7. Якимов И.М., Кирпичников А.П., Трусфус М.В., В.В. Мокшин // Вестник Технологического университета. – 2017. - 20, 15. – С. 118-122.

В.С. Сивков

АНАЛИЗ РАСПРЕДЕЛЕНИЯ ПАМЯТИ МИКРОКОНТРОЛЛЕРА ПРИ СОЗДАНИИ СИСТЕМ РЕАЛЬНОГО ВРЕМЕНИ

(Поволжский государственный университет
телекоммуникаций и информатики)

Одним из актуальных направлений развития информационных систем является повсеместное внедрение малогабаритных вычислительных устройств. Связано это, прежде всего, с развитием сенсорных систем, систем «умный дом», «умный город», «умная планета», и других подобных направлений. В большинстве своем, подобные малогабаритные устройства изготовлены на базе микроконтроллеров и работают в масштабе реального времени. Решая задачу создания программного обеспечения для таких устройств, исследователи сталкиваются с проблемой подбора оптимального архитектурного решения, с точки зрения ряда параметров. К таким параметрам можно отнести объем памяти, которая займет программа в устройстве, возможности по масштабированию программы, энергопотребление устройства.

Если расположить архетипы программ для подобных устройств по возрастанию возможностей масштабирования системы, то может получиться диаграмма подобная представленной на рисунке 1.

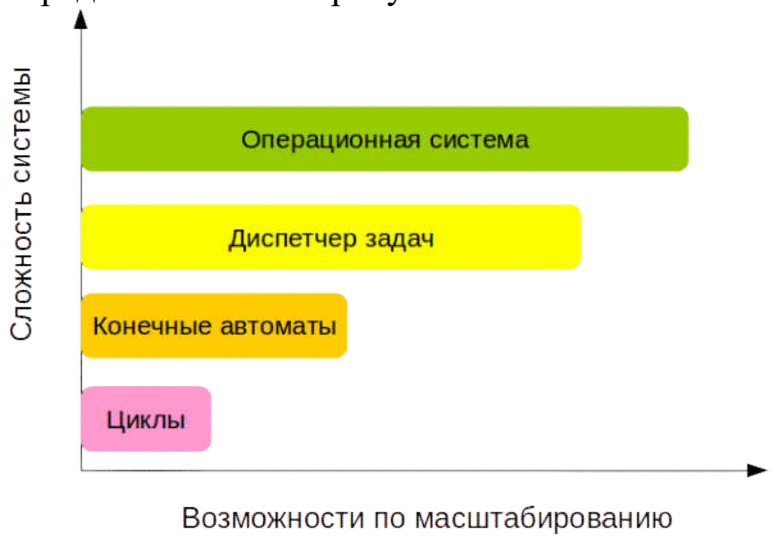

Рис. 1. Диаграмма архетипов систем реального времени.

Циклы или суперциклы (superloops) применяются для систем малого уровня сложности, обычно возможности по масштабированию систем на циклах достаточно скромные. Однако такие программы, как правило, занимают

меньше всего памяти. При грамотном подходе, конструкции на циклах могут показывать максимальное быстродействие, и минимальные накладные расходы.

Конечные автоматы применяются обычно для систем малой и средней сложности. Основное достоинство данного архетипа — возможность решать разноплановые задачи при минимальных накладных расходах вычислительных ресурсов. Однако чрезмерное усложнение системы, влечет за собой сильное усложнение самого конечного автомата, тем самым увеличивая время реализации и поддержки программного обеспечения. Данный фактор является основным ограничителем возможностей по масштабированию системы на конечных автоматах.

Для повышения возможностей по масштабированию систем, необходимы несколько иные конструкции, предоставляющие программисту удобный инструментарий выполнения задач. К таким инструментам можно отнести диспетчер задач и операционную систему. Однако применение усложненных сервисных программ (диспетчер или операционная система являются обслуживающей частью) приводит к увеличенному расходу ресурсов аппаратного обеспечения (процессорное время, память).

В данной работе рассматриваются результаты оценки расхода памяти микроконтроллера при создании систем реального времени на базе циклов и нескольких операционных систем реального времени (ОСРВ).

В качестве аппаратной платформы будем использовать микроконтроллер STM32F103C8T6, распаянный на отладочной плате (рисунок 2). Объем памяти под программу (flash) у данного устройства составляет 64 килобайта. Рассмотрим расход этого ресурса для одной и той же задачи, при использовании различных сервисных программ.

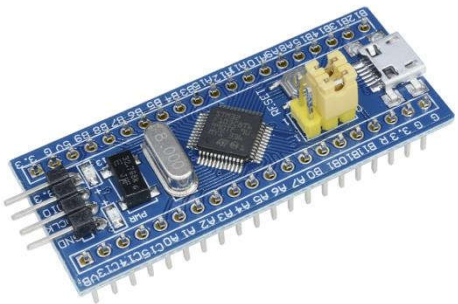

Рис. 2. Отладочная плата на базе микроконтроллера STM32F103C8T6

В качестве тестовой задачи, реализуем генератор импульсов с частотой 1 Гц. Наиболее рациональным, с точки зрения потребления ресурсов, решением данной задачи будет использование одного из аппаратных таймеров, входящих в состав микроконтроллера. С точки зрения интерфейса работы с начинкой микроконтроллера, наиболее эффективным, по производительности, будет непосредственное обращение к регистрам устройства. Однако такой подход

требует больших временных затрат, которые не всегда оправданы. Вот почему в большинстве случаев, используют специальные библиотеки, упрощающие доступ к периферии устройства. В нашем случае, такой библиотекой будет Cortex Microcontroller Software Interface Standard (CMSIS) [1].

Первый вариант реализации поставленной задачи (генератор), предполагает установку минимального набора средств разработки под данный тип микроконтроллеров (toolchain). Сам текст программы может занять до 30 строчек кода (стандартный стиль), большую часть кода займет инициализация периферии микроконтроллера и реализация грубой функции временной задержки. После компиляции, один из вариантов такой программы на языке Си занял 4040 байт.

Для следующего варианта реализации программы будем использовать ОСРВ FreeRTOS. Данная операционная система предоставляет программистам варианты вытесняющей и кооперативной многозадачности, базовый набор механизмов синхронизации и межзадачного взаимодействия [2]. FreeRTOS достаточно компактна, имеет открытый исходный код и распространяется по лицензии MIT, что несомненно делает ее удобной для использования в различных проектах. Первоначально необходимо загрузить исходные коды системы и настроить проект приложения (автор использовал вариант без лишних надстроек — файлы makefile). Сам текст программы может занять около 50 строчек кода, как и в предыдущем случае, большая часть программы это инициализация периферии и другие сервисные функции. После компиляции данный вариант программы на Си занял 9700 байт.

Далее рассмотрим реализацию нашей задачи в рамках одной из набирающих популярность в среде разработчиков ОСРВ Arm Mbed. Mbed OS - это операционная система с открытым исходным кодом для платформ, использующих микроконтроллеры ARM [3]. Одно из направлений применения таких микроконтроллеров - устройства Интернета вещей (IoT): маломощных, малогабаритных устройств, которые должны подключаться к Интернету. Mbed OS обеспечивает уровень абстракции для микроконтроллеров, на которых он работает, так что разработчики могут сосредоточиться на написании приложений верхнего уровня на C/C ++. Интересной особенностью является то, что приложения Mbed OS можно повторно использовать на любой Mbed-совместимой платформе. Данная ОСРВ выпускается под лицензией Apache 2.0, что делает ее пригодной к использованию в личных и коммерческих проектах. Подготовка проекта в данном случае занимает минимальное время (из рассмотренных в данной работе вариантов). Текст программы занимает около 20 строчек стандартным стилем. После компиляции данный вариант программы занял 39200 байт.

Попробуем сделать некоторые выводы по полученным результатам. По объему занимаемой памяти, как и прогнозировалось, лидером стал вариант задачи с использованием библиотеки CMSIS (~6% от общего объема flash). Неплохой результат показала ОСРВ FreeRTOS (16%), вариант с Mbed OS оказался самым объемным (~61% - рисунок 3).

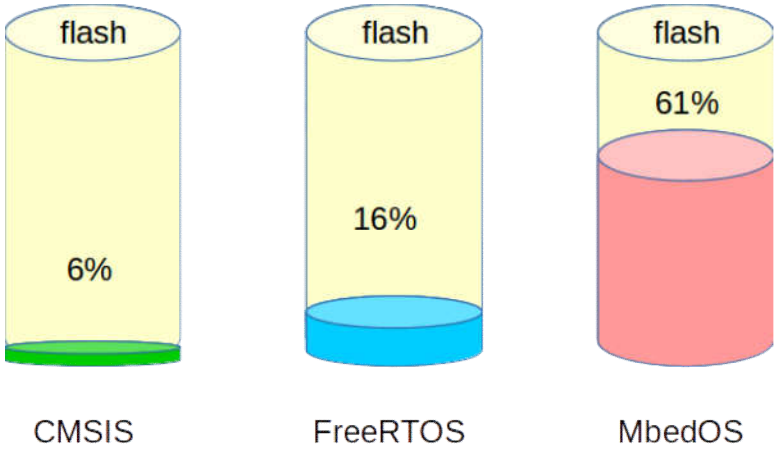

Рис. 3. Объем занимаемой flash памяти экспериментальных программ.

На первый взгляд может показаться, что MbedOS употребляет чрезмерное количество памяти микроконтроллера, однако это расплата за удобный и простой интерфейс для разработчика к аппаратуре. Так же остается открытым вопрос оптимизации проекта под конкретную задачу, возможно есть способы сократить объем занимаемой памяти для такой простой программы, однако данный вариант поставил бы MbedOS в более выигрышные условия, что явно противоречит задачам данного исследования. С точки зрения соотношения объем памяти — возможности масштабирования, наиболее привлекательным выглядит вариант с FreeRTOS, однако нужно помнить, что работа с периферийным оборудованием, драйвера для внешних устройств и интерфейсов, разработчик должен реализовать самостоятельно.

Вариант использования библиотеки CMSIS наиболее «экономный» с точки зрения занимаемой памяти, однако нужно помнить, что данный вариант пригоден в основном для простых систем, и трудоемок в масштабировании.

Литература

1. Cortex Microcontroller Software Interface Standard [Электронный ресурс] // URL: https://developer.arm.com/tools-and-software/embedded/cmsis

2. The FreeRTOS™ Kernel [Электронный ресурс] // URL: https://www.freertos.org/

3. Mbed OS [Электронный ресурс] // URL: https://www.mbed.com/en/platform/mbed-os/