Н.А.ТЫМАНОВИЧ[1], Ю.А.СКУДНЯКОВ[1]

# ПРОГРАММИРОВАНИЕ НА ЯЗЫКЕ RUST В EMBEDDED-СИСТЕМАХ

[1]*Учреждение образования «Белорусский государственный университет информатики и радиоэлектроники», г. Минск, Республика Беларусь*

**Rust** – это системный язык программирования, использование которого сосредоточено на трёх задачах: безопасность, скорость и параллелизм[1]. С его помощью решаются эти задачи без сборщика мусора, что делает его полезным в ряде случаев, когда использование других языков было бы нецелесообразно: при встраивании в другие языки, при написании программ с особыми пространственными и временными требованиями, при написании низкоуровневого кода, такого как драйверы устройств и операционные системы.

**Cargo** – это пакетный менеджер языка **Rust**. **Cargo** создает каркас проекта, следит за зависимостями, собирает и компилирует создаваемый проект.

**Crate.io** – репозиторий , с которого **cargo** может скачивать и устанавливать пакеты, библиотеки и готовые приложения.

Для начала программирования микроконтроллеров требуется скачать необходимый пакет, реализующий линкер-скрипт, самостоятельно заполнив лишь минимальную часть, связанную со спецификой программируемого «железа», или линкер-скрипт самостоятельно, скачав необходимый компилятор под целевую платформу средствами утлиты **rustup** и имея программатор и программы для работы с ним.

Когда компилируется исходный файл, на выходе формируется файл, который типично содержит в себе несколько секций с данными.

Четыре самые распространенные секции это:

3. **.text** – скомпилированный машинный код или секция исполняемых инструкицй;
4. **.data** – глобальные переменные;
5. **.rodata** – аналог **.data** для неизменяемых данных;
6. **.bss** – глобальные переменные, инициализироанные нулями.

Далее линкер (компоновщик) собирает каждый объектный файл, сортируя все элементы по таблицам в исполняемый файл. Обычно **cargo** или комплиятор **rust'a** может самостоятельно правильно скомпоновать и собрать проект, однако во встраиваемых системах нужны некоторые параметры, такие как: начало ОЗУ, начало флеша, если таковой есть, таблица векторов и т.п..

Таблица векторов – подсекция во флеш-памяти, которая содержит указатели на функции (обработчики), которые будут вызываться аппаратным прерыванием при настроенных программистом событиях.

Программу достаточно запрограммировать во флеш, чтобы микроконтроллер мог ее выполнять. Из-за задержек обращения во флеш-память рекомендуется для увеличения

производительности выгружать прошивку в ОЗУ. Программы для микроконтроллеров имеют некоторую специфику: периферия (аппаратные блоки) не работают по умолчанию и могут быть использованы только записью данных для настройки в специальные регистры

периферии. Перечень этих регистров, а также варианты использования описаны в документах **datasheet** и/или **reference manual**.

Периферия и таблица векторов могут быть как у производителя, так и в виде

архитектурных решений (вместе с такими архитектурами, как **risk-v** и **arm**, где часть таблицы векторов и периферии имеют одинаковые параметры и способы управления вне зависимости от конкретного производителя). После создания проекта следует запрограммировать микроконтроллер. Это делается связкой debug'гер + программатор; например: GDB (клиент отладки) + Open OCD (виртуальный сервер отладки) + st link (программатор) или BMP (программатор) + GDB. Таким образом, реализуется базовая работа с микроконтроллерами на языке программирования **Rust**.

ЛИТЕРАТУРА

1. https://rurust.github.io/rust_book_ru. – Дата доступа: 10.09.2020.