УДК 004.4

**Оценка эффективности подходов разработки программного обеспечения для микроконтроллеров ARM архитектура серии STM32**

Вершинин Евгений Владимирович

Кандидат физико-математических наук, доцент, заместитель директора по НИР, заведующий кафедрой ИУК5 "Системы обработки информации" КФ МГТУ им. Н.Э. Баумана, Калужский филиал федерального государственного бюджетного образовательного учреждения высшего образования "Московский государственный технический университет имени Н.Э. Баумана (национальный исследовательский университет)", 24800, Российская Федерация, город Калуга, улица Баженова, дом 4

[vershinin@bmstu.ru](mailto:vershinin@bmstu.ru)

Цимерман Илья Семенович

Студент 4 курса кафедры ИУК5 "Системы обработки информации" КФ МГТУ им. Н.Э. Баумана, Калужский филиал федерального государственного бюджетного образовательного учреждения высшего образования "Московский государственный технический университет имени Н.Э. Баумана (национальный исследовательский университет)", 24800, Российская Федерация, город Калуга, улица Баженова, дом 4

[ilya-tsimerman@mail.ru](mailto:ilya-tsimerman@mail.ru)

**Evaluation of the effectiveness of software development approaches for ARM microcontrollers STM32 series architecture**

Vershinin Evgeny Vladimirovich

Candidate of Physical and Mathematical Sciences, Associate Professor, Deputy Director for Research, Head of the Department of IUK5 "Information Processing Systems" of the Bauman Moscow State Technical University, Kaluga Branch of the Federal State Budgetary Educational Institution of Higher Education "Bauman Moscow State Technical University (National Research University)", 24800, 4 Bazhenova Street, Kaluga, Russian Federation

[vershinin@bmstu.ru](mailto:vershinin@bmstu.ru)

Tsimerman Ilya Semenovich

4th year student of the Department of IUK5 "Information Processing Systems" of the Bauman Moscow State Technical University, Kaluga Branch of the Federal State Budgetary Educational Institution of Higher Education "Bauman Moscow State Technical University (National Research University)", 4 Bazhenova Street, Kaluga, 24800, Russian Federation

[ilya-tsimerman@mail.ru](mailto:ilya-tsimerman@mail.ru)

В работе проводится сравнение эффективности выполнения программ, реализованных при помощи CMSIS и генератора кода периферии STM32CubeMX. Сравниваются значения, заносимые в регистры периферии, генерируемые код-конфигуратором, со значениями рекомендованными производителем продукции. Тестируется при помощи статического анализатора код, который генерирует STM32CubeMX. Рассматриваются плюсы и минусы использования код-конфигуратора и используемых им библиотек. The paper compares the performance of programs implemented using CMSIS and the STM32CubeMX peripheral code generator. The values entered into the peripheral registers generated by the code configurator are compared with the values recommended by the product manufacturer. The code generated by STM32CubeMX is tested using a static analyzer. The pros and cons of using the code configurator and the libraries used by it are considered.

STM32, ARM, генератор кода инициализации STM32CubeMX, CMSIS, HAL, LL

STM32, ARM, STM32CubeMX initialization code generator, CMSIS, HAL, LL

**Введение.** Самой популярной архитектурой процессоров, на базе которых на сегодняшний день функционирует подавляющее большинство электронных изделий, является ARM (Advanced RISC Machine или Acorn RISC Machine). ARM архитектура объединяет в себе семейство как 32, так и 64-разрядных микропроцессорных ядер, разработанных и лицензируемых компанией ARM Limited. Самым популярным представителем микроконтроллеров на базе процессора ARM является Семейство 32-разрядных микроконтроллеров STM32 от компании STMicroelectronics, которые также выпускают и ПО для своей продукции.

Целью данной работы является оценка эффективности подходов разработки ПО для микроконтроллеров ARM архитектуры, серии stm32.

**Обзор технологий средств разработки.** Keil uVision – это среда разработки, представляющая собой набор утилит для выполнения полного комплекса мероприятий по написанию программного обеспечения для микроконтроллеров.

Данная среда выбрана по следующим причинам:

1) работает с устройствами STMmicroelectronics на ядрах Cortex-M0/M0+, бесплатный набор инструментов для микроконтроллеров семейства STM32F0/G0/L0;

2) взаимодействует с конфигуратором STM32CubeMX;

3) присутствует возможность интеграции в неё статического анализатор кода PVS-Studio [1];

4) наличие функций оптимизации кода по памяти, а также скорости работы.

STM32CubeMX – кодогенератор с графическим интерфейсом от компании STMmicroelectronics, который максимально упрощает настройку микроконтроллеров STM32. Этот инструмент имеет широкий функционал и позволяет:

1) создавать и редактировать проекты для микроконтроллеров STM32 с последующей генерацией С-кода для конкретных IDE (IAR™ EWARM, Keil™MDK-ARM, Atollic® TrueSTUDIO и AC6 System Workbench (SW4STM32));

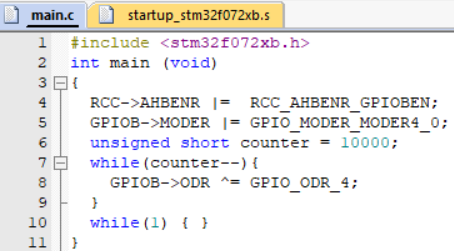
2) настраивать параметры ПО промежуточного уровня (файловая система, стеки протоколов, операционные системы и т.д.) и периферийные блоки с помощью утилиты Perepherial and middleware Wizard (вкладка Configuration). При этом для генерации С-кода могут использоваться различные библиотеки от STMicroelectronics (HAL или LL).

**Сравнительный анализ подходов разработки программного обеспечения для микроконтроллеров серии STM32.** Первым подходом разработки является прямая работа с регистрами микроконтроллера при помощи CMSIS (Common Microcontroller Software Interface Standard – общий стандарт программного интерфейса микроконтроллера) [2].

Вторым – использование генератора кода периферии STM32CubeMX, который подключает независимые друг от друга библиотеки – HAL либо LL [3].

1. Сравним ранее перечисленные подходы по памяти и по скорости работы на основе программы, которая 10000 раз меняет уровень напряжения на ножке микроконтроллера stm32f072c8 [4]. Данная задача выбрана для того, чтобы результаты экспериментов не зависели от сторонних сигналов внешних устройств, исключив узкие места в периферии:

1) создадим проект без сторонних библиотек и напишем код только при помощи CMSIS (рис. 1);



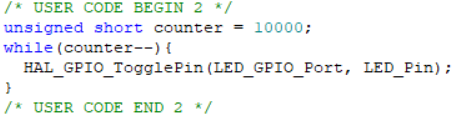
**Рис. 1.** Код программы, написанный при помощи CMSIS

2) установим оптимизацию кода по памяти level 0 и замерим осциллографом время, за которое уровень сигнала на ножке изменится 10000 раз;

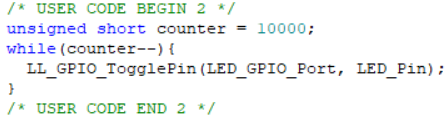
3) тоже самое проделаем с оптимизации по памяти – level 3;

4) и оптимизацией по времени.

Аналогичные тесты проведём с использованием код-конфигуратора STM32CubeMX с использованием библиотек HAL и LL (рис. 2, 3).



**Рис. 2**. Код программы, написанный при помощи библиотеки HAL



**Рис. 3**. Код программы, написанный при помощи библиотеки LL

Для реализации задачи используя HAL или LL потребовалось меньше строк кода, чем при использовании CMSIS. Помимо этого, можно отметить, что код, использующий библиотеки, прочитать и понять проще, чем код, использующий CMSIS. Результаты эксперимента занесены в таблицу 1.

**Табл. 1.** Сравнительные характеристики результатов тестирования

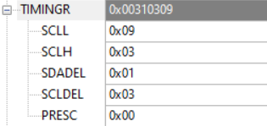
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Объекты исследования | | Характеристики | | | |
| Размер кода, байт | Время сборки, мс | | Время выполнения, мс |
| CMSIS | Оптимизация level 0 | 492 | | 1 | 21,20 |
| Оптимизация level 3 | 464 | | 3 | 21,20 |
| Оптимизация по времени | 452 | | 1 | 14,95 |
| LL | Оптимизация level 0 | 1384 | | 5 | 21,20 |
| Оптимизация level 3 | 1308 | | 5 | 21,20 |
| Оптимизация по времени | 1312 | | 6 | 21,20 |
| HAL | Оптимизация level 0 | 4456 | | 25 | 31,40 |
| Оптимизация level 3 | 3460 | | 28 | 31,40 |
| Оптимизация по времени | 3672 | | 26 | 31,40 |

Анализ результатов показывает, что максимально эффективный код по памяти и времени выполнения синтезирован без использования код-конфигуратора. После оптимизации по времени не удалось ускорить выполнения программ, с подключенными библиотеками. После оптимизации по памяти level 0, level 3 библиотека LL показала те же результаты скорости выполнения программы, что и с прямая работа с регистрами микроконтроллера при помощи CMSIS. HAL уступает LL по всем параметрам.

2. Проведём ещё один тест для код-конфигуратора. Возьмём уже более сложную периферию, например I2C (Inter-Integrated Circuit или последовательная асимметричная шина), и настроим скорость её работы:

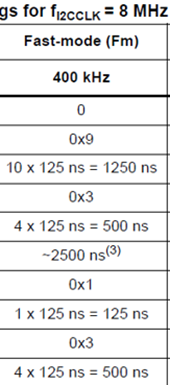
1) изменим значение частоты на 400Кгц, всё остальное конфигуратор сделал за нас;

2) переместим проект в Keil uVision и в режиме отладки (debag) посмотрим состояние регистра TIMINGR (рис. 4);



**Рис. 4**. Окно значения регистров периферии I2C в режиме debag в Keil uVision

3) сравним эти значения со значениями в таблице из Reference manual для нашего микроконтроллера (stm32f072c8) [5] с тактовой частотой 8МГц и параметром 4КГц (рис. 5).



**Рис. 5**. Таблица примера конфигурации регистра TIMINGR периферии I2C, рекомендованная производителем продукции

Значения полностью совпали с теми, что указаны в Reference manual, конфигуратор периферии задал самые оптимальные значения регистра.

3. Наконец протестируем при помощи статического анализатора нейтральный код на наличие ошибок, который генерирует STM32CubeMX, не внося в него никаких изменений.

Исходный код библиотек HAL и LL разработан в строгом соответствии с ANSI-C и полностью документирован и соответствует правилам MISRA-C 2004 [3, c. 1], что делает его независимым от инструментов разработки. Это значит, что генерируемый код CubeMX полностью соответствует правилам кодирования MISRA C®:2004. (мировые рекомендации по безопасному и надежному применению как встроенных систем управления, так и автономного программного обеспечения) [6].

Проведём анализ по следующим ошибкам по уровню опасности:

1) High – включает в себя предупреждения с максимальным уровнем достоверности. Такие предупреждения часто указывают на ошибки, требующие немедленного исправления;

2) Medium – содержит менее достоверные предупреждения, на которые все же стоит обратить пристальное внимание;

3) Low – предупреждения с минимальным уровнем достоверности, указывающие на несущественные неточности в коде. Среди таких предупреждений обычно велик процент ложных срабатываний.

Будем использовать характеристики ошибок:

1) General (GA) – диагностика общего плана. Основной набор диагностических правил PVS-Studio;

2) Optimization (ОP) – диагностика микрооптимизации. Указания по стойкости и безопасности кода;

3) MISRA – стандарт диагностики, разработанные в соответствии с MISRA.

Результаты экспериментов занесены в таблицы 2 и 3.

**Табл. 2.** Результаты тестирования библиотеки LL

|  |  |  |  |
| --- | --- | --- | --- |
| Характеристика ошибки | Степень ошибки | | |
| High | Medium | Low |
| GA | 4 | 2 | 11 |
| OP | 0 | 0 | 0 |
| MISRA | 0 | 335 | 408 |

**Табл. 3.** Результаты тестирования библиотеки HAL

|  |  |  |  |
| --- | --- | --- | --- |
| Характеристика ошибки | Степень ошибки | | |
| High | Medium | Low |
| GA | 3 | 26 | 7 |
| OP | 0 | 1 | 0 |
| MISRA | 0 | 2323 | 629 |

При использовании библиотеки HAL возникает в разы больше ошибок по сравнению с LL. Возникающие ошибки снижают скорость и эффективность работы, затрудняют отладку кода. В некоторых случаях возможна ситуация, когда отловить ошибку невозможно.

**Заключение.** Наиболее эффективный подход к написанию кода с точки зрения использования памяти, скорости выполнения и степени контроля является прямая работа с регистрами устройства в соответствии с документацией производителя. В то же время такой подход требует высокой квалификации программиста и увеличивает время разработки.

Использовать код-конфигуратор (без генерации кода) стоит для экономии времени, чтобы быстро узнать значения, заносимые в регистры, при инициализации периферии. Также код-конфигуратор можно использовать до этапа создания печатной платы, чтобы убедиться в отсутствии конфликта между программной и аппаратной реализациями.

HAL легка в освоении, но скрывает особенности работы микроконтроллера. Библиотека подойдёт для создания конфигураций сложной периферии, например USB. Также HAL можно использовать в небольших проектах критичных ко времени разработки.

LL требует глубоких знаний архитектуры микроконтроллера и периферийных устройств и сокращает время разработки только для высококвалифицированных программистов.

Список литературы

1. Интеграция PVS-Studio в uVision Keil. / PVS-Studio LLC Inc - 03 August 2021 / [Электронный ресурс]. - Режим доступа: URL: <https://pvs-studio.com/en/blog/posts/cpp/0847/> (дата обращения 01.10.2022).

2. ARM технологии CMSIS // Общий стандарт программного интерфейса микроконтроллера. / [Электронный ресурс]. - Режим доступа: URL: <https://arm-software.github.io/CMSIS_5/General/html/index.html> (дата обращения 01.10.2022).

3. Руководство пользователя, UM1785. // Описание STM32F0 HAL и низкоуровневых драйверов. / STMicroelectronics Inc, DocID026525 Rev 6, February 2020.

4. Техническое описание. // STM32F072x8, STM32F072xB. / STMicroelectronics Inc, DS9826 Rev 6, September 2019.

5. Справочное руководство, RM0091. // STM32F0x1/STM32F0x2/STM32F0x8 advanced Arm®-based 32-bit MCUs / STMicroelectronics Inc, RM0091 Rev 10, May 2022.

6. MISRA-C:2004 Guidelines for the use of the C language in critical systems / MIRA Limited Inc, October 2004.