Московский государственный университет имени М.В. Ломоносова
Факультет вычислительной математики и кибернетики
Кафедра автоматизации систем вычислительных комплексов

Маслов Никита Сергеевич

Создание инструмента анализа информационного обмена для интерфейса I2C, применяемого во встраиваемых вычислительных системах

КУРСОВАЯ РАБОТА

Научный руководитель:
А.В. Герасёв

Москва, 2016

Аннотация

Данная работа посвящена задаче создания инструмента для анализа и мониторинга информационного обмена для интерфейса I2C, часто применяемого для обмена данными между узлами встраиваемых вычислительных систем.

Раздел 3 содержит описание шины I2C, выбранной при постановке задачи данной работы в качестве целевой шины для реализации инструмента анализа.

В разделах 4 и 5 проведено исследование существующих на рынке свободных и коммерческих программно-аппаратных комплексов для анализа и мониторинга шины I2C, выявлены их особенности и сформулированы требования к разрабатываемому средству.

В разделах 6 и 7 приведен проект доработки разрабатываемой в ЛВК инструментальной среды Opermon и описана реализация. Проведена апробация, сделаны выводы об удобстве реализованного средства и его применимости для некоторых практических задач, сформулированы замечания и предложения по доработке средства.

Также ряд предложений по доработке содержится в разделе 8, посвященном задаче рефакторинга, основной целью которого является разделение инструмента на базовую часть и компоненты, ответственные за реализацию поддержки конкретных интерфейсов. Данное разделение необходимо для возможности раздельной сборки решения и опубликования анализатора для шины I2C в открытом доступе.

Содержание

1 Введение

При разработке и отладке встраиваемых систем иногда возникает необходимость мониторинга, регистрации, визуализации и анализа данных, передаваемых по шинам данных устройства [1]. У возникающих ошибок зачастую может быть разная природа, начиная от неправильной конфигурации приёмо-передающего блока и заканчивая ошибками в логике работы пользовательского программного обеспечения.

Возможностей традиционных отладчиков, предназначенных в первую очередь для поиска ошибок в ПО, в этом случае часто не хватает, так как в работе шины задействовано сразу несколько устройств, синхронизированных по времени. Более того, традиционный отладчик не даёт понимания того, что происходит на шине данных (невозможно или неоправданно сложно отследить состояние передачи и синхронизацию линий обмена данными). Поэтому для решения подобных задач используются специальные программно-аппаратные комплексы для мониторинга и анализа обменов.

Как правило, такой программно-аппаратный комплекс состоит из двух компонентов: адаптер для подключения персонального компьютера (ПК) к анализируемой шине данных и специального программного обеспечения (ПО) для обработки и визуализации данных.

В ЛВК ведется ряд работ, целью которых является создание инструментальной среды для мониторинга, регистрации, визуализации и анализа информационного обмена в различных бортовых авиационных каналах (ARINC 429, MIL STD 1553B, Fibre channel и др.) [2].

Рабочей группой достигнуты значимые результаты и получен богатый опыт в данной области. Однако данные работы ведутся в интересах конкретных заказчиков и их результаты являются закрытыми разработками. С целью популяризации ЛВК в научном и техническом сообществе может иметь смысл создание на основе существующих разработок свободного инструмента для анализа информационного обмена.

С другой стороны, в области встроенных вычислительных систем, исследованием которых также занимаются в ЛВК, есть много специализированных интерфейсов, для которых инструменты анализа информационного обмена либо совсем отсутствуют, либо существуют, но проигрывают по возможностям аналогичным инструментам, разрабатываемым в ЛВК.

В качестве одного из таких интерфейсов можно рассмотреть шину I2C [3]. Этот интерфейс широко используется для связи интегральных схем отдельных узлов встраиваемых вычислительных систем за счёт своей простоты и большого количества поддерживаемых устройств различ-

ного назначения. Однако, на сегодняшний день не существует свободного программно-аппаратного комплекса для мониторинга этой шины, ориентированного на анализ обменов на уровне сообщений. Основное множество доступных программных средств для мониторинга шины I2C ориентировано на анализ передач на физическом и канальном уровне.

2 Цель и задачи курсовой работы

Целью курсовой работы является адаптация существующей инструментальной среды Opermon для работы с адаптером шины I2C [3] и для визуализации получаемых обменов с учётом специфики устройства шины.

Для достижения поставленной цели необходимо решить следующие задачи:

- Провести анализ существующих программных и аппаратных средств анализа шины I2C (а также других специализированных интерфейсов встроенных вычислительных систем) с точки зрения:
  - возможностей для мониторинга и анализа передач;
  - организации пользовательского интерфейса;
  - доступности для разработчика.
- Выполнить их сравнение с разрабатываемой в ЛВК средой OperMon.
- Предложить проект реализации инструмента анализа информационного обмена по шине I2C на основе среды OperMon:
  - выбрать аппаратный интерфейс к шине;
  - спроектировать архитектуру средства;
  - предложить набор представлений информации об обмене и спроектировать пользовательский интерфейс.
- Реализовать инструмент анализа;
- Провести анализ существующей архитектуры среды Opermon и предложить проект рефакторинга с целью:
  - более явного выделения интерфейсо-специфичной части;

– реализации возможности вариативной сборки с заданием списка поддерживаемых в данном варианте интерфейсов;

– разбиения исходного кода и собираемых объектных компонентов на независимые модули и библиотеки с возможностью выделения части этих модулей, достаточной для сборки средства мониторинга шины I2C.

3 Шина I2C

I2C [3] — последовательная шина данных для связи интегральных схем, использующая две двунаправленные линии связи. Используется для соединения низкоскоростных периферийных компонентов с материнской платой, встраиваемыми системами и мобильными телефонами. Название представляет собой аббревиатуру слов Inter-Integrated Circuit. Шина данных поддерживает "горячее" подключение устройств (без перезапуска всей системы), а также одновременную работу нескольких ведущих устройств.

Данные передаются по двум проводам — *линии данных* (SDA) и *линии тактов* (SCL). В обмене всегда участвует два устройства: ведущий (*master*) и ведомый (*slave*). Тактовый сигнал на линии SCL генерируется ведущим устройством; ведомое устройство использует этот сигнал как опорный в том числе и для передачи данных к ведущему. Всего на одной двупроводной шине может быть до 127 устройств (адрес ведомого устройства кодируется 7 битами данных).

Каждая линия подключена к "плюсу" питания устройства через специальный *подтягивающий резистор*. Таким образом, когда шина свободна (обмены не производятся), с обеих линий считывается высокий уровень напряжения (логическая единица). Когда устройство начинает передачу, оно подключает требуемую линию к "минусу" питания и на линии устанавливается низкий уровень напряжения (логический ноль). Такая схема подключения называется *монтажное "И"*. Она позволяет нескольким устройствам устанавливать уровни одновременно без риска образования короткого замыкания и, как следствие, повреждения устройств.

Обмен начинается с события **START**, когда ведущее устройство устанавливает низкий уровень на линии данных (SDA) при высоком уровне на линии тактирования (SCL). В этот момент остальные устройства на линии определяют шину как занятую и ожидают окончания передачи.

После события **START** ведущее устройство передаёт байт, содержащий адрес ведомого устройства (7 бит) и бит режим обмена: 1 (R) - получение данных от ведомого устройства (чтение), 0 (W) - передача данных ведомому устройству (запись). Если на линии есть устройство с указанным адресом, оно должно установить линию SDA в низкий уровень для подтверждения приёма (событие **ACK**, от англ. acknowledge - подтверждение).

После этого происходит непосредственно обмен данными: в зависимости от выбранного режима, либо данные передаются от ведомого устройства, либо от ведущего. Получатель должен подтверждать приём каждого байта (исключение делается только для последнего байта в режиме

чтения данных - master должен сообщить ведомому устройству об окончании приёма). Тактовый сигнал генерируется ведущим устройством, но при необходимости ведомое устройство может устанавливать тактовую линию в низкий уровень и, таким образом, задерживать обмен (например, если не успевает обрабатывать данные).

В конце обмена ведущее устройство меняет состояние линии SDA с низкого на высокий при высоком уровне на линии SCL - событие **STOP**. После этого шина считается освобождённой и в ней могут начинаться новые обмены. Стандарт I2C допускает также начало нового обмена тем же ведущим устройством, которое вело предыдущий обмен, без освобождения линии (если между обменами нет значительной паузы) - так называемое событие “повторный START”.

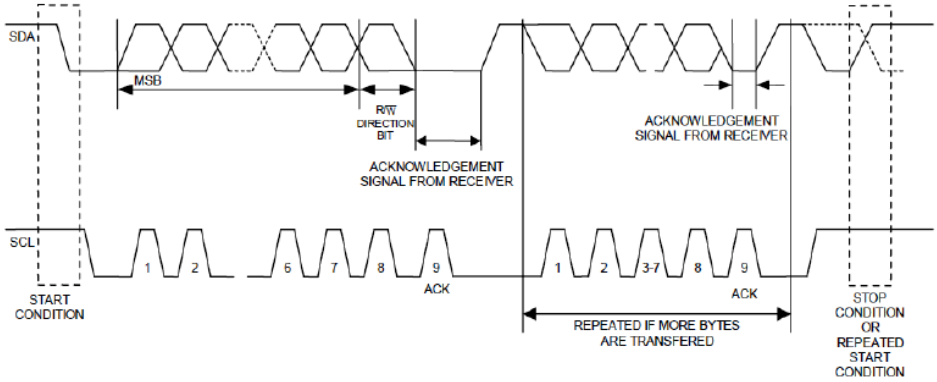


Рис. 1: Структура информационного обмена на шине I2C

Таким образом, **обменом** на шине I2C будем считать последовательность управляющих сигналов и данных, которая начинается с события **START** и заканчивается событием **STOP** или **START**.

Обмен может содержать ошибки. Следующие ошибки могут быть обнаружены на этапе считывания обмена:

- ведомое устройство не найдено - отсутствует ACK после первого байта обмена;
- ведомое устройство потеряно или неисправно - отсутствует ACK после переданного байта;
- некорректное начало передачи - отсутствует событие **START**.

Для представления корректного обмена нужно отобразить следующую информацию:

- **время начала обмена** - время возникновения события **START**;
- **длительность обмена** - время между событиями **START** и **STOP** (или повторным **START**;
- адрес ведомого устройства;
- режим передачи - R или W;
- передаваемые данные (полезная нагрузка, payload).

4 Анализ существующих решений

На сегодняшний день разработчикам встраиваемых систем доступен довольно широкий спектр различных программно-аппаратных комплексов для регистрации и анализа обменов на шине I2C. Для подготовки собственного решения есть смысл ознакомиться с ними, провести сравнительный анализ и выделить слабые и сильные стороны каждого из них.

4.1 Преобразователи I2C $\leftrightarrow$ TTY

Одним из самых простых решений для прослушивания шины является подключение к ней через простой по устройству адаптер, имеющий последовательный интерфейс, Разработчику доступно множество вариантов, начиная от несложных самодельных адаптеров, использующих микроконтроллер общего назначения в качестве преобразователя интерфейса, до серийно выпускаемых универсальных адаптеров.

С представителями первой группы можно ознакомиться по ссылкам: [5], [6].

Вторая группа представлена следующими адаптерами: [7].

Зачастую эти простые преобразователи подразумевают их использование вместе со специальным программным обеспечением, предоставляя помимо консольного интерфейса бинарный вариант.

4.1.1 Плюсы

- **Простота исполнения и дешевизна**. Как правило, при проектировании подобных адаптеров используются недорогие распространённые компоненты, при этом зачастую допуская изготовление платы в "кустарных"условиях. Как следствие, такой адаптер может позволить себе любой заинтересованный разработчик.

- **Отсутствие необходимости в специфическом ПО**. Большинство адаптеров этого типа имеют текстовый консольный интерфейс, поэтому для начала работы с ними требуется только программа-терминал последовательного порта.

- **Возможность интеграции с пользовательским ПО**. Как правило, программный интерфейс таких адаптеров открыт, достаточно прост и не использует специфических решений для обмена данными с ПК (работа с последовательным портом доступна практически

на всех платформах с использованием почти всех распространённых языков программирования). Таким образом, у пользователя есть возможность интегрировать взаимодействие с адаптером в своём собственном ПО.

Можно также заметить, что при использовании подобных адаптеров при определённой сноровке есть возможность записывать трассы обменов для последующего анализа, так как данные представляются в виде простого текстового (или бинарного) потока. Более того, записанные данные (в случае текстового потока) можно пытаться анализировать даже без использования специального ПО. Также есть возможность преобразовать записанные данные в требуемый пользователю формат (например, XML или CSV) с помощью несложных скриптов.

4.1.2 Минусы

Замечание: рассматриваются отрицательные стороны использования решения в консольном режиме, без специальных утилит.

- **Представление данных**. Чаще всего, консольный интерфейс сильно ограничен в плане представления данных (так как по своей природе может выводить только текст). При большом количестве обменов на линии такой интерфейс становится чрезвычайно неудобным.

- **Фильтрация передач**. Как правило, простые аппаратные анализаторы не имеют возможности предоставить выборку данных по шаблону (как и пытаться декодировать посылку стандартного формата).

4.1.3 Выводы

Решения, основанные на использовании простых преобразователей с консольным интерфейсом неплохо подходят для анализа редко возникающих событий на шине с небольшими объёмами передаваемых данных. Они дешевы и в общем случае не требуют специального (возможно, дорогостоящего) ПО для начала работы. Более того, за счёт относительной простоты и открытости интерфейса взаимодействия с адаптером, они оставляют пользователю определённую свободу в выборе инструмента анализа, начиная от ПО для графического представления и анализа данных (при преобразовании данных трасс в поддерживаемые форматы, такие как CSV или XML) и заканчивая специальными скриптами для разбора обменов.

4.2 Специализированные программно-аппаратные комплексы

Отдельного внимания заслуживают специализированные программно-аппаратные решения для визуализации и анализа обменов, в том числе коммерческие закрытые решения. При проектировании собственного решения очень полезно ознакомиться с опытом уже существующих, сравнить их с точки зрения предлагаемых возможностей, удобства в использовании, а также оценить сильные и слабые стороны каждого из них.

4.2.1 Beagle I2C/SPI Protocol Analyzer + Data Center

Beagle I2C/SPI Protocol Analyzer [8] - устройство для захвата и анализа, разработанное американской компанией Total Phase, специализирующейся на решениях для разработки и отладки встраиваемых систем. Оно подключается к ПК посредством интерфейса USB 2.0 и использует собственное API для взаимодействия с приложениями.

Анализатор способен считывать данные с линии I2C, работающей на скорости до 4 Мбод (спецификация I2C High-speed mode), при этом работает исключительно как сниффер (пассивный анализатор обменов).

Для работы с линейкой анализаторов от Total Phase разработано программное обеспечение Data Center [9]. По типу представления данных ПО идеологически близко к WireShark.

Пользовательский интерфейс инструмента Data Center включает в себя следующие базовые элементы:

- **Таблица обменов**. Отображает полученные обмены в расшифрованном виде (с учётом типа интерфейса и протокола обмена).
- **Сырые данные обмена**. Представляет данные в битовом виде (16-ричные значения полученных данных).
- **Командная строка**. Позволяет управлять функционалом ПО с помощью текстовых команд.
- **Информация об устройстве**. Отображает список подключенных к ПК устройств, доступных для анализа, а также подробную информацию о текущем выбранном устройстве.

Снимок экрана с рабочим окном Data Center представлен в приложении А на рисунке 5.

4.2.2 Aardwark Host Adapter + Control Center

Помимо линейки анализаторов шин данных, компания Total Phase предлагает устройства и ПО для активного вмешательства в работу шины. Это необходимо, например, для получения данных в шине без master-устройства (либо где роль master-устройства играет ПК с подключенным адаптером), либо для эмуляции оконечных устройств шины средствами ПК с адаптером.

Для работы с шиной I2C предлагается адаптер Aardwark I2C/SPI Host Adapter [10] и ПО Control Center [11]. Адаптер подключается к ПК посредством интерфейса USB 2.0 и использует собственное API для взаимодействия с пользовательскими приложениями.

ПО Control Center позволяет ПК выступать в роли master- или slave-устройства на шине I2C или SPI.

Возможности ПО Control Center:

- **Настройка напряжения логических уровней и питания целевого устройства**. Адаптер Aardwark I2C/SPI Host Adapter имеет встроенный конвертер логических уровней и программируемый источник питания, от которого можно передавать питание отлаживаемому устройству. Настройка этих блоков производится из пользовательского интерфейса Control Center.

- **Ручное взаимодействие с шиной**. Пользовательский интерфейс Control Center включает в себя виджеты для ручного управления шиной в разных режимах: генерирование запросов в режиме master, приём или передача данных в режимах master или slave.

- **Ведение лога**. В лог вносятся события о получении или передаче данных по шине, а также об изменениях настроек конвертеров уровней и источников питания.

Снимок экрана с рабочим окном Control Center представлен в приложении А на рисунке 6.

4.2.3 Логические анализаторы BitScope Logic

Отдельной категорией устройств для отладки и мониторинга шин данных являются универсальные логические анализаторы. Эти устройства имеют несколько (как правило, порядка 16) входных каналов, способных отслеживать логические уровни на линиях данных. Логические анализаторы обычно подключаются к ПК посредством USB или через более скоростную шину PCI.

Программное обеспечение для работы с логическими анализаторами обычно ориентировано на отображение текущего состояния входных каналов подобно осциллографу, в виде графика зависимости уровня от времени. Такие анализаторы чрезвычайно полезны для поиска ошибок на физическом уровне.

Интересным представителем ПО для работы с логическими анализаторами является BitScope Logic [12]. Оно используется для работы с оборудованием BitScope.

Возможности BitScope Logic:

- **Отображение состояния входов логического анализатора**. На экран выводится график зависимости уровня на входе от времени.
- **Расшифровка обменов**. BitScope Logic поддерживает работу с интерфейсами UART, I2C, SPI, CAN и многими другими и имеет возможность расшифровывать (представлять в текстовом виде) обмены этих интерфейсов.
- **Представление данных в табличном виде**. В ПО есть возможность представления обменов в традиционном табличном виде.

Снимок экрана с рабочим окном BitScope Logic представлен в приложении А на рисунке 7.

4.3 Выводы

Рассмотренные комплексы предлагают несколько базовых подходов к представлению данных об обменах:

- представление на уровне логических уровней во времени;
- представление в виде потока байтов и управляющих символов;
- табличное представление “сырых” данных обменов;
- табличное представление с расшифровкой данных обменов.

Каждое из представлений полезно в своих определённых задачах. Тем не менее, есть ограничения по использованию некоторых из них в задаче, поставленной в этой курсовой работе. Для того, чтобы определить требования к реализации поддержки шины I2C в среде Opermon, следует провести обзор уже существующего функционала Opermon.

5 Инструментальная среда Opermon

Opermon [2] - инструмент для отображения, анализа и сбора трасс, созданный изначально для работы с бортовыми авиационными шинами данных, такими как MSTD-1553, ARINC, FibreChannel и CAN. На сегодняшний день Opermon имеет следующие возможности:

- отображение обменов в табличном виде;
- сохранение трасс для последующего анализа (с возможностью удаления старых обменов из трассы для длительной непрерывной работы);
- разбор сообщений с выделением параметров;
- отображение значений параметров в табличном виде или в виде графиков.

Решение Opermon имеет клиент-серверную архитектуру, где разделены инструмент взаимодействия с адаптером и средство визуализации.

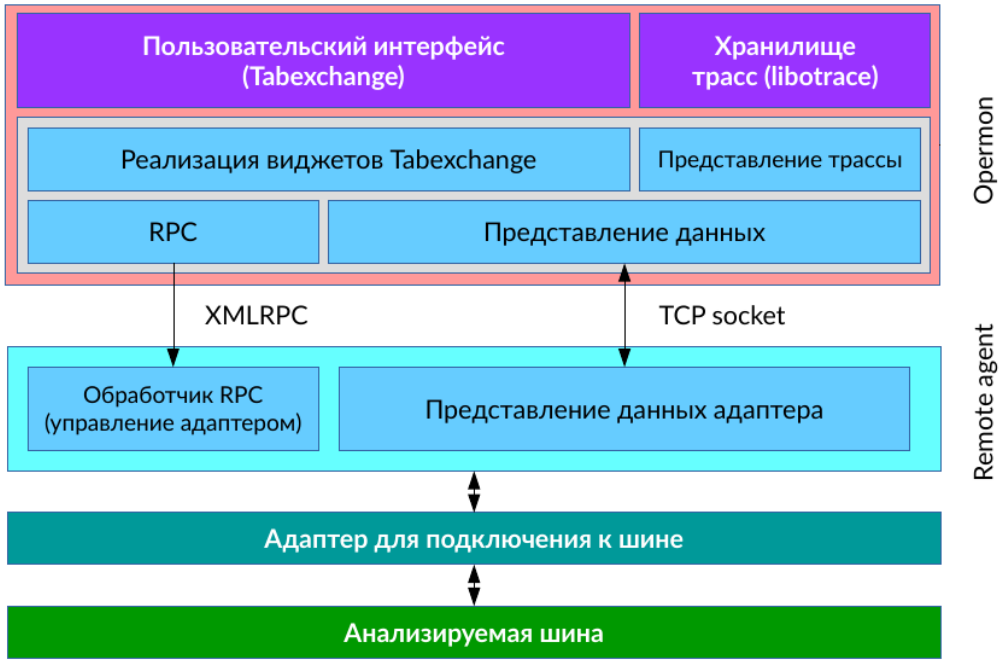


Рис. 2: Структурная схема связи компонентов среды Opermon

5.1 Серверная часть

Задача удалённого агента - устанавливать соединение с адаптером и обеспечивать обмен данными между адаптером и средством визуализации Opermon. Взаимодействие с Opermon происходит через пару TCP-соединений.

Первое соединение служит для передачи команд управления через RPC (Remote Procedure Call, вызов удалённых процедур) с помощью библиотеки qxmlrpc [13]. Агент ожидает подключения средства визуализации.

Второе TCP-соединение открывается по команде bind от средства визуализации. Через это соединение передаются данные о прочитанных обменах, а также сообщения в очередь на передачу через адаптер (если реализация интерфейса в Opermon поддерживает передачу данных на шину).

5.2 Клиентская часть

Средство визуализации оформлено в виде окна с набором отделяемых виджетов для отображения данных и настройки различных параметров.

В задачи средства визуализации также входит предоставление пользовательского интерфейса для подключения к агенту (или нескольким агентам) и для настройки параметров адаптеров.

Основное окно программы содержит следующие виджеты [14]:

- **Область отображения результатов**. В этом виджете отображается таблица обменов (несколько таблиц при использовании нескольких адаптеров), а также статистика обменов и параметры передаваемой циклограммы (если поддерживается адаптером).
- **Панели инструментов**. Позволяют выбирать текущие отображаемые виджеты, а также управлять текущим режимом работы анализатора.
- Виджеты боковой панели:
  - **Настройка адаптеров**. Предоставляет интерфейс для выбора адаптеров для работы, а также позволяет настроить каждый выбранный адаптер в отдельности (с возможностью получать данные от разных агентов).
  - **Расширенная информация о выбранном обмене**. Позволяет просматривать информацию о выбранном обмене в виде пар ключ-значение.

  - **Фильтр обменов**. Инструмент, дающий возможность отображать и обрабатывать только те обмены, которые соответствуют выбранным параметрам.
  - **Поиск**. Предоставляет возможность искать конкретные обмены по запросу пользователя.

- **Область информационных сообщений**. Отображает список произошедших за время работы анализатора событий (таких как загрузка сессии, подключение к адаптеру, начало и конец регистрации обменов) в хронологическом порядке.

- **Строка состояния**. Описывает текущий режим работы анализатора.

5.3 Инструмент хранения трасс обменов

Средство хранения трасс позволяет пользователю производить запись трассы получаемых обменов для последующего отображения, обработки и анализа.

Данный функционал в актуальной версии решения Opermon реализуется логическим модулем libotrace и частично в клиентской части. Доступ к функционалу предоставляется в рамках клиентской части среды.

В задачи инструмента хранения трасс обменов входят:

- **Сбор и экспорт данных трасс обменов**. Трассы обменов сохраняются на диск в отчуждаемом формате. Таким образом, есть возможность анализировать обмены вне рабочего места оператора (часто собранные данные полезны разработчикам и инженерам, работающим с анализируемой системой).

- **Удаление устаревших данных в процессе записи трассы**. Данная функция необходима в случае длительной непрерывной работы решения, когда свободного дискового пространства может не хватить для записи всех полученных обменов. В этом случае в зависимости от настройки Opermon может удалять старые обмены в процессе записи для экономии пространства.

5.4 Особенности архитектуры решения

В этом разделе описаны особенности архитектуры Opermon, определяющие порядок реализации нового типа адаптера в среде.

5.4.1 Представление данных в среде

Для корректного представления данных в среде и для возможности адекватно их обрабатывать требуется достаточно жёсткая структуризация получаемых от адаптера данных. В среде Opermon каждый обмен имеет как минимум два основных способа представления, это связано с тем, что в текущей версии обмены обрабатываются по отдельности средством визуализации и инструментом для хранения трасс:

- **Представление для инструмента отображения**. Здесь от представления требуется максимальная информативность, возможность разделения на логические поля для упрощения отображения.
- **Представление для инструмента хранения трасс обменов**. От представления требуется возможность сериализации данных (возможность сохранения их во внешнем файле для последующего импорта в инструмент анализа и обработки).

5.4.2 Обмен данными между агентом и средством визуализации

Как было описано в разделе 5.1, между клиентской и серверной частями устанавливается два TCP-соединения: для управления адаптерами через RPC и для обмена данными (для каждого адаптера).

В RPC серверной части решения требуется реализовать набор базовых функций управления:

- **list** - для получения списка доступных адаптеров данного типа;
- **bind** - для подключения к определённому адаптеру системы;
- **ping** - для периодической проверки доступности адаптера в системе во время работы;
- **connect** - для установки дополнительного TCP-соединения для обмена данными;
- **start** - для начала обмена данными с адаптером;
- **stop** - для остановки обмена данными с адаптером;
- **unbind** - для отсоединения от адаптера, что сделает его доступным для других приложений системы;
- **setOptions** - для чтения и установки параметров адаптера;

- **putData** - для передачи циклограмм обменов от клиента адаптеру (вывод данных в шину).

Формат данных для передачи обменов не стандартизирован и может быть выбран разработчиком реализации конкретного интерфейса в зависимости от особенностей анализируемой шины.

5.4.3 Виджеты для отображения и конфигурации

Решение Opermon разработано на базе фреймворка Qt [15] и использует его возможности для реализации графического пользовательского интерфейса.

Для стандартных виджетов Opermon подготовлены интерфейсы взаимодействия с ядром системы. Каждый из стандартных виджетов требует отдельной реализации для всех типов адаптеров.

6 Реализация инструмента для анализа обменов на шине I2C

В рамках курсовой работы стояла задача по реализации инструмента для анализа обменов на шине I2C на базе уже существующего решения для визуализации и анализа обменов Opermon. В этом разделе описаны детали реализации решения.

6.1 Выбор адаптера

Для получения данных с шины I2C был выбран адаптер Bus Pirate [7] по следующим причинам:

- Адаптер имеет простой и удобный интерфейс взаимодействия, как консольный (для работы без специализированного ПО), так и бинарный;
- Bus Pirate поддерживает режим I2C sniffer [4], что необходимо для реализации средства анализа;
- Bus Pirate - проект с открытым исходным кодом и открытой реализацией оборудования. Вокруг проекта образовалось обширное сообщество, постоянно занимающееся разработкой новых версий аппаратного и программного обеспечения;
- Доступная стоимость адаптера.

Сообщения в формате Bus Pirate (в бинарном режиме) имеют текстово-бинарный формат (отдельно кодируются символы, обозначающие события на шине). Данные об обменах представляются в следующем виде [17]:

- символ "**[**" обозначает начало обмена (событие **START**);
- символ "**]**" обозначает конец сообщения (событие **STOP**);
- ответы принимающей стороны - **ACK** и **NACK** кодируются символами "+" и "**-**" соответственно;
- передаваемые данные начинаются с символа "\", после чего следует считанный байт данных.

Пример фрагмента получаемого потока данных от адаптера (0xMN - бинарное значение, 1 байт):

```
[\\0xA0+\\0x01+\\0x02+][\\0xA1+\\0xDE+\\0xAD+\\0xF0+\\0x0D-]
```

6.2 Проектирование решения

6.2.1 Серверная часть

Задача удалённого агента - определять подключенные к системе совместимые адаптеры и осуществлять обмен данными между Opermon и выбранным адаптером. Обмен с адаптером происходит посредством последовательного соединения.

Алгоритм поиска доступных в системе адаптеров приведён на схеме ниже. С помощью такого алгоритма можно определить также устройства, совместимые с Bus Pirate. При проверке адаптер только переводится в режим бинарного интерфейса, что безопасно для подключенного оборудования. После проверки обнаруженные адаптеры перезапускаются для того, чтобы оставить пользователю возможность использовать часть адаптеров в режиме консольного интерфейса.

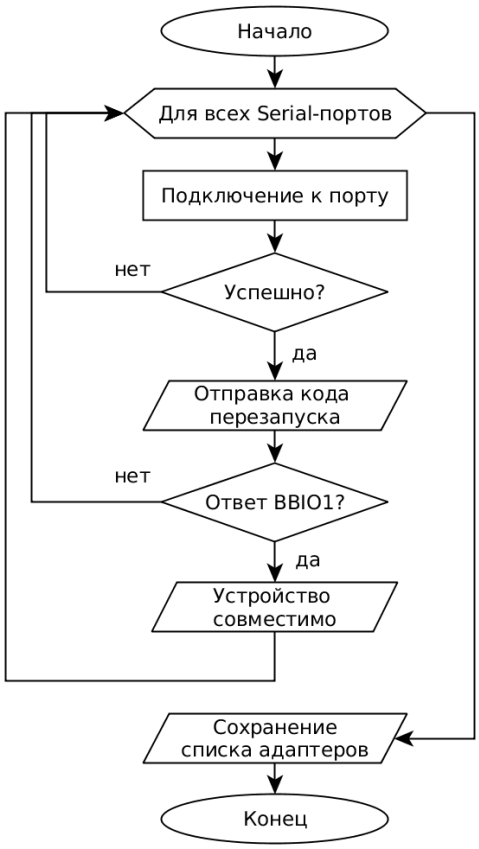


Рис. 3: Алгоритм обнаружения Bus Pirate - совместимых адаптеров в системе

Добавление нового адаптера в серверную часть означает следующую последовательность действий:

1. добавление класса (C++), соответствующего типу адаптера, а также реализация в нём методов для управления адаптером, список которых приведён в разделе 5.4.2;

2. добавление создания экземпляра класса и вызовов методов в процедуру обработки RPC-запросов;

3. реализация процедуры получения и сериализации данных от адаптера для передачи их клиентской части.

6.2.2 Клиентская часть

В клиентской части требуется проводить разбор сообщений из данных, приходящих от Bus Pirate, во внутренние представления Opermon, а также реализовать необходимые виджеты для отображения данных и настройки параметров.

Добавление нового типа адаптера в Opermon означает следующую последовательность действий:

1. описание нового типа обменов в компоненте Tabexchange;

2. описание нового типа представления адаптера в клиентской части;

3. реализация процедуры получения данных от серверной части и создание экземпляров нового типа обменов на их базе;

4. добавление создания экземпляра класса представления адаптера в соответствующую процедуру в Opermon;

5. реализация необходимых для работы виджетов на базе предлагаемых в решении интерфейсов в компоненте sma;

6. добавление создания экземпляров классов виджетов и код для их базовой настройки в фабрику виджетов.

6.3 Добавление поддержки адаптера в Opermon

6.3.1 Серверная часть

Серверная часть решения реализована с использованием библиотеки QSerialPort [16] для упрощения работы с последовательным интерфесом адаптера.

Для реализации поддержки адаптера на стороне сервера были выполнены следующие действия:

- Создана обёртка для библиотеки QSerialPort для сборки в рамках системы сборки cvslvk.
- Описан класс BusPirate представления адаптера, наследуемый от класса CommonCard. В базовом классе реализованы основные функции представления адаптера, такие как установление TCP-соединения с клиентской частью, проверка наличия новых данных по файловому дескриптору и т.д. В классе BusPirate реализованы методы для инициализации последовательного порта, конфигурирования адаптера и получения данных от адаптера с выделением отдельных обменов.
- В обработчик RPC-запросов RpcServer::processRequest() добавлено создание объекта представления адаптера и вызовы соответствующих методов.

6.3.2 Клиентская часть

Для реализации поддержки адаптера на клиентской стороне были выполены следующие действия:

- Описано представление обмена в Tabexchange. Для этого создан класс BPI2CExchange, наследуемый от Exchange. Каждый обмен в среде Opermon представлен в виде экземпляра этого класса, при этом в базовом классе Exchange реализуется работа с параметрами обмена, общими для разных типов адаптеров (такие как время получения, порядковый номер обмена и т.д.), в классе BPI2CExchange реализуется разбор обмена из полученных "сырых" данных от адаптера и хранение отдельных полей обмена: адреса slave-устройства, режима доступа и полезной нагрузки.
- Описано представление адаптера и реализован интерфейс взаимодействия с серверной частью в Opermon. Для этого создан класс BPI2CMTCard, наследуемый от OperMon::CommonCard. В базовом классе реализуется идентификация конкретного адаптера, проверка соединения с представлением адаптера на удалённом агенте, а также вызовы основных методов в RPC. В классе BPI2CMTCard дополнительно реализуется начальная конфигурация адаптера на

удалённом агенте и разбор полученных данных с выделением обменов. Передача нового полученного обмена в обработку в среде Opermon происходит через Qt-сигнал addExchange(Exchange *).

- Реализовано представление адаптера в пользовательском интерфейсе в классе BPI2CChannelInfo, наследуемом от ChannelInfo. Представление реализует настройку параметров адаптера: в базовом классе реализуется настройка соединения с удалённым агентом (адрес хоста, номер адаптера в системе и т.д.), в наследнике реализуется тип роли адаптера (на сегодняшний день только чтение канала - Sniffer) и настройка параметров, специфичных для адаптера (включение питания целевого устройства, резисторы подтяжки и т.д.).

- Добавлено значение BPI2CChannel в список типов адаптеров в ChannelType.

- Реализованы классы виджетов для отображения данных обменов и настройки параметров в sma: BPI2CExchangeDetails, BPI2CExchangeInfo, BPI2CExchangeTableModel, BPI2CExchangeTab, BPI2CChannelAttributeModel и т.д. (с соответствующими базовыми классами ExchangeDetails, ExchangeInfo, ExchangeTableModel, ExchangeTab, ChannelAttributeModel и т.д.).

- Добавлено создание экземпляров классов виджетов и их начальная настройка в фабрику виджетов (в sma в класс ChannelFactoryImpl, реализующий интерфейс ChannelFactory).

Снимок экрана с окном получившегося интерфейса можно посмотреть в приложении А на рисунке 8.

7 Апробация

Решение было проверено и применено в рамках задачи отладки бортовой вычислительной системы мобильного робота [18].

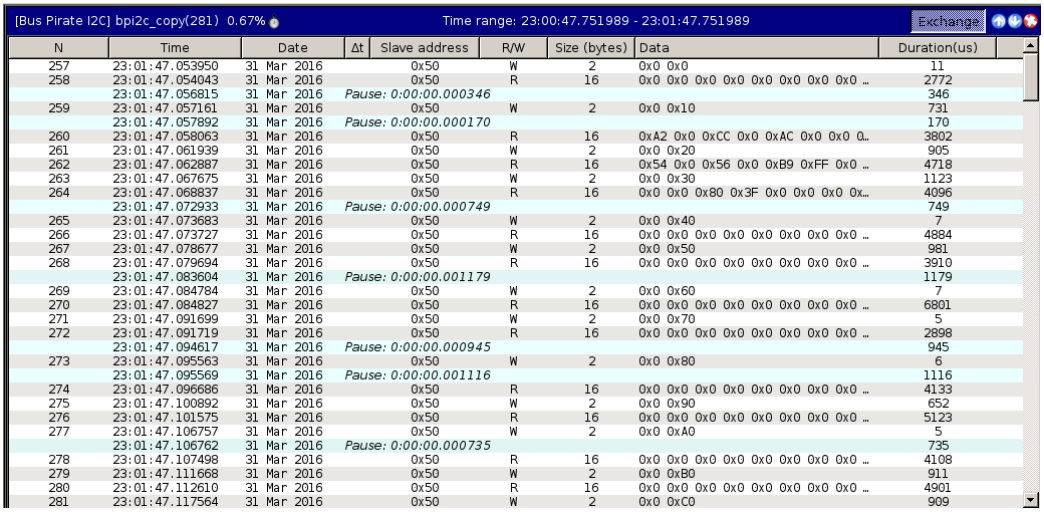
В рамках этой задачи требовалось получить данные об обменах на шине I2C между бортовым контроллером и EEPROM-памятью для хранения конфигурации мобильного робота. На шине регистрировалось большое количество обменов, так как работа с EEPROM в данной системе происходит в страничном режиме.

При считывании большого количества страниц возможностей консольного интерфейса явно не хватает, поскольку в результате мы получаем нагромождение из текста, где отдельные обмены сложно отделить визуально. По этой же причине возникают серьёзные трудности в навигации по считанной трассе.

Ниже приведены представления данных в интерфейсе командной строки адаптера Bus Pirate и в среде Opermon.

```
[\\0xa0+\\0x00+\\0x00+[\\0xa1+\\0x00+\\0x00+\\0x00+\\0x00+\\0x00+\\0
    x00+\\0x00+\\0x00+\\0x00+\\0x00+\\0x00+\\0x00+\\0x00+\\0x00+\\0
    x00+\\0x00-][\\0xa0+\\0x00+\\0x10+[\\0xa1+\\0xa2+\\0x00+\\0xcc
    +\\0x00+\\0xac+\\0x00+\\0x00+\\0x00+\\0x00+\\0x00+\\0x00+\\0x00
    +\\0x00+\\0x00+\\0x00+\\0x00-][\\0xa0+...
```

Листинг 1: Пример представления данных обменов в консольном интерфейсе Bus Pirate



N	Time	Date	Δt	Slave address	R/W	Size (bytes)	Data	Duration(us)
257	23:01:47.053950	31 Mar 2016		0x50	W	2	0x0 0x0	11
258	23:01:47.054043	31 Mar 2016		0x50	R	16	0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 …	2772
	23:01:47.056815	31 Mar 2016	*Pause: 0:00:00.000346*					346
259	23:01:47.057161	31 Mar 2016		0x50	W	2	0x0 0x10	731
	23:01:47.057892	31 Mar 2016	*Pause: 0:00:00.000170*					170
260	23:01:47.058063	31 Mar 2016		0x50	R	16	0xA2 0x0 0xCC 0x0 0xAC 0x0 0x0 0…	3802
261	23:01:47.061939	31 Mar 2016		0x50	W	2	0x0 0x20	905
262	23:01:47.062887	31 Mar 2016		0x50	R	16	0x54 0x0 0x56 0x0 0xB9 0xFF 0x0 …	4718
263	23:01:47.067675	31 Mar 2016		0x50	W	2	0x0 0x30	1123
264	23:01:47.068837	31 Mar 2016		0x50	R	16	0x0 0x0 0x80 0x3F 0x0 0x0 0x0 0x…	4096
	23:01:47.072933	31 Mar 2016	*Pause: 0:00:00.000749*					749
265	23:01:47.073683	31 Mar 2016		0x50	W	2	0x0 0x40	7
266	23:01:47.073727	31 Mar 2016		0x50	R	16	0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 …	4884
267	23:01:47.078677	31 Mar 2016		0x50	W	2	0x0 0x50	981
268	23:01:47.079694	31 Mar 2016		0x50	R	16	0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 …	3910
	23:01:47.083604	31 Mar 2016	*Pause: 0:00:00.001179*					1179
269	23:01:47.084784	31 Mar 2016		0x50	W	2	0x0 0x60	7
270	23:01:47.084827	31 Mar 2016		0x50	R	16	0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 …	6801
271	23:01:47.091699	31 Mar 2016		0x50	W	2	0x0 0x70	5
272	23:01:47.091719	31 Mar 2016		0x50	R	16	0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 …	2898
	23:01:47.094617	31 Mar 2016	*Pause: 0:00:00.000945*					945
273	23:01:47.095563	31 Mar 2016		0x50	W	2	0x0 0x80	6
	23:01:47.095569	31 Mar 2016	*Pause: 0:00:00.001116*					1116
274	23:01:47.096686	31 Mar 2016		0x50	R	16	0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 …	4133
275	23:01:47.100892	31 Mar 2016		0x50	W	2	0x0 0x90	652
276	23:01:47.101575	31 Mar 2016		0x50	R	16	0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 …	5123
277	23:01:47.106757	31 Mar 2016		0x50	W	2	0x0 0xA0	5
	23:01:47.106762	31 Mar 2016	*Pause: 0:00:00.000735*					735
278	23:01:47.107498	31 Mar 2016		0x50	R	16	0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 …	4108
279	23:01:47.111668	31 Mar 2016		0x50	W	2	0x0 0xB0	911
280	23:01:47.112610	31 Mar 2016		0x50	R	16	0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 …	4901
281	23:01:47.117564	31 Mar 2016		0x50	W	2	0x0 0xC0	909

Рис. 4: Пример представления данных обменов в интерфейсе Opermon

Очевидно, что работа с представлением данных в среде Opermon проще и информативней. Помимо содержимого обменов, которые, к тому же, оказываются разделёнными построчно, можно узнать длительность обмена и время его регистрации.

8 Предложения по рефакторингу

Актуальная версия Opermon имеет довольно сложный граф зависимостей от дополнительных компонентов. Несмотря на это, существующая архитектура решения остаётся недостаточно гибкой и не предоставляет возможностей по удобному разделению на компоненты.

Проведение рефакторинга требуется для того, чтобы получить возможность отделить ядро анализатора Opermon от интерфейсо-зависимых комопнентов и подготовить сборку Opermon только с поддержкой I2C для опубликования в открытом доступе.

Предложения по рефакторингу, приведённые в этой работе, нацелены на:

- **Возможность вариативной сборки**. В рамках Opermon реализована поддержка различных интерфейсов, при этом часть кода не предназначена для распространения. Необходимо явно разделить реализации поддержки различных интерфейсов и добавить возможность сборки только необходимых для конкретного заказчика компонентов. Более того, после этого может появиться возможность свободно распространять ядро Opermon, что повлияет на популярность решений лаборатории в сообществе.

- **Более явное выделение интерфейсо-специфичной части**. Возвращаясь к разделу 6.3.1, можно заметить, что актуальная версия решения имеет местами не очень очевидное разделение на файлы по компонентам. Также существуют общие процедуры, зависящие сразу от всех поддерживаемых интерфейсов, из-за чего возможно появление ошибок работы старых интерфейсов при добавлении новых.

- **Разбиение кода и собираемых компонентов на независимые модули**. Этот шаг позволит облегчить тестирование отдельных компонентов системы. Более того, использование системы подключаемых бинарных компонентов (плагинов) упростит распространение и поддержку Opermon для разных категорий пользователей, а также разработку дополнительных модулей для Opermon сторонними разработчиками.

8.1 Разбиение компонента sma

В актуальной версии решения реализация интерфейсо-зависимой части анализатора вынесена в подключаемый модуль sma. Тем не менее,

этот модуль включает в себя реализацию сразу всех поддерживаемых в Opermon интерфейсов. Исходя из целей рефакторинга, есть смысл перенести реализацию каждого из интерфейсов в отдельные динамически подключаемые модули. Для подгрузки модулей можно использовать, например, средство QLibrary [19], встроенное в фреймворк Qt. Динамическая подгрузка позволит переконфигурировать решение Opermon без необходимости повторной сборки, что также облегчит распространение модулей, созданных сторонними разработчиками.

Более того, следует перенести в эти модули объявление классов представления обменов из Tabexchange (файлы tabexchange/exchange.cpp и tabexchange/exchange.h).

8.2 Удаление интерфейсо-специфичной части из Opermon

Представления адаптеров объявлены и реализованы в исходном коде модуля Opermon (файлы opermon/cardagent.h и opermon/cardagent.cpp). Возможно, есть смысл также перенести эти представления в вышеописанные бинарные модули.

Также в исходном коде Opermon реализована фабрика представлений адаптеров, которую также стоит попытаться разбить с возможностью использования динамически подгружаемых бинарных модулей.

9 Заключение

В ходе выполнения курсовой работы были получены следующие результаты:

- проведён анализ существующих на рынке решений для анализа обменов на шинах данных, используемых во встраиваемых системах;
- выполнено сравнение этих решений с разрабатываемой в ЛВК средой Opermon;
- предложен и реализован проект инструмента для анализа обменов на шине I2C с использованием адаптера Bus Pirate;
- предложен проект рефакторинга существующей архитектуры среды Opermon с целью отделения интерфейсо-специфичных компонентов.

Предложения по дальнейшей работе:

- добавление в реализацию специальных форматов сообщений с учётом адреса устройства (например, для широко распространённых микросхем и сенсоров с интерфейсом I2C);
- добавление поддержки работы с другими интерфейсами, поддерживаемыми Bus Pirate (например, SPI или 1-Wire);
- реализация режимов работы Opermon как ведущего или ведомого устройства для активного анализа устройств или для имитационного моделирования во встраиваемых системах;
- реализация проекта рефакторинга и подготовка дистрибутива Opermon для опубликования в открытом доступе.

10 Литература

1. Sniffing I2C Traffic With a Bus Pirate [HTML] (http://www.digitalpeer.com/blog/sniffing-i2c-traffic-with-a-bus-pirate)
2. V.V. Balashov, V.A. Balakhanov, A.G. Bakhmurov, M.V. Chistolinov, P.E. Shestov, R.L. Smeliansky, N.V. Youshchenko. Tools for monitoring of data exchange in real-time avionics systems [DOC] (http://www.docfoc.com/download/documents/tools-for-monitoring-of-data-exchange-in-real-time-avionics-systems)
3. I2C-bus specification and user manual [PDF] (http://www.nxp.com/documents/user_manual/UM10204.pdf)
4. Bus Pirate I2C guide [HTML] (http://dangerousprototypes.com/bus-pirate-manual/i2c-guide/)
5. Arduino I2C sniffer project on Hackaday [HTML] (http://hackaday.com/2011/05/21/arduino-i2c-sniffer/)
6. Отладчик I2C, SPI, 1-Wire [HTML] (http://avrdevices.ru/otladchik-i2c-spi-1-wire/)
7. Bus Pirate documentation [HTML] (http://dangerousprototypes.com/docs/Bus_Pirate)
8. Beagle Protocol Analyzer User Manual [HTML] (http://www.totalphase.com/support/articles/200472426)
9. Data Center Software [HTML] (http://www.totalphase.com/products/data-center/)
10. Aardwark I2C/SPI Host Adapter User Manual [HTML] (http://www.totalphase.com/support/articles/200468316)
11. Control Center Software [HTML] (http://www.totalphase.com/products/control-center-serial/)
12. BitScope Logic [HTML] (http://www.bitscope.com/software/logic/)
13. Qxmlrpc library [Github] (https://github.com/commontk/qxmlrpc)
14. Redlab. Средство мониторинга и анализа мультиплексного канала информационного взаимодействия. Руководство оператора. [DOC]

15. Qt Home | Russian [HTML] (https://www.qt.io/ru/)

16. QSerialPort documentation [HTML] (http://doc.qt.io/qt-5/qserialport.html)

17. Bus Pirate I2C Sniffer macro [HTML] (http://dangerousprototypes.com/docs/Bus_Pirate_I2C#I2C_Bus_Sniffer_macro)

18. Rokkit team mobile robot development blog [HTML] (http://rokkit.su/)

19. QLibrary class documentation [HTML] (http://doc.qt.io/qt-4.8/qlibrary.html)

A Изображения и схемы

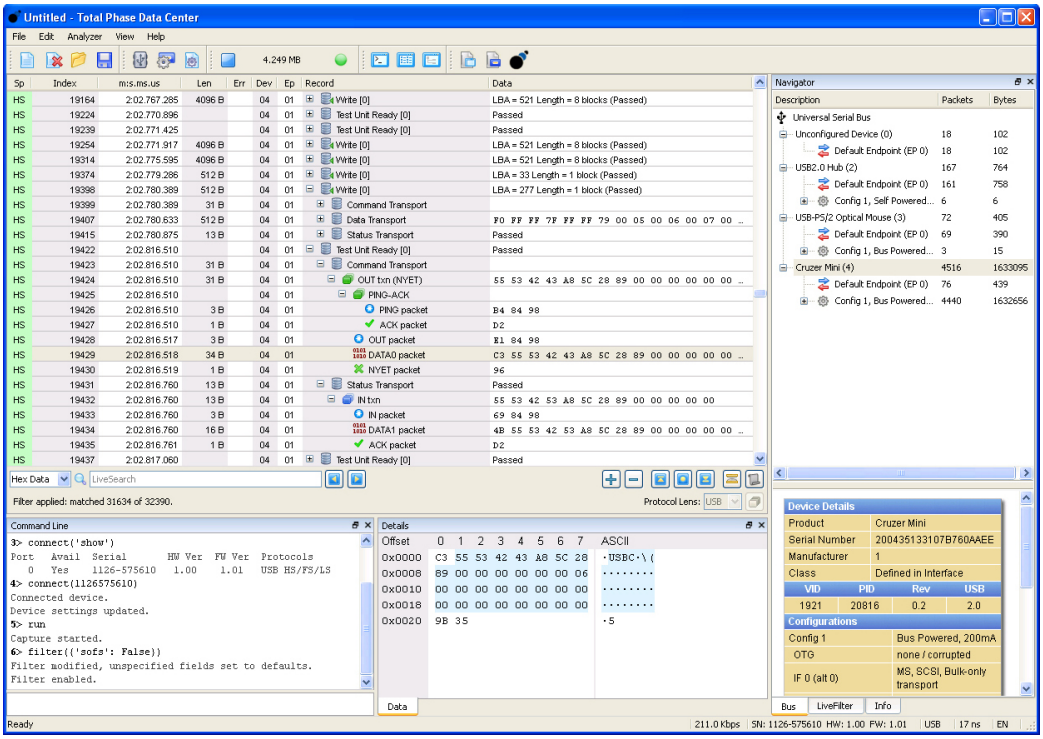


Рис. 5: Снимок экрана рабочего окна Total Phase Data Center

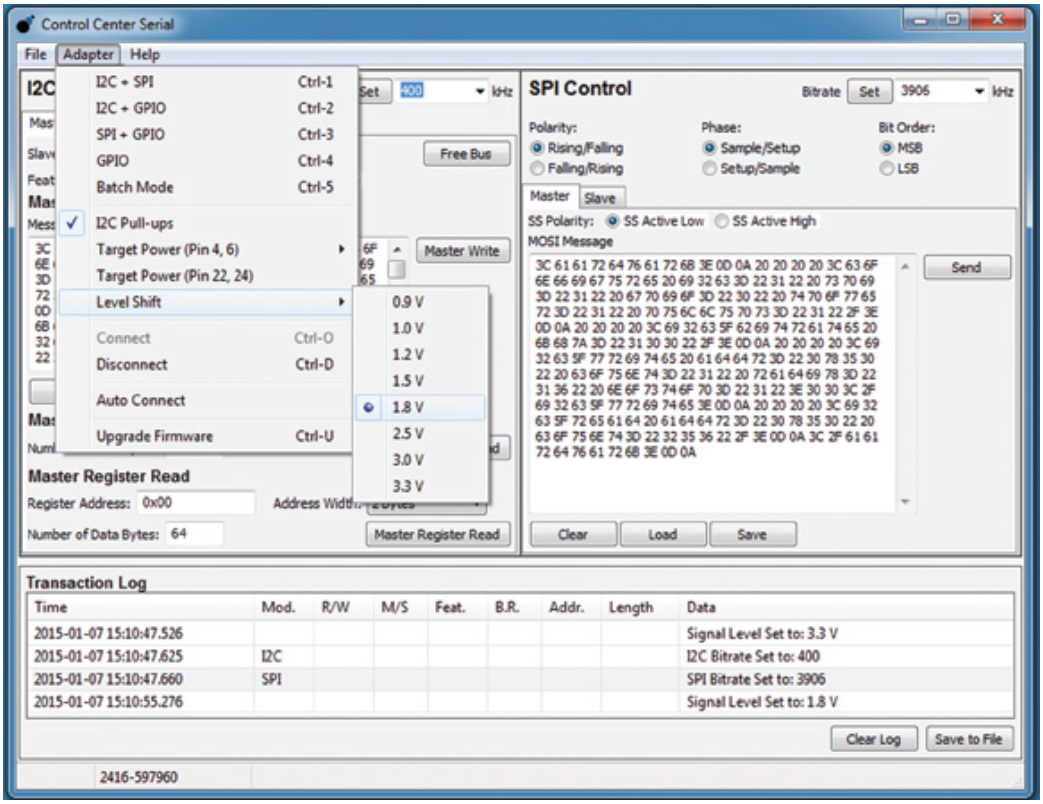


Рис. 6: Снимок экрана рабочего окна Total Phase Control Center

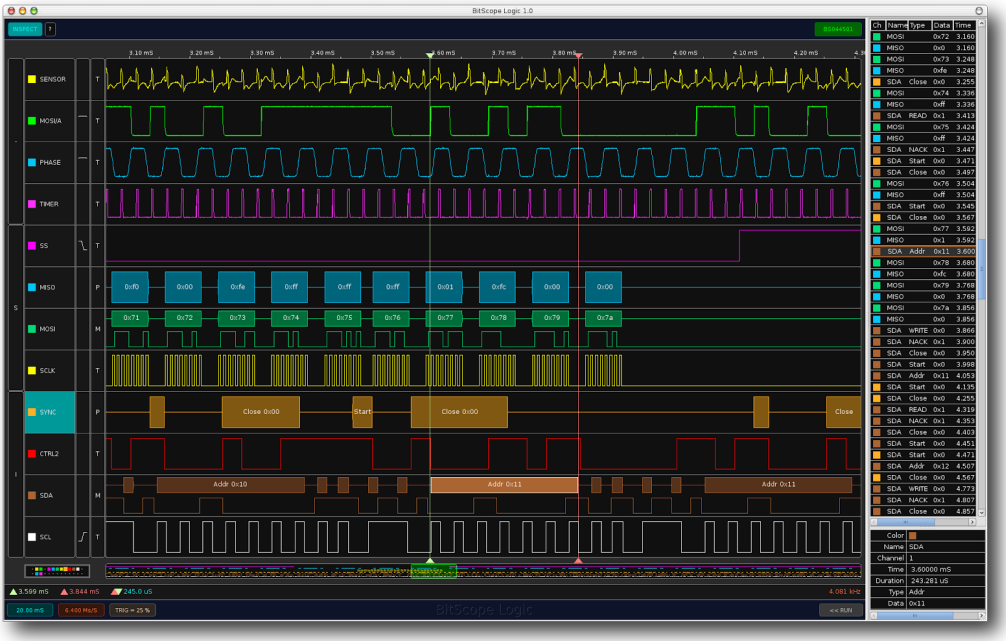


Рис. 7: Снимок экрана рабочего окна BitScope Logic

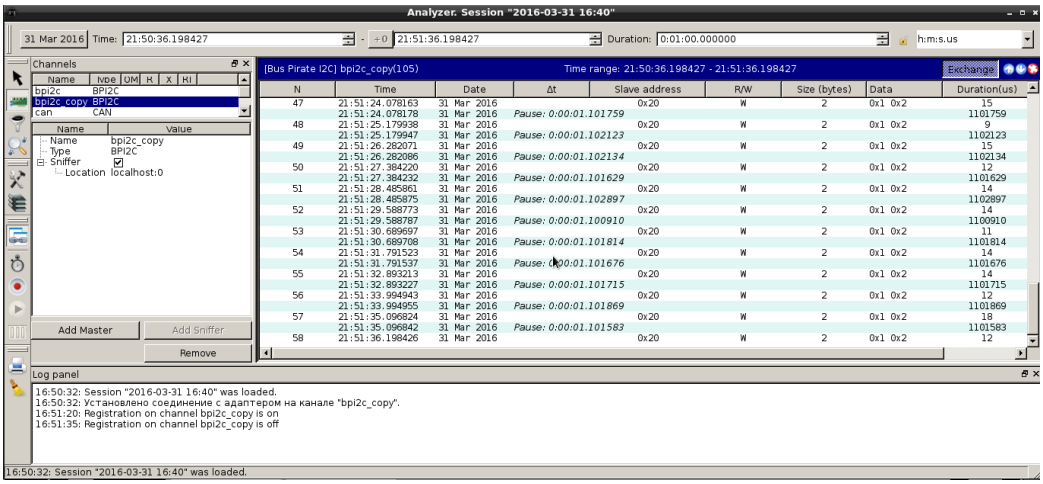


Рис. 8: Снимок экрана рабочего окна Opermon с поддержкой Bus Pirate I2C