError.cxx
smi\src\utilities\shellcmd.c
smi\src\utilities\shellcmd.skel
smi\src\utilities\tclTkGUI-Builder.cxx
smi\src\utilities\tellMonObjects.cxx
smi\src\utilities\waitObjectNotBusy.cxx
smi\src\utilities\efftcl\balloon.tcl
smi\src\utilities\efftcl\balloon1.tcl
smi\src\utilities\efftcl\confirm.tcl
smi\src\utilities\efftcl\dialog.tcl
smi\src\utilities\efftcl\my_dialog.tcl
smi\src\utilities\efftcl\notice.tcl
smi\src\utilities\efftcl\progressGauge.tcl
smi\src\utilities\efftcl\testProgressGauge.tcl
smi\src\utilities\efftcl\_orig_balloon.tcl
smi\src\utilities\efftcl\_orig_progressGauge.tcl
smi\Visual\cleanup.cmd
smi\Visual\domainExists.dsp
```

```
smi\Visual\generator.dsp
smi\Visual\generator.vcproj
smi\Visual\generator.vcxproj
smi\Visual\getdomainobjects.dsp
smi\Visual\getDomains.dsp
smi\Visual\History
smi\Visual\monObjects.dsp
smi\Visual\preprocessor.dsp
smi\Visual\run_smixxV8.cmd
smi\Visual\smirtl.dsp
smi\Visual\smirtl.vcproj
smi\Visual\smirtl.vcxproj
smi\Visual\smiuirtl.dsp
smi\Visual\smiuirtl.vcproj
smi\Visual\smiuirtl.vcxproj
smi\Visual\smixx.dsw
smi\Visual\smixx.sln
smi\Visual\smixxVS10.sln
smi\Visual\smixxVS13.sln
smi\Visual\smixxVS15.sln
smi\Visual\smixxVS8.sln
smi\Visual\smi_send_command.dsp
smi\Visual\smi_send_command.vcproj
smi\Visual\smi_send_command.vcxproj
smi\Visual\state_manager.dsp
smi\Visual\state_manager.vcproj
smi\Visual\state_manager.vcxproj
smi\Visual\tellMonObjects.dsp
smi\Visual\translator.dsp
smi\Visual\translator.vcproj
smi\Visual\translator.vcxproj
smi\Visual\UpdateSmiGUI.cmd
smi\Visual\UtilDnsRunning.dsp
smi\Visual\UtilDnsRunning.vcproj
smi\Visual\UtilDnsRunning.vcxproj
smi\Visual\UtilDomainExists.dsp
smi\Visual\UtilDomainExists.vcproj
smi\Visual\UtilDomainExists.vcxproj
smi\Visual\UtilGetDomainObjects.dsp
smi\Visual\UtilGetDomainObjects.vcproj
smi\Visual\UtilGetDomainObjects.vcxproj
smi\Visual\UtilGetDomains.dsp
smi\Visual\UtilGetDomains.vcproj
smi\Visual\UtilGetDomains.vcxproj
smi\Visual\UtilMonObjects.dsp
smi\Visual\UtilMonObjects.vcproj
smi\Visual\UtilMonObjects.vcxproj
smi\Visual\UtilMonObjectState.vcproj
smi\Visual\UtilMonObjectState.vcxproj
smi\Visual\UtilTclTkGUIBuilder.dsp
smi\Visual\UtilTclTkGUIBuilder.vcproj
smi\Visual\UtilTclTkGUIBuilder.vcxproj
smi\Visual\UtilTellMonObjects.dsp
smi\Visual\UtilTellMonObjects.vcproj
smi\Visual\UtilTellMonObjects.vcxproj
smi\Visual\resource\dnsRunning.rc
smi\Visual\resource\dnsRunning_rc.h
smi\Visual\resource\dnsRunning_rc.ico
smi\Visual\resource\domainExists.rc
smi\Visual\resource\domainExists_rc.h
smi\Visual\resource\domainExists_rc.ico
smi\Visual\resource\getDomainObjects.rc
smi\Visual\resource\getDomainObjects_rc.h
smi\Visual\resource\getDomainObjects_rc.ico
smi\Visual\resource\getDomains.rc
smi\Visual\resource\getDomains_rc.h
smi\Visual\resource\getDomains_rc.ico
smi\Visual\resource\monObjects.rc
smi\Visual\resource\monObjectState.rc
smi\Visual\resource\monObjectState_rc.h
smi\Visual\resource\monObjectState_rc.ico
smi\Visual\resource\monObjects_rc.h
smi\Visual\resource\monObjects_rc.ico
smi\Visual\resource\smiGen.rc
smi\Visual\resource\smiGen_rc.h
smi\Visual\resource\smiGen_rc.ico
smi\Visual\resource\smirtl.rc
smi\Visual\resource\smirtl_rc.h
smi\Visual\resource\smirtl_rc.ico
smi\Visual\resource\smiSM.rc
smi\Visual\resource\smiSM_rc.h
```

```
smi\Visual\resource\smiSM_rc.ico
smi\Visual\resource\smiTrans.rc
smi\Visual\resource\smiTrans_rc.h
smi\Visual\resource\smiTrans_rc.ico
smi\Visual\resource\smiuirtl.rc
smi\Visual\resource\smiuirtl_rc.h
smi\Visual\resource\smiuirtl_rc.ico
smi\Visual\resource\smixx-000.bmp
smi\Visual\resource\smixx-001.bmp
smi\Visual\resource\smixx-002.bmp
smi\Visual\resource\smixx-100.bmp
smi\Visual\resource\smixx-101.bmp
smi\Visual\resource\smixx-102.bmp
smi\Visual\resource\smixx.ico
smi\Visual\resource\smixx0.ico
smi\Visual\resource\smixx1.ico
smi\Visual\resource\smixxcopy.cmd
smi\Visual\resource\smi_send_command.rc
smi\Visual\resource\smi_send_command_rc.h
smi\Visual\resource\smi_send_command_rc.ico
smi\Visual\resource\tclTkGUIBuilder.rc
smi\Visual\resource\tclTkGUIBuilder_rc.h
smi\Visual\resource\tclTkGUIBuilder_rc.ico
smi\Visual\resource\tellMonObjects.rc
smi\Visual\resource\tellMonObjects_rc.h
smi\Visual\resource\tellMonObjects_rc.ico
```

Приложение 2. Модуль smirtl.pas.

Модуль **smirtl.pas** содержит программный интерфейс (**API**) для подключения динамической библиотеки **SMIRTL.DLL** – библиотеки времени выполнения для создания прокси (*proxy*) программ для распределенных систем управления на базе Конечных Автоматов **SMI**.

```
////////////////////////////////////////////////////////////////////////////////
//                                                                            //
// Copyright (c) 2021 Alexey Kuryakin kouriakine@mail.ru under LGPL license.  //
//                                                                            //
// Purpose:                                                                   //
//  Header for SMI++ Runtime Library SMIRTL.DLL                               //
//  See http://smi.web.cern.ch                                                //
//                                                                            //
// History:                                                                   //
//  20201123 - 1st release                                                    //
//  20210122 - Update to v56r1                                                //
////////////////////////////////////////////////////////////////////////////////

unit smirtl; // SMI++ runtime library header, see http://smi.web.cern.ch

interface

uses SysUtils;

 //
 // parameters.h
 //
const
  MAXRECL           = 81;
  MAXNAMESIZE       = 33;
  MAXOBJECTLINES    = 5000;
  MAXCLASSLINES     = 100;
  MAXSTATES         = 100;
  MAXATRIBUTES      = 10;
  MAXSUBOBJECTS     = 50;
  MAXACTIONS        = 100;
  MAXINS            = 100;
  MAXPARA           = 5;
  MAXWHENS          = 20;

 ////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
 //
 // smirtl.h
 //
 // SMIRTL implements the routines used by a proxy in order to connect,
 //        receive commands and transmit state changes to its associated object in a SM.
 //
 //
 // int smi_attach(obj_name, command_handler)
 // Parameters:
 //     char *obj_name              The name of the associated object to connect to. In the form <domain_name>::<object_name>
 //      void (*command_handler)()   The routine to be called when a command arrives from the SM, handle_command is called with no
parameters.
 // Description:
 //     smi_attach should be called by a proxy at startup in order to connect to its associated object in the SM.
 // Return:
 //     0/1 if object not/attached.
 //
 //
 // void smi_volatile()
 // Parameters:
 //     none
 // Description:
 //     Smi_volatile() makes a proxy live only while its associated SMI Domain is active, i.e., a proxy can call
 //     smi_volatile() (normally at start-up) in order to die whenever the SMI domain the proxy is attached to dies.
 //
 //
 // int smi_set_state(state)
 // Parameters:
 //     char *state         The state of the proxy.
 // Description:
 //     Set the initial or new state of the proxy by sending it to its associated object in the SM.
 //     An smi_set_state should be done at start-up in order to communicate the proxy's initial state.
 // Return:
 //     1.
 //
 //
 // smi_terminate_action(state)
 // Parameters:
 //     char *state         The state of the proxy.
 // Description:
 //     Set the new state of the proxy after executing an action, by sending it to its associated object in the SM.
 // Return:
 //     1.
 //
 //
 // int smi_get_action(action, n_params)
 // Parameters:
 //     char *action        The last action received
 //     int *n_params       Number of parameters with this action
 // Description:
 //     When inside the comman_handler the user can use this routine to get the action name and nr. of parameters.
 // Return:
 //     0/1 if action is not/valid.
 //
 //
 // int smi_get_command(cmnd)
 // Parameters:
 //     char *cmnd          The command just received
 //     int *size           The command size (length)
```

```
// Description:
//     When inside the comman_handler the user can use this routine to get the command, the command is composed
//     of the action name and any parameters in the form action/par0=val0/par1=val1… .
// Return:
//     0/1 if command not/valid.
//
//
// int smi_test_action(action)
// Parameters:
//     char *action        Action name to compare with the received action.
// Description:
//     Test if the action received is "action". The function returns 1 if it is, 0 if not.
// Return:
//     0/1 if action not/match.
//
//
// int smi_get_next_par(param, type, size)
// Parameters:
//     char *param         The name of the parameter
//     int *type           The type of the parameter (STRING, INTEGER, FLOAT)
//     int *size           The size of the parameter
// Description:
//     When inside the command handler this routine can be used to retrieve the parameters present on a received command.
//     The function will return 0 when all parameters have been read, otherwise it returns the parameter name,
//     the parameter type and the size necessary to store the parameter.
// Return:
//     0/1 if parameter not/retrieved.
//
//
// int smi_get_par_value(param, value)
// Parameters:
//     char *param         The name of the parameter
//     void *value         The parameter value
// Description:
//     When inside the command handler this routine can be used to retrieve the parameter value for the parameter
//     returned by smi_get_next_par. The value argument will be filled with either an integer, a double float or
//     a string (the size was specified in the previous routine) depending on the parameter type.
//     This function can also be used directly if the user knows the parameter name, type and size.
//     The function will return 0 if the specified parameter does not exist.
// Return:
//     0/1 if parameter not/retrieved.
//
//
// int smi_set_par(par, value, type)
// Parameters:
//     char *par           The parameter name.
//     void *value         The value of the parameter (passed by reference).
//     int type            The parameter type: (STRING, INTEGER or FLOAT).
// Description:
//     Smi_set_par should be used in order to set object parameters for associated objects.
//     If an associated objects has parameters, smi_set_par should be called once for each parameter before the first smi_set_state.
//     Further parameter setting can be done by the proxy at any time but they will only take effect at the next smi_set_state.
// Return:
//     0/1 if parameter was not/set.
//
//
// void smi_register_termination(exit_handler, context)
// Parameters:
//     int (*exit_handler)(void *, int *)
//     void *context
// Description:
//     Internal use.
//
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

const smirtl_dll                    = 'smirtl.dll';                    // SMI runtime DLL
const SMIXX_VERSION                 = 5601;                            // Version of SMI++
const MAX_NAME                      = 132;                             // Max. size of name
const SMI_STRING                    = 0;                               // String  parameter
const SMI_INTEGER                   = 1;                               // Integer parameter
const SMI_FLOAT                     = 2;                               // Float   parameter
type  PAR_TYPES                     = SMI_STRING .. SMI_FLOAT;         // Parameter types
type  smi_cmnd_handler_t            = procedure(); cdecl;              // Command handler
type  smi_exit_handler_t            = function(context:Pointer; var ExitCode:Integer):Integer; cdecl; // Exit handler

function  smi_attach(obj_name:PChar; command_handler: smi_cmnd_handler_t): Integer; cdecl;
procedure smi_volatile(); cdecl;
function  smi_set_state(state:PChar): Integer; cdecl;
function  smi_get_state(state:PChar; max_len:Integer): Integer; cdecl;
function  smi_terminate_action(state:PChar): Integer; cdecl;
function  smi_get_action(cmnd:PChar; var n_pars:Integer): Integer; cdecl;
function  smi_get_command(cmnd:PChar; var size:Integer): Integer; cdecl;
function  smi_test_action(cmnd:PChar): Integer; cdecl;
function  smi_get_next_par(param:PChar; var typ:Integer; var size:Integer): Integer; cdecl;
function  smi_get_par_value(param:PChar; value:Pointer): Integer; cdecl;
function  smi_set_par(param:PChar; value:Pointer; typ:Integer): Integer; cdecl;
procedure smi_register_termination(exit_handler:smi_exit_handler_t; context:Pointer); cdecl;

implementation

function  smi_attach;               external smirtl_dll name 'smi_attach';
procedure smi_volatile;             external smirtl_dll name 'smi_volatile';
function  smi_set_state;            external smirtl_dll name 'smi_set_state';
function  smi_get_state;            external smirtl_dll name 'smi_get_state';
function  smi_terminate_action;     external smirtl_dll name 'smi_terminate_action';
function  smi_get_action;           external smirtl_dll name 'smi_get_action';
function  smi_get_command;          external smirtl_dll name 'smi_get_command';
function  smi_test_action;          external smirtl_dll name 'smi_test_action';
function  smi_get_next_par;         external smirtl_dll name 'smi_get_next_par';
function  smi_get_par_value;        external smirtl_dll name 'smi_get_par_value';
function  smi_set_par;              external smirtl_dll name 'smi_set_par';
procedure smi_register_termination; external smirtl_dll name 'smi_register_termination';

end.
```

Приложение 3. Модуль smiuirtl.pas.

Модуль **smiuirtl.pas** содержит программный интерфейс (**API**) для подключения динамической библиотеки **SMIUIRTL.DLL** – библиотеки времени выполнения для создания клиентских программ графического интерфейса пользователя **GUI** для распределенных систем управления на базе Конечных Автоматов **SMI**.

```
////////////////////////////////////////////////////////////////////////////
//                                                                        //
// Copyright (c) 2021 Alexey Kuryakin kouriakine@mail.ru under LGPL license.  //
//                                                                        //
// Purpose:                                                               //
//  Header for SMI++ User Interface Runtime Library SMIUIRTL.DLL          //
//  See http://smi.web.cern.ch                                            //
//                                                                        //
// History:                                                               //
//  20201123 - 1st release                                                //
//  20210122 - Update to v56r1                                            //
////////////////////////////////////////////////////////////////////////////

unit smiuirtl; // SMI++ UI runtime library header, see http://smi.web.cern.ch

interface

uses SysUtils, smirtl;

 ////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
 //
 // smiuirtl.h
 //
 // SMIUIRTL implements the routines used by a user interface or any other process wanting to get information
 //          on the states of objects running in a certain SM and/or to send commands to them.
 //
 // smiui_current_state(obj_name, state, busy_action)
 // Parametes:
 //     char *obj_name      The name of the object in the form <domain_name>::<object_name>
 //     char *state         The buffer where the state will be stored on return.
 //     char *busy_action   If the object is busy the name of the action it is currently executing, otherwise a null string.
 // Description:
 //     Used to know the current state of an object in a given domain. It also returns information
 //     on whether the object is busy executing an action or idle.
 //     Return status 0/1 = object not/connected.
 //
 //
 // smiui_send_command(obj_name, cmnd)
 // smiui_ep_send_command(obj_name, cmnd)
 // Parameters:
 //     char *obj_name      The name of the object in the form <domain_name>::<object_name>
 //     char *cmnd          The cmnd in the form action/par="val"…
 // Description:
 //     Used to send a command to any object in any domain.
 //     *_ep_* send command to proxy directly.
 //     Return 1 always (nonblocking call).
 //
 //
 // smiui_send_command_wait(obj_name, cmnd)
 // smiui_ep_send_command_wait(obj_name, cmnd)
 // Parameters:
 //     char *obj_name      The name of the object in the form <domain_name>::<object_name>
 //     char *cmnd          The cmnd in the form action/par="val"…
 // Description:
 //     Used to send a command to any object in any domain.
 //     Return 0/1 if command was not/sent (blocking call).
 //     *_ep_* send command to proxy directly.
 //
 //
 // smiui_change_option(domain, option, value)
 // smiui_change_option_wait(domain, option, value)
 // Parameters:
 //     char *domain        The name of the domain
 //     char *option        Option name
 //     char *value         Option value
 // Description:
 //     Used to change option value of domain server.  *_wait uses blocking call.
 //     Return 0/1 (nonblocking always 1).
 //
 //
 // smiui_get_options(domain, options)
 // Parameters:
 //     char *domain        The name of the domain
 //     char *options       Options buffer >= 512 bytes
 // Description:
 //     Get all options values of domain server.
 //     Options=d/INT/3/Diagnostic Level|u/BOOL/0/Unlocked IFs|t/BOOL/1/New time format|dns/STRING/localhost/Dns node ...
 //     Return 0/1 status.
 //
 //
 // int smiui_connect_domain(domain)
 // Parameters:
 //     char *domain        The name of the domain to connect to.
 // Description:
 //     Used to connect to a specific domain, it's a blocking call, it waits until the domain responds.
 //     It returns the number of objects in the domain or 0 if it fails. The calls smiui_get_next_object,
 //     smiui_get_next_objectset call smiui_connect_domain implicitly if not done before.
 //
 //
 // int smiui_book_connect_domain(domain, domainchange_handler, param)
 // Parameters:
```

```
//     char *domain        The name of the domain.
//     void (*domainchange_handler)(dim_long *par, int *nobjs)  The routine to handle domain changes
//     dim_long param      A parameter to pass to the above routine
// Description:
//     Set up a callback routine to be called whenever the domain starts or stops.
//     The domainchange_handler routine will be called with two parameters: param and the number of objects
//     in the domain - both parameters are passed by reference.
//
//
// smiui_cancel_connect_domain(domain)
// Parameters:
//     char *domain        The name of the domain.
// Description:
//     Cancel attention on the domain, further changes will not be signaled.
//
//
// int smiui_number_of_objects(domain)
// Parameters:
//     char *domain        The name of the domain.
// Description:
//     Returns the number of objects in the domain or 0 if it fails.
//
//
// int smiui_get_next_object(domain, obj_name)
// Parameters:
//     char *domain        The name of the domain.
//     char *obj_name      The name of the object in the form <domain_name>::<object_name>.
// Description:
//     Get the name of the next object in a domain (several domains can be accessed at the "same time").
//     This routine returns 0 when all the objects of a particular domain have been retrieved
//
//
// int smiui_book_statechange(obj_name, statechange_handler, param)
// Parameters:
//     char *obj_name                  The name of the object.
//     void (*statechange_handler)(int *id, dim_long *par)   The routine to handle state changes
//     long param                      A parameter to pass to the above routine
// Description:
//     Set up a callback routine to be called whenever the specified object changes state.
//     smiui_book_statechange returns an identifier that should be used on the following calls.
//     The statechange_handler routine will be called with two parameters: the address of the identifier mentioned
//     above and the address of param (i.e. both parameters are passed by reference).
//     See the Example for clarification
//
// /**********************************************************************
//  This small selfcontained program will print a line of text everytime
//    either object DOMAIN::A   or  DOMAIN::B changes state
// **********************************************************************/
// #include <stdio.h>
// #include <smiuirtl.h>    /* the header file provided as part of SMIXX package*/
// /*  Look at the main program first */
// /*-----------------------------------------------------------*/
// void handler(int* pId, long* pParam)
// {
//     int Id; long Param;
//     int busy_flag,nac;
//     char objName[32];
//     char curr_state[32];
//     Id =  *pId;
//     Param = *pParam;
//      smiui_get_name(Id,objName);  /* Id of the booking call passed to handler is used to retrieve the object name of the object
that changed state */
//     smiui_get_state(Id,&busy_flag,curr_state,&nac);  /* Id of the booking call passed to handler is used to retrieve the name of
the object's current state */
//     printf("\n Object : %s   Current state : %s \n",objName,curr_state); fflush(stdout);
//     return;
// }
// /*-----------------------------------------------------------------*/
// int main(int argc, char* argv[])
// {
//     int Id1; char objectName1[] = "DOMAIN::A"; long param1 = 0;
//     int Id2; char objectName2[] = "DOMAIN::B"; long param2 = 0;
// /* The two following calls register callback function 'handler' that
//    will be called by SMI system everytime either of the two objects changes its state.
//    We could have register two different callbacks, but in this example we
//    opted for the same one for both objects.
//    Id1 and Id2 are identification codes assigned and returned by SMI
//    system and are passed to the callback function.
//    parm1 and parm2 are arbitrary user parameters that are also passed
//    to the callback function. In this example we make no use of them
//    and so they are set to zero
// */
//     Id1 = smiui_book_statechange( objectName1, handler ,param1);
//     Id2 = smiui_book_statechange( objectName2, handler ,param2);
//     while (1)
//     {
//         sleep(1);
//     }
//     return 0;
// }
//
//
// smiui_cancel_statechange(id)
// Parameters:
//     int id              The identifier returned by the previous call.
// Description:
//     Cancel attention on the object, further state changes will not be signaled.
//
//
// smiui_get_name(id, obj_name)
// Parameters:
//     int id              The identifier returned by smiui_book_statechange
//     char *obj_name      Returns the name of the current object
// Description:
//     This routine is normally called from the statechange_handler routine in order to get the name of the current object.
//     See the Example for clarification
//
//
```

```
// smiui_get_state(id, busi_flag, state, n_actions)
// Parameters:
//     int id              The identifier returned by smiui_book_statechange.
//     int *busy_flag      Returns a flag indicating if the current object is busy
//     char *state         Returns the current state of the object
//     int *n_actions      Returns the number of possible actions in the object's current state
// Description:
//      This routine is normally called from the statechange_handler routine in order to get the object state and some additional
parameters.
//     See the Example for clarification
//
//
// smiui_get_action_in_progress(id, action)
// Parameters:
//     int id              The identifier returned by smiui_book_statechange.
//     char *action        The action currently being executed.
// Description:
//     Called in order to get the action currently being executed by an object whenever the busy_flag in the previous routine is 1.
//
//
// int smiui_get_next_action(id, action, n_pars)
// Parameters:
//     int id              The identifier returned by smiui_book_statechange.
//     char *action        The action name.
//     int n_pars          The number of parameters this action takes.
// Description:
//     This routine retrieves the next action available to an object in the current state.
//
//
// smiui_get_first_action(id, action, n_pars)
// Parameters:
//     int id              The identifier returned by smiui_book_statechange.
//     char *action        The action name.
//     int n_pars          The number of parameters this action takes
// Description:
//     Positions the next_action pointer back to the first action and retrieves it.
//     smiui_get_next_action can be called again from then on.
//
//
// int smiui_get_next_param(id, param, type, default_value_size)
// Parameters:
//     int id              The identifier returned by smiui_book_statechange.
//     char *param         The action's parameter name.
//     int *type           The parameter type (SMI_STRING, SMI_ INTEGER, SMI_ FLOAT)
//     int *default_value_size     The size of the parameter's default value if a default has been specified, otherwise 0.
// Description:
//     This routine retrieves the next parameter specified for the current action available to an object in the current state.
//     It returns 0 when all parameters have been returned.
//
//
// int smiui_get_param_default_value(id, default_value)
// Parameters:
//     int id              The identifier returned by smiui_book_statechange.
//     void *default_value The parameter's default value.
// Description:
//     This routine retrieves the parameter default value, for the parameter received from smiui_get_next_param.
//     This routine should only be called if default_value_size was not zero.
//     It returns 0 if the parameter did not have a default value.
//     The contents of default_value can be of type integer, float or character string according to the previous routine.
//
//
// int smiui_get_next_obj_param(id, param, type, value_size)
// Parameters:
//     int id              The identifier returned by smiui_book_statechange.
//     char *param         The object's parameter name.
//     int *type           The parameter type (SMI_STRING, SMI_ INTEGER, SMI_ FLOAT)
//     int *value_size     The size of the parameter's value.
// Description:
//     This routine retrieves the next parameter specified for the current object.
//     It returns 0 when all parameters have been returned.
//
//
// int smiui_get_obj_param_value(id, value)
// Parameters:
//     int id              The identifier returned by smiui_book_statechange.
//     void *value         The parameter's value.
// Description:
//     This routine retrieves the parameter value, for the parameter received from smiui_get_next_obj_param.
//     The contents of value can be of type integer, float or character string according to the previous routine.
//
//
// int smiui_number_of_objectsets(domain)
// Parameters:
//     char *domain        The name of the domain.
// Description:
//     Returns the number of objects sets in the domain.
//
//
// int smiui_get_next_objectset(domain, objset_name)
// Parameters:
//     char *domain        The name of the domain.
//     char *objset_name   The name of the object set in the form <domain_name>::<objectset_name>.
// Description:
//     Get the name of the next object set in a domain (several domains can be accessed at the "same time").
//     This routine returns 0 when all the object sets of a particular domain have been retrieved
//
//
// int smiui_book_objectsetchange(objset_name, objectsetchange_handler, param)
// Parameters:
//     char *objset_name                   The name of the object set.
//     void (*objectsetchange_handler)(int *id, dim_long *par)   The routine to handle changes in the contents of the set.
//     long param                          A parameter to pass to the above routine
// Description:
//      Set up a callback routine to be called whenever the specified object set changes (i.e. objects are added or removed from the
set).
//     smiui_book_objectsetchange returns an identifier that should be used on the following calls.
//     The objectsetchange_handler routine will be called with two parameters: the address of the identifier mentioned
//     above and the address of param (i.e. both parameters are passed by reference).
```

```
//
//
// smiui_cancel_objectsetchange(id)
// Parameters:
//     int id              The identifier returned by the previous call.
// Description:
//     Cancel attention on the object set, further set changes will not be signaled.
//
//
// smiui_get_setname(id, objset_name)
// Parameters:
//     int id              The identifier returned by smiui_book_objectsetchange
//     char *objset_name   Returns the name of the current object set
// Description:
//     This routine is normally called from the objectsetchange_handler routine in order to get the name of the current object set.
//
//
// int smiui_get_next_object_in_set(id, object)
// Parameters:
//     int id              The identifier returned by smiui_book_objectsetchange.
//     char *object        The object name.
// Description:
//     This routine retrieves the next object contained in the current object set, to be called from within objectsetchange_handler.
//
//
// smiui_get_first_object_in_set(id, object)
// Parameters:
//     int id              The identifier returned by smiui_book_objectsetchange.
//     char *object        The object name.
// Description:
//     Positions the next_object pointer back to the first object is the set and retrieves it.
//     smiui_get_next_object_in_set can be called again from then on.
//
//
// int smiui_book_alloc_statechange(domain, alloc_statechange_handler, param)
// Parameters:
//     char *domain                            The SMI domain name.
//     void (*alloc_statechange_handler)(int *id, dim_long *par)     The routine to handle state changes
//     long param                              A parameter to pass to the above routine
// Description:
//     Set up a callback routine to be called whenever the allocation state of the specified domain changes state.
//     smiui_book_alloc_statechange returns an identifier that should be used on the following calls.
//     The alloc_statechange_handler routine will be called with two parameters: the address of the identifier
//     mentioned above and the address of param (i.e. both parameters are passed by reference).
//
//
// smiui_cancel_alloc_statechange(id)
// Parameters:
//     int id              The identifier returned by the previous call.
// Description:
//     Cancel attention on the allocation state of the domain, further state changes will not be signalled.
//
//
// smiui_get_alloc_state(id, busy_flag, state)
// Parameters:
//     int id              The identifier returned by smiui_book_alloc_statechange.
//     int *busy_flag      Returns a flag indicating if the domain allocation object is currently busy
//     char *state         Returns the current state of the allocation object
// Description:
 //       This routine is normally called from the alloc_statechange_handler routine in order to get the allocation state of the
domain.
 //
 //
 // int smiui_allocate(domain)
 // Parameters:
 //     char *domain        The domain name.
 // Description:
 //     This routine can be used by a user interface or another process in order to allocate an SMI domain to itself.
 //     An smiui_connect_domain should have been done before hand. Smiui_allocate domain returns 1 if the domain was
 //     successfully allocated and 0 if it was not connected or already allocated to someone else.
 //
 //
 // int smiui_allocate_to(domainA, domainB)
 // Parameters:
 //     char *domainA       The domain to be allocated.
 //     char *domainB       The domain to allocate to.
 // Description:
 //     This routine can be used by a user interface or another process in order to allocate an SMI domain to another SMI domain.
 //     An smiui_connect_domain should have been done before hand to the allocator domain (domainB).
 //     Smiui_allocate domain returns 1 if domainA was successfully allocated to domainB and 0
 //     if it domainB was not connected or domainA already allocated to someone else.
 //
 //
 // int smiui_release(domain)
 // Parameters:
 //     char *domain        The domain name.
 // Description:
 //     This routine can be used by a user interface or another process in order to release a previously allocated SMI domain.
 //     An smiui_connect_domain should have been done before hand. Smiui_release domain returns 1
 //     if the domain was successfully released and 0 if the domain was not allocated to this process.
 //
 //
 // int smiui_release_from(domainA, domainB)
 // Parameters:
 //     char *domainA       The domain allocated.
 //     char *domainB       The domain that allocated domainA.
 // Description:
 //     This routine can be used by a user interface or another process in order to release a domain from another one.
 //     An smiui_connect_domain should have been done before hand to domainB. Smiui_release domain returns 1
 //     if domainA was successfully released from domainB and 0 if domainA was not allocated to domainB.
 //
 ////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

const smiuirtl_dll                  = 'smiuirtl.dll';                   // SMI++ UI runtime DLL
const SMIXX_VERSION                 = 5601;                             // Version of SMI++
const MAX_NAME                      = 132;                              // Max. size of name
const SMI_STRING                    = 0;                                // String  parameter
const SMI_INTEGER                   = 1;                                // Integer parameter
```

```
const SMI_FLOAT                   = 2;                              // Float   parameter
type  dim_long_t                  = longint;                        // Should be pointer-size integer
type  smiui_domainchange_handler_t  = procedure(var par:dim_long_t; var nobjs:Integer); cdecl; // Domain change handler
type  smiui_statechange_handler_t   = procedure(var id:Integer; var par:dim_long_t); cdecl; // State change handler
type  smiui_setchange_handler_t     = procedure(var id:Integer; var par:dim_long_t); cdecl; // Object set change handler
type  smiui_message_handler_t       = procedure(var id:Integer; var par:dim_long_t); cdecl; // Message handler

function smiui_set_dns(node:PChar; port:Integer):dim_long_t; cdecl;
function smiui_current_state(obj:PChar; state:PChar; action:PChar):Integer; cdecl;
function smiui_current_state_dns( dnsid:dim_long_t; obj:PChar; state:PChar; action:PChar):Integer; cdecl;
function smiui_send_command(obj:PChar; cmnd:PChar):Integer; cdecl;
function smiui_send_command_dns(dnsid:dim_long_t; obj:PChar; cmnd:PChar):Integer; cdecl;
function smiui_ep_send_command(obj:PChar; cmnd:PChar):Integer; cdecl;
function smiui_ep_send_command_dns(dnsid:dim_long_t; obj:PChar; cmnd:PChar):Integer; cdecl;
function smiui_send_command_wait(obj:PChar; cmnd:PChar):Integer; cdecl;
function smiui_send_command_wait_dns(dnsid:dim_long_t; obj:PChar; cmnd:PChar):Integer; cdecl;
function smiui_ep_send_command_wait(obj:PChar; cmnd:PChar):Integer; cdecl;
function smiui_ep_send_command_wait_dns(dnsid:dim_long_t; obj:PChar; cmnd:PChar):Integer; cdecl;
function smiui_change_option(domain:PChar; option:PChar; value:PChar):Integer; cdecl;
function smiui_change_option_wait(domain:PChar; option:PChar; value:PChar):Integer; cdecl;
function smiui_get_options(domain:PChar; optString:PChar):Integer; cdecl;
function smiui_get_options_dns(dnsid:dim_long_t; domain:PChar; optString:PChar):Integer; cdecl;
function smiui_number_of_objects(domain:PChar):Integer; cdecl;
function smiui_number_of_objects_dns(dnsid:dim_long_t; domain:PChar):Integer; cdecl;
function smiui_connect_domain(domain:PChar):Integer; cdecl;
function smiui_connect_domain_dns(dnsid:dim_long_t; domain:PChar):Integer; cdecl;
function smiui_book_connect_domain(domain:PChar; command_handler:smiui_domainchange_handler_t; par:dim_long_t):Integer; cdecl;
function       smiui_book_connect_domain_dns(dnsid:dim_long_t;       domain:PChar;       command_handler:smiui_domainchange_handler_t;
par:dim_long_t):Integer; cdecl;
function smiui_cancel_connect_domain(domain:PChar):Integer; cdecl;
function smiui_shutdown_domain(domain:PChar):Integer; cdecl;
function smiui_check_proxy(proxy:PChar):Integer; cdecl;
function smiui_check_proxy_dns(dnsid:dim_long_t; proxy:PChar):Integer; cdecl;
function smiui_kill(server:PChar):Integer; cdecl;
function smiui_get_next_object(domain:PChar; obj:PChar):Integer; cdecl;
function smiui_get_next_attribute(obj:PChar; attribute:PChar):Integer; cdecl;
function smiui_book_statechange(obj:PChar; command_handler:smiui_statechange_handler_t; par:dim_long_t):Integer; cdecl;
function         smiui_book_statechange_dns(dnsid:dim_long_t;         obj:PChar;         command_handler:smiui_statechange_handler_t;
par:dim_long_t):Integer; cdecl;
function smiui_cancel_statechange(id:Integer):Integer; cdecl;
function smiui_get_name(id:Integer; name:PChar):Integer; cdecl;
function smiui_get_state(id:Integer; var busy_flg:Integer; state:PChar; var n_actions:Integer):Integer; cdecl;
function smiui_get_smi_message(id:Integer; dom:PChar; mess:PChar):Integer; cdecl;
function smiui_get_user_message(id:Integer; dom:PChar; mess:PChar):Integer; cdecl;
function smiui_get_action_in_progress(id:Integer; action:PChar):Integer; cdecl;
function smiui_get_first_action(id:Integer; action:PChar; var n_pars:Integer):Integer; cdecl;
function smiui_get_next_action(id:Integer; action:PChar; var n_pars:Integer):Integer; cdecl;
function smiui_get_next_param(id:Integer; par:PChar; var typ:Integer; var size:Integer):Integer; cdecl;
function smiui_get_param_default_value(id:Integer; value:Pointer):Integer; cdecl;
function smiui_get_next_obj_param(id:Integer; par:PChar; var typ:Integer; var size:Integer):Integer; cdecl;
function smiui_get_obj_param_value(id:Integer; value:Pointer):Integer; cdecl;
function smiui_book_alloc_statechange(domain:PChar; command_handler:smiui_statechange_handler_t; par:dim_long_t):Integer; cdecl;
function      smiui_book_alloc_statechange_dns(dnsid:dim_long_t;      domain:PChar;      command_handler:smiui_statechange_handler_t;
par:dim_long_t):Integer; cdecl;
function smiui_cancel_alloc_statechange(id:Integer):Integer; cdecl;
function smiui_get_alloc_state(id:Integer; var busy_flg:Integer; state:PChar):Integer; cdecl;
function smiui_allocate(domain:PChar):Integer; cdecl;
function smiui_release(domain:PChar):Integer; cdecl;
function smiui_allocate_to(domainA:PChar; domainB:PChar):Integer; cdecl;
function smiui_release_from(domainA:PChar; domainB:PChar):Integer; cdecl;
function smiui_get_next_objectset(domain:PChar; objectset:PChar):Integer; cdecl;
function smiui_book_objectsetchange(obj:PChar; command_handler:smiui_setchange_handler_t; par:dim_long_t):Integer; cdecl;
function        smiui_book_objectsetchange_dns(dnsid:dim_long_t;        obj:PChar;        command_handler:smiui_setchange_handler_t;
par:dim_long_t):Integer; cdecl;
function smiui_cancel_objectsetchange(id:Integer):Integer; cdecl;
function smiui_book_smi_message(dom:PChar; command_handler:smiui_message_handler_t; par:dim_long_t):Integer; cdecl;
function smiui_cancel_smi_message(id:Integer):Integer; cdecl;
function smiui_book_user_message(dom:PChar; command_handler:smiui_message_handler_t; par:dim_long_t):Integer; cdecl;
function smiui_cancel_user_message(id:Integer):Integer; cdecl;
function smiui_get_first_object_in_set(id:Integer; obj:PChar):Integer; cdecl;
function smiui_get_next_object_in_set(id:Integer; obj:PChar):Integer; cdecl;
function smiui_get_setname(id:Integer; objset:PChar):Integer; cdecl;
function smiui_number_of_objectsets(domain:PChar):Integer; cdecl;
function smiui_number_of_objectsets_dns(dnsid:dim_long_t; domain:PChar):Integer; cdecl;
function smiui_create_command(action:PChar):Pointer; cdecl;
function smiui_add_param_to_command(id:Pointer; name:PChar; value:Pointer; typ:Integer):Integer; cdecl;
function smiui_delete_command(cmnd:Pointer):Integer; cdecl;
function smiui_get_command_string(id:Pointer):PChar; cdecl;
function smiui_get_action_in_progress_size(id:Integer; var size:Integer):Integer; cdecl;

implementation

function  smiui_set_dns;                      external smiuirtl_dll name 'smiui_set_dns';
function  smiui_current_state;                external smiuirtl_dll name 'smiui_current_state';
function  smiui_current_state_dns;            external smiuirtl_dll name 'smiui_current_state_dns';
function  smiui_send_command;                 external smiuirtl_dll name 'smiui_send_command';
function  smiui_send_command_dns;             external smiuirtl_dll name 'smiui_send_command_dns';
function  smiui_ep_send_command;              external smiuirtl_dll name 'smiui_ep_send_command';
function  smiui_ep_send_command_dns;          external smiuirtl_dll name 'smiui_ep_send_command_dns';
function  smiui_send_command_wait;            external smiuirtl_dll name 'smiui_send_command_wait';
function  smiui_send_command_wait_dns;        external smiuirtl_dll name 'smiui_send_command_wait_dns';
function  smiui_ep_send_command_wait;         external smiuirtl_dll name 'smiui_ep_send_command_wait';
function  smiui_ep_send_command_wait_dns;     external smiuirtl_dll name 'smiui_ep_send_command_wait_dns';
function  smiui_change_option;                external smiuirtl_dll name 'smiui_change_option';
function  smiui_change_option_wait;           external smiuirtl_dll name 'smiui_change_option_wait';
function  smiui_get_options;                  external smiuirtl_dll name 'smiui_get_options';
function  smiui_get_options_dns;              external smiuirtl_dll name 'smiui_get_options_dns';
function  smiui_number_of_objects;            external smiuirtl_dll name 'smiui_number_of_objects';
function  smiui_number_of_objects_dns;        external smiuirtl_dll name 'smiui_number_of_objects_dns';
function  smiui_connect_domain;               external smiuirtl_dll name 'smiui_connect_domain';
function  smiui_connect_domain_dns;           external smiuirtl_dll name 'smiui_connect_domain_dns';
function  smiui_book_connect_domain;          external smiuirtl_dll name 'smiui_book_connect_domain';
function  smiui_book_connect_domain_dns;      external smiuirtl_dll name 'smiui_book_connect_domain_dns';
function  smiui_cancel_connect_domain;        external smiuirtl_dll name 'smiui_cancel_connect_domain';
function  smiui_shutdown_domain;              external smiuirtl_dll name 'smiui_shutdown_domain';
function  smiui_check_proxy;                  external smiuirtl_dll name 'smiui_check_proxy';
function  smiui_check_proxy_dns;              external smiuirtl_dll name 'smiui_check_proxy_dns';
function  smiui_kill;                         external smiuirtl_dll name 'smiui_kill';
```

```
function  smiui_get_next_object;                external smiuirtl_dll name 'smiui_get_next_object';
function  smiui_get_next_attribute;             external smiuirtl_dll name 'smiui_get_next_attribute';
function  smiui_book_statechange;               external smiuirtl_dll name 'smiui_book_statechange';
function  smiui_book_statechange_dns;           external smiuirtl_dll name 'smiui_book_statechange_dns';
function  smiui_cancel_statechange;             external smiuirtl_dll name 'smiui_cancel_statechange';
function  smiui_get_name;                       external smiuirtl_dll name 'smiui_get_name';
function  smiui_get_state;                      external smiuirtl_dll name 'smiui_get_state';
function  smiui_get_smi_message;                external smiuirtl_dll name 'smiui_get_smi_message';
function  smiui_get_user_message;               external smiuirtl_dll name 'smiui_get_user_message';
function  smiui_get_action_in_progress;         external smiuirtl_dll name 'smiui_get_action_in_progress';
function  smiui_get_first_action;               external smiuirtl_dll name 'smiui_get_first_action';
function  smiui_get_next_action;                external smiuirtl_dll name 'smiui_get_next_action';
function  smiui_get_next_param;                 external smiuirtl_dll name 'smiui_get_next_param';
function  smiui_get_param_default_value;        external smiuirtl_dll name 'smiui_get_param_default_value';
function  smiui_get_next_obj_param;             external smiuirtl_dll name 'smiui_get_next_obj_param';
function  smiui_get_obj_param_value;            external smiuirtl_dll name 'smiui_get_obj_param_value';
function  smiui_book_alloc_statechange;         external smiuirtl_dll name 'smiui_book_alloc_statechange';
function  smiui_book_alloc_statechange_dns;     external smiuirtl_dll name 'smiui_book_alloc_statechange_dns';
function  smiui_cancel_alloc_statechange;       external smiuirtl_dll name 'smiui_cancel_alloc_statechange';
function  smiui_get_alloc_state;                external smiuirtl_dll name 'smiui_get_alloc_state';
function  smiui_allocate;                       external smiuirtl_dll name 'smiui_allocate';
function  smiui_release;                        external smiuirtl_dll name 'smiui_release';
function  smiui_allocate_to;                    external smiuirtl_dll name 'smiui_allocate_to';
function  smiui_release_from;                   external smiuirtl_dll name 'smiui_release_from';
function  smiui_get_next_objectset;             external smiuirtl_dll name 'smiui_get_next_objectset';
function  smiui_book_objectsetchange;           external smiuirtl_dll name 'smiui_book_objectsetchange';
function  smiui_book_objectsetchange_dns;       external smiuirtl_dll name 'smiui_book_objectsetchange_dns';
function  smiui_cancel_objectsetchange;         external smiuirtl_dll name 'smiui_cancel_objectsetchange';
function  smiui_book_smi_message;               external smiuirtl_dll name 'smiui_book_smi_message';
function  smiui_cancel_smi_message;             external smiuirtl_dll name 'smiui_cancel_smi_message';
function  smiui_book_user_message;              external smiuirtl_dll name 'smiui_book_user_message';
function  smiui_cancel_user_message;            external smiuirtl_dll name 'smiui_cancel_user_message';
function  smiui_get_first_object_in_set;        external smiuirtl_dll name 'smiui_get_first_object_in_set';
function  smiui_get_next_object_in_set;         external smiuirtl_dll name 'smiui_get_next_object_in_set';
function  smiui_get_setname;                    external smiuirtl_dll name 'smiui_get_setname';
function  smiui_number_of_objectsets;           external smiuirtl_dll name 'smiui_number_of_objectsets';
function  smiui_number_of_objectsets_dns;       external smiuirtl_dll name 'smiui_number_of_objectsets_dns';
function  smiui_create_command;                 external smiuirtl_dll name 'smiui_create_command';
function  smiui_add_param_to_command;           external smiuirtl_dll name 'smiui_add_param_to_command';
function  smiui_delete_command;                 external smiuirtl_dll name 'smiui_delete_command';
function  smiui_get_command_string;             external smiuirtl_dll name 'smiui_get_command_string';
function  smiui_get_action_in_progress_size;    external smiuirtl_dll name 'smiui_get_action_in_progress_size';

end.
```

Приложение 4. Опции сервера SM

Когда работает **SM**, некоторые его действия можно контролировать с помощью параметров времени выполнения. Эти параметры изначально указываются в команде запуска (**smiSM**), например:

```
smiSM -d 6 -t 1  domain_name  file_name
```

Значения большинства из них могут динамически изменяться во время работы *State Manager*. Смотри ниже. Вот эти опции.
Доступные опции:

Option	Option Id	Comment
Уровень диагностики	d	Целое число, значение которого определяет объем генерируемой диагностической печати. Значение по умолчанию – 3. Максимальное значение – 9.
Разблокирован флаг IF	u	Значение 0 или 1. Значение по умолчанию – 0. Влияет на способ обработки инструкций IF. Подробно описано в другом месте.
Флаг нового формата времени	t	Значение 0 или 1. Значение по умолчанию – 0. Сообщения, такие как диагностические распечатки, имеют отметку времени. Эта опция определяет его формат. Когда значение равно 0, формат следующий: 'день' 'месяц' дд:мм:сс гггг напр. Пт ноя 25 15:11:55 2011 Когда значение равно 1, формат следующий: гггг.мм.дд чч:мм:сс.cccc напр. 2011.11.25 15:11:55.357
Узел DNS	dns	Это хост, на котором работает DNS-сервер. например lxplus086.cern.ch По умолчанию используется пустая строка, и в этом случае предполагается, что хост совпадает с хостом, на котором работает State Manager.
Макс. число изменений состояния в секунду	loopMaxChanges	Максимально допустимое количество изменений состояния в секунду
Фатальное число изменений состояния в секунду	loopMaxChangesFatal	Никакие изменения состояния в секунду сверх этого предела не считаются фатальными и приводят к выходу из State Manager.
Минимальная продолжи-тельность цикла (сек)	loopMinDuration	Минимальное время потенциальной петли, чтобы это считалось серьезным.

Изменение значения параметра во время работы State Manager

Для этого предусмотрена команда smiChangeOption. Он имеет следующий формат:

```
smiChangeOption domain-name option-id option-value
```

Например:

```
smiChangeOption RUNCONTROL d 3
```

Кроме того, предоставляется набор подпрограмм для облегчения этой задачи от пользовательского интерфейса до диспетчера состояний. См. описание SMIUIRTL.

Приложение 5. Пример кода SML

Файл **run_con.sml**:

```
object: AUTO_PILOT
    state: off
        action: ACTIVATE
            do RECOVER RUN
            do START_RUN RUN
            terminate_action/state=ON
    state: ON
    when (RUN not_in_state RUNNING) do X_ACTIVATE
        action: DISACTIVATE
            terminate_action/state=OFF
        action: X_ACTIVATE
            do RECOVER RUN
            do START_RUN RUN
            terminate_action/state=ON

object: RUN_TYPE
    state: TEST
        action: PHYSICS
            do LOG LOGGER
            if (LOGGER in_state LOGGING) then
                terminate_action/state=PHYSICS
            endif
            terminate_action/state=TEST
    state: PHYSICS
        action: TEST
            do NOLOG LOGGER
            if (LOGGER in_state NOT_LOGGING) then
                terminate_action/state=TEST
            endif
            terminate_action/state=PHYSICS

object: RUN
    parameters: int NUMBER_T = 0,
                int NUMBER_P = 0,
                RUN_MODE = "DEMO"
    state: STOPPED
        action: START_RUN
            set NUMBER_T = EVT_BUILDER<NUMBER_T>
            set NUMBER_P = EVT_BUILDER<NUMBER_P>
            if (not EVT_BUILDER in_state READY) then
                terminate_action/state=STOPPED
            endif
            if (RUN_TYPE in_state PHYSICS) then
                do LOG LOGGER
                if (not LOGGER in_state LOGGING) then
                    terminate_action/state=STOPPED
                endif
            endif
            if (LOGGER in_state LOGGING) then
                do X_OPEN_FILE LOGGER
            endif
            if (RUN_TYPE in_state PHYSICS) then
                set RUN_MODE = "PHYSICS"
                do START (TYPE="PHYSICS",NR=NUMBER_P) EVT_BUILDER
            else
                set RUN_MODE = "TEST"
                do START (TYPE="TEST",NR=NUMBER_T) EVT_BUILDER
            endif
            if (EVT_BUILDER in_state RUNNING) then
                terminate_action/state=RUNNING
            endif
            terminate_action/state=STOPPED
    state: RUNNING
    when (EVT_BUILDER in_state ERROR) do X_SET_ERROR
        action: STOP_RUN
            do STOP EVT_BUILDER
            if (LOGGER in_state WRITING) then
                do X_CLOSE_FILE LOGGER
            endif
            terminate_action/state=STOPPED
        action: X_SET_ERROR
            if (LOGGER in_state WRITING) then
                do X_CLOSE_FILE LOGGER
```

```
            endif
            terminate_action/state=ERROR
    state: ERROR
        action: RECOVER
            do RECOVER EVT_BUILDER
            terminate_action/state=STOPPED

object: LOGGER /associated
    state: DEAD /dead_state                 !color: DarkGray
    state: NOT_LOGGING                      !color: Aqua
        action: LOG
    state: LOGGING                          !color: Lime
        action: NOLOG
        action: X_OPEN_FILE
    state: WRITING                          !color: Lime
        action: X_CLOSE_FILE

object: EVT_BUILDER /associated
    parameters: int NUMBER_T, int NUMBER_P
    state: DEAD /dead_state                 !color: DarkGray
    state: READY                            !color: Aqua
        action: START(TYPE, int NR)
    state: RUNNING                          !color: Lime
        action: STOP
    state: ERROR                            !color: Red
        action: RECOVER
```

Приложение 6. Пример кода SML

Файл phc_col.sml:

```
class: TOP_cooling_top_CLASS
!panel: cooling_top.pnl
    state: OFF    !color: FwStateOKNotPhysics
        when ( any_in PHS_COL_DP_FWSETSTATES in_state ERROR )  move_to ERROR

        when ( any_in PHS_COL_DP_FWSETSTATES not_in_state OFF )  move_to STANDBY

        action: GO_STANDBY!visible: 1
            do GO_STANDBY all_in PHS_COL_DP_FWSETACTIONS
    state: STANDBY    !color: FwStateOKNotPhysics
        when ( any_in PHS_COL_DP_FWSETSTATES in_state ERROR )  move_to ERROR

        when ( all_in PHS_COL_DP_FWSETSTATES in_state OFF )  move_to OFF

        when ( any_in PHS_COL_DP_FWSETSTATES in_state CALIBRATING )  move_to CALIBRATING

        when ( any_in PHS_COL_DP_FWSETSTATES in_state WARMING )  move_to WARMING

        when (  ( any_in PHS_COL_DP_FWSETSTATES in_state COOLING ) or
( any_in PHS_COL_DP_FWSETSTATES in_state READY )  )  move_to COOLING

        action: SWITCH_OFF!visible: 1
            do SWITCH_OFF all_in PHS_COL_DP_FWSETACTIONS
        action: CALIBRATE  !visible: 1
            do CALIBRATE all_in PHS_COL_DP_FWSETACTIONS
        action: COOL  !visible: 1
            do COOL all_in PHS_COL_DP_FWSETACTIONS
        action: WARM  !visible: 1
            do WARM all_in PHS_COL_DP_FWSETACTIONS
    state: CALIBRATING    !color: FwStateAttention1
        when ( any_in PHS_COL_DP_FWSETSTATES in_state ERROR )  move_to ERROR

        when (  ( all_in PHS_COL_DP_FWSETSTATES in_state OFF ) or
( all_in PHS_COL_DP_FWSETSTATES in_state STANDBY ) or
( all_in PHS_COL_DP_FWSETSTATES in_state COOLING ) or
( all_in PHS_COL_DP_FWSETSTATES in_state WARMING ) or
( all_in PHS_COL_DP_FWSETSTATES in_state READY )  )  move_to STANDBY

        action: STOP  !visible: 1
            do STOP all_in PHS_COL_DP_FWSETACTIONS
    state: COOLING    !color: FwStateAttention1
        when ( any_in PHS_COL_DP_FWSETSTATES in_state ERROR )  move_to ERROR

        when (  ( any_in PHS_COL_DP_FWSETSTATES in_state WARMING ) or
( any_in PHS_COL_DP_FWSETSTATES in_state CALIBRATING ) or
( any_in PHS_COL_DP_FWSETSTATES in_state STANDBY ) or
( any_in PHS_COL_DP_FWSETSTATES in_state OFF )  )  move_to STANDBY

        when ( all_in PHS_COL_DP_FWSETSTATES in_state READY )  move_to READY

        action: GO_STANDBY!visible: 1
            do GO_STANDBY all_in PHS_COL_DP_FWSETACTIONS
        action: GO_READY  !visible: 1
            do GO_READY all_in PHS_COL_DP_FWSETACTIONS
    state: WARMING    !color: FwStateAttention1
        when ( any_in PHS_COL_DP_FWSETSTATES in_state ERROR )  move_to ERROR

        when (  ( all_in PHS_COL_DP_FWSETSTATES in_state STANDBY ) or
( all_in PHS_COL_DP_FWSETSTATES in_state OFF )  )  move_to STANDBY

        action: GO_STANDBY!visible: 1
            do GO_STANDBY all_in PHS_COL_DP_FWSETACTIONS
    state: READY  !color: FwStateOKPhysics
        when ( any_in PHS_COL_DP_FWSETSTATES in_state ERROR )  move_to ERROR

        when ( any_in PHS_COL_DP_FWSETSTATES not_in_state READY )  move_to WARMING

        action: WARM  !visible: 1
            do WARM all_in PHS_COL_DP_FWSETACTIONS
    state: ERROR  !color: FwStateAttention3
        when ( all_in PHS_COL_DP_FWSETSTATES not_in_state ERROR )  move_to STANDBY

        action: RESET !visible: 1
            do RESET all_in PHS_COL_DP_FWSETACTIONS
```

```
object: phs_col is_of_class TOP_cooling_top_CLASS

class: FwChildrenMode_CLASS
!panel: FwChildrenMode.pnl
    state: Complete   !color: _3DFace
        when ( any_in FWDEVMODE_FWSETSTATES in_state DISABLED )  move_to IncompleteDev
    state: Incomplete !color: FwStateAttention2
    state: IncompleteDev  !color: FwStateAttention1
        when (  ( all_in FWDEVMODE_FWSETSTATES not_in_state DISABLED )  ) move_to Complete
    state: IncompleteDead !color: FwStateAttention3

object: phs_col_FWCNM is_of_class FwChildrenMode_CLASS

class: FwMode_CLASS
!panel: FwMode.pnl
    state: Excluded   !color: FwStateOKNotPhysics
        action: Take(string OWNER = "",string EXCLUSIVE = "YES")  !visible: 1
            move_to InLocal
        action: Include(string OWNER = "",string EXCLUSIVE = "YES")   !visible: 0
            move_to Included
        action: Manual    !visible: 0
            move_to Manual
        action: Ignore    !visible: 0
            move_to Ignored
    state: Included   !color: FwStateOKPhysics
        action: Exclude(string OWNER = "")  !visible: 0
            move_to Excluded
        action: Manual(string OWNER = "")   !visible: 0
            move_to Manual
        action: Ignore(string OWNER = "")   !visible: 0
            move_to Ignored
        action: ExcludeAll(string OWNER = "")   !visible: 0
            move_to Excluded
        action: Include(string OWNER = "",string EXCLUSIVE = "YES")   !visible: 0
            move_to Included
        action: Free(string OWNER = "")!visible: 0
            move_to Included
        action: SetMode(string OWNER = "",string EXCLUSIVE = "YES")   !visible: 0
    state: InLocal    !color: FwStateOKNotPhysics
        action: Release(string OWNER = "")  !visible: 1
            move_to Excluded
        action: ReleaseAll(string OWNER = "")   !visible: 1
            move_to Excluded
        action: SetMode(string OWNER = "",string EXCLUSIVE = "YES")   !visible: 0
        action: Take(string OWNER = "",string EXCLUSIVE = "YES")  !visible: 1
            move_to InLocal
    state: Manual !color: FwStateOKNotPhysics
        action: Include(string OWNER = "",string EXCLUSIVE = "YES")   !visible: 0
            move_to Included
        action: Take(string OWNER = "")!visible: 1
            move_to InManual
        action: Exclude(string OWNER = "")  !visible: 0
            move_to Excluded
        action: Ignore    !visible: 0
            move_to Ignored
        action: Free(string OWNER = "")!visible: 0
            move_to Excluded
        action: ExcludeAll(string OWNER = "")   !visible: 0
            move_to Excluded
    state: InManual   !color: FwStateOKNotPhysics
        action: Release(string OWNER = "")  !visible: 1
            move_to Manual
        action: SetMode(string OWNER = "",string EXCLUSIVE = "YES")   !visible: 0
        action: ReleaseAll(string OWNER = "")   !visible: 0
            move_to Excluded
        action: SetInLocal!visible: 0
            move_to InLocal
    state: Ignored    !color: FwStateOKNotPhysics
        action: Include   !visible: 0
            move_to Included
        action: Exclude(string OWNER = "")  !visible: 0
            move_to Excluded
        action: Manual    !visible: 0
            move_to Manual
        action: SetMode(string OWNER = "",string EXCLUSIVE = "YES")   !visible: 0
        action: Free(string OWNER = "")!visible: 0
            move_to Included
        action: ExcludeAll(string OWNER = "")   !visible: 0
            move_to Excluded
```

```
object: phs_col_FWM is_of_class FwMode_CLASS

class: FwDevMode_FwDevMode_CLASS
    state: READY
        action: Disable(Device)
            remove &VAL_OF_Device from FWDEVMODE_FWSETSTATES
            remove &VAL_OF_Device from FWDEVMODE_FWSETACTIONS
            move_to READY
        action: Enable(Device)
            insert &VAL_OF_Device in FWDEVMODE_FWSETSTATES
            insert &VAL_OF_Device in FWDEVMODE_FWSETACTIONS
            move_to READY

object: FwDevMode_FWDM is_of_class FwDevMode_FwDevMode_CLASS


class: FwDevMode_CLASS/associated
!panel: FwDevMode.pnl
    state: ENABLED    !color: FwStateOKPhysics
    state: DISABLED   !color: FwStateAttention1

object: phs_col_FWDM is_of_class FwDevMode_CLASS

objectset: FWDEVMODE_FWSETSTATES is_of_class VOID {phs_col_FWDM }
objectset: FWDEVMODE_FWSETACTIONS is_of_class VOID {phs_col_FWDM }

class: phs_col_dp_FwDevMode_CLASS
    state: READY
        action: Disable(Device)
            remove &VAL_OF_Device from PHS_COL_DP_FWSETSTATES
            remove &VAL_OF_Device from PHS_COL_DP_FWSETACTIONS
            move_to READY
        action: Enable(Device)
            insert &VAL_OF_Device in PHS_COL_DP_FWSETSTATES
            insert &VAL_OF_Device in PHS_COL_DP_FWSETACTIONS
            move_to READY

object: phs_col_dp_FWDM is_of_class phs_col_dp_FwDevMode_CLASS


class: phs_col_dp_CLASS/associated
!panel: phs_col_dp.pnl
    state: OFF    !color: FwStateOKNotPhysics
        action: GO_STANDBY!visible: 1
    state: STANDBY    !color: FwStateOKNotPhysics
        action: CALIBRATE !visible: 1
        action: SWITCH_OFF!visible: 1
        action: COOL  !visible: 1
        action: WARM  !visible: 1
    state: CALIBRATING    !color: FwStateAttention1
        action: STOP  !visible: 1
    state: COOLING    !color: FwStateAttention1
        action: GO_STANDBY!visible: 1
        action: GO_READY  !visible: 1
    state: WARMING    !color: FwStateAttention1
        action: GO_STANDBY!visible: 1
    state: READY  !color: FwStateOKPhysics
        action: WARM  !visible: 1
        action: GO_STANDBY!visible: 1
    state: ERROR  !color: FwStateAttention3
        action: RESET !visible: 1

object: phs_col_dp is_of_class phs_col_dp_CLASS

objectset: PHS_COL_DP_FWSETSTATES is_of_class VOID {phs_col_dp }
objectset: PHS_COL_DP_FWSETACTIONS is_of_class VOID {phs_col_dp }


objectset: FWCHILDREN_FWSETACTIONS union {PHS_COL_DP_FWSETACTIONS } is_of_class VOID
objectset: FWCHILDREN_FWSETSTATES union {PHS_COL_DP_FWSETSTATES } is_of_class VOID
```

Приложение 7. Пример кода SML

Файл phs_gas.sml:

```
class: TOP_gas_CLASS
!panel: gas.pnl
    state: OFF    !color: FwStateOKNotPhysics
        when ( any_in PHS_GAS_DP_FWSETSTATES in_state ERROR )  move_to ERROR
        when ( all_in PHS_GAS_DP_FWSETSTATES in_state ON )  move_to ON
    state: ON!color: FwStateOKPhysics
        when ( any_in PHS_GAS_DP_FWSETSTATES in_state ERROR )  move_to ERROR
        when ( all_in PHS_GAS_DP_FWSETSTATES in_state OFF )  move_to OFF
    state: ERROR  !color: FwStateAttention3
        when (  ( all_in PHS_GAS_DP_FWSETSTATES in_state OFF ) or
       ( all_in PHS_GAS_DP_FWSETSTATES in_state ON )  )  move_to OFF

object: phs_gas is_of_class TOP_gas_CLASS

class: FwChildrenMode_CLASS
!panel: FwChildrenMode.pnl
    state: Complete   !color: _3DFace
        when ( any_in FWDEVMODE_FWSETSTATES in_state DISABLED )  move_to IncompleteDev
    state: Incomplete !color: FwStateAttention2
    state: IncompleteDev  !color: FwStateAttention1
        when (  ( all_in FWDEVMODE_FWSETSTATES not_in_state DISABLED )  ) move_to Complete
    state: IncompleteDead !color: FwStateAttention3

object: phs_gas_FWCNM is_of_class FwChildrenMode_CLASS

class: FwMode_CLASS
!panel: FwMode.pnl
    state: Excluded   !color: FwStateOKNotPhysics
        action: Take(string OWNER = "",string EXCLUSIVE = "YES")  !visible: 1
            move_to InLocal
        action: Include(string OWNER = "",string EXCLUSIVE = "YES")   !visible: 0
            move_to Included
        action: Manual    !visible: 0
            move_to Manual
        action: Ignore    !visible: 0
            move_to Ignored
    state: Included   !color: FwStateOKPhysics
        action: Exclude(string OWNER = "")  !visible: 0
            move_to Excluded
        action: Manual(string OWNER = "")   !visible: 0
            move_to Manual
        action: Ignore(string OWNER = "")   !visible: 0
            move_to Ignored
        action: ExcludeAll(string OWNER = "")   !visible: 0
            move_to Excluded
        action: Include(string OWNER = "",string EXCLUSIVE = "YES")   !visible: 0
            move_to Included
        action: Free(string OWNER = "")!visible: 0
            move_to Included
        action: SetMode(string OWNER = "",string EXCLUSIVE = "YES")   !visible: 0
    state: InLocal    !color: FwStateOKNotPhysics
        action: Release(string OWNER = "")  !visible: 1
            move_to Excluded
        action: ReleaseAll(string OWNER = "")   !visible: 1
            move_to Excluded
        action: SetMode(string OWNER = "",string EXCLUSIVE = "YES")   !visible: 0
        action: Take(string OWNER = "",string EXCLUSIVE = "YES")  !visible: 1
            move_to InLocal
    state: Manual !color: FwStateOKNotPhysics
        action: Include(string OWNER = "",string EXCLUSIVE = "YES")   !visible: 0
            move_to Included
        action: Take(string OWNER = "")!visible: 1
            move_to InManual
        action: Exclude(string OWNER = "")  !visible: 0
            move_to Excluded
        action: Ignore    !visible: 0
            move_to Ignored
        action: Free(string OWNER = "")!visible: 0
            move_to Excluded
        action: ExcludeAll(string OWNER = "")   !visible: 0
            move_to Excluded
```

```
    state: InManual   !color: FwStateOKNotPhysics
        action: Release(string OWNER = "")  !visible: 1
            move_to Manual
        action: SetMode(string OWNER = "",string EXCLUSIVE = "YES")   !visible: 0
        action: ReleaseAll(string OWNER = "")   !visible: 0
            move_to Excluded
        action: SetInLocal!visible: 0
            move_to InLocal
    state: Ignored    !color: FwStateOKNotPhysics
        action: Include   !visible: 0
            move_to Included
        action: Exclude(string OWNER = "")  !visible: 0
            move_to Excluded
        action: Manual    !visible: 0
            move_to Manual
        action: SetMode(string OWNER = "",string EXCLUSIVE = "YES")   !visible: 0
        action: Free(string OWNER = "")!visible: 0
            move_to Included
        action: ExcludeAll(string OWNER = "")   !visible: 0
            move_to Excluded

object: phs_gas_FWM is_of_class FwMode_CLASS

class: FwDevMode_FwDevMode_CLASS
    state: READY
        action: Disable(Device)
            remove &VAL_OF_Device from FWDEVMODE_FWSETSTATES
            remove &VAL_OF_Device from FWDEVMODE_FWSETACTIONS
            move_to READY
        action: Enable(Device)
            insert &VAL_OF_Device in FWDEVMODE_FWSETSTATES
            insert &VAL_OF_Device in FWDEVMODE_FWSETACTIONS
            move_to READY

object: FwDevMode_FWDM is_of_class FwDevMode_FwDevMode_CLASS


class: FwDevMode_CLASS/associated
!panel: FwDevMode.pnl
    state: ENABLED    !color: FwStateOKPhysics
    state: DISABLED   !color: FwStateAttention1

object: phs_gas_FWDM is_of_class FwDevMode_CLASS

objectset: FWDEVMODE_FWSETSTATES is_of_class VOID {phs_gas_FWDM }
objectset: FWDEVMODE_FWSETACTIONS is_of_class VOID {phs_gas_FWDM }

class: phs_gas_dp_FwDevMode_CLASS
    state: READY
        action: Disable(Device)
            remove &VAL_OF_Device from PHS_GAS_DP_FWSETSTATES
            remove &VAL_OF_Device from PHS_GAS_DP_FWSETACTIONS
            move_to READY
        action: Enable(Device)
            insert &VAL_OF_Device in PHS_GAS_DP_FWSETSTATES
            insert &VAL_OF_Device in PHS_GAS_DP_FWSETACTIONS
            move_to READY

object: phs_gas_dp_FWDM is_of_class phs_gas_dp_FwDevMode_CLASS


class: phs_gas_dp_CLASS/associated
!panel: phs_gas_dp.pnl
    state: OFF    !color: FwStateOKNotPhysics
    state: ON!color: FwStateOKPhysics
    state: ERROR  !color: FwStateAttention3

object: phs_gas_dp is_of_class phs_gas_dp_CLASS

objectset: PHS_GAS_DP_FWSETSTATES is_of_class VOID {phs_gas_dp }
objectset: PHS_GAS_DP_FWSETACTIONS is_of_class VOID {phs_gas_dp }


objectset: FWCHILDREN_FWSETACTIONS union {PHS_GAS_DP_FWSETACTIONS } is_of_class VOID
objectset: FWCHILDREN_FWSETSTATES union {PHS_GAS_DP_FWSETSTATES } is_of_class VOID
```

Приложение 8. Пример кода SML

Файл **psh_led.sml**:

```
class: TOP_monitoring_top_CLASS
!panel: monitoring_top.pnl
    state: OFF    !color: FwStateOKNotPhysics
        when ( any_in FWCHILDREN_FWSETSTATES in_state ERROR )  move_to ERROR

        when ( all_in VME_FWSETSTATES in_state READY )  move_to NOT_READY
        action: SWITCH_ON !visible: 1
            do CONFIGURE all_in VME_FWSETACTIONS
    state: NOT_READY  !color: FwStateOKNotPhysics
        when ( any_in FWCHILDREN_FWSETSTATES in_state ERROR )  move_to ERROR

        when (  ( all_in VME_FWSETSTATES in_state NOT_READY )  )  move_to OFF
        when (  ( all_in VME_FWSETSTATES in_state READY ) and
( any_in LED_FWSETSTATES in_state CONFIGURING ) or
( any_in LED_FWSETSTATES in_state DEACTIVATED ) or
( all_in LED_FWSETSTATES in_state READY )  )  move_to CONFIGURING
        action: SWITCH_OFF!visible: 1
            do GO_OFF all_in LED_FWSETACTIONS
            do RESET all_in VME_FWSETACTIONS
            do SWITCH_OFF all_in LED_FWSETACTIONS
        action: GO_READY  !visible: 1
            do CONFIGURE all_in VME_FWSETACTIONS
            do GO_READY all_in LED_FWSETACTIONS
        action: DEACTIVATE!visible: 1
            do DEACTIVATE all_in LED_FWSETACTIONS
    state: CONFIGURING    !color: FwStateAttention1
        when ( any_in FWCHILDREN_FWSETSTATES in_state ERROR )  move_to ERROR

        when (  ( all_in LED_FWSETSTATES not_in_state CONFIGURING ) and
(  ( any_in VME_FWSETSTATES in_state NOT_READY ) or
( any_in VME_FWSETSTATES in_state NO_CONTROL ) or
( any_in LED_FWSETSTATES in_state NOT_READY )  )  )  move_to NOT_READY
        when (  ( all_in VME_FWSETSTATES in_state READY ) and
( all_in LED_FWSETSTATES in_state READY )  )  move_to READY
        when (  ( all_in VME_FWSETSTATES in_state READY ) and
( all_in LED_FWSETSTATES in_state DEACTIVATED )  )  move_to DEACTIVATED
        action: DEACTIVATE!visible: 0
            do DEACTIVATE all_in LED_FWSETACTIONS
    state: READY  !color: FwStateOKPhysics
        when ( any_in FWCHILDREN_FWSETSTATES in_state ERROR )  move_to ERROR

        when (  ( any_in VME_FWSETSTATES in_state NOT_READY ) or
( any_in LED_FWSETSTATES not_in_state READY )  )  move_to CONFIGURING
        action: GO_OFF    !visible: 1
            do GO_OFF all_in LED_FWSETACTIONS
        action: DEACTIVATE!visible: 1
            do DEACTIVATE all_in LED_FWSETACTIONS
    state: DEACTIVATED    !color: FwStateOKPhysics
        when ( any_in FWCHILDREN_FWSETSTATES in_state ERROR )  move_to ERROR

        when (  ( any_in VME_FWSETSTATES in_state NOT_READY ) or
( any_in LED_FWSETSTATES not_in_state DEACTIVATED )  )  move_to CONFIGURING
        action: ACTIVATE  !visible: 1
            do ACTIVATE all_in LED_FWSETACTIONS
    state: ERROR  !color: FwStateAttention3
        when ( all_in FWCHILDREN_FWSETSTATES not_in_state ERROR )  move_to NOT_READY

        action: RESET !visible: 1
            do RESET all_in LED_FWSETACTIONS
            do RECOVER all_in VME_FWSETACTIONS

object: phs_led is_of_class TOP_monitoring_top_CLASS

class: FwChildrenMode_CLASS
!panel: FwChildrenMode.pnl
    state: Complete   !color: _3DFace
        when ( any_in FWCHILDRENMODE_FWSETSTATES in_state IncompleteDead ) move_to IncompleteDead
                when  (    (  any_in  FWCHILDRENMODE_FWSETSTATES  in_state  DEAD  )  and  (  any_in
FWCHILDMODE_FWSETSTATES in_state MANUAL )  ) move_to IncompleteDead
        when ( any_in FWCHILDRENMODE_FWSETSTATES in_state Incomplete )  move_to Incomplete
```

```
         when ( any_in FWCHILDMODE_FWSETSTATES not_in_state {Included,ExcludedPerm,LockedOutPerm} )
move_to Incomplete
        when ( any_in FWCHILDRENMODE_FWSETSTATES in_state IncompleteDev )  move_to IncompleteDev
    state: Incomplete !color: FwStateAttention2
        when ( any_in FWCHILDRENMODE_FWSETSTATES in_state IncompleteDead ) move_to IncompleteDead
                when (    (  any_in  FWCHILDRENMODE_FWSETSTATES  in_state  DEAD  )  and  (  any_in
FWCHILDMODE_FWSETSTATES in_state MANUAL )  ) move_to IncompleteDead
         when (  ( all_in FWCHILDMODE_FWSETSTATES in_state {Included,ExcludedPerm,LockedOutPerm} )
and
       ( all_in FWCHILDRENMODE_FWSETSTATES not_in_state Incomplete )  )  move_to Complete
    state: IncompleteDev  !color: FwStateAttention1
        when ( any_in FWCHILDRENMODE_FWSETSTATES in_state IncompleteDead ) move_to IncompleteDead
                when (    (  any_in  FWCHILDRENMODE_FWSETSTATES  in_state  DEAD  )  and  (  any_in
FWCHILDMODE_FWSETSTATES in_state MANUAL )  ) move_to IncompleteDead
                             when    (       (    any_in    FWCHILDMODE_FWSETSTATES    not_in_state
{Included,ExcludedPerm,LockedOutPerm} ) or
       ( any_in FWCHILDRENMODE_FWSETSTATES in_state Incomplete )  )  move_to Incomplete
        when (  ( all_in FWCHILDRENMODE_FWSETSTATES not_in_state IncompleteDev )  ) move_to Complete
    state: IncompleteDead !color: FwStateAttention3
            when (   (   (  all_in  FWCHILDRENMODE_FWSETSTATES  not_in_state  DEAD  ) or  (  all_in
FWCHILDMODE_FWSETSTATES  not_in_state  MANUAL  )    )  and  (  all_in  FWCHILDRENMODE_FWSETSTATES
not_in_state IncompleteDead )  )  move_to Complete

object: phs_led_FWCNM is_of_class FwChildrenMode_CLASS

class: ASS_FwChildrenMode_CLASS/associated
!panel: FwChildrenMode.pnl
    state: Complete   !color: _3DFace
    state: Incomplete !color: FwStateAttention2
    state: IncompleteDev  !color: FwStateAttention1
    state: IncompleteDead !color: FwStateAttention3

object: phs_led_sys::phs_led_sys_FWCNM is_of_class ASS_FwChildrenMode_CLASS

object: phs_vme::phs_vme_FWCNM is_of_class ASS_FwChildrenMode_CLASS

objectset: FWCHILDRENMODE_FWSETSTATES is_of_class VOID

class: FwMode_CLASS
!panel: FwMode.pnl
    state: Excluded   !color: FwStateOKNotPhysics
        action: Take(string OWNER = "",string EXCLUSIVE = "YES")  !visible: 1
            do Include(OWNER=OWNER,EXCLUSIVE=EXCLUSIVE) all_in FWCHILDMODE_FWSETACTIONS
            move_to InLocal
        action: Include(string OWNER = "",string EXCLUSIVE = "YES")   !visible: 0
            do Include(OWNER=OWNER,EXCLUSIVE=EXCLUSIVE) all_in FWCHILDMODE_FWSETACTIONS
            move_to Included
        action: Manual    !visible: 0
            move_to Manual
        action: Ignore    !visible: 0
            move_to Ignored
    state: Included   !color: FwStateOKPhysics
        action: Exclude(string OWNER = "")  !visible: 0
            do Free(OWNER=OWNER) all_in FWCHILDMODE_FWSETACTIONS
            move_to Excluded
        action: Manual(string OWNER = "")   !visible: 0
            do Free(OWNER=OWNER) all_in FWCHILDMODE_FWSETACTIONS
            move_to Manual
        action: Ignore(string OWNER = "")   !visible: 0
            move_to Ignored
        action: ExcludeAll(string OWNER = "")   !visible: 0
            do ExcludeAll(OWNER=OWNER) all_in FWCHILDMODE_FWSETACTIONS
            move_to Excluded
        action: Include(string OWNER = "",string EXCLUSIVE = "YES")   !visible: 0
            do Include(OWNER=OWNER,EXCLUSIVE=EXCLUSIVE) all_in FWCHILDMODE_FWSETACTIONS
            move_to Included
        action: Free(string OWNER = "")!visible: 0
            do Free(OWNER=OWNER) all_in FWCHILDMODE_FWSETACTIONS
            move_to Included
        action: SetMode(string OWNER = "",string EXCLUSIVE = "YES")   !visible: 0
            do SetMode(OWNER=OWNER,EXCLUSIVE=EXCLUSIVE) all_in FWCHILDMODE_FWSETACTIONS
    state: InLocal    !color: FwStateOKNotPhysics
        action: Release(string OWNER = "")  !visible: 1
            do Free(OWNER=OWNER) all_in FWCHILDMODE_FWSETACTIONS
            move_to Excluded
        action: ReleaseAll(string OWNER = "")   !visible: 1
            do ExcludeAll(OWNER=OWNER) all_in FWCHILDMODE_FWSETACTIONS
            move_to Excluded
        action: SetMode(string OWNER = "",string EXCLUSIVE = "YES")   !visible: 0
            do SetMode(OWNER=OWNER,EXCLUSIVE=EXCLUSIVE) all_in FWCHILDMODE_FWSETACTIONS
        action: Take(string OWNER = "",string EXCLUSIVE = "YES")  !visible: 1
```

```
            do Include(OWNER=OWNER,EXCLUSIVE=EXCLUSIVE) all_in FWCHILDMODE_FWSETACTIONS
            move_to InLocal
    state: Manual !color: FwStateOKNotPhysics
        action: Include(string OWNER = "",string EXCLUSIVE = "YES")   !visible: 0
            do Include(OWNER=OWNER,EXCLUSIVE=EXCLUSIVE) all_in FWCHILDMODE_FWSETACTIONS
            move_to Included
        action: Take(string OWNER = "")!visible: 1
            do Include(OWNER=OWNER) all_in FWCHILDMODE_FWSETACTIONS
            move_to InManual
        action: Exclude(string OWNER = "")  !visible: 0
            do Exclude(OWNER=OWNER) all_in FWCHILDMODE_FWSETACTIONS
            move_to Excluded
        action: Ignore    !visible: 0
            move_to Ignored
        action: Free(string OWNER = "")!visible: 0
            do Free(OWNER=OWNER) all_in FWCHILDMODE_FWSETACTIONS
            move_to Excluded
        action: ExcludeAll(string OWNER = "")   !visible: 0
            do ExcludeAll(OWNER=OWNER) all_in FWCHILDMODE_FWSETACTIONS
            move_to Excluded
    state: InManual   !color: FwStateOKNotPhysics
        action: Release(string OWNER = "")  !visible: 1
            do Free(OWNER=OWNER) all_in FWCHILDMODE_FWSETACTIONS
            move_to Manual
        action: SetMode(string OWNER = "",string EXCLUSIVE = "YES")   !visible: 0
            do SetMode(OWNER=OWNER,EXCLUSIVE=EXCLUSIVE) all_in FWCHILDMODE_FWSETACTIONS
        action: ReleaseAll(string OWNER = "")   !visible: 0
            do ExcludeAll(OWNER=OWNER) all_in FWCHILDMODE_FWSETACTIONS
            move_to Excluded
        action: SetInLocal!visible: 0
            move_to InLocal
    state: Ignored    !color: FwStateOKNotPhysics
        action: Include   !visible: 0
            move_to Included
        action: Exclude(string OWNER = "")  !visible: 0
            do Exclude(OWNER=OWNER) all_in FWCHILDMODE_FWSETACTIONS
            move_to Excluded
        action: Manual    !visible: 0
            move_to Manual
        action: SetMode(string OWNER = "",string EXCLUSIVE = "YES")   !visible: 0
            do SetMode(OWNER=OWNER,EXCLUSIVE=EXCLUSIVE) all_in FWCHILDMODE_FWSETACTIONS
        action: Free(string OWNER = "")!visible: 0
            do Free(OWNER=OWNER) all_in FWCHILDMODE_FWSETACTIONS
            move_to Included
        action: ExcludeAll(string OWNER = "")   !visible: 0
            do ExcludeAll(OWNER=OWNER) all_in FWCHILDMODE_FWSETACTIONS
            move_to Excluded

object: phs_led_FWM is_of_class FwMode_CLASS

class: ASS_FwMode_CLASS/associated
!panel: FwMode.pnl
    state: Excluded   !color: FwStateOKNotPhysics
        action: Take(string OWNER = "",string EXCLUSIVE = "YES")  !visible: 1
        action: Include(string OWNER = "",string EXCLUSIVE = "YES")   !visible: 0
        action: Manual    !visible: 0
        action: Ignore    !visible: 0
    state: Included   !color: FwStateOKPhysics
        action: Exclude(string OWNER = "")  !visible: 0
        action: Manual(string OWNER = "")   !visible: 0
        action: Ignore(string OWNER = "")   !visible: 0
        action: ExcludeAll(string OWNER = "")   !visible: 0
        action: Include(string OWNER = "",string EXCLUSIVE = "YES")   !visible: 0
        action: Free(string OWNER = "")!visible: 0
        action: SetMode(string OWNER = "",string EXCLUSIVE = "YES")   !visible: 0
    state: InLocal    !color: FwStateOKNotPhysics
        action: Release(string OWNER = "")  !visible: 1
        action: ReleaseAll(string OWNER = "")   !visible: 1
        action: SetMode(string OWNER = "",string EXCLUSIVE = "YES")   !visible: 0
        action: Take(string OWNER = "",string EXCLUSIVE = "YES")  !visible: 1
    state: Manual !color: FwStateOKNotPhysics
        action: Include(string OWNER = "",string EXCLUSIVE = "YES")   !visible: 0
        action: Take(string OWNER = "")!visible: 1
        action: Exclude(string OWNER = "")  !visible: 0
        action: Ignore    !visible: 0
        action: Free(string OWNER = "")!visible: 0
        action: ExcludeAll(string OWNER = "")   !visible: 0
    state: InManual   !color: FwStateOKNotPhysics
        action: Release(string OWNER = "")  !visible: 1
        action: SetMode(string OWNER = "",string EXCLUSIVE = "YES")   !visible: 0
        action: ReleaseAll(string OWNER = "")   !visible: 0
```

```
        action: SetInLocal!visible: 0
    state: Ignored    !color: FwStateOKNotPhysics
        action: Include   !visible: 0
        action: Exclude(string OWNER = "")  !visible: 0
        action: Manual    !visible: 0
        action: SetMode(string OWNER = "",string EXCLUSIVE = "YES")   !visible: 0
        action: Free(string OWNER = "")!visible: 0
        action: ExcludeAll(string OWNER = "")   !visible: 0

object: phs_led_sys::phs_led_sys_FWM is_of_class ASS_FwMode_CLASS

object: phs_vme::phs_vme_FWM is_of_class ASS_FwMode_CLASS

class: phs_led_sys_FwChildMode_CLASS

!panel: FwChildMode.pnl
    state: Excluded   !color: FwStateOKNotPhysics
        action: Include(string OWNER = "",string EXCLUSIVE = "YES")   !visible: 1
            if ( phs_led_sys::phs_led_sys_FWM in_state Dead ) then
              move_to Manual
            endif
            if ( phs_led_sys::phs_led_sys_FWM not_in_state {Excluded, Manual} ) then
            !    else
                    move_to Excluded
            !    endif
            else
                do Include(OWNER=OWNER,EXCLUSIVE=EXCLUSIVE) phs_led_sys::phs_led_sys_FWM
                insert phs_led_sys::phs_led_sys in LED_FWSETSTATES
                insert phs_led_sys::phs_led_sys in LED_FWSETACTIONS
                insert phs_led_sys::phs_led_sys_FWCNM in FWCHILDRENMODE_FWSETSTATES
            endif
            move_to Included
        action: Manual    !visible: 0
            do Manual phs_led_sys::phs_led_sys_FWM
            insert phs_led_sys::phs_led_sys in LED_FWSETSTATES
            remove phs_led_sys::phs_led_sys from LED_FWSETACTIONS
            move_to Manual
        action: Ignore    !visible: 0
            do Ignore phs_led_sys::phs_led_sys_FWM
            insert phs_led_sys::phs_led_sys in LED_FWSETACTIONS
            remove phs_led_sys::phs_led_sys from LED_FWSETSTATES
            move_to Ignored
        action: LockOut   !visible: 1
            move_to LockedOut
        action: Exclude(string OWNER = "")  !visible: 1
            do Exclude(OWNER=OWNER) phs_led_sys::phs_led_sys_FWM
            remove phs_led_sys::phs_led_sys from LED_FWSETSTATES
            remove phs_led_sys::phs_led_sys from LED_FWSETACTIONS
            remove phs_led_sys::phs_led_sys_FWCNM from FWCHILDRENMODE_FWSETSTATES
            move_to Excluded
        action: ExcludePerm(string OWNER = "")  !visible: 0
            move_to ExcludedPerm
        action: Exclude&LockOut(string OWNER = "")  !visible: 0
            move_to LockedOut
    state: Included   !color: FwStateOKPhysics
        when ( phs_led_sys::phs_led_sys_FWM in_state Excluded )  do Exclude
        when ( phs_led_sys::phs_led_sys_FWM in_state Ignored )  move_to IGNORED

        when ( phs_led_sys::phs_led_sys_FWM in_state Manual )  move_to MANUAL

        when ( phs_led_sys::phs_led_sys_FWM in_state Dead )  do Manual

        action: Exclude(string OWNER = "")  !visible: 1
            if ( phs_led_sys::phs_led_sys_FWM not_in_state Included ) then
                if ( phs_led_sys::phs_led_sys_FWM in_state InManual ) then
                    do Release(OWNER=OWNER) phs_led_sys::phs_led_sys_FWM
                    remove phs_led_sys::phs_led_sys from LED_FWSETSTATES
                    remove phs_led_sys::phs_led_sys from LED_FWSETACTIONS
                    remove phs_led_sys::phs_led_sys_FWCNM from FWCHILDRENMODE_FWSETSTATES
                else
                        do Exclude(OWNER=OWNER) phs_led_sys::phs_led_sys_FWM
                        remove phs_led_sys::phs_led_sys from LED_FWSETSTATES
                        remove phs_led_sys::phs_led_sys from LED_FWSETACTIONS
                        remove phs_led_sys::phs_led_sys_FWCNM from FWCHILDRENMODE_FWSETSTATES
            !       else
            !            move_to Included
            !        endif
                endif
            else
                do Exclude(OWNER=OWNER) phs_led_sys::phs_led_sys_FWM
                remove phs_led_sys::phs_led_sys from LED_FWSETSTATES
```

```
            remove phs_led_sys::phs_led_sys from LED_FWSETACTIONS
            remove phs_led_sys::phs_led_sys_FWCNM from FWCHILDRENMODE_FWSETSTATES
        endif
        move_to Excluded
    action: Manual(string OWNER = "")   !visible: 1
        do Manual(OWNER=OWNER) phs_led_sys::phs_led_sys_FWM
        insert phs_led_sys::phs_led_sys in LED_FWSETSTATES
        remove phs_led_sys::phs_led_sys from LED_FWSETACTIONS
        move_to Manual
    action: Ignore(string OWNER = "")   !visible: 1
        do Ignore(OWNER=OWNER) phs_led_sys::phs_led_sys_FWM
        insert phs_led_sys::phs_led_sys in LED_FWSETACTIONS
        remove phs_led_sys::phs_led_sys from LED_FWSETSTATES
        move_to Ignored
    action: ExcludeAll(string OWNER = "")   !visible: 1
        if ( phs_led_sys::phs_led_sys_FWM not_in_state {Included,Ignored,Manual} ) then
            if ( phs_led_sys::phs_led_sys_FWM in_state InManual ) then
                do ReleaseAll(OWNER=OWNER) phs_led_sys::phs_led_sys_FWM
                remove phs_led_sys::phs_led_sys from LED_FWSETSTATES
                remove phs_led_sys::phs_led_sys from LED_FWSETACTIONS
                remove phs_led_sys::phs_led_sys_FWCNM from FWCHILDRENMODE_FWSETSTATES
            else
                move_to Included
            endif
        else
            do ExcludeAll(OWNER=OWNER) phs_led_sys::phs_led_sys_FWM
            remove phs_led_sys::phs_led_sys from LED_FWSETSTATES
            remove phs_led_sys::phs_led_sys from LED_FWSETACTIONS
            remove phs_led_sys::phs_led_sys_FWCNM from FWCHILDRENMODE_FWSETSTATES
        endif
        move_to Excluded
    action: Include(string OWNER = "",string EXCLUSIVE = "YES")   !visible: 1
        do Include(OWNER=OWNER,EXCLUSIVE=EXCLUSIVE) phs_led_sys::phs_led_sys_FWM
        insert phs_led_sys::phs_led_sys in LED_FWSETSTATES
        insert phs_led_sys::phs_led_sys in LED_FWSETACTIONS
        insert phs_led_sys::phs_led_sys_FWCNM in FWCHILDRENMODE_FWSETSTATES
        move_to Included
    action: Free(string OWNER = "")!visible: 0
        do Free(OWNER=OWNER) phs_led_sys::phs_led_sys_FWM
        move_to Included
    action: SetMode(string OWNER = "",string EXCLUSIVE = "YES")   !visible: 0
        do SetMode(OWNER=OWNER,EXCLUSIVE=EXCLUSIVE) phs_led_sys::phs_led_sys_FWM
    action: ExcludePerm(string OWNER = "")  !visible: 0
        if ( phs_led_sys::phs_led_sys_FWM not_in_state Included ) then
            if ( phs_led_sys::phs_led_sys_FWM in_state InManual ) then
                do Release(OWNER=OWNER) phs_led_sys::phs_led_sys_FWM
                remove phs_led_sys::phs_led_sys from LED_FWSETSTATES
                remove phs_led_sys::phs_led_sys from LED_FWSETACTIONS
                remove phs_led_sys::phs_led_sys_FWCNM from FWCHILDRENMODE_FWSETSTATES
            else
                move_to Included
            endif
        else
            do Exclude(OWNER=OWNER) phs_led_sys::phs_led_sys_FWM
            remove phs_led_sys::phs_led_sys from LED_FWSETSTATES
            remove phs_led_sys::phs_led_sys from LED_FWSETACTIONS
            remove phs_led_sys::phs_led_sys_FWCNM from FWCHILDRENMODE_FWSETSTATES
        endif
        move_to ExcludedPerm
    action: Exclude&LockOut(string OWNER = "")  !visible: 1
        if ( phs_led_sys::phs_led_sys_FWM not_in_state Included ) then
            if ( phs_led_sys::phs_led_sys_FWM in_state InManual ) then
                do Release(OWNER=OWNER) phs_led_sys::phs_led_sys_FWM
                remove phs_led_sys::phs_led_sys from LED_FWSETSTATES
                remove phs_led_sys::phs_led_sys from LED_FWSETACTIONS
                remove phs_led_sys::phs_led_sys_FWCNM from FWCHILDRENMODE_FWSETSTATES
            else
                if ( phs_led_sys::phs_led_sys_FWM in_state Dead ) then
                    do Exclude(OWNER=OWNER) phs_led_sys::phs_led_sys_FWM
                    remove phs_led_sys::phs_led_sys from LED_FWSETSTATES
                    remove phs_led_sys::phs_led_sys from LED_FWSETACTIONS
                    remove phs_led_sys::phs_led_sys_FWCNM from FWCHILDRENMODE_FWSETSTATES
               else
                    move_to Included
                endif
            endif
        else
            do Exclude(OWNER=OWNER) phs_led_sys::phs_led_sys_FWM
            remove phs_led_sys::phs_led_sys from LED_FWSETSTATES
            remove phs_led_sys::phs_led_sys from LED_FWSETACTIONS
            remove phs_led_sys::phs_led_sys_FWCNM from FWCHILDRENMODE_FWSETSTATES
```

```
        endif
        move_to LockedOut
  state: Manual !color: FwStateOKNotPhysics
      when ( phs_led_sys::phs_led_sys_FWM in_state Included )  move_to EXCLUDED
      action: Include(string OWNER = "",string EXCLUSIVE = "YES")   !visible: 1
        if ( phs_led_sys::phs_led_sys_FWM in_state Dead ) then
          move_to Manual
        endif
        if ( phs_led_sys::phs_led_sys_FWM not_in_state InManual ) then
          do Include(OWNER=OWNER,EXCLUSIVE=EXCLUSIVE) phs_led_sys::phs_led_sys_FWM
          insert phs_led_sys::phs_led_sys in LED_FWSETSTATES
          insert phs_led_sys::phs_led_sys in LED_FWSETACTIONS
          insert phs_led_sys::phs_led_sys_FWCNM in FWCHILDRENMODE_FWSETSTATES
        endif
          if ( phs_led_sys::phs_led_sys_FWM in_state Included ) then
            move_to Included
          endif
        move_to Manual
      action: Exclude(string OWNER = "")  !visible: 1
              do Exclude(OWNER=OWNER) phs_led_sys::phs_led_sys_FWM
              remove phs_led_sys::phs_led_sys from LED_FWSETSTATES
              remove phs_led_sys::phs_led_sys from LED_FWSETACTIONS
              remove phs_led_sys::phs_led_sys_FWCNM from FWCHILDRENMODE_FWSETSTATES
        !    move_to Excluded
        !endif
        !    else
        !    endif
        !else
        !endif
        !    move_to Excluded
        !endif
        !move_to Manual
        if ( phs_led_sys::phs_led_sys_FWM in_state InManual ) then
             do SetInLocal phs_led_sys::phs_led_sys_FWM
        endif
        move_to Excluded
      action: Ignore    !visible: 0
        do Ignore phs_led_sys::phs_led_sys_FWM
        insert phs_led_sys::phs_led_sys in LED_FWSETACTIONS
        remove phs_led_sys::phs_led_sys from LED_FWSETSTATES
        move_to Ignored
      action: Free(string OWNER = "")!visible: 0
        do Free(OWNER=OWNER) phs_led_sys::phs_led_sys_FWM
        !move_to Manual
        if ( phs_led_sys::phs_led_sys_FWM in_state InManual ) then
             do SetInLocal phs_led_sys::phs_led_sys_FWM
        endif
        move_to Excluded
      action: ExcludeAll(string OWNER = "")   !visible: 1
        !    else
        !        move_to Included
        !    endif
        !else
        !endif
          do ExcludeAll(OWNER=OWNER) phs_led_sys::phs_led_sys_FWM
          remove phs_led_sys::phs_led_sys from LED_FWSETSTATES
          remove phs_led_sys::phs_led_sys from LED_FWSETACTIONS
          remove phs_led_sys::phs_led_sys_FWCNM from FWCHILDRENMODE_FWSETSTATES
        !endif
        !move_to Manual
        if ( phs_led_sys::phs_led_sys_FWM in_state InManual ) then
             do SetInLocal phs_led_sys::phs_led_sys_FWM
        endif
        move_to Excluded
      action: Manual    !visible: 0
        do Manual phs_led_sys::phs_led_sys_FWM
        insert phs_led_sys::phs_led_sys in LED_FWSETSTATES
        remove phs_led_sys::phs_led_sys from LED_FWSETACTIONS
        move_to Manual
      action: Exclude&LockOut(string OWNER = "")  !visible: 1
              do Exclude(OWNER=OWNER) phs_led_sys::phs_led_sys_FWM
              remove phs_led_sys::phs_led_sys from LED_FWSETSTATES
              remove phs_led_sys::phs_led_sys from LED_FWSETACTIONS
              remove phs_led_sys::phs_led_sys_FWCNM from FWCHILDRENMODE_FWSETSTATES
        !    move_to Excluded
        !endif
        !    else
        !    endif
        !else
        !endif
        !    move_to Excluded
```

```
        !endif
        !move_to Manual
        if ( phs_led_sys::phs_led_sys_FWM in_state InManual ) then
             do SetInLocal phs_led_sys::phs_led_sys_FWM
        endif
        move_to LockedOut
    state: Ignored    !color: FwStateOKNotPhysics
        when ( phs_led_sys::phs_led_sys_FWM in_state Included )  move_to INCLUDED

        when ( phs_led_sys::phs_led_sys_FWM in_state Excluded ) move_to EXCLUDED

        when ( phs_led_sys::phs_led_sys_FWM in_state Dead )  do Exclude

        action: Include(string OWNER = "",string EXCLUSIVE = "YES")   !visible: 1
            do Include(OWNER=OWNER,EXCLUSIVE=EXCLUSIVE) phs_led_sys::phs_led_sys_FWM
            insert phs_led_sys::phs_led_sys in LED_FWSETSTATES
            insert phs_led_sys::phs_led_sys in LED_FWSETACTIONS
            insert phs_led_sys::phs_led_sys_FWCNM in FWCHILDRENMODE_FWSETSTATES
            move_to Included
        action: Exclude(string OWNER = "")  !visible: 1
            if ( phs_led_sys::phs_led_sys_FWM not_in_state Included ) then
                if ( phs_led_sys::phs_led_sys_FWM in_state InManual ) then
                    do Release(OWNER=OWNER) phs_led_sys::phs_led_sys_FWM
                    remove phs_led_sys::phs_led_sys from LED_FWSETSTATES
                    remove phs_led_sys::phs_led_sys from LED_FWSETACTIONS
                    remove phs_led_sys::phs_led_sys_FWCNM from FWCHILDRENMODE_FWSETSTATES
                else
                    do Exclude(OWNER=OWNER) phs_led_sys::phs_led_sys_FWM
                    remove phs_led_sys::phs_led_sys from LED_FWSETSTATES
                    remove phs_led_sys::phs_led_sys from LED_FWSETACTIONS
                    remove phs_led_sys::phs_led_sys_FWCNM from FWCHILDRENMODE_FWSETSTATES
                endif
            else
                do Exclude(OWNER=OWNER) phs_led_sys::phs_led_sys_FWM
                remove phs_led_sys::phs_led_sys from LED_FWSETSTATES
                remove phs_led_sys::phs_led_sys from LED_FWSETACTIONS
                remove phs_led_sys::phs_led_sys_FWCNM from FWCHILDRENMODE_FWSETSTATES
            endif
            move_to Excluded
        action: Manual(string OWNER = "")   !visible: 0
            do Manual(OWNER=OWNER) phs_led_sys::phs_led_sys_FWM
            insert phs_led_sys::phs_led_sys in LED_FWSETSTATES
            remove phs_led_sys::phs_led_sys from LED_FWSETACTIONS
            move_to Manual
        action: SetMode(string OWNER = "",string EXCLUSIVE = "YES")   !visible: 0
            do SetMode(OWNER=OWNER,EXCLUSIVE=EXCLUSIVE) phs_led_sys::phs_led_sys_FWM
        action: Free(string OWNER = "")!visible: 0
            do Free(OWNER=OWNER) phs_led_sys::phs_led_sys_FWM
            move_to Included
        action: ExcludeAll(string OWNER = "")   !visible: 1
            if ( phs_led_sys::phs_led_sys_FWM not_in_state {Included,Ignored,Manual} ) then
                if ( phs_led_sys::phs_led_sys_FWM in_state InManual ) then
                    do ReleaseAll(OWNER=OWNER) phs_led_sys::phs_led_sys_FWM
                    remove phs_led_sys::phs_led_sys from LED_FWSETSTATES
                    remove phs_led_sys::phs_led_sys from LED_FWSETACTIONS
                    remove phs_led_sys::phs_led_sys_FWCNM from FWCHILDRENMODE_FWSETSTATES
                else
                    move_to Included
                endif
            else
                do ExcludeAll(OWNER=OWNER) phs_led_sys::phs_led_sys_FWM
                remove phs_led_sys::phs_led_sys from LED_FWSETSTATES
                remove phs_led_sys::phs_led_sys from LED_FWSETACTIONS
                remove phs_led_sys::phs_led_sys_FWCNM from FWCHILDRENMODE_FWSETSTATES
            endif
            move_to Excluded
    state: LockedOut  !color: FwStateOKNotPhysics
        action: UnLockOut !visible: 1
            move_to Excluded
        action: UnLockOut&Include(string OWNER = "",string EXCLUSIVE = "YES")  !visible: 1
            if ( phs_led_sys::phs_led_sys_FWM not_in_state Excluded ) then
            !    else
                    move_to LockedOut
            !    endif
            else
                do Include(OWNER=OWNER,EXCLUSIVE=EXCLUSIVE) phs_led_sys::phs_led_sys_FWM
                insert phs_led_sys::phs_led_sys in LED_FWSETSTATES
                insert phs_led_sys::phs_led_sys in LED_FWSETACTIONS
                insert phs_led_sys::phs_led_sys_FWCNM in FWCHILDRENMODE_FWSETSTATES
            endif
            move_to Included
```

```
        action: LockOutPerm    !visible: 0
            move_to LockedOutPerm
    state: ExcludedPerm   !color: FwStateOKNotPhysics
        action: Include(string OWNER = "",string EXCLUSIVE = "YES")   !visible: 1
            !    else
            !        move_to Excluded
            !    endif
            !else
            !endif
            !move_to Included
            if ( phs_led_sys::phs_led_sys_FWM not_in_state {Excluded, Manual} ) then
                move_to ExcludedPerm
            else
                do Include(OWNER=OWNER,EXCLUSIVE=EXCLUSIVE) phs_led_sys::phs_led_sys_FWM
                insert phs_led_sys::phs_led_sys in LED_FWSETSTATES
                insert phs_led_sys::phs_led_sys in LED_FWSETACTIONS
                insert phs_led_sys::phs_led_sys_FWCNM in FWCHILDRENMODE_FWSETSTATES
            endif
            move_to Included
        action: LockOut   !visible: 1
            move_to LockedOut
        action: Exclude(string OWNER = "")  !visible: 0
            move_to Excluded
    state: LockedOutPerm  !color: FwStateOKNotPhysics
        action: UnLockOut !visible: 1
            move_to Excluded
        action: UnLockOut&Include(string OWNER = "",string EXCLUSIVE = "YES")  !visible: 1
            if ( phs_led_sys::phs_led_sys_FWM not_in_state Excluded ) then
            !    else
                    move_to LockedOutPerm
            !    endif
            else
                do Include(OWNER=OWNER,EXCLUSIVE=EXCLUSIVE) phs_led_sys::phs_led_sys_FWM
                insert phs_led_sys::phs_led_sys in LED_FWSETSTATES
                insert phs_led_sys::phs_led_sys in LED_FWSETACTIONS
                insert phs_led_sys::phs_led_sys_FWCNM in FWCHILDRENMODE_FWSETSTATES
            endif
            move_to Included
        action: LockOut   !visible: 0
            move_to LockedOut

object: phs_led_sys_FWM is_of_class phs_led_sys_FwChildMode_CLASS

class: phs_vme_FwChildMode_CLASS

!panel: FwChildMode.pnl
    state: Excluded   !color: FwStateOKNotPhysics
        action: Include(string OWNER = "",string EXCLUSIVE = "YES")   !visible: 1
            if ( phs_vme::phs_vme_FWM in_state Dead ) then
              move_to Manual
            endif
            if ( phs_vme::phs_vme_FWM not_in_state {Excluded, Manual} ) then
            !    else
                    move_to Excluded
            !    endif
            else
                do Include(OWNER=OWNER,EXCLUSIVE=EXCLUSIVE) phs_vme::phs_vme_FWM
                insert phs_vme::phs_vme in VME_FWSETSTATES
                insert phs_vme::phs_vme in VME_FWSETACTIONS
                insert phs_vme::phs_vme_FWCNM in FWCHILDRENMODE_FWSETSTATES
            endif
            move_to Included
        action: Manual    !visible: 0
            do Manual phs_vme::phs_vme_FWM
            insert phs_vme::phs_vme in VME_FWSETSTATES
            remove phs_vme::phs_vme from VME_FWSETACTIONS
            move_to Manual
        action: Ignore    !visible: 0
            do Ignore phs_vme::phs_vme_FWM
            insert phs_vme::phs_vme in VME_FWSETACTIONS
            remove phs_vme::phs_vme from VME_FWSETSTATES
            move_to Ignored
        action: LockOut   !visible: 1
            move_to LockedOut
        action: Exclude(string OWNER = "")  !visible: 1
            do Exclude(OWNER=OWNER) phs_vme::phs_vme_FWM
            remove phs_vme::phs_vme from VME_FWSETSTATES
            remove phs_vme::phs_vme from VME_FWSETACTIONS
            remove phs_vme::phs_vme_FWCNM from FWCHILDRENMODE_FWSETSTATES
            move_to Excluded
        action: ExcludePerm(string OWNER = "")  !visible: 0
```

```
            move_to ExcludedPerm
        action: Exclude&LockOut(string OWNER = "")  !visible: 0
            move_to LockedOut
    state: Included   !color: FwStateOKPhysics
        when ( phs_vme::phs_vme_FWM in_state Excluded )  do Exclude
        when ( phs_vme::phs_vme_FWM in_state Ignored )  move_to IGNORED

        when ( phs_vme::phs_vme_FWM in_state Manual )  move_to MANUAL

        when ( phs_vme::phs_vme_FWM in_state Dead )  do Manual

        action: Exclude(string OWNER = "")  !visible: 1
            if ( phs_vme::phs_vme_FWM not_in_state Included ) then
                if ( phs_vme::phs_vme_FWM in_state InManual ) then
                    do Release(OWNER=OWNER) phs_vme::phs_vme_FWM
                    remove phs_vme::phs_vme from VME_FWSETSTATES
                    remove phs_vme::phs_vme from VME_FWSETACTIONS
                    remove phs_vme::phs_vme_FWCNM from FWCHILDRENMODE_FWSETSTATES
                else
                        do Exclude(OWNER=OWNER) phs_vme::phs_vme_FWM
                        remove phs_vme::phs_vme from VME_FWSETSTATES
                        remove phs_vme::phs_vme from VME_FWSETACTIONS
                        remove phs_vme::phs_vme_FWCNM from FWCHILDRENMODE_FWSETSTATES
            !       else
            !            move_to Included
            !        endif
                endif
            else
                do Exclude(OWNER=OWNER) phs_vme::phs_vme_FWM
                remove phs_vme::phs_vme from VME_FWSETSTATES
                remove phs_vme::phs_vme from VME_FWSETACTIONS
                remove phs_vme::phs_vme_FWCNM from FWCHILDRENMODE_FWSETSTATES
            endif
            move_to Excluded
        action: Manual(string OWNER = "")   !visible: 1
            do Manual(OWNER=OWNER) phs_vme::phs_vme_FWM
            insert phs_vme::phs_vme in VME_FWSETSTATES
            remove phs_vme::phs_vme from VME_FWSETACTIONS
            move_to Manual
        action: Ignore(string OWNER = "")   !visible: 1
            do Ignore(OWNER=OWNER) phs_vme::phs_vme_FWM
            insert phs_vme::phs_vme in VME_FWSETACTIONS
            remove phs_vme::phs_vme from VME_FWSETSTATES
            move_to Ignored
        action: ExcludeAll(string OWNER = "")   !visible: 1
            if ( phs_vme::phs_vme_FWM not_in_state {Included,Ignored,Manual} ) then
                if ( phs_vme::phs_vme_FWM in_state InManual ) then
                    do ReleaseAll(OWNER=OWNER) phs_vme::phs_vme_FWM
                    remove phs_vme::phs_vme from VME_FWSETSTATES
                    remove phs_vme::phs_vme from VME_FWSETACTIONS
                    remove phs_vme::phs_vme_FWCNM from FWCHILDRENMODE_FWSETSTATES
                else
                    move_to Included
                endif
            else
                do ExcludeAll(OWNER=OWNER) phs_vme::phs_vme_FWM
                remove phs_vme::phs_vme from VME_FWSETSTATES
                remove phs_vme::phs_vme from VME_FWSETACTIONS
                remove phs_vme::phs_vme_FWCNM from FWCHILDRENMODE_FWSETSTATES
            endif
            move_to Excluded
        action: Include(string OWNER = "",string EXCLUSIVE = "YES")   !visible: 1
            do Include(OWNER=OWNER,EXCLUSIVE=EXCLUSIVE) phs_vme::phs_vme_FWM
            insert phs_vme::phs_vme in VME_FWSETSTATES
            insert phs_vme::phs_vme in VME_FWSETACTIONS
            insert phs_vme::phs_vme_FWCNM in FWCHILDRENMODE_FWSETSTATES
            move_to Included
        action: Free(string OWNER = "")!visible: 0
            do Free(OWNER=OWNER) phs_vme::phs_vme_FWM
            move_to Included
        action: SetMode(string OWNER = "",string EXCLUSIVE = "YES")   !visible: 0
            do SetMode(OWNER=OWNER,EXCLUSIVE=EXCLUSIVE) phs_vme::phs_vme_FWM
        action: ExcludePerm(string OWNER = "")  !visible: 0
            if ( phs_vme::phs_vme_FWM not_in_state Included ) then
                if ( phs_vme::phs_vme_FWM in_state InManual ) then
                    do Release(OWNER=OWNER) phs_vme::phs_vme_FWM
                    remove phs_vme::phs_vme from VME_FWSETSTATES
                    remove phs_vme::phs_vme from VME_FWSETACTIONS
                    remove phs_vme::phs_vme_FWCNM from FWCHILDRENMODE_FWSETSTATES
                else
                    move_to Included
```

```
            endif
        else
            do Exclude(OWNER=OWNER) phs_vme::phs_vme_FWM
            remove phs_vme::phs_vme from VME_FWSETSTATES
            remove phs_vme::phs_vme from VME_FWSETACTIONS
            remove phs_vme::phs_vme_FWCNM from FWCHILDRENMODE_FWSETSTATES
        endif
        move_to ExcludedPerm
    action: Exclude&LockOut(string OWNER = "")  !visible: 1
        if ( phs_vme::phs_vme_FWM not_in_state Included ) then
            if ( phs_vme::phs_vme_FWM in_state InManual ) then
                do Release(OWNER=OWNER) phs_vme::phs_vme_FWM
                remove phs_vme::phs_vme from VME_FWSETSTATES
                remove phs_vme::phs_vme from VME_FWSETACTIONS
                remove phs_vme::phs_vme_FWCNM from FWCHILDRENMODE_FWSETSTATES
            else
                if ( phs_vme::phs_vme_FWM in_state Dead ) then
                    do Exclude(OWNER=OWNER) phs_vme::phs_vme_FWM
                    remove phs_vme::phs_vme from VME_FWSETSTATES
                    remove phs_vme::phs_vme from VME_FWSETACTIONS
                    remove phs_vme::phs_vme_FWCNM from FWCHILDRENMODE_FWSETSTATES
               else
                    move_to Included
                endif
            endif
        else
            do Exclude(OWNER=OWNER) phs_vme::phs_vme_FWM
            remove phs_vme::phs_vme from VME_FWSETSTATES
            remove phs_vme::phs_vme from VME_FWSETACTIONS
            remove phs_vme::phs_vme_FWCNM from FWCHILDRENMODE_FWSETSTATES
        endif
        move_to LockedOut
state: Manual !color: FwStateOKNotPhysics
    when ( phs_vme::phs_vme_FWM in_state Included )  move_to EXCLUDED
    action: Include(string OWNER = "",string EXCLUSIVE = "YES")   !visible: 1
        if ( phs_vme::phs_vme_FWM in_state Dead ) then
          move_to Manual
        endif
        if ( phs_vme::phs_vme_FWM not_in_state InManual ) then
          do Include(OWNER=OWNER,EXCLUSIVE=EXCLUSIVE) phs_vme::phs_vme_FWM
          insert phs_vme::phs_vme in VME_FWSETSTATES
          insert phs_vme::phs_vme in VME_FWSETACTIONS
          insert phs_vme::phs_vme_FWCNM in FWCHILDRENMODE_FWSETSTATES
        endif
          if ( phs_vme::phs_vme_FWM in_state Included ) then
            move_to Included
          endif
        move_to Manual
    action: Exclude(string OWNER = "")  !visible: 1
            do Exclude(OWNER=OWNER) phs_vme::phs_vme_FWM
            remove phs_vme::phs_vme from VME_FWSETSTATES
            remove phs_vme::phs_vme from VME_FWSETACTIONS
            remove phs_vme::phs_vme_FWCNM from FWCHILDRENMODE_FWSETSTATES
        !    move_to Excluded
        !endif
        !    else
        !    endif
        !else
        !endif
        !    move_to Excluded
        !endif
        !move_to Manual
        if ( phs_vme::phs_vme_FWM in_state InManual ) then
             do SetInLocal phs_vme::phs_vme_FWM
        endif
        move_to Excluded
    action: Ignore    !visible: 0
        do Ignore phs_vme::phs_vme_FWM
        insert phs_vme::phs_vme in VME_FWSETACTIONS
        remove phs_vme::phs_vme from VME_FWSETSTATES
        move_to Ignored
    action: Free(string OWNER = "")!visible: 0
        do Free(OWNER=OWNER) phs_vme::phs_vme_FWM
        !move_to Manual
        if ( phs_vme::phs_vme_FWM in_state InManual ) then
             do SetInLocal phs_vme::phs_vme_FWM
        endif
        move_to Excluded
    action: ExcludeAll(string OWNER = "")   !visible: 1
        !    else
        !        move_to Included
```

```
          !    endif
          !else
          !endif
            do ExcludeAll(OWNER=OWNER) phs_vme::phs_vme_FWM
            remove phs_vme::phs_vme from VME_FWSETSTATES
            remove phs_vme::phs_vme from VME_FWSETACTIONS
            remove phs_vme::phs_vme_FWCNM from FWCHILDRENMODE_FWSETSTATES
          !endif
          !move_to Manual
          if ( phs_vme::phs_vme_FWM in_state InManual ) then
               do SetInLocal phs_vme::phs_vme_FWM
          endif
          move_to Excluded
      action: Manual    !visible: 0
          do Manual phs_vme::phs_vme_FWM
          insert phs_vme::phs_vme in VME_FWSETSTATES
          remove phs_vme::phs_vme from VME_FWSETACTIONS
          move_to Manual
      action: Exclude&LockOut(string OWNER = "")  !visible: 1
              do Exclude(OWNER=OWNER) phs_vme::phs_vme_FWM
              remove phs_vme::phs_vme from VME_FWSETSTATES
              remove phs_vme::phs_vme from VME_FWSETACTIONS
              remove phs_vme::phs_vme_FWCNM from FWCHILDRENMODE_FWSETSTATES
          !    move_to Excluded
          !endif
          !    else
          !    endif
          !else
          !endif
          !    move_to Excluded
          !endif
          !move_to Manual
          if ( phs_vme::phs_vme_FWM in_state InManual ) then
               do SetInLocal phs_vme::phs_vme_FWM
          endif
          move_to LockedOut
    state: Ignored    !color: FwStateOKNotPhysics
      when ( phs_vme::phs_vme_FWM in_state Included )  move_to INCLUDED

      when ( phs_vme::phs_vme_FWM in_state Excluded ) move_to EXCLUDED

      when ( phs_vme::phs_vme_FWM in_state Dead )  do Exclude

      action: Include(string OWNER = "",string EXCLUSIVE = "YES")   !visible: 1
          do Include(OWNER=OWNER,EXCLUSIVE=EXCLUSIVE) phs_vme::phs_vme_FWM
          insert phs_vme::phs_vme in VME_FWSETSTATES
          insert phs_vme::phs_vme in VME_FWSETACTIONS
          insert phs_vme::phs_vme_FWCNM in FWCHILDRENMODE_FWSETSTATES
          move_to Included
      action: Exclude(string OWNER = "")  !visible: 1
          if ( phs_vme::phs_vme_FWM not_in_state Included ) then
              if ( phs_vme::phs_vme_FWM in_state InManual ) then
                  do Release(OWNER=OWNER) phs_vme::phs_vme_FWM
                  remove phs_vme::phs_vme from VME_FWSETSTATES
                  remove phs_vme::phs_vme from VME_FWSETACTIONS
                  remove phs_vme::phs_vme_FWCNM from FWCHILDRENMODE_FWSETSTATES
              else
                  do Exclude(OWNER=OWNER) phs_vme::phs_vme_FWM
                  remove phs_vme::phs_vme from VME_FWSETSTATES
                  remove phs_vme::phs_vme from VME_FWSETACTIONS
                  remove phs_vme::phs_vme_FWCNM from FWCHILDRENMODE_FWSETSTATES
              endif
          else
              do Exclude(OWNER=OWNER) phs_vme::phs_vme_FWM
              remove phs_vme::phs_vme from VME_FWSETSTATES
              remove phs_vme::phs_vme from VME_FWSETACTIONS
              remove phs_vme::phs_vme_FWCNM from FWCHILDRENMODE_FWSETSTATES
          endif
          move_to Excluded
      action: Manual(string OWNER = "")   !visible: 0
          do Manual(OWNER=OWNER) phs_vme::phs_vme_FWM
          insert phs_vme::phs_vme in VME_FWSETSTATES
          remove phs_vme::phs_vme from VME_FWSETACTIONS
          move_to Manual
      action: SetMode(string OWNER = "",string EXCLUSIVE = "YES")   !visible: 0
          do SetMode(OWNER=OWNER,EXCLUSIVE=EXCLUSIVE) phs_vme::phs_vme_FWM
      action: Free(string OWNER = "")!visible: 0
          do Free(OWNER=OWNER) phs_vme::phs_vme_FWM
          move_to Included
      action: ExcludeAll(string OWNER = "")   !visible: 1
          if ( phs_vme::phs_vme_FWM not_in_state {Included,Ignored,Manual} ) then
```

```
                if ( phs_vme::phs_vme_FWM in_state InManual ) then
                    do ReleaseAll(OWNER=OWNER) phs_vme::phs_vme_FWM
                    remove phs_vme::phs_vme from VME_FWSETSTATES
                    remove phs_vme::phs_vme from VME_FWSETACTIONS
                    remove phs_vme::phs_vme_FWCNM from FWCHILDRENMODE_FWSETSTATES
                else
                    move_to Included
                endif
            else
                do ExcludeAll(OWNER=OWNER) phs_vme::phs_vme_FWM
                remove phs_vme::phs_vme from VME_FWSETSTATES
                remove phs_vme::phs_vme from VME_FWSETACTIONS
                remove phs_vme::phs_vme_FWCNM from FWCHILDRENMODE_FWSETSTATES
            endif
            move_to Excluded
    state: LockedOut  !color: FwStateOKNotPhysics
        action: UnLockOut !visible: 1
            move_to Excluded
        action: UnLockOut&Include(string OWNER = "",string EXCLUSIVE = "YES")  !visible: 1
            if ( phs_vme::phs_vme_FWM not_in_state Excluded ) then
            !    else
                    move_to LockedOut
            !    endif
            else
                do Include(OWNER=OWNER,EXCLUSIVE=EXCLUSIVE) phs_vme::phs_vme_FWM
                insert phs_vme::phs_vme in VME_FWSETSTATES
                insert phs_vme::phs_vme in VME_FWSETACTIONS
                insert phs_vme::phs_vme_FWCNM in FWCHILDRENMODE_FWSETSTATES
            endif
            move_to Included
        action: LockOutPerm    !visible: 0
            move_to LockedOutPerm
    state: ExcludedPerm   !color: FwStateOKNotPhysics
        action: Include(string OWNER = "",string EXCLUSIVE = "YES")   !visible: 1
            !    else
            !        move_to Excluded
            !    endif
            !else
            !endif
            !move_to Included
            if ( phs_vme::phs_vme_FWM not_in_state {Excluded, Manual} ) then
                move_to ExcludedPerm
            else
                do Include(OWNER=OWNER,EXCLUSIVE=EXCLUSIVE) phs_vme::phs_vme_FWM
                insert phs_vme::phs_vme in VME_FWSETSTATES
                insert phs_vme::phs_vme in VME_FWSETACTIONS
                insert phs_vme::phs_vme_FWCNM in FWCHILDRENMODE_FWSETSTATES
            endif
            move_to Included
        action: LockOut   !visible: 1
            move_to LockedOut
        action: Exclude(string OWNER = "")  !visible: 0
            move_to Excluded
    state: LockedOutPerm  !color: FwStateOKNotPhysics
        action: UnLockOut !visible: 1
            move_to Excluded
        action: UnLockOut&Include(string OWNER = "",string EXCLUSIVE = "YES")  !visible: 1
            if ( phs_vme::phs_vme_FWM not_in_state Excluded ) then
            !    else
                    move_to LockedOutPerm
            !    endif
            else
                do Include(OWNER=OWNER,EXCLUSIVE=EXCLUSIVE) phs_vme::phs_vme_FWM
                insert phs_vme::phs_vme in VME_FWSETSTATES
                insert phs_vme::phs_vme in VME_FWSETACTIONS
                insert phs_vme::phs_vme_FWCNM in FWCHILDRENMODE_FWSETSTATES
            endif
            move_to Included
        action: LockOut   !visible: 0
            move_to LockedOut

object: phs_vme_FWM is_of_class phs_vme_FwChildMode_CLASS

objectset: FWCHILDMODE_FWSETSTATES is_of_class VOID {phs_led_sys_FWM,
    phs_vme_FWM }
objectset: FWCHILDMODE_FWSETACTIONS is_of_class VOID {phs_led_sys_FWM,
    phs_vme_FWM }

class: ASS_led_CLASS/associated
!panel: led.pnl
    state: NOT_READY  !color: FwStateOKNotPhysics
```

```
        action: GO_READY   !visible: 1
        action: DEACTIVATE !visible: 1
    state: CONFIGURING     !color: FwStateAttention1
        action: DEACTIVATE !visible: 1
    state: READY  !color: FwStateOKPhysics
        action: GO_OFF     !visible: 1
        action: DEACTIVATE !visible: 1
    state: DEACTIVATED     !color: FwStateOKPhysics
        action: ACTIVATE   !visible: 1
    state: ERROR  !color: FwStateAttention3
        action: RESET !visible: 1

object: phs_led_sys::phs_led_sys is_of_class ASS_led_CLASS

objectset: LED_FWSETSTATES is_of_class VOID
objectset: LED_FWSETACTIONS is_of_class VOID

class: ASS_vme_CLASS/associated
!panel: vme.pnl
    state: READY  !color: FwStateOKPhysics
        action: RESET !visible: 1
    state: NOT_READY  !color: FwStateOKNotPhysics
        action: CONFIGURE  !visible: 1
    state: ERROR  !color: FwStateAttention3
        action: RECOVER    !visible: 1
    state: NO_CONTROL !color: FwStateAttention2

object: phs_vme::phs_vme is_of_class ASS_vme_CLASS

objectset: VME_FWSETSTATES is_of_class VOID
objectset: VME_FWSETACTIONS is_of_class VOID


objectset: FWCHILDREN_FWSETACTIONS union {LED_FWSETACTIONS,
    VME_FWSETACTIONS } is_of_class VOID
objectset: FWCHILDREN_FWSETSTATES union {LED_FWSETSTATES,
    VME_FWSETSTATES } is_of_class VOID
```

Приложение 9. Примеры шаблонов прокси

Пример 1:

Шаблон **logger_skel.c** прокси для ассоциированного объекта **LOGGER**, полученный программой **smiGen** (см. раздел 3.4) из файла **run_con.sml**, см. Приложение 5.

```
#include <smirtl.h>

/* States */
void setNOT_LOGGING() {smi_set_state("NOT_LOGGING");};
void setLOGGING() {smi_set_state("LOGGING");};
void setWRITING() {smi_set_state("WRITING");};

/* Object Parameters */

/* Command Handler */
void handle_command()
{
   void LOG();
   void NOLOG();
   void X_OPEN_FILE();
   void X_CLOSE_FILE();

   if(smi_test_action("LOG"))
   {
      LOG();
   }
   else if(smi_test_action("NOLOG"))
   {
      NOLOG();
   }
   else if(smi_test_action("X_OPEN_FILE"))
   {
      X_OPEN_FILE();
   }
   else if(smi_test_action("X_CLOSE_FILE"))
   {
      X_CLOSE_FILE();
   }
}

/* Main Program */
int main(argc, argv)
int argc;
char **argv[];
{

   char proxy_name[64];
   void handle_command();
   sprintf(proxy_name,"%s::LOGGER",argv[1]);
   smi_attach(proxy_name, handle_command);
/*   setNOT_LOGGING() */

   while(1)
   {
      pause();
   }
}
```

```
/* Action LOG User code */
void LOG()
{
/*   setXXX();*/
}


/* Action NOLOG User code */
void NOLOG()
{
/*   setXXX();*/
}


/* Action X_OPEN_FILE User code */
void X_OPEN_FILE()
{
/*   setXXX();*/
}


/* Action X_CLOSE_FILE User code */
void X_CLOSE_FILE()
{
/*   setXXX();*/
}
```

Пример 2:

Шаблон **evt_builder_skel.c** прокси для ассоциированного объекта **EVT_BUILDER**, полученный программой **smiGen** (см. раздел 3.4) из файла **run_con.sml**, см. Приложение 5.

```
#include <smirtl.h>

/* States */
void setREADY() {smi_set_state("READY");};
void setRUNNING() {smi_set_state("RUNNING");};
void setERROR() {smi_set_state("ERROR");};

/* Object Parameters */
int setParNUMBER_T(int val) {return smi_set_par("NUMBER_T", &val, INTEGER);};
int setParNUMBER_P(int val) {return smi_set_par("NUMBER_P", &val, INTEGER);};

/* Command Handler */
void handle_command()
{
   void START();
   void STOP();
   void RECOVER();

   if(smi_test_action("START"))
   {
      START();
   }
   else if(smi_test_action("STOP"))
   {
      STOP();
   }
   else if(smi_test_action("RECOVER"))
   {
      RECOVER();
```

```
   }
}

/* Main Program */
int main(argc, argv)
int argc;
char **argv[];
{

   char proxy_name[64];
   void handle_command();
   sprintf(proxy_name,"%s::EVT_BUILDER",argv[1]);
   smi_attach(proxy_name, handle_command);
/*   setParNUMBER_T(value); */
/*   setParNUMBER_P(value); */
/*   setREADY() */

   while(1)
   {
      pause();
   }
}

/* Parameters of action START */
int getParTYPE(char *val) { return smi_get_par_value("TYPE",val);};
int getParNR(int *val) { return smi_get_par_value("NR",val);};

/* Action START User code */
void START()
{
/*   setXXX();*/
}


/* Action STOP User code */
void STOP()
{
/*   setXXX();*/
}


/* Action RECOVER User code */
void RECOVER()
{
/*   setXXX();*/
}
```