МЕЖДУНАРОДНЫЙ СОЮЗ ЭЛЕКТРОСВЯЗИ

МСЭ-Т Z.142

СЕКТОР СТАНДАРТИЗАЦИИ
ЭЛЕКТРОСВЯЗИ МСЭ

(03/2006)

СЕРИЯ Z: ЯЗЫКИ И ОБЩИЕ АСПЕКТЫ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ДЛЯ СИСТЕМ ЭЛЕКТРОСВЯЗИ

Методы формального описания (FDT) – Нотация тестирования и управления тестированием (TTCN)

Нотация тестирования и управления тестированием версии 3 (TTCN-3): Формат графического представления (GFT)

Рекомендация МСЭ-Т Z.142

РЕКОМЕНДАЦИИ МСЭ-Т СЕРИИ Z

ЯЗЫКИ И ОБЩИЕ АСПЕКТЫ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ДЛЯ СИСТЕМ ЭЛЕКТРОСВЯЗИ

МЕТОДЫ ФОРМАЛЬНОГО ОПИСАНИЯ (FDT)	
Язык спецификации и описания (SDL)	Z.100–Z.109
Применение методов формального описания	Z.110–Z.119
Диаграмма последовательности сообщений (MSC)	Z.120–Z.129
Расширенный язык описания объектов (eODL)	Z.130–Z.139
Нотация тестирования и управления тестированием (TTCN)	**Z.140–Z.149**
Нотация требований пользователя (URN)	Z.150–Z.159
ЯЗЫКИ ПРОГРАММИРОВАНИЯ	
CHILL: язык высокого уровня МСЭ-Т	Z.200–Z.209
ЯЗЫК "ЧЕЛОВЕК–МАШИНА"	
Общие принципы	Z.300–Z.309
Базисный синтаксис и диалоговые процедуры	Z.310–Z.319
Расширенный язык MML для видеотерминалов	Z.320–Z.329
Спецификация интерфейса "человек–машина"	Z.330–Z.349
Информационно-ориентированные интерфейсы "человек–машина"	Z.350–Z.359
Интерфейсы "человек–машина" для управления сетями электросвязи	Z.360–Z.379
КАЧЕСТВО	
Качество программного обеспечения электросвязи	Z.400–Z.409
Аспекты качества рекомендаций, относящихся к протоколам	Z.450–Z.459
МЕТОДЫ	
Методы проверки и тестирования	Z.500–Z.519
ПРОМЕЖУТОЧНОЕ ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ	
Среда распределенной обработки	Z.600–Z.609

Для получения более подробной информации просьба обращаться к перечню Рекомендаций МСЭ-Т.

Рекомендация МСЭ-Т Z.142

Нотация тестирования и управления тестированием версии 3 (TTCN-3): Формат графического представления (GFT)

Резюме

В настоящей Рекомендации определяется GFT – формат графического представления для базового языка TTCN-3 (*нотация тестирования и управления тестированием 3*), определенного в Рекомендации МСЭ-Т Z.140. Формат GFT используется для графического представления поведения испытаний в форме подмножества диаграмм последовательностей сообщений, определенного в Рекомендации МСЭ-Т Z.120, с расширениями, связанными с тестами. Формат GFT предоставляет ряд графических символов для обеспечения возможности графического представления тестовых случаев, функций, альтернативных шагов и частей управления TTCN-3. Формат GFT может применяться, когда необходимо графически определить или задокументировать поведение испытаний.

Настоящая Рекомендация основана на базовом языке TTCN-3, определенном в Рекомендации МСЭ-Т Z.140. Он особенно подходит для отображения испытаний в виде форматов GFT. Он не ограничивается каким-либо конкретным видом характеристики испытаний.

Источник

Рекомендация МСЭ-Т Z.142 была утверждена 16 марта 2006 года 17-й Исследовательской комиссией МСЭ-Т (2005–2008 гг.) в соответствии с процедурой, изложенной в Рекомендации МСЭ-Т A.8.

ПРЕДИСЛОВИЕ

Международный союз электросвязи (МСЭ) является специализированным учреждением Организации Объединенных Наций в области электросвязи. Сектор стандартизации электросвязи МСЭ (МСЭ-Т) – постоянный орган МСЭ. МСЭ-Т отвечает за изучение технических, эксплуатационных и тарифных вопросов и за выпуск Рекомендаций по ним с целью стандартизации электросвязи на всемирной основе.

На Всемирной ассамблее по стандартизации электросвязи (ВАСЭ), которая проводится каждые четыре года, определяются темы для изучения Исследовательскими комиссиями МСЭ-Т, которые, в свою очередь, вырабатывают Рекомендации по этим темам.

Утверждение рекомендаций МСЭ-Т осуществляется в соответствии с процедурой, изложенной в Резолюции 1 ВАСЭ.

В некоторых областях информационных технологий, которые входят в компетенцию МСЭ-Т, необходимые стандарты разрабатываются на основе сотрудничества с ИСО и МЭК.

ПРИМЕЧАНИЕ

В настоящей Рекомендации термин "администрация" используется для краткости и обозначает как администрацию электросвязи, так и признанную эксплуатационную организацию.

Соблюдение положений данной Рекомендации носит добровольный характер. Однако в Рекомендации могут содержаться определенные обязательные положения (например, для обеспечения возможности взаимодействия или применимости), и соблюдение положений данной Рекомендации достигается в случае выполнения всех этих обязательных положений. Для выражения необходимости выполнения требований используется синтаксис долженствования и соответствующие слова (такие, как "должен" и т. п.), а также их отрицательные эквиваленты. Использование этих слов не предполагает, что соблюдение положений данной Рекомендации является обязательным для какой-либо из сторон.

ПРАВА ИНТЕЛЛЕКТУАЛЬНОЙ СОБСТВЕННОСТИ

МСЭ обращает внимание на вероятность того, что практическое применение или реализация этой Рекомендации может включать использование заявленного права интеллектуальной собственности. МСЭ не занимает какую бы то ни было позицию относительно подтверждения, обоснованности или применимости заявленных прав интеллектуальной собственности, независимо от того, отстаиваются ли они членами МСЭ или другими сторонами вне процесса подготовки Рекомендации.

На момент утверждения настоящей Рекомендации МСЭ не получил извещение об интеллектуальной собственности, защищенной патентами, которые могут потребоваться для выполнения этой Рекомендации. Однако те, кто будет применять Рекомендацию, должны иметь в виду, что это может не отражать самую последнюю информацию, и поэтому им настоятельно рекомендуется обращаться к патентной базе данных БСЭ.

СОДЕРЖАНИЕ

Стр.

1 Сфера применения ... 1
2 Справочные документы ... 1
3 Сокращения ... 1
4 Обзор ... 1
5 Принципы языка GFT ... 3
6 Преобразование между GFT и базовым языком TTCN-3 ... 4
7 Структура модуля ... 5
8 Символы GFT ... 7
9 Диаграммы GFT ... 9
9.1 Общие свойства ... 9
9.2 Диаграмма управления ... 10
9.3 Диаграмма тестового случая ... 11
9.4 Диаграмма функции ... 11
9.5 Диаграмма альтернативных шагов ... 12
10 Варианты в диаграммах GFT ... 13
10.1 Вариант управления ... 13
10.2 Варианты тестовых компонентов ... 13
10.3 Варианты портов ... 14
11 Диаграммы GFT элементов ... 14
11.1 Общие правила изображения ... 14
11.2 Запуск диаграмм GFT ... 15
11.3 Объявления ... 17
11.4 Базовые операторы программ ... 19
11.5 Программные операторы поведения ... 22
11.6 Обработка по умолчинию ... 26
11.7 Операции конфигурирования ... 27
11.8 Операции связи ... 30
11.9 Операции таймера ... 46
11.10 Операции установления заключений по тестам ... 49
11.11 Внешние действия ... 49
11.12 Определение атрибутов ... 49
Приложение A – Форма BNF формата GFT ... 50
A.1 Метаязык для GFT ... 50
A.2 Соглашения об описании синтаксиса ... 50
A.3 Грамматика формата GFT ... 51
Приложение B – Справочное руководство по GFT ... 74
Приложение C – Примеры ... 97
C.1 Пример ресторана ... 97
C.2 Пример INRES ... 106

Введение

Формат графического представления TTCN-3 (GFT) основан на Рекомендации МСЭ-Т Z.120 [3], определяющей диаграммы последовательностей сообщений (MSC). Формат GFT использует подмножество MSC с расширениями, связанными с тестами. Большинство расширений являются только текстовыми расширениями. Графические расширения определены с тем, чтобы облегчить читабельность диаграмм GFT. При наличии возможности GFT определяется как MSC с тем, чтобы созданный с небольшими изменениями инструментарий MSC мог бы использоваться для графического определения случаев тестирования TTCN-3 в виде GFT.

Базовый язык TTCN-3 определен в Рекомендации МСЭ-Т Z.140 [1] и предоставляет полный синтаксис, основанный на тексте, статические и оперативные семантики, а также определение для использования языка с ASN.1. Формат представления GFT предоставляет альтернативный способ отображения базового языка (см. рисунок 1).

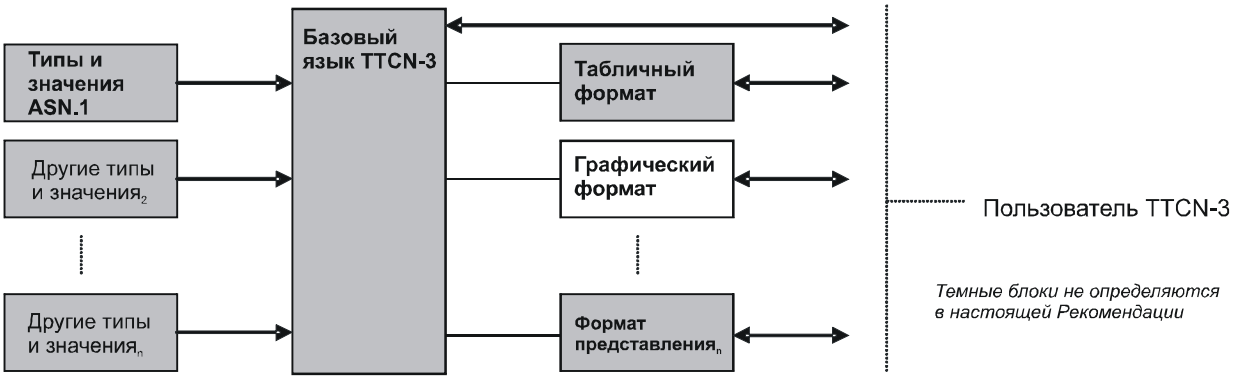


Рисунок 1/Z.142 – Взгляд пользователя на базовый язык и различные форматы представления

Базовый язык может использоваться независимо от GFT. Однако GFT не может использоваться без базового языка. Использование и реализация GFT должны осуществляться на основе базового языка.

В настоящей Рекомендации определяются:

- принципы языка GFT;
- руководящие принципы для использования GFT;
- грамматика GFT;
- преобразование из базового языка TTCN-3 и в базовый язык TTCN-3.

Вместе эти характеристики образуют GFT – графический формат представления TTCN-3.

Рекомендация МСЭ-Т Z.142

Нотация тестирования и управления тестированием версии 3 (TTCN-3): Формат графического представления (GFT)

1 Сфера применения

В настоящей Рекомендации определяется графический формат представления для базового языка TTCN-3, определенного в Рекомендации МСЭ-Т Z.140 [1]. В этом формате представления используется подмножество диаграмм последовательностей сообщений, определенных в Рекомендации МСЭ-Т Z.120 [3] с расширениями, связанными с тестами.

Настоящая Рекомендация основана на базовом языке TTCN-3, определенном в Рекомендации МСЭ-Т Z.140 [1]. Он особенно подходит для отображения испытаний в виде форматов GFT. Он не ограничивается каким-либо конкретным видом характеристики испытаний.

Характеристики других форматов находятся за пределами сферы применения настоящей Рекомендации.

2 Справочные документы

Нижеследующие Рекомендации МСЭ-Т и другие ссылки содержат пункты, на которые имеются ссылки в тексте этих Рекомендаций. Во время опубликования все перечисленные издания были в силе. Все Рекомендации и другие ссылки могут пересматриваться: все пользователи настоящих Рекомендаций должны использовать возможность применения наиболее современного издания Рекомендаций и других ссылок приведенных ниже. Список действующих в настоящее время Рекомендаций МСЭ-Т регулярно публикуется. Ссылка на документ, приведенный в настоящей Рекомендации, не придает ему как отдельному документу статуса рекомендации.

[1] ITU-T Recommendation Z.140 (2006), *Testing and Test Control Notation version 3 (TTCN-3): Core language*.

[2] Рекомендация МСЭ-Т Z.141 (2006), *Нотация тестирования и управления тестированием, версия 3 (TTCN-3): Формат табличного представления (TFT)*.

[3] ITU-T Recommendation Z.120 (2004), *Message sequence chart (MSC)*.

[4] ITU-T Recommendation X.292 (2002), *OSI conformance testing methodology and framework for protocol Recommendations for ITU-T applications – The Tree and Tabular Combined Notation (TTCN)*.

ISO/IEC 9646-3:1998, *Information technology – Open Systems Interconnection – Conformance testing methodology and framework – Part 3: The Tree and Tabular Combined Notation (TTCN)*.

3 Сокращения

В настоящей Рекомендации используются следующие сокращения:

BNF	Backus-Naur Form	Форма Бэкуса-Наура
CATG	Computer-Aided Test Generation	Создание компьютеризированных тестов
GFT	Graphical presentation Format of TTCN-3	Формат графического представления TTCN-3
MSC	Message Sequence Chart	Диаграмма последовательностей сообщений
MTC	Main Test Component	Основной тестовый компонент
PTC	Parallel Test Component	Параллельный тестовый компонент
SUT	System Under Test	Тестируемая система
TFT	Tabular presentation Format of TTCN-3	Формат табличного представление TTCN-3
TTCN	Testing and Test Control Notation	Нотация тестирования и управления тестированием

4 Обзор

В соответствии с методикой тестирования на соответствие ВОС, определенной в Рекомендации МСЭ-Т X.292 [4], тестирование обычно начинается с определения целей проведения тестов. Цель проведения тестов определяется следующим образом:

"Строгое описание хорошо определенной цели тестирования, сконцентрированное на требовании соответствия или множестве связанных с ним требований соответствия, определенных в надлежащей технической характеристике ВОС".

После определения целей проведения тестов создается абстрактная тестовая последовательность, которая состоит из одного или нескольких абстрактных тестовых случаев. Абстрактный тестовый случай определяет действия процессов испытательного устройства, необходимые для подтверждения правильности части цели (или всей цели) проведения тестов.

Применяя эти условия к диаграммам последовательностей сообщений (MSC), мы можем определить две категории их использования:

1) *Использование диаграмм MSC для определения целей проведения тестов* – Как правило, в качестве цели проведения испытания может рассматриваться характеристика MSC, разработанная в качестве сценария использования или в качестве части характеристики системы, т. е. она описывает требование к SUT в форме описания поведения, которое может тестироваться. Например, на рисунке 2 показана простая MSC, описывающая взаимодействие вариантов SUT и ее интерфейсов A, B и C. При реальной реализации такой системы интерфейсы A, B и C могут отображаться на точки или порты доступа к обслуживанию. Диаграмма MSC на рисунке 2 описывает только взаимодействие с SUT и не описывает действия тестовых компонентов, необходимых для подтверждения правильности поведения SUT; т. е. она является описанием цели проведения испытаний.

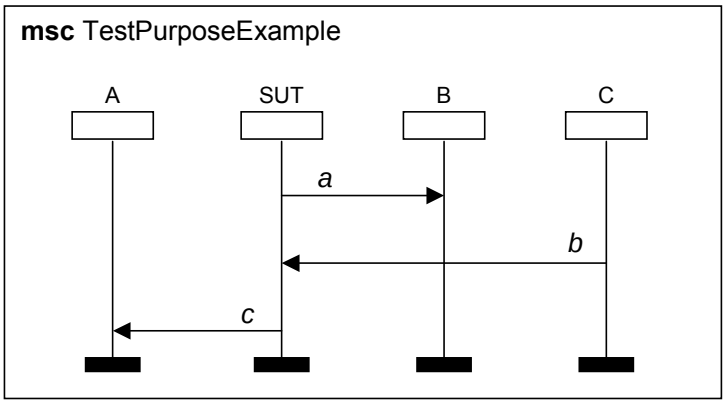


Рисунок 2/Z.142 – Диаграмма MSC, описывающая взаимодействие SUT с ее интерфейсами

2) *Использование диаграмм MSC для определения случаев абстрактного тестирования* – Характеристика диаграммы MSC, описывающей случай абстрактного тестирования, определяет поведение тестовых компонентов, необходимое для подтверждения правильности цели проведения соответствующего теста. На рисунке 3 представлено описание простой MSC случая абстрактного теста. Он показывает один основной тестовый компонент (MTC), который обменивается с SUT сообщениями *a*, *b* и *c* через порты PortA, PortB и PortC в целях достижения цели проведения тестов, представленной на рисунке 2. Сообщения *a* и *c* направляются системой SUT через порты A и B (рисунок 2) и принимаются MTC (рисунок 3) через те же самые порты. Сообщение *b* направляется MTC и принимается SUT.

ПРИМЕЧАНИЕ. – Примеры на рисунках 2 и 3 являются только простыми примерами для иллюстрации различных случаев использования MSC для тестирования. Диаграммы будут более сложными в случае распределенной SUT, которая состоит из нескольких процессов, или распределенной конфигурации теста с несколькими тестовыми компонентами.

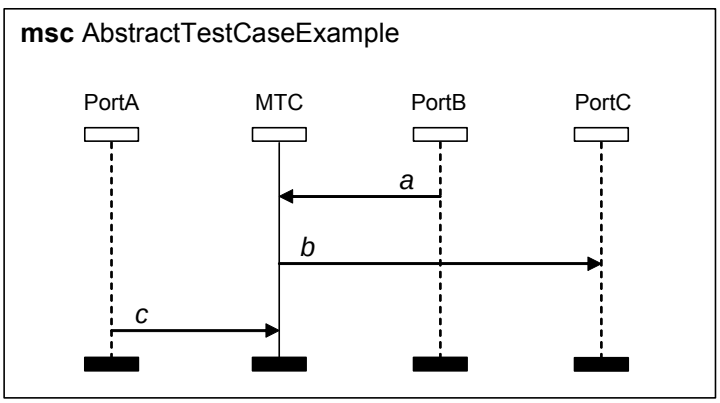


Рисунок 3/Z.142 – Диаграмма MSC, описывающая взаимодействие MTC с интерфейсами SUT

При определении этих двух категорий использования MSC могут быть определены две различные области работы (см. рисунок 4):

a) *Создание случаев абстрактных тестов на основе описаний целей проведения тестов MSC* – базовый язык TTCN-3 или GFT может использоваться для представления случаев абстрактных тестов. Однако предполагается, что создание тестовых случаев на основе целей проведения тестов не является простой задачей и включает применение и разработку методов создания компьютеризированных тестов (CATG).

b) *Разработка графического представления формата для TTCN-3 (GFT) и определение преобразования между GFT и TTCN-3*.

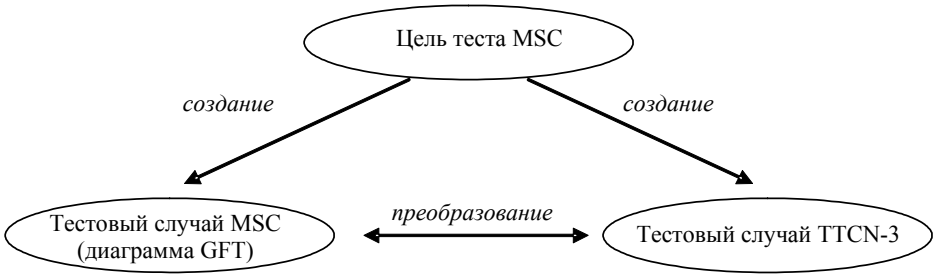


Рисунок 4/Z.142 – Связи между описанием цели проведения испытаний MSC, описаниями тестовых случаев MSC и TTCN-3

В настоящей Рекомендации внимание сосредоточено на пункте b), т. е. в ней определяется GFT и преобразование между GFT и базовым языком TTCN-3.

5 Принципы языка GFT

Графически формат GFT представляет собой такие аспекты поведения TTCN-3, как поведение тестового случая или функции. Он не предоставляет графиков для таких аспектов данных, как объявление типов и шаблонов.

Формат GFT не определяет графическое представление для структуры модуля TTCN-3, а указывает требования к такому графическому представлению (см. также пункт 7).

ПРИМЕЧАНИЕ. – Порядок и группирование определений и объявлений в части определений модуля определяет структуру модуля TTCN-3.

Формат GFT не определяет графическое представление для:

- определений параметров модуля;
- определений импорта;
- определений типов;
- определений подписи;
- определений шаблонов;
- объявлений констант;
- внешних объявлений констант; и
- внешних объявлений функций.

Определения и объявления TTCN-3 без соответствующего представления GFT могут быть представлены в базовом языке TTCN-3 или в табличном представлении формата для TTCN-3 (TFT) (Рек. МСЭ-Т Z.141 [2]).

Формат GFT предоставляет графики для описаний поведения TTCN-3. Это означает, что диаграмма GFT обеспечивает графическое представление следующего:

- части управления модуля TTCN-3;
- тестового случая TTCN-3;
- функции TTCN-3; или
- альтернативного шага TTCN-3.

Связь между модулем TTCN-3 и соответствующим представлением GFT показана на рисунке 5.

Рисунок 5/Z.142 – Связь между базовым языком TTCN-3 и соответствующим описанием GFT

Формат GFT основан на MSC (Рек. МСЭ-Т Z.120 [3]) и, следовательно, диаграмма GFT преобразуется в диаграмму MSC. Несмотря на то, что в GFT используется большинство графических символов MSC, записи некоторых символов MSC были адаптированы к потребностям тестирования и, кроме того, был определен ряд новых символов в целях выделения аспектов, связанных с тестами. Однако новые символы могут быть преобразованы в подходящую MSC.

- представление вариантов портов;
- создание тестовых компонентов;
- запуск тестовых компонентов;
- возврат из процедуры вызова функции;
- повторение альтернатив;
- время контроля за вызовом, основанном на процедуре;
- выполнение тестовых случаев;
- включение и выключение состояний по умолчанию;
- маркирование и переход; и
- таймеры в операторах вызова.

Полный список символов, используемых в GFT, представлен в пункте 8.

6 Преобразование между GFT и базовым языком TTCN-3

Формат GFT предоставляет графические средства для определений поведения TTCN-3. Часть управления и каждая функция, альтернативный шаг и тестовый случай модуля базового языка TTCN-3 могут быть преобразованы в диаграмму GFT и наоборот. Это означает, что:

- часть управления модулем может быть преобразована в диаграмму управления (см. п. 9.2) и наоборот;
- тестовый случай может быть преобразован в диаграмму тестового случая (см. п. 9.3) и наоборот;
- функция в базовом языке может быть преобразована в диаграмму функции (см. п. 9.4) и наоборот;
- альтернативный шаг может быть преобразован в диаграмму альтернативного шага (см. п. 9.5) и наоборот.

ПРИМЕЧАНИЕ 1. – Формат GFT не предоставляет графических представлений для определений параметров модуля, типов, констант, подписей, шаблонов, внешних констант и внешних функций в части определений модуля. Эти определения могут быть представлены непосредственно в базовом языке или путем использования другого формата представления, например формата табличного представления.

Каждое объявление, операция и оператор в управлении модулем и каждый тестовый случай, альтернативный шаг или функция могут быть преобразованы в соответствующее представление GFT и наоборот.

Порядок объявлений, операций и операторов в определении управления модулем, тестового случая, альтернативного шага или функции идентичен порядку соответствующих представлений GFT в соответствующей диаграмме управления, тестового случая, альтернативного шага или функции.

ПРИМЕЧАНИЕ 2. – Порядок конструктивных элементов GFT в диаграмме GFT определяется порядком конструктивных элементов GFT в заголовке диаграммы (только объявления) и порядком конструктивных элементов GFT в варианте управления (диаграмме управления) или варианте компонента (диаграмма тестового случая, диаграмма альтернативного шага или диаграмма функции).

7 Структура модуля

Как показано на рисунке 6, модуль TTCN-3 имеет структуру дерева. Структура модуля TTCN-3 состоит из части определений модуля и части управления модуля. Часть определений модуля состоит из определений и объявлений, структура которых может далее формироваться с помощью групп. Часть управления модуля не может быть структурирована на подструктуры; она определяет порядок выполнения и условия выполнения тестовых случаев.

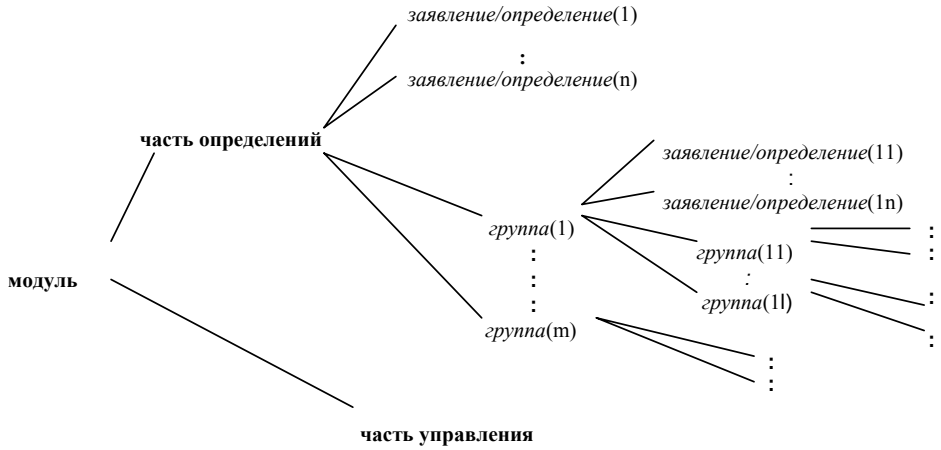


Рисунок 6/Z.142 – Структура модулей TTCN-3

Формат GFT предоставляет диаграммы для всех "поведенческих" листьев древовидной структуры модуля, т. е. для части управления модулем, для функций, для альтернативных шагов и для тестовых случаев. Формат GFT не определяет конкретных графиков для древовидной структуры модуля, однако надлежащая инструментальная поддержка для GFT требует графического представления структуры модуля TTCN-3. Структура модуля TTCN-3 может иметь форму в виде организатора (рисунок 7) или документоподобного представления MSC (рисунок 8). Усовершенствованный инструмент может даже поддерживать различные представления одного и того же объекта; например вид организатора на рисунке 7 указывает, что некоторые определения предоставляются в нескольких форматах представления: например, в базовом языке функция MySpecialFunction имеется в форме таблицы TFT и в виде диаграмма GFT.

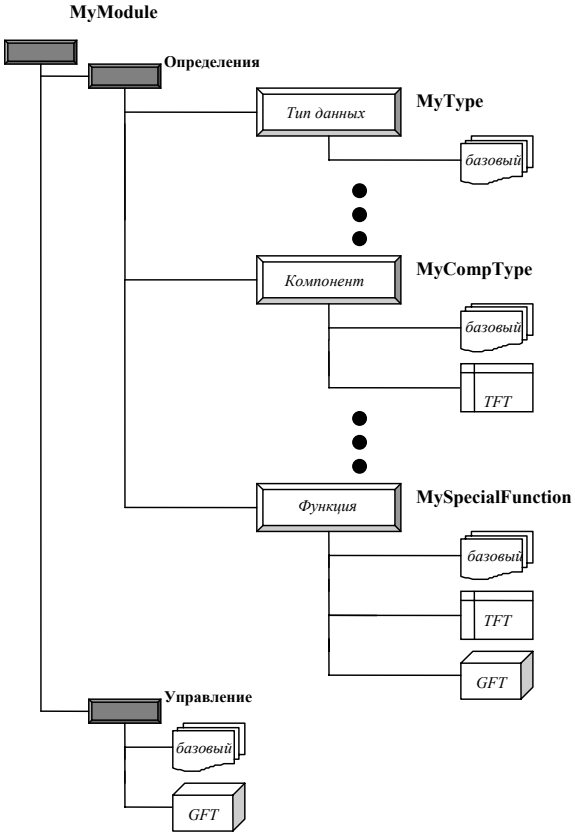


Рисунок 7/Z.142 – Различные форматы представления в виде организатора структуры модуля TTCN-3

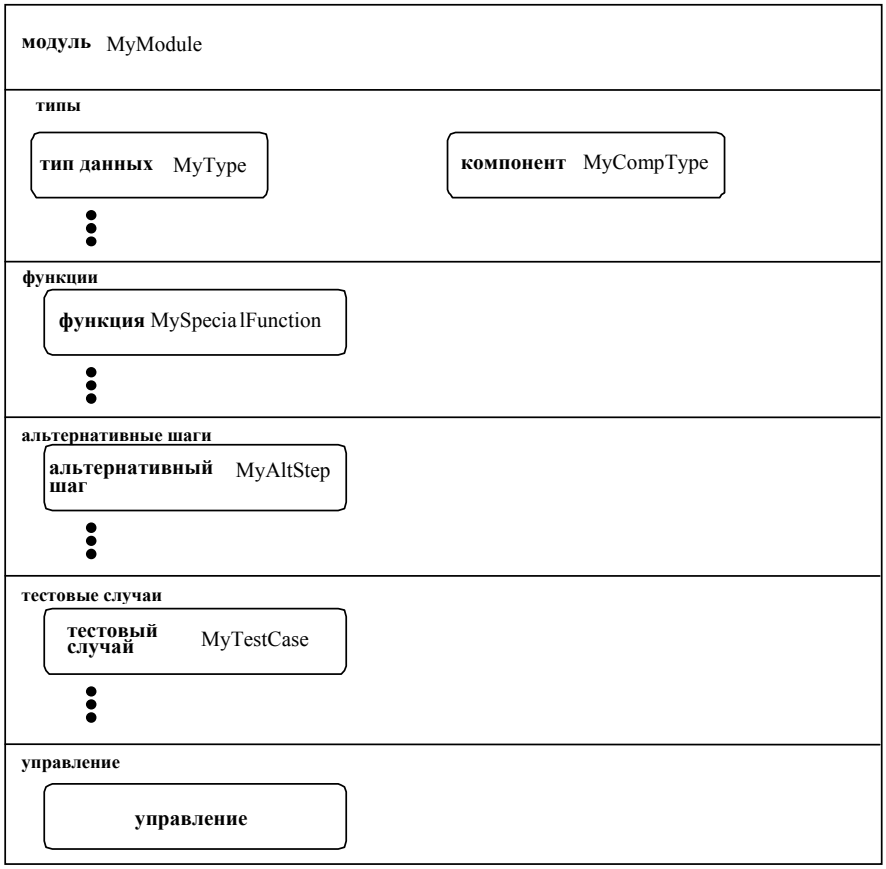


Рисунок 8/Z.142 – Графическое документоподобное представление MSC структуры модуля TTCN-3

8 Символы GFT

В настоящем пункте приводятся все графические символы, используемые в диаграммах GFT, и комментарии в отношении их типичного использования в GFT.

Таблица 1/Z.142 – Символы GFT

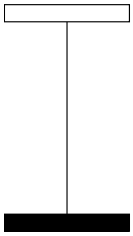
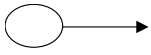
Элемент GFT	Символ	Описание
Символ кадра		Используется для помещения диаграмм GFT в кадры
Символ ссылки		Используется для представления вызова функций и альтернативных шагов
Символ варианта порта		Используется для представления вариантов портов
Символ варианта компонента		Используется для представления тестовых компонентов и варианта управления
Символ блока действия		Используется для текстовых объявлений и операторов TTCN-3, должен быть присоединен к символу компонента
Символ условия		Используется для текстовых булевых выражений TTCN-3, установления заключений, операций портов (запуска, остановки и очистки) и оператора выполнения, должен быть присоединен к символу компонента
Символ маркирования		Используется для операций маркирования и перехода TTCN-3, должен быть присоединен к символу компонента
Символ перехода		Используется для операций маркирования и перехода TTCN-3, должен быть присоединен к символу компонента

Таблица 1/Z.142 – Символы GFT

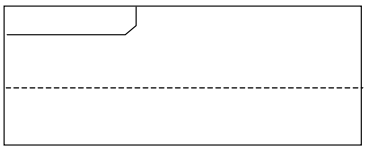
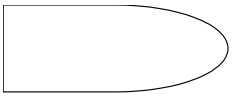
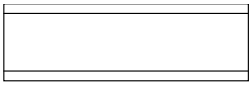
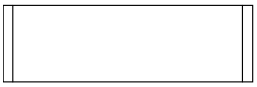
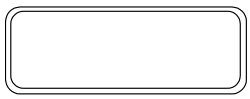
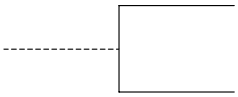
Элемент GFT	Символ	Описание
Символ встроенного выражения		Используется для оператора if-else, операторов for, while, do-while, alt, call и interleave TTCN-3, должен быть присоединен к символу компонента
Символ “по умолчанию”		Используется для оператора включения и выключения TTCN-3 должен быть присоединен к символу компонента
Символ остановки		Используется для оператора остановки TTCN-3, должен быть присоединен к символу компонента
Символ возврата		Используется для оператора возврата TTCN-3, должен быть присоединен к символу компонента
Символ повторения		Используется для оператора повторения TTCN-3, должен быть присоединен к символу компонента
Символ создания		Используется для оператора создания TTCN-3, должен быть присоединен к символу компонента
Символ запуска		Используется для оператора запуска TTCN-3, должен быть присоединен к символу компонента
Символ сообщения		Используется для операторов send, call, reply, raise, receive, getcall, getreply, catch, trigger и check TTCN-3, должен быть присоединен к символу компонента и символу порта
Символ обнаружения		Используется для представления операторов receive, getcall, getreply, catch, trigger и check TTCN-3 из любого порта, должен быть присоединен к символу компонента
Символ области приостановки		Используется в сочетании с блокирующим вызовом, должен находиться во встроенном выражении вызова и быть присоединен к символу компонента
Символ запуска таймера		Используется для операции запуска таймера TTCN-3, должен быть присоединен к символу компонента
Символ тайм-аута таймера		Используется для операции тайм-аута TTCN-3, должен быть присоединен к символу компонента
Символ остановки таймера		Используется для операции остановки таймера TTCN-3, должен быть присоединен к символу компонента

Таблица 1/Z.142 – Символы GFT

Элемент GFT	Символ	Описание
Символ запуска скрытого таймера		Используется для запуска скрытого таймера TTCN-3 в блокирующем вызове, должен быть во встроенной строке вызова и присоединен к символу компонента
Символ тайм-аута скрытого таймера		Используется для исключения тайм-аута TTCN-3 в блокирующем вызове, должен быть во встроенной строке вызова и присоединен к символу компонента
Символ выполнения		Используется для оператора выполнения тестового случая TTCN-3, должен быть присоединен к символу компонента
Текстовый символ		Используется для TTCN-3 с оператором и комментариями, должен быть размещен в диаграмме GFT
Символ комментария к событию		Используется для комментариев TTCN-3, связанных с событиями, должен быть присоединен к событиям на уровне событий варианта компонента или варианта порта

9 Диаграммы GFT

Формат GFT предоставляет следующие типы диаграмм:

a) *диаграмму управления* для графического представления части управления модуля TTCN-3;

b) *диаграмму тестового случая* для графического представления тестового случая TTCN-3;

c) *диаграмму альтернативного шага* для графического представления альтернативного шага TTCN-3; и

d) *диаграмму функции* для графического представления функции TTCN-3.

Различные типы диаграмм имеют общие свойства.

9.1 Общие свойства

Общие свойства диаграмм GFT относятся к области диаграммы, заголовку диаграммы и страничной организации.

9.1.1 Область диаграммы

Каждая диаграмма GFT управления, тестового случая, альтернативного шага и функции должна иметь символ кадра (называемый также кадром диаграммы) для определения области диаграммы. Все символы и текст, необходимые для определения полной и синтаксически правильной диаграммы GFT, должны находиться внутри области диаграммы.

ПРИМЕЧАНИЕ. – Формат GFT не имеет таких конструктивных элементов языка, как логические элементы MSC, которые размещаются вне кадра диаграммы, но соединены с ним.

9.1.2 Заголовок диаграммы

Каждая диаграмма GFT должна иметь заголовок диаграммы. Заголовок диаграммы должен быть размещен в верхнем левом углу кадра диаграммы.

Заголовок диаграммы должен однозначно определять каждый тип диаграммы GFT. Общее правило для достижения этого состоит в создании заголовка из ключевых слов **`testcase`**, **`altstep`** или **`function`,** за которыми следует подпись TTCN-3 тестового случая, альтернативного шага или функции, которые должны быть представлены графически. Для диаграммы управления GFT единственный заголовок создается из ключевого слова **`module`,** за которым следует название модуля.

ПРИМЕЧАНИЕ. – В MSC ключевое слово **msc** всегда предшествует названию диаграммы для определения диаграмм MSC. Диаграммы GFT не имеют такого общего ключевого слова для определения диаграмм GFT.

9.1.3 Страничная организация

Диаграммы GFT могут быть организованы по страницам, а большая диаграмма GFT может подразделяться на несколько страниц. Каждая страница разделенной диаграммы должна иметь нумерацию в верхнем правом углу, которая однозначно определяет страницу. Нумерация необязательна, если диаграмма не разделена.

ПРИМЕЧАНИЕ 1. – Конкретная схема нумерации считается вопросом, относящимся к инструментарию, и поэтому находится вне сферы применения настоящей Рекомендации. Простая схема нумерации может только присваивать номер страницы, тогда как усовершенствованная схема нумерации может обеспечивать восстановление диаграммы лишь путем использования информации о нумерации на различных страницах.

ПРИМЕЧАНИЕ 2. – Требования к страничной организации, выходящие за рамки общей нумерации, считаются вопросом, относящимся к инструментарию, и поэтому находятся вне сферы применения настоящей Рекомендации. В целях обеспечения читабельности заголовок диаграммы может быть представлен на каждой странице, линия варианта того или иного варианта, которая переходит на другую страницу, может быть показана у нижнего края страницы, а заголовок варианта продолжаемого варианта может быть повторен на странице, на которой описывается продолжение.

9.2 Диаграмма управления

Диаграмма управления GFT обеспечивает графическое представление части управления модуля TTCN-3. Заголовком диаграммы управления должно быть ключевое слово **module,** после которого следует название модуля. Диаграмма управления GFT должна включать только один вариант компонента (называемый также вариантом управления) с названием **control** без какой-либо информации о типе. Вариант управления описывает поведение части управления модуля TTCN-3. Атрибуты, связанные с частью управления модуля TTCN-3, указываются в текстовом символе диаграммы управления. Принципиальный вид диаграммы управления GFT и соответствующее базовое описание TTCN-3 схематично изображены на рисунке 9.

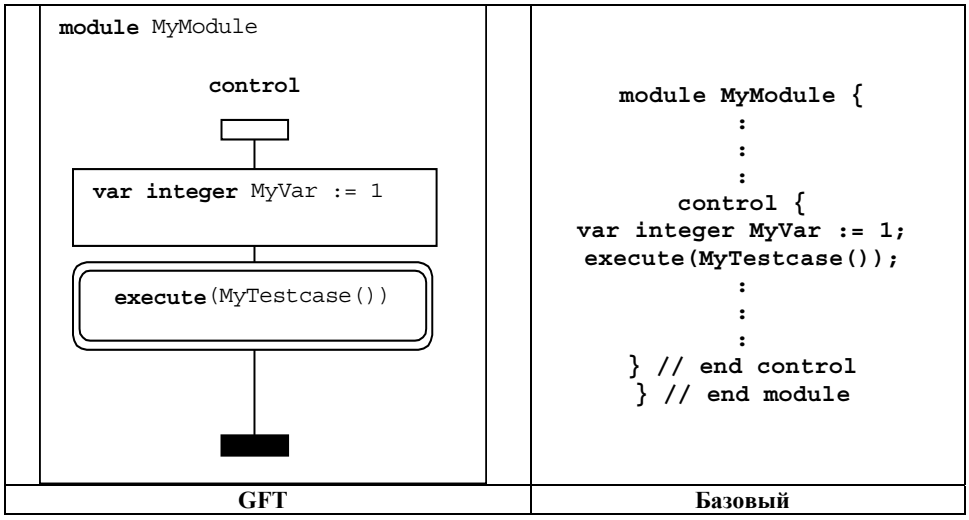


Рисунок 9/Z.142 – Принципиальный вид диаграммы управления GFT и соответствующий базовый язык

В части управления выбор или отмена выбора тестовых случаев для их выполнения могут осуществляться с использованием булевых выражений. Для управления выполнением тестовых случаев могут использоваться выражения, присвоения, операторы **log**, операторы **label** и **goto**, операторы **if-else**, операторы цикла **for**, операторы цикла **while**, операторы цикла **do while**, операторы выполнения **stop** и операторы таймеров. Кроме того, функции, которые запускаются частью управления модуля, могут использоваться для группирования тестовых случаев с их входными условиями выполнения.

Представление в формате GFT этих свойств языка осуществляется как описано в соответствующих пунктах ниже за исключением того, что для части управления модуля графические символы присоединяются к варианту управления, а не к варианту тестового компонента.

Представление в формате GFT выражений, присвоений, **log**, **label** и **goto**, **if-else**, цикла **for**, цикла **while**, цикла **do while** и **stop** см. в 11.4, операции таймеров – в п. 11.9, функции и их запуск – в пп. 9.4 и 11.2.2.

9.3 Диаграмма тестового случая

Диаграмма тестового случая GFT обеспечивает графическое представление тестового случая TTCN-3. Заголовком диаграммы тестового случая должно быть ключевое слово **testcase**, после которого следует полная подпись тестового случая. "Полная" означает, что должны быть представлены, по крайней мере, название тестового случая и список параметров. В базовом языке раздел **runs on** является обязательным, а раздел **system** необязательным. Если оператор системы определен в соответствующем базовом языке, он представляется также в заголовке диаграммы тестового случая.

Диаграмма GFT тестового случая должна включать один вариант тестового компонента, описывающий поведение **mtc** (называемый также вариантом mtc), и один вариант порта для каждого порта, принадлежащего **mtc**. Названием, связанным с вариантом mtc, должно быть **mtc**. Тип, связанный с вариантом mtc, является необязательным, однако если информация о нем имеется, то он должен быть идентичен типу компонента, упоминаемому в разделе **runs on** подписи тестового случая. Названия, связанные с вариантами портов, должны быть идентичны названиям портов, определенным в определении типа компонента **mtc**. Соответствующая информация о типе для вариантов портов является необязательной. Если имеется информация о типе, то названия и типы портов должны соответствовать определению типа компонента **mtc**. Типы **mtc** и портов отображаются в символе заголовка варианта компонента или порта.

Атрибуты, которые связаны с тестовым случаем, представленным в GFT, должны быть указаны в текстовом символе в диаграмме тестового случая. Принципиальный вид диаграммы тестового случая GFT и соответствующее базовое описание TTCN-3 схематично показаны на рисунке 10.

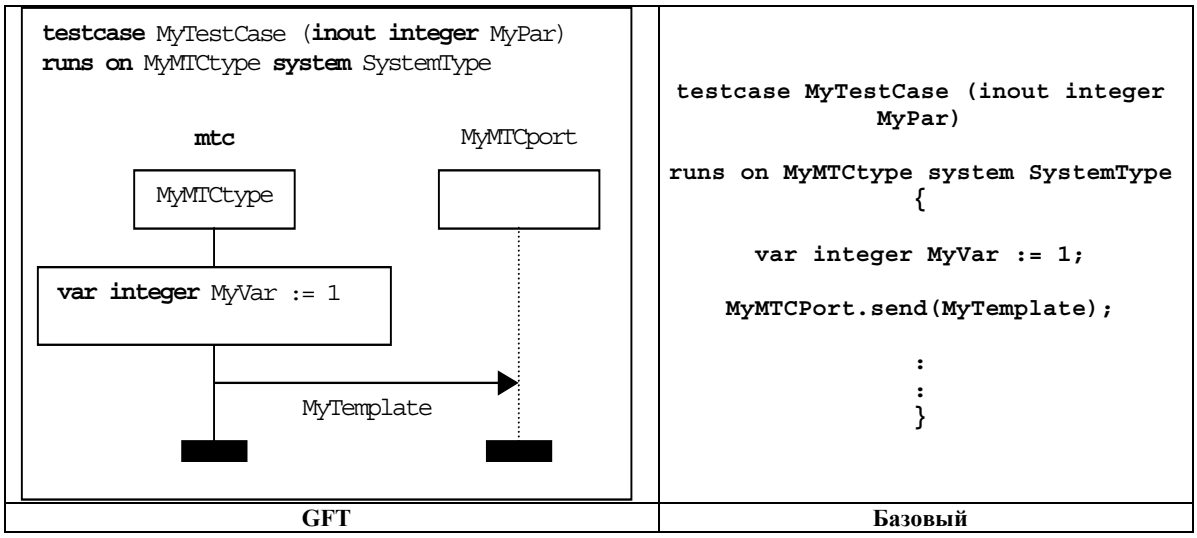


Рисунок 10/Z.142 – Принципиальный вид диаграммы тестового случая GFT и соответствующий базовый язык

Тестовый случай представляет собой динамическое поведение теста и может создавать тестовые компоненты. Тестовый случай может включать объявления, операторы, операции связи и таймеров и запуск функций или альтернативных шагов.

9.4 Диаграмма функции

В формате GFT функции TTCN-3 представляются посредством диаграмм функций. Заголовок диаграммы функции должен быть ключевым словом **function,** после которого следует полная подпись функции. "Полная" означает, что должны быть представлены, по крайней мере, название функции и список параметров. Раздел **return** и раздел **runs on** являются необязательными в базовом языке. Если эти разделы указаны в соответствующем базовом языке, они также должны быть представлены в заголовке диаграммы функции.

Диаграмма функции GFT должна включать один вариант тестового компонента, описывающий поведение функции и один вариант порта для каждого порта, используемого функцией.

ПРИМЕЧАНИЕ. – Названия и типы портов, которые используются функцией, передаются в виде параметров или являются названиями и типами портов, которые описаны в определении типа компонента, упоминаемом в разделе **runs on**.

Названием, связанным с вариантом тестового компонента, должно быть **self**. Тип, связанный с вариантом тестового компонента, является необязательным, однако если информация о нем имеется, то он должен соответствовать типу компонента в разделе **runs on**.

Названия и типы, связанные с вариантами портов, должны соответствовать параметрам портов (если используемые порты передаются в виде параметров) или объявлениям портов в определении типа компонента, упомянутого в разделе **runs on**. Информация о типе для вариантов портов является необязательной.

Названия портов и **Self** отображаются поверх символа заголовка варианта компонента и соответствующего варианта порта. Типы компонентов и типы портов отображаются в символе заголовка варианта компонента и соответствующего варианта порта.

Атрибуты, связанные с функцией, которая представлена в GFT, должны быть указаны в текстовом символе в диаграмме функции. Принципиальный вид диаграммы функции GFT и соответствующее базовое представление TTCN-3 схематично показаны на рисунке 11.

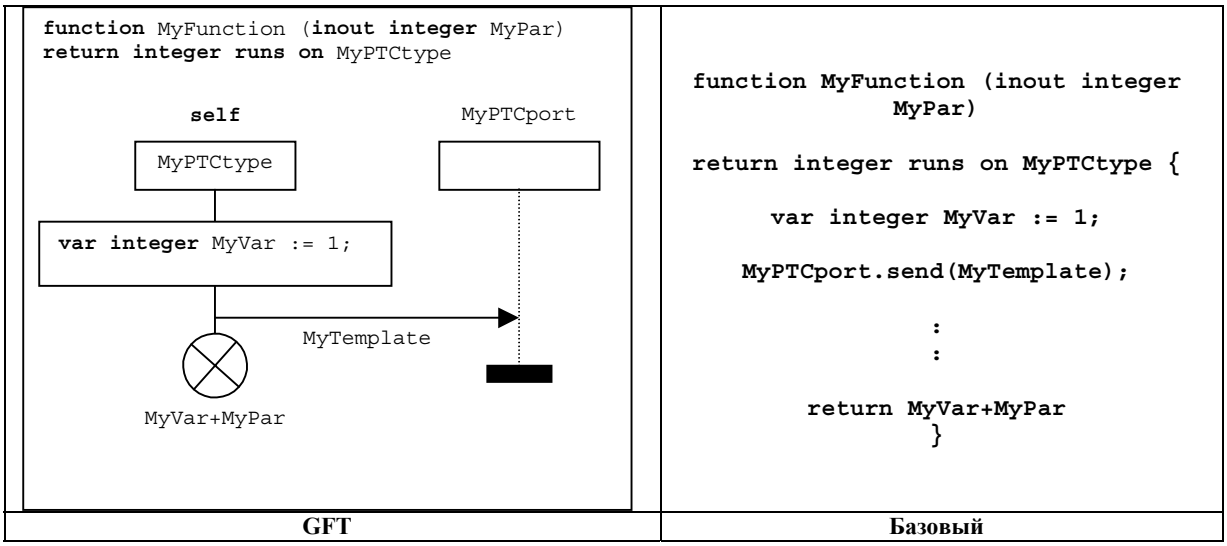


Рисунок 11/Z.142 – Принципиальный вид диаграммы функции GFT и соответствующий базовый язык

Та или иная функция используется, чтобы определять и структурировать поведение тестов, определять поведение по умолчанию или структурировать вычисление в модуле. Функция может включать объявления, операторы, операции связи и таймеров, запуск функций или альтернативных шагов и дополнительный оператор возврата.

9.5 Диаграмма альтернативных шагов

В формате GFT альтернативные шаги TTCN-3 представляются посредством диаграмм альтернативных шагов. Заголовок диаграммы альтернативного шага должен быть ключевым словом **altstep,** после которого следует полная подпись альтернативного шага. "Полная" означает, что должны быть представлены, по крайней мере, название альтернативного шага и список параметров. Раздел **runs on** является необязательным в базовом языке. Если раздел **runs on** указан в соответствующем базовом языке, он также должны быть представлен в заголовке диаграммы альтернативного шага.

Диаграмма альтернативного шага GFT должна включать один вариант тестового компонента, описывающий поведение альтернативного шага и один вариант порта для каждого порта, используемого альтернативным шагом.

ПРИМЕЧАНИЕ. – Названия и типы портов, которые используются альтернативным шагом, передаются в виде параметров или являются названиями и типами портов, которые описаны в определении типа компонента, упоминаемом в разделе **runs on**.

Названием, связанным с вариантом тестового компонента, должно быть **self**. Тип, связанный с вариантом тестового компонента, является необязательным, однако если информация о нем имеется, то он должен соответствовать типу компонента в разделе **runs on**.

Названия и типы, связанные с вариантами портов, должны соответствовать параметрам портов (если используемые порты передаются в виде параметров) или объявлениям портов в определении типа компонента, упомянутого в разделе **runs on**. Информация о типе для вариантов портов является необязательной.

Названия портов и **Self** отображаются поверх символа заголовка варианта компонента и соответствующего варианта порта. Типы компонентов и типы портов отображаются в символе заголовка варианта компонента и соответствующего варианта порта.

Атрибуты, связанные с альтернативным шагом, должны быть указаны в текстовом символе в диаграмме альтернативного шага GFT. Принципиальный вид диаграммы альтернативного шага GFT и соответствующий базовый язык TTCN-3 схематично показаны на рисунке 12.

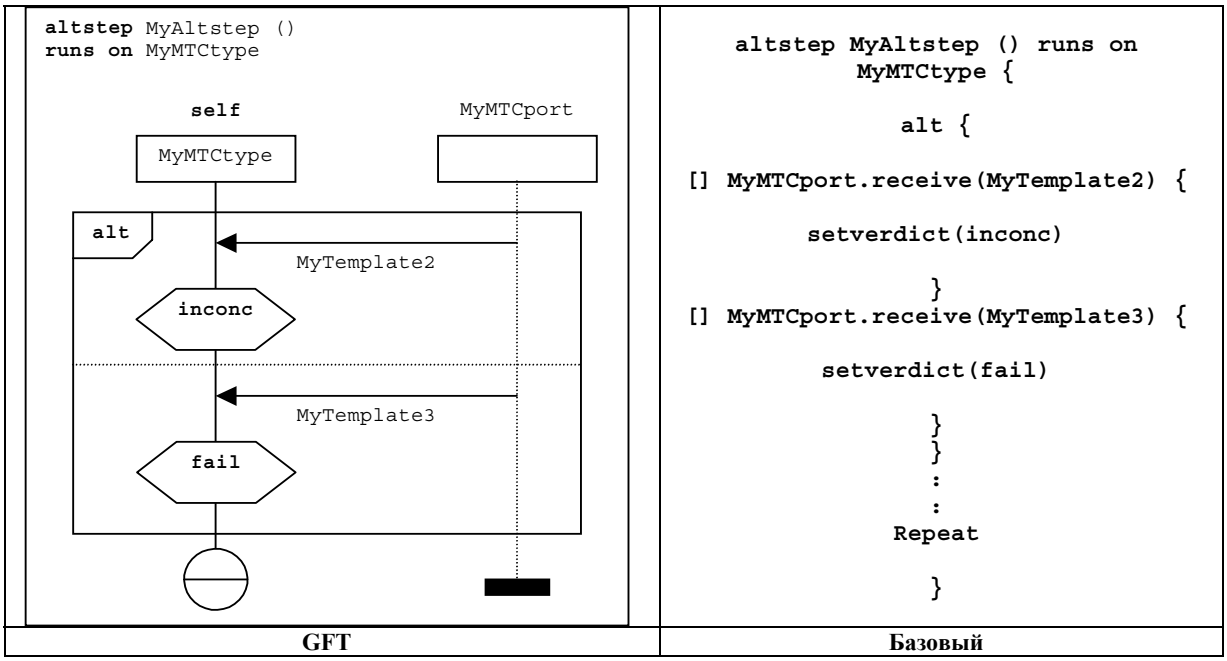


Рисунок 12/Z.142 – Принципиальный вид диаграммы альтернативного шага GFT и соответствующий базовый язык

Альтернативный шаг используется для определения поведения по умолчанию или для структурирования альтернатив оператора **alt**. Альтернативный шаг может включать операторы, операции связи и таймеров и запуск функции или альтернативных шагов.

10 Варианты в диаграммах GFT

Диаграммы GFT включают следующие виды вариантов:

- *варианты управления*, описывающие поток управления в части управления модулем;
- *варианты тестовых компонентов*, описывающие поток управления для тестового компонента, выполняющего тестовый случай, функцию или альтернативный шаг;
- *варианты портов*, представляющие порты, используемые различными тестовыми компонентами.

10.1 Вариант управления

В диаграмме управления GFT должен существовать только один вариант управления (см. п. 9.2). Вариант управления описывает поток управления части управления модуля. Вариант управления GFT должен описываться графически символом варианта компонента с обязательным названием **control**, помещенном сверху символа заголовка варианта. Информация о типе варианта не связана с вариантом управления. Принципиальный вид примера управления представлен на рисунке 13 а).

10.2 Варианты тестовых компонентов

Каждая диаграмма GFT тестового случая, функции или альтернативного шага включает один вариант тестового компонента, который описывает поток управления этого варианта. Вариант тестового компонента GFT должен описываться графически символом варианта с:

- обязательным названием **mtc**, размещенном сверху символа заголовка варианта, в случае диаграммы тестового случая;
- обязательным названием **self,** размещенном сверху символа заголовка варианта, в случае диаграммы функции или альтернативного шага.

Необязательный тип тестового компонента может быть предоставлен в символе заголовка варианта. Он должен соответствовать типу тестового компонента, который дается после ключевого слова **runs on** в заголовке диаграммы GFT.

Принципиальный вид варианта тестового компонента в диаграмме тестового случая представлен на рисунке 13 b). Принципиальный вид варианта тестового компонента в диаграмме функции или альтернативного шага показан на рисунке 13 c).

10.3 Варианты портов

Варианты портов GFT могут использоваться в диаграммах тестового случая, альтернативного шага и функции. Вариант порта представляет собой порт, который используется тестовым компонентом, выполняющим указанный тестовый случай, альтернативный шаг или функцию. Вариант порта GFT графически описывается символом варианта компонента со штриховой линией варианта. Название представляемого порта является обязательной информацией, которая должна размещаться сверху символа заголовка варианта. Тип порта (необязательный) может быть предоставлен в символе заголовка варианта. Принципиальный вид варианта порта показан на рисунке 13 d).

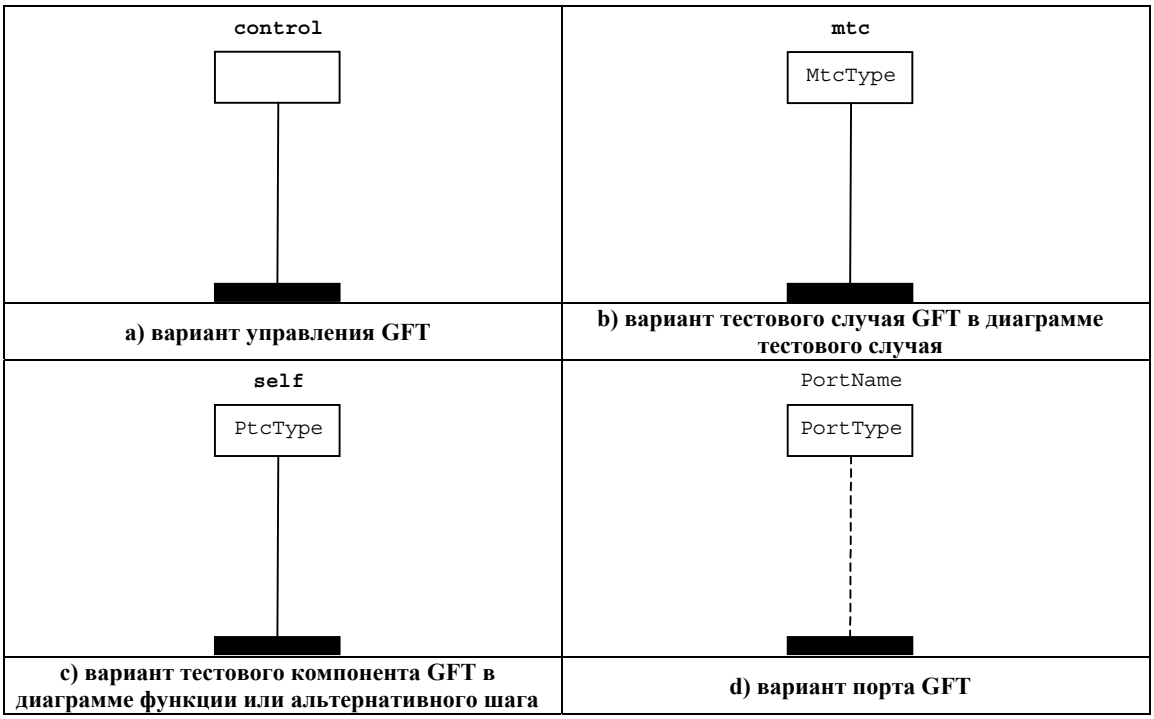


Рисунок 13/Z.142 – Принципиальная форма видов вариантов в диаграммах GFT

11 Диаграммы GFT элементов

В настоящем пункте определяются общие правила изображения для представления конкретных элементов синтаксиса TTCN-3 (точки с запятыми, комментарии). В нем описывается, каким образом отображается выполнение диаграмм GFT и графических символов, связанных с элементами языка TTCN-3.

11.1 Общие правила изображения

Общие правила изображения в формате GFT касаются использования точек с запятыми, операторов TTCN-3 в символах действия и комментариях.

11.1.1 Использование точек с запятыми

Все символы GFT за исключением символа действия должны включать только один оператор в базовом языке TTCN-3. Только символ действия может включать последовательность операторов TTCN-3 (см. п. 11.1.2).

Точка с запятой необязательна, если символ GFT включает только один оператор в базовом языке TTCN-3 (см. рисунки 14 a) и 14 b)).

Точки с запятыми должны разделять операторы в последовательности операторов в рамках символа действия. Точка с запятой необязательна для последнего оператора в последовательности (рисунок 14 c)).

Последовательность объявлений переменных, констант и таймеров может быть также указана в простом базовом языке TTCN-3 после заголовка диаграммы GFT. Точки с запятыми должны также разделять эти объявления. Точка с запятой необязательна для последнего объявления в этой последовательности.

11.1.2 Использование символов действия

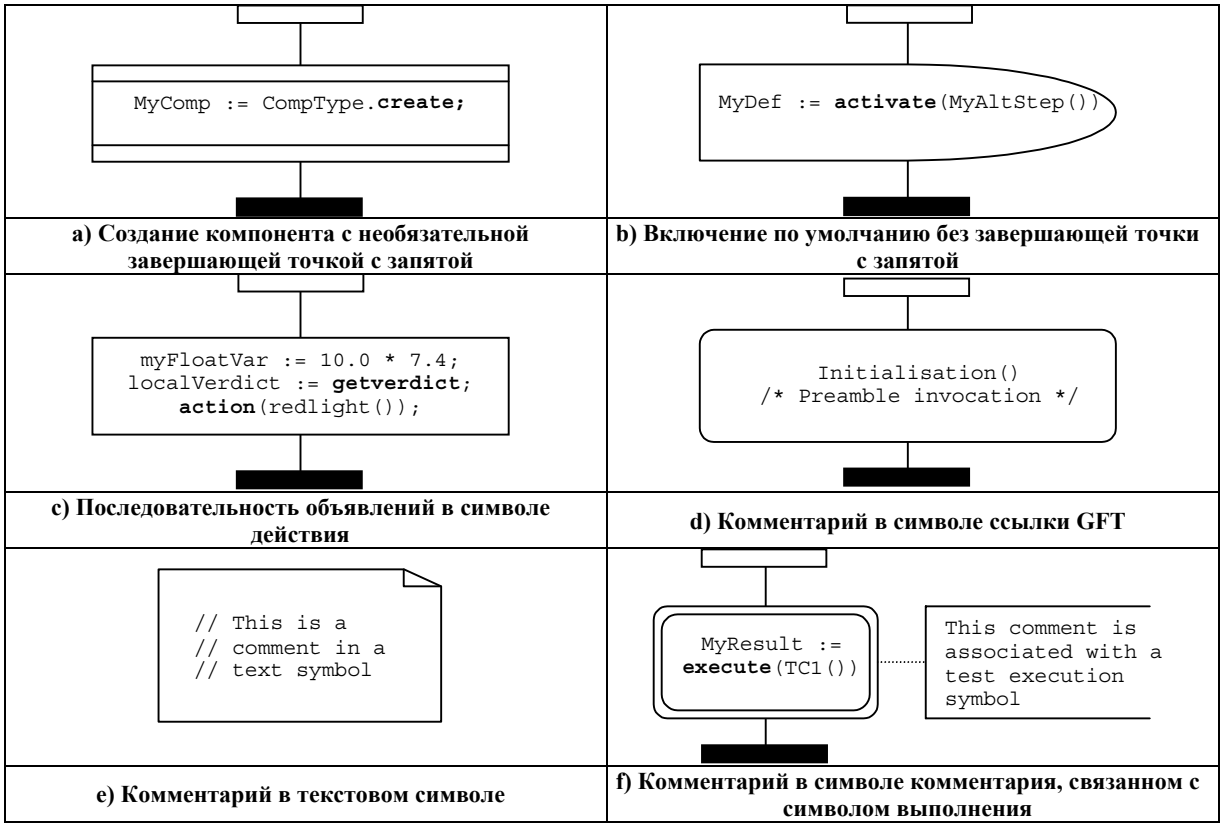
В символах действия указываются следующие объявления, операторы и операции TTCN-3: объявления (с ограничением, определенном в п. 11.3), присвоения, **`log`**, **`connect`**, **`disconnect`**, **`map`**, **`unmap`** и **`action`**.

Последовательность объявлений, операторов и операций, которая должна указываться в различных символах действий, может быть указана в одном символе действия. Нет необходимости использовать отдельный символ действия для каждого объявления, оператора или операции.

11.1.3 Комментарии

Формат GFT предоставляет три возможности для размещения комментариев в диаграммах GFT:

- Комментарии могут быть размещены в символах GFT после записи символа с использованием синтаксиса для комментариев базового языка TTCN-3 (рисунок 14 d)).
- Комментарии, созданные в синтаксисе для комментариев базового языка TTCN-3, могут быть размещены в текстовых символах и помещены в любом месте области диаграммы GFT (рисунок 14 e)).
- Символ комментария может использоваться для увязки комментариев с символами GFT. Тот или иной комментарий в символе комментария может быть предоставлен в форме свободного текста, т. е. не следует использовать разделитель комментария "/*", "*/" и "//" базового языка (рисунок 14 f)).



a) Создание компонента с необязательной завершающей точкой с запятой

b) Включение по умолчанию без завершающей точки с запятой

c) Последовательность объявлений в символе действия

d) Комментарий в символе ссылки GFT

e) Комментарий в текстовом символе

f) Комментарий в символе комментария, связанном с символом выполнения

Рисунок 14/Z.142 – Примеры результатов применения общих правил изображения

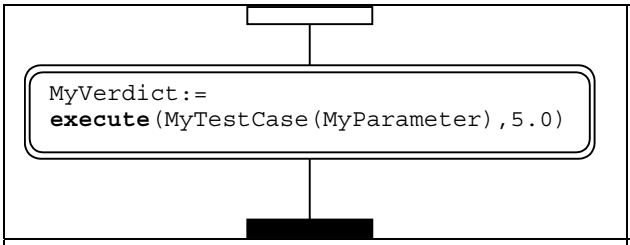
11.2 Запуск диаграмм GFT

В настоящем пункте описывается, каким образом запускаются отдельные виды диаграмм GFT. Поскольку не существует оператора для выполнения части управления в TTCN-3 (так как это сравнимо с выполнением программы через основной модуль и не входит в сферу применения TTCN-3), в этом пункте рассматривается выполнение тестовых случаев, функций и альтернативных шагов.

11.2.1 Выполнение тестовых случаев

Выполнение тестовых случаев представляется путем использования символа выполнения тестового случая (см. рисунок 15). Синтаксис оператора **execute** размещается в этом символе. Данный символ может включать:

- оператор **execute** для тестового случая с дополнительными параметрами и временным контролем;
- дополнительно – присвоение возвращенного заключения переменной **verdicttype**; и
- дополнительно – встроенное объявление переменной **verdicttype**.



GFT	Базовый
	MyVerdict := execute(MyTestCase(MyParameter),5.0);

Рисунок 15/Z.142 – Выполнение тестового случая

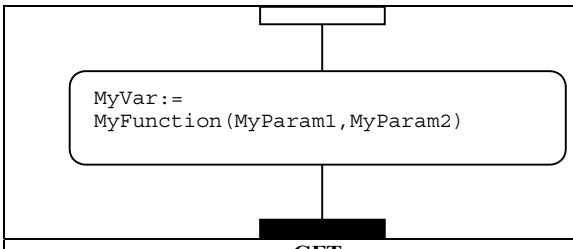
11.2.2 Вызов функций

Вызов функций представляется с помощью символа ссылки (рисунок 16) за исключением внешних и заранее определенных функций (рисунок 17) и за исключением случая, когда функция вызывается внутри элемента языка TTCN-3, имеющего представление в формате GFT (рисунок 18).

Синтаксис вызова функции размещается в символе ссылки. Символ может включать:

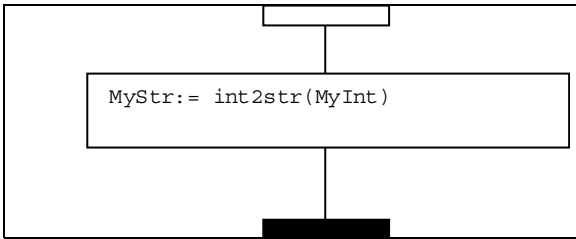
- вызов функции с дополнительными параметрами;
- дополнительное присвоение возвращенной величины переменной величине; и
- дополнительное встроенное объявление переменной.

Символ ссылки используется только для функций, определяемых пользователем в активном модуле. Он не должен использоваться для внешних или предопределенных функций TTCN-3, которые должны представляться в их текстовой форме в символе действия (рисунок 17) или других символах GFT (см. пример на рисунке 18).



GFT	Базовый
	MyVar:= MyFunction(MyParam1,MyParam2);

Рисунок 16/Z.142 – Вызов функции, определяемой пользователем



GFT	Базовый
	MyStr:= int2str(MyInt);

Рисунок 17/Z.142 – Вызов предопределенной/внешней функции

Функции, вызываемые внутри конструктивного элемента TTCN-3 со связанным с ним символом GFT, представляются в этом символе в виде текста.

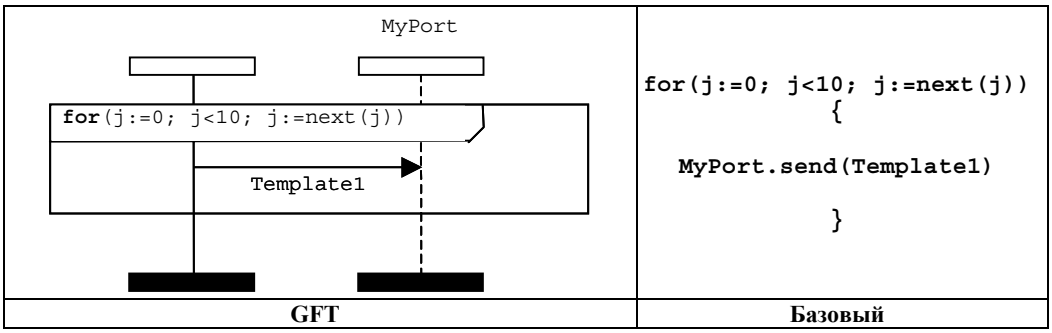


Рисунок 18/Z.142 – Вызов функции, определяемой пользователем, с символом GFT

11.2.3 Вызов альтернативных шагов

Вызов альтернативных шагов представляется путем использования символа ссылки (см. рисунок 19). Синтаксис вызова альтернативного шага размещается в этом символе. Символ может включать вызов альтернативного шага с дополнительными параметрами. Он должен использоваться только в рамках альтернативного поведения в случае, когда вызов альтернативного шага должен быть одним из операндов альтернативных операторов (см. также рисунок 32 в п. 11.5.2).

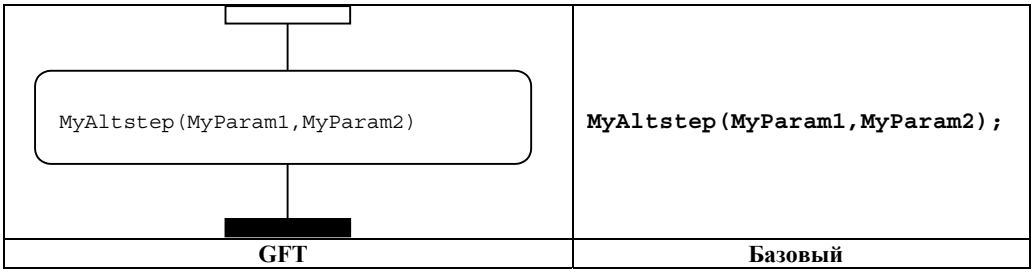


Рисунок 19/Z.142 – Вызов альтернативного шага

Другая возможность состоит в косвенном вызове альтернативных шагов с помощью включенных переменных по умолчанию. Дополнительная информация приведена в п. 11.6.2.

11.3 Объявления

Нотация TTCN-3 позволяет осуществлять объявление и инициализацию таймеров, констант и переменных в начале блоков операторов. Формат GFT использует синтаксис базового языка TTCN-3 для объявлений в нескольких символах. Тип символа зависит от характеристики инициализации; например, переменная типа **`default`**, которая инициализируется с помощью операции **`activate`**, должна указываться в символе по умолчанию (см. п. 11.6).

11.3.1 Объявление таймеров, констант и переменных в символах действия

Следующие объявления должны быть сделаны в символах действия:

- объявления таймеров;
- объявления переменных без инициализации;
- объявления переменных и констант с инициализацией;
  - если инициализация не осуществлена при помощи функций, включающих функции связи; или
  - если объявление является:
    - объявлением типа компоненты, которое не инициализируется с помощью операции **`create`**;
    - объявлением типа **`default`**, которое не инициализируется с помощью операции **`activate`**;
    - объявлением типа **`verdicttype`**, которое не инициализируется с помощью оператора **`execute`**;
    - объявлением простого базового типа;
    - объявлением типа базовой строки;
    - объявлением типа **`anytype`**;
    - объявлением типа порта;
    - объявлением типа **`address`**; или
    - объявлением определяемого пользователем структурированного типа с полями, которые удовлетворяют всем ограничениям, упомянутым в этом подпункте "Объявление переменных и констант с инициализацией".

ПРИМЕЧАНИЕ. – С обзором типов TTCN-3 можно ознакомиться в Таблице 3 в Рекомендации МСЭ-Т Z.140 [1].

Последовательность объявлений, которая должна быть сделана в символах действия, может быть помещена в один символ действия без необходимости осуществления этого в отдельных символах действия. Примеры объявлений в символах действия представлены на рисунках 20 a) и 20 b).

11.3.2 Объявление констант и переменных в символах встроенного выражения

Объявления констант и переменных типа компонента, которые инициализируются в операторах **`if-else`**, **`for`**, **`while`**, **`do-while`**, **`alt`** или **`interleave`**, должны быть представлены в том же символе встроенного выражения.

11.3.3 Объявление констант и переменных в символах создания

Объявления констант и переменных типа компонента, которые инициализируются с помощью операций **`create`**, должны быть сделаны в символе создания. В отличие от объявлений в символах действия, каждое объявление, которое инициализируется с помощью операции **`create`**, должно быть представлено в отдельном символе создания. Пример объявления переменной в символе создания представлен на рисунке 20 c).

11.3.4 Объявление констант и переменных в символах по умолчанию

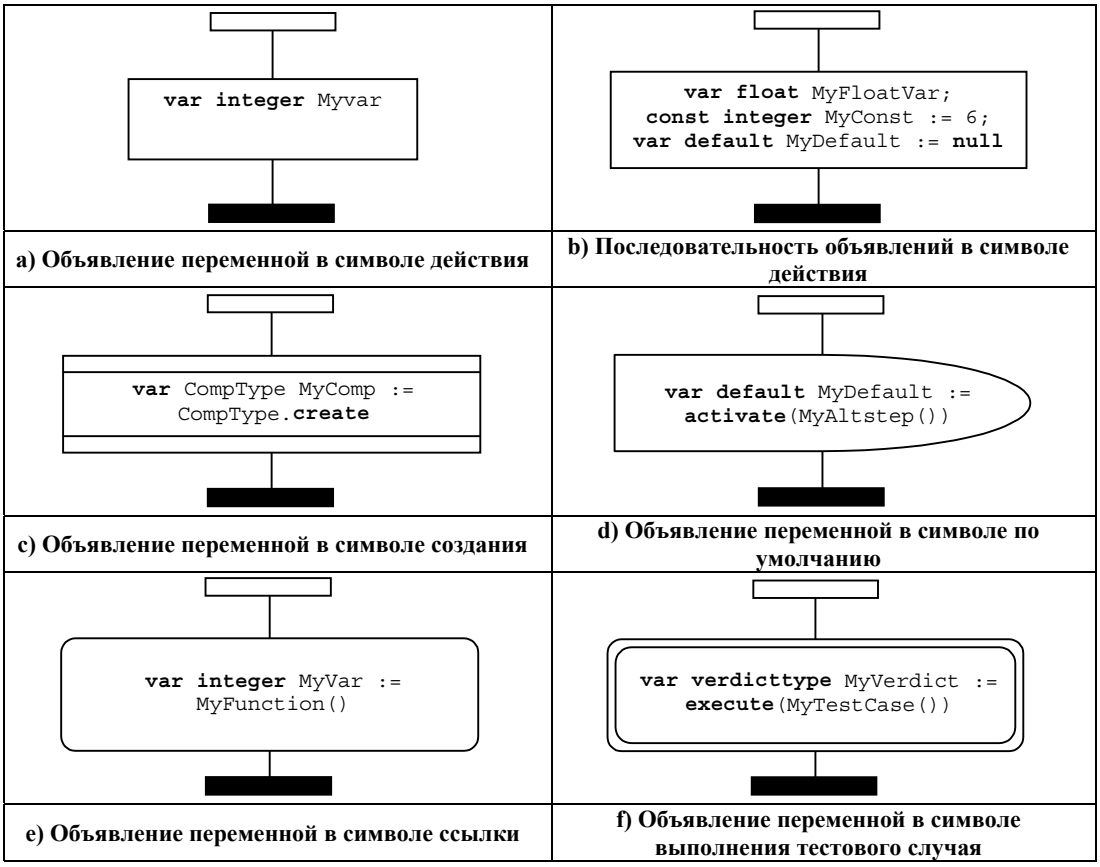
Объявления констант и переменных типа **`default`**, которые инициализируются с помощью операций **`activate`**, должны быть сделаны в символе по умолчанию. В отличие от объявлений в символах действия, каждое объявление, которое инициализируется с помощью операции **`activate`**, должно быть представлено в отдельном символе по умолчанию. Пример объявления переменной в символе по умолчанию представлен на рисунке 20 d).

11.3.5 Объявление констант и переменных в символах ссылки

Объявления констант и переменных, которые инициализируются с помощью операций связи, должны быть сделаны в символах ссылки. В отличие от объявлений в символах действия, каждое объявление, которое инициализируется с помощью функции, которая включает функции связи, должно быть представлено в отдельном символе ссылки. Пример объявления переменной в символе ссылки представлен на рисунке 20 e).

11.3.6 Объявление констант и переменных в символах выполнения тестового случая

Объявления констант и переменных типа **verdicttype**, которые инициализируются с помощью операторов **execute**, должны быть сделаны в символах тестового случая. В отличие от объявлений в символах действия, каждое объявление, которое инициализируется с помощью оператора **execute**, должно быть представлено в отдельном символе тестового случая. Пример объявления переменной в символе выполнения тестового случая представлен на рисунке 20 f).



a) Объявление переменной в символе действия

b) Последовательность объявлений в символе действия

c) Объявление переменной в символе создания

d) Объявление переменной в символе по умолчанию

e) Объявление переменной в символе ссылки

f) Объявление переменной в символе выполнения тестового случая

Рисунок 20/Z.142 – Примеры объявлений в GFT

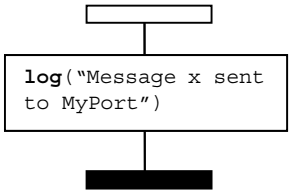
11.4 Базовые операторы программ

Базовыми операторами программ являются выражения, присвоения, операции, конструктивные элементы цикла и др. Все базовые операторы программ могут использоваться с диаграммами GFT для части управления, тестовых случаев, функций и альтернативных шагов.

Формат GFT не предоставляет какого-либо графического представления для выражений и присвоений. Они обозначаются текстом в тех местах, где они используются. Графика предоставляется для операторов **log**, **label**, **goto**, **if-else**, **for**, **while** и **do-while**.

11.4.1 Оператор Log

Оператор **log** должен быть представлен в символе действия (см. рисунок 21).




```
log('Message x sent to MyPort');
```

GFT | **Базовый**

Рисунок 21/Z.142 – Оператор Log

11.4.2 Оператор Label

Оператор **label** должен быть представлен в символе маркирования, соединенным с вариантом компонента. На рисунке 22 приведен простой пример оператора **label** с названием MyLabel.

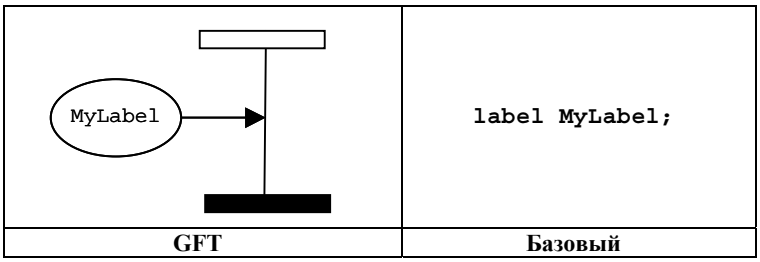


Рисунок 22/Z.142 – Оператор Label

11.4.3 Оператор Goto

Оператор **goto** должен быть представлен в символе goto. Он должен размещаться в конце варианта компонента или в конце операнда во встроенной строке выражения. На рисунке 23 приведен простой пример оператора **goto**.

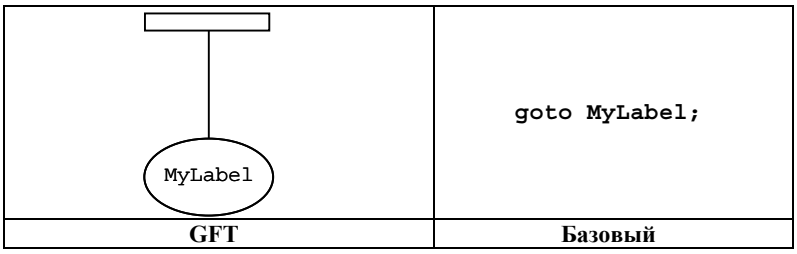


Рисунок 23/Z.142 – Оператор Goto

11.4.4 Оператор If-else

Оператор **if-else** должен быть представлен символом встроенного выражения, маркированным ключевым словом **if** и булевым выражением, как определено в п. 19.6 Рекомендации МСЭ-Т Z.140 [1]. Символ встроенного выражения может содержать один или два операнда, разделенных штриховой линией. Рисунок 24 иллюстрирует оператор **if** с одним операндом, выполняемым, если булево выражение x>1 оценивается как истинное. Рисунок 25 иллюстрирует оператор **if-else**, в котором верхний операнд выполняется, если булево выражение x>1 оценивается как истинное, а нижний операнд выполняется, если булево выражение оценивается как ложное.

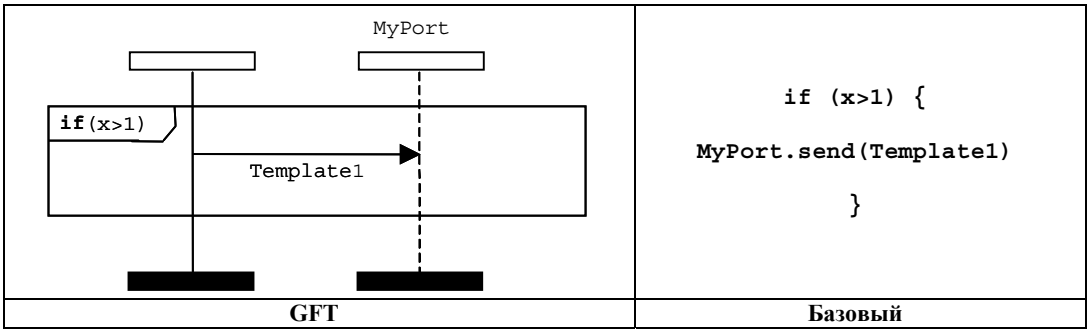


Рисунок 24/Z.142 – Оператор If

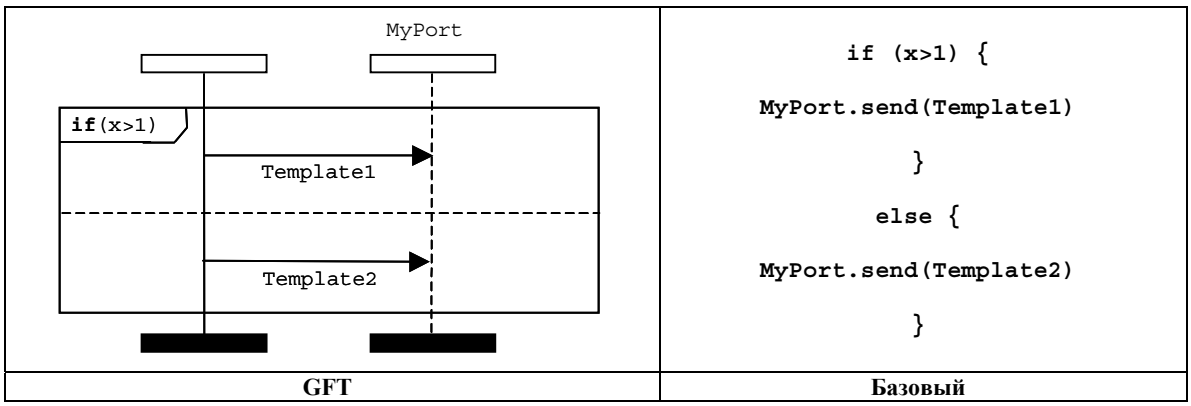


Рисунок 25/Z.142 – Оператор If-else

11.4.5 Оператор For

Оператор **for** должен быть представлен символом встроенного выражения, маркированным определением **for**, определенным в п. 19.7 Рекомендации МСЭ-Т Z.140 [1]. Тело **for** должно быть представлено операндом символа встроенного выражения *for*. На рисунке 26 представлен простой цикл **for**, в котором переменная цикла объявлена и инициализирована в операторе **for**.

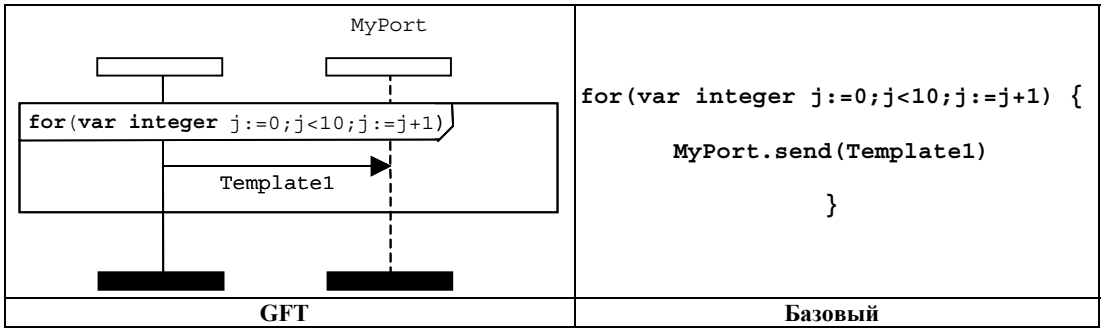


Рисунок 26/Z.142 – Оператор For

11.4.6 Оператор While

Символ **while** должен быть представлен символом встроенного выражения, маркированным определением **while**, определенным в п. 19.8 Рекомендации МСЭ-Т Z.140 [1]. Тело **while** должно быть представлено операндом символа встроенного выражения *while*. На рисунке 27 представлен пример оператора **while**.

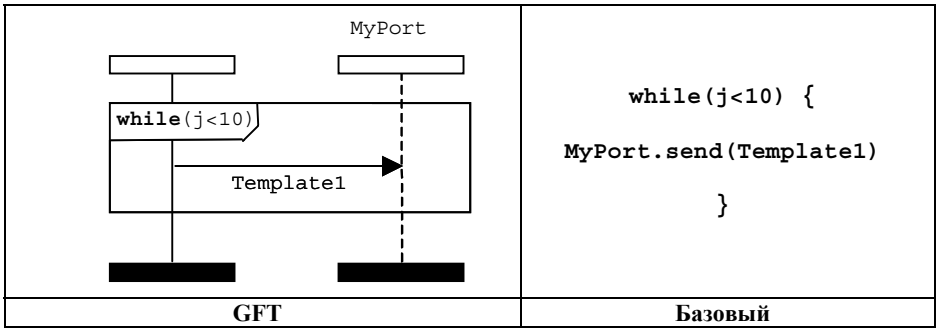


Рисунок 27/Z.142 – Оператор While

11.4.7 Оператор Do-while

Оператор **do-while** должен быть представлен символом встроенного выражения, маркированным определением **do-while**, определенным в п. 19.9 Рекомендации МСЭ-Т Z.140 [1]. Тело **do-while** должно быть представлено операндом символа встроенного выражения *do-while*. На рисунке 28 представлен пример оператора **do-while**.

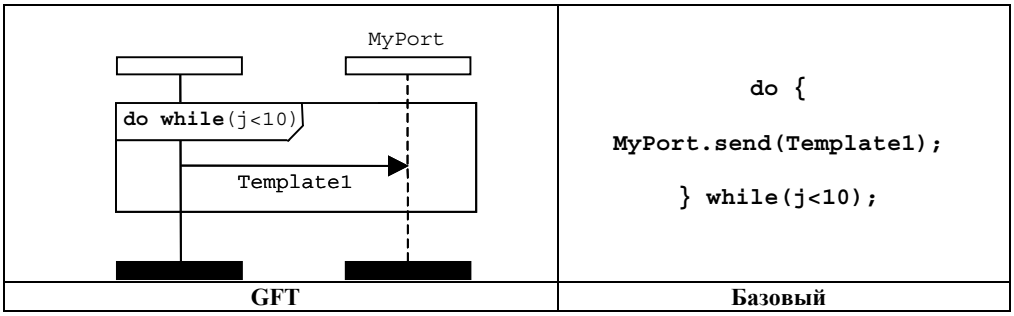


Рисунок 28/Z.142 – Оператор Do-while

11.5 Программные операторы поведения

Операторы поведения могут использоваться в тестовых случаях, функциях и альтернативных шагах, единственным исключением является оператор возврата, который может использоваться только в функциях. Поведение испытания может выражаться последовательно как набор альтернатив или с использованием оператора перемежения. Возврат и повторение используются для управления потоком поведения.

11.5.1 Последовательное поведение

Последовательное поведение представляется порядком событий, расположенных после варианта тестового компонента. События располагаются сверху вниз, причем события, находящиеся ближе к верху символа варианта компонента оцениваются в первую очередь. Рисунок 29 иллюстрирует простой случай, в котором тестовый компонент прежде всего оценивает выражение, содержащееся в символе действия, а затем направляет сообщение в порт `MyPort`.

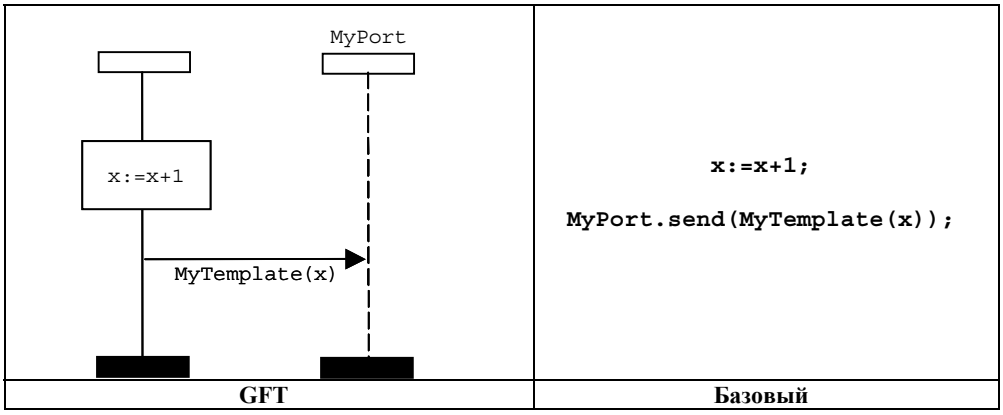


Рисунок 29/Z.142 – Последовательное поведение

Установление последовательности может быть также описано с использованием ссылок на тестовые случаи, функции и альтернативные шаги. В этом случае порядок, в котором располагаются ссылки после оси варианта компонента, определяет порядок их оценивания. На рисунке 30 представлена простая диаграмма GFT, на которой вызывается `MyFunction1`, после которой следует `MyFunction2`.

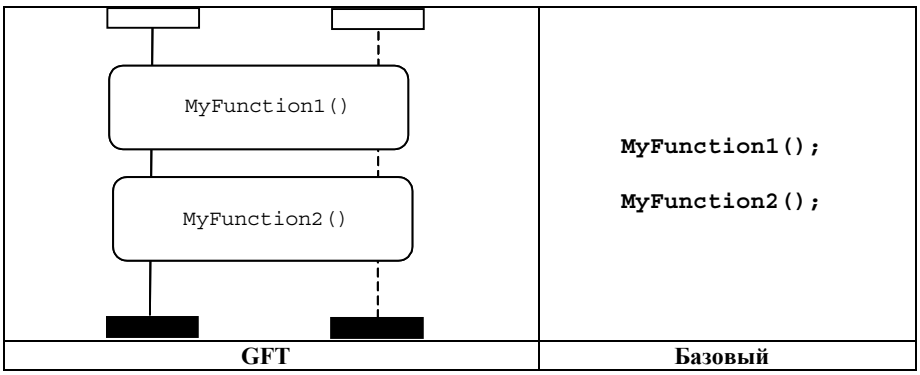


Рисунок 30/Z.142 – Установление последовательности с использованием ссылок

11.5.2 Альтернативное поведение

Альтернативное поведение должно быть представлено с использованием символа встроенного выражения, при этом ключевое слово **`alt`** располагается в верхнем левом углу. Каждый операнд альтернативного поведения должен быть отделен с использованием штриховой линии. Операнды оцениваются в направлении сверху вниз.

Отметим, что альтернативное встроенное выражение должно всегда распространяться на все варианты портов, если задействованы операторы связи. Рисунок 31 иллюстрирует альтернативное поведение, при котором событие сообщения принимается со значением, определяемым `Template1`, или со значением, определяемым `Template2`. Вызов альтернативного шага в альтернативном встроенном выражении представлен на рисунке 32.

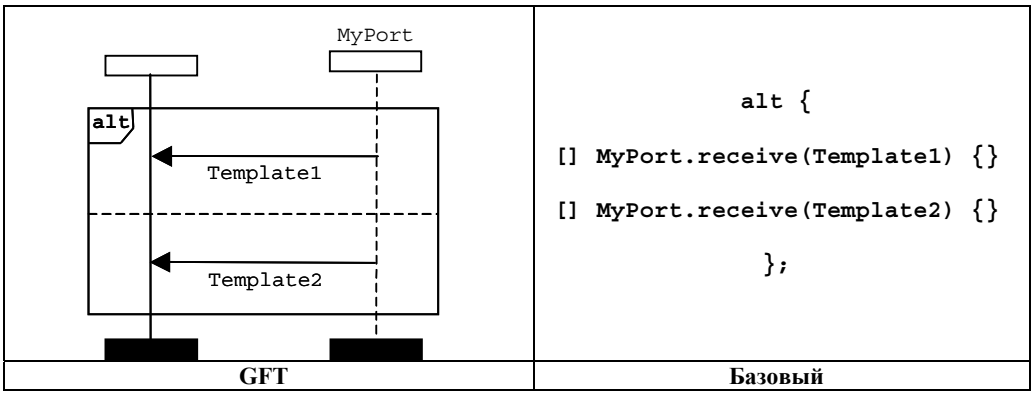


Рисунок 31/Z.142 – Оператор альтернативного поведения

Кроме того, альтернативный шаг можно вызвать только как событие в альтернативном операнде. Это должно изображаться с использованием символа ссылки (см. п. 11.2.3).

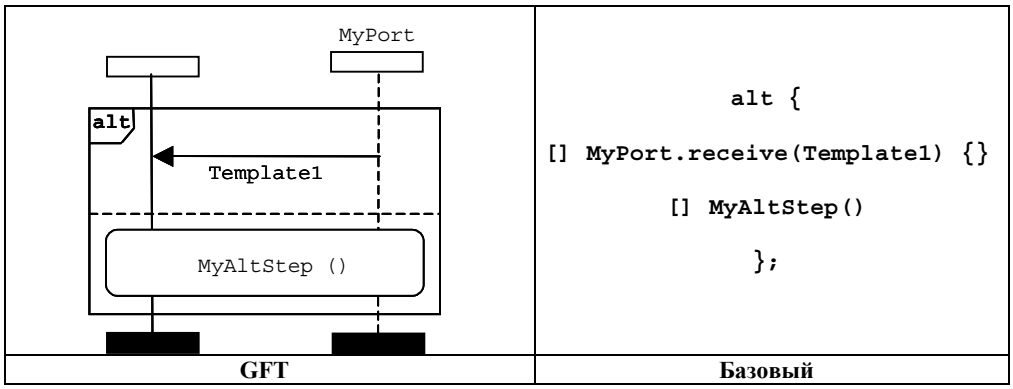


Рисунок 32/Z.142 – Альтернативное поведение с вызовом альтернативного шага

11.5.2.1 Выбор/отмена выбора альтернативы

Можно отменить действие/задействовать альтернативный операнд с помощью булева выражения, содержащего символ условия, который расположен после варианта тестового компонента. На рисунке 33 представлен простой альтернативный оператор, в котором первый вариант защищен выражением `x > 1`, а второй – выражением `x ≤ 1`.

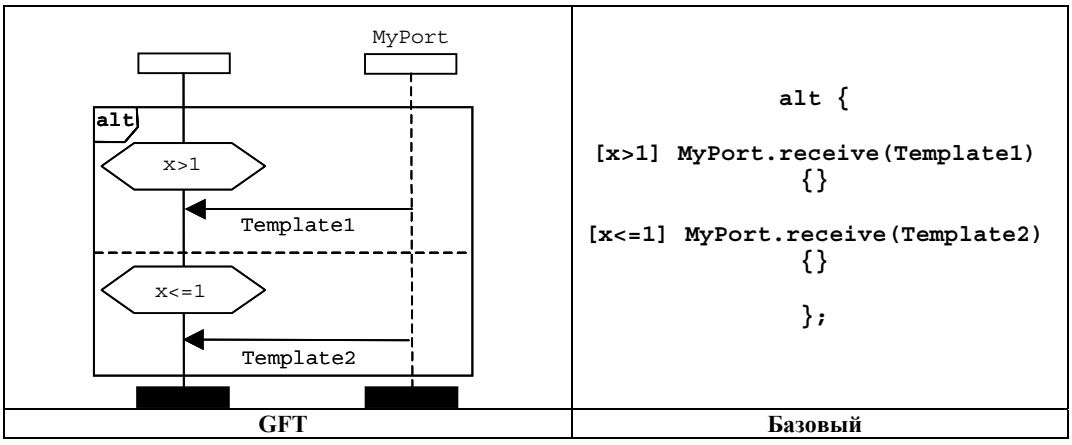


Рисунок 33/Z.142 – Выбор/отмена выбора альтернативы

11.5.2.2 Операция перехода Else в альтернативах

Операция перехода **else** должна обозначаться с использованием символа условия, расположенного после оси варианта тестового компонента и маркированного ключевым словом **else**. На рисунке 34 представлен простой оператор альтернативы в случае, когда второй операнд представляет оператор перехода **else**.

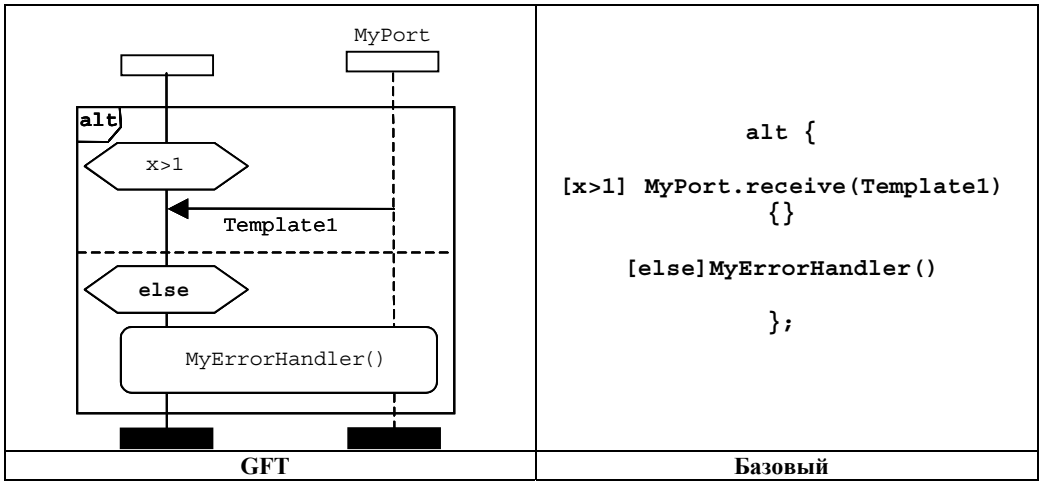


Рисунок 34/Z.142 – Else в альтернативе

Отметим, что символ ссылки в операторе перехода else должен всегда распространяться на все варианты портов, если задействованы операции связи.

Повторная оценка оператора alt может указываться с использованием оператора повтора, представленного символом повтора (см. п. 11.5.3).

Вызов альтернативных шагов в альтернативах представляется с использованием символа ссылки (см. п. 11.2.3).

11.5.3 Оператор повтора

Оператор **repeat** должен представляться символом повтора. Этот символ должен использоваться только как последнее событие операнда альтернативы в операторе **alt** или как последнее событие операнда самой верхней альтернативы в определении альтернативного шага. На рисунке 35 представлен оператор альтернативы, в котором второй операнд после успешного приема сообщения со значением, соответствующим `Template2`, вызывает повтор альтернативы.

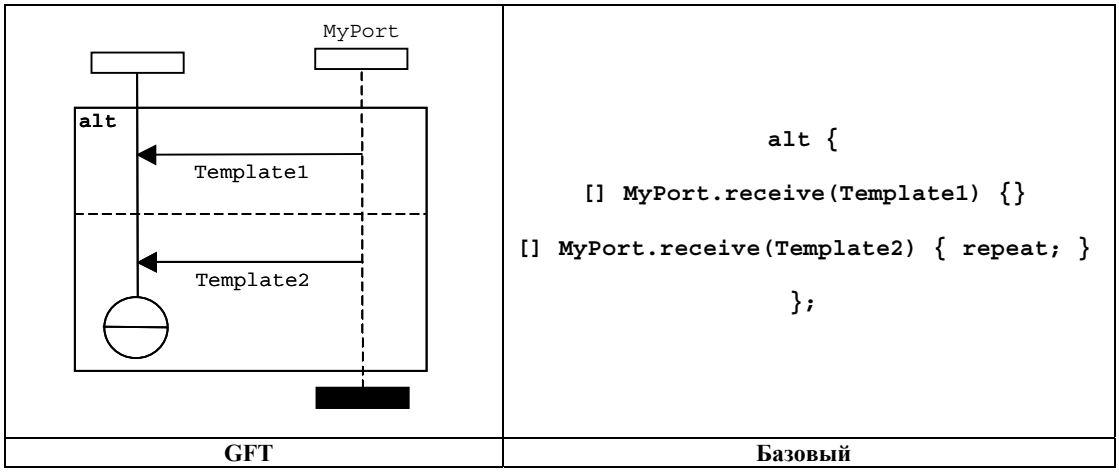
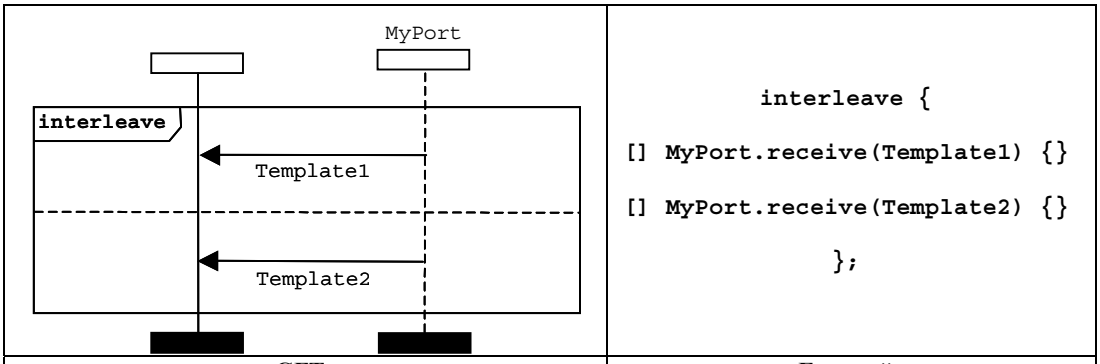


Рисунок 35/Z.142 – Повтор в альтернативе

11.5.4 Поведение перемежения

Поведение перемежения должно быть представлено с использованием символа встроенного выражения с ключевым словом **interleave**, расположенным в верхнем левом углу (см. рисунок 36). Каждый операнд должен быть отделен с использованием штриховой линии. Операторы оцениваются в направлении сверху вниз.



ПРИМЕЧАНИЕ. – Встроенное выражение перемежения всегда должно распространяться на все варианты портов, если задействованы операторы связи.

Рисунок 36/Z.142 – Оператор перемежения

11.5.5 Оператор возврата

Оператор **return** должен представляться символом возврата. Он может быть при желании связан со значением возврата. Символ возврата должен использоваться только в диаграмме функции GFT. Он должен использоваться только в качестве последнего события варианта компонента или в качестве последнего события операнда в символе встроенного выражения. На рисунке 37 показана простая функция без возвращения величины с использованием оператора возврата, а на рисунке 38 – функция, которая возвращает величину.

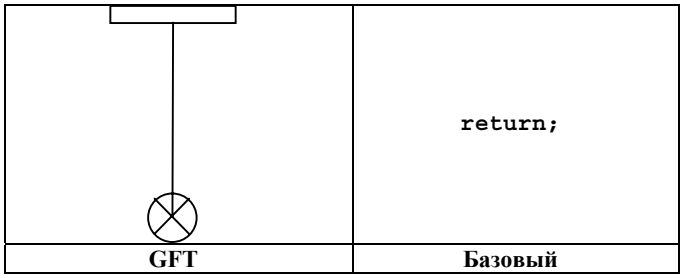


Рисунок 37/Z.142 – Символ возврата без величины возврата

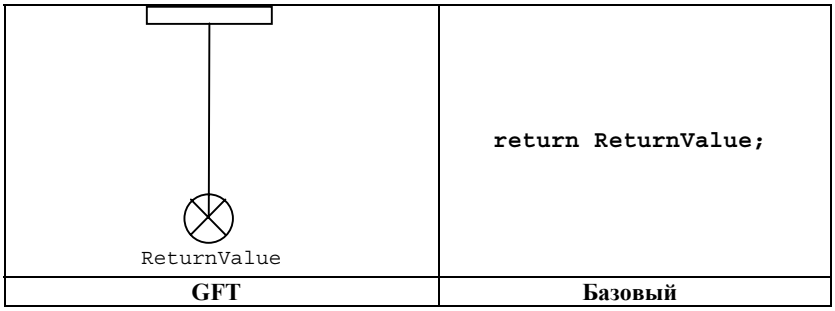


Рисунок 38/Z.142 – Символ возврата с величиной возврата

11.6 Обработка по умолчанию

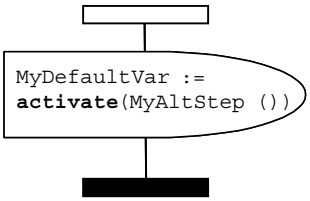
Формат GFT обеспечивает графическое представление активирования и отключения значений по умолчанию (см. пункт 21 Рекомендации МСЭ-Т Z.140 [1]).

11.6.1 Ссылки по умолчанию

Переменные типа **default** могут быть объявлены в символе действия или в символе по умолчанию в составе оператора включения. В пунктах 11.3.1 и 11.3.4 показывается, каким образом переменная, называемая MyDefaultType, объявляется в GFT.

11.6.2 Операция активирования

Активирование значений по умолчанию должно быть представлено путем размещения оператора **activate** в символе по умолчанию (см. рисунок 39).



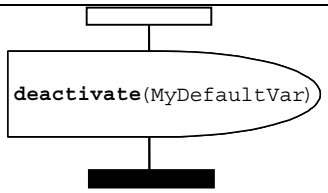

```
MyDefaultVar:=activate(MyAltStep());
```

GFT	Базовый

Рисунок 39/Z.142 – Активирование значений по умолчанию

11.6.3 Операция отключения

Отключение значений по умолчанию должно быть представлено путем размещения оператора **deactivate** в символе по умолчанию (см. рисунок 40). Если оператору **deactivate** не передаются операнды, то тогда все значения по умолчанию отключаются.




```
deactivate(MyDefaultVar);
```

GFT	Базовый

Рисунок 40/Z.142 – Отключение значений по умолчанию

11.7 Операции конфигурирования

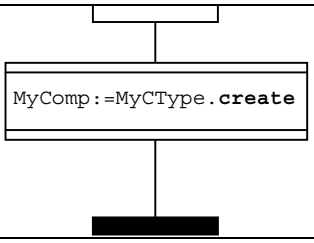
Операции конфигурирования используются для установки тестовых компонентов и управления ими. Эти операции должны использоваться только для диаграмм тестового случая GFT, функции и альтернативного шага.

Операции **mtc**, **self** и **system** не имеют графического представления; они обозначаются с помощью текста в местах их использования.

Формат GFT не обеспечивает какого-либо представления текущей операции (являющейся булевым выражением). Она обозначаются с помощью текста в местах ее использования.

11.7.1 Операция создания

Операция **create** должна быть представлена в символе создания, который присоединяется к варианту тестового компонента, который выполняет операцию создания (см. рисунок 41). Символ создания содержит оператор создания.




```
MyComp:=MyCType.create;
```

GFT	Базовый

Рисунок 41/Z.142 – Операция создания

11.7.2 Операции соединения и отображения

Операции **connect** и **map** должны быть представлены в символе блока действия, который присоединен к варианту тестового компонента, выполняющего операцию **connect** или **map** (см. рисунок 42). Символ блока действия содержит оператор **connect** или **map**.

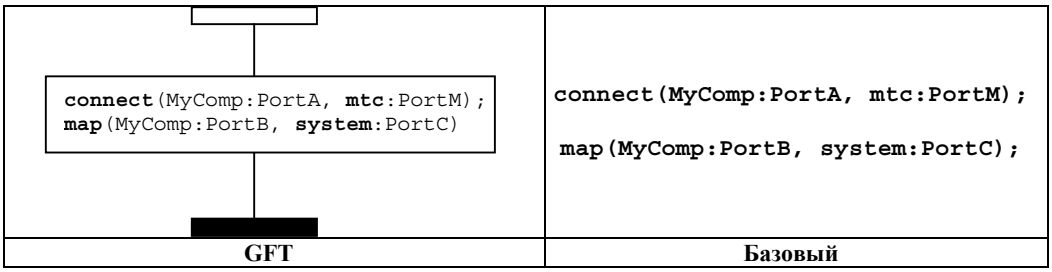


Рисунок 42/Z.142 – Операции соединения и отображения

11.7.3 Операции отключения и прекращения отображения

Операции **disconnect** и **unmap** должны быть представлены в символе блока действия, присоединенного к варианту тестового компонента, который выполняет операцию **disconnect** или **unmap** (см. рисунок 43). Символ блока действия содержит оператор **disconnect** или **unmap**.

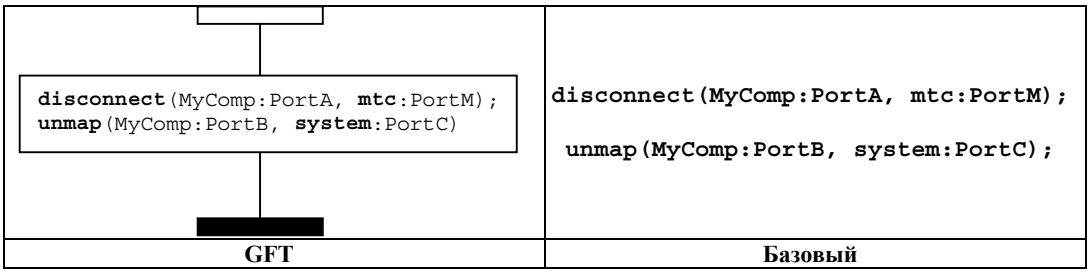


Рисунок 43/Z.142 – Операции отключения и прекращения отображения

11.7.4 Операция запуска тестового компонента

Операция запуска тестового компонента **start** должна быть представлена в символе **start**, присоединенном к варианту тестового компонента, который выполняет операцию **start** (см. рисунок 44). Символ запуска содержит оператор **start**.

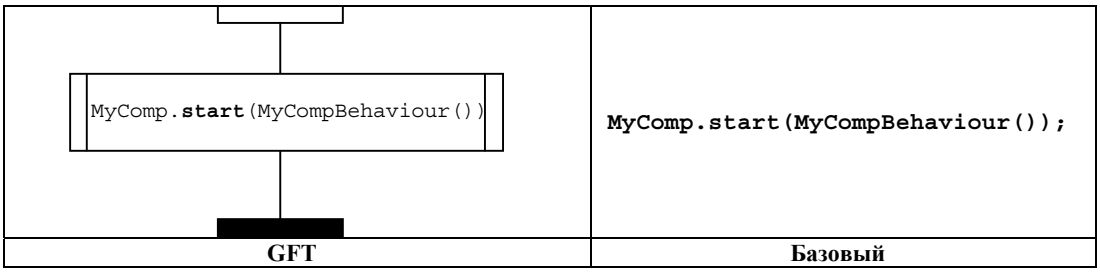


Рисунок 44/Z.142 – Операция запуска

11.7.5 Операции остановки выполнения и остановки тестового компонента

В нотации TTCN-3 имеются две операции остановки: управление модулем и тестовые компоненты могут останавливаться сами путем использования *операции остановки выполнения*, или тестовый компонент может останавливать другие тестовые компоненты путем использования *операции остановки тестовых компонентов*.

Операция выполнения остановки **stop** должна быть представлена символом остановки, присоединенным к варианту тестового компонента, который осуществляет операцию выполнения остановки **stop**

(см. рисунок 45). Она должна использоваться только как последнее событие варианта компонента или как последнее событие операнда символа встроенного выражения.

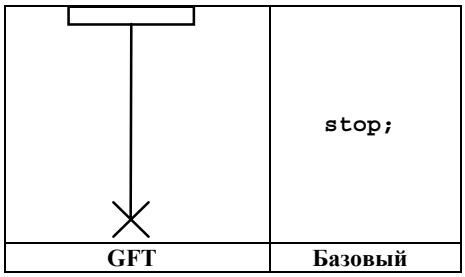


Рисунок 45/Z.142 – Операция выполнения остановки

Операция остановки тестового компонента **stop** должна быть представлена символом остановки, присоединенным к варианту тестового компонента, который выполняет операцию остановки тестового компонента **stop**. В нем должно содержаться соответствующее выражение, определяющее компонент, который следует остановить (см. рисунок 46). Компонент MTC может остановить все PTC за один шаг, используя операцию остановки компонента с ключевым словом **all** (см. рисунок 47 a)). Компонент PTC может остановить выполнение теста путем остановки MTC (см. рисунок 47 b)). Операция остановки тестового компонента **stop** должна использоваться как последнее событие варианта компонента или как последнее событие операнда в символе встроенного выражения, если компонент останавливается сам (например, **self.stop**) или останавливает выполнение теста (например, **mtc.stop**) (см. рисунки 47 c) и d)).

ПРИМЕЧАНИЕ. – Символ остановки имеет соответствующее выражение. Не всегда можно статистически определить, останавливает ли операция остановки компонента вариант, который выполняет операцию остановки, или останавливает выполнение теста.

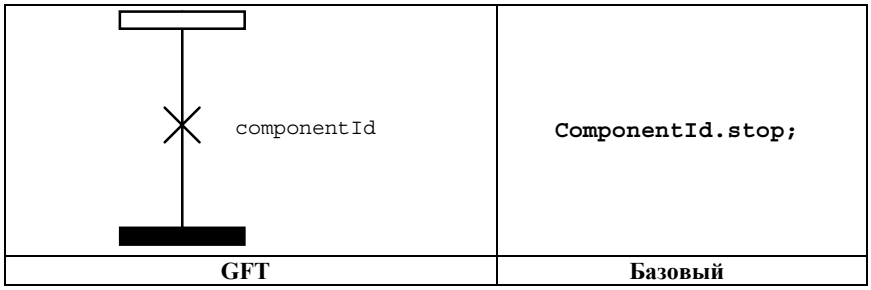


Рисунок 46/Z.142 – Операция остановки тестового компонента

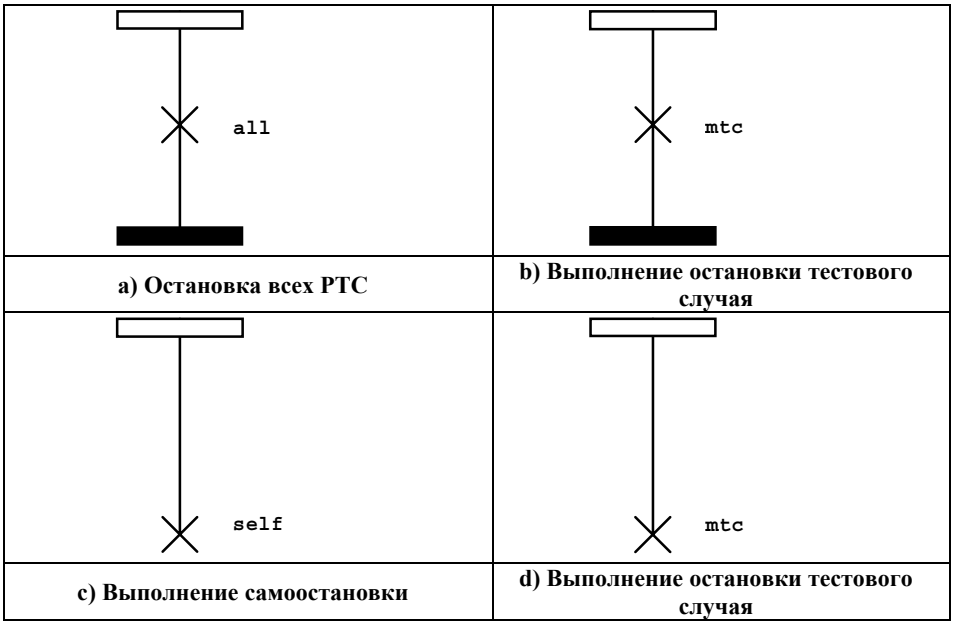


Рисунок 47/Z.142 – Случаи специального использования операции остановки тестового компонента

11.7.6 Операция "сделано"

Операция **done** должна быть представлена в символе условия, присоединенном к варианту тестового компонента, который выполняет операцию **done** (см. рисунок 48). Символ условия содержит оператор **done**.

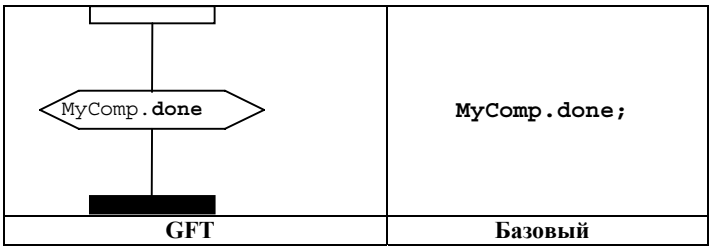


Рисунок 48/Z.142 – Операция "сделано"

Ключевые слова **any** и **all** могут использоваться для операций **running** и **done**, но только исходящих от варианта МТС. Они не имеют графического представления, но обозначаются в виде текста в местах их использования.

11.8 Операции связи

Операции связи подразделяются на две группы:

a) *Операции отправки:* тестовый компонент направляет сообщение (операция **send**), вызывает процедуру (операция **call**), отвечает на принятый вызов (операция **reply**) или устанавливает исключение (операция **raise**).

b) *Операции приема:* компонент осуществляет прием сообщения (операция **receive**), принимает вызов процедуры (операция **getcall**), осуществляет прием ответа на ранее вызванную процедуру (операция **getreply**) или обнаруживает исключение (операция **catch**).

11.8.1 Общий формат операций отправки

Для всех операций отправки используется символ сообщения, который изображается направляемым от варианта тестового компонента, выполняющего операцию отправки, в вариант порта, на который передается информация (см. рисунок 49).

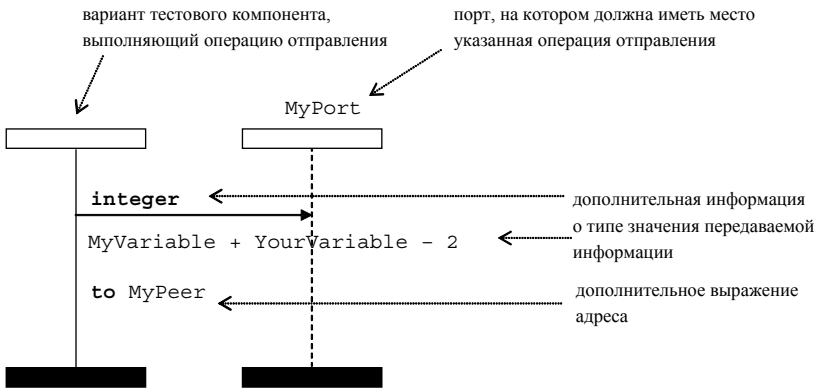


Рисунок 49/Z.142 – Общий формат операций отправки

Операции отправления состоят из части *направления* и - в случае операции **call**, основанной на блокирующей процедуре, - части *приема* и *обработки исключения*.

Часть *направления*:

- определяет порт, на котором должна выполняться указанная операция;
- определяет дополнительные тип и значение информации, которая должна быть передана;
- предоставляет дополнительное выражение адреса, которое однозначно определяет партнера по связи в случае соединения из пункта во многие пункты.

Порт должен быть представлен вариантом порта. Название операции для операций **call**, **reply** и **raise** должно быть обозначено в верхней части символа сообщения перед дополнительной информацией о типе. Операция **send** является неявной, т. е. ключевое слово **send** не должно обозначаться. Значение передаваемой информации должно располагаться под символом сообщения. Дополнительное выражение адреса (обозначаемое ключевым словом **to**) должно быть расположено под значением передаваемой информации.

Структура операции **call** является еще более конкретной. Более подробную информацию см. в п. 11.8.4.1.

11.8.2 Общий формат операций приема

Во всех операциях приема используется символ сообщения, который изображается направляемым от варианта порта к варианту тестового компонента, принимающего информацию (см. рисунок 50).

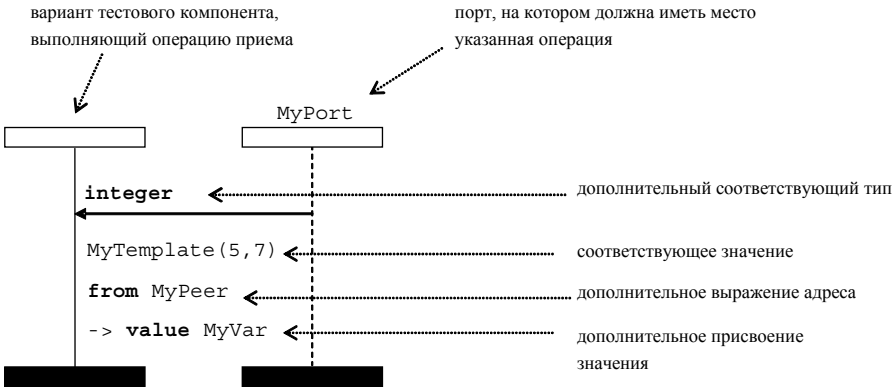


Рисунок 50/Z.142 – Общий формат операций приема с присвоением адреса и значения

Операция осуществления приема состоит из части *приема* и дополнительной части *присвоения*.

Часть *приема*:

a) определяет порт, на котором должна выполняться операция;

b) определяет соответствующую часть, включающую дополнительную информацию о типе и соответствующее значение, которое определяет приемлемый входной сигнал, который будет соответствовать оператору;

c) предоставляет выражение (дополнительное) адреса, которое однозначно определяет партнера по связи (в случае соединений из пункта во многие пункты).

Порт должен быть представлен вариантом порта. Название операции для операций **getcall**, **getreply** и **catch** должно быть обозначено в верхней части символа сообщения перед информацией (дополнительной) о типе. Операция **receive** задается в неявном виде, т. е. ключевое слово **receive** не должно обозначаться. Соответствующее значение приемлемого входного сигнала должно располагаться под символом сообщения. Выражение (дополнительное) адреса (обозначаемое ключевым словом **from**) должно быть расположено под значением передаваемой информации.

Часть (дополнительная) *присвоение* (обозначаемая как "**->**") должна быть расположена под значением передаваемой информации или под выражением адреса, если оно имеет место. Она может состоять из нескольких строк, например, чтобы каждое присвоение значения, параметра и отправителя располагалось в отдельной строке (см. рисунок 51).

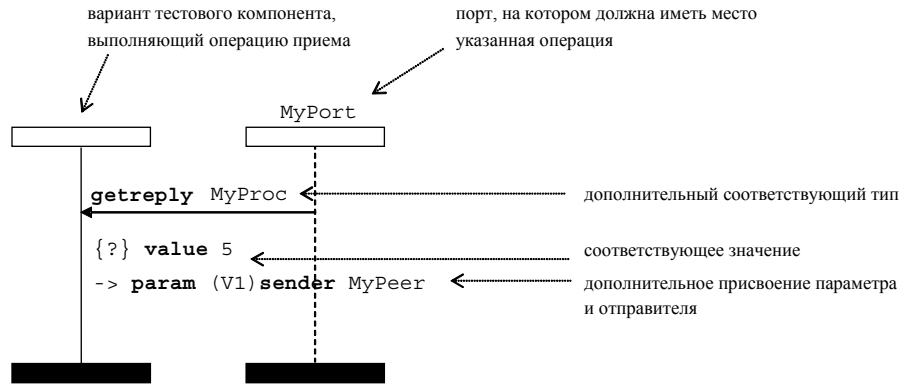


Рисунок 51/Z.142 – Общий формат операций приема с присвоением параметра и отправителя

11.8.3 Связь на основе сообщений

11.8.3.1 Операция отправления

Операция отправления должна быть представлена исходящего символом сообщения, направленного от тестового компонента к варианту порта. Дополнительная информация о типе должна располагаться над стрелкой сообщения. Шаблон (встроенный шаблон) должен быть размещен под стрелкой сообщения (см. рисунки 52 и 53).

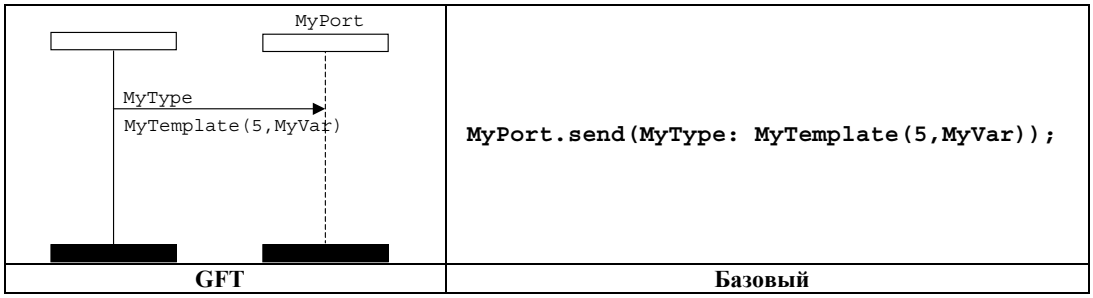


Рисунок 52/Z.142 – Операция отправления с ссылкой на шаблон

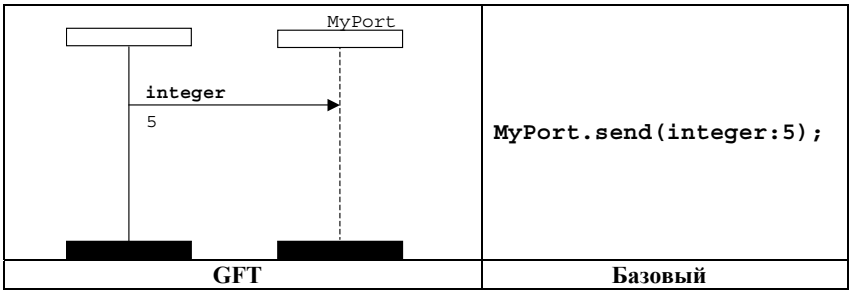


Рисунок 53/Z.142 – Операция отправления с встроенным шаблоном

11.8.3.2 Операция приема

Операция приема должна быть представлена стрелкой входящего сообщения, направленной от варианта порта к тестовому компоненту. Дополнительная информация о типе должна размещаться над стрелкой сообщения. Шаблон (встроенный шаблон) должен размещаться под стрелкой сообщения (см. рисунки 54 и 55).

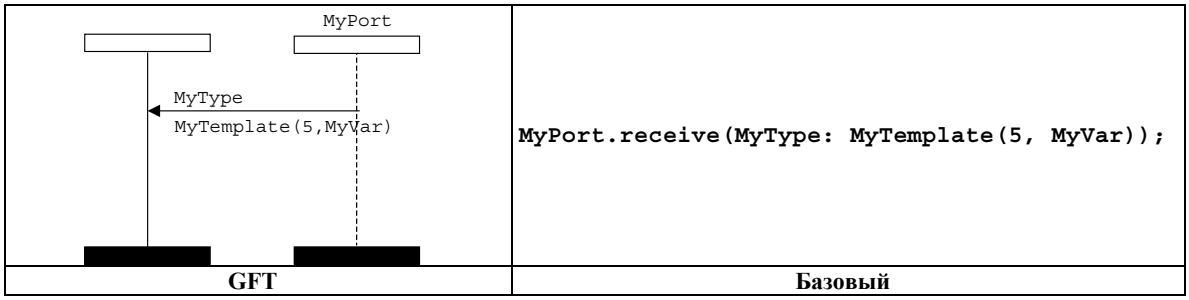


Рисунок 54/Z.142 – Операция приема с ссылкой на шаблон

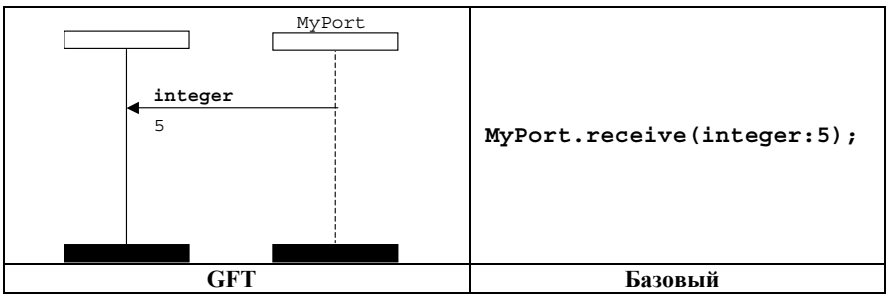


Рисунок 55/Z.142 – Операция приема с встроенным шаблоном

11.8.3.2.1 Прием любого сообщения

Операция приема любого сообщения должна быть представлена стрелкой входящего сообщения, направленной от варианта порта к тестовому компоненту без присоединения к нему какой-либо дополнительной информации (см. рисунок 56).

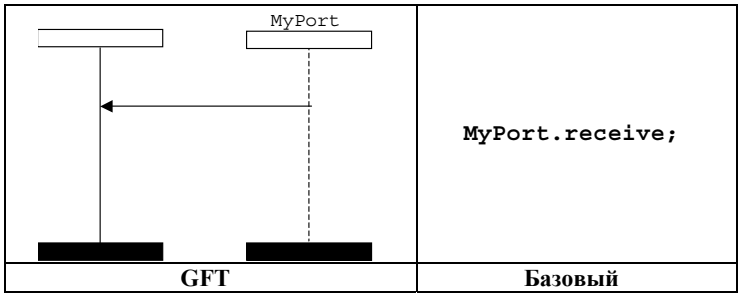


Рисунок 56/Z.142 – Прием любого сообщения

11.8.3.2.2 Прием на любой порт

Операция приема на любой порт должна быть обозначена символом обнаружения, представляющим любой порт и направленным к тестовому компоненту (см. рисунок 57).

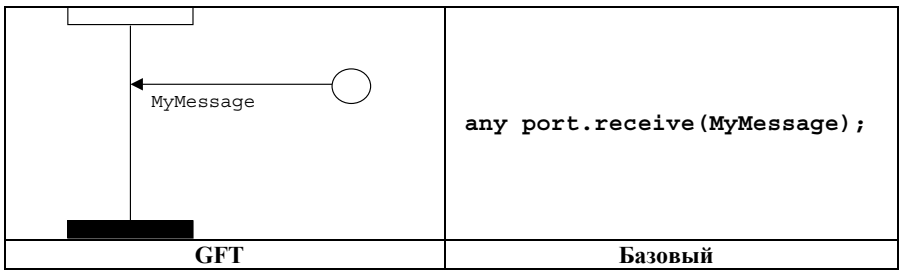


Рисунок 57/Z.142 – Прием на любой порт

11.8.3.3 Операция срабатывания

Операция срабатывания должна быть представлена стрелкой входящего сообщения, направленной от варианта порта к тестовому компоненту, и ключевым словом **trigger** над стрелкой сообщения, предшествующим информации о типе, если она имеет место. Дополнительная информация о типе размещается над стрелкой сообщения после ключевого слова **trigger**. Шаблон (встроенный шаблон) помещается под стрелкой сообщения (см. рисунки 58 и 59).

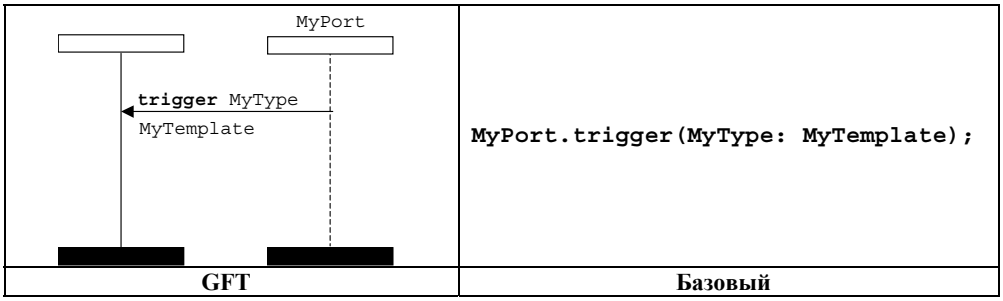


Рисунок 58/Z.142 – Операция срабатывания с ссылкой на шаблон

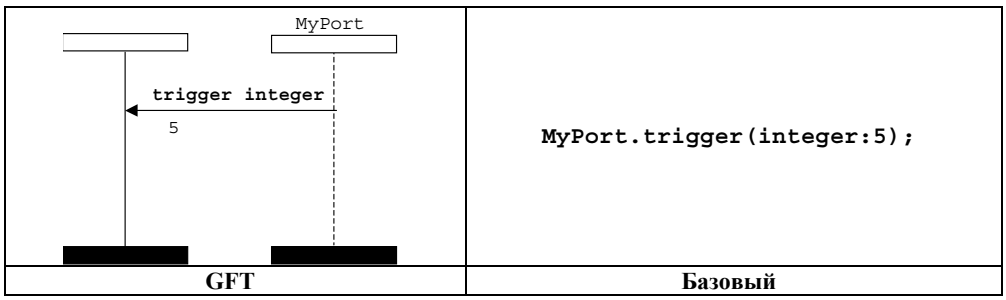


Рисунок 59/Z.142 – Операция срабатывания с встроенным шаблоном

11.8.3.3.1 Срабатывание на любом сообщении

Операция срабатывания на любом сообщении должна быть представлена стрелкой входящего сообщения, направленной от варианта порта к тестовому компоненту, и ключевым словом **trigger** над стрелкой сообщения без присоединения к нему какой-либо дополнительной информации (см. рисунок 60).

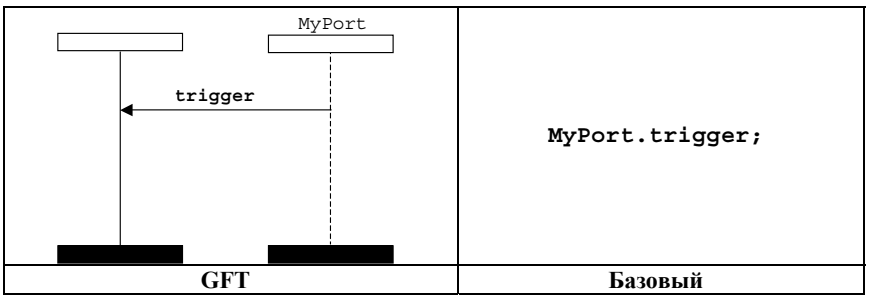


Рисунок 60/Z.142 – Срабатывание на любом сообщении

11.8.3.3.2 Срабатывание на любом порте

Операция срабатывания на любом порте должна быть обозначена символом обнаружения, представляющим любой порт и направленным к тестовому компоненту (см. рисунок 61).

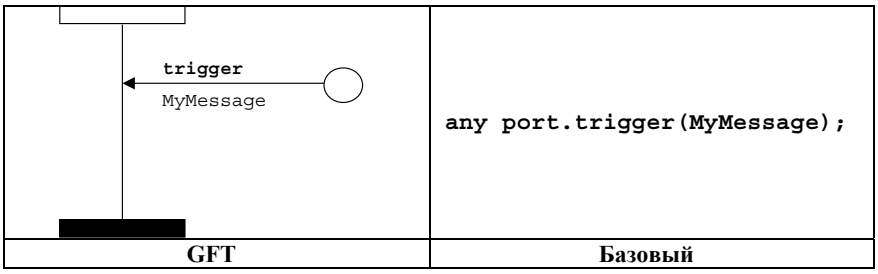


Рисунок 61/Z.142 – Срабатывание на любом порте

11.8.4 Связь, основанная на процедурах

11.8.4.1 Операция вызова

11.8.4.1.1 Процедуры блокирующего вызова

Операция блокирующего вызова **call** представляется символом исходящего сообщения, направленным от тестового компонента к варианту порта, с последующей областью приостановки на тестовом компоненте и ключевым словом **call** над стрелкой сообщения, предшествующим подписи, если она имеет место. Шаблон (встроенный шаблон) размещается под стрелкой сообщения (см. рисунки 62 и 63).

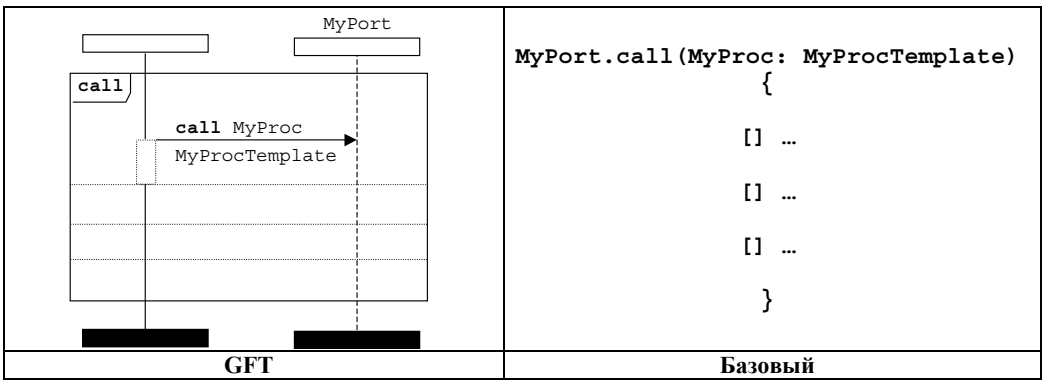


Рисунок 62/Z.142 – Операция блокирующего вызова с ссылкой на шаблон

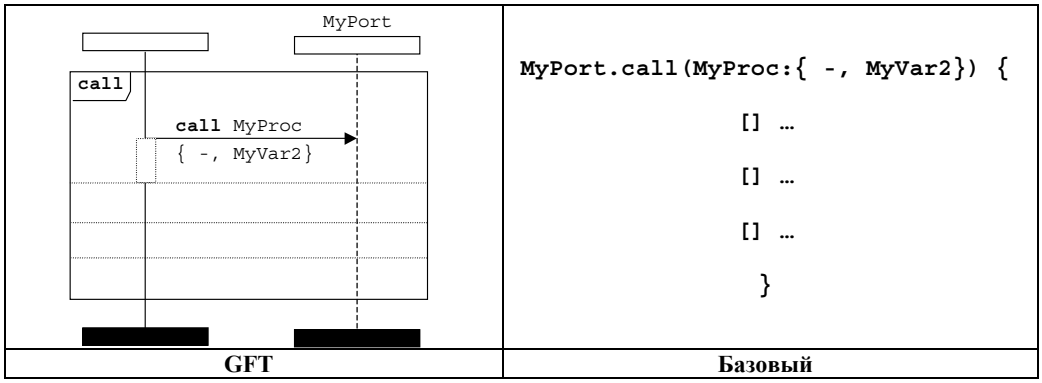


Рисунок 63/Z.142 – Операция блокирующего вызова с встроенным шаблоном

Встроенное выражение вызова вводится с целью упрощения характеристик альтернатив возможных ответов на операцию блокирующего вызова. За операцией вызова могут следовать альтернативы операторов getreply, catch и timeout. Ответы на вызов указываются во встроенном выражении вызова, следующим после операции вызова и отделенным пунктирными линиями (см. рисунок 64).

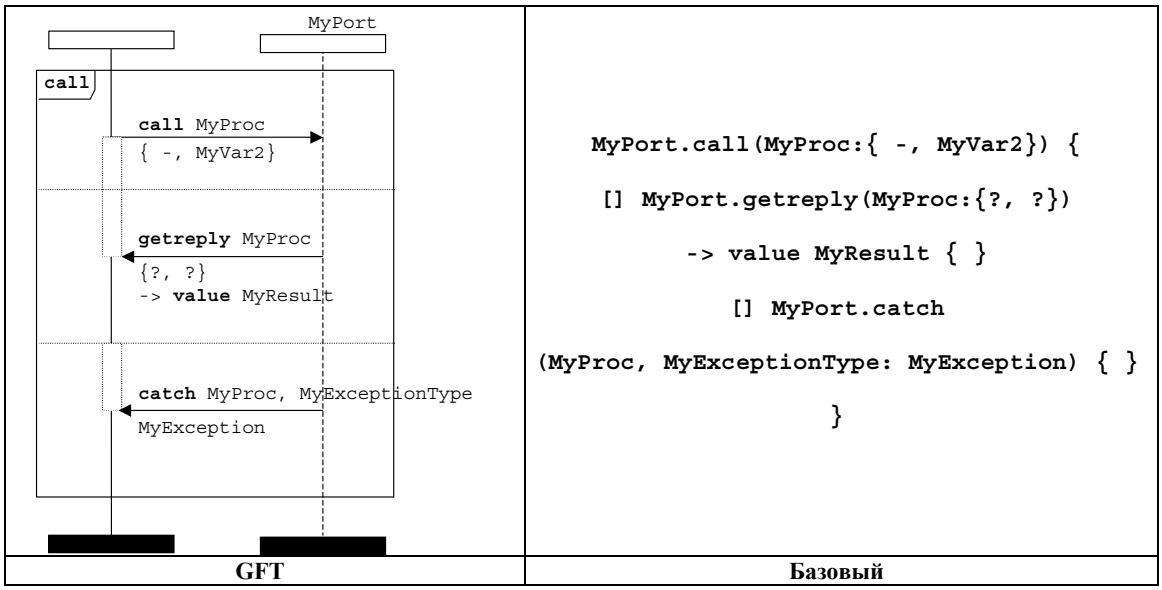


Рисунок 64/Z.142 – Операция блокирующего вызова со следующими за ней альтернативами операторов getreply и catch

Операция вызова может дополнительно включать тайм-аут. С этой целью для начала отсчета этого периода времени используется символ неявного таймера запуска. Символ неявного таймера тайм-аута используется для представления исключения тайм-аута (см. рисунок 65).

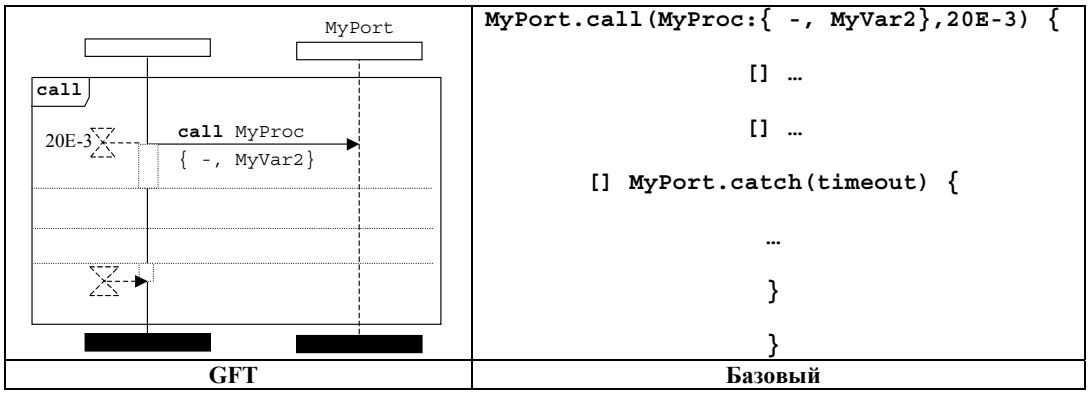


Рисунок 65/Z.142 – Операция блокирующего вызова, после которой следует исключение тайм-аута

11.8.4.1.2 Процедуры неблокирующего вызова

Операция неблокирующего вызова должна быть представлена символом исходящего сообщения, направленным от тестового компонента к порту, и ключевым словом **call** над стрелкой сообщения, предшествующим подписи. В этом случае не должно быть символа области приостановки, присоединенного к символу сообщения. Дополнительная подпись представляется над стрелкой сообщения. Шаблон (встроенный шаблон) размещается под стрелкой сообщения (см. рисунки 66 и 67).

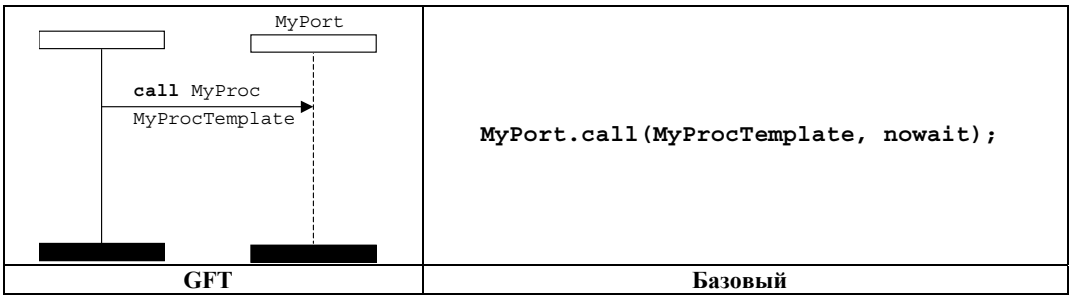


Рисунок 66/Z.142 – Операция неблокирующего вызова с ссылкой на шаблон

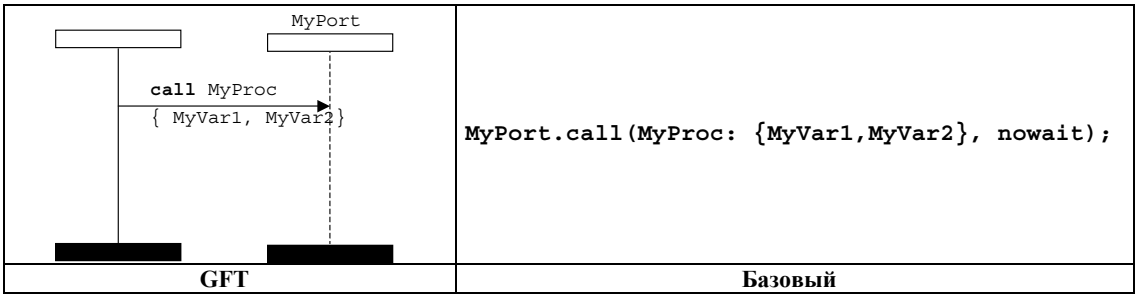


Рисунок 67/Z.142 – Операция неблокирующего вызова с встроенным шаблоном

11.8.4.2 Операция получения вызова

Операция получения вызова должна быть представлена стрелкой входящего сообщения, направленной от тестового компонента к порту, и ключевым словом **getcall** над стрелкой сообщения, предшествующим подписи. Подпись располагается над стрелкой сообщения после ключевого слова **getcall**. Шаблон (встроенный шаблон) размещается под стрелкой сообщения (см. рисунки 68 и 69).

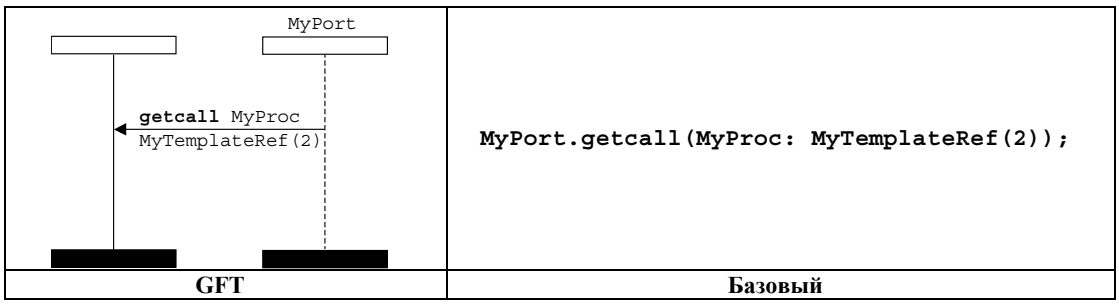


Рисунок 68/Z.142 – Операция получения вызова с ссылкой на шаблон

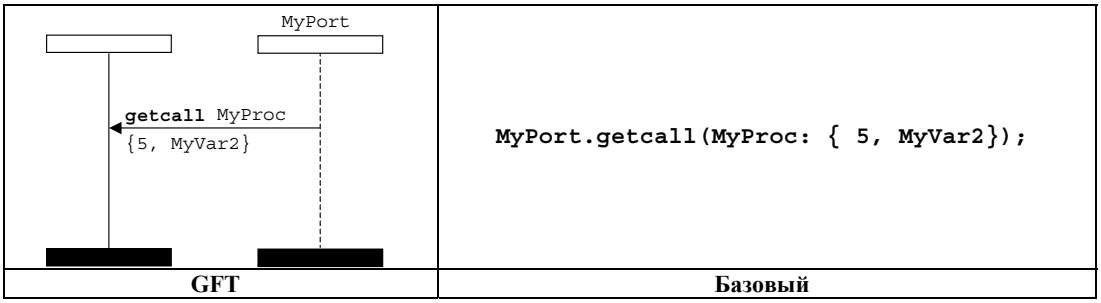


Рисунок 69/Z.142 – Операция получения вызова с встроенным шаблоном

11.8.4.2.1 Осуществление приема любого вызова

Операция осуществления приема любого вызова должна быть представлена стрелкой входящего сообщения, направленной от варианта порта к тестовому компоненту, и ключевым словом **getcall** над стрелкой сообщения. К символу сообщения не присоединяется никакая дополнительная информация (см. рисунок 70).

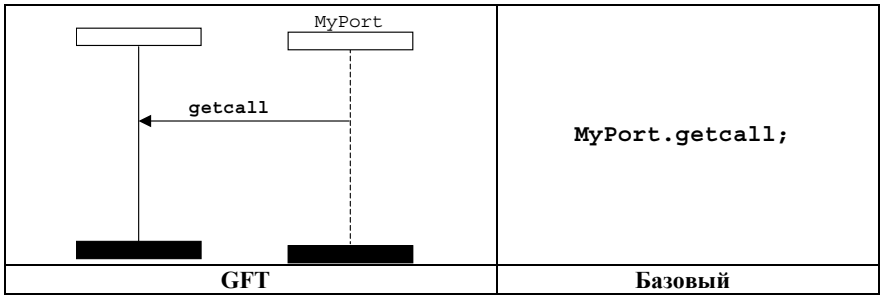


Рисунок 70/Z.142 – Операция получения любого вызова

11.8.4.2.2 Получение вызова на любой порт

Операция получения вызова на любой порт обозначается с помощью символа обнаружения, представляющего любой порт и направленного к тестовому компоненту, и ключевого слова **getcall** над стрелкой сообщения, предшествующего подписи при ее наличии. Шаблон (встроенный шаблон) помещается под стрелкой сообщения (см. рисунок 71).

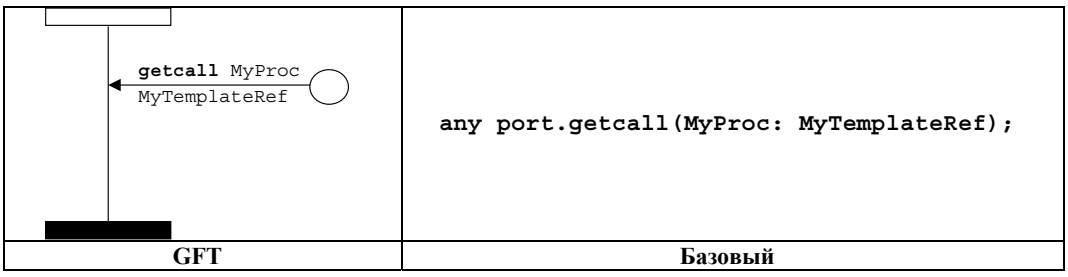


Рисунок 71/Z.142 – Операция получения любого вызова на любой порт со ссылкой на шаблон

11.8.4.3 Операция ответа

Операция ответа должна быть представлена символом исходящего сообщения, направленного от тестового компонента к варианту порта, и ключевым словом **reply** над стрелкой сообщения, предшествующим подписи. Подпись должна быть размещена над стрелкой сообщения после ключевого слова **reply**. Шаблон (встроенный шаблон) помещается под стрелкой сообщения (см. рисунки 72 и 73).

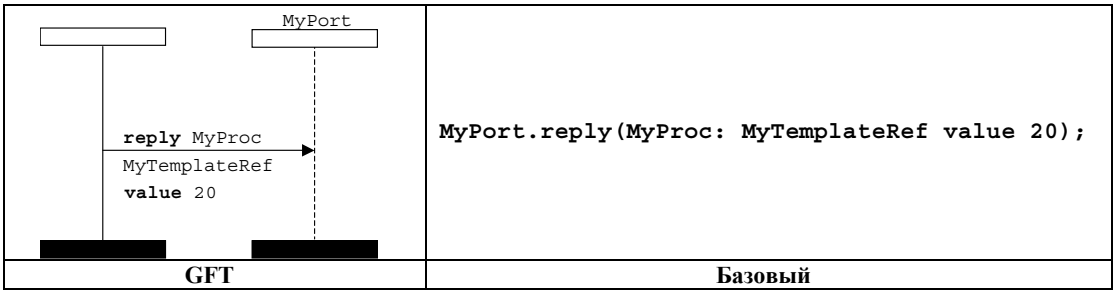


Рисунок 72/Z.142 – Операция ответа с ссылкой на шаблон

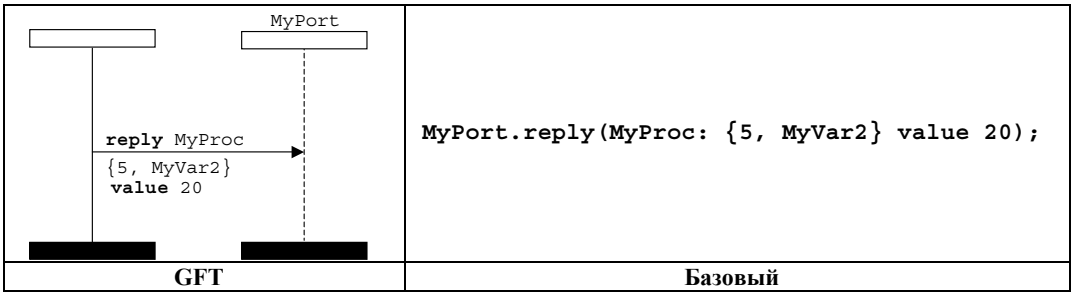


Рисунок 73/Z.142 – Операция ответа с встроенным шаблоном

11.8.4.4 Операция получения ответа

Операция получения ответа должна быть представлена стрелкой входящего сообщения, направленной от варианта порта к тестовому компоненту, и ключевым словом **getreply** над стрелкой сообщения, предшествующим подписи. В пределах символа вызова вершина стрелки сообщения должна быть присоединена к предшествующей области приостановки на тестовом компоненте (см. рисунки 74 и 75). Вне символа вызова вершина стрелки сообщения не должна быть присоединена к предшествующей области приостановки на тестовом компоненте (см. рисунки 76 и 77).

Подпись должна располагаться над стрелкой сообщения после ключевого слова **getreply**. Шаблон (встроенный шаблон) должен размещаться под стрелкой сообщения.

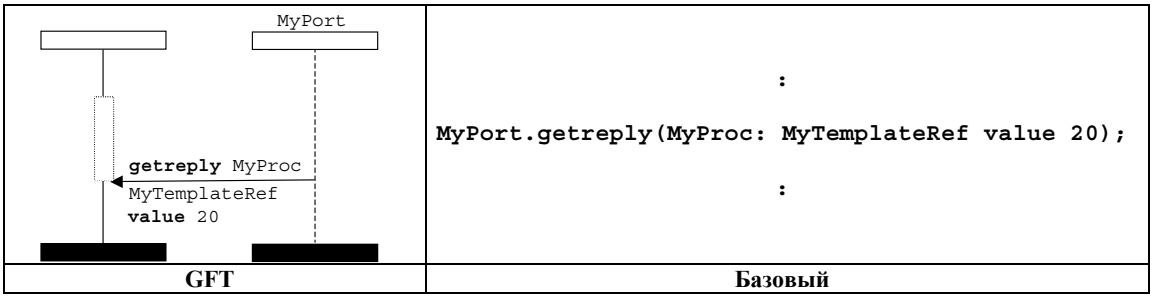


Рисунок 74/Z.142 – Операция получения ответа с ссылкой на шаблон (в пределах символа вызова)

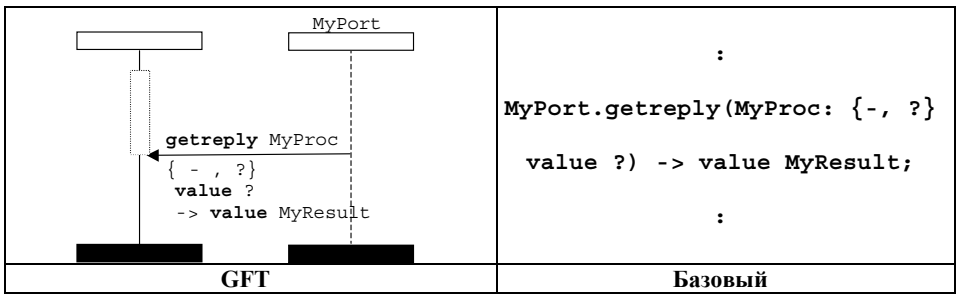


Рисунок 75/Z.142 – Операция получения ответа с встроенным шаблоном (в пределах символа вызова)

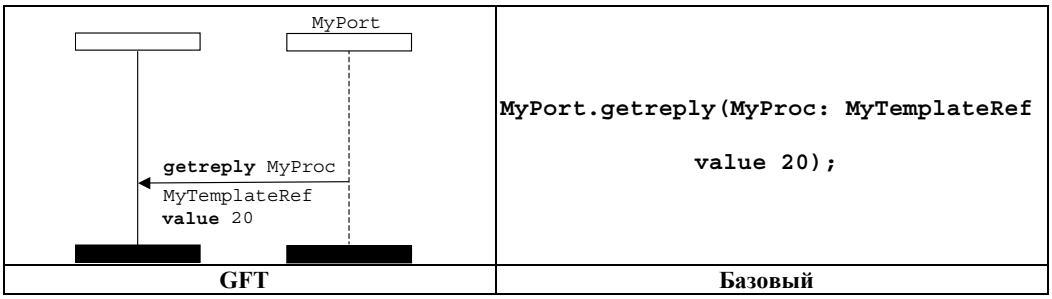


Рисунок 76/Z.142 – Операция получения ответа с ссылкой на шаблон (вне символа вызова)

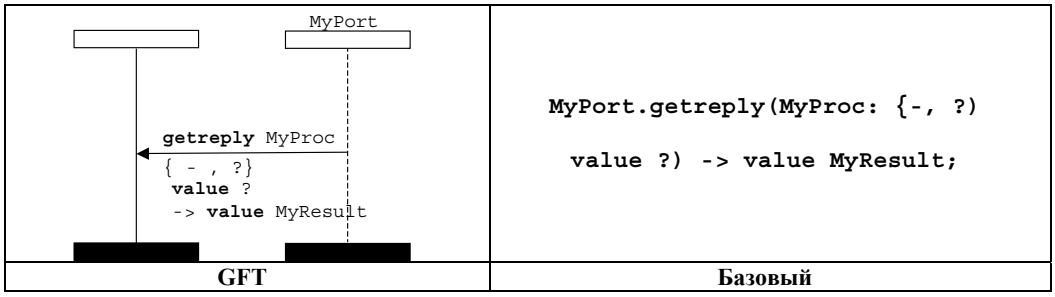


Рисунок 77/Z.142 – Операция получения ответа с встроенным шаблоном (вне символа вызова)

11.8.4.4.1 Получение любого ответа на любой вызов

Операция получения любого ответа на любой вызов должна быть представлена стрелкой входящего сообщения, направленной от варианта порта к тестовому компоненту, и ключевым словом **getreply** над сообщением. После ключевого слова **getreply** не должна следовать подпись. В пределах символа вызова вершина стрелки сообщения должна быть присоединена к предшествующей области приостановки на тестовом компоненте (см. рисунок 78). Вне символа вызова вершина стрелки сообщения не должна быть присоединена к предшествующей области приостановки на тестовом компоненте (см. рисунок 79).

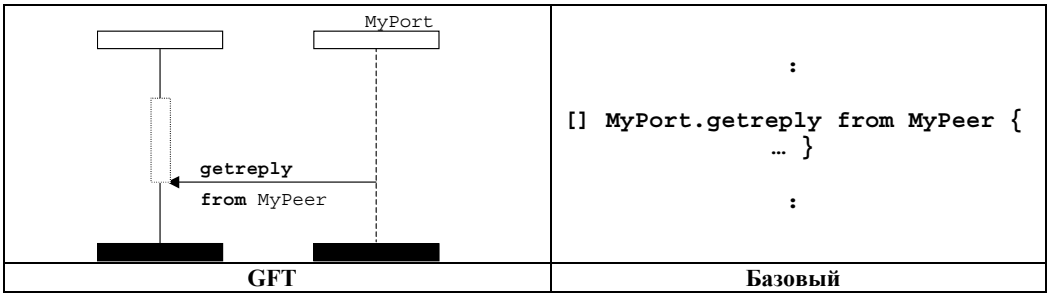


Рисунок 78/Z.142 – Получение любого ответа на любой вызов (в пределах символа вызова)

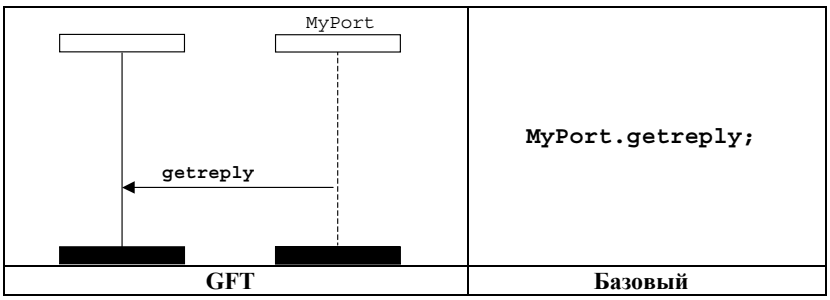


Рисунок 79/Z.142 – Получение любого ответа на любой вызов (вне символа вызова)

11.8.4.4.2 Получение ответа на любом порте

Операция получения ответа на любом порте обозначается символом обнаружения, представляющим любой порт и направленным к текстовому компоненту. После ключевого слова **getreply**, которое должно быть расположено над стрелкой сообщения, следует подпись, если она имеет место. В пределах символа вызова вершина стрелки сообщения должна быть присоединена к предшествующей области приостановки на тестовом компоненте (см. рисунок 80). Вне символа вызова вершина стрелки сообщения не должна быть присоединена к предшествующей области приостановки на тестовом компоненте (см. рисунок 81).

Подпись (если имеет место) должна быть расположена над стрелкой сообщения после ключевого слова **getreply**. Дополнительный шаблон (встроенный шаблон) располагается под стрелкой сообщения.

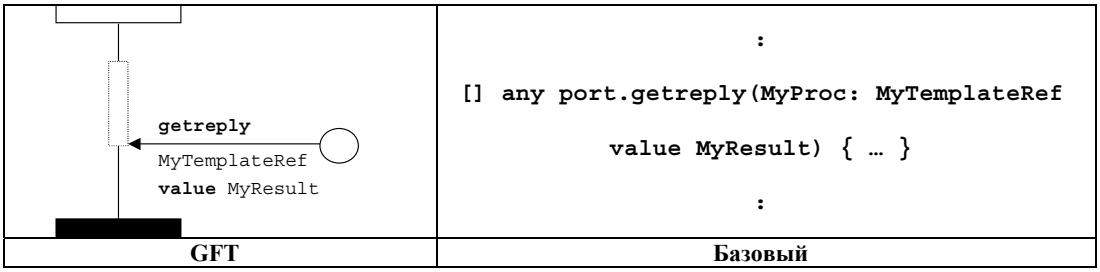


Рисунок 80/Z.142 – Получение ответа на любом порте (в пределах символа вызова)

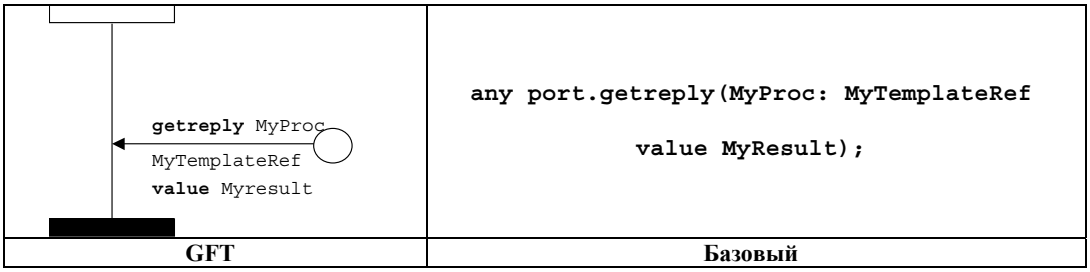


Рисунок 81/Z.142 – Получение ответа на любом порте (вне символа вызова)

11.8.4.5 Операция установления исключения

Операция установления исключения должна быть представлена символом исходящего сообщения, направленного от тестового компонента к варианту порта. Ключевое слово **raise** должно быть расположено над стрелкой сообщения перед подписью и типом исключения, разделенными запятой. Шаблон (встроенный шаблон) должен быть расположен под стрелкой сообщения (см. рисунки 82 и 83).

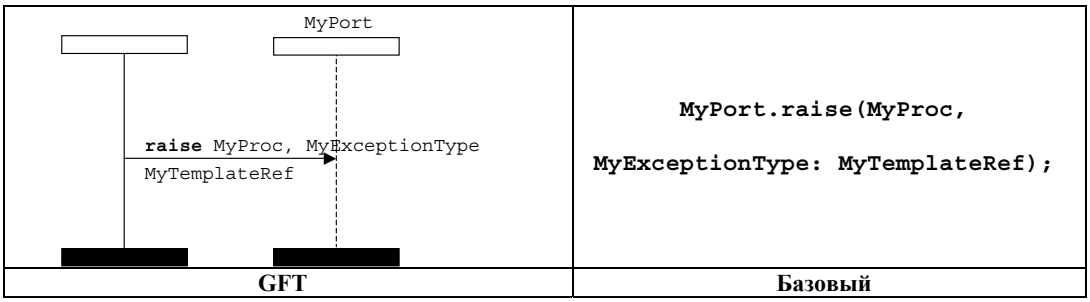


Рисунок 82/Z.142 – Операция установления исключения с ссылкой на шаблон

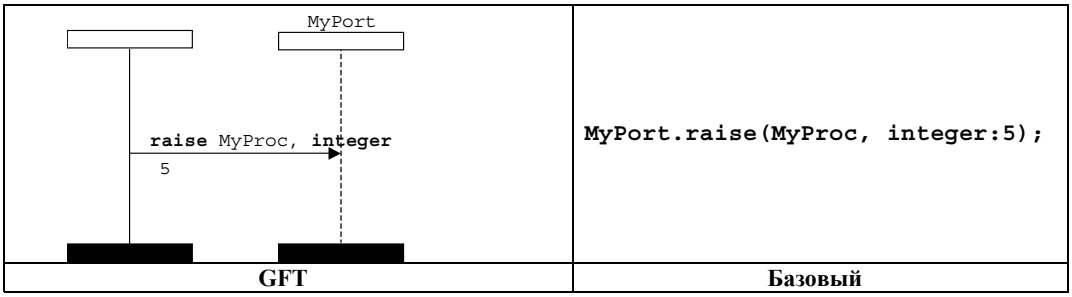


Рисунок 83/Z.142 – Операция установления исключения с встроенным шаблоном

11.8.4.6 Операция обнаружения исключения

Операция обнаружения исключения должна быть представлена стрелкой входящего сообщения, направленной от варианта порта к тестовому компоненту, и ключевым словом **catch** над стрелкой сообщения перед подписью и типом исключения (если имеет место). В пределах символа вызова вершина стрелки сообщения должна быть присоединена к предшествующей области приостановки на тестовом компоненте (см. рисунки 84 и 85). Вне символа вызова вершина стрелки сообщения не должна быть присоединена к предшествующей области приостановки на тестовом компоненте (см. рисунки 86 и 87).

Подпись и дополнительная информация о типе исключения должны быть расположены над стрелкой сообщения после ключевого слова **catch** и отделены запятой, если тип исключения имеет место. Шаблон (встроенный шаблон) располагается под стрелкой сообщения.

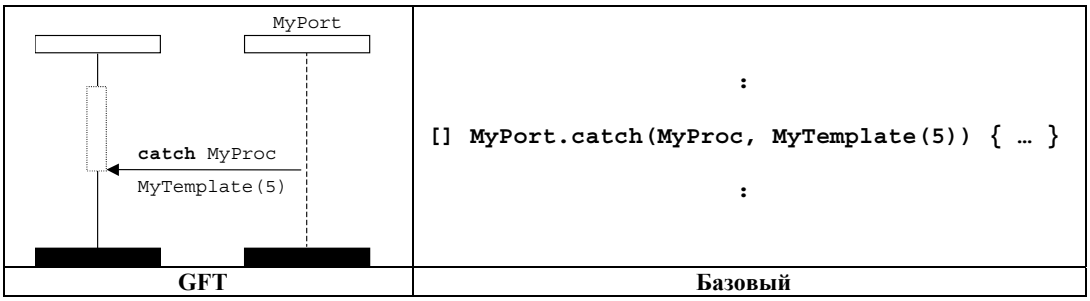
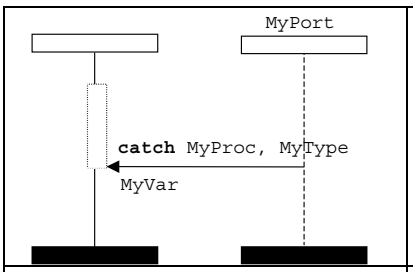


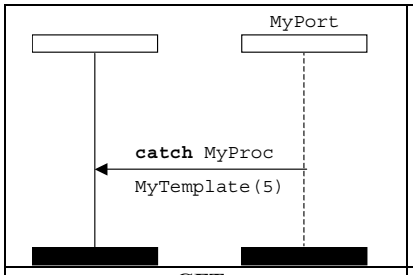
Рисунок 84/Z.142 – Операция обнаружения исключения с ссылкой на шаблон (в пределах символа вызова)




```
:
[] MyPort.catch(MyProc, MyType: MyVar) { … }
:
```

GFT	Базовый

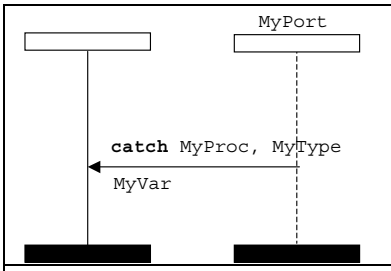
Рисунок 85/Z.142 – Операция обнаружения исключения с встроенным шаблоном (в пределах символа вызова)




```
MyPort.catch(MyProc, MyTemplate(5));
```

GFT	Базовый

Рисунок 86/Z.142 – Операция обнаружения с ссылкой на шаблон (вне символа вызова)



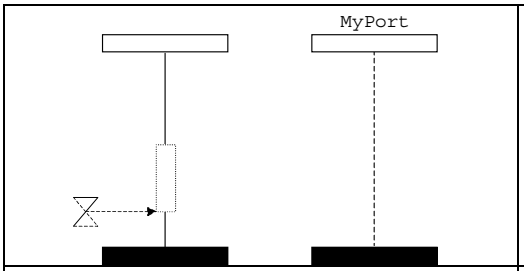

```
MyPort.catch(MyProc, MyType: MyVar);
```

GFT	Базовый

Рисунок 87/Z.142 – Операция обнаружения с встроенным шаблоном (вне символа вызова)

11.8.4.6.1 Исключение тайм-аута

Операция исключения тайм-аута должна быть представлена символом тайм-аута со стрелкой, соединенной с тестовым компонентом (см. рисунок 88). Никакая дополнительная информация не должна быть присоединена к символу тайм-аута. Он должен использоваться только в символе вызова. Вершина стрелки сообщения должна быть присоединена к предшествующей области приостановки на тестовом компоненте.




```
:
[] MyPort.catch(timeout) { … }
:
```

GFT	Базовый

Рисунок 88/Z.142 – Исключение тайм-аута (в пределах символа вызова)

11.8.4.6.2 Обнаружение любого исключения

Операция обнаружения любого исключения должна быть представлена стрелкой входящего сообщения, направленной от варианта порта к тестовому компоненту, и ключевым словом **`catch`** над стрелкой сообщения. Внутри символа вызова вершина стрелки сообщения должна быть присоединена к предшествующей области приостановки на тестовом компоненте (см. рисунок 89). Вне символа вызова вершина стрелки сообщения не должна быть присоединена к предшествующей области приостановки на тестовом компоненте (см. рисунок 90). Операция обнаружения любого исключения не должна иметь шаблона и типа исключения.

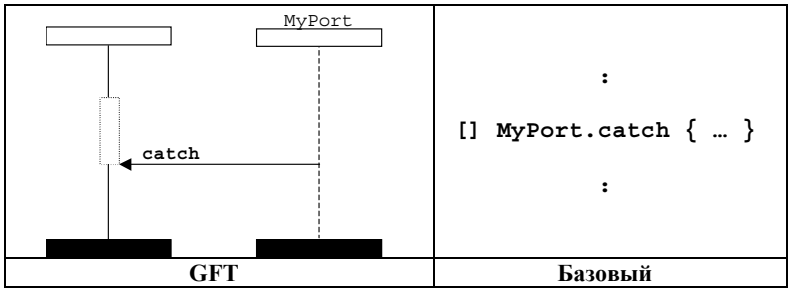


Рисунок 89/Z.142 – Обнаружение любого исключения (в пределах символа вызова)

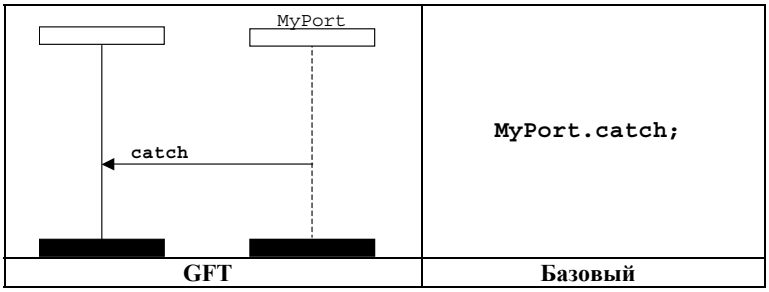


Рисунок 90/Z.142 – Обнаружение любого исключения (вне символа вызова)

11.8.4.6.3 Обнаружение на любом порту

Операция обнаружения на любом порту представляется символом обнаружения, представляющим любой порт и направленным к тестовому компоненту, и ключевым словом **`catch`** над стрелкой сообщения. Внутри символа вызова вершина стрелки сообщения должна быть присоединена к предшествующей области приостановки на тестовом компоненте (см. рисунок 91). Вне символа вызова вершина стрелки сообщения не должна быть присоединена к предшествующей области приостановки на тестовом компоненте (см. рисунок 92). В случае наличия шаблона он размещается под стрелкой сообщения.

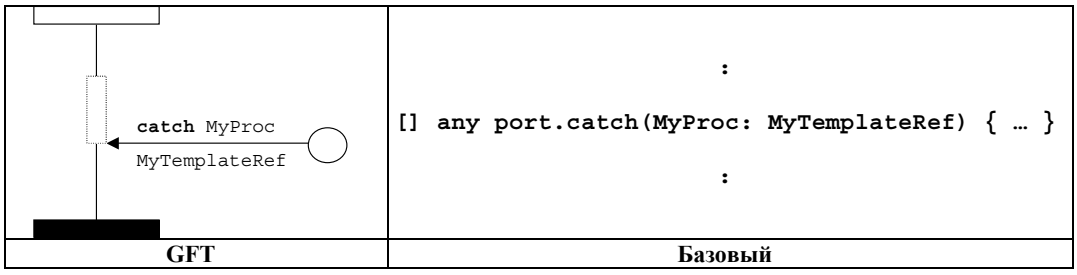
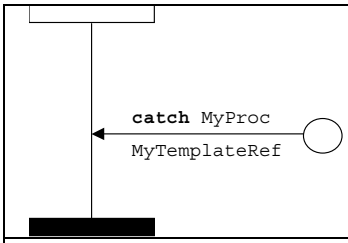


Рисунок 91/Z.142 – Обнаружение на любом порту (вне символа вызова)



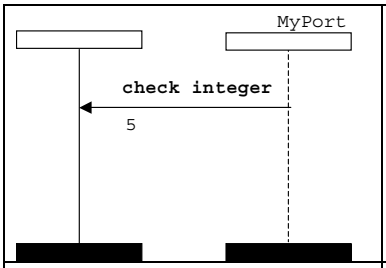

```
any port.catch(MyProc: MyTemplateRef);
```

GFT | **Базовый**

Рисунок 92/Z.142 – Обнаружение на любом порту (вне символа вызова)

11.8.5 Операция проверки

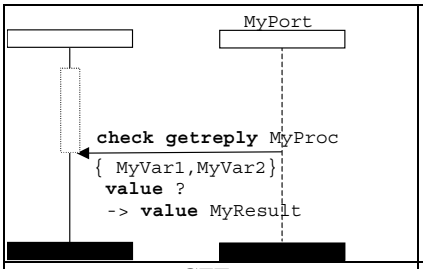
Операция проверки должна быть представлена стрелкой входящего сообщения, направленной от варианта порта к тестовому компоненту. Ключевое слово **check** должно располагаться над стрелкой сообщения. Присоединенная информация, относящаяся к операциям **receive** (см. рисунок 93), **getcall**, **getreply** (см. рисунки 94 и 95) и **catch,** следует за ключевым словом **check** и соответствует правилам, применяемым к представлению этих операций.




```
MyPort.check(receive(integer: 5));
```

GFT | **Базовый**

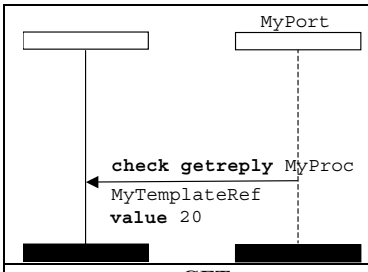
Рисунок 93/Z.142 – Проверка операции receive с встроенным шаблоном




```
:
[] MyPort.check(getreply(MyProc1:{MyVar1, MyVar2}
value ?) -> value MyResult)
{ … }
:
```

GFT | **Базовый**

Рисунок 94/Z.142 – Проверка операции getreply (в пределах символа вызова)




```
MyPort.check(getreply(MyProc:MyTemplateRef value 20);
```

GFT | **Базовый**

Рисунок 95/Z.142 – Проверка операции getreply (вне символа вызова)

11.8.5.1 Операция какой-либо проверки

Операция какой-либо проверки должна быть представлена стрелкой входящего сообщения, направленной от варианта порта к тестовому компоненту, и ключевым словом **check** над стрелкой сообщения (см. рисунок 96). К ней не должно быть присоединено ключевое слово операции приема, тип и шаблон. Дополнительно может присоединяться информация об адресе и запомненная информация об отправителе.

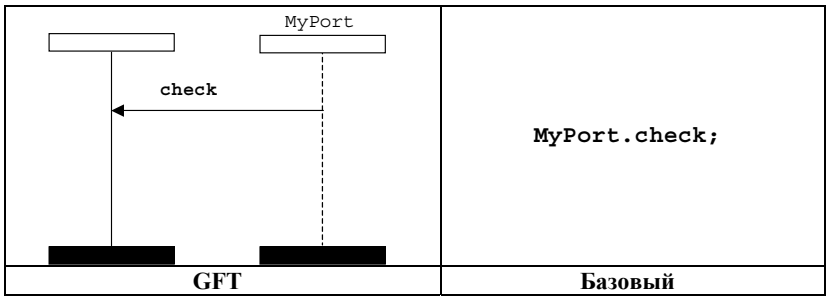


Рисунок 96/Z.142 – Операция какой-либо проверки

11.8.5.2 Проверка на любом порте

Операция проверки на любом порте представляется символом обнаружения, представляющим любой порт и направленным в сторону тестового компонента, и ключевым словом **check** над стрелкой сообщения (см. рисунок 97). Присоединенная информация, относящаяся к операциям **receive**, **getcall**, **getreply** и **catch** следует за ключевым словом **check** и соответствует правилам, применяемым к представлению этих операций.

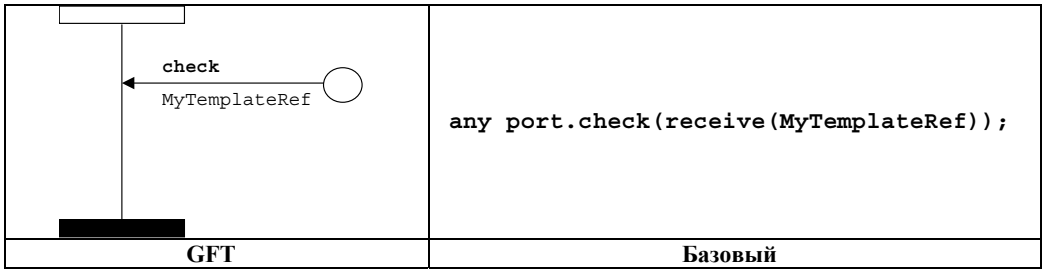


Рисунок 97/Z.142 – Проверка приема на любом порте

11.8.6 Управление портами связи

11.8.6.1 Операция очистки порта

Операция очистки порта должна быть представлена символом условия с ключевым словом **clear**. Он присоединен к варианту тестового компонента, который выполняет операцию очистки порта, и к очищаемому порту (см. рисунок 98).

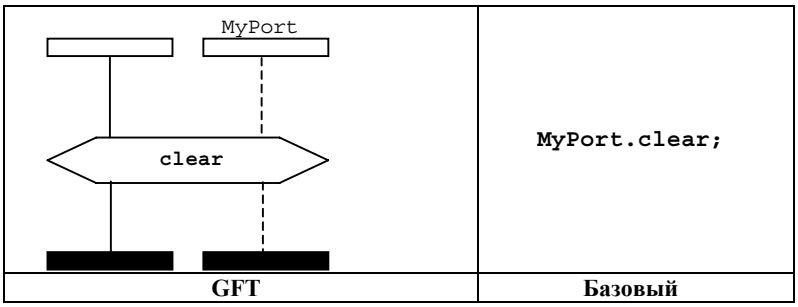


Рисунок 98/Z.142 – Операция очистки порта

11.8.6.2 Операция открытия порта

Операция открытия порта должна быть представлена символом условия с ключевым словом **start**. Он присоединен к варианту тестового компонента, который выполняет операцию открытия порта, и к открываемому порту (см. рисунок 99).

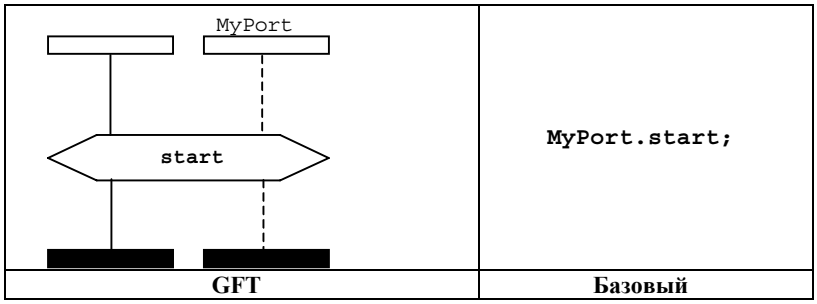


Рисунок 99/Z.142 – Операция открытия порта

11.8.6.3 Операция закрытия порта

Операция закрытия порта должна быть представлена символом условия с ключевым словом **stop**. Он присоединен к варианту тестового компонента который выполняет операцию закрытия порта, и к закрываемому порту (см. рисунок 100).

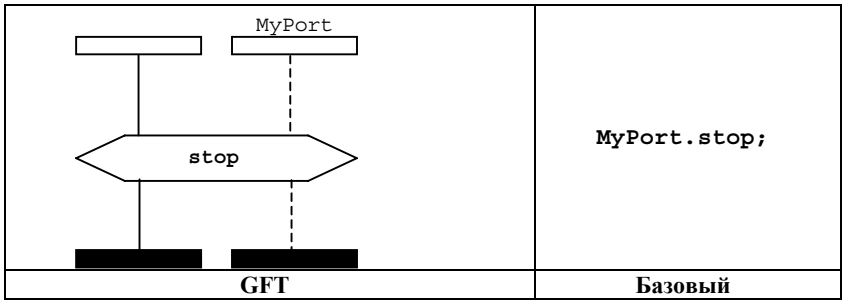


Рисунок 100/Z.142 – Операция закрытия порта

11.8.6.4 Использование ключевых слов `any` и `all` с портами

В соответствующих подпунктах п. 11.8 поясняется представление в формате GFT ключевого слова **any** для портов вместе с операциями **receive**, **trigger**, **getcall**, **getreply**, **catch** и **check**.

Ключевое слово **all** для портов в случае операции очистки, открытия и закрытия представляется путем присоединения символа условия, содержащего операцию **clear**, **start** или **stop,** ко всем вариантам порта, представленным в диаграмме GFT для тестового случая, функции или альтернативного шага.

11.9 Операции таймера

В GFT существует два различных символа таймера: один для определения таймеров и один для таймеров вызова (см. рисунок 101). Они различны по виду: символы таймера, изображаемые сплошной линией, используются для определения таймеров, а символы таймера, изображаемые штриховой линией, - для таймеров вызова. Тот или иной определенный таймер должен снабжаться названием, присоединенным к его символу, тогда как таймер вызова не имеет имени. В данном пункте описаны определенные таймеры. Таймер вызова рассматривается в п. 11.8.4.

Рисунок 101/Z.142 – Определенный таймер и таймеры вызова

Формат GFT не предоставляет какого-либо графического представления для операции таймера **`running`** (являющегося булевым выражением). Он обозначается в виде текста в местах его использования.

11.9.1 Операция запуска таймера

В случае операции запуска таймера символ запуска таймера должен быть присоединен к варианту компонента. С ним может быть связано название таймера и дополнительное значение длительности (в скобках) (см. рисунок 102).

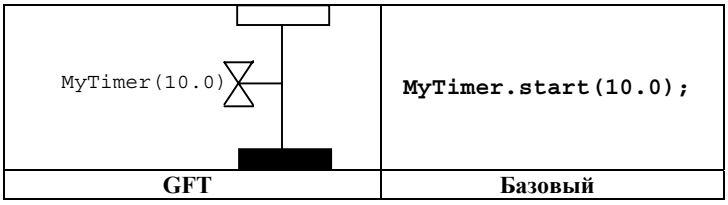


Рисунок 102/Z.142 – Операция запуска таймера

11.9.2 Операция остановки таймера

В случае операции остановки таймера символ остановки таймера должен быть присоединен к варианту компонента. С ним может быть связано дополнительное название таймера (см. рисунок 103).

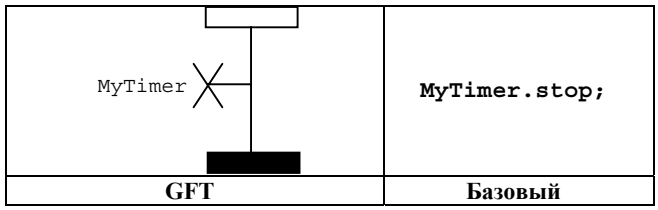


Рисунок 103/Z.142 – Операция остановки таймера

Символы операции запуска и остановки таймера могут быть соединены вертикальной линией. В этом случае следует только указывать идентификатор таймера после символа запуска таймера (см. рисунок 104).

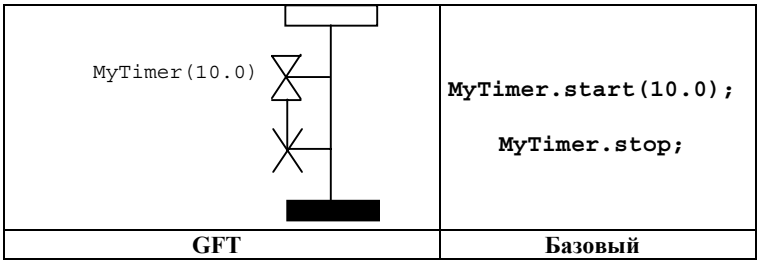


Рисунок 104/Z.142 – Соединенные символы запуска и остановки таймера

11.9.3 Операция тайм-аута

В случае операции тайм-аута символ тайм-аута символ таймера должен быть присоединен к варианту компонента. С ним может быть связано дополнительное название таймера (см. рисунок 105).

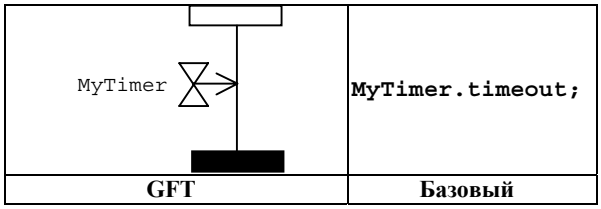


Рисунок 105/Z.142 – Операция тайм-аута

Символы запуска таймера и операции тайм-аута могут быть соединены вертикальной линией. В этом случае следует только указывать идентификатор таймера после символа запуска таймера (см. рисунок 106).

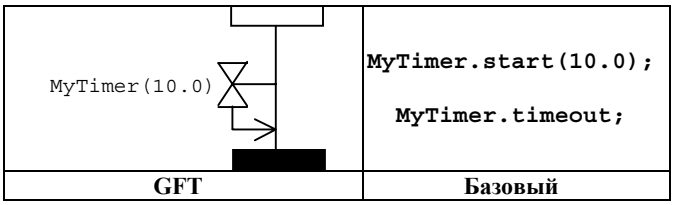


Рисунок 106/Z.142 Соединенные символы запуска таймера и тайм-аута

11.9.4 Операция считывания таймера

Операция считывания таймера должна быть помещена в блок действия (см. рисунок 107).

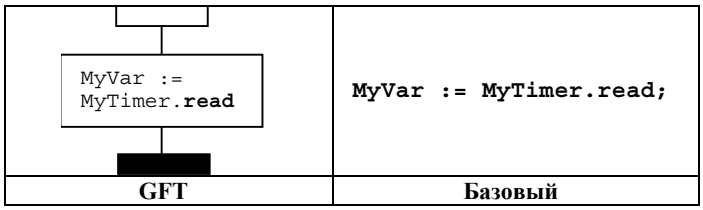


Рисунок 107/Z.142 – Операция считывания таймера

11.9.5 Использование слов `any` и `all` с таймерами

Операция остановки таймера может быть применена ко всем (**`all`**) таймерам (см. рисунок 108).

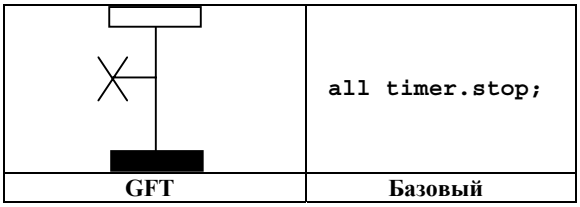


Рисунок 108/Z.142 – Остановка всех таймеров

Операция тайм-аута может быть применена к любому (**`any`**) таймеру (см. рисунок 109).

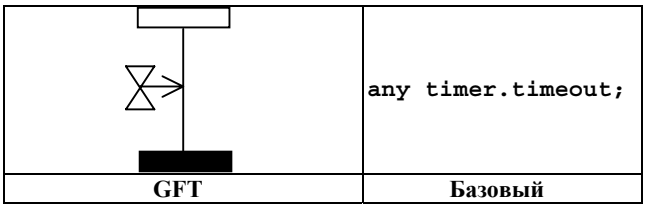


Рисунок 109/Z.142 – Тайм-аут с любого таймера

11.10 Операции установления заключений по тестам

В формате GFT операция установления заключений **setverdict** представляется символом условия, в котором обозначены значения **pass**, **fail**, **inconc** или **none** (см. рисунок 110).

ПРИМЕЧАНИЕ. – Правила установления нового заключения следуют обычным правилам TTCN-3 в отношении перезаписи заключений по тестам.

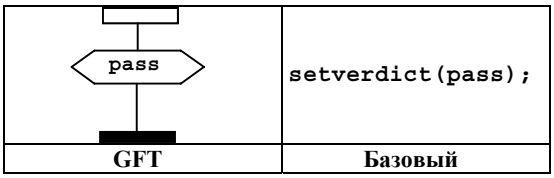


Рисунок 110/Z.142 – Установка местного заключения

Формат GFT не дает никакого графического представления операции **getverdict** (являющейся выражением). Она указывается в текстовом виде в местах ее использования.

11.11 Внешние действия

Внешние действия представлены внутри символов блоков действия (см. рисунок 111). Синтаксис внешнего действия размещается внутри этого символа.

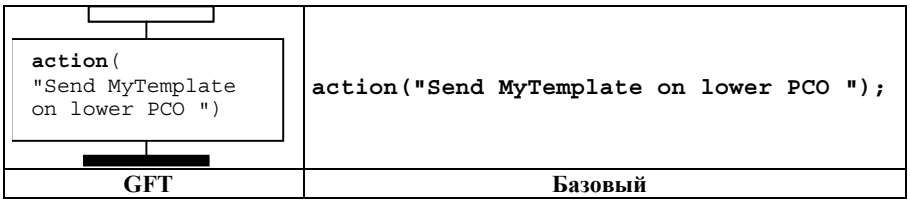


Рисунок 111/Z.142 – Внешние действия

11.12 Определение атрибутов

Атрибуты, определенные для части управления модулем, тестовых случаев, функций и альтернативных шагов, представлены внутри текстового символа. Синтаксис оператора **with** размещается внутри этого символа. Пример приведен на рисунке 112.

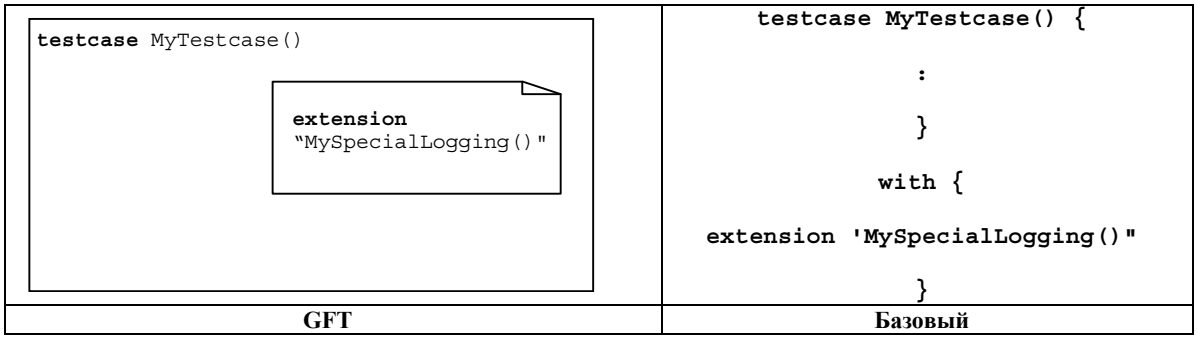


Рисунок 112/Z.142 – Определение атрибутов

Приложение A

Форма BNF формата GFT

A.1 Метаязык для GFT

Графический синтаксис для GFT определен на основе графического синтаксиса MSC (Рек. МСЭ-Т Z.120 [3]). В определении графического синтаксиса используется метаязык, разъяснения по которому приведены в п. 1.4 Рек. МСЭ-Т Z.120 [3]:

"Графический синтаксис не является достаточно строгим, чтобы при описании графики не допускать никаких графических отклонений. Допустимы незначительные отклонения в действительной форме графических терминальных символов. К ним относятся, например, тонирование заполненных символов, форма вершины стрелки и относительный размер графических элементов. Когда это необходимо, графический синтаксис будет дополняться неформальным объяснением возникновения определенных конструкций. Данный метаязык содержит нотацию, аналогичную форме BNF, со специальными метаконструкциями: ***contains (содержит)***, ***is followed by*** (сопровождается), ***is associated with*** (связан с), ***is attached to*** (присоединен к)**,** ***above*** (над) и ***set*** (множество)**.** Поведение этих конструкций аналогично правилам вывода для нормальной формы BNF, но, кроме того, предполагается некоторая геометрическая или логическая связь между аргументами. Конструкция ***is attached to*** ведет себя несколько иначе, по сравнению с тем, как объяснено ниже. Слева от всех этих конструкций, за исключением конструкции ***above***, должен быть символ. Этот нетерминальный символ в каждой последовательности выводов приводит к одному определенному графическому терминальному символу. Рассмотрим символ, который ***присоединен к*** другим областям или который ***связан с*** текстовой строкой тоже в качестве символа. Это объяснение не является формальным, поэтому данный метаязык не дает точного описания геометрических зависимостей."

С более подробной информацией можно ознакомиться в Рек. МСЭ-Т Z.120 [3].

A.2 Соглашения об описании синтаксиса

В таблице A.1 определена метанотация, используемая для определения грамматики GFT. Она идентична метанотации, используемой нотацией TTCN-3, но отличается от метанотации, используемой формой MSC. В целях улучшения читабельности дополнительно приводится соответствие метанотации MSC и указываются различия.

Таблица A.1/Z.142 – Синтаксическая метанотация

Смысл	TTCN-3	GFT	MSC	Различия
определен с тем, чтобы быть	::=	::=	::=	
После abc следует xyz	abc xyz	abc xyz	abc xyz	
Альтернатива	|	|	|	
0 вариантов или 1 вариант abc	[abc]	[abc]	[abc]	
0 или более вариантов abc	{abc}	{abc}	{abc}*	X
1 или более вариантов abc	{abc} +	{abc} +	{abc} +	
Текстовое группирование	(...)	(...)	{...}	X
Нетерминальный символ abc	abc	abc (для нетерминального формата GFT) или <u>abc</u> (для нетерминальной TTCN)	<abc>	X
Терминальный символ abc	**abc**	**abc**	**abc** or <name> or <character string>	X

A.3 Грамматика формата GFT

A.3.1 Диаграммы

A.3.1.1 Диаграмма управления

```
ControlDiagram ::=
        Frame contains ( ControlHeading ControlBodyArea )

ControlHeading ::=
        TTCN3ModuleKeyword TTCN3ModuleId
        { LocalDefinition [ SemiColon ] }

ControlBodyArea ::=
        { ControlInstanceArea TextLayer ControlEventLayer } set

TextLayer ::=
        { TextArea } set

ControlEventLayer::=
        ControlEventArea | ControlEventArea above ControlEventLayer

ControlEventArea ::=
    (
        InstanceTimerEventArea
    |   ControlActionArea
    |   InstanceInvocationArea
    |   ExecuteTestcaseArea
    |   ControlInlineExpressionArea )
    [ is associated with { CommentArea } set ]
```

A.3.1.2 Диаграмма тестового случая

```
TestcaseDiagram ::=
        Frame contains ( TestcaseHeading TestcaseBodyArea )

TestcaseHeading ::=
        TestcaseKeyword TestcaseIdentifier
        '('[ TestcaseFormalParList ] ')'
        ConfigSpec
        { LocalDefinition [ SemiColon ] }

TestcaseBodyArea ::=
        { InstanceLayer TextLayer InstanceEventLayer PortEventLayer ConnectorLayer } set

InstanceLayer ::=
        { InstanceArea } set


InstanceEventLayer ::=
        InstanceEventArea | InstanceEventArea above InstanceEventLayer

InstanceEventArea ::=
    (
        InstanceSendEventArea
    |   InstanceReceiveEventArea
    |   InstanceCallEventArea
    |   InstanceGetcallEventArea
    |   InstanceReplyEventArea
    |   InstanceGetreplyWithinCallEventArea
    |   InstanceGetreplyOutsideCallEventArea
    |   InstanceRaiseEventArea
    |   InstanceCatchWithinCallEventArea
    |   InstanceCatchTimeoutWithinCallEventArea
    |   InstanceCatchOutsideCallEventArea
    |   InstanceTriggerEventArea
    |   InstanceCheckEventArea
    |   InstanceFoundEventArea
    |   InstanceTimerEventArea
    |   InstanceActionArea
    |   InstanceLabellingArea
    |   InstanceConditionArea
    |   InstanceInvocationArea
    |   InstanceDefaultHandlingArea
    |   InstanceComponentCreateArea
    |   InstanceComponentStartArea
```

```
    |   InstanceComponentStopArea
    |   InstanceInlineExpressionArea )
    [ is associated with { CommentArea } set ]
```

/* СТАТИЧЕСКАЯ СЕМАНТИКА – Область условия, содержащая булево выражение, должна использоваться только во встроенном выражении, т. е. AltArea, и встроенном выражении вызова, т. е. CallArea, */

```
InstanceCallEventArea ::=
        InstanceBlockingCallEventArea
    |   InstanceNonBlockingCallEventArea


PortEventLayer ::=
        PortEventArea | PortEventArea above PortEventLayer

PortEventArea ::=
        PortOutEventArea
    |   PortOtherEventArea

PortOutEventArea ::=
        PortOutMsgEventArea
    |   PortGetcallOutEventArea
    |   PortGetreplyOutEventArea
    |   PortCatchOutEventArea
    |   PortTriggerOutEventArea
    |   PortCheckOutEventArea

PortOtherEventArea ::=
        PortInMsgEventArea
    |   PortCallInEventArea
    |   PortReplyInEventArea
    |   PortRaiseInEventArea
    |   PortConditionArea
    |   PortInvocationArea
    |   PortInlineExpressionArea

ConnectorLayer ::=
    {
        SendArea
    |   ReceiveArea
    |   NonBlockingCallArea
    |   GetcallArea
    |   ReplyArea
    |   GetreplyWithinCallArea
    |   GetreplyOutsideCallArea
    |   RaiseArea
    |   CatchWithinCallArea
    |   CatchOutsideCallArea
    |   TriggerArea
    |   CheckArea
    |   ConditionArea
    |   InvocationArea
    |   InlineExpressionArea
    } set
```

A.3.1.3 Диаграмма функции

```
FunctionDiagram ::=
        Frame contains ( FunctionHeading FunctionBodyArea )

FunctionHeading ::=
        FunctionKeyword FunctionIdentifier
        '('[ FunctionFormalParList ] ')'
        [ RunsOnSpec ] [ ReturnType ]
        { LocalDefinition [ SemiColon ] }

FunctionBodyArea ::=
        TestcaseBodyArea
```

A.3.1.4 Диаграмма альтернативного шага

```
AltstepDiagram ::=
        Frame contains (AltstepHeading AltstepBodyArea )

AltstepHeading ::=
        AltstepKeyword AltstepIdentifier
        '('[AltstepFormalParList ] ')'
```

```
        [ RunsOnSpec ]
        { LocalDefinition [ SemiColon ] }

AltstepBodyArea ::=
        TestcaseBodyArea
```

/* СТАТИЧЕСКАЯ СЕМАНТИКА – Область тела altstep должна содержать единственное встроенное выражение */

A.3.1.5 Комментарии

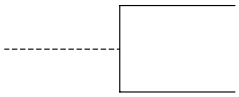
```
TextArea ::=
        TextSymbol
        contains ( { TTCN3Comments } [ MultiWithAttrib ] { TTCN3Comments } )
```

Отметим, что нет четкого правила в отношении комментариев TTCN-3, они поясняются в пункте A.1.4 Рекомендации МСЭ-Т Z.140 [1].

/* СТАТИЧЕСКАЯ СЕМАНТИКА – В диаграмме должно быть не более одного текстового символа, определяющего оператор `with` */

```
TextSymbol ::=
```

```
CommentArea ::=
        EventCommentSymbol contains TTCN3Comments
EventCommentSymbol ::=
```



/* СТАТИЧЕСКАЯ СЕМАНТИКА - Символ комментария может быть присоединен к любому графическому символу в GFT */

A.3.1.6 Диаграмма

```
Frame ::=
```

```
LocalDefinition ::=
            ConstDef
        |   VarInstance
        |   TimerInstance
```

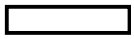
/* СТАТИЧЕСКАЯ СЕМАНТИКА – В LocalDefinition объявления констант и переменных с операторами `create, activate,` и `execute`, а также с функциями, которые включают функции связи, не должны иметь текстовый вид, а должны быть сделаны графически с использованием символов создания, по умолчанию, выполнения и ссылки, соответственно */

A.3.2 Варианты

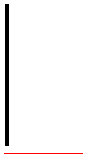
A.3.2.1 Варианты компонента

```
InstanceArea ::=
        ComponentInstanceArea
        | PortInstanceArea

ComponentInstanceArea ::=
        ComponentHeadArea is followed by ComponentBodyArea

ComponentHeadArea::=
        ( MTCOp | SelfOp )
        is followed by ( InstanceHeadSymbol [ contains ComponentType ] )
```

```
InstanceHeadSymbol ::=
```



```
ComponentBodyArea ::=
        InstanceAxisSymbol
        is attached to { InstanceEventArea } set
        is followed by ComponentEndArea

InstanceAxisSymbol::=
```



```
ComponentEndArea ::=
        InstanceEndSymbol
        | StopArea
        | ReturnArea
        | RepeatSymbol
        | GotoArea
```

/* СТАТИЧЕСКАЯ СЕМАНТИКА – Символ возврата должен использоваться только в диаграммах функции */
/* СТАТИЧЕСКАЯ СЕМАНТИКА – Символ повтора должен завершать вариант компонента только диаграммы альтернативного шага */

A.3.2.2 Варианты порта

```
PortInstanceArea ::=
        PortHeadArea is followed by PortBodyArea

PortHeadArea::=
        Port
        is followed by ( InstanceHeadSymbol [ contains PortType ] )

PortBodyArea ::=
        PortAxisSymbol
        is attached to { PortEventArea } set
        is followed by InstanceEndSymbol

PortAxisSymbol::=
```


A.3.2.3 Варианты управления

```
ControlInstanceArea ::=
        ControlInstanceHeadArea is followed by ControlInstanceBodyArea

ControlInstanceHeadArea ::=
        ControlKeyword
        is followed by InstanceHeadSymbol

ControlInstanceBodyArea::=
        InstanceAxisSymbol
        is attached to { ControlEventArea } set
        is followed by ControlInstanceEndArea

ControlInstanceEndArea ::=
        InstanceEndSymbol
```

A.3.2.4 Завершение варианта

```
InstanceEndSymbol ::=
```

```
StopArea::=
        StopSymbol
        is associated with ( Expression )
```

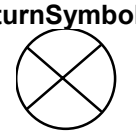
/* СТАТИЧЕСКАЯ СЕМАНТИКА – Выражение (Expression) должно относиться к mtc или к self */

StopSymbol::=

(символ остановки)

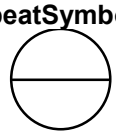
```
ReturnArea::=
        ReturnSymbol
        [ is associated with Expression ]
```

ReturnSymbol::=



(символ возврата)

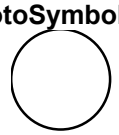
RepeatSymbol::=



(символ повтора)

```
GotoArea::=
        GotoSymbol
        contains LabelIdentifier
```

GotoSymbol::=



(символ операции Goto)

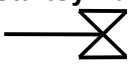
A.3.3 Таймер

```
InstanceTimerEventArea::=
        InstanceTimerStartArea
        | InstanceTimerStopArea
        | InstanceTimeoutArea

InstanceTimerStartArea ::=
        TimerStartSymbol
        is associated with ( TimerRef ["(" TimerValue ")"] )
        is attached to InstanceAxisSymbol
        [is attached to { TimerStopSymbol2 | TimeoutSymbol3 } ]

TimerStartSymbol ::=
        TimerStartSymbol1 | TimerStartSymbol2
```

TimerStartSymbol1 ::=

(1-й символ запуска таймера)

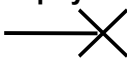
TimerStartSymbol2 ::=



(2-й символ запуска таймера)

```
InstanceTimerStopArea ::=
        TimerStopArea1 | TimerStopArea2

TimerStopArea1 ::=
        TimerStopSymbol1
        is associated with TimerRef
        is attached to InstanceAxisSymbol

TimerStopArea2 ::=
        TimerStopSymbol2
        is attached to InstanceAxisSymbol
        is attached to TimerStartSymbol
```

TimerStopSymbol1 ::=



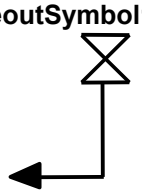
(1-й символ запуска таймера)

TimerStopSymbol2 ::=

(2-й символ запуска таймера)

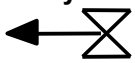
```
InstanceTimeoutArea ::=
        TimeoutArea1 | TimeoutArea2

TimeoutArea1 ::=
        TimeoutSymbol
        is associated with TimerRef
        is attached to InstanceAxisSymbol

TimeoutArea2 ::=
        TimeoutSymbol3
        is attached to InstanceAxisSymbol
        is attached to TimerStartSymbol

TimeoutSymbol ::=
        TimeoutSymbol1 | TimeoutSymbol2
```

TimeoutSymbol1 ::=



(1-й символ тайм-аута)

TimeoutSymbol2 ::=



(2-й символ тайм-аута)

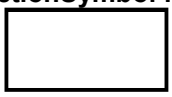
TimeoutSymbol3 ::=

(3-й символ тайм-аута)

A.3.4 Действие

```
InstanceActionArea ::=
        ActionSymbol
        contains { ActionStatement [SemiColon] }+
        is attached to InstanceAxisSymbol
```

ActionSymbol ::=



(символ действия)

```
ActionStatement ::=
        SUTStatements
        | ConnectStatement
        | MapStatement
        | DisconnectStatement
        | UnmapStatement
        | ConstDef
        | VarInstance
        | TimerInstance
        | Assignment
        | LogStatement
        | LoopConstruct
        | ConditionalConstruct
```

/* СТАТИЧЕСКАЯ СЕМАНТИКА – Объявления констант и переменных с операторами `create, activate,` и `execute,` а также с вызовами функций, определяемых пользователем, в блоке действия не должны иметь текстовый вид, а должны быть реализованы графически с использованием символов создания, по умолчанию, выполнения и ссылки, соответственно*/
/* СТАТИЧЕСКАЯ СЕМАНТИКА – Присвоения с операторами `create, activate,` и `execute,` а также с вызовами функций, определяемых пользователем, в блоке действия не должны иметь текстовый вид, а должны быть реализованы графически с использованием символов создания, по умолчанию, выполнения и ссылки, соответственно */
/* СТАТИЧЕСКАЯ СЕМАНТИКА – В блоки действия могут быть включены только те операторы цикла и условные операторы, которые не задействуют операции связи, т. е. только только операторы с “функциями данных” */

```
ControlActionArea ::=
        ActionSymbol
        is attached to InstanceAxisSymbol
        contains { ControlActionStatement [SemiColon] }+

ControlActionStatement ::=
        SUTStatements
        | ConstDef
        | VarInstance
        | TimerInstance
        | Assignment
        | LogStatement
```

/* СТАТИЧЕСКАЯ СЕМАНТИКА – Объявления констант и переменных с операторами `create, activate,` и `execute,` а также с вызовами функций, определяемых пользователем, в блоке действия не должны иметь текстовый вид, а должны быть реализованы графически с использованием символов создания, по умолчанию, выполнения и ссылки, соответственно */
/* СТАТИЧЕСКАЯ СЕМАНТИКА – Присвоения с операторами `create, activate,` и `execute,` а также с вызовами функций, определяемых пользователем, в блоке действия не должны иметь текстовый вид, а должны быть реализованы графически с использованием символов создания, по умолчанию, выполнения и ссылки, соответственно */

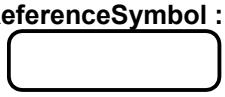
A.3.5 Вызов

```
InvocationArea ::=
        ReferenceSymbol
        contains Invocation
        is attached to InstanceAxisSymbol
        [ is attached to { PortAxisSymbol } set ]
```

/* СТАТИЧЕСКАЯ СЕМАНТИКА – Все варианты порта должны быть охвачены символом ссылки для вызываемой функции, если она работает в соответствии со спецификацией, а также для вызываемого альтернативного шага */
/* СТАТИЧЕСКАЯ СЕМАНТИКА – Только те варианты порта, которые прошли в функцию по параметрам порта, должны быть охвачены символом ссылки для вызываемой функции, не работающей в соответствии со спецификацией. Отметим, что символ ссылки может быть присоединен к вариантам порта, не прошедшим в функцию в качестве параметров порта. */

```
Invocation ::=
        FunctionInstance
        | AltstepInstance
        | ConstDef
        | VarInstance
        | Assignment
```

ReferenceSymbol ::=



Символ ссылки

A.3.5.1 Вызов функции и альтернативного шага на вариантах компонета/управления

```
InstanceInvocationArea ::=
        InstanceInvocationBeginSymbol
        is followed by InstanceInvocationEndSymbol
        is attached to InstanceAxisSymbol
        is attached to InvocationArea

InstanceInvocationBeginSymbol ::=
        VoidSymbol

InstanceInvocationEndSymbol ::=
        VoidSymbol
```

A.3.5.2 Вызов функции и альтернативного шага на портах

```
PortInvocationArea ::=
        PortInvocationBeginSymbol
        is followed by PortInvocationEndSymbol
        is attached to PortAxisSymbol
        is attached to InvocationArea
```

/* СТАТИЧЕСКАЯ СЕМАНТИКА – Только вызовы с вариантами функций и вариантами тестового шага должны быть присоединены к варианту порта, в этом случае все варианты порта должны быть охвачены символом ссылки для вызываемой функции, если она работает в соответствии со спецификацией, а также для вызываемого альтернативного шага */

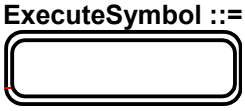
```
PortInvocationBeginSymbol ::=
        VoidSymbol

PortInvocationEndSymbol ::=
        VoidSymbol
```

A.3.5.3 Выполнение Testcase (тестовый случай)

```
ExecuteTestcaseArea ::=
        ExecuteSymbol
        contains TestCaseExecution
        is attached to InstanceAxisSymbol

TestCaseExecution::=
        TestcaseInstance
        | ConstDef
        | VarInstance
        | Assignment
```

/* СТАТИЧЕСКАЯ СЕМАНТИКА – Объявления констант и переменных, а также присвоений должны использовать оператор выполнения в качестве крайнего правого выражения */

ExecuteSymbol ::=



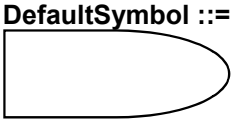
(символ выполнения)

A.3.6 Включение/выключение состояний по умолчанию

```
InstanceDefaultHandlingArea ::=
        DefaultSymbol
        contains DefaultHandling
        is attached to InstanceAxisSymbol

DefaultHandling::=
        ActivateOp
        | DeactivateStatement
        | ConstDef
        | VarInstance
        | Assignment
```

/* СТАТИЧЕСКАЯ СЕМАНТИКА – Объявления констант и переменных, а также присвоений должны использовать оператор включения в качестве крайнего правого выражения */

DefaultSymbol ::=



(символ по умолчанию)

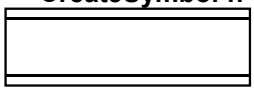
A.3.7 Тестовые компоненты

A.3.7.1 Создание тестовых компонентов

```
InstanceComponentCreateArea ::=
        CreateSymbol
        contains Creation
        is attached to InstanceAxisSymbol

Creation ::=
        CreateOp
        | ConstDef
        | VarInstance
        | Assignment
```

/* СТАТИЧЕСКАЯ СЕМАНТИКА – Объявления констант и переменных, а также присвоений должны использовать оператор создания в качестве крайнего правого выражения */

CreateSymbol ::=

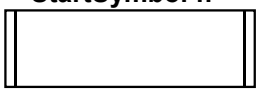


(символ создания)

A.3.7.2 Запуск тестовых компонентов

```
InstanceComponentStartArea ::=
        StartSymbol
        contains StartTCStatement
        is attached to InstanceAxisSymbol
```

StartSymbol ::=



(символ запуска)

A.3.7.3 Остановка тестовых компонентов

```
InstanceComponentStopArea ::=
        StopSymbol
        is associated with ( Expression | AllKeyword )
        is attached to InstanceAxisSymbol
```

/* СТАТИЧЕСКАЯ СЕМАНТИКА – Выражение должно относиться к идентификатору компонента */
/* СТАТИЧЕСКАЯ СЕМАНТИКА – Область остановки варианта компонента должна использоваться в качестве последнего события операнда в символе встроенного выражения, если компонент останавливается самостоятельно (например, self.stop) или останавливает выполнение теста (например, mtc.stop). */

A.3.8 Встроенные выражения

```
InlineExpressionArea ::=
        IfArea
        | ForArea
        | WhileArea
        | DoWhileArea
        | AltArea
        | InterleaveArea
        | CallArea

IfArea ::=
        IfInlineExpressionArea
        is attached to InstanceInlineExpressionBeginSymbol
        [ is attached to InstanceInlineExpressionSeparatorSymbol ]
        is attached to InstanceInlineExpressionEndSymbol
        [ is attached to { PortInlineExpressionBeginSymbol } set
         [ is attached to { PortInlineExpressionSeparatorSymbol } set ]
         is attached to { PortInlineExpressionEndSymbol } set ]
```

/* СТАТИЧЕСКАЯ СЕМАНТИКА – Если SeparatorSymbol содержится в символе встроенного выражения, то символы InstanceInlineExpressionSeparatorSymbols на вариантах компонента и порта используются для присоединения SeparatorSymbol к соответствующим вариантам. */

```
InstanceInlineExpressionBeginSymbol ::=
        VoidSymbol

InstanceInlineExpressionSeparatorSymbol::=
        VoidSymbol

InstanceInlineExpressionEndSymbol ::=
        VoidSymbol

VoidSymbol ::=      .

IfInlineExpressionArea ::=
        InlineExpressionSymbol
        contains ( IfKeyword '(' BooleanExpression ')'
                is followed by OperandArea
                [ is followed by SeparatorSymbol
                    is followed by OperandArea ] )
OperandArea ::=
        ConnectorLayer
```

/* СТАТИЧЕСКАЯ СЕМАНТИКА – Уровень события с областью операнда не должны иметь условия с булевым выражением */

```
ForArea ::=
        ForInlineExpressionArea
        is attached to InstanceInlineExpressionBeginSymbol
        is attached to InstanceInlineExpressionEndSymbol
        [ is attached to { PortInlineExpressionBeginSymbol } set
         is attached to { PortInlineExpressionEndSymbol } set ]

ForInlineExpressionArea ::=
        InlineExpressionSymbol
        contains ( ForKeyword '(' Initial [SemiColon] Final [SemiColon] Step ')'
                is followed by OperandArea )

WhileArea ::=
        WhileInlineExpressionArea
        is attached to InstanceInlineExpressionBeginSymbol
        is attached to InstanceInlineExpressionEndSymbol
        [ is attached to { PortInlineExpressionBeginSymbol } set
         is attached to { PortInlineExpressionEndSymbol } set ]

WhileInlineExpressionArea ::=
        InlineExpressionSymbol
        contains ( WhileKeyword '(' BooleanExpression ')'
                is followed by OperandArea )

DoWhileArea ::=
        DoWhileInlineExpressionArea
        is attached to InstanceInlineExpressionBeginSymbol
        is attached to InstanceInlineExpressionEndSymbol
        [ is attached to { PortInlineExpressionBeginSymbol } set
         is attached to { PortInlineExpressionEndSymbol } set ]
```

```
DoWhileInlineExpressionArea ::=
        InlineExpressionSymbol
        contains ( DoKeyword WhileKeyword '(' BooleanExpression ')'
                is followed by OperandArea )

AltArea ::=
        AltInlineExpressionArea
        is attached to InstanceInlineExpressionBeginSymbol
        { is attached to InstanceInlineExpressionSeparatorSymbol }
        is attached to InstanceInlineExpressionEndSymbol
        [ is attached to { PortInlineExpressionBeginSymbol } set
         [ is attached to { PortInlineExpressionSeparatorSymbol } set ]
         is attached to { PortInlineExpressionEndSymbol } set ]
```

/* СТАТИЧЕСКАЯ СЕМАНТИКА – Ряд InstanceInlineExpressionSeparatorSymbol на каждый из вариантов компонента и порта должен присоединяться к ряду символов SeparatorSymbols, содержащихся в символе встроенного выражения: символ InstanceInlineExpressionSeparatorSymbol на вариантах компонента и порта используется для присоединения символов SeparatorSymbols к соответствующим вариантам. */

```
AltInlineExpressionArea ::=
        InlineExpressionSymbol
        contains ( AltKeyword
                is followed by GuardedOperandArea
                { is followed by SeparatorSymbol
                    is followed by GuardedOperandArea }
                [ is followed by SeparatorSymbol
                    is followed by ElseOperandArea ] )

GuardedOperandArea ::=
        GuardOpLayer is followed by
        ConnectorLayer
```

/* СТАТИЧЕСКАЯ СЕМАНТИКА – Для отдельных операндов встроенного выражения альтернативы сначала следует задать InstanceTimeoutArea на варианте компонента или GuardOpLayer */

```
GuardOpLayer ::=
        DoneArea
        | ReceiveArea
        | TriggerArea
        | GetcallArea
        | CatchOutsideCallArea
        | CheckArea
        | GetreplyOutsideCallArea

ElseOperandArea ::=
        ElseConditionArea
        is followed by ConnectorLayer

InterleaveArea ::=
        InterleaveInlineExpressionArea
        is attached to InstanceInlineExpressionBeginSymbol
        { is attached to InstanceInlineExpressionSeparatorSymbol }
        is attached to InstanceInlineExpressionEndSymbol
        [ is attached to { PortInlineExpressionBeginSymbol } set
         [ is attached to { PortInlineExpressionSeparatorSymbol } set ]
         is attached to { PortInlineExpressionEndSymbol } set ]
```

/* СТАТИЧЕСКАЯ СЕМАНТИКА – Ряд InstanceInlineExpressionSeparatorSymbol на каждый из вариантов компонента и порта должен присоединяться к ряду символов SeparatorSymbols, содержащихся в символе встроенного выражения: символ InstanceInlineExpressionSeparatorSymbol на вариантах компонента и порта используется для присоединения символов SeparatorSymbols к соответствующим вариантам. */

```
InterleaveInlineExpressionArea ::=
        InlineExpressionSymbol
        contains ( InterleavedKeyword
                is followed by UnguardedOperandArea
                { is followed by SeparatorSymbol
                    is followed by UnguardedOperandArea } )

UnguardedOperandArea ::=
        UnguardedOpLayer is followed by
        ConnectorLayer
```

/* СТАТИЧЕСКАЯ СЕМАНТИКА – Уровень символа соединения во встроенном выражении перемежения может не содержать операторы цикла, goto, включения, выключения, остановки или вызова функций*/

```
UnguardedOpLayer ::=
        ReceiveArea
        | TriggerArea
```

```
        | GetcallArea
        | CatchOutsideCallArea
        | CheckArea
        | GetreplyOutsideCallArea

CallArea ::=
        CallInlineExpressionArea
        is attached to InstanceInlineExpressionBeginSymbol
        { is attached to InstanceInlineExpressionSeparatorSymbol }
        is attached to InstanceInlineExpressionEndSymbol
        [ is attached to { PortInlineExpressionBeginSymbol } set
         [ is attached to { PortInlineExpressionSeparatorSymbol } set ]
         is attached to { PortInlineExpressionEndSymbol } set ]
```

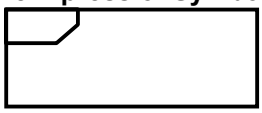
/* СТАТИЧЕСКАЯ СЕМАНТИКА – Число InstanceInlineExpressionSeparatorSymbol на каждый из вариантов компонента и порта должно присоединяться к числу символов SeparatorSymbols, содержащихся в символе встроенного выражения: символ InstanceInlineExpressionSeparatorSymbol на вариантах компонента и порта используется для присоединения символов SeparatorSymbols к соответствующим вариантам. */

```
CallInlineExpressionArea ::=
        InlineExpressionSymbol
        contains ( CallOpKeyword '(' TemplateInstance ')' [ ToClause ]
                is followed by InstanceCallEventArea
                { is followed by SeparatorSymbol
                    is followed by GuardedCallOperandArea } )

GuardedCallOperandArea ::=
        [ GuardedConditionLayer is followed by ]
        CallBodyOpsLayer
        is attached to SuspensionRegionSymbol
        is followed by ConnectorLayer
```

/* СТАТИЧЕСКАЯ СЕМАНТИКА – Для отдельных операндов в GuardedCallOperandArea встроенного выражения вызова сначала следует задать InstanceCatchTimeoutWithinCallEventArea на варианте компонента или CallBodyOpsLayer */

```
GuardedConditionLayer ::=
        BooleanExpressionConditionArea
        | DoneArea

CallBodyOpsLayer ::=
        GetreplyWithinCallArea
        | CatchWithinCallArea
```

InlineExpressionSymbol ::=



SeparatorSymbol::=

- - - - - - - - - - - -

(символ встроенного выражения)
(символ разделения)

A.3.8.1 Встроенные выражения на вариантах компонента

```
InstanceInlineExpressionArea ::=
        InstanceIfArea
        | InstanceForArea
        | InstanceWhileArea
        | InstanceDoWhileArea
        | InstanceAltArea
        | InstanceInterleaveArea
        | InstanceCallArea

InstanceIfArea ::=
        ( InstanceInlineExpressionBeginSymbol
         { is followed by InstanceEventArea }
         { is followed by InstanceInlineExpressionSeparatorSymbol
            { is followed by InstanceEventArea } ]
         is followed by InstanceInlineExpressionEndSymbol )
        is attached to InstanceAxisSymbol
        is attached to IfInlineExpressionArea
```

```
InstanceForArea ::=
        ( InstanceInlineExpressionBeginSymbol
         { is followed by InstanceEventArea }
         is followed by InstanceInlineExpressionEndSymbol )
        is attached to InstanceAxisSymbol
        is attached to ForInlineExpressionArea

InstanceWhileArea ::=
        ( InstanceInlineExpressionBeginSymbol
         { is followed by InstanceEventArea }
         is followed by InstanceInlineExpressionEndSymbol )
        is attached to InstanceAxisSymbol
        is attached to WhileInlineExpressionArea

InstanceDoWhileArea ::=
        ( InstanceInlineExpressionBeginSymbol
         { is followed by InstanceEventArea }
         is followed by InstanceInlineExpressionEndSymbol )
        is attached to InstanceAxisSymbol
        is attached to DoWhileInlineExpressionArea

InstanceAltArea ::=
        ( InstanceInlineExpressionBeginSymbol
         [ is followed by InstanceBooleanExpressionConditionArea ]
         is followed by InstanceGuardArea
         { is followed by InstanceInlineExpressionSeparatorSymbol
            is followed by InstanceGuardArea }
         [ is followed by InstanceInlineExpressionSeparatorSymbol
            is followed by InstanceElseGuardArea ]
         is followed by InstanceInlineExpressionEndSymbol )
        is attached to InstanceAxisSymbol
        is attached to AltInlineExpressionArea

InstanceGuardArea ::=
        ( InstanceInvocationArea
        | InstanceGuardOpArea )
        { is followed by InstanceEventArea }
        is attached to InstanceAxisSymbol
```

/* СТАТИЧЕСКАЯ СЕМАНТИКА – Область вызова варианта должна содержать только вариант альтернативного шага */

```
InstanceGuardOpArea ::=
        (   InstanceTimeoutArea
        |   InstanceReceiveEventArea
        |   InstanceTriggerEventArea
        |   InstanceGetcallEventArea
        |   InstanceGetreplyOutsideCallEventArea
        |   InstanceCatchOutsideCallEventArea
        |   InstanceCheckEventArea
        |   InstanceDoneArea )
        is attached to InstanceAxisSymbol

InstanceElseGuardArea ::=
        ElseConditionArea
        { is followed by InstanceEventArea }
        is attached to InstanceAxisSymbol

InstanceInterleaveArea ::=
        ( InstanceInlineExpressionBeginSymbol
         is followed by InstanceInterleaveGuardArea
         { is followed by InstanceInlineExpressionSeparatorSymbol
            is followed by InstanceInterleaveGuardArea }
         is followed by InstanceInlineExpressionEndSymbol )
        is attached to InstanceAxisSymbol
        is attached to InterleaveInlineExpressionArea

InstanceInterleaveGuardArea ::=
        InstanceGuardOpArea
        { is followed by InstanceEventArea }
        is attached to InstanceAxisSymbol
```

/* СТАТИЧЕСКАЯ СЕМАНТИКА – В области события варианта могут не содержаться операторы цикла, goto, включения, выключения, остановки или вызовов функций */

```
InstanceCallArea ::=
        ( InstanceInlineExpressionBeginSymbol
         [ is followed by InstanceBooleanExpressionConditionArea ]
         [ is followed by InstanceCallOpArea ]
         { is followed by InstanceInlineExpressionSeparatorSymbol
            is followed by InstanceCallGuardArea}
```

```
         is followed by InstanceInlineExpressionEndSymbol )
        is attached to InstanceAxisSymbol
        is attached to CallInlineExpressionArea

InstanceCallOpArea ::=
        InstanceCallEventArea
        is followed by SuspensionRegionSymbol
        [ is attached to InstanceCallTimerStartArea ]
        is attached to InstanceAxisSymbol
        is attached to CallInlineExpressionArea
```

SuspensionRegionSymbol ::=

(символ области приостановки)

```
InstanceCallGuardArea ::=
        SuspensionRegionSymbol
        [ is attached to InstanceGetreplyWithinCallEventArea
                       |    InstanceCatchWithinCallEventArea
                       |    InstanceCatchTimeoutWithinCallEventArea ]
        { is followed by InstanceEventArea }
        is attached to InstanceAxisSymbol
        is attached to CallInlineExpressionArea
```

A.3.8.2 Встроенные выражения на портах

```
PortInlineExpressionArea ::=
        PortIfArea
        | PortForArea
        | PortWhileArea
        | PortDoWhileArea
        | PortAltArea
        | PortInterleaveArea
        | PortCallArea

PortIfArea ::=
        (PortInlineExpressionBeginSymbol
         { is followed by PortEventArea }
         [ is followed by PortInlineExpressionSeparatorSymbol
            { is followed by PortEventArea }]
         is followed by PortInlineExpressionEndSymbol )
        is attached to PortAxisSymbol
        is attached to IfInlineExpressionArea

PortInlineExpressionBeginSymbol ::=
        VoidSymbol

PortInlineExpressionSeparatorSymbol::=
        VoidSymbol

PortInlineExpressionEndSymbol::=
        VoidSymbol

PortForArea ::=
        (PortInlineExpressionBeginSymbol
         { is followed by PortEventArea }
         is followed by PortInlineExpressionEndSymbol )
        is attached to PortAxisSymbol
        is attached to ForInlineExpressionArea

PortWhileArea ::=
        (PortInlineExpressionBeginSymbol
         { is followed by PortEventArea }
         is followed by PortInlineExpressionEndSymbol )
        is attached to PortAxisSymbol
        is attached to WhileInlineExpressionArea

PortDoWhileArea ::=
        ( PortInlineExpressionBeginSymbol
         { is followed by PortEventArea }
         is followed by PortInlineExpressionEndSymbol )
        is attached to PortAxisSymbol
        is attached to DoWhileInlineExpressionArea

PortAltArea ::=
        (PortInlineExpressionBeginSymbol
         [ is followed by PortOutEventArea ]
```

```
         { is followed by PortEventArea }
         { is followed by PortInlineExpressionSeparatorSymbol
         [ is followed by PortOutEventArea ]
            { is followed by PortEventArea } }
         is followed by PortInlineExpressionEndSymbol )
        is attached to PortAxisSymbol
        is attached to AltInlineExpressionArea

PortInterleaveArea ::=
        ( PortInlineExpressionBeginSymbol
         [ is followed by PortOutEventArea ]
         { is followed by PortEventArea }
         { is followed by PortInlineExpressionSeparatorSymbol
         [ is followed by PortOutEventArea ]
            { is followed by PortEventArea } }
         is followed by PortInlineExpressionEndSymbol )
        is attached to PortAxisSymbol
        is attached to InterleaveInlineExpressionArea

PortCallArea ::=
        (PortInlineExpressionBeginSymbol
         [ is followed by PortCallInEventArea]
         { is followed by PortEventArea }
         { is followed by PortInlineExpressionSeparatorSymbol
         [ is followed by PortOutEventArea ]
            { is followed by PortEventArea } }
         is followed by PortInlineExpressionEndSymbol )
        is attached to InstanceAxisSymbol
        is attached to CallInlineExpressionArea
```

A.3.8.3 Встроенные выражения на вариантах управления

```
ControlInlineExpressionArea ::=
        ControlIfArea
        | ControlForArea
        | ControlWhileArea
        | ControlDoWhileArea
        | ControlAltArea
        | ControlInterleaveArea

ControlIfArea ::=
        ( InstanceInlineExpressionBeginSymbol
         [ is followed by ControlEventArea ]
         [ is followed by InstanceInlineExpressionSeparatorSymbol
        is followed by ControlEventArea ]
        is followed by InstanceInlineExpressionEndSymbol )
        is attached to InstanceAxisSymbol
        is attached to IfInlineExpressionArea

ControlForArea ::=
        ( InstanceInlineExpressionBeginSymbol
         [ is followed by ControlEventArea ]
         is followed by InstanceInlineExpressionEndSymbol )
        is attached to InstanceAxisSymbol
        is attached to ForInlineExpressionArea

ControlWhileArea ::=
        ( InstanceInlineExpressionBeginSymbol
         [ is followed by ControlEventArea ]
         is followed by InstanceInlineExpressionEndSymbol )
        is attached to InstanceAxisSymbol
        is attached to WhileInlineExpressionArea

ControlDoWhileArea ::=
        ( InstanceInlineExpressionBeginSymbol
         [ is followed by ControlEventArea ]
         is followed by InstanceInlineExpressionEndSymbol )
        is attached to InstanceAxisSymbol
        is attached to DoWhileInlineExpressionArea

ControlAltArea ::=
        ( InstanceInlineExpressionBeginSymbol
         [ is followed by ControlGuardArea ]
         { is followed by InstanceInlineExpressionSeparatorSymbol
        is followed by ControlGuardArea }
         [ is followed by InstanceInlineExpressionSeparatorSymbol
        is followed by ControlElseGuardArea ]
        is followed by InstanceInlineExpressionEndSymbol )
        is attached to InstanceAxisSymbol
        is attached to AltInlineExpressionArea
```

```
ControlGuardArea ::=
        ( InstanceInvocationArea
        | InstanceTimeoutArea)
        { is followed by ControlEventArea }
        is attached to InstanceAxisSymbol
```

/* СТАТИЧЕСКАЯ СЕМАНТИКА – В области вызова варианта должен содержаться только вариант альтернативного шага */

```
ControlElseGuardArea ::=
        ElseConditionArea
        { is followed by ControlEventArea }
        is attached to InstanceAxisSymbol

ControlInterleaveArea ::=
        ( InstanceInlineExpressionBeginSymbol
         [ is followed by ControlInterleaveGuardArea ]
         { is followed by InstanceInlineExpressionSeparatorSymbol
           is followed by ControlInterleaveGuardArea }
         is followed by InstanceInlineExpressionEndSymbol )
        is attached to InstanceAxisSymbol
        is attached to InterleaveInlineExpressionArea

ControlInterleaveGuardArea ::=
        InstanceTimeoutArea
        { is followed by ControlEventArea }
        is attached to InstanceAxisSymbol
```

/* СТАТИЧЕСКАЯ СЕМАНТИКА – В области события варианта могут не содержаться операторы цикла, goto, включения, выключения, остановки или вызовов функций */

A.3.9 Условие

```
ConditionArea ::=
        PortOperationArea

BooleanExpressionConditionArea ::=
        ConditionSymbol
        contains BooleanExpression
        is attached to InstanceConditionBeginSymbol
        is attached to InstanceConditionEndSymbol
```

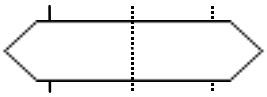
/* СТАТИЧЕСКАЯ СЕМАНТИКА – Булевы выражения должны использоваться в качестве защиты только во встроенных выражениях альтернативы и вызова. Они должны быть присоединены только к одному тестовому компоненту или варианту управления.*/

```
InstanceConditionBeginSymbol ::=
        VoidSymbol

InstanceConditionEndSymbol ::=
        VoidSymbol

DoneArea ::=
        ConditionSymbol
        contains DoneStatement
        is attached to InstanceConditionBeginSymbol
        is attached to InstanceConditionEndSymbol

SetVerdictArea ::=
        ConditionSymbol
        contains SetVerdictText
        is attached to InstanceConditionBeginSymbol
        is attached to InstanceConditionEndSymbol

SetVerdictText ::=
        ( SetVerdictKeyword "(" SingleExpression ")" )
        | pass
        | fail
        | inconc
        | none

/* СТАТИЧЕСКАЯ СЕМАНТИКА - Решением SingleExpression должно быть значение типа заключения */
/* СТАТИЧЕСКАЯ СЕМАНТИКА - SetLocalVerdict не должно использоваться для присвоения ошибки величины
*/
/* СТАТИЧЕСКАЯ СЕМАНТИКА – если используются ключевые слова pass, fail, inconc и fail, то не должна
использоваться форма с ключевым словом setverdict */
PortOperationArea ::=
        ConditionSymbol
        contains PortOperationText
        is attached to InstanceConditionBeginSymbol
```

```
        is attached to InstanceConditionEndSymbol
        is attached to { PortInlineExpressionBeginSymbol }+ set
     is attached to { PortInlineExpressionEndSymbol }+ set ]
        is attached to InstancePortOperationArea
        is attached to PortConditionArea
```

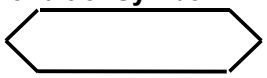
/* СТАТИЧЕСКАЯ СЕМАНТИКА – Символ условия должен быть присоединен ко всем портам или лишь к одному порту */

Если символ условия пересекает символ оси порта того или иного порта, не задействованного в операции этого порта, то символ оси порта прочеркивается:



```
PortOperationText ::=
        ClearOpKeyword
        | StartKeyword
        | StopKeyword

ElseConditionArea ::=
        ConditionSymbol
        contains ElseKeyword
        is attached to InstanceAxisSymbol
```

ConditionSymbol ::=



(символ условия)

A.3.9.1 Условие на вариантах компонента

```
InstanceConditionArea ::=
        InstanceDoneArea
        | InstanceSetVerdictArea
        | InstancePortOperationArea

InstanceBooleanExpressionConditionArea ::=
        InstanceConditionBeginSymbol
        is followed by InstanceConditionEndSymbol
        is attached to InstanceAxisSymbol
        is attached to BooleanExpressionConditionArea

InstanceDoneArea ::=
        InstanceConditionBeginSymbol
        is followed by InstanceConditionEndSymbol
        is attached to InstanceAxisSymbol
        is attached to DoneArea

InstanceSetVerdictArea ::=
        InstanceConditionBeginSymbol
        is followed by InstanceConditionEndSymbol
        is attached to InstanceAxisSymbol
        is attached to SetVerdictArea

InstancePortOperationArea ::=
        InstanceConditionBeginSymbol
        is followed by InstanceConditionEndSymbol
        is attached to InstanceAxisSymbol
        is attached to PortOperationArea
```

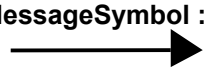
A.3.9.2 Условие на портах

```
PortConditionArea ::=
        PortConditionBeginSymbol
        is followed by PortConditionEndSymbol
        is attached to PortAxisSymbol
        is attached to PortOperationArea

PortConditionBeginSymbol ::=
        VoidSymbol

PortConditionEndSymbol ::=
        VoidSymbol
```

A.3.10 Связь на основе сообщений

```
SendArea ::=
        MessageSymbol
        [ is associated with Type ]
        is associated with ( [ DerivedDef AssignmentChar ] TemplateBody
                             [ ToClause ] )
        is attached to InstanceSendEventArea
        is attached to PortInMsgEventArea

/* СТАТИЧЕСКАЯ СЕМАНТИКА – Тип, если существует, должен быть размещен над символом сообщения */
/* СТАТИЧЕСКАЯ СЕМАНТИКА – Полученное определение, если существует, должно быть размещено под символом
сообщения */
/* СТАТИЧЕСКАЯ СЕМАНТИКА – Шаблон должен быть размещен под символом сообщения */
/* СТАТИЧЕСКАЯ СЕМАНТИКА – ToClause, если существует, должен быть размещен под символом сообщения */

ReceiveArea ::=
        MessageSymbol
        [ is associated with Type ]
        is associated with ( [ [ DerivedDef AssignmentChar ] TemplateBody ]
                             [ FromClause ] [ PortRedirect ] )
        is attached to InstanceReceiveEventArea
        is attached to PortOutMsgEventArea

/* СТАТИЧЕСКАЯ СЕМАНТИКА – Тип, если существует, должен быть размещен над символом сообщения */
/* СТАТИЧЕСКАЯ СЕМАНТИКА – Полученное определение, если существует, должно быть размещено под символом
сообщения */
/* СТАТИЧЕСКАЯ СЕМАНТИКА – Шаблон должен быть размещен под символом сообщения */
/* СТАТИЧЕСКАЯ СЕМАНТИКА – FromClause, если существует, должен быть размещен под символом сообщения */
/* СТАТИЧЕСКАЯ СЕМАНТИКА – PortRedirect, если существует, должен быть размещен под символом сообщения */
```

MessageSymbol ::=



(символ сообщения)

A.3.10.1 Связь на основе сообщений на вариантах компонента

```
InstanceSendEventArea ::=
        MessageOutSymbol
        is attached to InstanceAxisSymbol
        is attached to MessageSymbol

MessageOutSymbol ::=
        VoidSymbol
```

The VoidSymbol is a geometric point without spatial extension.

```
InstanceReceiveEventArea ::=
        MessageInSymbol
        is attached to InstanceAxisSymbol
        is attached to MessageSymbol

MessageInSymbol ::=
        VoidSymbol
```

A.3.10.2 Связь на основе сообщений на вариантах порта

```
PortInMsgEventArea ::=
        MessageInSymbol
        is attached to PortAxisSymbol
        is attached to MessageSymbol
PortOutMsgEventArea ::=
        MessageOutSymbol
        is attached to PortAxisSymbol
        is attached to MessageSymbol
```

A.3.11 Связь на основе подписи

```
NonBlockingCallArea ::=
        MessageSymbol
        is associated with CallKeyword [ Signature ]
        is associated with ( [ DerivedDef AssignmentChar ] TemplateBody
                             [ ToClause ] )
        is attached to InstanceCallEventArea
        is attached to PortCallInEventArea
```

```
/* СТАТИЧЕСКАЯ СЕМАНТИКА – Подпись, если существует. должна быть размещена над символом сообщения */
/* СТАТИЧЕСКАЯ СЕМАНТИКА – Полученное определение, если существует, должно быть размещено под символом
сообщения */
/* СТАТИЧЕСКАЯ СЕМАНТИКА – Шаблон должен быть размещен под символом сообщения */
/* СТАТИЧЕСКАЯ СЕМАНТИКА – ToClause, если существует, должен быть размещен под символом сообщения */

GetcallArea ::=
        MessageSymbol
        is associated with GetcallKeyword [ Signature ]
        is associated with ( [ [ DerivedDef AssignmentChar ] TemplateBody ]
                             [ FromClause ] [ PortRedirectWithParam ] )
        is attached to InstanceGetcallEventArea
        is attached to PortGetcallOutEventArea

/* СТАТИЧЕСКАЯ СЕМАНТИКА – Подпись, если существует. должна быть размещена над символом сообщения */
/* СТАТИЧЕСКАЯ СЕМАНТИКА – Полученное определение, если существует, должно быть размещено под символом
сообщения */
/* СТАТИЧЕСКАЯ СЕМАНТИКА – Шаблон должен быть размещен под символом сообщения */
/* СТАТИЧЕСКАЯ СЕМАНТИКА – ToClause, если существует, должен быть размещен под символом сообщения */
/* СТАТИЧЕСКАЯ СЕМАНТИКА – PortRedirect, если существует, должен быть размещен под символом сообщения */

ReplyArea ::=
        MessageSymbol
        is associated with ReplyKeyword [ Signature ]
        is associated with ( [ DerivedDef AssignmentChar ] TemplateBody
                             [ ReplyValue ] [ ToClause ] )
        is attached to InstanceReplyEventArea
        is attached to PortReplyInEventArea

/* СТАТИЧЕСКАЯ СЕМАНТИКА – Подпись, если существует, должна быть размещена над символом сообщения */
/* СТАТИЧЕСКАЯ СЕМАНТИКА – Полученное определение, если существует, должно быть размещено под символом
сообщения */
/* СТАТИЧЕСКАЯ СЕМАНТИКА – Шаблон, если существует, должен быть размещен под символом сообщения */
/* СТАТИЧЕСКАЯ СЕМАНТИКА – Значение ответа, если существует, должно быть размещено под символом сообщения */
/* СТАТИЧЕСКАЯ СЕМАНТИКА – ToClause, если существует, должен быть размещен под символом сообщения */

GetreplyWithinCallArea ::=
        MessageSymbol
        is attached to SuspensionRegionSymbol
        is associated with GetreplyKeyword [ Signature ]
        is associated with ( [ [ DerivedDef AssignmentChar ] TemplateBody ]
                             [ ValueMatchSpec ]
                             [ FromClause ] [ PortRedirectWithParam ] )
        is attached to InstanceGetreplyEventArea
        is attached to PortGetreplyOutEventArea

/* СТАТИЧЕСКАЯ СЕМАНТИКА – Подпись, если существует, должна быть размещена над символом сообщения */
/* СТАТИЧЕСКАЯ СЕМАНТИКА – Полученное определение, если существует, должно быть размещено под символом
сообщения */
/* СТАТИЧЕСКАЯ СЕМАНТИКА – Шаблон, если существует, должен быть размещен под символом сообщения */
/* СТАТИЧЕСКАЯ СЕМАНТИКА – Спецификация соответствия величине, если существует, должна быть размещена под
символом сообщения */
/* СТАТИЧЕСКАЯ СЕМАНТИКА – FromClause, если существует, должен быть размещен под символом сообщения */
/* СТАТИЧЕСКАЯ СЕМАНТИКА – PortRedirect, если существует, должен быть размещен под символом сообщения */

GetreplyOutsideCallArea ::=
        MessageSymbol
        is associated with GetreplyKeyword [ Signature ]
        is associated with ( [ [ DerivedDef AssignmentChar ] TemplateBody ]
                             [ ValueMatchSpec ]
                             [ FromClause ] [ PortRedirectWithParam ] )
        is attached to InstanceGetreplyEventArea
        is attached to PortGetreplyOutEventArea

/* СТАТИЧЕСКАЯ СЕМАНТИКА – Подпись, если существует, должна быть размещена над символом сообщения */
/* СТАТИЧЕСКАЯ СЕМАНТИКА – Полученное определение, если существует, должно быть размещено под символом
сообщения */
/* СТАТИЧЕСКАЯ СЕМАНТИКА – Шаблон, если существует, должен быть размещен под символом сообщения */
/* СТАТИЧЕСКАЯ СЕМАНТИКА – Спецификация соответствия величине, если существует, должна быть размещена под
символом сообщения */
/* СТАТИЧЕСКАЯ СЕМАНТИКА – FromClause, если существует, должен быть размещен под символом сообщения */
/* СТАТИЧЕСКАЯ СЕМАНТИКА – PortRedirect, если существует, должен быть размещен под символом сообщения */

RaiseArea ::=
        MessageSymbol
        is associated with RaiseKeyword Signature [ ',' Type ]
```

```
        is associated with ( [ DerivedDef AssignmentChar ] TemplateBody
                             [ ToClause ] )
        is attached to InstanceRaiseEventArea
        is attached to PortRaiseInEventArea

/* СТАТИЧЕСКАЯ СЕМАНТИКА – Подпись должна быть размещена над символом сообщения */
/* СТАТИЧЕСКАЯ СЕМАНТИКА – Тип исключения, если существует, должен быть размещен над символом сообщения */
/* СТАТИЧЕСКАЯ СЕМАНТИКА – Полученное определение, если существует, должно быть размещено под символом
сообщения */
/* СТАТИЧЕСКАЯ СЕМАНТИКА – Шаблон должен быть размещен под символом сообщения */
/* СТАТИЧЕСКАЯ СЕМАНТИКА – ToClause, если существует, должен быть размещен под символом сообщения */

CatchWithinCallArea ::=
        MessageSymbol
        is attached to SuspensionRegionSymbol
        is associated with CatchKeyword Signature [ ',' Type ]
        is associated with ( [ [ DerivedDef AssignmentChar ] TemplateBody ]
                             [ FromClause ] [ PortRedirect ] )
        is attached to InstanceCatchEventArea
        is attached to PortCatchOutEventArea

/* СТАТИЧЕСКАЯ СЕМАНТИКА – Подпись должна быть размещена над символом сообщения */
/* СТАТИЧЕСКАЯ СЕМАНТИКА – Тип исключения, если существует, должен быть размещен над символом сообщения */
/* СТАТИЧЕСКАЯ СЕМАНТИКА – Полученное определение, если существует, должно быть размещено под символом
сообщения */
/* СТАТИЧЕСКАЯ СЕМАНТИКА – Шаблон, если существует, должен быть размещен под символом сообщения */
/* СТАТИЧЕСКАЯ СЕМАНТИКА – FromClause, если существует, должен быть размещен под символом сообщения */
/* СТАТИЧЕСКАЯ СЕМАНТИКА – PortRedirect, если существует, должен быть размещен под символом сообщения */

CatchOutsideCallArea ::=
        MessageSymbol
        is associated with CatchKeyword Signature [ ',' Type ]
        is associated with ( [ [ DerivedDef AssignmentChar ] TemplateBody ]
                             [ FromClause ] [ PortRedirect ] )
        is attached to InstanceCatchEventArea
        is attached to PortCatchOutEventArea

/* СТАТИЧЕСКАЯ СЕМАНТИКА – Подпись должна быть размещена над символом сообщения */
/* СТАТИЧЕСКАЯ СЕМАНТИКА – Тип исключения, если существует, должен быть размещен над символом сообщения */
/* СТАТИЧЕСКАЯ СЕМАНТИКА – Полученное определение, если существует, должно быть размещено под символом
сообщения */
/* СТАТИЧЕСКАЯ СЕМАНТИКА – Шаблон, если существует, должен быть размещен под символом сообщения */
/* СТАТИЧЕСКАЯ СЕМАНТИКА – FromClause, если существует, должен быть размещен под символом сообщения */
/* СТАТИЧЕСКАЯ СЕМАНТИКА – PortRedirect, если существует, должен быть размещен под символом сообщения */
```

A.3.11.1 Связь на основе подписи на вариантах компонента

```
InstanceBlockingCallEventArea ::=
        InstanceSendEventArea
        [ is attached to InstanceCallTimerStartArea ]
        is attached to SuspensionRegionSymbol

InstanceCallTimerStartArea ::=
        CallTimerStartSymbol
        is associated with TimerValue
        is attached to InstanceAxisSymbol
        is attached to SuspensionRegionSymbol
        [is attached to CallTimeoutSymbol3 ]
```

CallTimerStartSymbol ::=

(символ запуска таймера вызова)

```
InstanceNonBlockingCallEventArea ::=
        InstanceSendEventArea

InstanceGetcallEventArea ::=
        InstanceReceiveEventArea

InstanceReplyEventArea ::=
        InstanceSendEventArea

InstanceGetreplyWithinCallEventArea ::=
        InstanceReceiveEventArea
        is attached to SuspensionRegionSymbol

InstanceGetreplyOutsideCallEventArea ::=
```

```
        InstanceReceiveEventArea

InstanceRaiseEventArea ::=
        InstanceSendEventArea

InstanceCatchWithinCallEventArea ::=
        InstanceReceiveEventArea
        is attached to SuspensionRegionSymbol

InstanceCatchTimeoutWithinCallEventArea ::=
        CallTimeoutSymbol
        is attached to SuspensionRegionSymbol
        is attached to InstanceAxisSymbol
```

CallTimeoutSymbol ::=

(символ тайм-аута вызова)

```
InstanceCatchOutsideCallEventArea ::=
        InstanceReceiveEventArea
```

A.3.11.2 Связь на основе подписи на портах

```
PortGetcallOutEventArea::=
        PortOutMsgEventArea

PortGetreplyOutEventArea::=
        PortOutMsgEventArea

PortCatchOutEventArea::=
        PortOutMsgEventArea

PortCallInEventArea::=
        PortInMsgEventArea

PortReplyInEventArea::=
        PortInMsgEventArea

PortRaiseInEventArea::=
        PortInMsgEventArea
```

A.3.12 Срабатывание и проверка

A.3.12.1 Срабатывание и проверка на вариантах компонента

```
TriggerArea ::=
        MessageSymbol
        is associated with ( TriggerOpKeyword [ Type ] )
        is associated with ( [ [ DerivedDef AssignmentChar ] TemplateBody ]
                             [ FromClause ] [ PortRedirect ] )
        is attached to ReceiveEventArea
        is attached to PortOutMsgEventArea

/* СТАТИЧЕСКАЯ СЕМАНТИКА – Ключевое слово trigger должно быть размещено над символом сообщения */
/* СТАТИЧЕСКАЯ СЕМАНТИКА – Тип, если существует, должен быть размещен над символом сообщения */
/* СТАТИЧЕСКАЯ СЕМАНТИКА – Полученное определение, если существует, должно быть размещено под символом
сообщения */
/* СТАТИЧЕСКАЯ СЕМАНТИКА – Шаблон, если существует, должен быть размещен под символом сообщения */
/* СТАТИЧЕСКАЯ СЕМАНТИКА – FromClause, если существует, должен быть размещен под символом сообщения */
/* СТАТИЧЕСКАЯ СЕМАНТИКА – PortRedirect, если существует, должен быть размещен под символом сообщения */

CheckArea ::=
        MessageSymbol
        is associated with ( CheckOpKeyword [ CheckOpInformation ] )
        is associated with CheckData
        is attached to ReceiveEventArea
        is attached to PortOutMsgEventArea

/* СТАТИЧЕСКАЯ СЕМАНТИКА – Ключевое слово check должно быть размещено над символом сообщения */
/* СТАТИЧЕСКАЯ СЕМАНТИКА – Информация CheckOp, если существует, должна быть размещена над символом сообщения
*/
/* СТАТИЧЕСКАЯ СЕМАНТИКА – Данные проверки, если существуют, должны быть размещены под символом сообщения */

CheckOpInformation ::=
        Type
        | ( GetCallOpKeyword [ Signature ] )
```

```
        | ( GetReplyOpKeyword [ Signature ] )
        | ( CatchOpKeyword Signature [ Type ] )

CheckData ::=
        ( [ [ DerivedDef AssignmentChar ] TemplateBody [ ValueMatchSpec ] ]
         [ FromClause ] [ PortRedirect | PortRedirectWithParam ] )
        | ( [ FromClause ] [ PortRedirectSymbol SenderSpec ] )
```

/* СТАТИЧЕСКАЯ СЕМАНТИКА – Спецификация соответствия значениям должна использоваться только в сочетании с getreply */
/* СТАТИЧЕСКАЯ СЕМАНТИКА – PortRedirect с параметрами должен использоваться только в сочетании с getcall и getreply */

```
InstanceTriggerEventArea ::=
        InstanceReceiveEventArea

InstanceCheckEventArea ::=
        InstanceReceiveEventArea
```

A.3.12.2 Срабатывание и проверка на вариантах порта

```
PortTriggerOutEventArea ::=
        PortOutMsgEventArea

PortCheckOutEventArea ::=
        PortOutMsgEventArea
```

A.3.13 Обработка связи с любого порта

```
InstanceFoundEventArea ::=
        FoundSymbol
        contains FoundEvent
        is attached to InstanceAxisSymbol
```

/* СТАТИЧЕСКАЯ СЕМАНТИКА – Идентификатор маркера должен быть расположен внутри круга символа маркирования */

```
FoundEvent ::=
        FoundMessage
    |   FoundTrigger
    |   FoundGetCall
    |   FoundGetReply
    |   FoundCatch
    |   FoundCheck

FoundMessage ::=
        FoundSymbol
        [ is associated with Type ]
        is associated with ( [ [ DerivedDef AssignmentChar ] TemplateBody ]
                             [ FromClause ] [ PortRedirect ] )
        is attached to InstanceAxisSymbol
```

/* СТАТИЧЕСКАЯ СЕМАНТИКА – Тип, если существует, должен быть размещен над символом сообщения */
/* СТАТИЧЕСКАЯ СЕМАНТИКА – Полученное определение, если существует, должно быть размещено под символом сообщения */
/* СТАТИЧЕСКАЯ СЕМАНТИКА – Шаблон, если существует, должен быть размещен под символом сообщения */
/* СТАТИЧЕСКАЯ СЕМАНТИКА – FromClause, если существует, должен быть размещен под символом сообщения */
/* СТАТИЧЕСКАЯ СЕМАНТИКА – PortRedirect, если существует, должен быть размещен под символом сообщения */

```
FoundTrigger ::=
        FoundSymbol
        is associated with ( TriggerOpKeyword [ Type ] )
        is associated with ( [ [ DerivedDef AssignmentChar ] TemplateBody ]
                             [ FromClause ] [ PortRedirect ] )
        is attached to InstanceAxisSymbol
```

/* СТАТИЧЕСКАЯ СЕМАНТИКА – Ключевое слово trigger должно быть размещено над символом сообщения */
/* СТАТИЧЕСКАЯ СЕМАНТИКА – Тип, если существует, должен быть размещен над символом сообщения */
/* СТАТИЧЕСКАЯ СЕМАНТИКА – Полученное определение, если существует, должно быть размещено под символом сообщения */
/* СТАТИЧЕСКАЯ СЕМАНТИКА – Шаблон, если существует, должен быть размещен под символом сообщения */
/* СТАТИЧЕСКАЯ СЕМАНТИКА – FromClause, если существует, должен быть размещен под символом сообщения */
/* СТАТИЧЕСКАЯ СЕМАНТИКА – PortRedirect, если существует, должен быть размещен под символом сообщения */

```
FoundGetCall ::=
        FoundSymbol
        is associated with GetcallKeyword [ Signature ]
        is associated with ( [ [ DerivedDef AssignmentChar ] TemplateBody ]
                             [ FromClause ] [ PortRedirectWithParam ] )
        is attached to InstanceAxisSymbol
```

```
/* СТАТИЧЕСКАЯ СЕМАНТИКА – Подпись должна быть размещена над символом сообщения */
/* СТАТИЧЕСКАЯ СЕМАНТИКА – Полученное определение, если существует, должно быть размещено под символом
сообщения */
/* СТАТИЧЕСКАЯ СЕМАНТИКА – Шаблон, если существует, должен быть размещен под символом сообщения */
/* СТАТИЧЕСКАЯ СЕМАНТИКА – FromClause, если существует, должен быть размещен под символом сообщения */
/* СТАТИЧЕСКАЯ СЕМАНТИКА – PortRedirect, если существует, должен быть размещен под символом сообщения */

FoundGetReply ::=
        FoundSymbol
        is associated with GetreplyKeyword [ Signature ]
        is associated with ( [ [ DerivedDef AssignmentChar ] TemplateBody ]
                             [ ValueMatchSpec ]
                             [ FromClause ] [ PortRedirectWithParam ] )
        is attached to InstanceAxisSymbol

/* СТАТИЧЕСКАЯ СЕМАНТИКА – Подпись, если существует, должна быть размещена над символом сообщения */
/* СТАТИЧЕСКАЯ СЕМАНТИКА – Полученное определение, если существует, должно быть размещено под символом
сообщения */
/* СТАТИЧЕСКАЯ СЕМАНТИКА – Шаблон, если существует, должен быть размещен под символом сообщения */
/* СТАТИЧЕСКАЯ СЕМАНТИКА – Спецификация соответствия значениям, если существует, должен быть размещена под
символом сообщения  */
/* СТАТИЧЕСКАЯ СЕМАНТИКА – FromClause, если существует, должен быть размещен под символом сообщения */
/* СТАТИЧЕСКАЯ СЕМАНТИКА – PortRedirect, если существует, должен быть размещен под символом сообщения */

FoundCatch ::=
        FoundSymbol
        is associated with CatchKeyword Signature [ ',' Type ]
        is associated with ( [ [ DerivedDef AssignmentChar ] TemplateBody ]
                             [ FromClause ] [ PortRedirect ] )
        is attached to InstanceAxisSymbol

/* СТАТИЧЕСКАЯ СЕМАНТИКА – Подпись должна быть размещена над символом сообщения */
/* СТАТИЧЕСКАЯ СЕМАНТИКА – Тип исключения, если существует, должен быть размещен над символом сообщения */
/* СТАТИЧЕСКАЯ СЕМАНТИКА – Полученное определение, если существует, должно быть размещено под символом
сообщения */
/* СТАТИЧЕСКАЯ СЕМАНТИКА – Шаблон, если существует, должен быть размещен под символом сообщения */
/* СТАТИЧЕСКАЯ СЕМАНТИКА – FromClause, если существует, должен быть размещен под символом сообщения */
/* СТАТИЧЕСКАЯ СЕМАНТИКА – PortRedirect, если существует, должен быть размещен под символом сообщения */

FoundCheck ::=
        FoundSymbol
        is associated with ( CheckOpKeyword [ CheckOpInformation ] )
        is associated with CheckData
        is attached to ReceiveEventArea
        is attached to InstanceAxisSymbol

/* СТАТИЧЕСКАЯ СЕМАНТИКА – Ключевое слово check должен быть размещено над символом сообщения */
/* СТАТИЧЕСКАЯ СЕМАНТИКА – Информация CheckOp, если существует, должна быть размещена над символом сообщения
*/
/* СТАТИЧЕСКАЯ СЕМАНТИКА – Данные проверки, если существуют, должны быть размещены под символом сообщения */
```

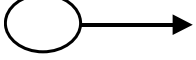
FoundSymbol ::=

(символ обнаружения)

A.3.14 Маркирование

```
InstanceLabellingArea ::=
        LabellingSymbol
        contains LabelIdentifier
        is attached to InstanceAxisSymbol

/* СТАТИЧЕСКАЯ СЕМАНТИКА – Идентификатор маркера должен быть расположен внутри круга символа маркирования */
```

LabellingSymbol ::=

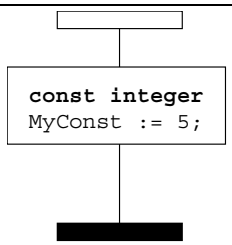


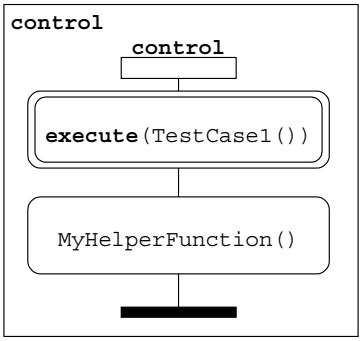
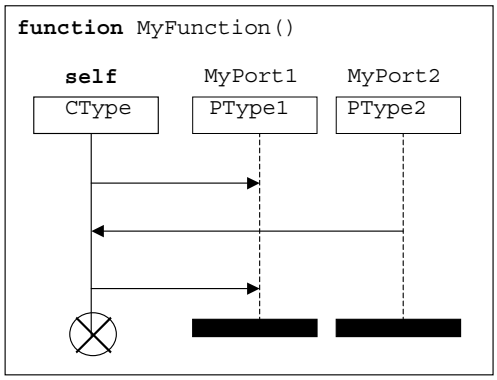
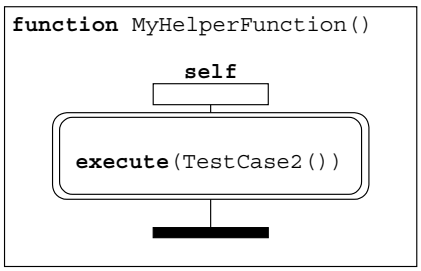
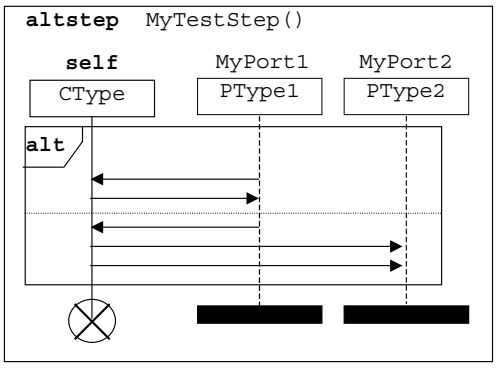
(символ маркирования)

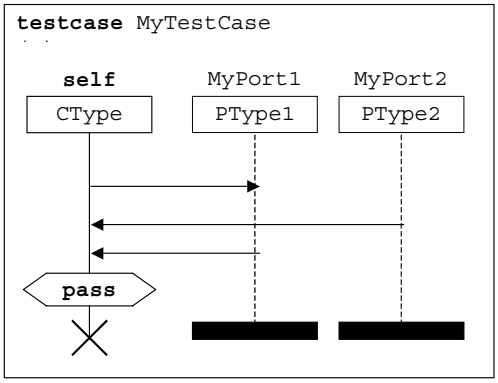
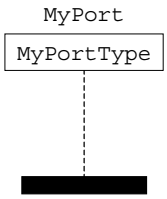
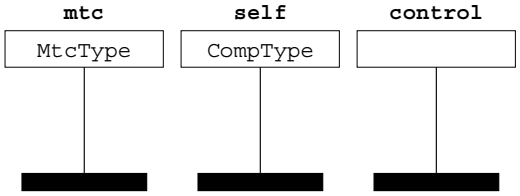
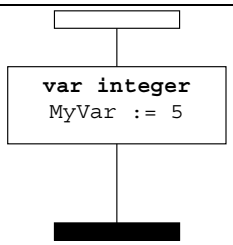
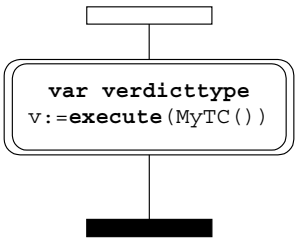
Приложение B

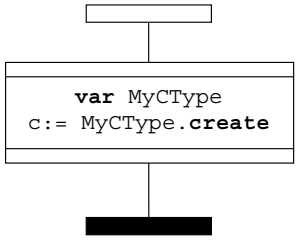
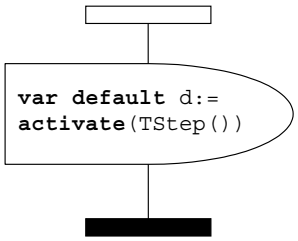
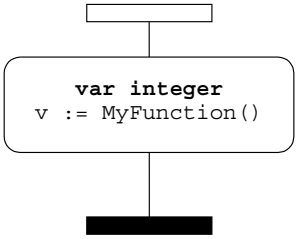
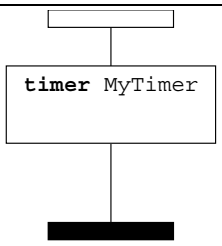
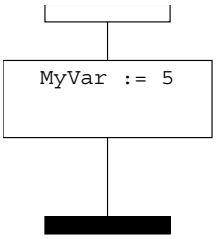
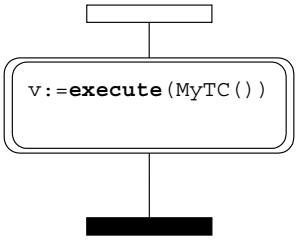
Справочное руководство по GFT

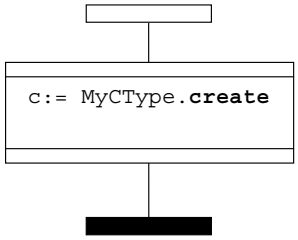
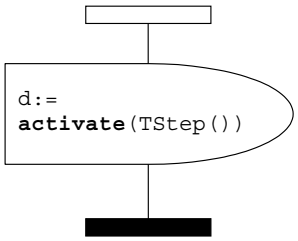
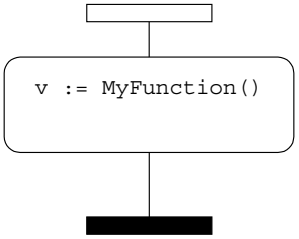
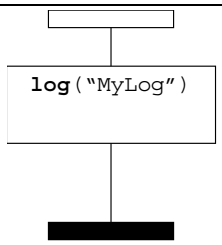
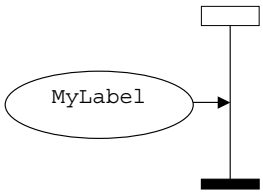
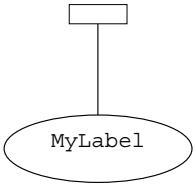
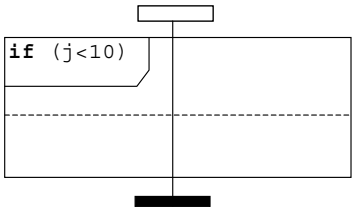
В настоящем приложении перечисляются основные элементы языка TTCN-3 и их представление в GFT. С полным описанием символов GFT и их использованием можно ознакомиться в основном тексте.

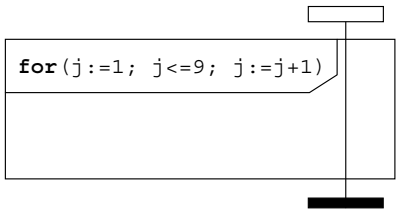
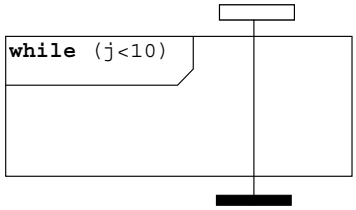
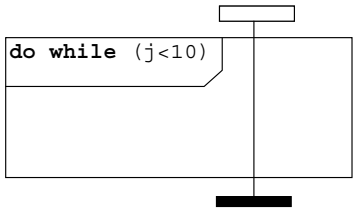
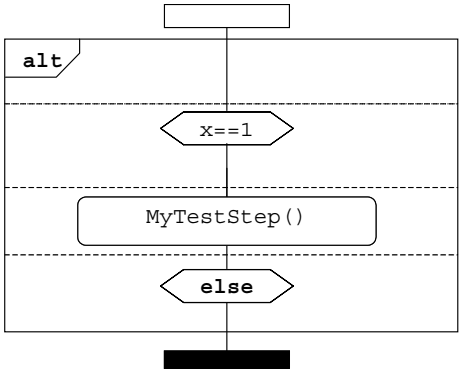
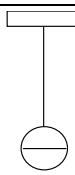
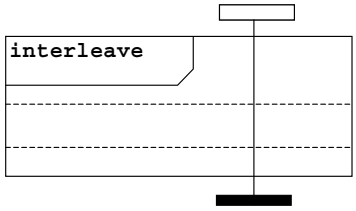
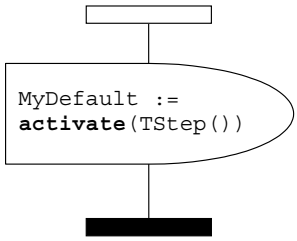
Элемент языка	Соответствующее ключевое слово	Символы GFT, если существуют, и типичное использование	Примечание
Определения модулей			
Определение модуля TTCN-3	**module**		Специальный символ GFT отсутствует, т. е. может быть использован базовый язык или другой формат представления.
Импорт определений из другого модуля	**import**		Специальный символ GFT отсутствует, т. е. может быть использован базовый язык или другой формат представления.
Группирование определений	**group**		Специальный символ GFT отсутствует, т. е. может быть использован базовый язык или другой формат представления.
Определения типа данных	**type**		Специальный символ GFT отсутствует, т. е. может быть использован базовый язык или другой формат представления.
Определения портов связи	**port**		Специальный символ GFT отсутствует, т. е. может быть использован базовый язык или другой формат представления.
Определения тестовых компонентов	**component**		Специальный символ GFT отсутствует, т. е. может быть использован базовый язык или другой формат представления.
Определения подписи	**signature**		Специальный символ GFT отсутствует, т. е. может быть использован базовый язык или другой формат представления.
Определения внешних функций/констант	**external**		Специальный символ GFT отсутствует, т. е. может быть использован базовый язык или другой формат представления.
Определения констант	**const**	**const** integer MyConst := 5;	Текстовое объявление констант в заголовке управления, диаграмме тестового случая, тестового шага или функции.
			Местное объявление константы в блоке действия.
Определения шаблонов данных/подписей	**template**		Специальный символ GFT отсутствует, т. е. может быть использован базовый язык или другой формат представления.

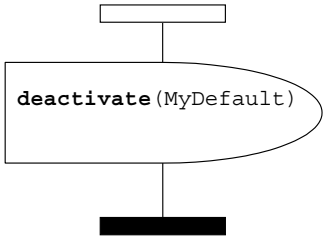
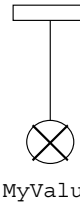
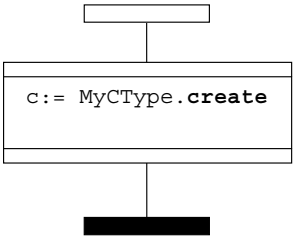
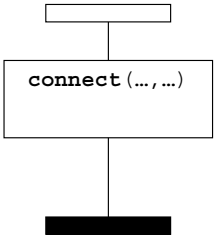
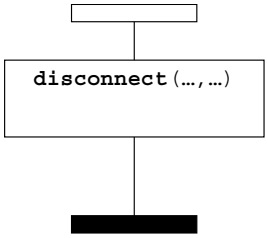
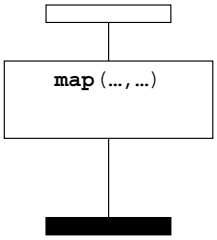
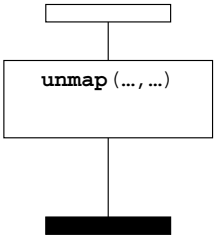
Элемент языка	Соответствующее ключевое слово	Символы GFT, если существуют, и типичное использование	Примечание
Определения управления	**control**		Диаграмма управления GFT представляет часть управления модуля TTCN-3.
Определения функции	**function**		Диаграммы функций GFT используются для представления функций.
			Диаграммы функций GFT могут быть определены в целях структуризации поведения части управления модуля TTCN-3.
Определения альтернативных шагов	**altstep**		Диаграммы альтернативных шагов GFT используются для представления альтернативных шагов.

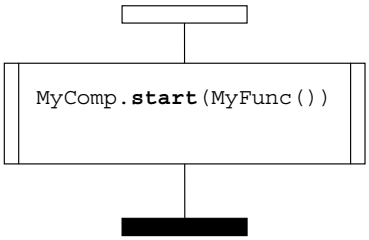
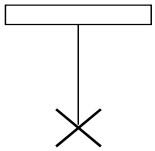
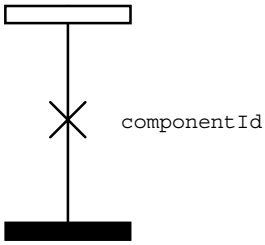
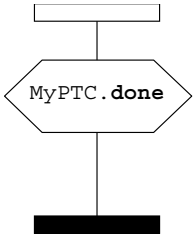
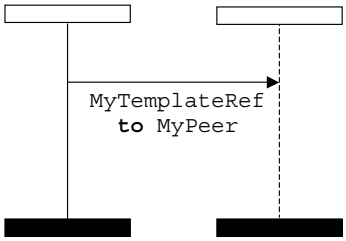
Элемент языка	Соответствующее ключевое слово	Символы GFT, если существуют, и типичное использование	Примечание
Определения тестовых случаев	**testcase**		Диаграммы тестовых случаев GFT используются для представления тестовых случаев.
Использование вариантов компонент и портов			
Вариант порта			Тот или иной порт в диаграмме тестового случая, тестового шага и функции представлен вариантом со штриховой линией варианта. Название порта указано выше, а тип порта (дополнительно) описан в заголовке варианта.
Вариант тестового компонента			Вариант **mtc** представляет собой основной тестовый компонент в диаграмме тестового случая. Вариант **self** представляет собой тестовый компонент в диаграмме тестового шага или функции. Вариант **control** представляет собой вариант, который реализует часть управления модуля в диаграмме управления.
Объявления			
Объявления переменных	**var**	**var integer** MyVar := 5	Текстовое объявление переменной в заголовке управления, диаграмме тестового случая, тестового шага или функции.
			Объявление переменной в блоке действия.
			Объявление переменной в символе выполнения тестового случая.

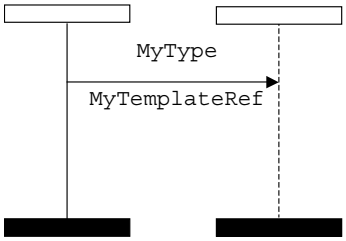
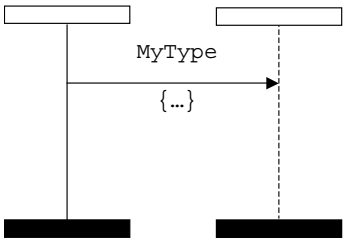
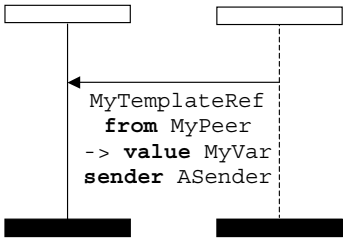
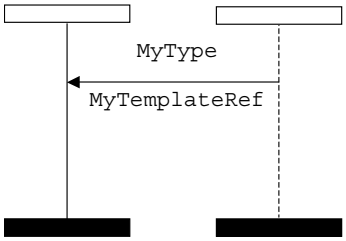
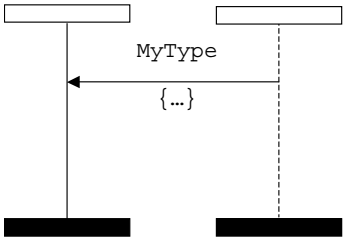
Элемент языка	Соответствующее ключевое слово	Символы GFT, если существуют, и типичное использование	Примечание
			Объявление переменной в пределах символа создания тестового случая.
			Объявление переменной в пределах символа включения состояния по умолчанию.
			Объявление переменной в пределах символа ссылки.
Объявления таймеров	**timer**	**timer** MyTimer	Текстовое объявления таймера в заголовка управления, диаграмме тестового случая, тестового шага или функции.
			Объявление таймера в блоке действия.
Базовые программные операторы			
Выражения	**(…)**		Специальный символ GFT отсутствует, т. е. можно использовать базовый язык или другой формат представления.
Присвоения	**:=**		Присвоение в блоке действия.
			Присвоение в символе выполнения тестового случая.

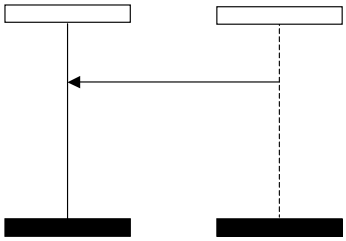
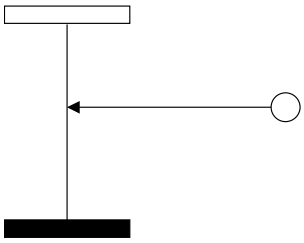
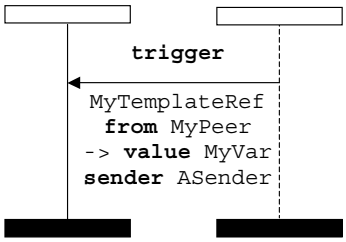
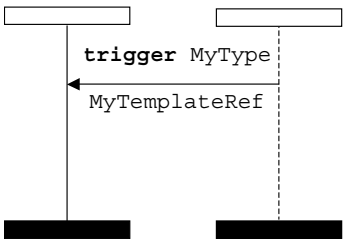
Элемент языка	Соответствующее ключевое слово	Символы GFT, если существуют, и типичное использование	Примечание
			Присвоение в символе создания тестового компонента.
			Присвоение в символе включения состояния по умолчанию.
			Присвоение в символе ссылки.
Регистрация	**log**		Оператор log располагают в блоке действия.
Маркер и Goto (переход)	**label**		Определение маркера.
	goto		Перейти к маркеру.
Оператор if-else	**if (…) {…} else {…}**		

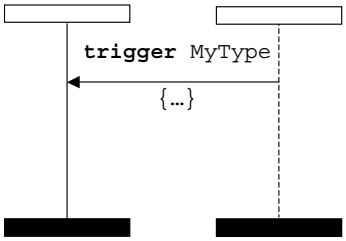
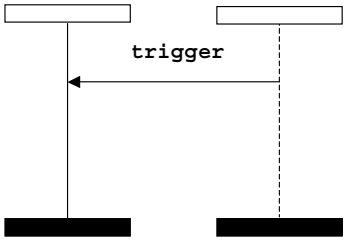
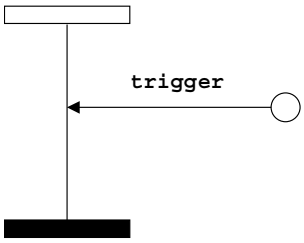
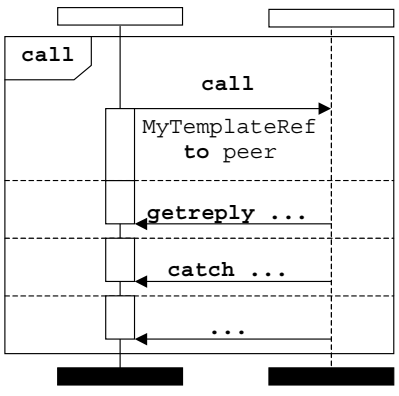
Элемент языка	Соответствующее ключевое слово	Символы GFT, если существуют, и типичное использование	Примечание
Цикл **for**	**for (…) {…}**		
Цикл while	**while (…) {…}**		
Цикл do while	**do {…} while (…)**		
Программные операторы поведения			
Альтернативное поведение	**alt {…}**		
Повторить	**repeat**		Должен использоваться в рамках шагов альтернативного поведения и альтернативных тестовых шагов.
Поведение перемежения	**interleave {…}**		
Включить состояние по умолчанию	**activate**		Оператор включения помещается в символ по умолчанию.

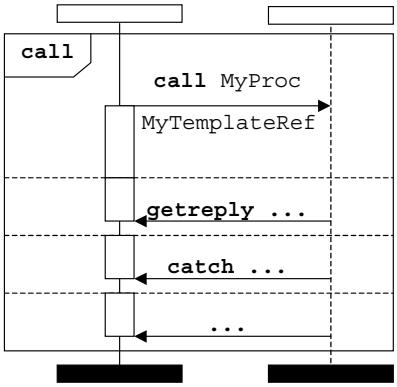
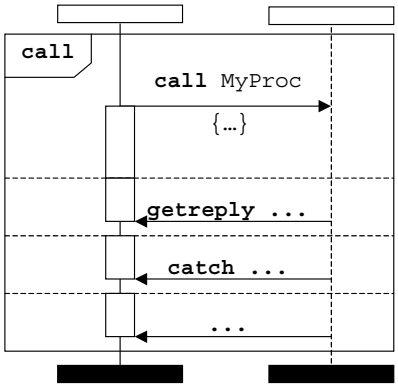
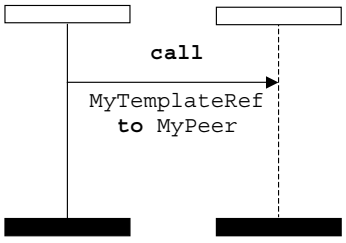
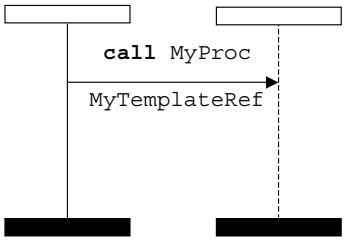
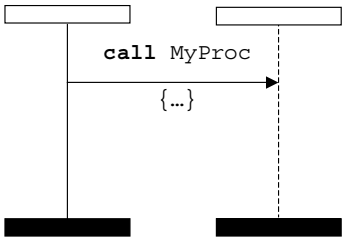
Элемент языка	Соответствующее ключевое слово	Символы GFT, если существуют, и типичное использование	Примечание
Выключить состояние по умолчанию	**deactivate**		Оператор выключения помещается в символ по умолчанию.
Управление возвращением	**return**		Дополнительная величина возврата присоединяется к символу возврата.
Операции конфигурации			
Создать параллельный тестовый компонент	**create**		Оператор `create` помещается в символе создания тестового компонента.
Соединить компонент с компонентом	**connect**		Оператор `connect` помещается в блоке действия.
Разъединить два компонента	**disconnect**		Оператор `disconnect` помещается в блоке действия.
Отобразить порт на интерфейс системы теста	**map**		Оператор `map` помещается в блоке действия.
Прекратить отображение порта на интерфейс системы теста	**unmap**		Оператор `unmap` помещается в блоке действия.

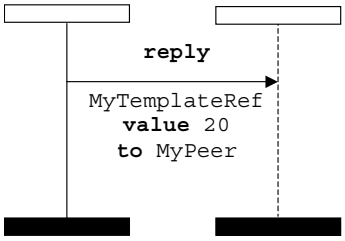
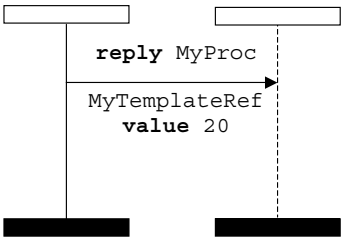
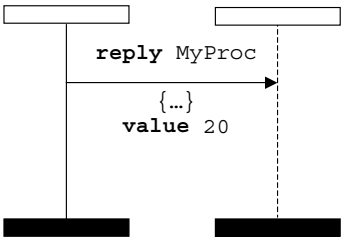
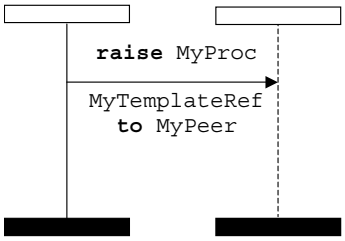
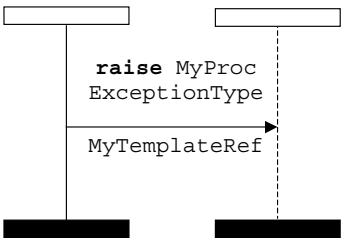
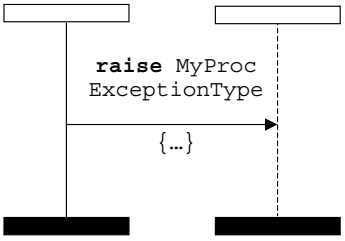
Элемент языка	Соответствующее ключевое слово	Символы GFT, если существуют, и типичное использование	Примечание
Получить адрес MTC	**mtc**		Специальный символ GFT, используемый в операторах, выражениях или в качестве идентификатора тестового компонента, отсутствует.
Получить адрес интерфейса системы теста	**system**		Специальный символ GFT, используемый в операторах или выражениях, отсутствует.
Получить собственный адрес	**self**		Специальный символ GFT, используемый в операторах, выражениях или в качестве идентификатора тестового компонента, отсутствует.
Запустить выполнение тестового компонента	**start**		Оператор **start** помещается в символ запуска .
Остановка тестовым компонентом выполнения своей операции	**stop**		Завершение mtc завершает также все другие тестовые компоненты. Варианты порта не могут быть остановлены.
Остановка выполнения другого тестового компонента			Идентификатор компонента помещается рядом с символом остановки.
Проверка завершения PTC	**running**		Специальный символ GFT, используемый в выражениях, отсутствует.
Ожидать завершения PTC	**done**		Оператор done помещается в символ условия.
Операции связи			
Направить сообщение	**send**		Операция “направить сообщение” определяется ссылкой на шаблон, но без информации о типе. Приемник однозначно определяется указателем (дополнительным) **to**.

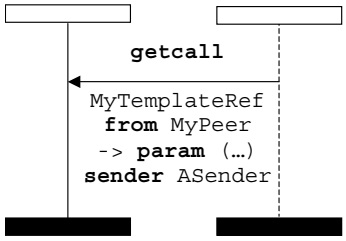
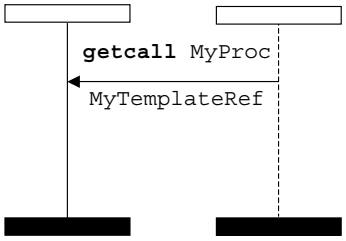
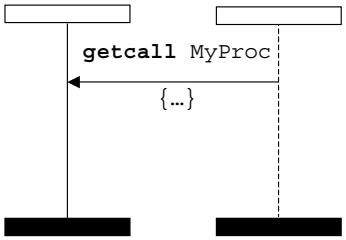
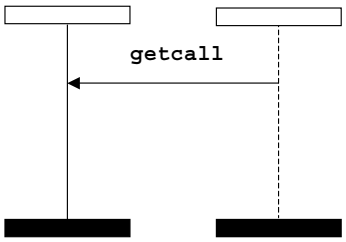
Элемент языка	Соответствующее ключевое слово	Символы GFT, если существуют, и типичное использование	Примечание
			Операция “направить сообщение” определяется ссылкой на шаблон с информацией о типе. Указатель (дополнительный) **to** может иметь место для однозначного определения одноранговой объекта.
			Операция “направить сообщение” задается определением встроенного шаблона. Указатель (дополнительный) **to** может иметь место для однозначного определения одноранговой объекта.
Принять сообщение	**receive**		Операция “принять сообщение” со значением определяется ссылкой на шаблон, но без информации о типе. Указатель (дополнительный) **from** означает, что отправитель сообщения должен определяться переменной `MyPeer`. Указатель (дополнительный) **value** присваивает принимаемое сообщение переменной `MyVar`. Указатель (дополнительный) **sender** загружает идентификатор отправителя и хранит его в переменной `ASender`.
			Операция “принять сообщение” со значением определяется ссылкой на шаблон с информацией о типе. Дополнительные указатели **from**-, **value**-и **sender** могут иметь место для определения отправителя сообщения, присвоения сообщения переменной или загрузки идентификатора одноранговой объекта.
			Операция “принять сообщение” со значением задается определением встроенного шаблона. Дополнительные указатели **from**-, **value**-и **sender** могут иметь место для определения отправителя сообщения, присвоения сообщения переменной или загрузки идентификатора одноранговой объекта.

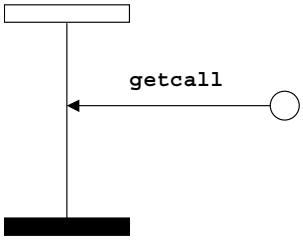
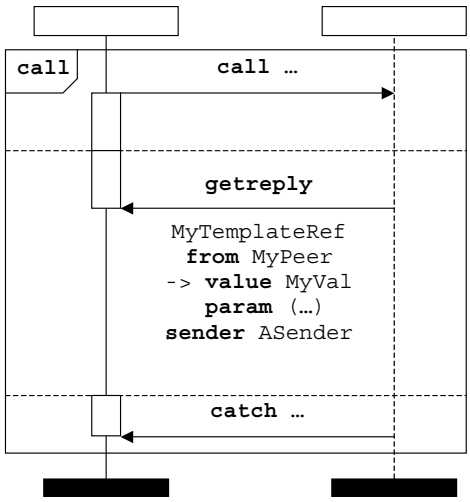
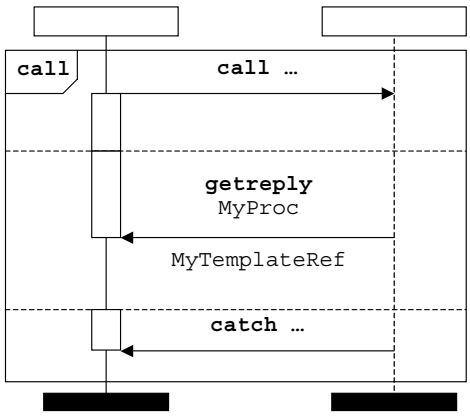
Элемент языка	Соответствующее ключевое слово	Символы GFT, если существуют, и типичное использование	Примечание
			Операция "принять любое сообщение" (значение и тип не указаны). Дополнительные указатели **from**-, **value**-и **sender** могут иметь место для определения отправителя сообщения, присвоения сообщения переменной или загрузки идентификатора однорангового объекта.
			Операция "принять любое сообщение" (значение и тип не указаны) с любого порта. Значение сообщения, которое должно быть принято с любого порта, может быть ограничено посредством ссылки на шаблоны или путем использования встроенных шаблонов. Дополнительные указатели **from**-, **value**-и **sender** могут иметь место для определения отправителя сообщения, присвоения сообщения переменной или загрузки идентификатора однорангового объекта.
Сообщение срабатывания	**trigger**		Операция срабатывания от сообщения со значением определяется ссылкой на шаблон, но без информации о типе. Указатель (дополнительный) **from** означает, что отправитель сообщения должен определяться переменной `MyPeer`. Указатель (дополнительный) **value** присваивает принимаемое сообщение переменной `MyVar`. Указатель (дополнительный) **sender** загружает идентификатор отправителя и хранит его в переменной `ASender`.
			Операция срабатывания от сообщения со значением определяется ссылкой на шаблон, и с информацией о типе. Дополнительные указатели **from**-, **value**-и **sender** могут иметь место для определения отправителя сообщения, присвоения сообщения переменной или загрузки идентификатора однорангового объекта.

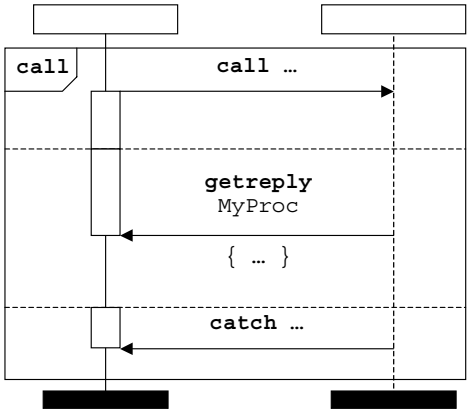
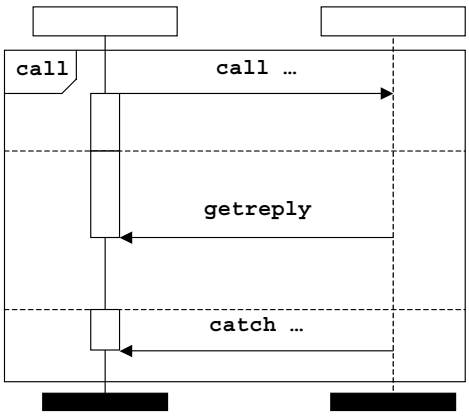
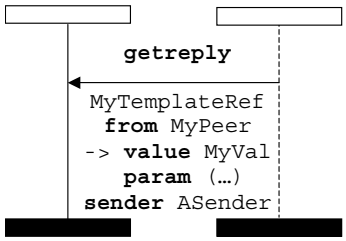
Элемент языка	Соответствующее ключевое слово	Символы GFT, если существуют, и типичное использование	Примечание
			Операция срабатывания от сообщения со значением задается определением встроенного шаблона. Дополнительные указатели **from**-, **value**-и **sender** могут иметь место для определения отправителя сообщения, присвоения сообщения переменной или загрузки идентификатора однорангового объекта.
			Срабатывание от любого сообщения (значение и тип не указаны). Дополнительные указатели **from**-, **value**-и **sender** могут иметь место для определения отправителя сообщения, присвоения сообщения переменной (типа **anytype**) и загрузки идентификатора однорангового объекта.
			Срабатывание от любого сообщения (значение и тип не указаны) с любого порта. Значение сообщения, которое должно вызвать срабатывание с любого порта, может быть ограничено посредством ссылки на шаблоны или путем использования встроенных шаблонов. Дополнительные указатели **from**-, **value**-и **sender** могут иметь место для определения отправителя сообщения, присвоения сообщения переменной (типа **anytype**) и загрузки идентификатора однорангового объекта.
Инициировать вызов блокирующей процедуры	**call**		Вызов блокирующей процедуры путем использования шаблона подписи. Приемник однозначно определяется указателем (дополнительным) **to**. Тело вызова, т. е. возможные операции **getreply** и **catch**, показано только схематически.

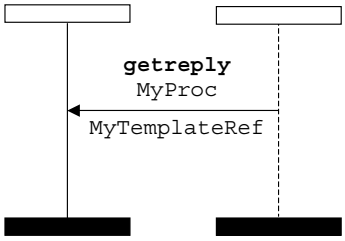
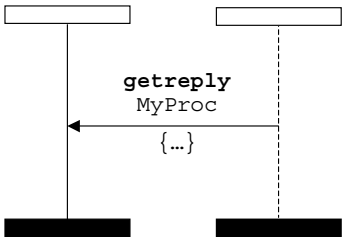
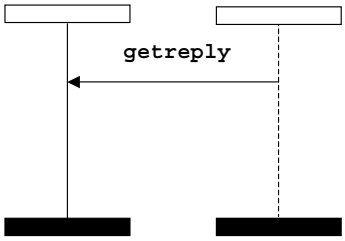
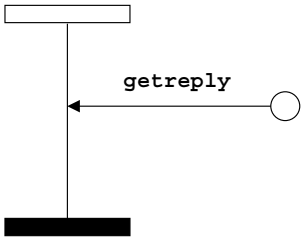
Элемент языка	Соответствующее ключевое слово	Символы GFT, если существуют, и типичное использование	Примечание
			Вызов блокирующей процедуры путем использования шаблона подписи и информации о подписи. Указатель (дополнительный) **to** может иметь место для однозначного определения одноранговой объекта. Тело вызова, т. е. возможные операции **getreply** и **catch**, показано только схематически.
			Вызов блокирующей процедуры путем использования встроенного шаблона. Указатель (дополнительный) **to** может иметь место для однозначного определения одноранговой объекта. Тело вызова, т. е. возможные операции **getreply** и **catch**, показано только схематически.
Инициировать вызов неблокирующей процедуры	`call`		Вызов удаленной процедуры. Вызов определяется ссылкой на шаблон, но без информации о подписи. Приемник однозначно определяется указателем (дополнительным) **to**.
			Вызов удаленной процедуры `MyProc`. Вызов определяется ссылкой на шаблон. Указатель (дополнительный) **to** может иметь место для однозначного определения одноранговой объекта.
			Вызов удаленной процедуры `MyProc`. Вызов определяется встроенным шаблоном. Указатель (дополнительный) **to** может иметь место для однозначного определения одноранговой объекта.

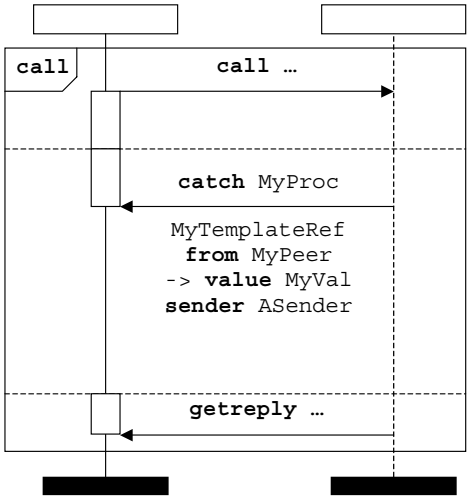
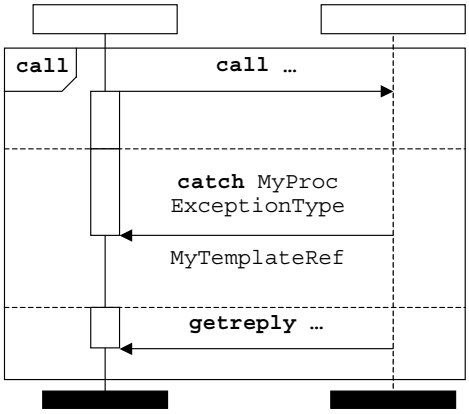
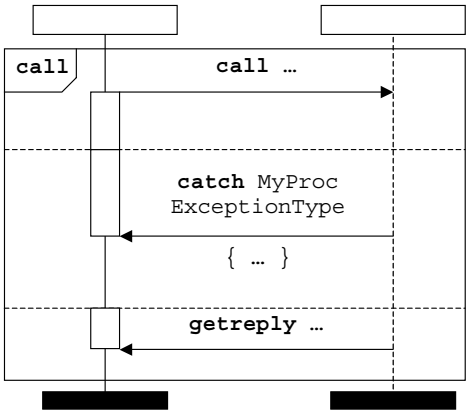
Элемент языка	Соответствующее ключевое слово	Символы GFT, если существуют, и типичное использование	Примечание
Ответить на вызов процедуры из удаленного объекта	**reply**		Ответить на вызов удаленной процедуры. Ответ определяется ссылкой на шаблон и возможным значением возврата (указатель **value**). ПРИМЕЧАНИЕ. – Информация о подписи является частью определения шаблона. Приемник однозначно определяется указателем (дополнительным) **to**.
			Ответить на вызов удаленной процедуры, относящейся к MyProc. Ответ определяется ссылкой на шаблон и возможным значением возврата (указатель **value**). Указатель (дополнительный) **to** может иметь место для однозначного определения одноранговой объекта.
			Ответить на вызов удаленной процедуры, относящейся к MyProc. Ответ определяется встроенным шаблоном и возможным значением возврата (указатель **value**). Указатель (дополнительный) **to** может иметь место для однозначного определения одноранговой объекта.
Установить исключение (для принятого вызова)	**raise**		Установить исключение для принятого вызова, относящегося к MyProc. Исключение определяется ссылкой на шаблон. ПРИМЕЧАНИЕ. – Тип исключения задается в определении шаблона. Приемник однозначно определяется указателем (дополнительным) **to**.
			Установить исключение для принятого вызова, относящегося к MyProc. Исключение определяется его типом (дополнительно) и ссылкой на шаблон. Указатель (дополнительный) **to** может иметь место для однозначного определения одноранговой объекта.
			Установить исключение для принятого вызова, относящегося к MyProc. Исключение определяется его типом и встроенным шаблоном. Указатель (дополнительный) **to** может иметь место для однозначного определения одноранговой объекта.

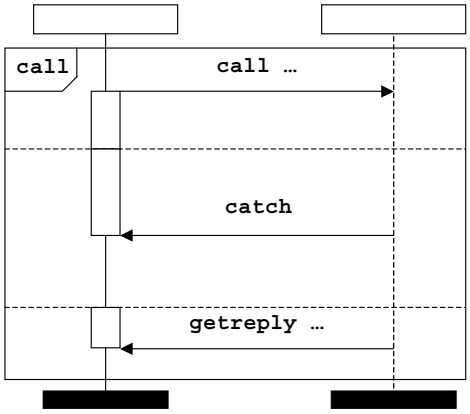
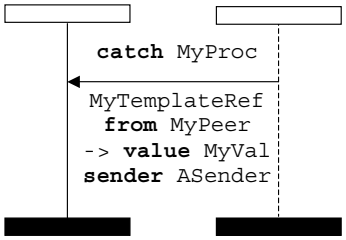
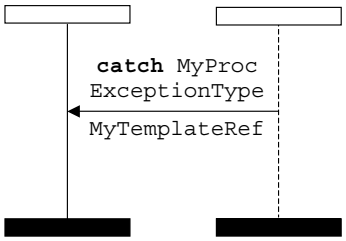
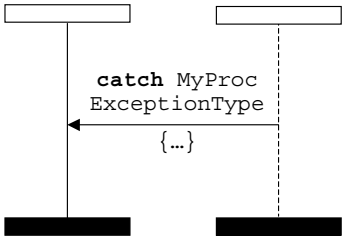
Элемент языка	Соответствующее ключевое слово	Символы GFT, если существуют, и типичное использование	Примечание
Принять вызов процедуры из удаленного объекта	**getcall**		Принять вызов процедуры из удаленного объекта. Подпись вызова должна соответствовать условиям, определяемым ссылкой на шаблон. ПРИМЕЧАНИЕ. – Информация о подписи является частью определения шаблона. Указатель (дополнительный) **from** означает, что отправитель сообщения должен определяться переменной MyPeer. Указатель (дополнительный) **param** присваивает переменным значения входных параметров. Указатель (дополнительный) **sender** загружает идентификатор отправителя и хранит его в переменной ASender.
			Принять вызов процедуры из удаленного объекта. Подпись вызова должна соответствовать условиям, определяемым ссылкой на подпись и ссылкой на шаблон. Дополнительные указатели **from**, **param** и **sender** могут иметь место для определения отправителя сообщения, присвоения переменным входных параметров или загрузки идентификатора однорангового объекта.
			Принять вызов процедуры из удаленного объекта. Подпись вызова должна соответствовать условиям, определяемым ссылкой на подпись и определением встроенного шаблона. Дополнительные указатели **from**, **param** и **sender** могут иметь место для определения отправителя сообщения, присвоения входных параметров переменным или загрузки идентификатора однорангового объекта.
			Принять любой вызов процедуры из любого удаленного объекта. Дополнительные указатели **from** и **sender** могут иметь место для определения отправителя сообщения или загрузки идентификатора однорангового объекта.

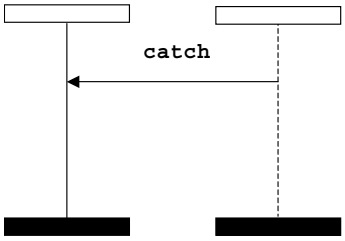
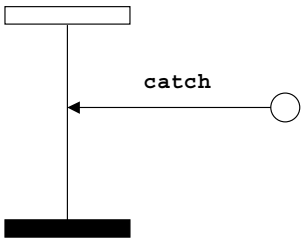
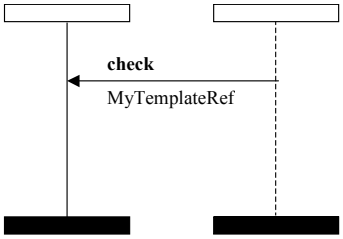
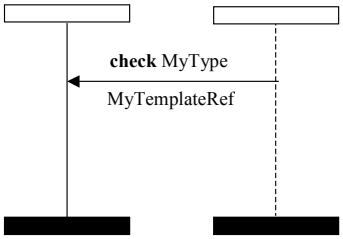
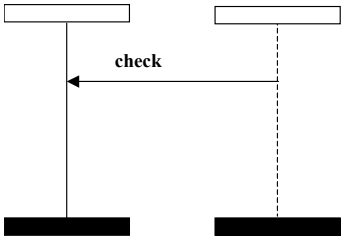
Элемент языка	Соответствующее ключевое слово	Символы GFT, если существуют, и типичное использование	Примечание
			Принять любой вызов процедуры из любого удаленного объекта на любом порте. Вызов, который должен быть принят с любого порта, может быть ограничен посредством ссылки на шаблоны или путем использования встроенных шаблонов. Дополнительные указатели **from**, **param** и **sender** могут иметь место для определения отправителя сообщения, присвоения входных параметров переменным или загрузки идентификатора одноранговой объекта.
Обработка ответа от предыдущего блокирующего вызова	**getreply**		Принять ответ от блокирующего вызова. Ответ должен соответствовать условиям, определенным ссылкой на шаблон. ПРИМЕЧАНИЕ. – Информация о подписи является частью определения шаблона. Указатель (дополнительный) **from** означает, что отправитель сообщения должен определяться переменной `MyPeer`. Указатель (дополнительный) **value** присваивает переменной `MyVal` возможное значение возврата, относящееся к процедуре. Указатель (дополнительный) **param** присваивает переменным значения выходных параметров. Указатель (дополнительный) **sender** загружает идентификатор отправителя и хранит его в переменной `ASender`.
			Принять ответ от блокирующего вызова. Ответ должен соответствовать условиям, определяемым ссылкой на подпись и ссылкой на шаблон. Дополнительные указатели **from**, **value**, **param** и **sender** могут иметь место для определения отправителя ответа, загрузки значения возврата процедуры, присвоения переменным входных параметров или загрузки идентификатора одноранговой объекта.

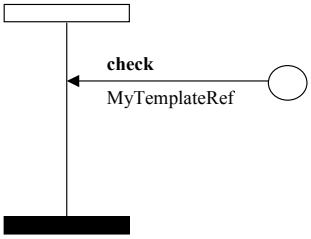
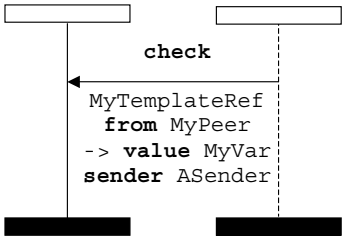
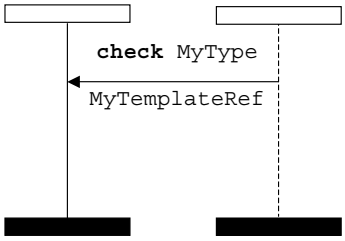
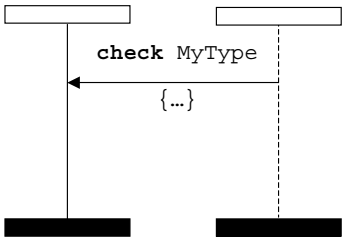
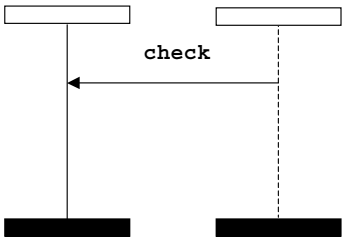
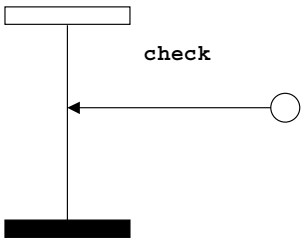
Элемент языка	Соответствующее ключевое слово	Символы GFT, если существуют, и типичное использование	Примечание
			Принять ответ от блокирующего вызова. Ответ должен соответствовать условиям, задаваемым ссылкой на подпись и определением встроенного шаблона. Дополнительные указатели **from**, **value**, **param** и **sender** могут иметь место для определения отправителя ответа, загрузки значения возврата процедуры, присвоения переменным входных параметров или загрузки идентификатора одноранговой объекта.
			Принять любой ответ от того или иного блокирующего вызова.
Обработать ответ от предыдущего неблокирующего вызова или ответ, независимый от вызова	**getreply**		Принять ответ от предыдущего вызова. Ответ должен соответствовать условиям, определяемым ссылкой на шаблон. ПРИМЕЧАНИЕ. – Информация о подписи является частью определения шаблона. Указатель (дополнительный) **from** означает, что отправитель сообщения должен определяться переменной `MyPeer`. Указатель (дополнительный) **value** присваивает переменной `MyVal` возможное значение возврата, относящееся к процедуре. Указатель (дополнительный) **param** присваивает переменным значения выходных параметров. Указатель (дополнительный) **sender** загружает идентификатор отправителя и хранит его в переменной `ASender`.

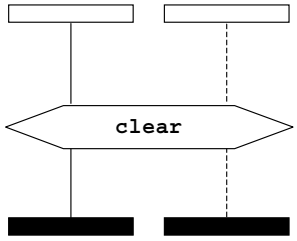
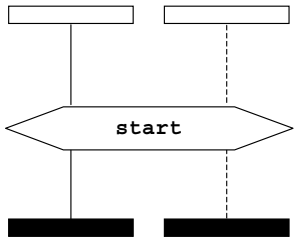
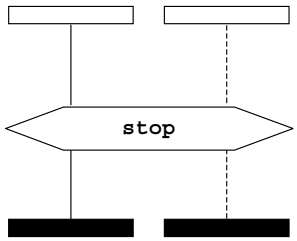
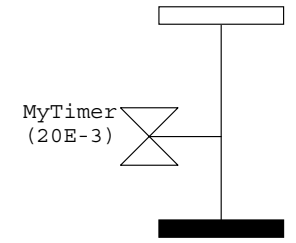
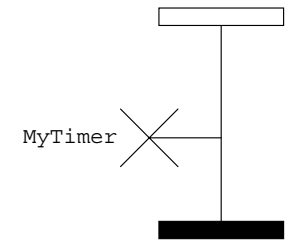
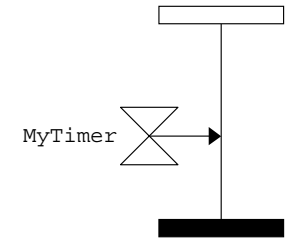
Элемент языка	Соответствующее ключевое слово	Символы GFT, если существуют, и типичное использование	Примечание
			Принять ответ от предыдущего вызова. Ответ должен соответствовать условиям, определяемым ссылкой на подпись и ссылкой на шаблон. Дополнительные указатели **from**, **value**, **param** и **sender** могут иметь место для определения отправителя ответа, загрузки значения возврата процедуры, присвоения переменным входных параметров или загрузки идентификатора одноранговой объекта.
			Принять ответ от предыдущего вызова. Ответ должен соответствовать условиям, определяемым ссылкой на подпись и ссылкой на шаблон. Дополнительные указатели **from**, **value**, **param** и **sender** могут иметь место для определения отправителя ответа, загрузки значения возврата процедуры, присвоения переменным входных параметров или загрузки идентификатора одноранговой объекта.
			Принять любой ответ от любого предыдущего вызова. Дополнительные указатели **from**- и **sender** могут иметь место для определения отправителя ответа или загрузки идентификатора одноранговой объекта.
			Принять любой ответ от любого предыдущего вызова на любом порте. Ответ, который должен быть принят с любого порта, может быть ограничен посредством ссылки на шаблоны или путем использования встроенных шаблонов. Дополнительные указатели **from**, **value**, **param** и **sender** могут иметь место для определения отправителя ответа, загрузки значения возврата процедуры, присвоения переменным входных параметров или загрузки идентификатора одноранговой объекта.

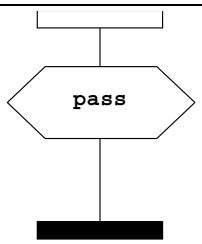
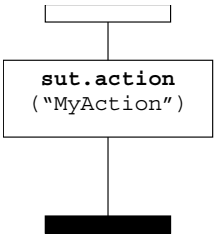
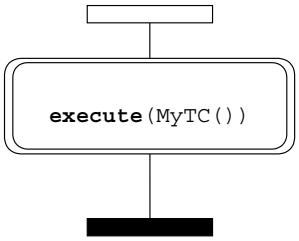
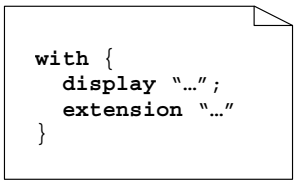
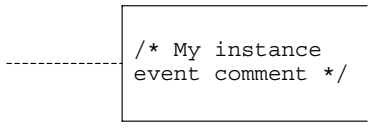
Элемент языка	Соответствующее ключевое слово	Символы GFT, если существуют, и типичное использование	Примечание
Обнаружить исключение, исходящее из предыдущего блокирующего вызова	**catch**		Обнаружить исключение, исходящее из предыдущего вызова. Исключение должно соответствовать условиям, определенным ссылкой на шаблон. ПРИМЕЧАНИЕ. – Информация о подписи является частью определения шаблона. Указатель (дополнительный) **from** означает, что отправитель сообщения должен определяться переменной `MyPeer`. Указатель (дополнительный) **value** присваивает переменной `MyVal` значение исключения. Указатель (дополнительный) **sender** загружает идентификатор отправителя и хранит его в переменной `ASender`.
			Обнаружить исключение, исходящее из предыдущего вызова. Исключение должно соответствовать условиям, определенным типом исключения и ссылкой на шаблон. Дополнительные указатели **from**, **value** и **sender** могут иметь место для определения отправителя исключения, загрузки значения исключения или загрузки идентификатора одноранговой объекта.
			Обнаружить исключение, исходящее из предыдущего вызова. Исключение должно соответствовать условиям, заданным типом исключения и определением встроенного шаблона. Дополнительные указатели **from**, **value** и **sender** могут иметь место для определения отправителя исключения, загрузки значения исключения или загрузки идентификатора одноранговой объекта.

Элемент языка	Соответствующее ключевое слово	Символы GFT, если существуют, и типичное использование	Примечание
			Принять исключение, исходящее из блокирующего вызова. Дополнительные указатели **from**, **value** и **sender** могут иметь место для определения отправителя исключения, загрузки значения исключения или загрузки идентификатора одноранговой объекта.
Обнаружить исключение, исходящее из предыдущего неблокирующего вызова, или исключение, независимое от вызова	**catch**		Принять исключение, исходящее из предыдущего вызова. Исключение должно соответствовать условиям, определяемым ссылкой на шаблон. ПРИМЕЧАНИЕ. – Информация о типе является частью определения шаблона. Указатель (дополнительный) **from** означает, что отправитель исключения должен определяться переменной `MyPeer`. Указатель (дополнительный) **value** присваивает переменной `MyVal` значение исключения. Указатель (дополнительный) **sender** загружает идентификатор отправителя и хранит его в переменной `ASender`.
			Обнаружить исключение, исходящее из предыдущего вызова. Исключение должно соответствовать условиям, определенным типом исключения и ссылкой на шаблон. Дополнительные указатели **from**, **value** и **sender** могут иметь место для определения отправителя исключения, загрузки значения исключения или загрузки идентификатора одноранговой объекта.
			Обнаружить исключение, исходящее из предыдущего вызова. Исключение должно соответствовать условиям, заданным типом исключения и определением встроенного шаблона. Дополнительные указатели **from**, **value** и **sender** могут иметь место для определения отправителя исключения, загрузки значения исключения или загрузки идентификатора одноранговой объекта.

Элемент языка	Соответствующее ключевое слово	Символы GFT, если существуют, и типичное использование	Примечание
			Обнаружить любое исключение, исходящее из любого предыдущего вызова. Дополнительные указатели **from**, **value** и **sender** могут иметь место для определения отправителя исключения, загрузки значения исключения (и его присвоения переменной типа **anytype**) или загрузки идентификатора однорангового объекта.
			Обнаружить любое исключение из любого предыдущего вызова на любом порте. Исключение, которое должно быть принято с любого порта, может быть ограничено посредством ссылки на шаблоны или путем использования встроенных шаблонов. Дополнительные указатели **from**, **value** и **sender** могут иметь место для определения отправителя исключения, загрузки значения исключения или загрузки идентификатора однорангового объекта.
Проверить принимаемое (текущее) сообщение/вызов	**check**		с шаблоном, без типа
			с шаблоном, с типом
			без шаблона, без типа (любое сообщение с данного порта) с шаблоном, без типа, без порта (данное сообщение с данного порта)

Элемент языка	Соответствующее ключевое слово	Символы GFT, если существуют, и типичное использование	Примечание
		 Может быть также использовано в сочетании с getcall, getreply и catch	
Проверить текущие сообщение, вызов, ответ или исключение	**check**		Проверить, было ли получено то или иное сообщение со значением, определяемым ссылкой на шаблон . Синтаксис следует синтаксису для приема сообщений. ПРИМЕЧАНИЕ. – Проверка может также использоваться в сочетании с **getcall**, **getreply** и **catch**.
			Проверить, было ли получено то или иное сообщение со значением, определяемым ссылкой на шаблон. Синтаксис соответствует синтаксису для приема сообщений. ПРИМЕЧАНИЕ. – Проверка может также использоваться в сочетании с **getcall**, **getreply** и **catch**.
			Проверить, было ли получено то или иное сообщение со значением, задаваемым определением встроенного шаблона. Синтаксис соответствует синтаксису для приема сообщений. ПРИМЕЧАНИЕ. – Проверка может также использоваться в сочетании с **getcall**, **getreply** и **catch**.
			Проверить, было ли получено какое-либо сообщение (значение и тип не указываются). Синтаксис соответствует синтаксису для приема сообщений. ПРИМЕЧАНИЕ. – Проверка может также использоваться в сочетании с **getcall**, **getreply** и **catch**.
			Проверить, было ли получено какое-либо сообщение на каком-либо порте (значение и тип не указываются). Синтаксис соответствует синтаксису для приема сообщений. ПРИМЕЧАНИЕ. – Проверка может также использоваться в сочетании с **getcall**, **getreply** и **catch**.

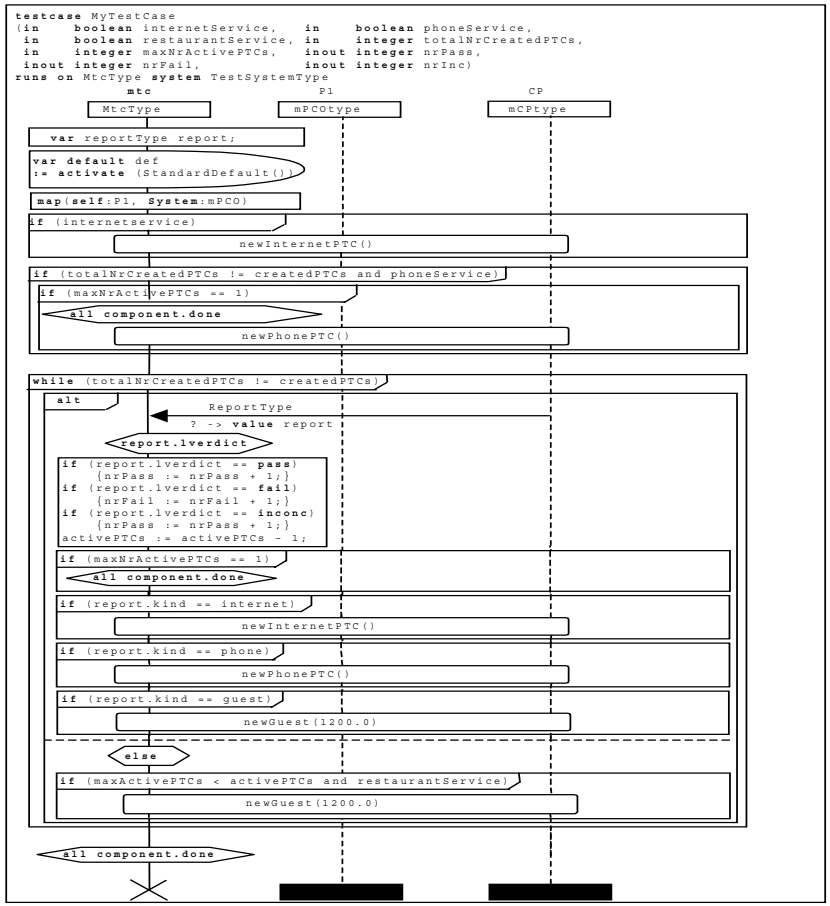
Элемент языка	Соответствующее ключевое слово	Символы GFT, если существуют, и типичное использование	Примечание
Очистить порт	**clear**		Оператор очистки порта помещается в символ условия. Условие должно охватывать только вариант порта, подлежащий очистке.
Очистить порт и предоставить доступ к нему	**start**		Оператор запуска порта помещается в символе условия. Условие должно охватывать только вариант порта, подлежащий запуску.
Остановить доступ (на прием и передачу) на порте	**stop**		Оператор остановки порта помещается в символе условия. Условие должно охватывать только вариант порта, подлежащий остановке.
Операции таймера			
Таймер запуска	**start**		
Таймер остановки	**stop**		
Считать истекшее время	**read**		Специальный символ GFT, используемый в операторах и выражениях, отсутствует.
Проверить, работает ли таймер	**running**		Специальный символ GFT, используемый в операторах и выражениях, отсутствует.
Операция тайм-аута	**timeout**		

Элемент языка	Соответствующее ключевое слово	Символы GFT, если существуют, и типичное использование	Примечание
Установить местное заключение	**verdict.set**		Заключение помещается в символ условия.
Получить местное заключение	**verdict.get**		Специальный символ GFT, используемый в операторах и выражениях, отсутствует.
Операции SUT			
Отдаленное действие, осуществляемое SUT	**sut.action**		Оператор действия помещается в блок действия.
Выполнение тестовых случаев			
Выполнить тестовый случай	**execute**		Оператор **execute** помещается в символ выполнения тестового случая.
Аттрибуты			
Определение атрибутов для управления, тестовых случаев, тестовых шагов и функций	**with**		Оператор **with** помещается в символ текста.
Комментарии			
Комментарии в тексте		/* Мой комментарий в несколько строк */ // Мой комментарий в одну строку	Может использоваться повсюду, где может размещаться текст.
Комментарии в отношении событий вариантов			Должны быть присоединены к событиям на варианте управления, тестового компонента или порта
Комментарии в отношении диаграмм управления, тестового случая, функции или тестового шага			Должны быть присоединены к событиям на варианте управления, тестового компонента или порта

Приложение C

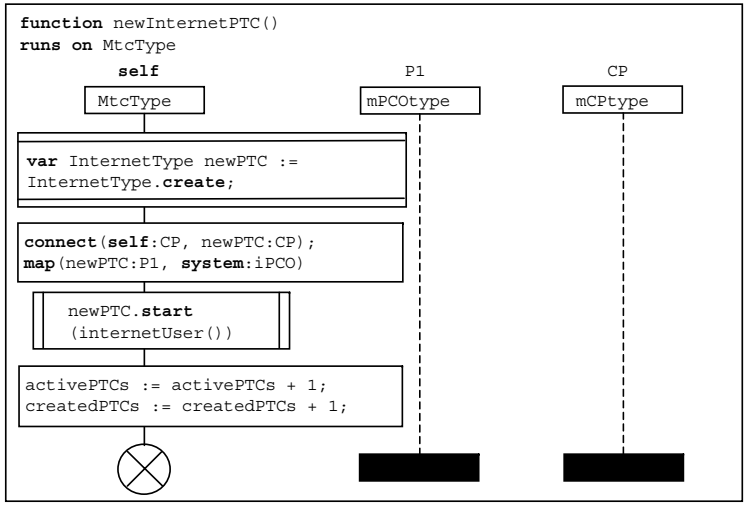
Примеры

C.1 Пример ресторана

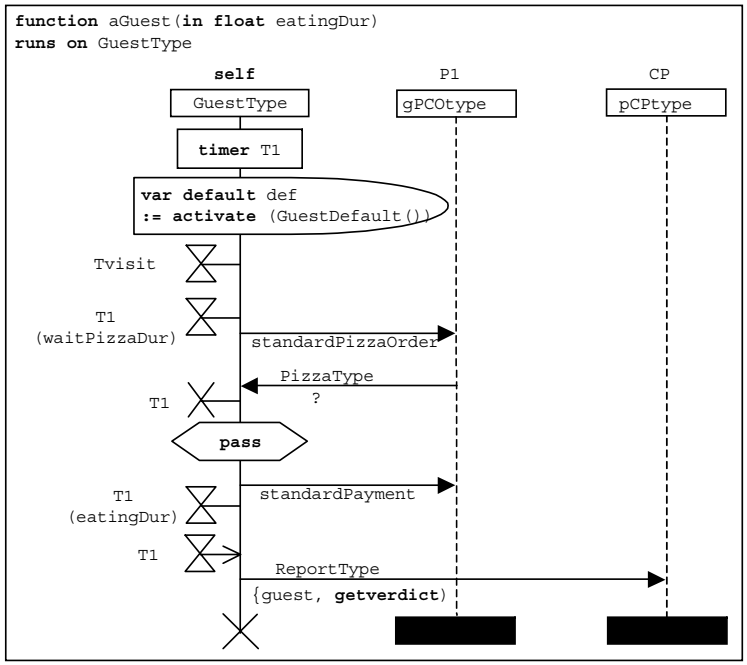



```
testcase MyTestCase (
    in boolean          internetService,     // SERVICES
    in boolean          phoneService,
    in boolean          restaurantService,
    in integer          totalNrCreatedPTCs,  // TERMINATION
    in integer          maxNrActivePTCs,     // CONTROL
    inout integer       nrPass,              // RETURN
    inout integer       nrFail,
    inout integer       nrInc
    )
  runs on MtcType
  system TestSystemType
{
    var ReportType report;

    var default def := activate (StandardDefault());

    map(self:P1, system:mPCO);

    if (internetService)  {
        newInternetPTC();
    }

    if (totalNrCreatedPTCs != createdPTCs
        and phoneService) {

        if (maxNrActivePTCs == 1) {
            all component.done;
        }
        newPhonePTC();
    }

    while ( totalNrCreatedPTCs != createdPTCs ) {

        alt {
            []  CP.receive(ReportType:?) -> value report {

                setverdict(report.lverdict);

                if (report.lverdict == pass) { nrPass := nrPass + 1; }
                if (report.lverdict == fail) { nrFail := nrFail + 1; }
                if (report.lverdict == inconc) { nrInc := nrInc + 1; }
                activePTCs := activePTCs - 1;

                if (maxNrActivePTCs == 1) {
                    all component.done;
                }
                if (report.kind == internet) {
                    newInternetPTC();
                }
                if (report.kind == phone) {
                    newPhonePTC();
                }
                if (report.kind == guest) {
                    newGuest(1200.0);
                }
            }


            [else] {

                if (maxNrActivePTCs < activePTCs
                    and restaurantService) {
                      newGuest(1200.0);
                    }
            }
        }
    }

    all component.done;
    stop;
  }
```

Рисунок C.1/Z.142 – Пример ресторана – тестовый случай MyTestCase

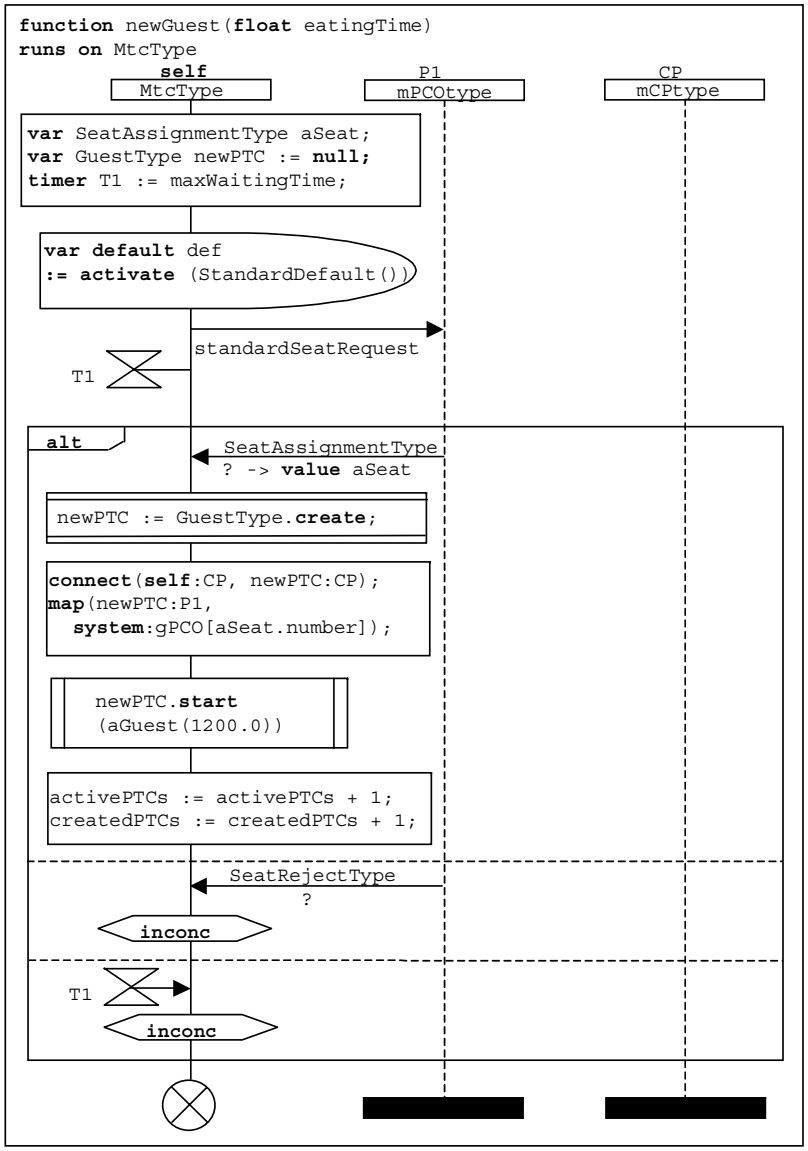



```
function newInternetPTC ()
   runs on MtcType {

   var InternetType newPTC := InternetType.create;

   connect(self:CP, newPTC:CP);
   map(newPTC:P1, system:iPCO);

   newPTC.start(internetUser());

   activePTCs  := activePTCs + 1;
   createdPTCs := createdPTCs + 1;

   return;

}
```



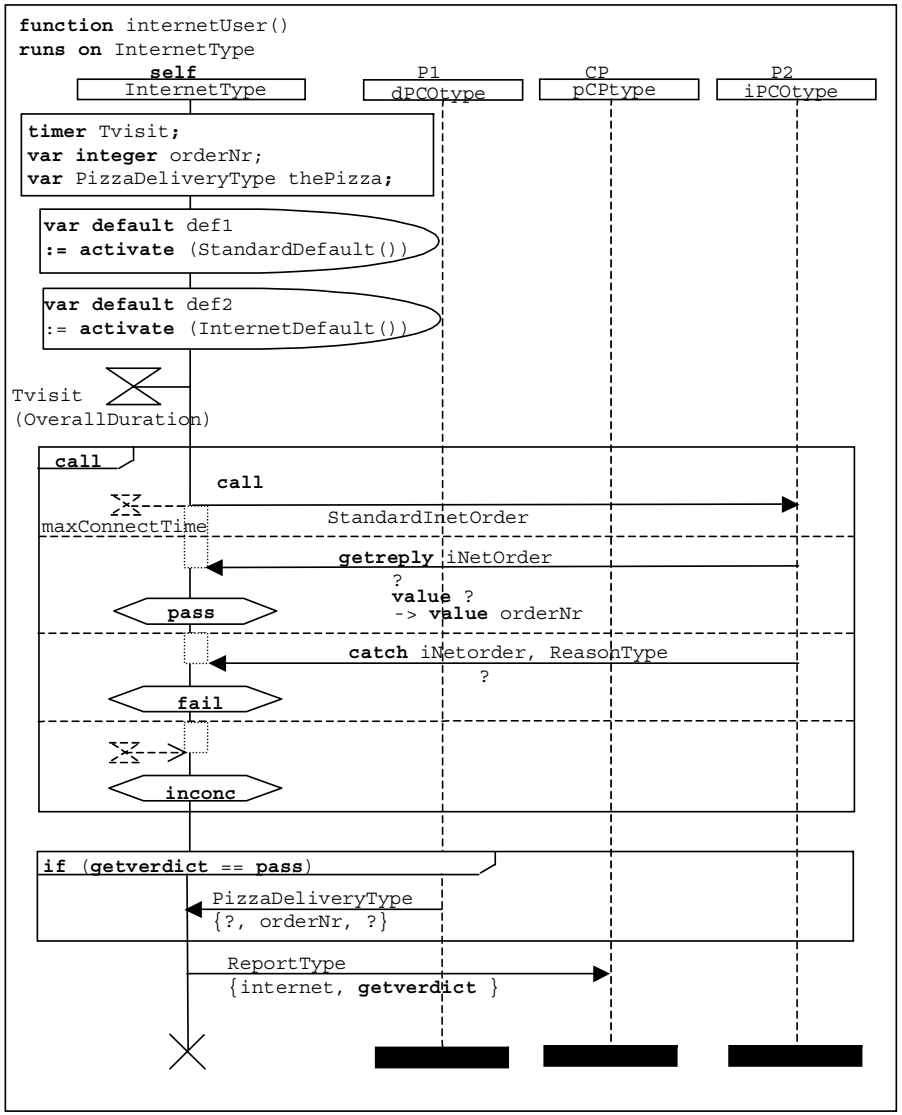

```
function aGuest (in float eatingDur) runs on GuestType {

   timer T1;

   var default def := activate(GuestDefault());
   Tvisit.start; // component timer
   T1.start(waitPizzaDur);
   P1.send(standardPizzaOrder);
   P1.receive(PizzaType : ?);
   T1.stop;
   setverdict(pass);
   P1.send(standardPayment);
   T1.start(eatingDur);  // eating
   T1.timeout;
   CP.send(ReportType : {guest, getverdict});
   stop;
  } // end function aGuest
```

Рисунок C.2/Z.142 – Пример ресторана – функции newInternetPTC (новый интернет-PTC) и aGuest (гость)



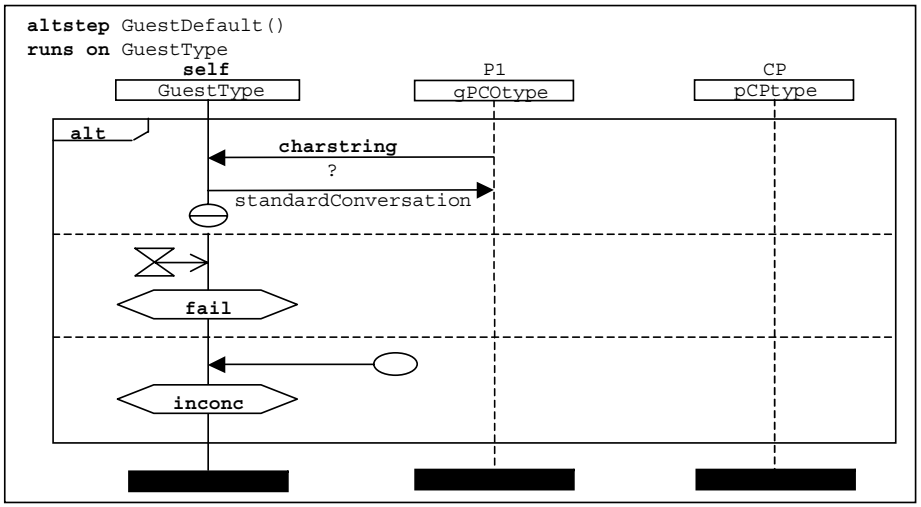

```
function newGuest (float eatingTime) runs on MtcType {

   var SeatAssignmentType aSeat;
   var GuestType newPTC := null;
   timer T1 := maxWaitingTime;

   var default def := activate(StandardDefault());

   // Request for a seat
   P1.send(standardSeatRequest);
   T1.start;

   alt {
      [] P1.receive(SeatAssignmentType:?) -> value aSeat {
         newPTC := GuestType.create;

         connect(self:CP, newPTC:CP);
         map(newPTC:P1, system:gPCO[aSeat.number]);

         newPTC.start(aGuest(1200.0));

         activePTCs  := activePTCs+1; // Update MTC variables
         createdPTCs := createdPTCs+1;
      }

      [] P1.receive(SeatRejectType:?) { // No seat assigned
         setverdict(inconc);
      }

      [] T1.timeout { // No answer on seat request
         setverdict(inconc);
      }
    }
    return;
}
```

Рисунок C.3/Z.142 – Пример ресторана – функция newGuest (новый гость)

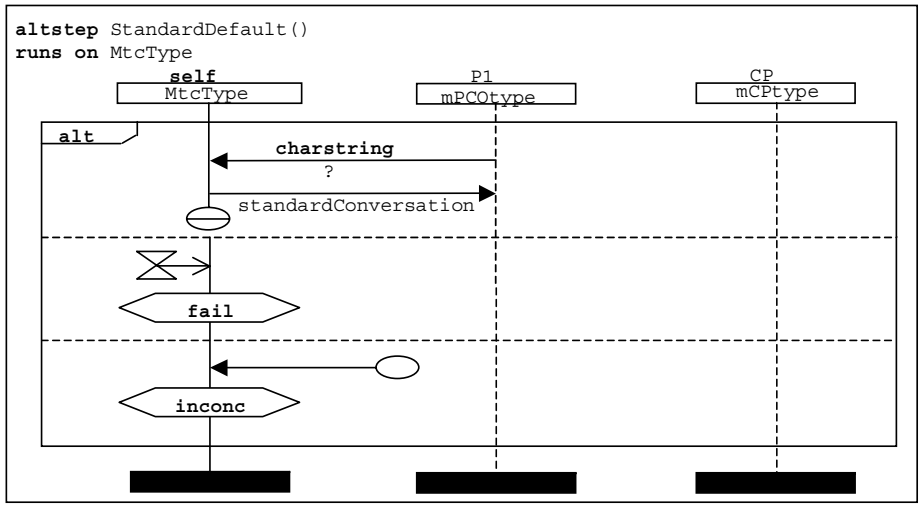



```
function internetUser () runs on InternetType {
  // ***
  // *** Purpose: Specifies the behaviour of an
  // *** internet guest
  // ***

  timer Tvisit;
  var integer orderNr;
  var PizzaDeliveryType thePizza;

  var default def1 := activate(StandardDefault());
  var default def2 := activate(InternetDefault());
  Tvisit.start(OverallDuration);

  P2.call(StandardINetOrder, maxConnectTime) {
     [] P2.getreply (iNetOrder:? value ?)
              -> value orderNr {
          setverdict(pass);
        }
     [] P2.catch (iNetOrder, ReasonType : ?) {

          setverdict(fail);
        }
     [] P2.catch (timeout) {
          setverdict(inconc);
        }
   };
   if (getverdict == pass) {
     P1.receive(PizzaDeliveryType
          : { ?, orderNr, ?});
   }
   CP.send(ReportType : {internet, getverdict});
   stop;
 }
```

Рисунок C.4/Z.142 – Пример ресторана – функция internetUser (пользователь интернета)

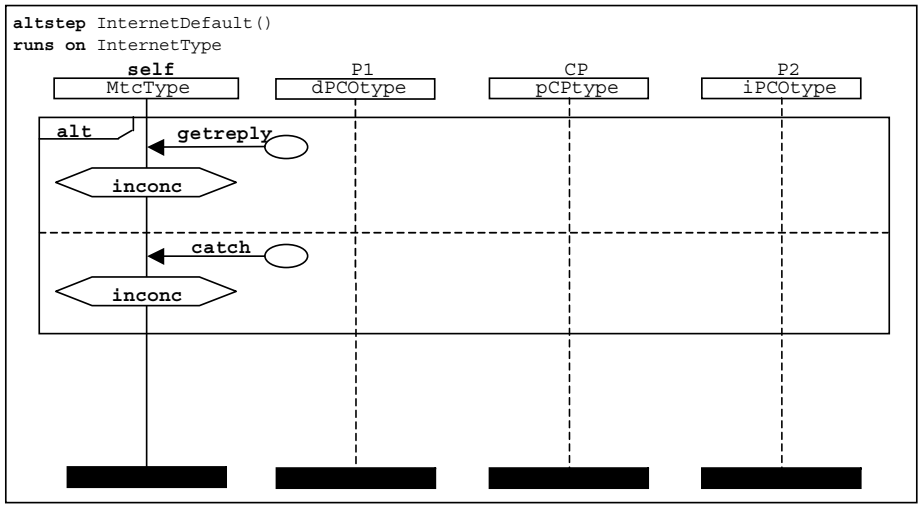



```
altstep GuestDefault() runs on GuestType {
   // ***
   // *** Purpose: Default behaviour for
   // *** message based ports
   // ***

   [] P1.receive(charstring : ?) {
      P1.send(standardConversation);
      repeat;
   }

   [] any timer.timeout {
      setverdict(fail);
   }

   [] any port.receive {
      setverdict(inconc);
   }
}
```

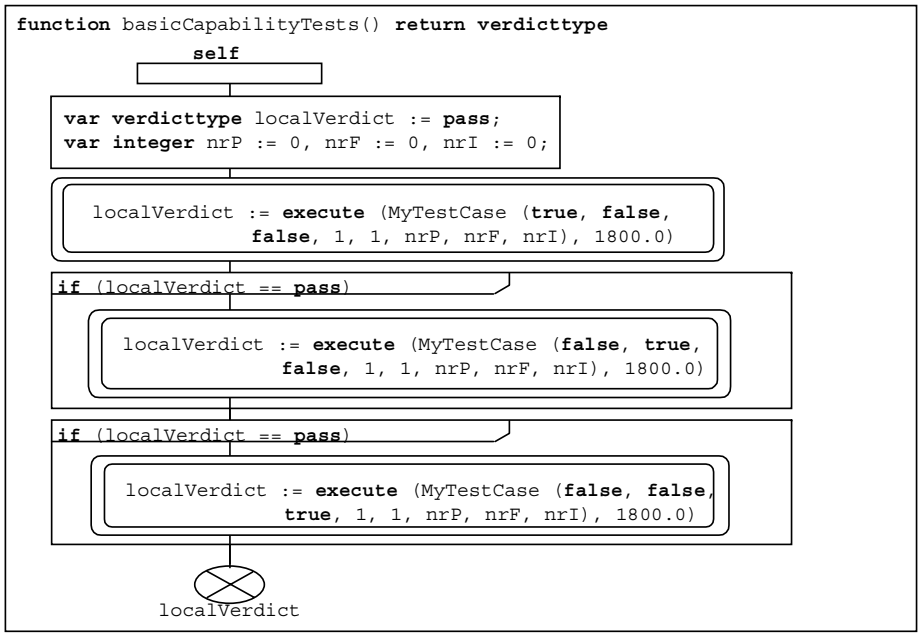



```
altstep StandardDefault() runs on MtcType {
   // ***
   // *** Purpose: Default behaviour for
   // **** message based ports
   // ***

   [] P1.receive(charstring : ?) {
      P1.send(standardConversation);
      repeat;
   }

   [] any timer.timeout {
      setverdict(fail);
   }

   [] any port.receive {
      setverdict(inconc);
   }
}
```

Рисунок C.5/Z.142 – Пример ресторана – функции GuestDefault (гость, по умолчанию) и StandardDefault (стандартный, по умолчанию)

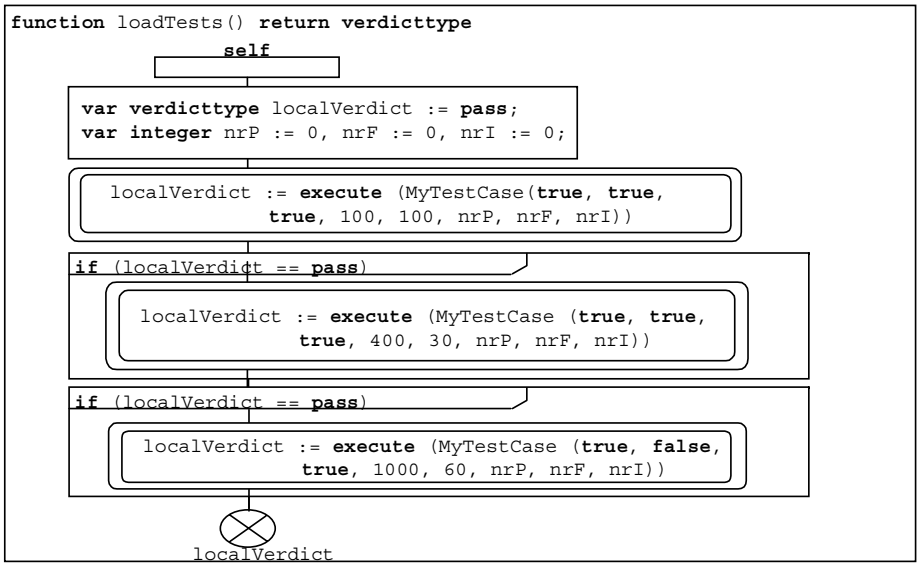



```
altstep InternetDefault()
   runs on InternetType {
   // ***
   // *** Purpose: Default behaviour for
   // **** the procedure based port
   // ***

   [] any port.getreply {
      setverdict(inconc);
   }

   [] any port.catch {
      setverdict(inconc);
   }
}
```

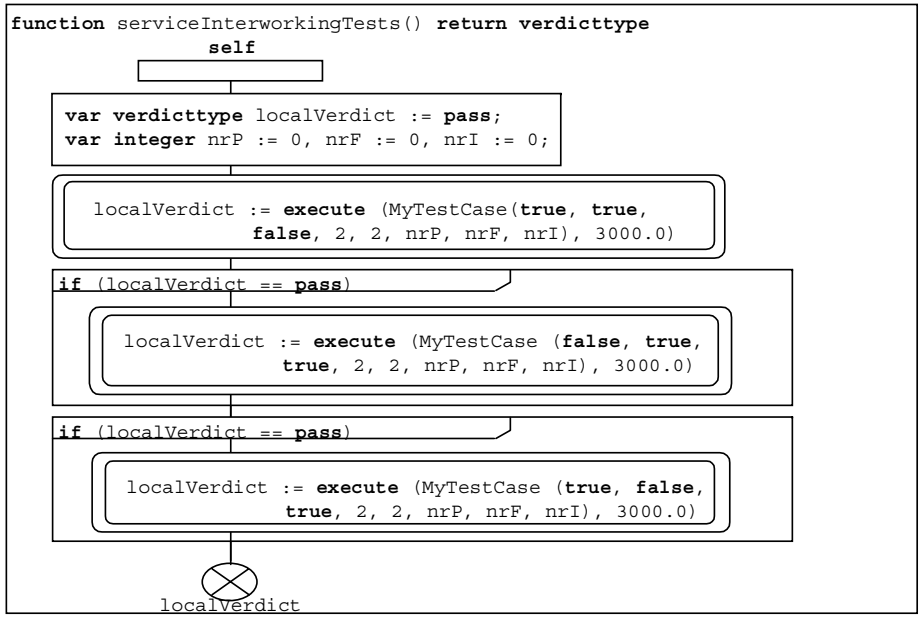



```
function basicCapabilityTests ()
   return verdicttype {
   var verdicttype localVerdict := pass;
   var integer nrP := 0, nrF := 0, nrI  := 0;

   // *** INTERNET ORDER ***
   localVerdict := execute(MyTestCase (true,false,
         false,1,1,nrP,nrF,nrI),1800.0);

   // *** PHONE ORDER
   if (localVerdict == pass) {
      localVerdict := execute(MyTestCase
      (false,true,false,1,1,nrP,nrF,nrI),1800.0);
   }

   // *** RESTAURANT ORDER ***
   if (localVerdict == pass) {
      localVerdict := execute(MyTestCase
      (false,false,true,1,1,nrP,nrF,nrI),1800.0);
   }
   return (localVerdict);
}
```

Рисунок C.6/Z.142 – Пример ресторана – альтернативный шаг internetDefault (интернет, по умолчанию) и функция basicCapabilityTests (тесты основных возможностей)

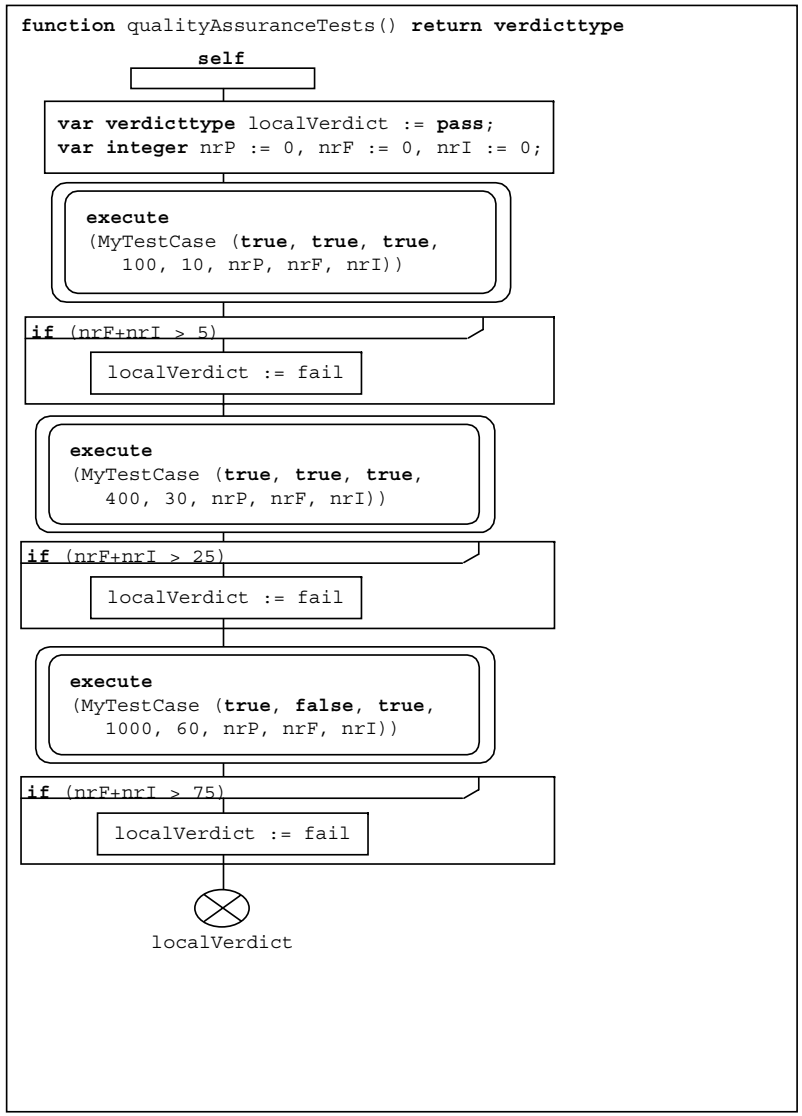



```
function loadTests () return verdicttype {
   var verdicttype localVerdict := pass;
   var integer nrP := 0, nrF := 0, nrI  := 0;

   // *** Minimal load ***
   localVerdict := execute(MyTestCase(
         true,true,true,100,10,nrP,nrF,nrI));

   // *** Medium load ***
   if (localVerdict == pass) {
      localVerdict := execute(MyTestCase(
         true,true,true,400,30,nrP,nrF,nrI));
    }

   // *** Maximal load ***
   if (localVerdict == pass) {
      localVerdict := execute(MyTestCase(
         true,false,true,1000,60,nrP,nrF,nrI));
    }
    return (localVerdict);
}
```



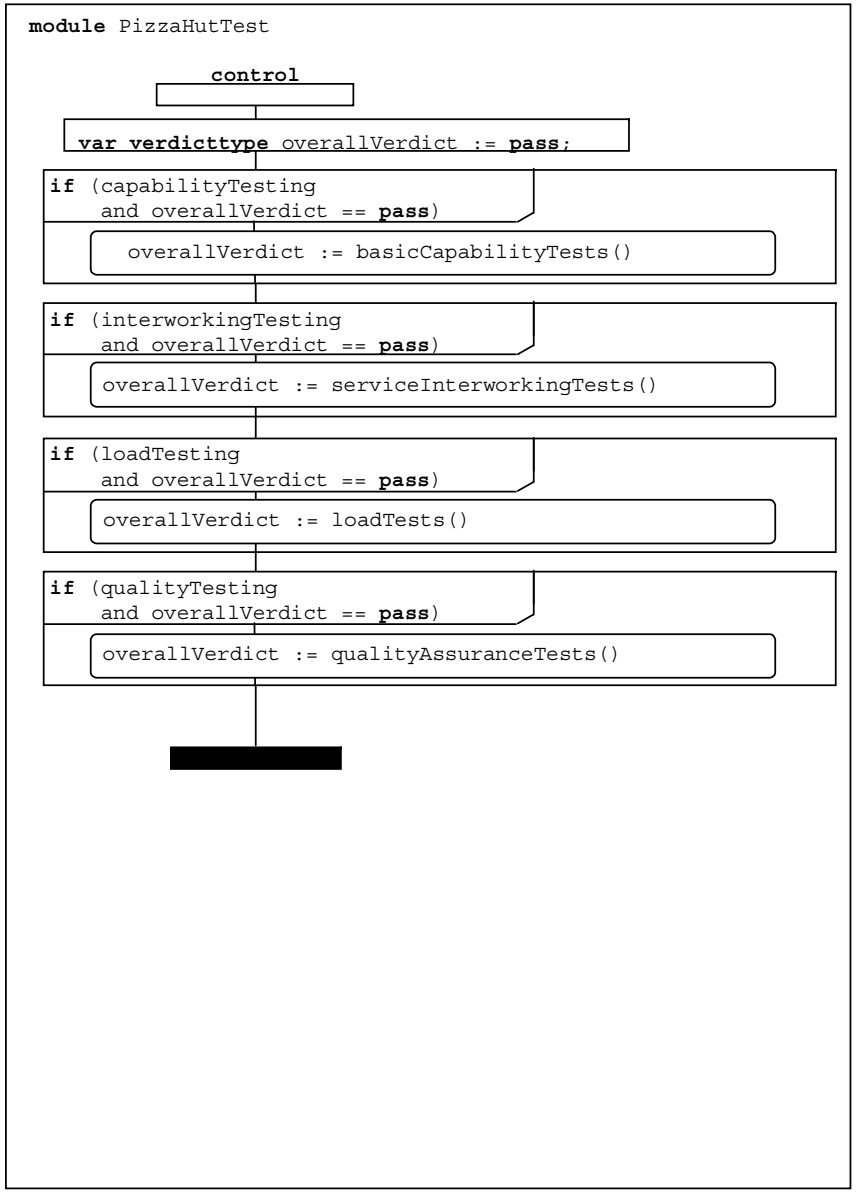

```
function serviceInterworkingTests ()
   return verdicttype {
   var verdicttype localVerdict := pass;
   var integer nrP := 0, nrF := 0, nrI  := 0;

   // *** INTERNET ORDER & PHONE ORDER ***
   localVerdict := execute(MyTestCase(
      true,true,false,2,2,nrP,nrF,nrI),3000.0);

   // *** PHONE ORDER & RESTAURANT ORDER
   if (localVerdict == pass) {
      localVerdict := execute(MyTestCase(
      false,true,true,2,2,nrP,nrF,nrI),3000.0);
   }

   // *** RESTAURANT ORDER & INTERNET ORDER***
   if (localVerdict == pass) {
      localVerdict := execute(MyTestCase(
      true,false,true,2,2,nrP,nrF,nrI),3000.0);
   }

   return (localVerdict);
}
```

Рисунок C.7/Z.142 – Пример ресторана – функции loadTests (загрузка тестов) и serviceInterworkingTests (тесты взаимодействия услуг)




```
function qualityAssuranceTests ()
   return verdicttype {

   var verdicttype localVerdict := pass;
   var integer nrP := 0,
               nrF := 0,
               nrI  := 0;

   // *** Quality under Minimal load ***
   execute(MyTestCase(true,true,true,100,10,
         nrP,nrF,nrI));

   if ( nrF + nrI > 5 ) {
      localVerdict := fail;
   }

   // *** Quality under Medium load ***
   execute(MyTestCase(true,true,true,400,30,
         nrP,nrF,nrI));

   if ( nrF + nrI > 25 ) {
      localVerdict := fail;
   }

   // *** Quality under Maximal load ***
   execute(MyTestCase(true,false,true,1000,60,
         nrP,nrF,nrI));

   if ( nrF + nrI > 75 ) {
      localVerdict := fail;
   }

   return (localVerdict);

} // end function qualityAssuranceTests
```

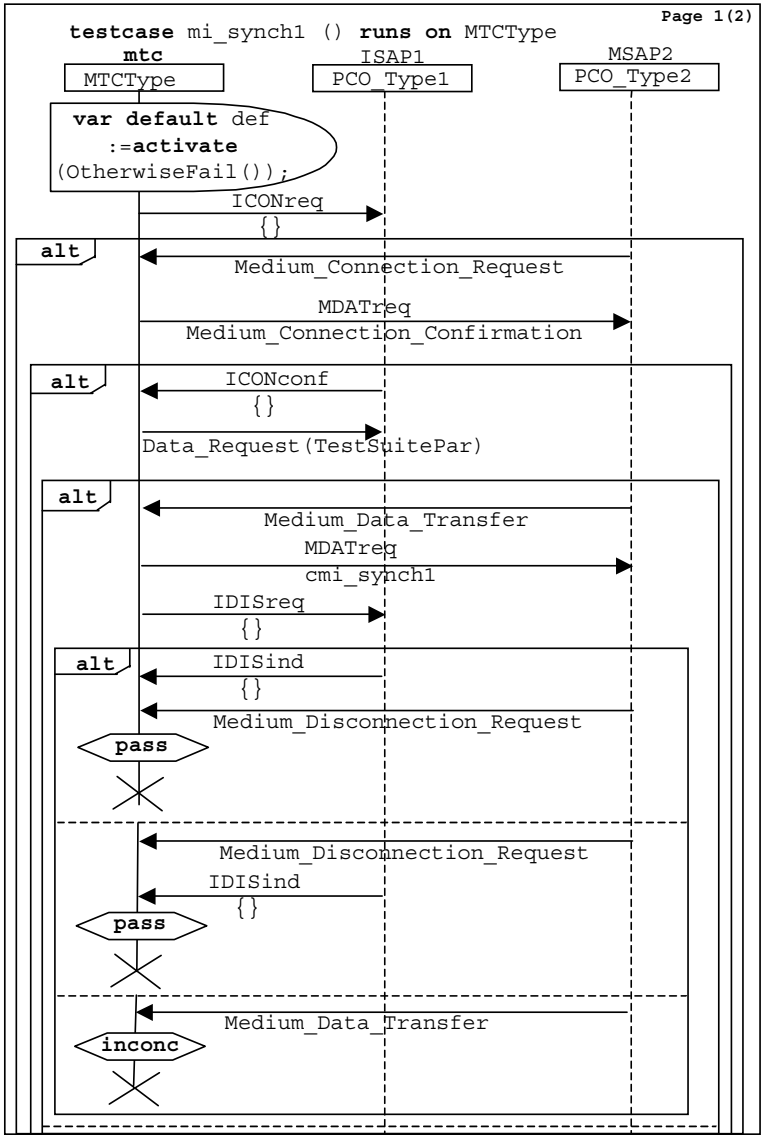
Рисунок C.8/Z.142 – Пример ресторана – qualityAssuranceTests (тесты обеспечения качества)




```
module PizzaHutTest (
      boolean capabilityTesting,
      boolean interworkingTesting,
      boolean loadTesting,
      boolean qualityTesting ) {

   control {

   var verdicttype overallVerdict := pass;

   // Basic Capability Tests
   if (capabilityTesting and overallVerdict == pass) {
      overallVerdict := basicCapabilityTests();
   }

   // Interworking Tests
   if (interworkingTesting and overallVerdict == pass) {
      overallVerdict := serviceInterworkingTests();
   }

   // Load Tests
   if (loadTesting and overallVerdict == pass) {
      overallVerdict := loadTests();
   }

   // Quality Assurance Tests
   if (qualityTesting and overallVerdict == pass) {
      overallVerdict := qualityAssuranceTests();
   }

}
```

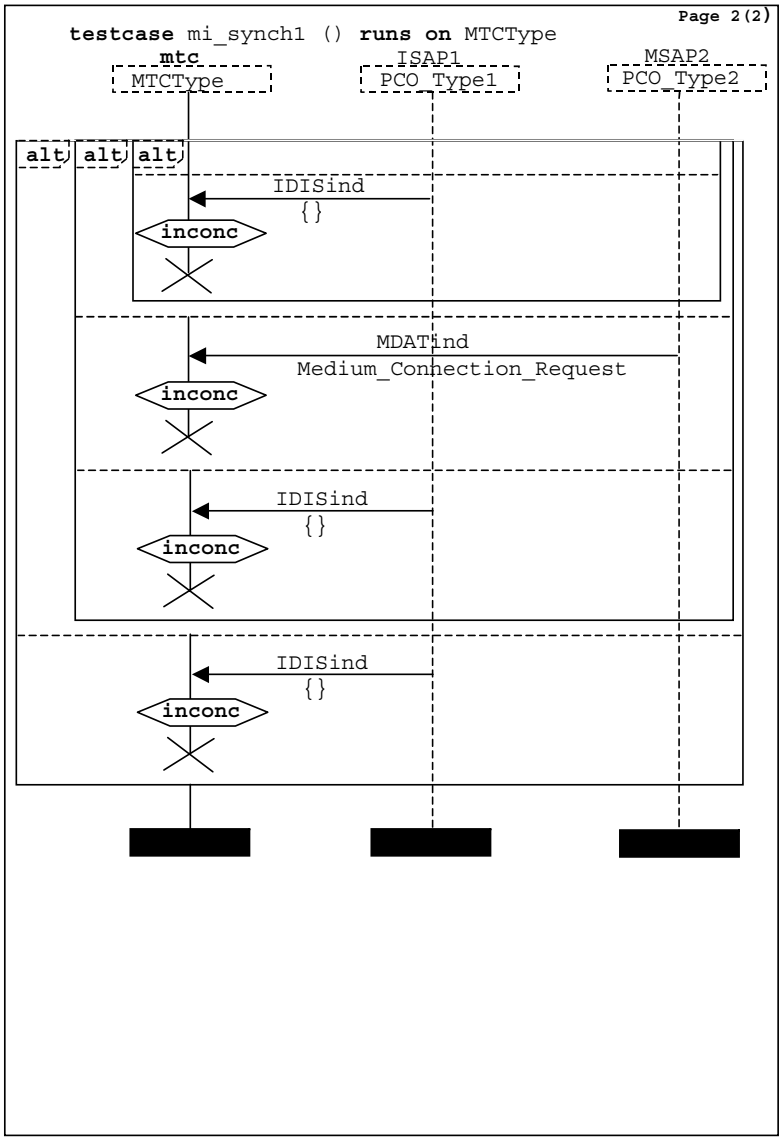
Рисунок C.9/Z.142 – Пример ресторана – модуль PizzaHutTest (тест PizzaHut)

C.2 Пример INRES



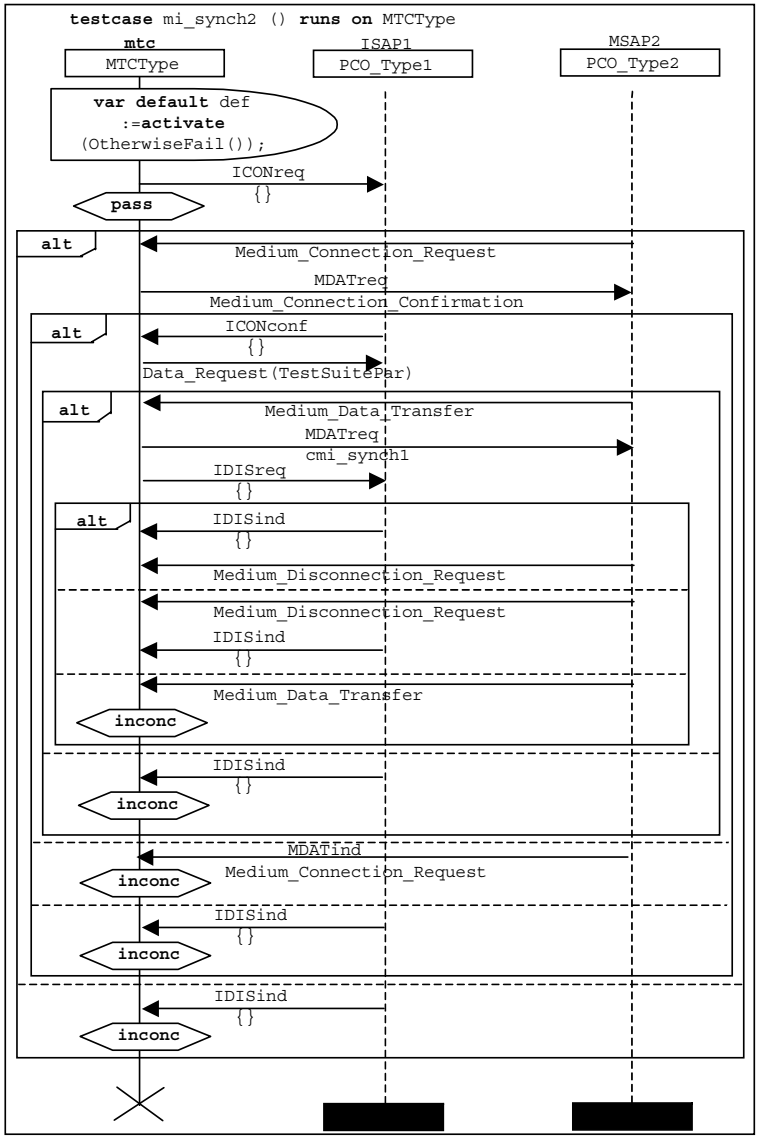

```
testcase mi_synch1 () runs on MTCType {

   /* Default activation */
   var default def := activate(OtherwiseFail());

   /* Inline template definition */
   ISAP1.send( ICONreq:{} );

   alt { /* alt1 */
      [] MSAP2.receive( Medium_Connection_Request ) {
                        /* use of a template */
         MSAP2.send( MDATreq:Medium_Connection_Confirmation );
                        /*optional template type*/

      alt { /* alt2 */
         [] ISAP1.receive ( ICONconf:{} ) {
            ISAP1.send ( Data_Request(TestSuitePar) );


            alt { /* alt3 */
               [] MSAP2.receive( Medium_Data_Transfer ) {
                  MSAP2.send ( MDATreq:cmi_synch1() );
                  ISAP1.send ( IDISreq:{} );



                  alt { /* alt4 */
                     [] ISAP1.receive (IDISind:{} ) {
                        MSAP2.receive(
                              Medium_Disconnection_Request );
                        setverdict(pass);
                        stop;
                     }

                     [] MSAP2.receive(
                              Medium_Disconnection_Request ){
                        ISAP1.receive( IDISind:{} );
                        setverdict(pass);
                        stop;
                     }

                     [] MSAP2.receive(Medium_Data_Transfer ) {
                        setverdict(inconclusive);
                        stop;
                     }
                  } /* end alt4 */
               }
```

Рисунок C.10/Z.142 – Пример INRES – тестовый случай mi_synch1 1(2)



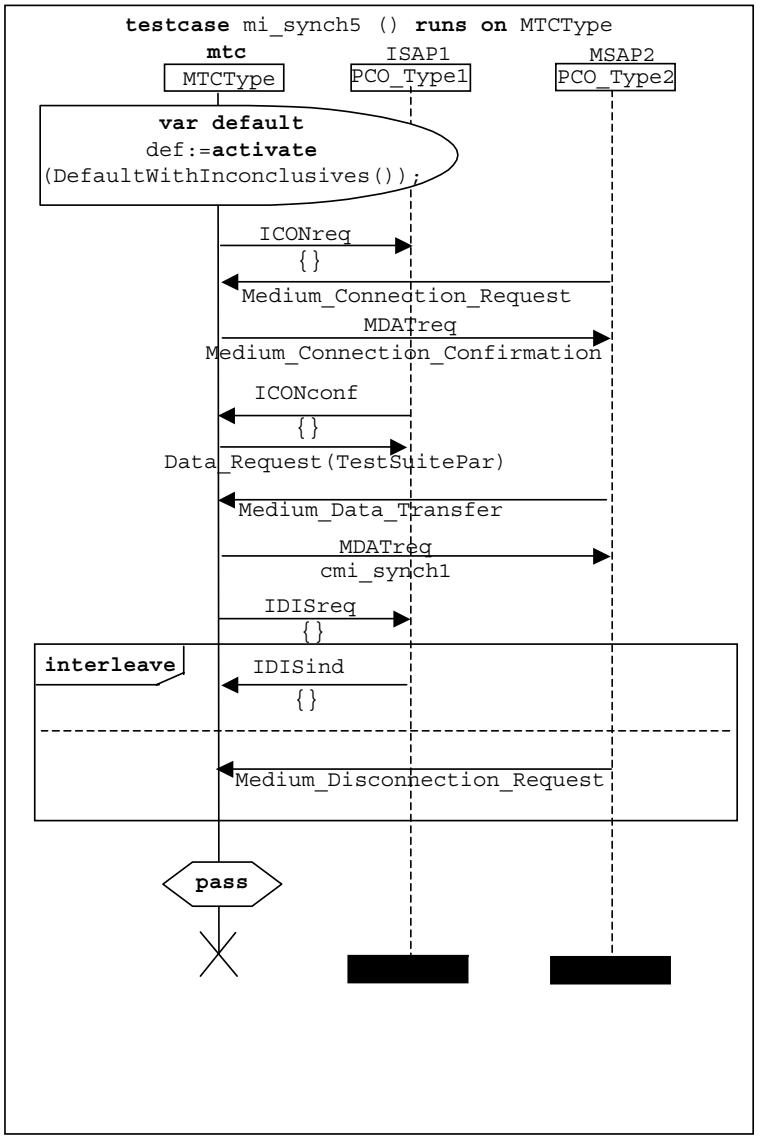

```
/* testcase mi_synch1 () continuation */




              [] ISAP1.receive( IDISind:{} ) {
                 setverdict(inconclusive);
                 stop;
              }
           } /* end alt3 */
        }

        [] MSAP2.receive(
                MDATind:Medium_Connection_Request) {
           setverdict(inconclusive);
        stop;
        }

        [] ISAP1.receive( IDISind:{} ) {
        setverdict(inconclusive);
        stop;
        }
     }
   } /* end alt2 */


[] ISAP1.receive( IDISind:{} ) {
      setverdict(inconclusive);
      stop;
    }
   } /* end alt1 */
 } /* End testcase mi_synch1 */
```

Рисунок C.11/Z.142 – Пример INRES – тестовый случай mi_synch1 2(2)



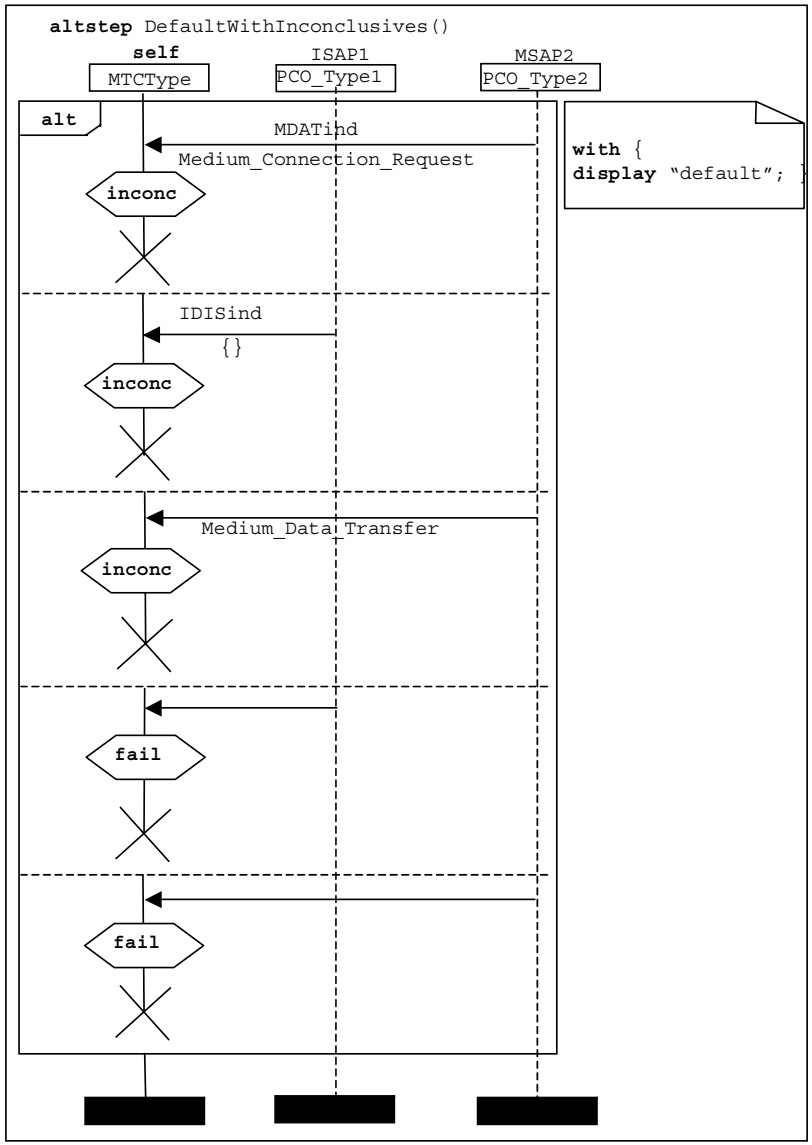

```
testcase mi_synch2 () runs on MTCType {



   var default def := activate(OtherwiseFail());
                      /* Default activation */

   ISAP1.send( ICONreq:{} );
   setverdict(pass);

   alt {
       [] MSAP2.receive( Medium_Connection_Request ) {
          MSAP2.send ( MDATreq:Medium_Connection_Confirmation );
          alt {
              [] ISAP1.receive ( ICONconf:{} ) {
                 ISAP1.send ( Data_Request(TestSuitePar) );
                 alt {
                     [] MSAP2.receive ( Medium_Data_Transfer ) {
                        MSAP2.send ( MDATreq:cmi_synch1 );
                        ISAP1.send ( IDISreq:{} );
                        alt {
                            [] ISAP1.receive (IDISind:{} ) { /* PASS */
                               MSAP2.receive(
                                   Medium_Disconnection_Request );
                            }
                            [] MSAP2.receive(
                                   Medium_Disconnection_Request ){
                               ISAP1.receive( IDISind:{} ); /* PASS */
                            }

                            [] MSAP2.receive ( Medium_Data_Transfer ) {
                               setverdict(inconclusive);
                            }
                        }
                     }
                     [] ISAP1.receive( IDISind:{} ) {
                        setverdict(inconclusive);
                     }
                 }
              }
              [] MSAP2.receive( MDATind:Medium_Connection_Request ) {
                 setverdict(inconclusive);
              }
              [] ISAP1.receive( IDISind:{} ) {
                 setverdict(inconclusive);
              }
          }
       }
       [] ISAP1.receive( IDISind:{} ) {
          setverdict(inconclusive);
       }
   }
   stop; } /* End testcase mi_synch2 */
```

Рисунок C.12/Z.142 – Пример INRES – тестовый случай mi_synch2



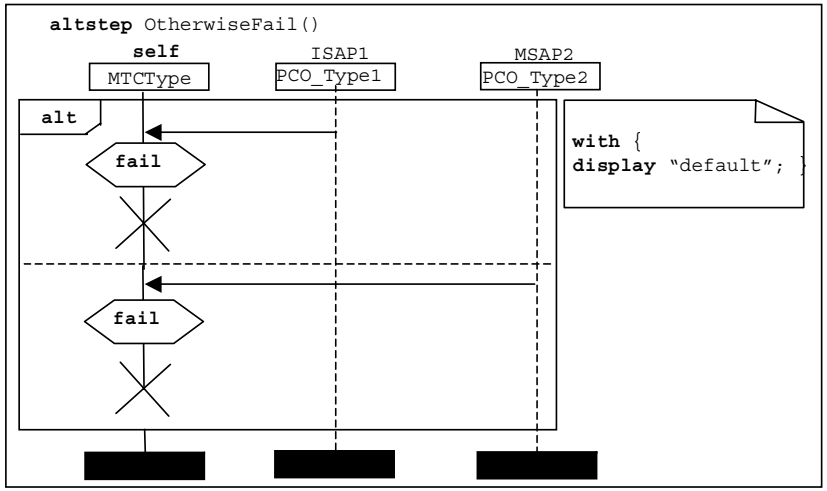

```
testcase mi_synch5 () runs on MTCType {



   var default
        def := activate(DefaultWithInconclusives );
                     /* Default activation */
   /* message ONE and response to ONE */
   ISAP1.send( ICONreq:{} );
   MSAP2.receive(Medium_Connection_Request );

   /* message TWO and response to TWO */
   MSAP2.send(
           MDATreq:Medium_Connection_Confirmation );
   ISAP1.receive ( ICONconf:{} );

   /* message THREE and response to THREE */
   ISAP1.send ( Data_Request(TestSuitePar) );
   MSAP2.receive ( Medium_Data_Transfer );

   /* messages FOUR and FIVE */
   MSAP2.send ( MDATreq:cmi_synch1 );
   ISAP1.send ( IDISreq:{} );

  interleave {
     /* the two responses to messages FOUR and
        FIVE can arrive in any order */
     [] ISAP1.receive(IDISind:{}) {};
     [] MSAP2.receive(
           Medium_Disconnection_Request ) {};
   }

   setverdict(pass);

   stop;

} /* End testcase mi_synch5 */
```

Рисунок C.13/Z.142 – Пример INRES – тестовый случай mi_synch5

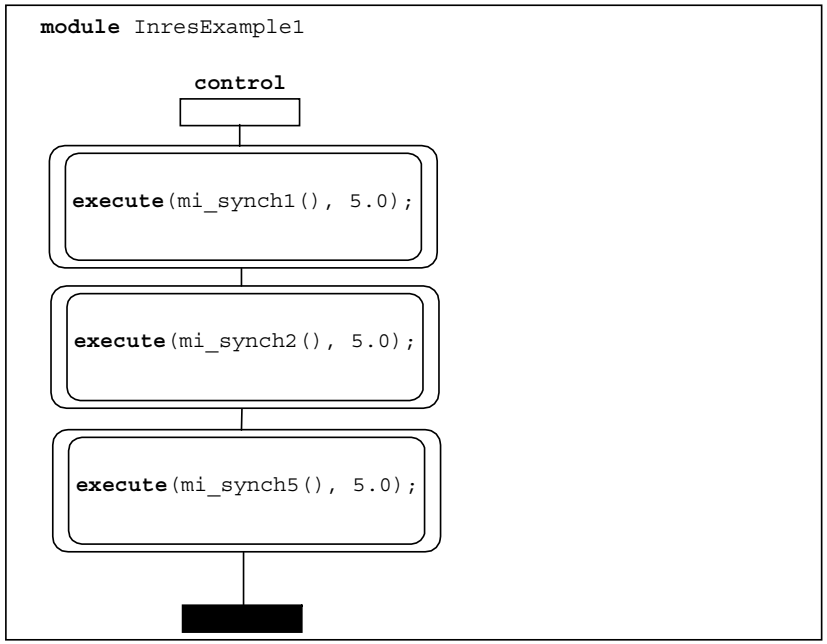



```
altstep DefaultWithInconclusives() {

    /* INCONCLUSIVE CASES */


   [] MSAP2.receive( MDATind:Medium_Connection_Request) {

      setverdict(inconclusive);
      stop;
   }


   [] ISAP1.receive ( IDISind:{} ) {

      setverdict(inconclusive);
      stop;
   }



   [] MSAP2.receive ( Medium_Data_Transfer ) {

      setverdict(inconclusive);
      stop;
   }


  /* FAIL CASES */

   [] ISAP1.receive {

      setverdict(fail);
      stop;

   }


   [] MSAP2.receive {

      setverdict(fail);
      stop;
   }


} with { display "default"; }
```

Рисунок C.14/Z.142 – Пример INRES – Альтернативный шаг DefaultWithInconclusives (с неопределенными результатами, по умолчанию)




```
altstep OtherwiseFail() {

   [] ISAP1.receive {

      setverdict(fail);

      stop;
   }

   [] MSAP2.receive {

      setverdict(fail);

      stop;
  }
} with { display "default"; }
```




```
module InresExample1 {
   ...

   control InresExample {

      execute (mi_synch1(), 5.0);

      execute (mi_synch2(), 5.0);

      execute (mi_synch5(), 5.0);

   } // end control part

}
```

Рисунок C.15/Z.142 – Пример INRES – Альтернативный шаг OtherwiseFail (сбой в ином случае) и определения модуля InresExample1 (пример 1 Inres)

СЕРИИ РЕКОМЕНДАЦИЙ МСЭ-Т

Серия A	Организация работы МСЭ-Т
Серия D	Общие принципы тарификации
Серия E	Общая эксплуатация сети, телефонная служба, функционирование служб и человеческие факторы
Серия F	Нетелефонные службы электросвязи
Серия G	Системы и среда передачи, цифровые системы и сети
Серия H	Аудиовизуальные и мультимедийные системы
Серия I	Цифровая сеть с интеграцией служб
Серия J	Кабельные сети и передача сигналов телевизионных и звуковых программ и других мультимедийных сигналов
Серия K	Защита от помех
Серия L	Конструкция, прокладка и защита кабелей и других элементов линейно-кабельных сооружений
Серия M	Управление электросвязью, включая СУЭ и техническое обслуживание сетей
Серия N	Техническое обслуживание: международные каналы передачи звуковых и телевизионных программ
Серия O	Требования к измерительной аппаратуре
Серия P	Качество телефонной передачи, телефонные установки, сети местных линий
Серия Q	Коммутация и сигнализация
Серия R	Телеграфная передача
Серия S	Оконечное оборудование для телеграфных служб
Серия T	Оконечное оборудование для телематических служб
Серия U	Телеграфная коммутация
Серия V	Передача данных по телефонной сети
Серия X	Сети передачи данных, взаимосвязь открытых систем и безопасность
Серия Y	Глобальная информационная инфраструктура, аспекты межсетевого протокола и сети последующих поколений
Серия Z	**Языки и общие аспекты программного обеспечения для систем электросвязи**

Отпечатано в Швейцарии
Женева, 2007 г.