**Описание к блок-схеме приложения rip3p**

Rip3p предназначена для ухода с непереносимой платформы Borland. Название было сохранено, но дополненно. Цифра 3 указывает на работу в трех измерениях, а буква p указывает на возможнось написания плагинов, компилируемых отдельно и без необходимости менять основную программу.

Приложение написано с использованием среды разработки Qt5. Эта среда подразумевает только фиксированную точку входа, но с изменяемыми параметрами. В отличии от предыдущей программы, каждый файл является именем класса, описанного в нем, переведенный в нижний регистр.

1. main.cpp - точка входа в графическое приложение. В нем описаны пользовательские типы, учавствующие в обмене между потоками и указана версия OpenGL. Так же в нем запускается основной класс GWindow(2) и инициализируется потокобезопасное хранилище данных(4 и 5)

2. gwindow.cpp - основное окно программы. В нем реализован функционал MDI приложений и подгрузка и запуск плагинов. Так же в нем инициализируется класс управления проектом MControl(3) и передается ему управление mdi окнами.

3. mcontrol.cpp - класс управления. Это центральный узел проекта. Он инициирует и распределяет сигналы и слоты между классами, учавствующими в работе. Так же в нем заложен функционал управления настройками - сохранения в файл, в случае внесения изменений и загрузка при инициализации. Параметры после инициализации переносятся в потокобезопасное хранилище данных(4 и 5) и при сохранении берутся от туда. Так же класс при подаче команды от GWindow(2) создает mdi окна и присоеденяет его сигналы к основным массивам сигналов. При запуске он создает только DebugDialog (16) и Worker(6). Остальные, только по команде от MControl(3)

4. Memory.cpp - статический класс хранения данных. Функционирует как статический интерфейс к классу Rdata(5). Реализован в связи с большим объемом обращений к конфигурационным элементам. Сигнал-слотовая модель слишком объемен для подобного, а статический класс посностью избавляет от избыточности.

5. rdata.cpp - потокобезопасный класс хранения данных. Потокобезопасность реализована на мьютексах. Отдельные блокировщики для больших массивов данных (хранилище сырых данных, хранилище обработанных данных и массив для вывода на графики) и для большого количества переменных простых типов (float. int, string, bool).

6. worker.cpp - класс, производящий основные вычисления. Работает в отдельно потоке. Реализует основную математику программы. Создает в отдельных потоках сетевые воркеры (7 и 8).

7. tcpwork.cpp - класс - tcp клиент. Отправляет команды управления на РЛС и получает данные с него. Выделяет отдельные пачки сигналов и передает поблочно в Worker(6).

8. udpwork.cpp - класс - udp клиент. Отправляет команды управления на РЛС (по старому протоколу) и получает данные с него. Выделяет отдельные пачки сигналов и передает поблочно в Worker(6).

9. formSettings.cpp - класс - окно с настройками для РЛС.

10. gr3d.cpp - класс - окно для 3d графика.

11. scobject.cpp - класс для хранения данных для передачи в виджет OpenGL графика. Инициализирует дополнительный класс ScWorker(13) в отдельном потоке.

12. plotgl.cpp - класс - OpenGL виджет. Реализует вывод на экрана 3d графика.

13. scworker.cpp - класс, преобразующий данные из Rdata(5), а именно массива для вывода на графики, в примитивы и объекты, родные OpenGL виджету. Для улучшение отклика программы реализован в отдельном потоке.

14. plotpolarization.cpp - класс, реализующий вывод растрового изображения Амплитудно-фазовой характеристики.

15. ppworker.cpp - класс, преобразующий данные из Rdata(5), а именно массива для вывода на графики, в массив, который требуется для отображения битовой карты PlotPolarization(14).

16. debugdialog.cpp форма для вывода отладочной информации.

17. plugwin.cpp - класс для загрузки и включения/отключения отдельно написанных и скомпилированных плагинов.

18. pluginterface.h содержит интерфейс, реализация которого позволяет написать совместимый с программой плагин.

19 и 20. Пример реализации стороннего плагина.