**Описание к блок-схеме приложения RIP**

Приложение управления РЛС RIP написана с использованием среды разработки Borland C++ 6. В связи с этим было использована стандартная для этой среды архетиктура.

1. RIP\_wp.cpp - точка входа в приложение RIP. Это служебный файл среды разработки для инициализации всех форм и запуска основного окна приложения.

2. MainUnit.cpp (MainUnit.cpp и MainUnit.dfm) - основной класс приложения, а так же основное окно с фреймом для MDI окон. Класс несет основную нагрузку приложения. Большенство операций проходит именно в нем:

2.1 OnRecieve - модуль класса TformMain (основного класса файла MainUnit). Является приемником данных с РЛС. На вход модуля поступают уже выделенные блоки данных для их дальнейшей обработки. В данном модуле формируется номер команды и передается в следующий модуль.

2.2 ProcessAnswer - модуль режимов работы. По указанному выше номеру модуль начинает обработку данных. В зависимости от команды он или производит запись данных или вторичную обработку по заданному в нем алгоритму. В случае, если выбрана команда обработки, информация поступает в модуль MathAndPlot.

2.3 MathAndPlot - модуль кор. обработки и вывода данных. В мервую очередь модуль вызывает отдельный модуль Math, в котором производится вся обработка данных. Затем обработанные данные подготавливаются для вывода на экран и передаются в соответствующие классы.

2.4 После модуля MathAndPlot начинает свою работу модуль ControlAndMeasure. Он предназначен для поиска цели и измерения ее координат. В случае режима автосопровождения модуль так же подает команды на поворот РЛС.

3. Hardware.cpp - файл, в котором собраны несколько классов, активно обменивающиеся данными в разных потоках, для того, чтобы небыло потерь от РЛС и для выделения из потока данных отдельных пачек сигналов, для отправки в MainUnit.

4. SettingsUnit.cpp - класс, содержащий функционал управления настройками РЛС и загрузки/сохранения настроек.

5. CommandUnit.cpp - дополнительный класс для работы с сетью: формирование управляющих сигналов, вычисление из массива данных именно пакет и номер команды.

6. RasterSettingsUnit.cpp - класс, пренданзаченный для настройки окон вывода данных (растров). В нем задается яркость и порог отображения данных.

7. ScatterUnit.cpp - Дополнительный класс для изучения сигналов при изменении геометрии (угла поворота) цели.

8. RasterUnit.cpp - форма с графиком растра (горизонтальной или вертикальной поляризации)

9. Raster.cpp - класс хранилище обработанных данных для отображения битовой маски на графике RasterUnit.

10. DrawUnit.cpp - дополнительные графики, отображающие мгновенные значения (по пакетам) либо накапливающие данные в случае планшета.