# Библиотека Modbus сканнер основанная на boost::coroutine2

Сканнер Modbus получает поток байт на входе и генерирует обратные вызовы содержащие протокольные пакеты (PDU) на выходе.

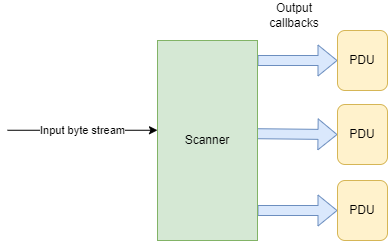
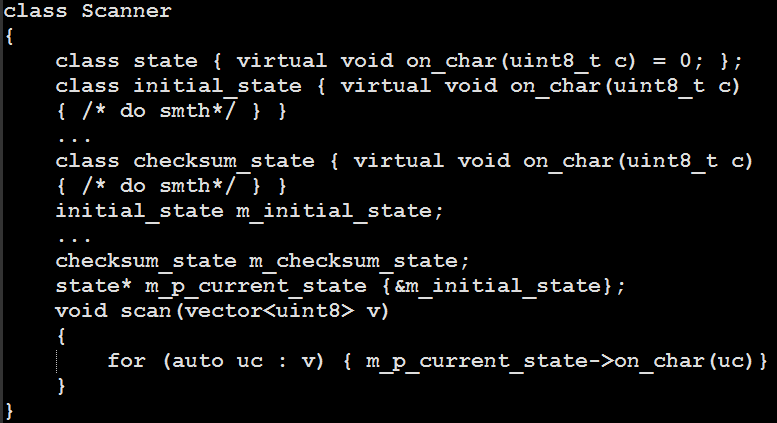


Схема работы сканнера.

Это выглядит довольно просто на первый взгляд, но надо учитывать что текущая порция байт на входе может содержать только часть PDU или несколько PDU. По этой причине сканнер должен запоминать свое состояние после каждого вызова.

Большинство сканнеров выполнены как автоматы конечных состояний (FSM Finite State Machine ). При этом сканнер постоянно находится в одном из возможных состояний и реагирует на входную порцию байт по разному, в зависимости от текущего состояния. Этот проект реализует сканнер протокола MODBUS при помощи boost::coroutine2 в качестве FSM и сравнивает coroutine-based сканнер со сканнером выполненным по традиционному паттерну (множество под-объектов состояний с виртуальной функцией on\_char ). Обычный подход к реализации сканнера требует наличия нескольких под-классов "состояние" каждый из которых реализует свой виртуальный метод on\_char() и указатель на экземпляр класса "состояние" который определяет состояние сканнера в данный момент. Это выглядит примерно так:



Такой подход много раз описан в литературе в частности в книгах Герберта Шилдта.

При всей привлекательности данного подхода он имеет и недостатки. Требуется написание большого количества бойлерплейт кода. Каждый подкласс должен быть определен. Сканнер должен иметь экземпляры каждого из подклассов, Неочевидная логика переключения состояний. Логика работы сканнера разбросана по множеству методов on\_char() и при чтении такого кода трудно охватить работу сканнера в целом. Все это может приводить к ошибкам в программе.

С другой стороны подход с использованием coroutine свободен от вышеперечисленных недостатков. Он значительно уменьшает размер программного кода. Логика переключения состояний очевидна. Фактически каждый оператор coroutine и есть новое состояние сканнера. С этой стороны преимущество coroutine-подхода очевидно. Не очевидно была эффективность работы coroutine-сканнера в сравнении с обычным подходом. Переключение стека и состояния регистра-указателя команд при входе в coroutine и возврате из coroutine требует временных затрат и было не ясно насколько это скажется на производительности работы сканнера.

Для того чтобы сравнить работу coroutine-сканнера с обычным сканнером в рамках этого проекта был сделан unit тест который в цикле выполнял сканирование разнообразных MODBUS-запросов с фиксацией времени затраченного на сканирование. Результаты теста: coroutine-сканнер 42 сек. обычный сканнер 48 сек. Полученный результат подтвердил конкурентноспособность coroutine-сканнера.

Тестирование корректной работы сканнеров двух видов было сделано при помощи open source Modbus Master-a https://github.com/stephane/libmodbus . Этот мастер шлет запросы на запись регистров и при последующем чтении проверяет соответствие прочитанных-записанных данных.

Для того чтобы была возможность использовать оба типа сканнеров в составе Modbus сервера, был написан простой Modbus-сервер который использовал библиотеку mb\_scanner создавая либо coroutine-сканнер, либо обычный сканнер и положительно отвечал на все запросы записи-чтения запоминая данные в запросах записи. Оба типа сканнеров безошибочно проходят тест.

Следует отметить, что данный проект не преследовал цели создать Yet Another Modbus Server. Таких серверов сделано превеликое множество, в том числе и open source. Скорее преследовалась цель "набить руку" в применении boost::coroutine2, получить паттерн который может быть использован не только для создания более сложных сканнеров других протоколов (Modbus протокол очень прост), но и для большого класса задач требующих применения шаблона FSM.