Ознакомительное описание разработки универсальной системы автоматизации контроля и диагностики MAX15

Описание функциональности продукта

Общее описание проблемы

В настоящее время огромное количество областей человеческой деятельности немыслимо без использования компьютеров. Можно выделить области применения, характеризующиеся общим набором абстрактных задач:

- получение компьютером различных данных из внешней среды с помощью функциональных преобразователей,
- использование этих данных для автоматического принятия решений без участия человека,
- управление внешними устройствами в соответствии с принятыми решениями,
- представление информации о собранных данных в удобной для человека форме,
- хранение данных с возможностью их использования при необходимости.

Как правило, решением этих задач занимаются специалисты в соответствующих предметных областях. Не являясь профессиональными программистами, они вынуждены привлекать системных программистов для разработки соответствующих программных средств. Программисты изначально не знают предметные области, а их заказчики изначально не знакомы с программированием. Поэтому поиск решения получается длительным и дорогостоящим. Если заказчик располагает значительными финансовыми возможностями и достаточным временем, он привлекает профессиональных программистов, ставит над ними менеджеров и в конце концов получает результат. Но в большом числе случаев непосредственно специалистам предметной области приходится самим становиться программистами, и разрабатывать программы своими силами. Качество таких программ обычно не высокое, так как современные средства разработки требуют глубоких специальных знаний системного программирования. Существует значительный разрыв между требуемой для использования существующих инструментов квалификацией, и способностями специалистов, не обучавшихся профессиональному программированию. Настоящая разработка делается для того, чтобы ликвидировать этот разрыв. **Разрабатываемое программное обеспечение рассчитано на быстрое создание рабочих управляющих программ силами специалистов в своих предметных областях, при этом не являющихся профессиональными программистами.** Предполагается, что пользоваться этим ПО будут инженеры, имеющие базовое образование по курсу информатики, знакомые с простой алгоритмизацией. При этом они смогут создавать программы с *современным графическим пользовательским интерфейсом*, с поддержкой *многозадачности* (параллельного выполнения нитей кода), *мультиплатформенные* (работающие без изменения на различных программных и аппаратных компьютерных платформах), легко *адаптируемые* (изменяемые при изменении среды выполнения) и легко *масштабируемые* (изменяемые при изменении объемов обрабатываемых данных) – то есть, программы создание которых без привлечения профессиональных системных программистов в настоящее время невозможно. С помощью данного ПО сложные управляющие программы смогут создавать даже дети, так как процесс разработки ПО в большой степени становится похож на игру.

Назначение и поддерживаемые среды выполнения

Назначение

Разрабатываемый программный продукт с рабочим названием MAX15 предназначен для широкого применения на следующих направлениях:

- автоматизация проверки работоспособности различного оборудования и систем на разных этапах производства и эксплуатации — входной производственный контроль, выходной производственный контроль, периодические плановые проверки, срочный и плановый ремонт;
- сбор, обработка, сохранение и представление данных в процессе проведения научных и исследовательских экспериментов;
- сбор, обработка, сохранение и представление данных в автоматизированных системах контроля производственных процессов в случаях, когда использование профессиональных систем SCADA является неоправданно дорогостоящим;
- обучение в разных учебных заведениях основам системотехники, радиоэлектроники, прикладного программирования, комплексных программно-аппаратных решений;
- управление от центрального компьютера в системах типа «умный дом» и «Интернет вещей»
- системы управления роботами, выполняющими одновременно несколько операций
- многие другие приложения, в которых требуется асинхронное получение информации с датчиков, измерительных приборов или команд от пользователя одновременно, автономное принятие решений в соответствии с накопленным опытом и алгоритмическое управление многими исполнительными устройствами одновременно
- в перспективе — интеллектуальное автоматизированное управление различными процессами, с накоплением опыта и прогнозированием развития ситуации, прогностическая диагностика работы различных систем, предотвращение аварийных ситуаций до их возникновения.

Поддерживаемые среды выполнения

Программное обеспечение MAX15 может функционировать в операционных системах, которые поддерживаются мультиплатформенной библиотекой Digia Qt. В настоящий момент времени это операционные системы Windows (от 2000 и до 8.1, включая подсемейство Embeeded), MacOS X, а также большое семейство UINX-подобных ОС, в том числе наиболее распространенные Linux, Solaris, FreeBSD, QNX. Таким образом, применение разрабатываемого ПО используемой операционной системой почти не ограничивается. Необходимо отметить, что ОС Linux применяется в отечественных средствах ВВТ под маркой МС ВС. На базе ОС Solaris создана отечественная ОС «Эльбрус», используемая на одноименных отечественных микропроцессорах. ОС QNX также использована в качестве прототипа при создании отечественной защищенной ОС реального времени «Нейтрино». Всё это позволяет практически без изменений использовать разрабатываемое ПО почти на всех ОС, применяемых в отечественных ВВТ и используемых на соответствующих предприятиях.

Проблема есть только с ОС реального времени VxWorks, поскольку в этой ОС не поддерживается загрузка динамических библиотек во время выполнения приложения. Но применение этой ОС очень ограниченно, она используется в жестких системах реального времени, в которых время реакции критично на уровне миллисекунд (например, это основная бортовая ОС на

самолетах Boeing). Эта ОС крайне редко применяется для решения задач, для которых предназначено разрабатываемое ПО. Хотя, в принципе, при необходимости ПО может быть переработано на использование статически компонуемых библиотек, что позволит ограниченно использовать его в VxWorks – создавать и отлаживать программы можно будет на любой многозадачной ОС, а в VxWorks использовать их со специально скомпонованной версией настоящего ПО.

В настоящее время Qt перенесен в операционные системы Android и iOS. Хотя эти ОС предназначены для потребительских мобильных устройств, и не рассчитаны на выполнение профессиональных приложений, потенциально разрабатываемое ПО может быть использовано и в них.

Для разработки ПО, кроме библиотеки Qt, используется только штатная библиотека и компилятор C/C++ семейства GNU CPP. Эти средства также широко распространены практически на всех платформах, где может применяться разрабатываемое ПО. Речь идет не только об операционных системах, но и об аппаратных платформах. Компилятор GNU CPP позволяет создавать приложения для процессорных архитектур x86, SPARC, MIPS, ARM, PowerPC. Потенциально приложение может работать на всех перечисленных архитектурах без изменения исходного кода. Для PowerPC может потребоваться незначительная переработка средств языка Ci, используемого в приложении.

Размеры компьютеров постоянно уменьшаются. В настоящее время на потребительском рынке есть устройства размером с пачку жевательной резинки, на которых может работать создаваемое ПО. Несмотря на размеры, это полноценные ПК. В недалеком будущем ПО MAX15 можно будет использовать в тех случаях, где сейчас используются PIC-контроллеры, но при этом создание прикладных программ будет на порядки проще.

Разработка и отладка ПО MAX15 производится одновременно в трех вариантах операционных систем: Windows 32 бита (XP SP3), Linux Kubuntu 14 32 бита и Linux Kubuntu 14 64 бита. Во всех трёх ОС ПО работает идентично, с возможностью обмена данными путем простого копирования. Без изменений ПО в любой момент времени может быть собрано для Windows 64 бит любой версии, а также для любой поддерживаемой Qt операционной системы.

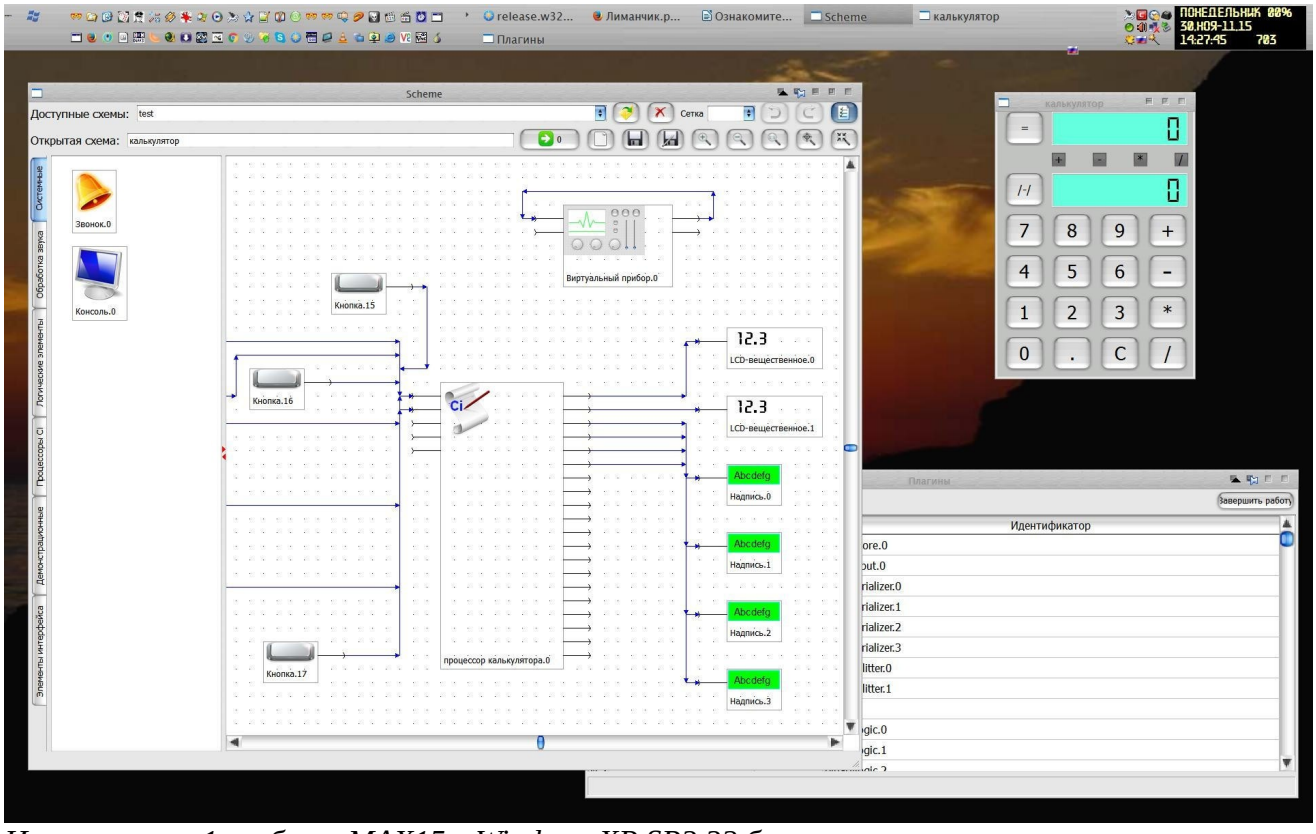

Иллюстрация 1: работа MAX15 в Windows XP SP3 32 бита

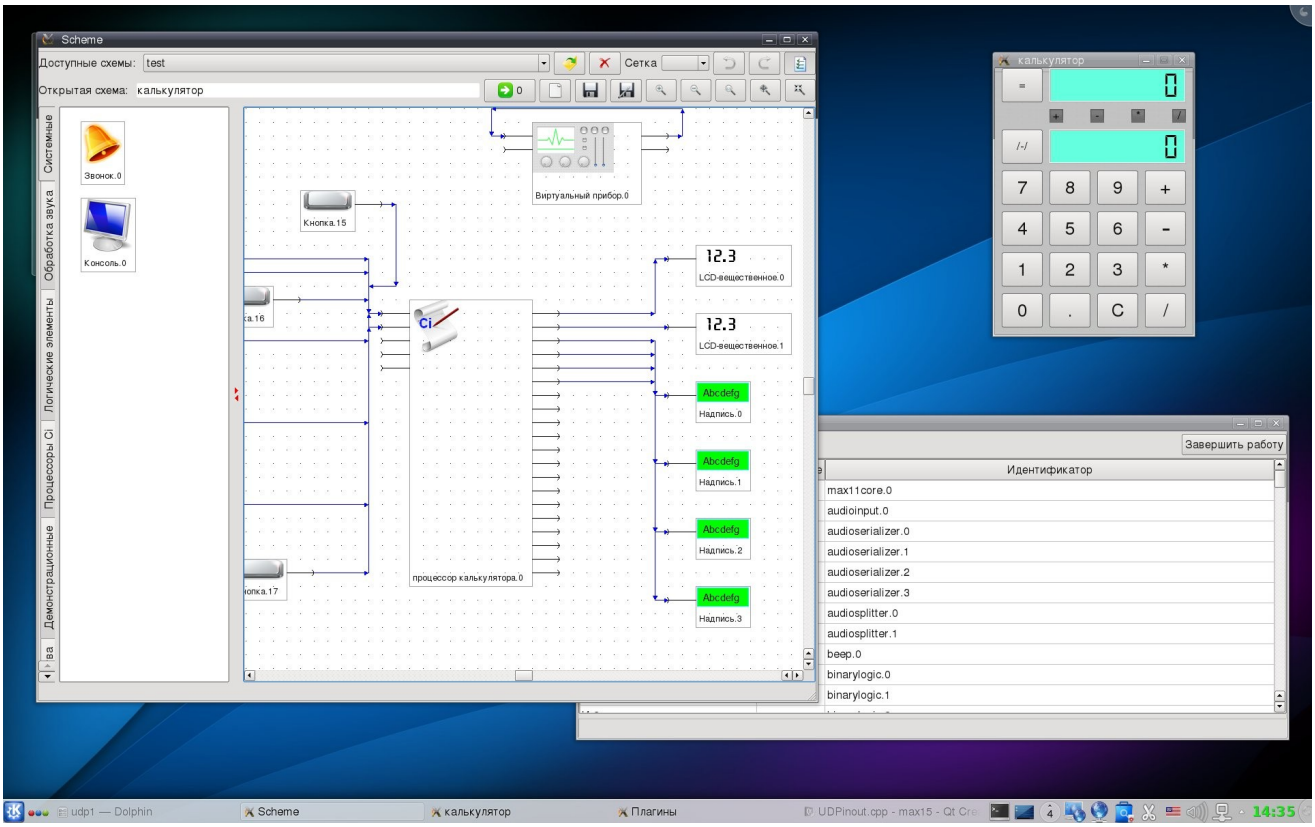

Иллюстрация 2: работа MAX15 в Linux Kubuntu 14.04 64 бита

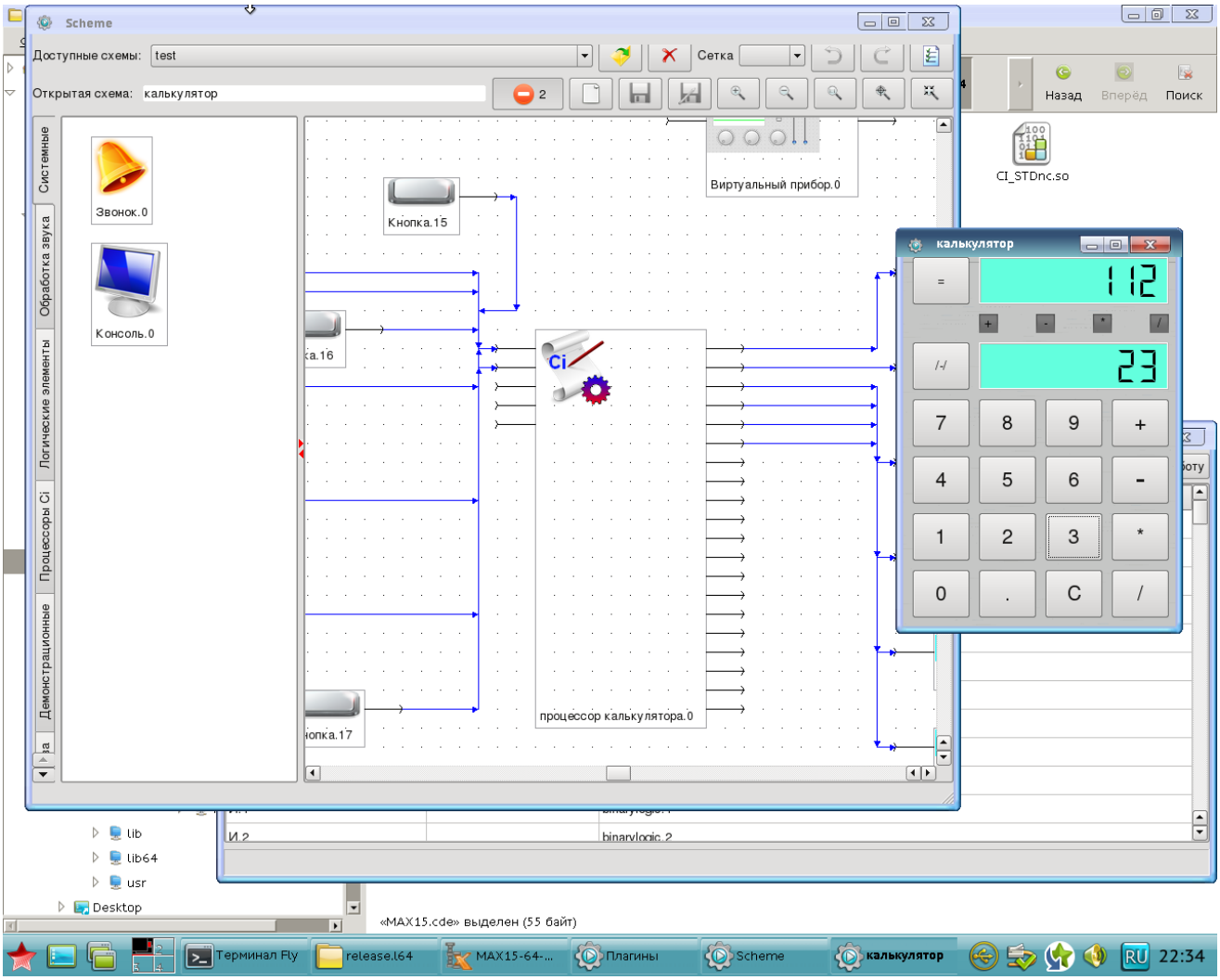

Иллюстрация 3: работа MAX15 в Astra Linux "Орел" 1.10.7

Общее описание

Модульный принцип

Существенной особенностью ПО MAX15 является его модульное построение. Основой всей системы является ядро, главные функции которого — загрузка модулей и установка между ними необходимых взаимосвязей. На носителе модули существуют, как динамически загружаемые библиотеки, причем собираемые в том формате, который является «родным» для используемой операционной системы. То есть в Windows это будут модули в формате DLL, в ОС семейства UNIX это будут разделяемые библиотечные архивы (.so) стандартного для каждой ОС формата. Это обеспечивается использованием соответствующих классов Qt, которые дают универсальный механизм, не зависящий от операционной системы.

При создании модулей необходимо придерживаться некоторой несложной схемы, задаваемой специальными классами, входящими в исходные тексты приложения. Таким образом, разработчику модулей не требуется заботиться об их загрузке и выгрузке, в простейшем случае модуль может вообще не содержать никакого специфичного кода, тем не менее, он может быть опознан и загружен ядром.

Ядро также обеспечивает для модулей несколько «стандартных» механизмов — в первую очередь, это механизм хранения, редактирования и восстановления настроек. Разработчик модуля должен использовать некоторые стандартизованные в приложении функции, наследовать определенные классы, и тогда ядро будет само сохранять настройки модуля, а если модуль создает окно, то также автоматически будут сохраняться и восстанавливаться его положение и размеры на экране.

Модули взаимодействуют друг с другом и с ядром посредством динамически изменяемых соединений, реализованных на предоставленном библиотекой Qt механизме сигналов-слотов. Это позволяет легко заменять модули при необходимости даже без остановки приложения. Такая замена может потребоваться, например, при горячей замене контролирующего оборудования, которым будет управлять ПО.

Также модульность существенно упрощает конфигурирование системы под требования задачи. Если разработка приложения на базе этой системы ведётся на полностью укомплектованном приложении, со всеми доступными модулями, то после того, как приложение отлажено, в результирующую поставку могут быть включены только те модули, которые необходимы для дальнейшей работы.

Наконец, модульность позволяет наращивать функциональность системы и созданных на её основе приложений практически без ограничений. При этом те части, которые не имеют отношения к новым функциям, не затрагиваются. Это обеспечивает преемственность и стабильность в работе.

Ниже (Рис. 1) приводится общая структура разрабатываемого приложения. Чтобы не загромождать её, связи между модулями не показаны — соединения могут быть установлены между любыми модулями в любом направлении, в том числе, изменены динамически.

Рис. 1

Обязательным на рисунке является только ядро, оно является основным исполнимым файлом ПО. Состав всех остальных модулей может изменяться в зависимости от решаемой с помощью ПО задачи. Показанные пунктиром модули еще не реализованы.

В англоязычной терминологии такие модули называются *плагинами*, это обозначение используется в исходных текстах ПО, и возможно будет использоваться в документации, как фактически общепринятое.

Парадигма “схема+программы“

Традиционно при создании современных управляющих систем используется один и тот же подход — к одному или нескольким компьютерам подключены различные устройства ввода или вывода данных. Это могут быть самые разные приборы, с разными уровнями сложности, от простой кнопки, до сложных дорогостоящих измерительных приборов. Сложность и мощность компьютеров тоже может быть различной — от простых PIC-контроллеров с одной функциональной программой, и до многопроцессорных комплексов, работающих под управлением многозадачных операционных систем. Структура практически всегда примерно одна и та же — компьютер под управлением некоторого алгоритма получает и обрабатывает данные от

внешних приборов. Использование компьютера позволяет гибко перестраивать работу системы, а классический императивный алгоритм управления легко модифицируется. Алгоритмические языки хорошо развиты, и позволяют решать большое количество самых разных задач. Схема с центральным управляющим компьютером отлично себя зарекомендовала, хорошо известна и понятна практически всем современным инженерам.

По этим причинам при разработке этого ПО была принята парадигма «схема+программа», по которой решение задачи создается традиционным для управляющих систем способом — в центре один или несколько «виртуальных процессоров» работающие по простым алгоритмам. Процессоров может быть несколько, они работают совершенно независимо друг от друга. Каждый имеет собственные каналы ввода и вывода, они могут принимать, обрабатывать и выдавать данные независимо от работы всей остальной части системы. Их можно рассматривать, как программные аналоги PIC-контроллеров. Программы для этих процессоров пишутся на алгоритмическом языке, специально разработанном для подобного использования. Подробнее речь о нём пойдет далее.

Кроме процессоров, решение может содержать различное количество виртуальных устройств ввода, вывода и обработки данных, которые реализуются на языке C/C++ с использованием библиотеки Qt или других необходимых библиотек. Компоновка виртуального устройства с драйвером физической аппаратуры позволяет легко получить возможность ввода и вывода данных из хост-компьютера в реальный мир. Каждое виртуальное устройство имеет входы и выходы, принимающие или выдающие данные определенного типа. Однотипные входы и выходы могут быть соединены однонаправленными линиями, или *сигналами*.

Таким образом, совокупность виртуальных устройств, виртуальных процессоров с программами, и соединений между ними создает схему *виртуального прибора*, который решает поставленную задачу. Процесс создания прибора в разрабатываемом ПО состоит из:

- размещения на схеме необходимых виртуальных устройств и необходимого числа виртуальных процессоров
- установки между ними необходимых соединений
- задания виртуальным устройствам необходимых характеристик в их настройках
- написания и отладки программ для виртуальных процессоров
- проверки и отладки.

После выполнения этих этапов останется только скопировать каталоги с необходимыми файлами и удалить не используемые библиотеки. Созданное с помощью MAX15 приложение будет готово к использованию или поставке.

Активные и пассивные элементы

Элементы, из которых строится виртуальный прибор, могут быть двух основных типов — «пассивные» и «активные».

«Пассивными» являются те элементы прибора, которые при размещении на схеме немедленно готовы к использованию. Для их активации, кроме размещения на схеме, подключения и настроек характеристик, не требуется более никаких действий. Использование таких элементов показано в видео-файле record_000001.avi[1] на примере системной консоли и кнопки.

Пассивные элементы работают в основной нити приложения, поэтому параллельно несколько таких элементов активироваться не могут, в каждый момент времени может быть задейство-

1 Этот и остальные видео-фрагменты записаны на мониторе диагональю 24” с разрешением 1920x1200. Поэтому их рекомендуется просматривать на мониторах с аналогичными параметрами. Рекомендуется развернуть изображение на полный экран.

ван только один такой элемент. Для тех случаев, когда необходима параллельная обработка, можно использовать «активные» элементы. «Активными» называются элементы прибора, которые выполняются «в фоне», то есть каждый работает в своей отдельной нити приложения. Для управления нитями используются соответствующие функции библиотеки Qt, которые в свою очередь, задействуют механизмы операционной системы. Это гарантирует корректность параллельного выполнения в многозадачных операционных системах, а также обеспечивает эффективное распределение нитей по имеющимся ресурсам — процессорным ядрам. При этом разработчику виртуального прибора нет необходимости заботиться о параллельном выполнении — за него всё делает ядро разрабатываемого приложения. Для пользователя MAX15 разница между активными и пассивными элементами только в том, что для использования активных элементов надо дополнительно запустить их выполнение.

Для одновременного запуска и остановки всех активных элементов имеется специальная кнопка на редакторе схемы. Элементы также могут быть запущены или остановлены по сигналу внутри самой схемы. Если необходимо запускать элементы автоматически после загрузки схемы, для этого имеется соответствующая опция в настройках. При выполнении их нитей активные элементы на схеме помечаются вращающимися шестеренками. Пример создания схемы с активными элементами показан в видео-фрагменте record_000002.avi. В примере используются два виртуальных генератора, которые реализованы программно. В каждом из генераторов для создания сигналов при включении в фоне выполняется собственная нить, заполняющая буфер вычисленными отсчетами. Содержимое буфера периодически выдается в выходном сигнале. Виртуальный осциллограф имитирует простейший осциллограф с пластинами для отклонения луча по горизонтали и по вертикали. В нём также выполняется собственная нить, которая сканирует полученные в буферах входных сигналов данные, и рисует точки или линии на «экране». Все нити работают независимо друг от друга, и от основной нити приложения, в которой работает пользовательский интерфейс. Аналогичным образом можно создавать приложения, в которых работает сколь угодно большое количество параллельных нитей. Благодаря этому, можно легко создавать приложения, эффективно использующие большое количество процессорных ядер — это пригодится в скором будущем, когда многоядерные процессоры появятся на рынке.

ПО разработано таким образом, что во время работы активных элементов остаются доступны все средства для редактирования. Пример динамического изменения соединений показан в видео-фрагменте record_000003.avi[2] (фрагмент содержит звуковое сопровождение, поэтому необходимо включить звук и рекомендуется сделать его погромче, он записан очень тихо). Кроме редактирования соединений, во время работы можно добавлять и удалять элементы прибора (как пассивные, так и активные), изменять их настройки и т.д.

Простое редактирование

Создание схемы виртуального прибора

Как видно из приведенных ранее видео-фрагментов, редактирование схемы интуитивно простое. Элементы прибора перетаскиваются мышью из палитры, расположенной в левой части редактора схемы. Палитра разделена на группы элементов. Количество групп не ограничено, они автоматически появляются и исчезают при добавлении в состав ПО или удалении из

2 При записи фрагмента видео-рекордер создавал значительную нагрузку на процессор, из-за чего была потеряна плавность отображения на экране осциллографа.

него соответствующих элементов.

Также в видео-фрагментах показана простота установки соединений. В простейшем случае для этого надо навести курсор на соединитель элемента, нажать кнопку мыши и потащить линию на любой совместимый соединитель другого элемента. При установке соединений редактор автоматически подсвечивает совместимые входы или выходы, и показывает флажки с их именами. «Совместимость» означает, что входы могут напрямую принимать данные, передаваемые через совместимый выход. При перетаскивании линия от соединителя остаётся красной, пока курсор не будет над совместимым соединителем, тогда она станет зелёной. Если при этом отпустить кнопку мыши, будет установлено соединение между выходом и входом, а на схеме будет автоматически проложена линия из ортогональных отрезков. Пример установки соединения виден в видео-фрагменте record_000001.avi. Линия всегда прокладывается так, чтобы не пересекать изображения элементов схемы, и чтобы она содержала минимальное количество отрезков.

Перетаскиванием конца соединителя можно одновременно соединить только один соединитель с еще одним. Если требуется соединить один соединитель с несколькими, то двойным кликом на нём, либо через его локальное меню можно вызвать диалог с перечислением всех совместимых «разъемов».

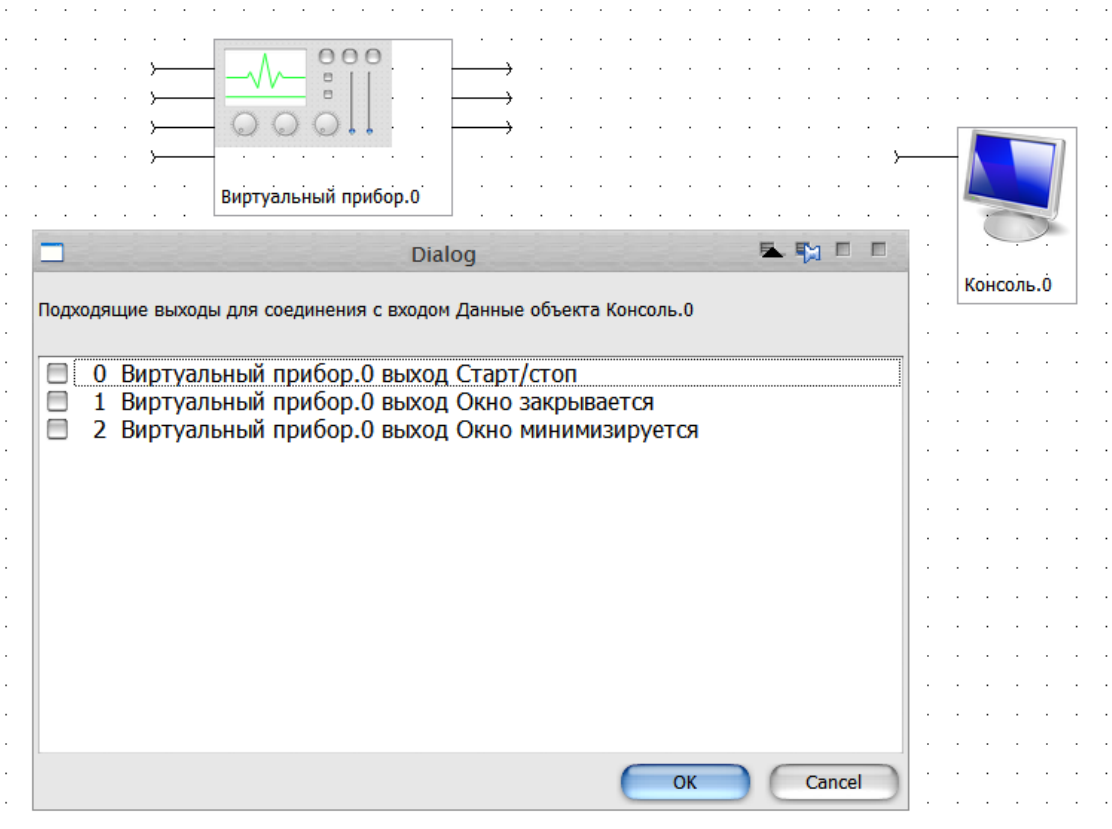

Рис. 2

Соединенные элементы можно перетаскивать по схеме, при этом линии соединений будут автоматически прокладываться заново.

Перетаскиваемые элементы располагаются с привязкой к узлам сетки схемы. Также по этой сетке накладываются изображения разъемов, и прокладываются соединители. Отменить это нельзя, наличие узлов схемы обязательно для автоматического алгоритма прокладки соединителей. Можно изменить шаг сетки, сделав его вдвое более частым, это выполняется с помощью выбора пункта «Сетка мелкая» в правом верхнем углу редактора схемы. Такой режим

предназначен для мониторов с небольшим разрешением.

При перетаскивании элемента со схемы в любую группу палитры, он автоматически переносится в свою группу, его соединения разрываются, а если он был в «активном» состоянии, то он выключается.

При наведении на линию соединения, она подсвечивается, появляются флажки у входа и выхода, куда он подключен.

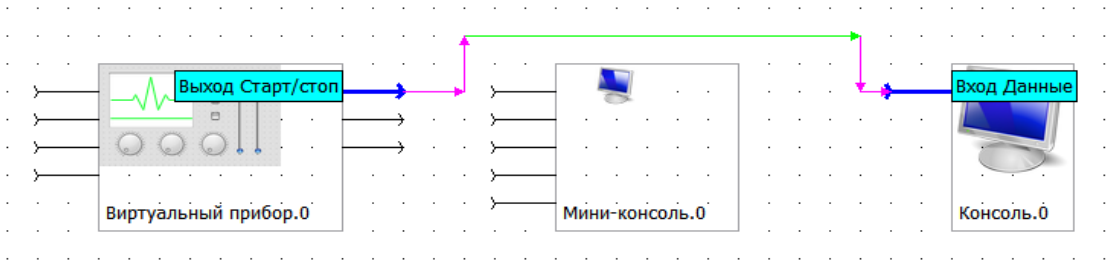

Рис. 3

Если отрезок прямо под курсором можно перетаскивать, он подсвечивается зеленым цветом, и тогда можно нажать левую кнопку мыши, и перетаскивать отрезок по схеме в перпендикулярном отрезку направлении. Если линия при этом пересекает элементы схемы, она становится красной, если в этот момент отпустить отрезок, линия вернется в свое исходное положение.

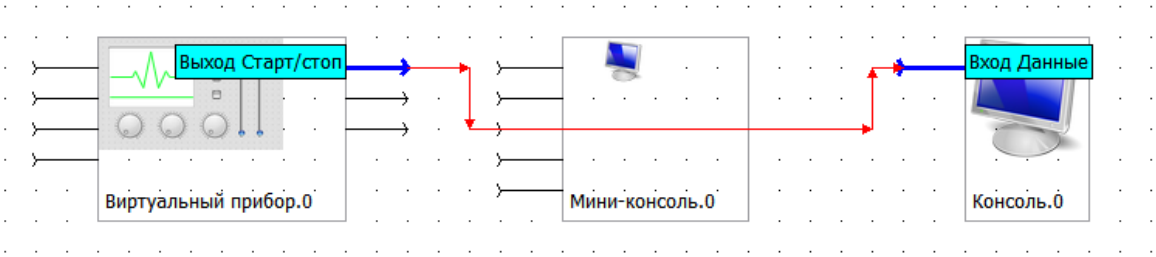

Рис. 4

Если линия не красная, то при отпускании она останется на новом месте, и будет на нём сохранена в схеме.

При нажатии на линию правой кнопки мыши, появляется локальное меню, позволяющее отключить линию, проложить заново, либо перейти к входу или к выходу элементов, которые она соединяет.

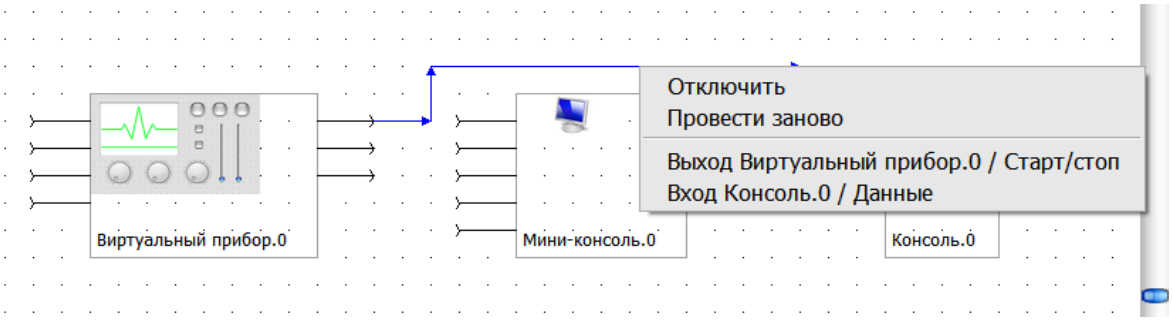

Рис. 5

Для удобства редактирования на схеме можно одновременно переносить несколько элементов. Также поддерживается копирование настроек между элементами, если у них есть одинаковые настройки с одинаковыми типами. Можно настроить один элемент, выбрать на схеме такие же, и в два клика мышки скопировать в них все настройки.

Редактор схемы поддерживает отмену и повтор выполненных изменений схемы (undo/redo).

Создание интерфейса виртуального прибора

Как правило, создаваемый виртуальный прибор имеет собственный пользовательский интерфейс. Для его создания предназначены элемент «Виртуальный прибор», находящийся в группе «Системные», и группа «Элементы интерфейса». В настоящее время в ней 16 стандартных типов элементов, но их состав может быть расширен при необходимости. При перетаскивании «Виртуального прибора» на схему, на экране открывается окно прибора, на котором можно располагать элементы интерфейса. После этого можно перетаскивать на схему эти элементы из палитры, и они появятся на окне прибора. Это можно увидеть в видео-примере record_000001.avi — в нём на схему перетаскивается элемент интерфейса «Кнопка», при этом автоматически на схему переносится и открывается окно виртуального прибора с этой кнопкой.

По-умолчанию окно виртуального прибора открывается в режиме редактирования. В этом режиме на окне отображается сетка, размеры окна можно менять, перетаскивая его границы. Элементы интерфейса можно располагать на окне прибора, перетаскивая их по узлам сетки с помощью мыши, с помощью колёсика мыши можно легко менять их размеры, а двойным кликом на любой элемент вызывается диалог его настроек, где можно менять многие параметры, специфичные для каждого элемента.

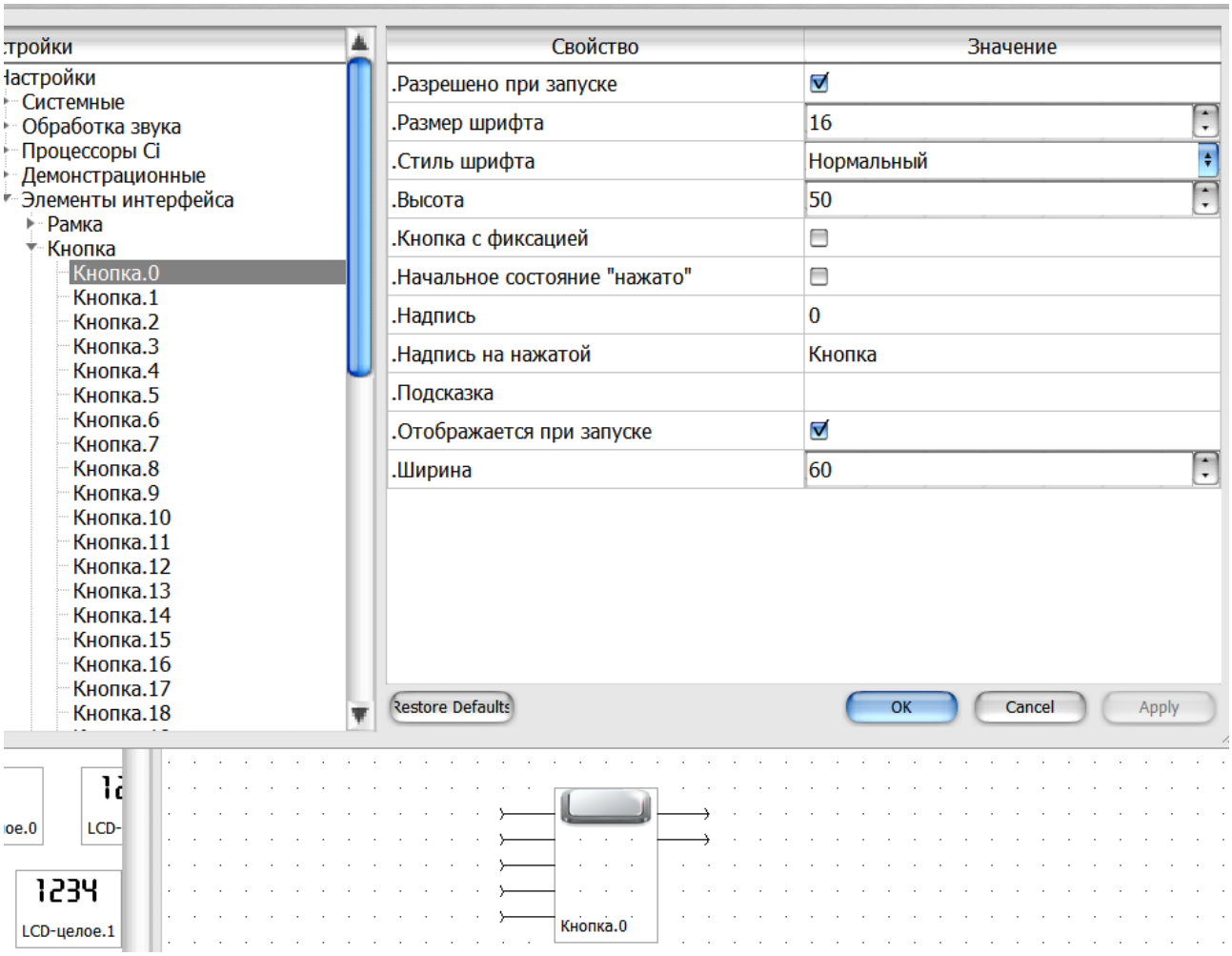

Рис. 6

При нажатии правой кнопки мыши в любом свободном месте окна, появится локальное меню, где с помощью пункта «Использовать», можно перевести окно в рабочий режим. В этом режиме изменение элементов на окне невозможно, но они начинают работать, выполняя свои

основные функции. Это также можно увидеть в примере record_000001.avi после 00:33. Снова перевести окно в режим редактирования можно аналогичным нажатием правой кнопки на свободном месте, и выбором пункта «Редактировать».

Параметры интерфейса виртуального прибора сохраняются при сохранении его схемы. При последующем открытии этой схемы, виртуальный прибор появится в том состоянии, в котором он был сохранен, и редактирование может быть продолжено. Но для каждого элемента интерфейса виртуального прибора сохранение его состояния управляется в отдельности.

Ядро ПО обеспечивает автоматическое сохранение и восстановление окна прибора при завершении его работы и последующем запуске также и в режиме «использования». То же касается состояния элементов управления, если разработчик прибора задал такой режим в настройках элементов.

В режиме редактирования интерфейса виртуального прибора поддерживается отмена и повтор изменений (undo/redo). Но это работает только для изменения места и размеров управляющих элементов на интерфейсе. Изменение других настроек элементов пока не реализовано.

В настоящее время все необходимые функции редактирования схемы прибора еще не реализованы. Требуется разработка некоторых нужных для удобного редактирования функций, но это исключительно техническая задача, требующая некоторого времени.

Виртуальные процессоры

Особенности языка Ci

Для программирования алгоритмов, выполняемых виртуальными процессорами, используется императивный алгоритмический язык Ci (от C interface), который был разработан специально для быстрого освоения персоналом, не имеющим образования системных программистов. Этот язык имеет предельно упрощенную семантику, и базируется на тех же идеях, которые лежат в основе языка ЛОГО, используемого для обучения алгоритмическому программированию в российских средних школах. Для пользователя он не является объект-ориентированным языком, что существенно облегчает его использование. От пользователя не требуется знания методик ООП, понимания механизмов наследования, полиморфизма и инкапсуляции, знания специальных синтаксических обозначений и способов решения множества сложных специфичных для ООП проблем. В то же время, под простыми в применении функциями языка скрыт мощный объект-ориентированный механизм, позволяющий наращивать язык новыми функциями и операторами без изменения исходных текстов его компилятора и исполняющей системы. Язык изначально разработан, как расширяемый, а текущая реализация позволяет это делать максимально удобно — расширения реализуются в виде динамически загружаемых библиотек, аналогично плагинам основной части ПО MAX15. Такие загружаемые модули могут содержать реализации функций для управления различным оборудованием, программного управления основными плагинами, новые типы данных (в том числе, структурные) и многое другое. Это означает, что если в разрабатываемое ПО необходимо добавить поддержку программного управления каким-то специфичным оборудованием, то не требуется даже пересобирать ни ядро, ни какой-либо другой компонент приложения — необходимо реализовать функции управления по определенным для языка Ci правилам, и положить полученную динамическую библиотеку в каталог для модулей расширения языка Ci. При запуске ПО загрузит эту библиотеку, и функции

управления новым оборудованием можно будет сразу использовать в Ci-программах. Они будут доступны везде — для компилятора, в функциональном редакторе, в отладчике.

Процессор языка реализован в виде отдельного компилятора и исполняющей системы. Компилятор создает так называемый «шитый код», который является позиционно- и машинно-независимым промежуточным кодом. Этот код исполняется виртуальной машиной, написанной на языке Си, что обеспечивает очень высокую скорость исполнения и полную переносимость. Виртуальные машины очень компактны, поэтому их в разрабатываемом ПО может быть ровно столько, сколько требуется виртуальных процессоров для разрабатываемого прибора. Они работают параллельно, одновременно и независимо друг от друга, давая таким образом, возможность *параллельного программирования* в рамках этого ПО. Как видно из следующего раздела, виртуальные машины скрыты от разработчика, использующего ПО. Это даёт возможность людям, не имеющим опыта параллельного программирования, создавать многозадачные алгоритмы, решая таким образом сложные технические задачи.

В настоящее время базовая реализация языка Ci имеет 6 базовых типов данных и около 120-ти операций в 8-и группах:

- логические
- управление данными
- арифметические
- поразрядные
- математические
- строковые
- управление алгоритмом
- помощь при отладке.

В языке Ci реализована редкая возможность писать программы не только в англоязычной нотации, но и с использованием слов из национального языка. Для этого в компилятор может быть загружена таблица «псевдонимов», которая создается, как текстовый файл в любом внешнем текстовом редакторе. Каждой операции, кроме её оригинального англоязычного названия, придаётся название на национальном языке, которое компилятор воспринимает идентично англоязычному.

Язык Ci прошел серьезную апробацию при использовании в двух промышленных автоматизированных системах контроля — БЛИК-120/240ЦС (включая более поздние системы на его основе) и серии АСКД СКИП модификаций П, У и С. В обеих этих системах на языке Ci решались по сути те же задачи, которые надо будет решать в новом разрабатываемом ПО. Использование этого языка позволило быстро создать программы автоматизированного контроля для различных узлов электронной аппаратуры — как цифровых, так и аналоговых, с использованием различных измерительных приборов, в том числе по стандартным интерфейсам. Язык быстро осваивали радио-инженеры, не имевшие специализации программистов, и при этом за короткий срок создавали рабочие программы контроля, прошедшие в том числе гос-приёмку. Как показал практический опыт, человек с базовым образованием информатики по программе российской средней школы, когда-либо изучавший хотя бы один язык алгоритмического программирования, получает первые самостоятельные результаты уже на второй день после начала изучения языка Ci, через 10 дней уверенно пишет несложные рабочие программы в несколько сотен строк, а через месяц владеет этим языком полностью и способен создавать программные продукты в тысячи строк. Когда язык Ci было предложено использовать профессиональному программисту с дипломом математического факультета университета и большим практическим опытом программирования на разных языках, ему было достаточно один раз прочесть описание языка, и он сразу написал сложную рабочую программу. То есть,

для профессионала обучение Ci вообще не требуется. При этом абсолютно все, кто использовали язык Ci, осваивали его в режиме самообучения, без участия в каких-либо курсах или семинарах, исключительно с использованием документации и инструментов для создания программ. Эти результаты показывают высокую эффективность языка Ci для решения тех задач, для которых предназначено разрабатываемое ПО.

Создание программ

Для создания программ виртуальных процессоров в базовый комплект ПО входит специализированный редактор исходных текстов. При выборе пункта «Редактировать» в локальном меню виртуального процессора открывается окно редактора с закладкой, имя которой совпадает с именем процессора. Программа привязана к процессору по имени, которое можно удобно задавать различными способами.

Редактор имеет практически все необходимые для удобного создания программ функции, такие как

- копирование и вставку фрагментов,
- отмену и повтор действий,
- поиск и замену, в том числе по всем открытым файлам
- быстрое перемещение по последним позициям курсора
- возможность редактирования нескольких текстов в отдельных закладках,
- нумерацию строк,
- автоотступ вложенных блоков,
- подсветку синтаксиса,
- автодополнение ключевых слов
- быструю справку по функциям.

Также редактор поддерживает создание программ из «проектов». Проект — это список исходных текстов программ, которые компилируются в порядке, указанном в проекте, и создают один выполняемый файл для виртуального процессора. То есть, проект — это простейший способ организации групповой работы нескольких человек над одной программой, если такой режим требуется для решения задачи.

Как было сказано ранее, виртуальные процессоры работают параллельно. Это демонстрируется в видео-примере record_000004.avi. В нём два виртуальных процессора запускаются одновременно, один выдаёт на свой 1-й выход текстовое сообщение каждые 200 мсек, другой работает аналогично, но через каждую 1 секунду. Выходы обоих подключены к консоли, и на ней видно, как сообщения асинхронно появляются через указанные интервалы времени.

Необходимо отметить, что при параллельном выполнении программ атомарным является один отправленный сигнал, или один цельный объект. Им может быть текстовая строка, вещественное или целое число, блок данных и т.д. Атомарным — значит неделимым. Это означает, что увидеть в консоли вывод типа «OuO ut1 2» невозможно. Следующий сигнал будет принят только после того, как будет принят и обработан предыдущий, но не во время этого. Гарантия целостности данных существенно облегчает создание параллельно работающих программ.

В видео-фрагменте record_000005.avi показано, как независимые процессоры могут быть соединены друг за другом в цепочку. Один процессор, как и в предыдущем примере, выдает на выход сообщение каждую 1 секунду. Другой процессор принимает сообщение, преобразовывает текст к верхнему регистру, и выдаёт на выход. Первоначально выходы обоих подключены к консоли, в которой видно исходное и преобразованное сообщения. Затем выход исходного сообщения отключен от консоли (можно было и не останавливать выполнение), и в ней виден

только выход преобразователя. Несмотря, на то, что это простой пример, он показывает возможность создания последовательных «фильтров» для различных данных, требующих какой-то сложной обработки.

Отладка

Очень часто написанная программа ведет себя совсем не так, как ожидалось, особенно если её писал не профессиональный программист. Поэтому, кроме средств для создания программ необходимо иметь средства для их отладки. В настоящее время технологии отладки хорошо развиты, хорошо известно, какие средства надо для этого иметь. Современным решением является интерактивный отладчик, обладающий рядом обязательных функций. Такой отладчик имеется в составе ПО MAX15. Как и другие отладчики для различных языков программирования, здесь отладчик позволяет:

- видеть исходный код программы, с оригинальными именами переменных, с сохранением структуры и отступов вложенных блоков
- назначать на любые операции точки останова, на которых выполнение программы приостанавливается, давая возможность просмотра и изменения текущего состояния переменных
- продолжать автоматическое выполнение программы после точки останова
- проходить программу пошагово, инструкция за инструкцией, наблюдая доступные величины
- заходить внутрь вложенных блоков, либо выполнять их целиком, без захода внутрь
- выполнять программу до выхода из блока, с последующей остановкой
- выполнять программу до любой операции, без установки на ней точки останова.

Но так как программа в разрабатываемом ПО выполняется виртуальной машиной, это позволило реализовать в отладчике некоторые уникальные возможностей, отсутствующие в аналогичных распространенных средствах отладки:

- просмотр и изменение промежуточных значений, полученных после возврата из операции, и перед входом в другую операцию
- замена англоязычных названий операций на названия на национальном языке
- вставка автоматических комментариев, помогающих понять логику работы алгоритма
- пошаговая трассировка, то есть пошаговое выполнение программы с визуализацией выполняемых действий в коде программы.

Последняя возможность очень эффективна не только для отладки, но и вообще для изучения программирования. Её наличие положительно скажется на простоте использования ПО MAX15 людьми, не являющимися специалистами в программировании. Видео-фрагмент record_000006.avi (внимание, фрагмент содержит звуковое сопровождение) показывает работу в режиме пошаговой трассировки.

Предпоследняя возможность (вставка автоматических комментариев) — тоже уникальное средство, способствующее изучению имеющихся средств программирования, и упрощающее понимание алгоритма. На рисунке приведен исходный текст, как он выглядит в редакторе программы:

```
int sample;
uint threshold 10000000, lednumber 1;
loop
{
    switch (wait sample)
    {
        if( (abs sample) > threshold )
        {
            send (concat "Данные " lednumber) FALSE;
            lednumber = (? (lednumber == 16) 1 (++ lednumber));
            uint colors (torgb
                            (random 200 255)
                            (random 200 255)
                            (random 200 255) );
            textlabelSetOnColor (lednumber - 1) colors ;
            send (concat "Данные " lednumber) TRUE;
        }
    }
    {
        threshold = sample;
    };
}
```

Рис. 7

А на следующем рисунке эта же программа в отладчике, без вставки автоматических комментариев:

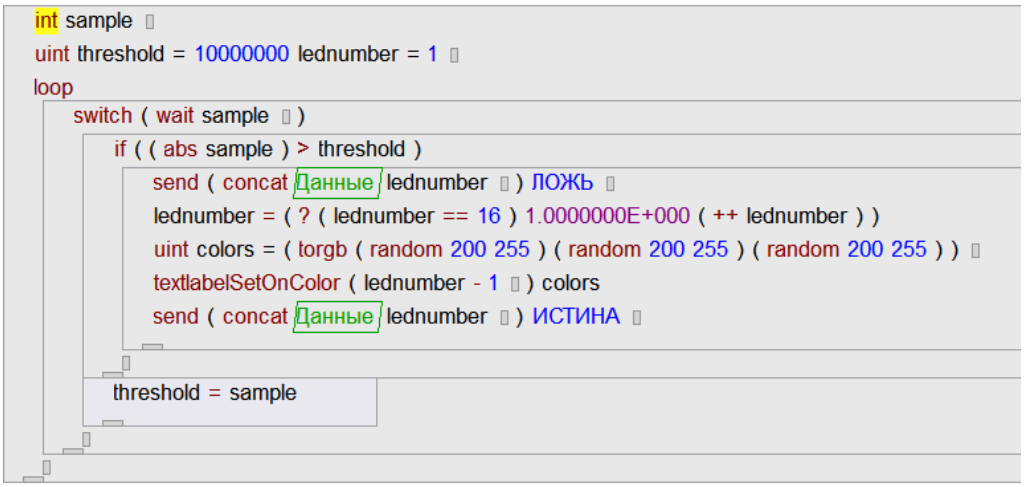

Рис. 8

Алгоритм здесь отображается также, как он был написан в редакторе исходного текста. Но после включения режима автоматического комментирования (это пока можно сделать только в настройках, потом будет флаг наверху окна) текст будет выглядеть так:

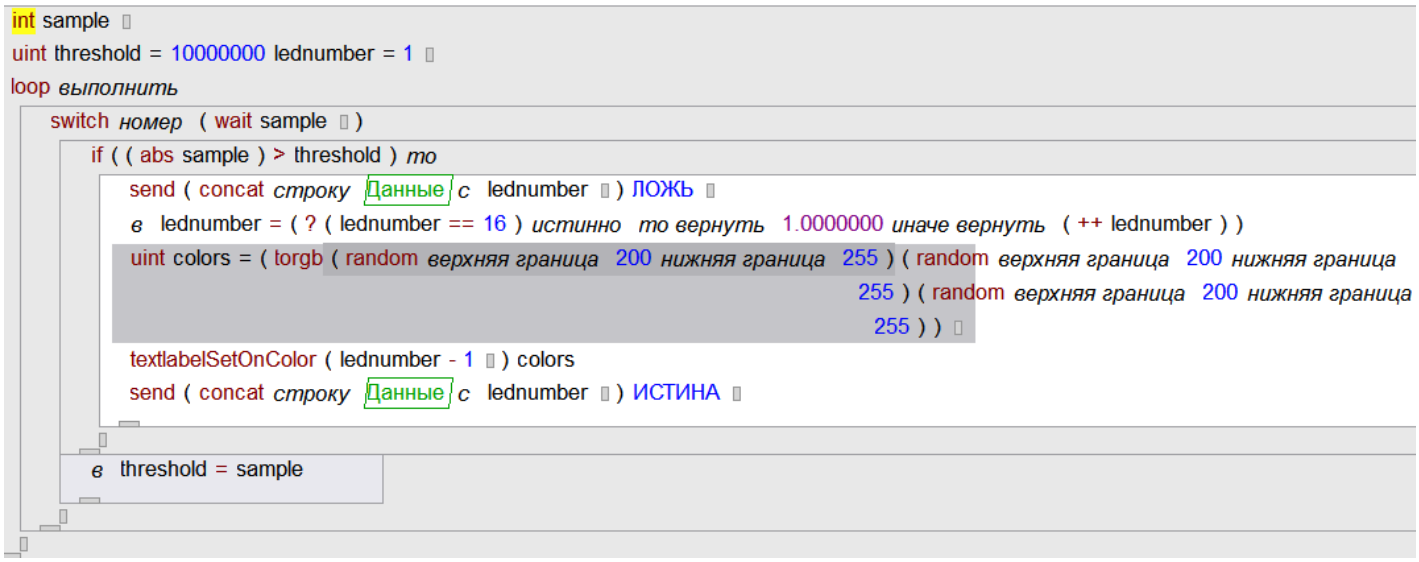

Рис. 9

Как видно из рисунка, не все операции снабжены автоматическими комментариями (на рисунке они написаны наклонным шрифтом). Комментарии задаются в отдельном внешнем текстовом файле, который доступен для редактирования в любом текстовом редакторе. Это позволяет добавить любые комментарии к любым операциям, в том числе, сделать комментарии на любом языке.

Еще одна уникальная возможность отладчика — замена имен операций на написанные на национальном языке. Она согласуется с тем, что программа в редакторе исходных текстов не обязательно должна быть написана в англоязычной нотации (см. ранее «Особенности языка Ci»). При включении в настройках отладчика отображения псевдонимов операций исходный текст будет выглядеть так:

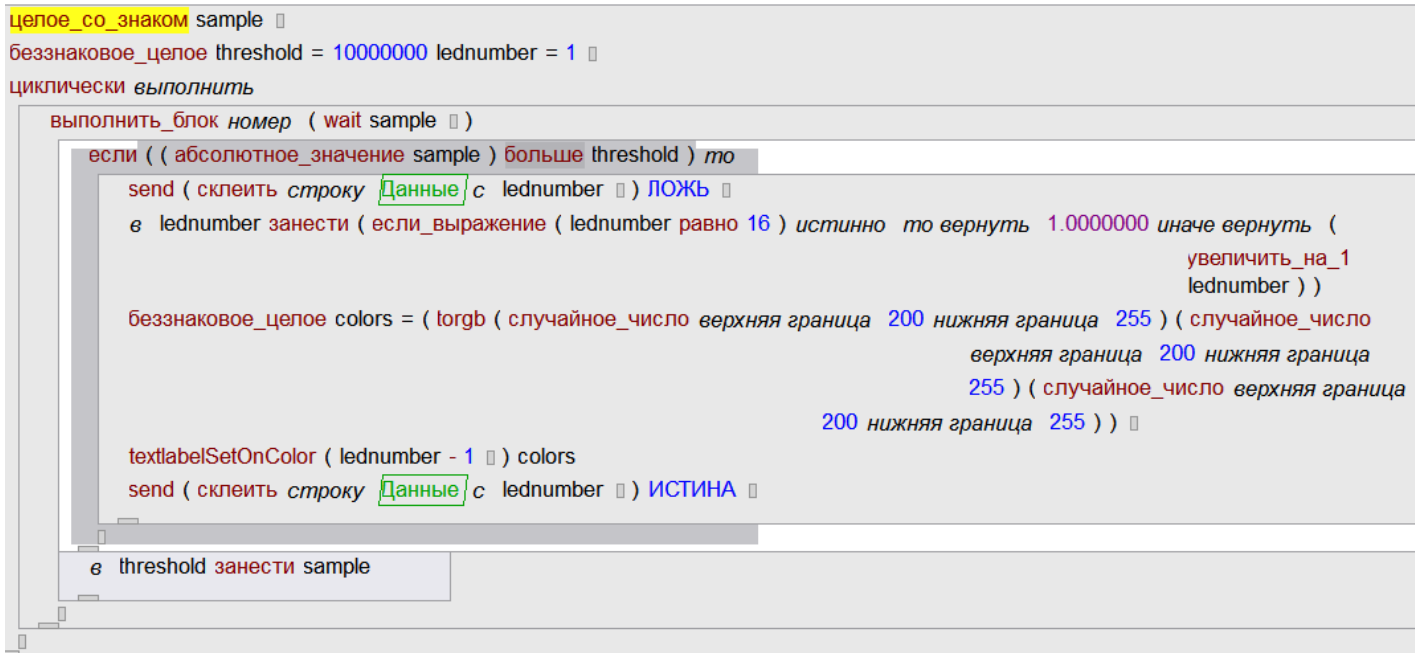

Рис. 10

Как видно из рисунка, основная часть алгоритма теперь написана практически на русском языке (переменные тоже могут иметь русские имена, но это надо учесть при их создании). Для русскоязычных пользователей, особенно не являющихся профессиональными программистами, это может существенно упростить понимание алгоритма и ускорить поиск возможных

ошибок. Некоторые операции (например, **wait** в четвертой сверху строке) остались англоязычными — просто потому, что в файле, где задаются псевдонимы, для них они пока не заданы. При поставке ПО MAX15 это может быть сделано за несколько минут.

Кроме наблюдения за алгоритмом, просмотра и изменения значений переменных, отладчик позволяет изменять и саму программу, то есть, он является также и редактором. В текущей версии это экспериментальная возможность, которая до конца еще не проработана. Возможно, что её в поставочном варианте ПО следует отключить, но при этом продолжить её разработку, и в дальнейшем сделать удобный инструмент для альтернативного способа создания программ. В отличие от набора текста в традиционном редакторе, этот способ будет позволять «собирать» программу из доступных элементов, перетаскивая их мышью из палитры слева. Это значительно снизит количество возможных ошибок и ускорит создание программ непрофессиональными программистами. Возможно, в дальнейшем отладчик станет основным средством для создания программ, более мощным, чем текстовый редактор

Наконец, отладчик имеет встроенный реверсный компилятор, переводящий скомпилированную ранее программу обратно в исходный текст на языке Ci. Это средство может быть востребовано в исключительных случаях, при утере пользователями исходных текстов своих программ.

Если разрабатываемая схема содержит несколько процессоров языка Ci, то для любого из них можно открыть отдельный отладчик. Все отладчики независимы друг от друга, выполняемые ими программы также работают параллельно, как и в случае обычного выполнения. То есть, можно отлаживать параллельное выполнение программ в двух или более отладчиках одновременно. Либо закрыть любой отладчик, тогда программа соответствующего процессора будет выполняться обычным способом, но любую другую можно наблюдать в отладчике.

Пример цикла создания программы

В видео-фрагментах record_000007 с номерами через тире с 1 по 8 показан процесс создания с нуля простого прибора, реализующего базовые действия булевой алгебры. Фрагменты надо просматривать последовательно. Для плеера Media Player Classic прилагается плейлист record_000007.mpcpl, позволяющий просмотреть весь процесс создания без ручного запуска фрагментов.

В демонстрации намерено делаются некоторые ошибки, которые потом исправляются[3]. Основное назначение фрагмента — показать насколько просто с помощью MAX15 создавать и модифицировать законченные приложения. В конце демонстрации MAX15 запускается без модуля компилятора, с установленным автоматическим запуском последней загруженной схемы — это выглядит, как запуск автономного приложения с булевой алгеброй. Задержки при запуске обусловлены тем, что параллельно работает видео-запись, влияющая на работу приложений. В реальности приложение запускается значительно быстрее.

Пример сетевого взаимодействия

В видео-фрагменте record_000008 показан пример взаимодействия программ в среде MAX15, работающих в разных ОС на разных компьютерах. В начале видео появляется экран ОС Windows XP с работющим MAX15. Затем на нём появляется окно системы дистанционного

3 В конце одного из фрагментов можно заметить, что изменилось соединение одного из индикаторов. Это была ошибка в исходном тексте редактора схемы, которая была немедленно исправлена.

управления вторым компьютером, на котором работает 64-х разрядная ОС Linux Kubuntu 14. Там тоже запущено ПО MAX15. В на обоих вариантах загружена одинаковая схема, в которой используется плагин для передачи сигналов между компьютерами по протоколу UDP. Плагины настроены таким образом, что компьютеры передают друг другу сигналы, в двунаправленном режиме. Нажатие кнопки на любом из них вызывает появление эхо в консоли другого. В настоящий момент атомарными передаваемыми данными могут быть — логическое значение, целое число со знаком, целое число без знака, вещественное число двойной точности и текстовая строка виртуально неограниченной длины. Это простейшее средство для создания кластеров из множества компьютеров на различных ОС, параллельно работающих над решением одной задачи. При необходимости плагин для сетевого взаимодействия может быть легко расширен дополнительными поддерживаемыми форматами данных, добавлена поддержка TCP/IP протокола, автоматизирована настройка.

Конфигурирование для поставки

Если пользователь ПО MAX15 сам является разработчиком, и с помощью этого ПО создаёт прикладное решение специфичной задачи для своего Заказчика, то очевидно, что в его интересах не поставлять далее своему Заказчику использованные средства разработки и исходные тексты. Это учтено в конструкции ПО — компилятор выделен в отдельную динамическую библиотеку, то же самое касается редактора исходных текстов и отладчика. Достаточно не копировать соответствующие файлы при передаче готового решения, и далее не будет возможности получить исходные тексты программ, модифицировать их или создавать новые. Получатели не смогут также создавать новые выполнимые программы. Будет возможность только их использовать. Для демонстрации такого «комплекта поставки» служит готовое приложение «Калькулятор». Оно находится в каталоге calc. Содержимое этого каталога надо распаковать на свободный носитель в любое место. Установка не требуется. В каталоге calc надо запустить программу MAX15.exe – на самом деле, это ядро ПО MAX15. Оно настроено на автоматическую загрузку и запуск приложения, реализующего функции простого калькулятора с четырьмя арифметическими действиями. В файле screen.JPG (лежит в этом же каталоге) находится картинка схемы этого калькулятора. А в файле «процессор калькулятора.0.ci» исходный текст программы, выполняемой процессором Ci. При обычной поставке исходный текст может не передаваться. Кроме настроек использованных элементов, каталог calc содержит всё, что должен передать разработчик при создании аналогичной программы.

В настройках ядра предусмотрено задание каталогов для хранения настроек плагинов. Это необходимо, если поставляемое приложение не будет иметь возможности сохранять настройки в тех же каталогах, где оно записано. Такой механизм позволяет легко создавать как «инсталлируемые» приложения, настройки которых специфичны для разных пользователей, так и «мобильные» приложения, настройки которых путешествуют вместе с приложениями на картах памяти.

В планах текущей разработки создание «генератора приложений». Это будет модуль, который будет копировать только необходимые файлы в выбираемый разработчиком приложения каталог. Сейчас для создания приложения калькулятора потребовалось около 30 минут и активные «ручные» действия по созданию каталогов и копированию файлов. Потребовалось также изменить содержимое некоторых текстовых файлов. В генераторе приложений нужно будет

только указать каталог для создаваемого приложения, и нажать кнопку «Создать» - все необходимые действия будут выполнены автоматически за время значительно меньшее. Для Калькулятора потребовалось бы около половины секунды.

Сравнение с LabVIEW

Непосредственные аналоги ПО MAX15 на настоящий момент времени не известны. Существуют несколько программных систем, использующих блочные схемы для создания приложений. По ряду выполняемых функций и по назначению наиболее близким функциональным аналогом является National Instruments LabVIEW. Поэтому интересно рассмотреть что общего, и в чем различия у этого широко известного программного пакета, и у создаваемого в рамках данной работы ПО.

Общее

Классы решаемых задач

С помощью ПО MAX15 можно решать те же задачи, на которые рассчитана система NI LabVIEW. Вот как позиционируется LabVIEW на сайте National Instruments: “LabVIEW is a highly productive development environment for creating custom applications that interact with real-world data or signals in fields such as science and engineering” — «LabVIEW это высоко продуктивная среда разработки для создания собственных приложений, которые взаимодействуют с данными или сигналами реального мира в таких сферах, как наука и инженерия». Про разрабатываемое ПО можно сказать в точности то же самое. Причем степень продуктивности гораздо выше, благодаря значительно более простым в использовании элементам и приёмам.

Работа в различных ОС

Как и NI LabVIEW, разрабатываемое ПО может функционировать в различных операционных системах, поддерживаемые платформы почти полностью совпадают.

Открытая архитектура

NI LabVIEW имеет открытую архитектуру, что позволяет сторонним производителям создавать программные модули, которые могут использоваться в LabVIEW. Аналогично, предполагается, что будет выпущена документация на разрабатываемое ПО, которая позволит всем желающим создавать динамически загружаемые модули, которые ядро будет «понимать» и с которыми будут корректно работать остальные модули.

Различия

Области применения

Несмотря на то, что разрабатываемое ПО MAX15 и NI LabVIEW имеют очень много общего в классах решаемых задач, конкретные области применения у них полностью не совпадают. Как видно из раздела «Назначение», первичной сферой применения ПО MAX15 является

автоматизированное управление при промышленных применениях. Изначально LabVIEW на это не было рассчитано, и хотя NI много лет пытается внедрить LabVIEW в эту область, это до сих пор не случилось. В то же время, здесь при разработке учитывается это применение, поэтому разрабатываемое ПО имеет перед LabVIEW очевидное преимущество. Гарантией применения для автоматизированного контроля является то, что в ПО MAX15 будут входить модули, традиционные для систем автоматизированного контроля — это модули для взаимодействия с различными СУБД и модуль генератора отчетов. А добавление в дальнейшем модуля нечеткой логики позволит создавать на основе ПО MAX15 мощные средства диагностики и прогнозирования аварийных ситуаций.

Разница парадигм

И NI LabVIEW, и разрабатываемое ПО имеют средства для программного управления обработкой данных. В обеих системах такая обработка может быть параллельной. Но реализовано она существенно по-разному, и на этом следует подробно остановиться.

Самое очевидное и существенное различие — это то, что в NI LabVIEW для создания программ используется графическое средство программирования «G», а в создаваемом ПО — классический императивный алгоритмический язык Ci. Необходимо отметить, что в LabVIEW для программирования алгоритмов можно использовать вариант языка Basic, но этот язык не является основным средством создания программ непрофессиональными программистами. Разработчики LabVIEW тем самым признали убогость этого языка, появившегося в 60-е годы 20-го века, и распростившегося исключительно благодаря личным пристрастиям владельца компании MicroSoft Била Гейтса. National Instruments продвигает G, как основное средство.

Компания National Instruments в качестве основного достоинства языка G преподносит то, что этот язык могут освоить и использовать те, кто вообще не изучал программирование. Но на самом деле, это достоинство является главным недостатком этого языка, поскольку программирование сейчас изучают в средней школе, и это не язык G. В российской средней школе для изучения алгоритмического программирования используется язык ЛОГО, созданный в 90-х годах 20-го века Геннадием Лебедевым и Анатолием Кушниренко. Таким образом, при изучении языка G у российских специалистов возникает определенный когнитивный диссонанс, поскольку свои знания о программировании для создания программ на G они использовать не могут. Простой пример: сложение и умножение двух чисел на классическом языке будет записано примерно так:

```
int a, b, sum, mul; sum = a + b; mul = a * b;
```

при этом вычисления sum и mul выполняются последовательно. Результат mul не появится раньше результата sum. Но на языке G это будет такая диаграмма:

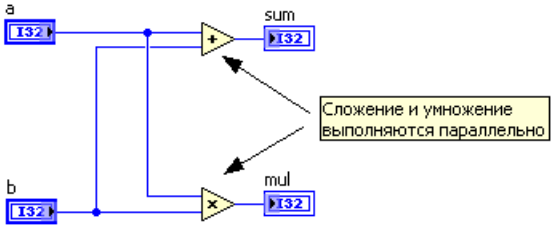

Рис. 11

то есть, mul может быть вычислено раньше, либо позже sum. На самом деле, это принципиальная разница, поскольку при использовании G разработчик должен думать совершенно иначе. Получается, что для использования языка G знание классического алгоритмического программирования оказывается бесполезно. То есть, изучающий язык G человек, даже будучи специалистом в области программирования, оказывается в тех же условиях, что и человек, никогда программирование не изучавший. А это, в свою очередь, означает что бесполезно были затрачены усилия преподавателей, потеряно время, затраченное на обучение традиционному программированию. И требуется еще время, новые затраты на обучение. Причем, как показывает опыт, в случае языка G суммарное время на обучение является очень значительным. Первые результаты ученики получают довольно быстро, но проблемы начинаются, когда они сталкиваются с необходимостью модификации программ, или программы работают совсем не так, как ожидалось. Тем более, что «синтаксис» языка G нельзя назвать «интуитивно понятным» - это набор большого количества идеограмм, семантический смысл большинства которых ни с чем в реальности не связан. В некотором смысле, язык G – это «религия», поскольку специалист, его изучивший, но не изучивший обычное программирование, сможет пользоваться только G. И это **главный недостаток** LabVIEW.

Складывается впечатление, что процессоры языка Ci совпадают с узлом **Формула**, имеющимся в LabVIEW. Но это не так. Узел Формула по доступной функциональности принципиально уступает процессору Ci. Процессор Ci может решать те же задачи, которые решает узел Формула, но не наоборот. Программа на Ci – это полноценная программа, которая может содержать объявления и вызовы функций (в том числе, вложенные объявления), ожидание и обработку входящих сигналов, вызовы операций для управления внешним оборудованием и многое другое. Можно также добавлять в Ci новые типы данных, в том числе сложные, а также объявлять новые константы. Благодаря модульному расширению языка Ci, его функциональность виртуально не ограничена. Тогда как узел Формула предназначен для ограниченных вычислений, и не может быть расширен.

Для профессионального использования G требуется значительное время на обучение. Но если для создания прикладных программ используется традиционный алгоритмический язык, российские разработчики могут приступить к его использованию практически сразу, им только требуется изучить новый синтаксис и семантику. Именно поэтому был разработан язык Ci, который имеет с «школьным» языком ЛОГО почти одинаковый синтаксис и очень похожую семантику. Пара примеров одних и тех же программ на ЛОГО и на языке Ci, если к нему подключить операции управления «черепашкой» (стандартное рисующее средство языка ЛОГО).

ЛОГО	Ci
поднять перо место 50, 300 цвет 10 перо 3 опустить перо повторить 10 {	перо_поднято ИСТИНА; место 50, 300; цвет 10; перо 3; перо_поднято ЛОЖЬ; цикл {

повторить 7 {
вперед 70
направо 90
}
направо 90
}
	цикл {
	вперед 70;
	направо 90;
	} 7;
	направо 90;
	} 10;

переменная z
переменная t
переменная x
переменная y
переменная k
переменная m
переменная s
 	
спрятать черепаху
 	
s = 0.001 * pi
 	
повторить для z = 0 до 100000 {
t = t + s
x = Sin(0.99 * t) - 0.7 * Cos(3.01 * t)
y = Cos(1.01 * t) + 0.1 * Sin(15.03 * t)
k = 200 * x + 400
m = 200 * y + 300
точка k, m
}
	вещественное z, t, x, y, k, m, s;
	
	черепаха_видна ЛОЖЬ;
	
	s = 0.001 * Pi;
	
	для z 0 100000 1 {
	t = t + s;
	x = ((sin (0.99 * t)) - (0.7 * (cos (3.01 * t))));
	y = ((cos (1.01 * t)) + (0.1 * (sin(15.03 * t))));
	k = ((200 * x) + 400);
	m = ((200 * y) + 300);
	точка k, m;
	}

Таблица 1

Таким образом, по сравнению NI LabVIEW разрабатываемое ПО имеет следующее преимущество: **значительная экономия времени, сил и средств при обучении использованию, благодаря применению традиционных методик, для которых обучаемые уже имеют предварительную подготовку**.

Еще очень существенными различиями у программ на традиционных языках, по сравнению с программами на языке G, являются **компактность** и **скорость создания**. Эти параметры тесно связаны друг с другом. Как видно из рисунка 11 и выражения на Ci немного выше него — для получения одного и того же конечного результата для программы на Ci требуется гораздо меньше рабочего пространства, чем для программы на G. И одновременно, для создания такой программы надо значительно меньше времени, включая время на обдумывание. Если кто-то сомневается в этом, он может попробовать сделать те же самые действия, что приведены в таблице 1, средствами языка G. Для написания второго примера на языке Ci требуется около 4-х минут при непрофессиональном вводе по 1-му символу в секунду, и примерно столько же для обдумывания решения. При этом последовательность выполнения действий естественна и интуитивна. Желающие могут попробовать решить эту задачу на G, с учетом перетаскивания мышью изображений всех компонент и задания значений констант. И не забыть при этом про обязательную последовательность выполнения.

На самом деле, решения на языке G имеют те же недостатки, что и программы на языке Basic – их очень сложно модифицировать и развивать при изменении требований. По отзывам специалистов, использовавших в реальной работе язык G, программы на нём очень плохо структурируются, и с некоторого момента становятся похожи на снежный ком, в который становится почти невозможно добавлять новую логику. Часто решение на G занимает пространство в несколько экранов компьютера, при этом его очень сложно анализировать и модифицировать.

В то же время, при реальной разработке программ контроля для АСКД СКИП пять человек за два года написали на языке Ci более 100 программ со средней длиной около 1000 строк, в среднем по 3 операции (в том числе, арифметические) на строку. Это порядка 300000 операций, или по 60000 операций на человека за 2 года, 2500 операций в месяц, или около 125 операций в день. Это были отлаженные и проверенные программы, которые были сданы в учёт. Ни один из работавших не был профессиональным программистом. Все радио-инженеры, причем четверо из пяти только перед этим закончили технологический университет, то есть, не имели опыта практического программирования. Нет никаких сомнений, что такой темп разработки в подобных условиях на языке G невозможен.

Различия в наборах функций

Ниже приводится таблица, в левой колонке которой находится список функций NI LabVIEW, в правой колонке список соответствующих функций ПО MAX15. Функции NI LabVIEW взяты из книги Суранов А.Я «LabVIEW 8.20. Справочник по функциям», 2007 г.

Функции LabVIEW	Функции ПО MAX15	Оценка трудоёмкости доработки (человеко-дней)	
Структуры и массивы	В разрабатываемом ПО нет прямого эквивалента «структур» NI LabVIEW из-за различия парадигм. Их функции выполняют традиционные операторы цикла и ветвления, с использованием вложенных блоков кода. Также язык Ci имеет возможность создания и вызова именованных функций с возвратом значения и переменным числом параметров.	Доработка не требуется	0
	Массивы в языке Ci функционально аналогичны массивам LabVIEW. Но в Ci массивы одномерные. Есть возможность эмулировать двумерные массивы на одномерных с помощью специальных операций, если в этом возникнет необходимость	Добавление в библиотеки Ci эмулятора многомерных массивов на одномерных	5
	Кластеры NI LabVIEW являются функциональными аналогами гетерогенных структур в алгоритмических языках. В языке Ci такие структуры отсутствуют, но могут быть добавлены при необходимости. Это относительно сложная доработка, требующая времени. Однако за всю обширную практику использования Ci это ни разу не потребовалось.	Добавление в компилятор и исполняющую систему гетерогенных структур, доработка отладчика	40-60

Базовые функции и манипуляция данными (кроме файлов и баз данных)	Большая часть базовых функций и констант реализованы в языке Ci.	Реализация отсутствующих операций, около 80 (в среднем 1 чел/день на операцию)	80
Функции для работы с файлами и базами данных	В разрабатываемом ПО аналоги этих функций планируется реализовать принципиально иначе.	Разработка функций для работы с файлами и СУБД, включая генератор отчетов	60-80
Дополнительные функции — диалог и интерфейс	Большинство аналогичных функций в разрабатываемом ПО будут реализованы иначе. Сравнивать можно только комплексно. Соответствующий функционал сейчас реализован примерно на 40% по сравнению с LabVIEW.	Реализация различных функций для пользовательского интерфейса	10-20
Функции управления приложением	Запланированная реализация аналогов принципиально отличается от имеющейся в LabVIEW	Реализация функций для управления и взаимодействия с другими приложениями	20-40
Функции записи и воспроизведения звуковых сигналов	В настоящее время реализована демонстрационная функция ввода звука со стандартного устройства, непригодная для профессионального использования. Реализация профессионального набора функций требует иного подхода.	Реализация функций для ввода и вывода звуковых сигналов	20-30
Математические функции	В разрабатываемом приложении математические функции должны быть реализованы, как расширения языка Ci.	Реализация математических и статистических функций, около 110 (в среднем 1 чел/день на операцию)	110
Функции генерации и обработки сигналов	Обработка сигналов должна быть реализована в виде набора соответствующих плагинов	Реализация плагинов для обработки сигналов, около 100 (в среднем 1,5 человеко-дня на плагин)	150
Функции для работы с осциллограммами	Эти функции частично должны быть реализованы, как расширения языка Ci, но частично как	Реализация функций для обработки ос-	80

	плагины.	циллограмм, около 80 (в среднем 1 чел/день на функцию)	
Функции разделяемых переменных	В разрабатываемом ПО функционально аналогичный механизм будет реализован принципиально иначе — через механизм прозрачного межплатформенного взаимодействия, осуществляемого поверх TCP/IP соединений.	Разработка механизма межплатформенного взаимодействия	20-30
Функции поддержки приложений	Поддержка ActiveX и .NET не планируется, так как основной их платформой является ОС Windows. Для ПО MAX15 основной платформой предполагается *NIX семейство ОС, в которых ActiveX и .NET не используются. В отдельных случаях взаимодействие с ActiveX и .NET может быть реализовано при особых требованиях Заказчиков.		
Библиотеки с динамической компоновкой.	В разрабатываемом приложении библиотеки с динамической компоновкой должны быть подключены через переходные модули, написанные на языке C++ с использованием библиотеки Qt той же версии, которая используется самим приложением. Подключение сторонних библиотек без удовлетворения этих условий не планируется, хотя такая возможность может быть реализована при особых требованиях ов.		
Доступ к реестру Windows	В разрабатываемом приложении доступ к реестру Windows не планируется, поскольку основной платформой предполагается *NIX семейство ОС. В них реестр отсутствует. В отдельных случаях доступ к реестру может быть реализован при особых требованиях Заказчиков.		
Управление устройствами и портами ввода-выводв	Функции управления стандартными устройствами и ввод-вывод через физические порты могут быть реализованы при необходимости. Это можно сде-	Разработка функций управления устройствами и ввод-вывод через фи-	10

	лать как с помощью плагинов, так и расширением языка Ci	зические порты, около 10 (в среднем 1 чел/день на функцию)	
Функции ввода-вывода данных через стандартные интерфейсы	В разрабатываемом ПО планируется реализация функций для ввода-вывода данных через стандартные интерфейсы: VISA, RS-232/422/485, I^2C, GPIB. Данные функции будут реализовываться и как плагины, и как расширения языка Ci. Разработка функций для ввода-вывода через фирменный интерфейс National Instruments DAQmx не планируется, но он может быть реализован по специальному требованию Заказчиков.	Разработка функций для стандартных интерфейсов: VISA RS-232/422/485 I^2C GPIB *все*	 10 5 5 5 *25*
		Итого, человеко-дней	630-720

Таблица 2

Итого, для разработки всех функций нового приложения, которые сделали бы его практически полным эквивалентом NI LabVIEW требуется 3,2-3,6 человеко-года (20 дней в человеко-месяце с учетом выходных, 10 человеко-месяцев в году с учетом отпуска и длительных праздников). Но необходимо отметить, что во всех функциях сразу нет и не будет необходимости. Приложение можно использовать и распространять, а доработку функций производить по мере появления соответствующих требований. Как было сказано ранее, приложение специально так сконструировано, чтобы можно было заменять и добавлять отдельные модули без изменений как системной части приложения, так и других прикладных модулей. Большая часть работ может выполняться параллельно несколькими людьми. Например, функции ввода-вывода данных через параллельные интерфейсы можно одновременно разрабатывать в виде плагинов, и в виде расширений языка Ci. Это практически удвоит скорость их разработки.

Стратегия развития

Основные отрасли применения

Предполагается, что разрабатываемое ПО найдет применение в следующих отраслях хозяйственной деятельности:

Отрасли промышленности и науки	Использование	Положительный эффект
Оборонная	Автоматизация регулярных проверок ВВТ, находящейся на длительном хра-	Ускорение и повышение качества проверок, своевременное

	нении с целью выявления узлов, подлежащих замене.	обнаружение неисправных и потенциально неисправных узлов. Повышение боевой готовности сложной техники.
	Автоматизация проверок узлов электронной аппаратуры ВВТ при тестировании на предприятиях-изготовителях.	Ускорение проверок и повышение качества продукции предприятий оборонной промышленности.
Средства связи	Автоматизация контроля за состоянием действующих каналов связи.	Своевременное обнаружение отклонений в параметрах линий связи. Сокращение времени на восстановление качества линий связи.
	Проверка параметров оборудования связи.	Своевременная замена оборудования. Повышение качества услуг связи.
Автомобильная	Автоматизация проверок параметров автомобилей и состояния автомобильной электроники.	Быстрая настройка контрольных систем на различные модели автомобилей. Повышение безопасности движения.
Космическая	Автоматизация проверок электронных блоков и соединений перед отправкой в космос.	Сокращение времени подготовки. Повышение надежности электронных космических систем.
Авиационная	Автоматизация проверок бортовой авионики.	Сокращение сроков регулярного обслуживания. Повышение безопасности полётов.
Атомная	Автоматизация контроля за работой узлов АЭС. Независимое наблюдение за различными узлами.	Своевременное обнаружение неисправного оборудования. Повышение безопасности АЭС.
Нефтеперерабатывающая	Автоматизация тестирования оборудования нефтеперерабатывающей промышленности.	Своевременное обнаружение неисправного оборудования. Снижение числа аварий.
Наука	Научные исследования и эксперименты, особенно проводимые длительное время без участия человека.	Упрощение проведения экспериментов, автоматизация сбора и представления данных.
Образование	Обучение основам радио-электроники и системотехники. Одновременно закрепление навыков алгоритмического программирования.	Повышение качества образования.

Таблица 3

Разработка документации

Отдельная задача, которую необходимо решить — разработка документации. Хотя создаваемое ПО интуитивно понятно, без качественной документации его освоение займет больше времени. В настоящее время документация к подобными программным системам традиционно

поставляется в виде доступного из ПО контекстно-зависимого справочника с гипертекстовыми связями. Это основная форма документации, которую предполагается использовать и для ПО MAX15. В то же время, если и потребуют документацию в другой форме, например, в соответствии с ГОСТ 19 (ЕСПД), она может быть отдельно разработана.

Кроме традиционной документации, предполагается разработать специальные средства для обучения использованию ПО в практической деятельности.

Подробнее о разработке документации, с определением трудоёмкости, описано в трех следующих подразделах.

Разработка справочной системы

Под справочной системой понимается доступный в любой момент времени при использовании ПО контекстно-зависимый справочник. Традиционно его вызов назначается на клавишу F1. В настоящее время такой справочник отсутствует, также в ПО отсутствуют встроенные технические средства для его привязки к контексту. Для привязки и поддержки справочников в используемых в ПО библиотеках Qt есть набор соответствующих средств. Но отсутствие привязки в коде обусловлено тем, что справочник нет смысла привязывать к ПО до того, как будет разработана основная часть ПО хотя бы до состояния альфа-версии. В процессе разработки модификации ПО автоматически влекут за собой изменения привязки страниц справочника. Поэтому необходимо сначала сделать ПО консистентным, зафиксировать версию, перестать изменять и добавлять в него новые функции.

Написание текстов справочника — отдельная крупная задача, которую должны решать специально отобранные для этого специалисты: технические писатели. Сейчас уже можно начать создать документацию с гипертекстовыми связями. Процесс создания документации имеет итерационный характер, сходный с разработкой ПО. В процессе необходимо корректировать грамматические и пунктуационные ошибки, проверять документацию на соответствие контексту.

Трудоёмкость создания контекстного справочника примерно в 2-3 раза меньше трудоёмкости разработки самого ПО. Это означает, что для выпуска первой версии требуется примерно около 1-го человеко-года. Но созданием первой версии справочника его разработка не заканчивается. Во-первых, справочник корректируется при добавлении или изменении функций ПО. Во-вторых, практически никогда первая версия документации не является полноценной. Поэтому работа над справочником должна продолжаться далее параллельно с разработкой ПО.

Разработка документации по ГОСТ

Помимо контекстного справочника, отдельным Заказчикам может потребоваться документация в электронном виде, соответствующая ГОСТ 19, известному также, как ЕСПД. Этот ГОСТ был преимущественно принят в 1977-м году, с изменениями в 1983-м году, после чего дважды подвергался незначительным коррекциям. В настоящее время он полностью устарел и не соответствует современному техническому уровню, но многие Заказчики по-прежнему требуют его исполнения. Изначально разработка документации по ГОСТ 19 не планируется, но она может быть проведена по обозначенному в ТЗ требованию Заказчика и после согласования с ним стоимости выполнения этой работы в виде отдельного пункта в таблице согласованной цены. Необходимо заметить, что разработка документации в строгом соответствии с ГОСТ 19 имеет очень высокую трудоёмкость, поскольку ГОСТ содержит много

требований, которые могут быть удовлетворены либо ручной обработкой, либо с использованием автоматизированных средств обработки документации семейства TeX. Второй способ подразумевает высокую квалификацию технического писателя, на уровне, который в настоящее время является большой редкостью.

Две главные проблемы при создании документации по ГОСТ 19 — объемы современного ПО, и модульная структура ПО. Основная часть ГОСТ 19 разработана в 1977-м году, когда программы хранились на перфокартах или перфолентах, одна программа решала какую-то одну выделенную задачу, и была объемом всего в несколько сотен строк. Разрабатываемое ПО MAX15 в данный момент времени имеет объем около 75000 строк, решает сотни различных подзадач, состоит из более чем 500 файлов и 37 проектов. При дальнейшей разработке все эти объемы будут расти. Строгое исполнение ГОСТ 19 не позволяет описать такие объемы. Но еще большую проблему представляет модульность. Фактически, ПО MAX15 — это набор из множества программ, каждая из которых является модулем. Они даже могут поставляться в различной комплектации, то есть, набор программ не постоянен. Но в то же время, при их совместном использовании для пользователя они выглядят, как единая программа. ГОСТ 19 не обеспечивает механизмов для описания ПО такого типа. Поэтому в соответствии с ним можно будет выпустить только небольшую часть документов. Очевидно, что в соответствии с ГОСТ 19 настоящее ПО является *комплексом*, поэтому для него обязательным документом является только «Спецификация». Отдельные модули являются *компонентами*, причем их самостоятельное применение вне комплекса невозможно. Поэтому для них единственный обязательный документ - № 12 «Текст программы». С этим документом есть проблема. Он был введён в 1977-м году, когда утеря перфоленты влекла за собой утерю программы, а исходный текст в электронном виде вообще не сохранялся. Поэтому был введён обязательный документ «Текст программы», который должен был содержать весь исходный текст в распечатке на бумаге, и по которому можно было заново набрать программу на перфоленту или перфокарту. В настоящее время исходные тексты хранятся исключительно в электронном виде, в неограниченном количестве копий, поэтому в документе «Текст программы» в редакции 1977 года абсолютно нет необходимости. Фактически, он был бы бесполезным дублированием файлов исходных текстов, причём имеющих значительный объем. Современные средства редактирования позволяют за несколько секунд найти нужный класс или функцию, независимо от объемов проекта. Поэтому в документе «Текст программы» каждого модуля достаточно будет просто привести список всех файлов, которые необходимы для сборки этого модуля, с указанием даты-времени их последней модификации. Это может быть полезно для проверки целостности исходных текстов.

При появлении требования для создания документации в соответствии с ГОСТ 19 нужно будет по-максимуму использовать контекстный справочник. На его основе можно создать самый важный для Заказчиков документ — «Руководство пользователя». В ГОСТ 19 такой документ отсутствует, но его можно ввести под одним из «свободных» номеров с № 90 по № 99. Похожий на него документ из ГОСТ «Руководство оператора» не является подходящим по структуре, так как пользователь ПО MAX15 не является оператором ЭВМ, а является квалифицированным пользователем с гораздо более высоким уровнем. Содержание этих документов также различается.

Описание языка Ci может быть выпущено под № 35, как «Описание языка», это стандартный по ГОСТ 19 документ. При этом «Описание языка» будет выпускаться для каждого компонента в отдельности, и будет включать операции языка, добавляемые этим компонентом. Это

будет не очень удобно для использования, но это не имеет значения, так как описание всех загруженных операций будет доступно в контекстном справочнике. Фактически такой документ не нужен.

Должен быть выпущен документ под одним из свободных номеров №№ 90-99 под названием «Инструкция по копированию, установке и настройке». В нём должны содержаться сведения о том, как следует обращаться с программой как целым при её получении, как устанавливать её на рабочие компьютеры и как настраивать под свои требования и оборудование.

Для внутреннего использования может быть выпущена «Инструкция по сборке проекта», в которой будут приводиться сведения о том, какие необходимы средства для сборки проекта из исходных текстов, и какие действия в каком порядке следует выполнять. ГОСТ 19 не содержит такого документа (32 «Руководство системного программиста» и 33 «Руководство программиста» должны содержать другие сведения).

Разработка автоматизированных средств для обучения

При обучении использованию проектов такого уровня сложности и объемов одной документации обычно не бывает достаточно. Чаще всего, для этого организуются учебные семинары с привлечением специалистов Заказчика, на которых проводится демонстрация использования, решаются определенные примеры, и в конце выдаются сертификаты. Иногда используются видео-записи таких семинаров.

Для настоящего продукта предполагается разработать автоматизированные обучающие средства — отдельный модуль, который будет записывать действия пользователя: перемещения указателя мыши, нажатия кнопок мыши и клавиатуры. Затем они могут воспроизводиться, с возможной приостановкой для реакции пользователя и ветвлением в зависимости от его действий. Так можно будет быстро создавать интерактивные учебные материалы для самого ПО, более эффективные, чем запись видео-уроков. Кроме создания самого модуля, потребуется подготовка сценариев и запись самих уроков. Трудоёмкость этой работы не менее 3-х человеко-месяцев.

Дальнейшее развитие и сопровождение

Планируемые новые возможности

В первую очередь, конечно необходимо отыскать и исправить все имеющиеся очевидные ошибки. Они всегда есть в ПО, разработка которого ведется практически с ноля. От ошибок фактически свободен только язык Ci – их практически нет в компиляторе и исполняющей системе, поскольку они активно использовались в других проектах. Тем не менее, необходимо организовать всестороннее тестирование с целью выявления максимального числа ошибок.

В разрабатываемом ПО планируется реализовать новые возможности, которые отсутствуют в NI LabVIEW. В первую очередь речь идет о функции «нечеткой» (fuzzy) логики, которую в совокупности с накопленными данными и результатами текущих измерений можно использовать для решения широкого круга задач, пока не решаемых с помощью компьютеров. Наиболее интересным здесь является прогнозирование возникновения нештатных ситуаций и критичных состояний контролируемых объектов. На эту тему есть множество научных работ, в том числе, российских. Многие доступны в сети Интернет. Разрабатываемое ПО по своим базовым функциям будет иметь возможность производить автоматические измерения по многим

каналам одновременно, и накапливать их результаты в быстрые базы данных. Такие возможности, в сочетании с простым программным управлением измерительными приборами и исполнительными устройствами — идеальная среда для искусственного интеллекта (ИИ) с нечеткой логикой. Такой ИИ планируется разработать в виде отдельного плагина. Ориентировочная трудоёмкость разработки модуля нечеткой логики составляет 3-4 человеко-года.

Из более простых возможностей, которые также отсутствуют в NI LabVIEW, наиболее полезным будет генератор отчетов. Это программный модуль, который позволит выбирать из баз данных значения измерений по необходимым критериям, и выводить их в грамотно сформированные отчеты в стандартных форматах. В первую очередь будет реализован вывод в PDF, тем более, что все средства для этого имеются в библиотеке Qt. Отчеты могут включать в себя таблицы, графики, тексты и другую необходимую информацию. Также возможно представление отчетов в HTML коде для немедленной публикации в сети Интернет. Для создания отчетов модуль генератора должен будет иметь ряд расширений языка Ci, а также возможна разработка интерактивного редактора форм страниц. Ориентировочная трудоёмкость модуля генератора отчетов составляет 0.5-1 человеко-год.

Еще одной планируемой интересной возможностью является создание «моста» между несколькими компьютерами, на каждом из которых работает ПО MAX15. Это позволит создавать «кластеры» из таких компьютеров, на случай, если мощности одного недостаточно для решения задачи. На самом деле, такая возможность требуется в очень большом числе практических задач, особенно, если обработка производится в реальном времени. Несмотря на высокую вычислительную мощность отдельных ПК, при решении задач реального времени каждый компьютер как правило успевает работать только с одной задачей. Если несколько копий ПО MAX15 на нескольких компьютерах смогут обмениваться сигналами, как если бы это работала одна схема на одном компьютере, это существенно расширит возможности для применения. Например, модули с системами реального времени смогут производить сборку данных, и сохранять их в своих локальных базах данных, а через соединения будут передавать данные в центральный компьютер, на котором будет работать модуль нечеткой логики, принимающий решения. Всё это может быть сделано и в настоящее время другими средствами, но использование MAX15 позволит создавать такие сети из готовых модулей со значительным снижением трудоёмкости.

Планируемые доработки

Кроме перечисленных ранее возможных направлений развития, и кроме разработки модулей -аналогов NI LabVIEW, в на настоящий момент есть еще набор функций, которые надо реализовать, чтобы сделать ПО удобным и простым в использовании. Сюда входят следующие функции:

- Полезно иметь возможность «инкапсулировать» целиком любую схему в один простой элемент схемы, с возможностью его многократного использования. Это даст возможность создавать схемы с виртуально неограниченным уровнем вложенности. Это очень серьезная переработка, которая потребует изменения внутренней идеологии ядра (на плагинах она не скажется). Ориентировочная трудоёмкость около 2-х человеко-месяцев.
- В будущем для критичных задач может потребоваться режим «горячей» замены плагинов — выгрузки устаревшей версии, и загрузки вместо неё обновлённой, подробнее далее в разделе «Обновление имеющихся функций». Ориентировочная трудоёмкость до-

работки около 2-х человеко-месяцев.

- Запуск сервера обновлений. Ориентировочная трудоёмкость около 2-х человеко-месяцев.
- Защита от перегрузок процессорного ресурса — в настоящее время в ПО есть возможность задать такой режим обмена сообщениями между процессорами Ci, что приложение перестанет обрабатывать события и реагировать на внешние воздействия. Планируется разработать автоматический алгоритм, определяющий такую ситуацию, и гасящий процессы, которые приводят к потере реактивности. Это не простая задача, решение может потребовать времени. Ориентировочная трудоёмкость около 1.5 человеко-месяцев.
- Редактор схемы надо доработать, чтобы иметь возможность открывать одновременно несколько схем. Это упростит работу с «инкапсулированными» схемами. Ориентировочная трудоёмкость около 1 человеко-месяца.
- Нужно сделать генератор приложения — модуль, который будет автоматически копировать в указанный каталог только те плагины и другие файлы, которые используются в текущем открытом виртуальном приборе. Немного подробнее об этом в разделе «Конфигурирование для поставки» этого документа. Ориентировочная трудоёмкость около 1 человеко-месяца.
- Модуль для автоматизированного обучения, подробнее описан в разделе этого документа, посвященном документации. Ориентировочная трудоёмкость около 1 чел-месяца.
- Для создания более разнообразных виртуальных приборов нужно разработать несколько модулей элементов управления (не имеющих аналогов в NI LabVIEW). Ориентировочная трудоёмкость около 1-го человеко-месяца.
- Доработки редактора исходных текстов — в нём недостаёт некоторых удобных функций. Ориентировочная трудоёмкость около 1 человеко-месяцев.
- Есть ряд возможных, хотя и не обязательных доработок отладчика. Ориентировочная трудоёмкость около 1 чел-месяцев.

Перечисленные работы требуют около 15-ти человеко-месяцев, но могут выполняться параллельно с любыми другими при условии найма дополнительных сотрудников. Ни одна из доработок в настоящее время не является критичной или обязательной, они могут производиться в любом порядке по мере необходимости или возможности.

Поддержка и обновление имеющихся функций

Создание новых функций по требованиям Заказчиков

Это направление дальнейшего развития предполагается сделать одним из основных способов получения доходов. По требованиям Заказчиков решения могут производиться следующими основными способами:

- Разработка плагинов для управления оборудованием и обработки данных Заказчика на языке C/C++
- Разработка программ для виртуальных процессоров, решающих прикладные задачи Заказчика на языке Ci
- Разработка конечных виртуальных приборов с использованием всех возможностей этого ПО.

Первый способ решений наиболее универсален. ПО может быть поставлено Заказчику для решения его текущих задач. Далее, при изменении требований, либо появлении новых, по же-

ланию Заказчика могут быть разработаны плагины для использования с новыми приборами, специальных расчетов или обработки сигналов и т.д. При этом у Заказчика уже будет основная часть ПО, и ему только потребуется добавить или заменить плагины в его составе, и возможно немного изменить схему обработки. То есть, для Заказчика изменения и затраты будут минимальны, на что собственно и рассчитана архитектура ПО MAX15.

Первоначально не предполагается открывать архитектуру плагинов, чтобы Заказчики не могли сами их разрабатывать. Хотя в дальнейшем это будет необходимо сделать, так как в ПО изначально заложена возможность разработки плагинов сторонними лицами в соответствии со спецификациями. Такая возможность послужит популяризации и распространению этого ПО. В дальнейшем, если количество заказов будет превосходить возможности их реализации своими силами, Заказчикам за дополнительную плату и на условиях нераспространения могут быть переданы соответствующие спецификации.

Обновление имеющихся функций

Перезагрузка плагинов

Модульная архитектура ПО MAX15 специально рассчитана на упрощение обновлений. Такие обновления необходимы при обнаружении ошибок и добавлении новой функциональности. В данном ПО нет необходимости заменять всё ПО целиком, достаточно заменять отдельные модули. Текущая версия не поддерживает «горячую» замену, но в дальнейшем это может быть реализовано. Это можно обеспечить, поскольку соединения в схемах могут меняться динамически. Таким образом, есть почти всё необходимое, надо только при получении новой версии плагина сохранить его в нужный каталог, загрузить в оперативную память, и быстро переключить соединения предыдущей версии на новую. После чего старую версию плагина можно выгрузить, а файл удалить. При этом надо корректно передать новой версии плагина настройки от старой версии. В принципе, это можно делать без остановки работы схемы, то есть без выключения виртуального прибора, не говоря уже о перезагрузке ПО. Это уникальная возможность, недоступная в NI LabVIEW.

В настоящем варианте такой возможности нет, хотя для её реализации нет препятствий. Это отдельная доработка, которую можно сделать в дальнейшем. Сейчас любой плагин может быть загружен в любое время, а также плагины могут быть выгружены, если только они не используют класс «главного окна» из библиотек Qt (без этого можно легко обходиться). Динамическое изменение связей поддерживается изначально. Необходимо только разработать и тщательно отладить последовательность действий по обнаружению, загрузке, смене соединений и выгрузке.

Если же динамическая замена плагинов не требуется, то это обновление заменой плагинов сейчас полностью поддерживается архитектурой, необходимо только отработать механизм распространения обновлений.

Сервер обновлений и поддержки

Предполагается, что обновления ПО пользователи будут получать со специального сервера, на котором будут доступны различные версии ПО и плагинов. Доступ к серверу будет по осуществляться по платной подписке (возможно для учебных заведений и энтузиастов бесплатной). В настоящее время в ПО не предусмотрено автоматическое скачивание

обновлений, но его можно легко реализовать в дополнительном плагине.

Сервер обновлений должен быть выделенным, собственным, работающим на базе ОС семейства *NIX. На этом же сервере предполагается запустить веб-сайт продукта, доступный зарегистрированным пользователям. Для популяризации продукта предполагается бесплатное распространение демо-версии с урезанной функциональностью и ограниченным составом плагинов. Планируется так проработать состав плагинов демо-версии, чтобы они оказались интересны домашним энтузиастам. Это позволит создать интернет-сообщество, которое будет выполнять следующие функции, причем совершенно бесплатно:

- массированное тестирование функций ПО, особенно новых выпускаемых версий
- «ползучий» маркетинг, то есть не требующее затрат распространение информации о продукте, его достоинствах и возможностях
- пользовательское консультирование и поддержка, то есть режим, когда проблемы одних пользователей решают другие, более продвинутые пользователи — это позволит иметь развитую поддержку без увеличения расходов на неё.

Для этого на сервере необходимо будет запустить и постоянно поддерживать веб-форум. Ориентировочные сроки запуска сервера с веб-сайтом и форумом — около 2-х человеко-месяцев.

Для поддержки пользователей, подписанных на платную услугу, в будущем необходимо будет содержать Q&A-специалистов, хорошо знающих продукт. Необходимость в этом возникнет, если количество Заказчиков с платной подпиской будет значительным. В первое время такую поддержку можно будет организовать без специальных сотрудников.

График работ на 3 года при получении финансирования:

Период	Работы	Результаты
1-й год	• Исправление имеющихся ошибок • Разработка контекстного справочника • Разработка генератора приложений • Разработка защиты от перегрузок • Разработка модуля для управления приборами через VISA • Мелкие доработки • Запуск сервера обновлений, веб-сайта и форумов	• Выпуск 1-й версии продукта • Заключение договоров с Заказчиками • Начало поставок
2-й год	• Разработка системы автоматизированного обучения • Разработка режима работы редактора с несколькими закладками • Разработка ядра и режима работы редактора с вложенными схемами • Разработка части модулей с поддержкой функций, аналогичных NI LabVIEW • Разработка модулей дополнительных элементов управления • Разработка плагинов для различных протоколов • Решение задач Заказчиков	• Выпуск 2-й версии продукта • Создание сообщества пользователей • Расширение присутствия на российском рынке

3-й год	• Разработка режима горячей замены • Разработка моста между компьютерами с несколькими системами MAX15 • Разработка остальных модулей с поддержкой функций, аналогичных NI LabVIEW • Перевод на английский язык, включая документацию • Решение задач Заказчиков • Создание англоязычной версии сайта и форумов • Поиск партнеров в Китае для выпуска китайской версии продукта	• Выпуск 3-й версии продукта, двуязычной • Расширение присутствия на рынке • Выход на международный уровень

Таблица 5