Лабораторная работа 3

Изучение симулятора в пакете PragmaDev Studio

Цель выполнения данной работы – познакомиться с симулятором, который имеется в составе пакете PragmaDev Studio и позволяет увидеть в динамике, как функционирует созданная модель.

В данной лабораторной работе используется модель системы связи, построенная средствами языка SDL, которая была реализована в лабораторных работах 1 и 2 по алгоритмам п.2.

Задание:

1. С помощью раздела 2 в методических указаниях выполнить демонстрационный пример, который реализуете в проекте, сделанном в лабораторных работах 1 и 2 с помощью пакета PragmaDev Studio.
2. Выполнить индивидуальное задание – используя проект, созданный по заданию 1, выполнить симуляцию для своего варианта исходных данных (вариант определяется последней цифрой пароля, если цифра 0, то вариант 10).
3. Результат симуляции оформить в виде отчета.

1. Методические указания к выполнению лабораторной работы

По опыту разработки программного обеспечения можно утверждать, что любая достаточно сложная программа всегда нуждается в отладке. Более того, если рассматривать программирование как деятельность, то сразу станет понятным, что в этой деятельности отладка зачастую оказывается самым длительным и трудным этапом[1].

Ключевым инструментом для отладки и оптимизации программных продуктов являются симуляторы. В случае компьютерной модели, построенной на основе концепции конечного автомата, симуляторы предоставляют возможность имитировать в динамике поведение автомата. В частности, можно в пошаговом ре-

[1] Безбородов Ю.М. Индивидуальная отладка программ. — М.: Наука. Гл. ред. физ.-мат. литературы, 1982. — 192 с.

жиме прослеживать работу модели и наблюдать за последовательностью переходов между состояниями автомата, одновременно контролируя значения тех переменных, от которых зависит эта последовательность. Таким способом пользователь получает предметное понимание того, как система реагирует на определенные внешние и внутренние стимулы.

По существу, для построения и отладки модели используются одни и те же графические представления. В результате во время тестового прогона модели можно видеть текущие состояния автоматов в составе модели, динамически порождаемые и уничтожаемые экземпляры процессов, передаваемые сообщения и изменяющиеся значения их атрибутов. Это позволяет создавать графические исполняемые спецификации, а затем проверять их до начала программирования на языке высокого уровня.

Симуляция бывает весьма полезной для быстрой первоначальной оценки качества построенной модели. Она в меньшей степени подходит для поиска тонких ошибок, т.к. симулировать большое количество разных сценариев поведения системы непрактично и часто не представляется возможным.

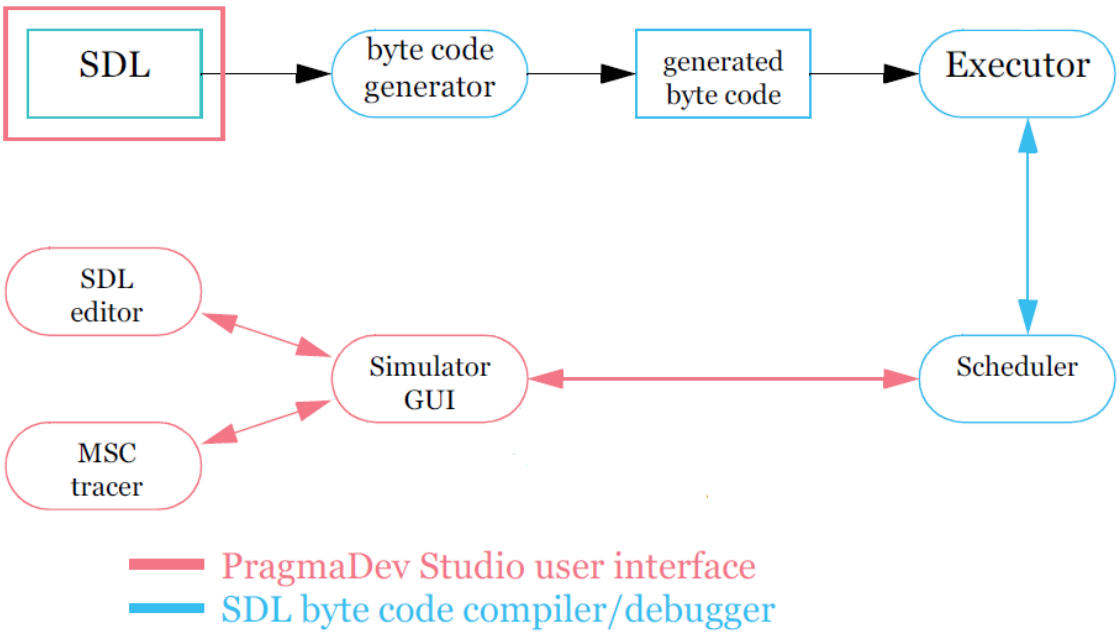

Рис. 1.1. Архитектура встроенного симулятора PragmaDev Studio

Одним из преимуществ пакета PragmaDev Studio является наличие встроенного симулятора, который можно применять в роли отладчика модели. Как следует из архитектуры этого симулятора (рис. 1.1), исходный код модели на языке

SDL предварительно трансформируется в байт-код[2], который передается на исполнение под управлением планировщика операционной системы реального времени (RTOS). Планировщик (Scheduler) также взаимодействует со средствами графического интерфейса симулятора (Simulator GUI), которые дают возможность пользователю в интерактивном режиме участвовать в отладке модели. Отображение хода исполнения модели обеспечивается с помощью трассировщика (MSC tracer) и редактора SDL-диаграмм.

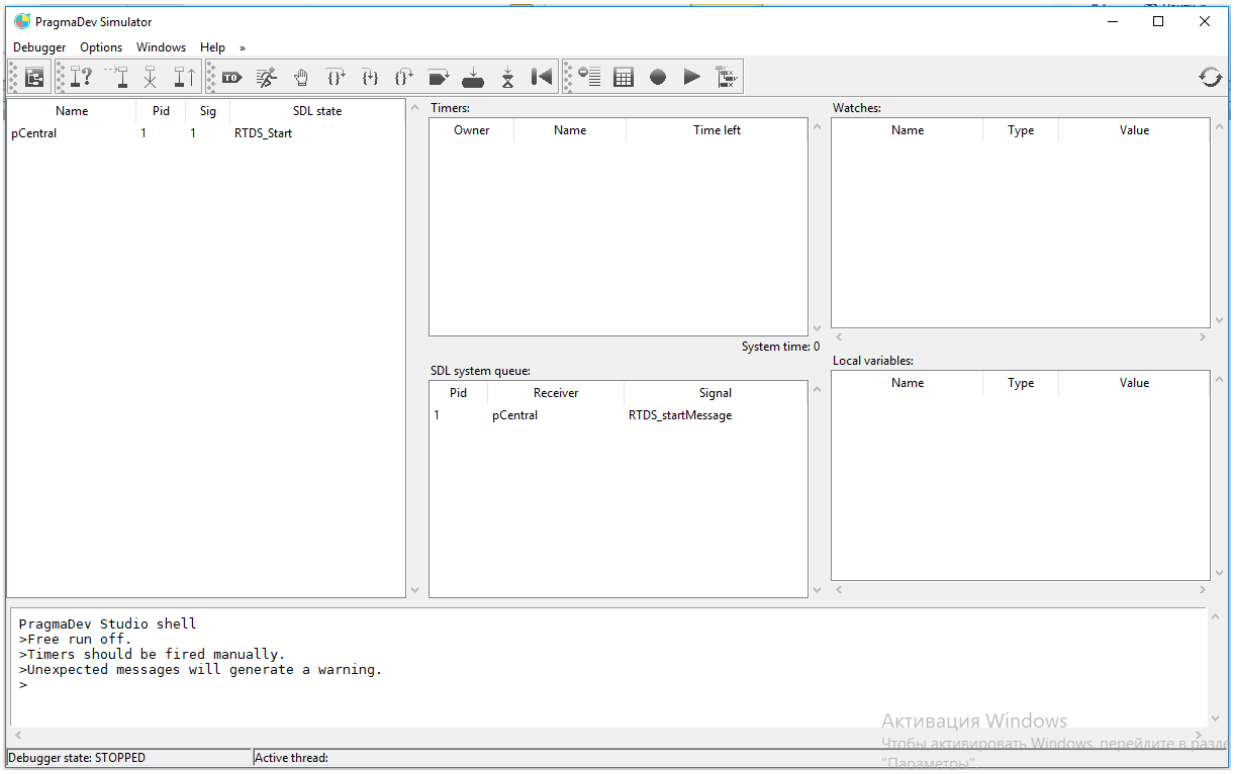

Рис. 1.2. Окно симулятора PragmaDev Studio

Внешний вид рабочего окна встроенного симулятора PragmaDev Studio (сразу после его запуска для модели Phone) представлен на рис. 1.2. Здесь постоянно отображается подробная информация, которая относится к текущему состоянию отлаживаемой модели:

- исполняемые процессы в составе моделируемой системы (левая часть окна);

[2] Байт-код (byte-code) – это способ представления компьютерной программы, которая изначально написана с помощью языка высокого уровня (ЯВУ), в виде специального промежуточного кода. В отличие от исходного кода на ЯВУ, байт-код обеспечивает компактное представление программы, которая уже прошла синтаксический и семантический анализ. С технической точки зрения байт-код представляет собой машинно-независимый код низкого уровня, генерируемый транслятором из исходного кода. По форме байт-код похож на машинный код, но предназначен для исполнения не реальным процессором, а виртуальной машиной.

- активные таймеры (Timers);
- отправленные сообщения (SDL system queue);
- значения переменных, которые находятся под наблюдением (Watches);
- локальные переменные (Local variables), которые относятся к выбранному исполняемому процессу.

2. Выполнение лабораторной работы (демонстрационный пример)

Для демонстрационного примера используются исходные данные, которые остались в модели после выполнения предыдущих лабораторных работ:

- число абонентов в системе NUM_PHONE=3;
- списочные номера абонентов – SN(1)=11, SN(2)=22, SN(3)=33.

1) В окне менеджера проектов пакета PragmaDev Studio отметить значок системы Phone и на панели инструментов нажать кнопку *Execute* (). При этом запускается генерация байт-кода для построения исполняемой модели. Прежде всего, проводится глобальная синтаксическая и семантическая проверка построенной модели системы. Если обнаруживается ошибка, то будет выдано сообщение с указанием характера ошибки. При двойном щелчке мышью на тексте сообщения автоматически откроется окно редактора SDL-диаграмм с отметкой конкретного фрагмента диаграммы, в котором присутствует ошибка.
2) Если построенная модель системы не содержит никаких ошибок, то выдается протокол успешной генерации байт-кода (рис. 2.1) и сразу же запускается встроенный симулятор PragmaDev Studio (рис. 1.2).

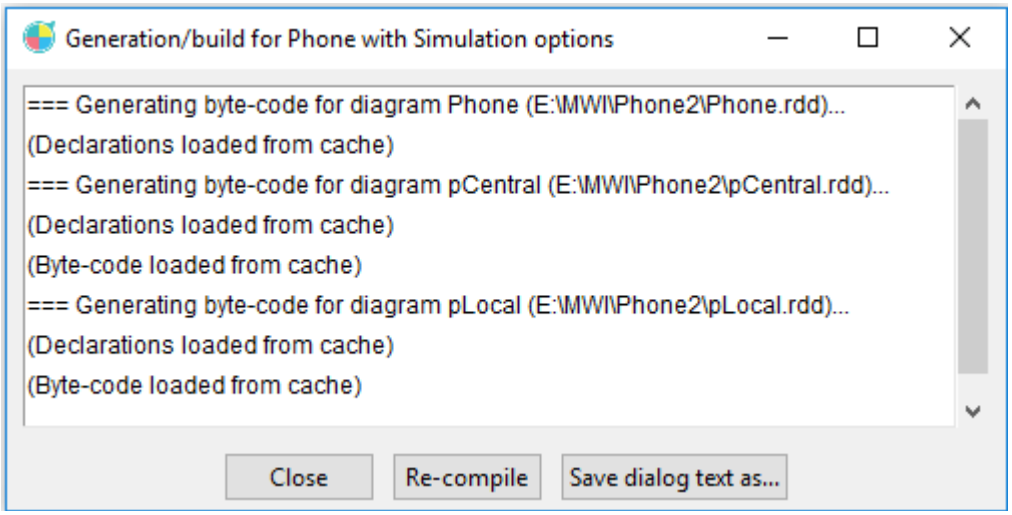

Рис. 2.1

3) На панели инструментов симулятора нажать кнопку *Start MSC trace* ( ). Это делается для того, чтобы открыть параллельное окно MSC Tracer, в котором при прогоне модели последовательность взаимодействия элементов системы будет отображаться с помощью MSC-диаграммы.

4) Для запуска исполняемой модели нажать кнопку *Run the system* ( ) на панели инструментов симулятора. Из предыдущей лабораторной работы следует вспомнить, что функционирование системы Phone начинается с выполнения процесса pCentral, который порождает необходимое количество экземпляров процесса pLocal, отправляет сообщение sReady в окружающую среду и переходит в состояние ожидания (Idle). Каждый экземпляр процесса pLocal после его запуска также переходит в аналогичное состояние.

В графическом виде все эти действия будут отображены в окне MSC Tracer (рис. 2.2). Здесь после названия процесса в скобках указано числовое значение персонального идентификатора (присваивается в момент возникновения в системе рассматриваемого функционального компонента).

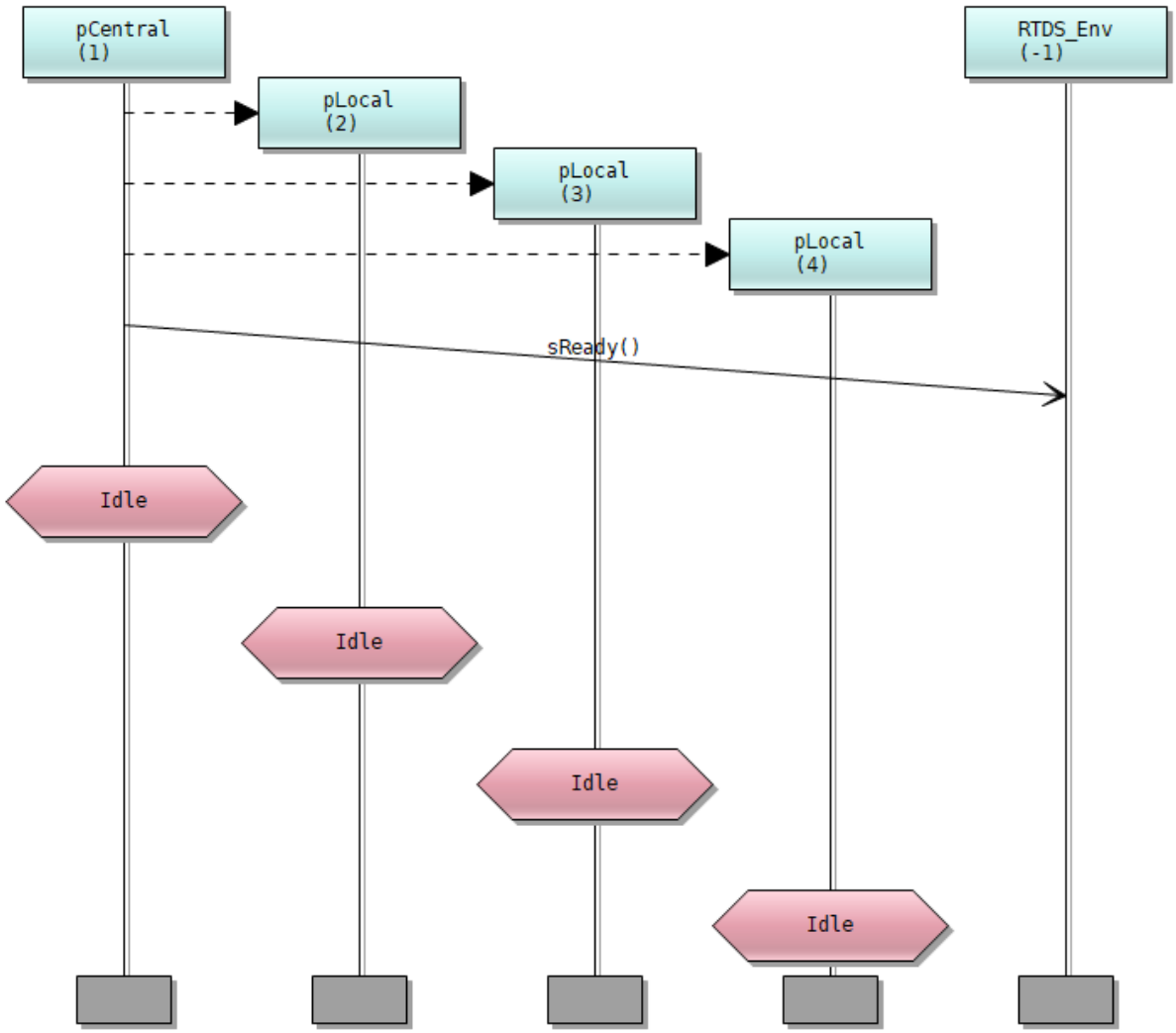

Рис. 2.2

Следует также пояснить, что на рис. 2.2 окружающая среда представлена псевдо-процессом RTDS_Env, который никогда не отображается в окне симулятора в списке выполняемых процессов, а используется только при отслеживании взаимодействия системы с объектами за её границами.

5) Проведём подготовительные действия для имитации поступления вызова от одного из абонентов, которым система Phone предоставляет услуги связи. Прежде всего, нужно нажать кнопку *Stop* ( ) на панели инструментов симулятора и тем самым приостановить исполняемую модель. Затем следует воспользоваться кнопкой *Send an SDL message to the running system* (), которая размещается на этой же панели инструментов и вызывает специальное диалоговое окно для настройки всех параметров, относящихся к операции отправки сообщения (рис. 2.3).

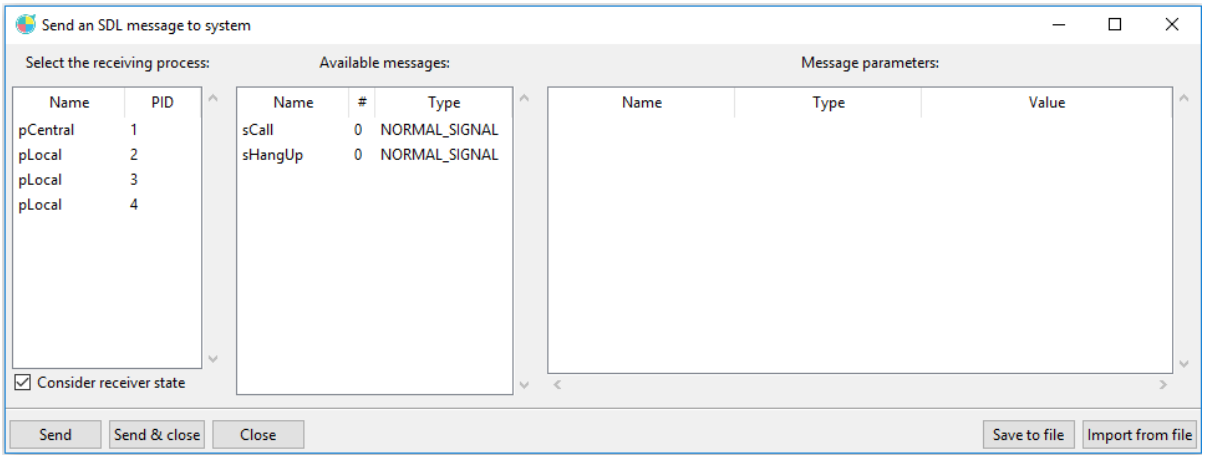

Рис. 2.3

6) Список сообщений (*Available messages*) в средней части окна *Send an SDL message to system* формируется с учетом текущего состояния элементов системы Phone. Сейчас в этом списке представлены только сообщения, которые могут поступить из окружающей среды, поскольку ранее в системе были выполнены всего лишь действия по её инициализации (рис. 2.2). Среди элементов списка *Available messages* нужно отметить сообщение sCall, которое соответствует поступлению в систему нового вызова (рис. 2.4).

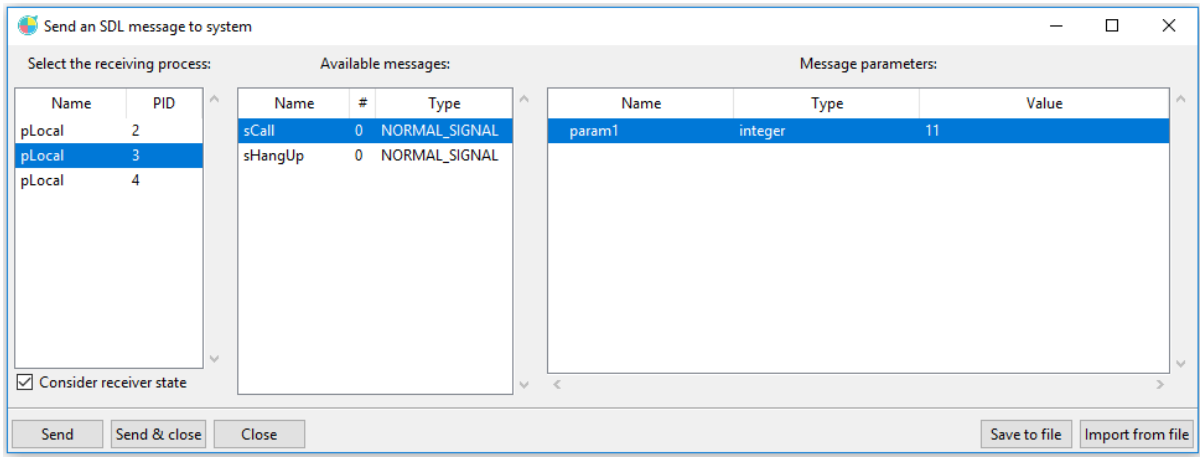

Рис. 2.4

7) В левой части окна *Send an SDL message to system* содержится список процессов, которые могут быть потенциальными получателями поступающего сообщения sCall. Заметим, что после отметки сообщения sCall из этого списка исчезает процесс pCentral, т.к. по правилам функционирования системы Phone он не может получать рассматриваемое сообщение. Пусть новый вызов поступает в терминальное устройство, которое находится под

управлением экземпляра процесса pLocal с PID=3. В списке *Select the receiving process* нужно сделать отметку именно с учетом этого предположения (рис. 2.4).

8) Правая часть окна *Send an SDL message to system* позволяет присвоить конкретные значения тем параметрам, которые относятся к выбранному сообщению. Для каждого параметра в столбце Name указывается условное название параметра (param1, param2 и т.д.), в столбце Type для информации выводится соответствующий тип данных. Остается только сделать двойной щелчок мышью по знакоместу в столбце Value и ввести нужное значение параметра. Завершение этой операции фиксируется нажатием Enter на клавиатуре.

   Напомним, что сообщение sCall обладает единственным параметром, смысл которого – списочный номер вызываемого абонента (один из элементов массива SN в функциональном описании процесса pCentral). Следовательно, пример на рис. 2.4 означает, что вызывающий абонент требует соединения с абонентом, который имеет списочный номер 11.

9) Для отправки сформированного сообщения нужно нажать кнопку «Send & close» (левый нижний угол окна *Send an SDL message to system*).

10) Напомним, что перед тем, как проделать описанные действия по имитации поступления вызова, исполняемая модель системы Phone была приостановлена. Теперь нажатием кнопки *Run the system* ( ) на панели инструментов симулятора нужно возобновить работу этой модели.

11) Трассировка дальнейших действий, которые происходили в системе Phone после приема и обработки рассмотренного сообщения, отображается в окне MSC Tracer (рис. 2.5).

12) Сценарий, представленный на рис. 2.5, получит свое продолжение, когда один из абонентов решает завершить сеанс связи. Пусть, к примеру, абонент A, который ранее инициировал вызов, дает отбой, что сопровождается

поступлением сигнала sHangUp. Имитация этого события проводится по аналогии с действиями, описанными выше в пунктах 5-10. Некоторые отличия возникают только в диалоговом окне *Send an SDL message to system* (рис. 2.6).

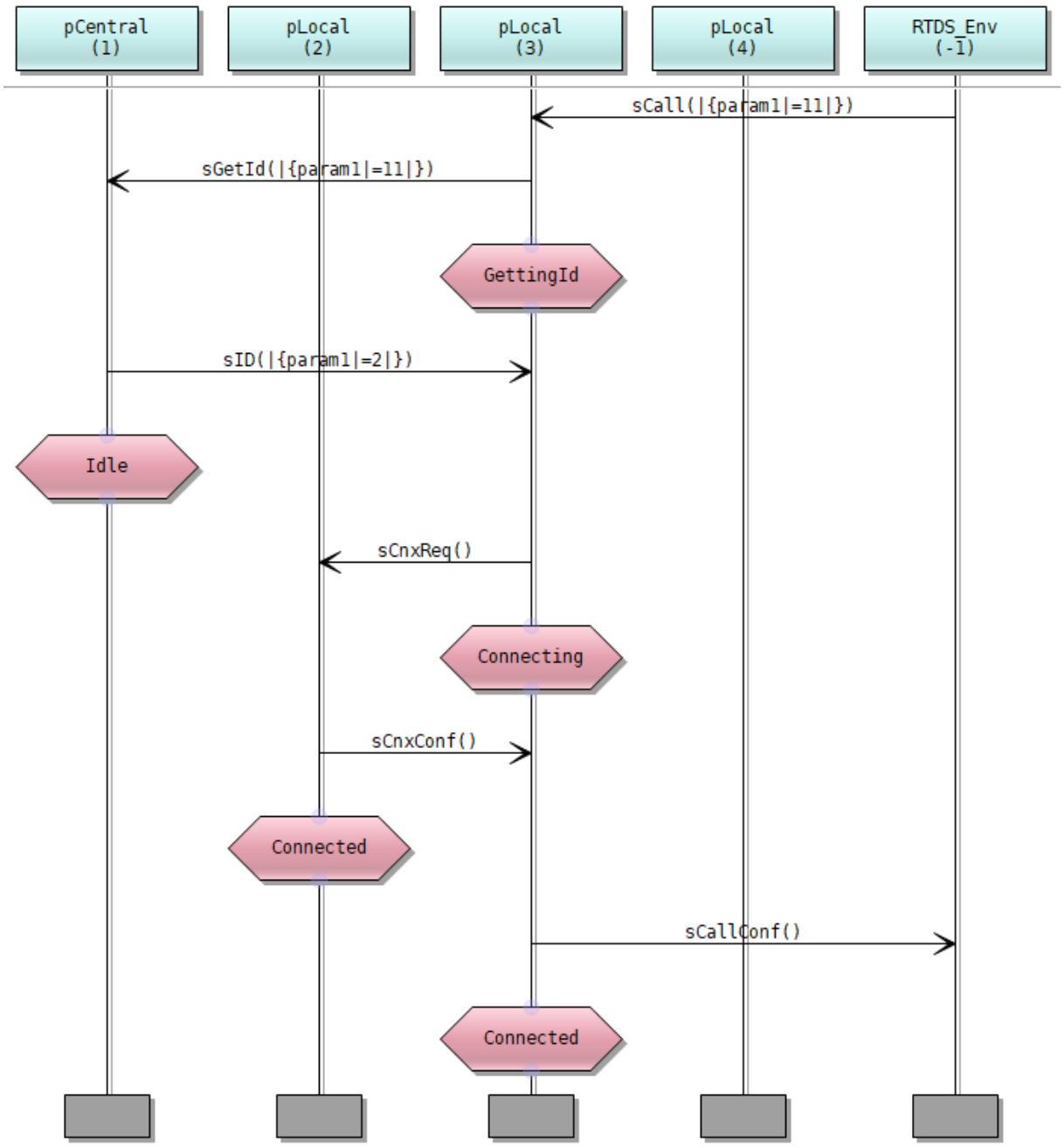

Рис. 2.5

13) Нажатием кнопки «Send & close» в левом нижнем углу окна на рис. 2.6 нужно произвести отправку сформированного сообщения, а затем с помощью кнопки *Run the system* ( ) на панели инструментов симулятора следует возобновить работу исполняемой модели.

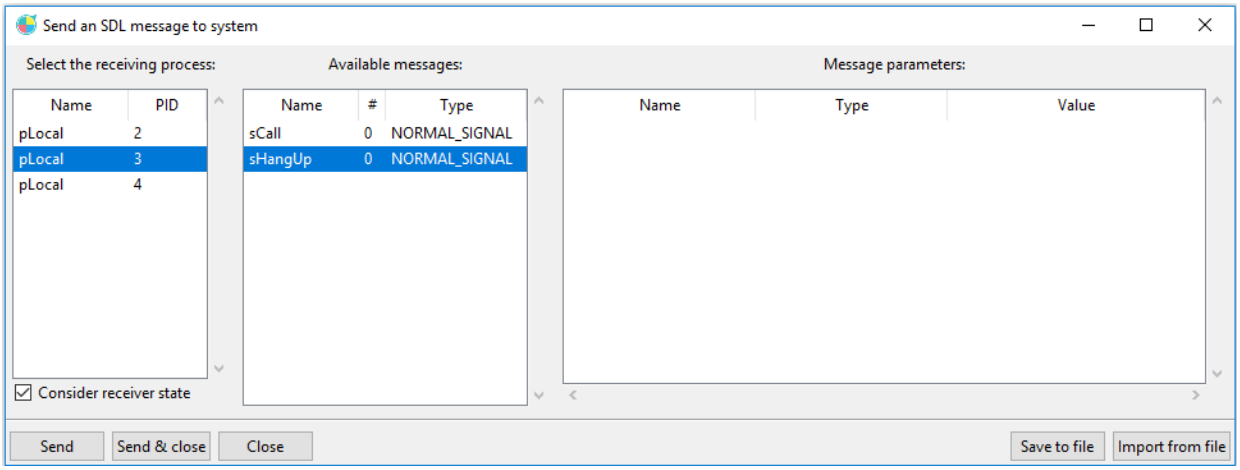

Рис. 2.6

14) Трассировка действий, связанных с приемом и обработкой сигнала отбоя от абонента A, отображается в окне MSC Tracer (рис. 2.7). Эти действия завершаются тем, что экземпляры процесса pLocal, которые участвовали в обслуживании вызова, возвращаются в исходное состояние Idle.

Если диаграммы, представленные на рисунках 2.2, 2.5 и 2.7, сравнить с соответствующими фрагментами диаграммы на рис. 1.1 в методических указаниях к 1-й лабораторной работе, то можно сделать вывод, что общие сценарии взаимодействия элементов системы являются идентичными на содержательном уровне. Следовательно, модель, построенная с использованием языка SDL, полностью реализует требуемый алгоритм функционирования рассматриваемой телекоммуникационной системы.

15) Диаграмму, полученную в окне MSC Tracer, рекомендуется сохранить в виде отдельного файла. В дальнейшем этот файл можно открыть для редактирования с помощью MSC Editor. В частности, диаграммы на рисунках 2.2, 2.5 и 2.7 приведены в отредактированном виде и выглядят гораздо более компактными.

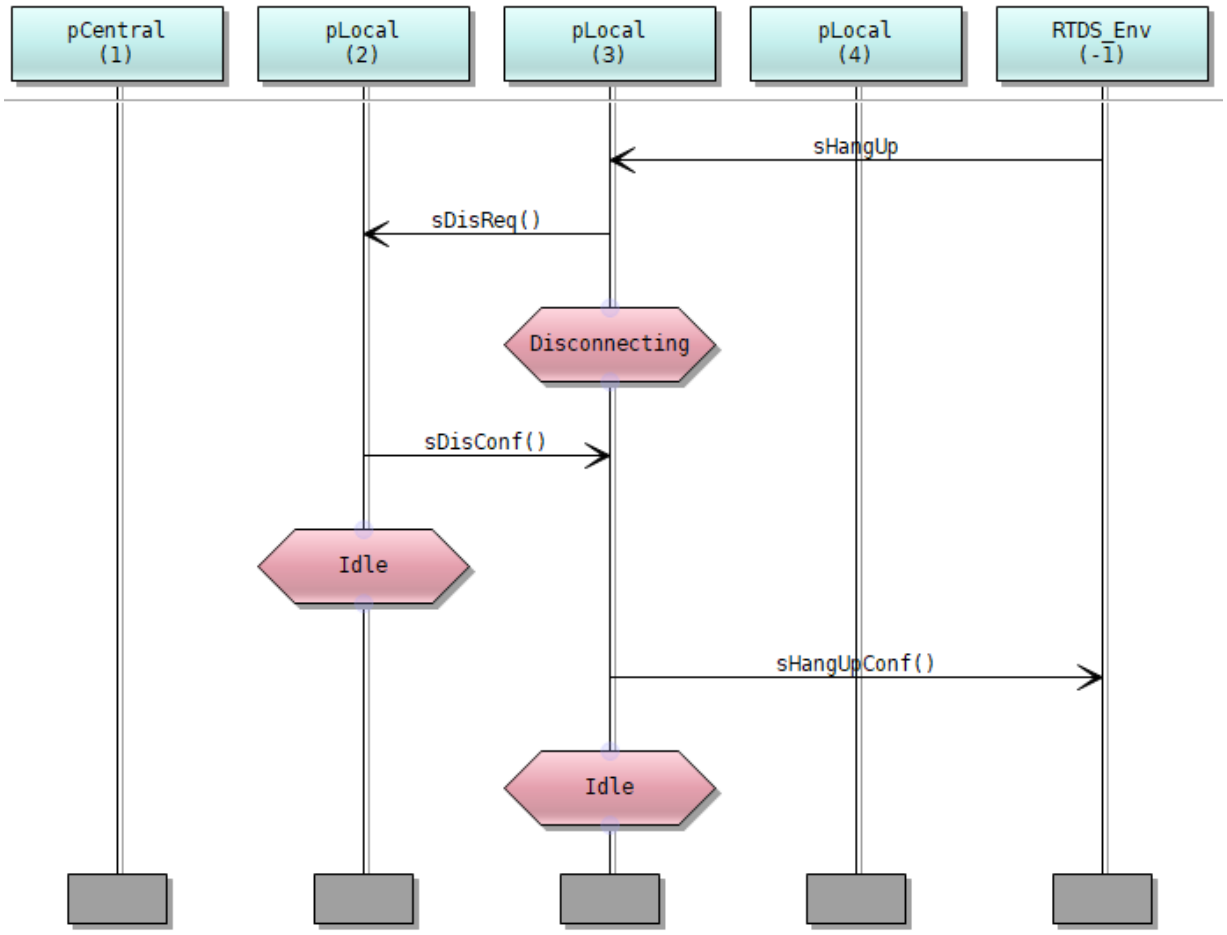

Рис. 2.7

3. Варианты исходных данных для лабораторной работы

Вариант	Списочные номера абонентов	Этапы сценария симуляции
0	21, 58, 60, 83	1) Поступление вызова от абонента 21 и установление соединения с абонентом 83 2) Поступление вызова от абонента 58 к абоненту 92 (несуществующий номер) 3) Поступление сигнала отбоя от абонента 21
1	10, 12, 15, 18	1) Поступление вызова от абонента 12 и установление соединения с абонентом 15 2) Поступление вызова от абонента 18 к абоненту 12 (находится в занятом состоянии) 3) Поступление сигнала об отбое на стороне абонента 15
2	22, 24, 25, 28	1) Поступление вызова от абонента 28 и установление соединения с абонентом 24 2) Поступление вызова от абонента 22 к абоненту 24 (находится в занятом состоянии) 3) Поступление сигнала отбоя от абонента 28
3	30, 34, 35, 39	1) Поступление вызова от абонента 34 и установление соединения с абонентом 35 2) Поступление вызова от абонента 30 к абоненту 66 (несуществующий номер) 3) Поступление сигнала отбоя от абонента 35
4	40, 41, 44, 45	1) Поступление вызова от абонента 45 и установление

		соединения с абонентом 40 2) Поступление вызова от абонента 41 к абоненту 40 (находится в занятом состоянии) 3) Поступление сигнала об отбое на стороне абонента 40
5	50, 52, 56, 57	1) Поступление вызова от абонента 56 и установление соединения с абонентом 50 2) Поступление вызова от абонента 52 к абоненту 56 (находится в занятом состоянии) 3) Поступление сигнала об отбое на стороне абонента 56
6	61, 63, 64, 68	1) Поступление вызова от абонента 61 и установление соединения с абонентом 68 2) Поступление вызова от абонента 64 и установление соединения с абонентом 63 3) Поступление сигнала отбоя от абонента 61 4) Поступление сигнала отбоя от абонента 64
7	74, 76, 78, 79	1) Поступление вызова от абонента 74 и установление соединения с абонентом 78 2) Поступление вызова от абонента 79 и установление соединения с абонентом 76 3) Поступление сигнала об отбое на стороне абонента 78 4) Поступление сигнала отбоя от абонента 79
8	80, 81, 87, 88	1) Поступление вызова от абонента 80 и установление соединения с абонентом 81 2) Поступление вызова от абонента 88 и установление соединения с абонентом 87 3) Поступление сигнала об отбое на стороне абонента 81 4) Поступление сигнала об отбое на стороне абонента 87
9	93, 94, 96, 98	1) Поступление вызова от абонента 93 и установление соединения с абонентом 98 2) Поступление вызова от абонента 94 и установление соединения с абонентом 96 3) Поступление сигнала отбоя от абонента 94 4) Поступление сигнала об отбое на стороне абонента 98

Примечание. Для всех вариантов число абонентов в системе NUM_PHONE=4.

4. Порядок выполнения лабораторной работы

1) Для подготовки к лабораторной работе изучить раздел 1 в методических указаниях.
2) С помощью раздела 2 в методических указаниях выполнить демонстрационный пример, который дает возможность ознакомиться с технологией проведения симуляции в пакете PragmaDev Studio.
3) Выполнить симуляцию для своего варианта исходных данных.
4) Подготовить отчет по выполненной лабораторной работе с описанием про-

цесса симуляции системы для своего варианта исходных данных.

5) К отчету приложить архивный файл с проектом, который был получен при работе с пакетом PragmaDev Studio.