Драйверная модель Zephyr (RTOS)

Парьев Сергей Евгеньевич
АО «Лаборатория Касперского»
125212, г. Москва, Ленинградское шоссе, д. 39А, стр. 2
Sergey.Paryev@kaspersky.com

Введение

В 2025 году мы провели большое обзорное исследование-сравнение 21-й микроконтроллерной RTOS и обнаружили, что бесспорным лидером по многим факторам является Zephyr. Причем по популярности у разработчиков эта операционная система обогнала предыдущего лидера (FreeRTOS) буквально в последние годы. В нашем исследовании все RTOS в итоге разбились на два класса – 1) RTOS представляет собой фактически только ядро и 2) “полноценная RTOS” с драйверами, сервисами, полноценным SDK для прикладной разработки и др. FreeRTOS является классическим представителем первого класса, а Zephyr относится ко второму. В данном докладе мы рассмотрим только один из крайне важных факторов, который позволил Zephyr выбиться вперед, а именно, её драйверную модель (в англоязычном варианте “device-driver model”).

Основные отличия микроконтроллеров от микропроцессов

Для целей дальнейшего изложения приведем основные отличия микроконтроллеров от микропроцессоров (т.к. большинство специалистов хорошо знакомо с микропроцессорами): назначение, состав аппаратных блоков, размеры ОЗУ и ППЗУ, цена, электропотребление, тактовая частота, используемые модули защиты памяти и др. Эти отличия существенно влияют на архитектуру и функциональность операционных систем, которые работают на микроконтроллерах. Микроконтроллер представляет собой фактически законченный “компьютер” с крайне ограниченными ресурсами.

Назначение драйверной модели

У драйверной модели можно выделись три основные цели: 1) облегчение написания драйверов за счет наличия стандартизированных API и, обычно, готовых примеров под разные классы устройств; 2) повышение удобства работы с аппаратурой и переносимости кода для прикладного разработчика и 3) унификация и упрощение работы ОС в сценариях, когда требуется согласованная работа групп драйверов – при начальной загрузке, останове, управлении электропитанием, работы стеков драйверов, работы нескольких одинаковых драйверов и др.

Взаимосвязь устройств и драйверов

В Zephyr доступ к аппаратуре в прикладном коде осуществляется через абстракцию “устройство”. С устройством связывается драйвер соответствующего типа. Прикладной код работает с аппаратурой через интерфейс “устройства”, запросы к которому транслируются в соответствующие вызовы к драйверу. В Zephyr есть ряд механизмов для сигнализации из драйвера прикладному коду (очереди сообщений, алерты, callback-и). Также может быть несколько устройств одного типа, которые имеет под собой разные экземпляры одного и того же драйвера.

Использование технологии Device Tree

Многие знакомы с этой технологией по Linux-like операционным системам. В последние года она начала проникать и в микроконтроллерные RTOS, в том числе была применена и в Zephyr, что позволило значительно абстрагировать соответствующие драйвера от вариативности настроек аппаратуры. Парсинг device tree производится на этапе компиляции, далее происходит связывание устройств с конкретными драйверами.

Типы устройств/драйверов

В Zephyr есть 4 типа устройств (и соответствующих драйверов), которые обязательны при портировании любой платы. Это контроллер прерываний, системный таймер, последовательный порт и, что довольно необычно, источник энтропии (что подчеркивает внимание Zephyr к информационной безопасности). Есть множество необязательных типов – будет дан полный список. Один тип может быть “производным” от другого, при этом он должен имплементировать все функции от “родителя”, а также имплементировать функции, расширяющие “родительский” интерфейс.

Интерфейс устройства

Интерфейс всех устройств стандартизирован. С его помощью есть доступ к имени, конфигурации, неизменяемым данным, состояния, информация о зависимости от других устройств, а также к API соответствующего типа – т.е. фактически к API драйвера, реализующего функциональность этого устройства.

Процесс инициализации

Как пример групповой работы с драйверами разберем процесс инициализация. В Zephyr он разбит на несколько логических стадий: PRE_KERNEL_1 – для драйверов, которые не используют ядро и другие драйвера, PRE_KERNEL_2 – для драйверов, которые также не используют ядро, однако используют драйвера, загруженные на предыдущей стадии и POST_KERNEL для всех остальных. Есть понятие “приоритета драйвера”, которые позволяет выстраивать четкий порядок инициализации. Есть также возможность не инициализировать драйвер на этапе загрузки, а сделать это позже.

Заключение

Zephyr имеет (по мерках микроконтроллерных RTOS) хорошо развитую и гибкую драйверную модель, позволяющую сильно сократить время разработки драйверов и в целом процесс портирования на новые платы, а также облегчить работу с аппаратурой для прикладного ПО. Неудивительно, что количество официально поддерживаемых Zephyr плат перевалило за тысячу и продолжает быстро расти.

Ключевые слова

RTOS, ОСРВ, Zephyr OS, FreeRTOS, драйверная модель, device driver model.