Protocol : BaseInterface

```
start(settings &);
GetFloats(pars)
{
        addToInQueue(CommandStruct);
}
getDiscreets();
getAlarms();
```

ProtocolThread

```
Run()
{
        CommandStruct cst;
        deQueue(cst);
        this->startCommand(cst);
}
OnReceive()
{
        SignalStruct sig;
        this->Parse(sig);
        addSignalToOutList(sig);
}
```

Coma

Bda_timer

Module *m_Module

```
new ...checkDialog(BaseInterface iface);
new ...confDialog(BaseInterface iface);
new InfoDialog(BaseInterface iface);
new ...startupDialog(BaseInterface iface);

foreach (*Dialog)
{
        connect (Bda_timer, this->update());
}
```

Udialog ...checkDialog(iface)

```
BaseInterface *m_iface;
update()
{
        m_iface→getFloats();
        m_iface→getDiscreets();
        m_iface→getAlarms();
        this->updateFloats();
        this->updateDiscreets();
        this->updateAlarms();
}

updateFloats()
{
        SignalStruct sig;
        getSignals(sig);
        this->updateFloatsVisible();
}
```

DataTypes.h

```
SignalStruct;
SignalTypeEnum;
CommandStruct;
CommandEnum;
```

DataManager

```
addToInQueue(CommandStruct);
deQueue(CommandStruct &);
addSignalToOutList(SignalStruct &);

getSignals(...);
getFile(…);
getConfig(...);
getResponse(...);
```

Примерное описание работы АВМ-Сервиса (накорябано по памяти)

1. В главном классе Coma при подключении к модулю по определённому интерфейсу (USB/Ethernet/RS485) активизируется один из классов протоколов — Protocom/Iec104/Modbus — вызывается метод start() и экземпляр класса протокола передаётся в конструктор класса Module.
2. В статическом классе DataManager определяются методы addToInQueue (добавление данных в очередь на отправку в интерфейс), addToOutList (добавление результатов, полученных из интерфейса от модуля в список), deQueue (извлечение данных из входной очереди inQueue для отсылки в интерфейс) и для каждого типа данных методы get(Signals|File|Config|Response) (извлечение результатов из OutList, перевод в стандартные типы и возврат их в требуемом виде).
3. В классе Module для данного типа модуля создаются конкретные диалоговые окна из базового класса Udialog и конкретных диалоговых классов, определённых для каждого модуля, в конструкторе Udialog принимается переменная типа BaseInterface. В каждом диалоговом классе, производном от Udialog определён метод update, вызываемый по срабатыванию таймера Bda_Timer, определённому в Coma. В методе update возможно изменение частоты обновления путём деления счётчика на необходимое количество.
4. При срабатывании Bda_Timer и вызове update соответствующего диалога происходит запрос требуемых данных с помощью методов BaseInterface, и проверяется в выходном списке наличие необработанных данных, принятых на предыдущем запросе.
5. В случае наличия данные извлекаются соответствующим методом get… из DataManager и возвращаются в диалог для отображения.
6. В фоновом режиме работает метод Run соответствующего потока, создаваемого в соответствующем BaseInterface. При наличии данных во входной очереди команд команда вместе с сопутствующими параметрами извлекается из очереди методом deQueue класса DataManager, сериализуется и отправляется в модуль.
7. При получении из модуля данных они десериализуются и отправляются в выходной список с помощью addSignalToOutList класса DataManager.