ISSN 2413–9599 (Print)
ISSN 2541–870X (Online)

www.radioprom.org

DOI: 10.21778/2413-9599-2019-29-1-16-23
УДК 004.451.86

Отладка и мониторинг прикладных программ в операционной системе реального времени «БагрОС-4000» на базе архитектуры «Эльбрус»

Р.Г. Гордиенко[1], О.Г. Федоренко[1], А.А. Демидов[2], А.В. Федоров[2,3]

[1] ПАО «Авиационная холдинговая компания “Сухой”», Москва, Россия
[2] АО «МЦСТ», Москва, Россия
[3] ПАО «ИНЭУМ им. И.С. Брука», Москва, Россия

Статья посвящена проблемам мониторинга и отладки процессов операционных систем, эффективность которых в варианте операционных систем жесткого реального времени не допускает остановки для анализа состояния программного обеспечения и/или аппаратуры. В работе описана концепция системы отладки и мониторинга, разработанной с учетом этой особенности в опытно-конструкторском бюро «Сухой» для операционной системы жесткого реального времени «БагрОС-4000» на архитектурной платформе «Эльбрус» совместно со специалистами АО «МЦСТ». Обсуждается метод безостановочного мониторинга и сбора данных в процессах жесткого реального времени в многопроцессных многомодульных системах. Изложен подход к управлению отладочными целями в терминах исходного кода с использованием спецификации отладочной информации DWARF. Описан переход от инструментальной машины к встроенному в целевую ЭВМ серверу системы. Дано обоснование применения клиент-серверной архитекторы в системе отладки и мониторинга для «БагрОС-4000». Проведен сравнительный анализ ключевого функционала системы отладки и мониторинга с существующими системами отладки, рассмотрены ключевые аспекты архитектуры COM, обсуждается проектирование машино-зависимого интерфейса, необходимого для интеграции в «БагрОС-4000» независимых аппаратных платформ при реализации системы на интегрированном модуле авионики бортового комплекса. Результаты апробации системы отладки и мониторинга проанализированы на предмет эффективности в сравнении с классическим приемом использования отладочных консольных печатей при отладке операционной системы реального времени. Большинство приведенных решений универсальны и были успешно апробированы при использовании других микропроцессорных платформ на многопоточных прикладных программах операционных систем реального времени, исполняемых на многоядерных процессорах, включая платформы MIPS, Power PC, Intel.

Ключевые слова: операционная система реального времени, система отладки и мониторинга, точка наблюдения, сетевые компоненты системы отладки и мониторинга, обработчик исключения, машинно-зависимая часть сервера, архитектура «Эльбрус»

Для цитирования:
Отладка и мониторинг прикладных программ в операционной системе реального времени «БагрОС-4000» на базе архитектуры «Эльбрус» / Р.Г. Гордиенко, О.Г. Федоренко, А.А. Демидов, А.В. Федоров // Радиопромышленность. 2019. Т. 29, № 1. С. 16–23. DOI: 10.21778/2413-9599-2019-29-1-16-23

Debugging and monitoring of application programs in the BagrOS-4000 real-time operation system based on the Elbrus architecture

R.G. Gordienko[1], O.G. Fedorenko[1], A.A. Demidov[2], A.V. Fedorov[2,3]

[1] Aviation Holding Company «Sukhoi» PJSC, Moscow, Russia
[2] MCST JSC, Moscow, Russia
[3] Brook INEUM PJSC, Moscow, Russia

The article is concerned with the problems of monitoring and debugging of operating system processes, the effectiveness of which in the hard real-time operating system version does not allow any stopping to analyze the state of software and/or hardware. The paper describes the concept of a debugging and monitoring system developed taking into account this feature in the Sukhoi design bureau for the BagrOS-4000 hard real-time operating system on the Elbrus architectural platform together with the specialists of MCST JSC. The method of non-stop monitoring and data collection in hard real-time processes in the multiprocess multimodular systems is discussed. An approach to the management of debugging targets in terms of source code using the DWARF debugging information specification is presented. The transition from the instrumental machine to the system server built into the target computer is described. Given the rationale for the use of client-server architecture in the debugging and monitoring system for BagrOS-4000. A comparative analysis of the key functionality of the debugging and monitoring system with the existing debugging systems has been carried out; the key aspects of the DMS architecture have been considered. The design of a machine-dependent interface required for the integration of the independent hardware platforms into the BagrOS-4000 system when implementing the system on an integrated avionics module of the onboard complex is discussed. The results of testing of the debugging and monitoring systems are analyzed in terms of efficiency versus the classical method of using the debug console prints when debugging a real-time operating system. Most of the above solutions are universal and have been successfully tested using other microprocessor platforms on multi-threaded application programs of real-time operating systems running on multi-core processors, including the MIPS, Power PC, Intel platforms.

Keywords: real-time operating system, debugging and monitoring system, watching point, network components of the debugging and monitoring system, exception handler, machine-dependent server part, the Elbrus architecture

For citation:
Gordienko R.G., Fedorenko O.G., Demidov A.A., Fedorov A.V. Debugging and monitoring of application programs in the BagrOS-4000 real-time operation system based on the Elbrus architecture. Radiopromyshlennost, 2019, vol. 29, no. 1, pp. 16–23 (In Russian). DOI: 10.21778/2413-9599-2019-29-1-16-23

Введение

Логика взаимодействия процессов операционной системы жесткого реального времени (ОС РВ) предусматривает мониторинг и тестирование исключительно во время выполнения этих процессов. Их останов для отладки недопустим, т.к. это приводит к рассогласованию работы смежных процессов и невозможности добиться поставленных целей отладки. Данный фактор имеет определяющее значение в ряде важнейших областей применения, в частности в авионике, и в статье он представлен как проблема первого плана. Проблемы второго плана связаны с атрибутикой классических инструментальных средств отладки (консольный режим, привязанность к инструментальной машине и необходимость ее монопольного использования), а также их трудоемкостью и маломобильностью при формировании отладочных целей (установки адресов точек мониторинга и наблюдаемых параметров).

Совокупность этих проблем обусловила постановку представленной в статье работы, которая была проведена на интегрированной платформе, включающей специфицированную операционную систему «БагрОС-4000» и процессоры семейства «Эльбрус».

Особые требования к средствам отладки в «БагрОС-4000», базирующейся на спецификациях ARINC-653 и POSIX, сформировали необходимость в специальном подходе – организации мониторинга и отладки без полного останова прикладных процессов, но с прерыванием работы по инструкции BREAK и чтением данных мониторинга «на лету» в обработчике исключения по точке останова. Данный подход лег в основу новой системы отладки и мониторинга.

Выбор в пользу отечественного производителя вычислительных средств был обусловлен передовыми для российской микроэлектроники показателями их модульного ряда, которые в совокупности соответствуют основным требования применения в жестком реальном времени. В частности, это касается производительности, энергопотребления, архитектурно поддерживаемых средств защищенного программирования, конструктивных и температурных характеристик.

Стратегия отладки и мониторинга

Система отладки и мониторинга в ОС РВ «БагрОС-4000» состоит из двух основных частей, каждая из которых решает проблемы соответственно первого и второго планов. Структура системы отладки и мониторинга представлена на рис. 1.

Серверная часть реализует основную стратегию системы – организует постоянные точки наблюдения watchpoints, которые характеризуются списком наблюдаемых параметров. Они базируются на инструкциях BREAK и выдают клиенту значения заказанных в списке данных при каждом прохождении точки. Это происходит во время выполнения системных и прикладных процессов без их останова. Вместо останова выполняется вход в обработчик исключения (именно такой механизм выбран в архитектуре MIPS [1, 2]) по BREAK, считывание значений мониторинговых данных и выход из обработчика, после чего данные параллельным потоком отправляются клиенту. Межпроцессные взаимодействия при этом могут растягиваться по времени (на период выполнения обработчика), но не останавливаться. Этот принцип решает определенную выше проблему первого плана.

DWARF (debugging with attributed record formats) – формат отладочной информации [3, 4], генерируемой компилятором GCC [5], который используется системой отладки и мониторинга для автоматического вычисления адресов и типов точек наблюдения, а также самих наблюдаемых параметров. DWARF-reader – утилита-анализатор для чтения отладочной информации и переработки ее в технических целях системы.

Сетевой клиент системы отладки и мониторинга – потребитель данных мониторинга – получает их от сервера, устанавливая с ним TCP/IP-соединение. Фактически клиент соединяется с целевой машиной, т.к. сервер системы функционирует на ней как часть операционной системы. Такая клиент-серверная архитектура решает проблему второго плана – получается современная реализация клиента с динамичным пользовательским GUI-интерфейсом, включающим следующий функционал:

- открытие или закрытие проекта отладки;
- просмотр структуры папок и файлов проекта;
- открытие файла проекта в окне просмотра исходного кода с подсветкой его синтаксиса;
- интерактивная (с графической обратной связью) постановка или снятие точки наблюдения по тексту исходного кода;
- интерактивный выбор в окне исходного кода имени переменной и добавление ее в список мониторинга;
- удаление переменной из списка мониторинга по ее имени;
- интерактивный выбор и мониторинг в реальном времени подготовленных списков мониторинга;
- фиксация в файлах трассировки и отображение в отдельном окне истории изменений переменных, полученных в процессе их мониторинга.

Задачу повышения эффективности клиентских инструментов (из проблематики второго плана) в данном случае удалось решить с помощью

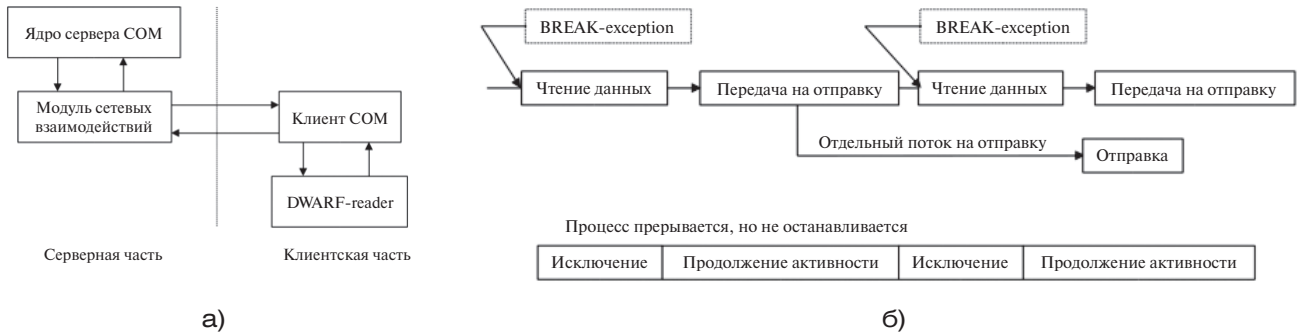

Рисунок 1. Принцип реализации механизмов системы отладки и мониторинга (COM) в «БагрОС-4000»: а – общая структура системы; б – параллельная активность механизмов системы отладки и мониторинга в операционной системе
Figure 1. The principle of implementation of the debugging and monitoring system (COM) mechanisms in BagrOS-4000: a – overall structure of the system; б – parallel activity of the debugging and monitoring system mechanisms in the operating system

интерактивной настройки компонентов процесса отладки, которые доступны в современном графическом интерфейсе. Весь процесс настройки списков идет в терминах исходного кода программного обеспечения, работа которого подлежит мониторингу. Все адреса, типы переменных, виды локации данных вычисляются автоматически.

Основной эффект, ожидаемый от внедрения системы отладки и мониторинга в «БагрОС 4000», – это возможность мобильно организовать и поставить на безостановочный мониторинг списки глобальных, локальных и регистровых переменных, формальных параметров функций, а также параметров следующих типов:

- скалярных целых и вещественных, знаковых и беззнаковых переменных;
- указателей;
- массивов;
- структур;
- объединений, включая составные композиции типов (массивы структур, указатели на структуру, массивы указателей, массивы указателей на структуру и др.).

Стратегия системы, безусловно, не идеальна: обработчик точки наблюдения в любом случае потребляет временные ресурсы и смещает картину временных взаимодействий процессов от точки, когда система работает без системы. Но это время ничтожно мало в сравнении с обычным остановом процесса при классической отладке.

Тем не менее эффективность данной стратегии можно легко увидеть на примере простого использования точек наблюдения системы вместо вставок printf в код отлаживаемых проектов, а именно:

- использование точек наблюдения не требует перекомпиляции исходного кода, как в случае printf;
- точки наблюдения системы отладки и мониторинга не изменяют размер отлаживаемого образа в отличие от printf;
- исполнение printf требует значительного процессорного времени (printf всегда ресурсоемкий вызов), а обработчик системы выполняется во много раз быстрее;
- по завершению отладки не нужно убирать из кода вызовы printf.

Принятые к реализации в системе отладки и мониторинга подходы обладают достаточной эффективностью в сравнении с существующими решениями в части отладки, среди которых можно выделить достаточно близкий по области применения отладчик GDB. Аналитические заключения на этот счет собраны в таблице. Анализ, проведенный относительно точки наблюдения системы, показывает, что, хотя в ряде средств отладки, например в GDB, есть возможность устанавливать точку контроля прохождения трассы tracepoints, она не соответствует требованию проблематики первого плана – сбору значений наблюдаемых параметров в нужных точках без остановки процесса. Что касается мониторинга и отладки многомодульной архитектуры, отмечено следующее: не существует отладчика, обрабатывающего как единое целое поле из нескольких аппаратных модулей с функцией удаленного (по всем модулям) мониторинга списка наблюдаемых параметров.

Ключевые аспекты архитектуры и реализации системы отладки и мониторинга

Главная идея безостановочных точек наблюдения повлияла на все детали как серверной, так и клиентской частей. Архитектура серверной части представлена на рис. 2.

На реализацию функционала здесь работают три параллельных потока: чтения и отправки данных в модуле сетевого взаимодействия, а также конфигурирования (перестроения) учетных данных сервера.

Поток отправки данных освобождает обработчик исключения от временных затрат при передаче клиенту системы данных мониторинга. Реализация самого процесса отправки примечательна тем, что сервер имеет возможность отправить клиенту данные большой длины такими, какими они будут на момент входа в обработчик исключения. Для этого используется специальный буфер копий, куда перед отправкой помещаются данные обработчиком. Если данные больше размера буфера копий, то они отправятся без предварительной фиксации, т.е. «как есть» на момент начала отправки. Объем буфера данных конфигурируется, по умолчанию он равен 128 Кбайт.

Поток конфигурирования используется для реализации в отложенном режиме операций добавления (изменения, удаления) точек наблюдения и наблюдаемых параметров. Это позволяет управлять целями отладки и мониторинга «на ходу», после того как что-то уже было поставлено под наблюдение.

Все процессы мониторинга приводит в действие обработчик исключения по команде BREAK. Он исполняется в режиме ядра, поэтому и память под учетные данные выделяется ему как системная, чтобы обработчик мог ее штатно использовать. Сами данные организованы так, чтобы сократить время поиска по адресу в списке установленных точек наблюдения и адресов переменных:

- для навигации в массивах адресов применяется алгоритм упорядоченного поиска;

Таблица. Сравнение выбранных характеристик ОС РВ «БагрОС» с существующими решениями в части системы отладки и мониторинга
Table. Comparison of selected features of real-time operating system «BagrOS» with the existing solutions in terms of debugging and monitoring system

Подход, выбранный в «БагрОС-4000» / The approach implemented in BagrOS-4000	Аналоги / Analogues	Преимущества / Advantages
Точки наблюдения системы отладки и мониторинга	Нет явных аналогов	Возможность вести наблюдение и трассировку в процессах ОС РВ без их остановки
Клиент-серверная архитектура системы	Есть в отладчике GDB в виде удаленных клиентов классического отладчика	Открытый клиент с возможностью удаленного мониторинга списков наблюдаемых параметров
Платформенная универсальность системы	В GDB есть экземпляры под Unix/Linux-платформы и для Windows в варианте WinGW	Выбранная реализация в среде Eclipse на языке Java позволяет рассматривать варианты Windows и Linux/Unix как единый проект в исходных кодах на Java
Функционал графического клиента системы	Есть частичный аналог в GDB при использовании пакетов эмуляции графического интерфейса, но без аппаратно-зависимых скинов	Возможность интегрировать окна мониторинга и окна исходного кода отлаживаемого процесса. Автоматическое вычисление по исходному коду адреса точки наблюдения и параметров. Возможность по дополнительному техническому заданию развивать интерфейс до специализированных аппаратно-зависимых скинов
Мониторинг и отладка многомодульной архитектуры	Нет явных аналогов	Возможность в одном клиенте системы отладки и мониторинга наблюдать и сопоставлять значения по спискам параметров из аппаратных модулей, разных в том числе по типу и модели

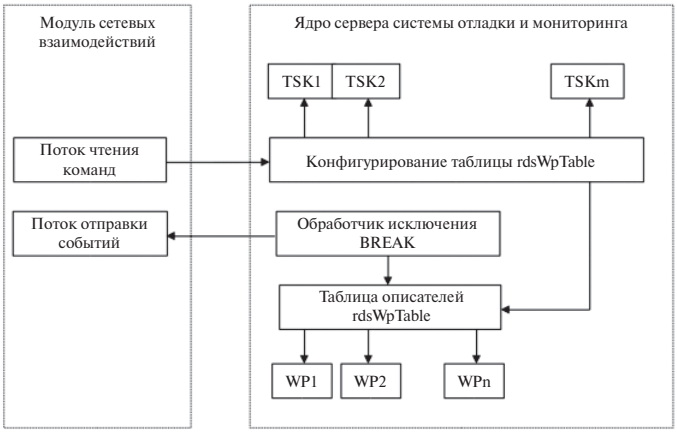

Рисунок 2. Серверная часть системы отладки и мониторинга: TSK1, TSK2, TSKm – набор описателей задач на конфигурирование учетной информации сервера (задачи на постановку и снятие точки наблюдения, добавление и удаление наблюдаемых параметров и пр.); WP1, WP2, WPn – описатели точки наблюдения (watchpoint), которые принадлежат к структуре данных rdsWpTable – таблице учетных записей точек мониторинга во внутренних данных системы COM
Figure 2. The server part of the debugging and monitoring system: TSK1, TSK2, TSKm – a set of task descriptors for configuring server account information (tasks for setting and removing the observation points, adding and deleting the monitored parameters, etc.); WP1, WP2, WPn – watchpoint descriptors that belong to the data structure of the rdsWpTable – table of accounts of monitoring points in the internal data of the debugging and monitoring system

- для копирования наблюдаемых данных в буфер отправки используются специально организованные таблицы быстрого обращения, их обход позволяет минимизировать время считывания данных для мониторинга.

Цель всех этих оптимизаций – сократить время прерывания процесса, поставленного на мониторинг. Оптимизирована была также архитектура сервера в части разграничения связей ядра системы отладки и мониторинга и поддерживаемой аппаратной части.

Сервер имеет четко очерченный интерфейс для адаптации машинно-зависимых компонентов. Специальные программные адаптеры системы были разработаны на процессорах «Эльбрус-2С», «Эльбрус-4С», в составе которых прошли продолжительную опытную эксплуатацию, а также на других платформах.

Что касается клиентского программного обеспечения, то здесь ключевой особенностью является мультиплатформенность. Клиент системы разработан на языке Java для платформы Eclipse в виде дополнительного модуля расширения (plugin); клиентское программное обеспечение способно запускаться на разных операционных системах общего назначения без перекомпиляции.

Такие подходы демонстрируют современность решения и позволяют уверенно говорить о перспективах развития системы отладки и мониторинга в будущем.

Машинно-зависимая часть

Программный адаптер системы отладки и мониторинга, реализованный на платформе «Эльбрус», соответствует интерфейсу машинно-зависимой части сервера и поддерживает следующие функции:

- dmtaSetBreakpoint – установка точки останова;
- dmtaBreakpointOff – снятие точки останова;
- dmtaRegSize – запрос размера регистра;
- dmtaRegsCount – запрос количества регистров;
- dmtaReadCommand – чтение команды из памяти;
- dmtaIsBreakCommand – проверка, является ли команда точкой останова;
- dmtaGetCommandSizeByVAddr – запрос размера команды по указанному адресу.

При адаптации машинно-зависимой части системы на платформу «Эльбрус» возникли следующие задачи:

1. Перевод сообщений от клиента к серверной части из big endian в little endian.
2. Реализация интерфейсных функций, а именно:
   - получение кода инструкции по виртуальному адресу;
   - запись инструкции setsft по виртуальному адресу;
   - обработка прерывания software trap;
   - удаление инструкции setsft и возврат оригинальной;
   - вычисление следующей исполняемой инструкции.

Между клиентом и сервером устанавливается TCP/IP-соединение, что подразумевает перевод сообщений из big endian в little endian в силу особенностей архитектуры процессора. Данная задача была решена с помощью builtin-функций компилятора __builtin_bswap и встроена в систему отладки и мониторинга.

Особенность программной реализации адаптера системы для платформы «Эльбрус» состоит в том, что инструкции процессоров этого семейства упаковываются в широкие команды, которые одновременно дешифруются и параллельно выполняются каждая в своем отдельном конвейере [6]. Это является спецификой архитектуры, которая заключается в возможности при компиляции программы предопределить оптимальное распараллеливание вычислительного процесса.

В адаптере для платформы «Эльбрус» постанов и снятие breakpoint выполняются корректно исходя из длины команды, заменяемой на инструкцию setsft. На уровне реализации машинно-зависимого кода серверной части постановка breakpoint реализована как запись инструкции setsft по виртуальному адресу в USER CODE-сегменте, который система вычислит исходя из выбранной строки в файле на Си и данных из DWARF-таблицы, в которой для каждого объектного файла содержится соответствие <строка исходного кода:: адрес в исполняемом модуле>. Поставить setsft клиент может благодаря команде dmtaSetBreakpoint, реализованной в серверной части. Запись кода инструкции setsft происходит путем вычисления физического адреса из виртуального и непосредственного копирования кода инструкции по физическому адресу. Аналогичным образом происходят удаление и возврат оригинальной инструкции (dmtaBreakpointOff и dmtaReadCommand).

Когда исполняется инструкция setsft, возникает исключительная ситуация software trap, и вызывается обработчик _rdsWatchpoint, который считывает наблюдаемые параметры из таблицы и возвращает код оригинальной инструкции на место, а также вычисляет адрес следующей исполняемой инструкции, куда записывается код setsft. Это необходимо, чтобы оригинальная инструкция выполнилась и не исчез сам breakpoint. После исполнения

широкой команды, где находится оригинальная инструкция, выполняется следующая команда, где уже установлена инструкция с кодом setsft. Далее вызывается обработчик, из которого код оригинальной инструкции снова заменится на код setsft, а текущая исполняемая инструкция станет прежней.

Поиск следующей исполняемой инструкции после breakpoint заключается в вычислении размера широкой команды. Это можно сделать, используя слог-заголовок HS, который является первым в команде. Он содержит информацию о длине и структуре команды. Поле lng указывает на количество двойных слов в структуре команды.

Система откладки и мониторинга в составе ОС РВ «БагрОС-4000» после адаптации машинно-зависимой части показала высокую эффективность на процессорах семейства «Эльбрус».

Перспективы развития системы отладки и мониторинга в «БагрОС-4000»

Основная перспектива системы, являющаяся также главным резервом развития, – это автоматизация мониторинга, тестирования и трассировки.

В этом смысле принципиальным понятием процесса автоматизации в системе является профиль мониторинга. Это набор данных, хранимых в файле и описывающих совокупность точек наблюдения, их адреса и связанные с ними группы списков наблюдаемых параметров. Профиль мониторинга сохраняется из IDE (integrated development environment – интегрированная среда разработки) в файл, а потом применяется вновь при повторной отладке исполняемого кода путем открытия из файла. Это минимальная автоматизация.

На данный момент IDE в системе отладки и мониторинга развивается как клиентское мультиплатформенное программное обеспечение в среде Eclipse [7, 8], разработанное на языке Java [9]. Эти же технологии планируется применить в развитии IDE в будущем.

Следующий уровень развития – это формирование проекта автоматизации мониторинга и тестирования – функционала в IDE, который добавляет к профилю скрипты, управляющие постановкой (снятием) точек наблюдения, добавляет (удаляет) параметры в списках наблюдения, управляет активностью тех или иных точек наблюдения в зависимости от значения нужных наблюдаемых переменных. На базе этих возможностей эффективным решением является создание библиотеки исходного кода макротестов, использующих встроенный скриптовый язык манипуляции системы и доступа к значениям наблюдаемых переменных.

Этот же подход будет эффективным и для создания макросценариев управления трассировкой в ОС РВ. Созданный однажды сценарий можно сохранить в файл, а затем открыть и применить к трассируемому прикладному коду. Это обеспечит автоматический старт кода, создание трассы, обработку и сохранение трассировки в файле результата.

Выводы

1. Рассмотрено решение безостановочного мониторинга для многопроцессных многомодульных систем в ОС РВ «БагрОС-4000», позволяющее без пересборки проекта управлять отладочными целями в терминах исходного кода, используя спецификацию DWARF отладочной информации компиляторов языка высокого уровня.
2. Подход апробирован на высокопроизводительной мультипроцессорной платформе «Эльбрус» с практическим подтверждением эффективности рассматриваемого подхода.
3. Всесторонне рассмотрена архитектура системы отладки и мониторинга как программного решения.
4. Проанализированы интерфейс системы откладки и мониторинга с машино-зависимой частью отладчика, интерфейс с клиентом системы и интерфейс с модулем анализа DWARF-информации.
5. Сделано сравнение выбранных характеристик ОС РВ «БагрОС» с существующими решениями.
6. Проанализированы перспективы автоматизации мониторинга, тестирования и трассировки в системе отладки и мониторинга.
7. Обсуждается работа по созданию IDE для ОС РВ «БагрОС-4000» на базе клиентского мультиплатформенного программного обеспечения в среде Eclipse.

Определенную ценность на текущем этапе развития средств поддержки отечественной авионики представляет и тот факт, что система отладки и мониторинга разработана на отечественном отраслевом предприятии для современной развивающейся отечественной ОС РВ «БагрОС-4000» в интеграции с отечественной высокопроизводительной мультипроцессорной платформой «Эльбрус».

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. *IDT MIPS* microprocessor family software developer's guide [Электронный ресурс]. California, IDT Inc., 2005. URL: https://www.idt.com/document/mas/idt-mips-software-developers-guide-vol-1 (дата обращения: 06.01.2019).
2. *Heinrich J.* MIPS R4000 microprocessor user's manual. 2nd ed. MIPS Technologies Inc., 1994.

3. *The DWARF* debugging standard version 5 [Электронный ресурс]. 2017. URL: http://www.dwarfstd.org/Dwarf5Std.php (дата обращения: 06.01.2019).
4. *Random* testing for concurrency compiler bugs. Indian Institute of Technology Kanpur, 2012.
5. *Гриффитс А.* GCC. Полное руководство. М.: ООО «ТИД “ДС”», 2004. 624 с.
6. *Ким А.К., Перекатов В.И., Ермаков С.Г.* Микропроцессоры и вычислительные комплексы семейства «Эльбрус». СПб.: Питер, 2013. 272 с.
7. *Хорстманн К., Корнелл Г.* Java 2. Библиотека профессионала. Т. 1. Основы. 8-е изд. М.: Вильямс, 2012. 816 с.
8. *Silva V.* Practical eclipse rich client platform projects. Apress, 2009. 343 с.
9. *Шилдт Г.* Полный справочник по Java. Java SE 6 edition. 7-е изд. М.: Вильямс, 2009. 1040 с.

REFERENCES

1. *IDT MIPS microprocessor family software developer's guide.* California, IDT Inc., 2005. Avaliable at: https://www.idt.com/document/mas/idt-mips-software-developers-guide-vol-1 (accessed 06.01.2019).
2. Heinrich J. *MIPS R4000 microprocessor user's manual.* 2nd ed. MIPS Technologies Inc., 1994.
3. *The DWARF debugging standard version 5.* 2017. Avaliable at: http://www.dwarfstd.org/Dwarf5Std.php (accessed 06.01.2019).
4. *Random testing for concurrency compiler bugs.* Indian Institute of Technology Kanpur, 2012.
5. Griffith A. GCC. *Polnoe rukovodstvo* [GCC. The complete reference]. Moscow, Ltd DiaSoft Publ., 2004, 624 p. (In Russian).
6. Kim A.K., Perekatov V.I., Ermakov S.G. *Mikroprotsessory i vychislitelnye kompleksy semeistva «Elbrus»* [«Elbrus» family microprocessors and computing systems]. Saint-Petersburg: Piter Publ., 2013, 272 p. (In Russian).
7. Hortsmann C., Cornell G. *Java 2. Biblioteka professionala. T. 1. Osnovy* [Java 2. Professional library. Vol. 1. The basics]. 8th ed. Moscow, Vilyams Publ., 2012, 816 p. (In Russian).
8. Silva V. *Practical eclipse rich client platform projects.* Apress, 2009. 343 p.
9. Schildt H. *Polnyj spravochnik po Java. Java SE 6 edition.* [Comprehensive guide to Java. Java SE 6 edition]. 7th ed. Moscow, Vilyams Publ., 2009, 1040 p. (In Russian).

ИНФОРМАЦИЯ ОБ АВТОРАХ

Гордиенко Роман Григорьевич, начальник сектора, ПАО «Авиационная холдинговая компания “Сухой”», 125284, Москва, ул. Поликарпова, д. 23Б, а/я 604, тел.: +7 (499) 550-01-06, e-mail: rg_gordienko@mail.ru.
Федоренко Олег Григорьевич, к.т.н., ведущий инженер, ПАО «Авиационная холдинговая компания “Сухой”», 125284, Москва, ул. Поликарпова, д. 23Б, а/я 604, тел.: +7 (499) 550-01-06, e-mail: olegf1974@inbox.ru.
Демидов Арсений Александрович, инженер-программист, АО «МЦСТ», 119334, Москва, ул. Вавилова, д. 24, тел.: +7 (499) 135-33-21, e-mail: Arseniy.A.Demidov@mcst.ru.
Федоров Александр Вадимович, старший инженер-программист, АО «МЦСТ», ПАО «ИНЭУМ им. И.С. Брука», 119334, Москва, ул. Вавилова, д. 24, тел.: +7 (499) 135-33-21, e-mail: Alexander.V.Fedorov@mcst.ru.

AUTHORS

Roman G. Gordienko, head of sector, PJSC Aviation Holding Company «Sukhoi», 23B, ulitsa Polikarpova, Moscow, 125284, Russia, tel.: +7 (499) 550-01-06, e-mail: rg_gordienko@mail.ru.
Oleg G. Fedorenko, Ph.D. (Engineering), leading engineer, PJSC Aviation Holding Company «Sukhoi», 23b, ulitsa Polikarpova, Moscow, 125284, Russia, tel.: +7 (499) 550-01-06, e-mail: olegf1974@inbox.ru.
Arseniy A. Demidov, software engineer, JSC MCST, 24, ulitsa Vavilova, Moscow, 119334, Russia, tel.: +7 (499) 135-33-21, e-mail: Arseniy.A.Demidov@mcst.ru.
Aleksandr V. Fedorov, senior software engineer, JSC MCST, PJSC Brook INEUM, 24, ulitsa Vavilova, Moscow, 119334, Russia, tel.: +7 (499) 135-33-21, e-mail: Alexander.V.Fedorov@mcst.ru.

Поступила 30.10.2018; принята к публикации 20.12.2018; опубликована онлайн.
Submitted 30.10.2018; revised 20.12.2018; published online.