Новое программное обеспечение прибора MASS/DIMM. Управляющее ядро Turbina-core

О. Возякова, В. Корнилов, Н. Шатский

21 сентября 2006

Введение

Используемая для управления работой приборов MASS и MASS/DIMM, предназначенных для измерения вертикального профиля оптической турбулентности, программа Turbina, в своей основе, разрабатывалась в 2001 г. Первоначальные требования к этой программе были сформулированы на основе представлений о процессе работы с прибором как о постоянном интерактивном взаимодействии между наблюдателем и прибором. В этот период наша группа не обладала достаточным опытом в разработке программ управления приборами, сбора и обработки данных в реальном времени под операционной системой Linux. Дополнительным обстоятельством послужило отсутствие ясного понимания перспектив в развитии и применении самого прибора MASS.

Первая работающая версия программы вышла в 2002 г. В течение 2002 – 2005 г. управляющая программа постоянно поддерживалась и модифицировалась в соответствии с накапливающимся опытом работы и изменением условий применений прибора MASS и, впоследствие, MASS/DIMM. Существенным изменениям Turbina подверглась в 2004 г., когда были начаты работы по включению ее как отдельной компоненты в автоматическую систему, включающую также программное обеспечение DIMM, телескопной монтировки, программу–супервизор и другие программные модули. Для этого было реализовано параллельное с графическим интерфейсом пользователя управление работой программой Turbina через сокетные соединения. В настоящий момент последняя рабочая версия программы — Turbina 2.54.

Основным препятствием для работы программы в составе сложной автоматической системы и/или при удаленном управлении оказалось наличие глубоко имплантированной интерактивности. Первоначальные требования, чтобы Turbina максимально верифицировала действия наблюдателя и сообщала ему о всех возникших проблемах, привели к потенциальной неустойчивости и непредсказуемости ее работы. Хотя мешающие автоматическому режиму функции были максимально подавлены, заложенные при программировании особенности являются серъезным ограничением для дальнейшего развития программного обеспечения.

В этот же период разрабатывалось программное обеспечение и для других проектов, выполняемых нашей группой. В процессе этой работы постепенно появилась программная инфраструктура на основе общих для всех управляющих программ модулей и провереных на практике принципов построения управляющих программ. Следующие обстоятельства послужили поводом для начала разработки нового программного обеспечения прибора MASS/DIMM

- Обязательства перед группой TMT по исследованию этого вопроса.
- Начало программы по выбору места для телескопа ELT ESO, в которой будет участвовать как минимум 4 наших прибора.
- Начало нашей собственной программы по исследованию оптической турбулентности в плане прогноза перспектив развития адаптивной оптики на обсерваториях России, поддержанной грантом РФФИ 06-02-16902а.
- Необходимость всесторонних астроклиматических исследований в месте установки 2.5 м телескопа ГАИШ МГУ в районе Кисловодска.

Цели и методы построения нового программного обеспечения

Основные цели разработки нового поколения программного обеспечения ПО для прибора MASS/DIMM кратко можно сформулировать так:

- Новое ПО предназначено в первую очередь для автоматической работы без непосредственного участия наблюдателя или для проведения удаленных измерений.
- Необходимо повысить устойчивость работы и надежность получаемых данных, упростить установку и обслуживание ПО.
- В новом ПО необходимо значительно увеличить ресурс для адаптируемости и модифицируемости.

В частности, хорошая адаптируемость и модифицируемость необходима для реализации в дальнейшем новых алгоритмов обработки, позволяющих корректно обрабатывать ситуацию насыщения атмосферных мерцаний. Мы также предусматриваем параллельную работу программного обеспечения DIMM канала и использования его данных для восстановления профиля турбулентности в приземном слое. Параллельно работе над ядром Turbina-core(M) новой версии для контроля и сбора данных с канала MASS, разрабатывается и ядро Turbina-core(D) для получения данных измерений дифференциальных дрожаний в канале DIMM.

Достижение целей разработки нового поколения реализуется на основе и при помощи:

- Разделение программы на три независимых программых компоненты в соответствии с тремя основными функциями: управление прибором и сбор данных, обработка данных и обеспечение пользовательского интерфейса. Минимизация числа поддеживаемых функций.
- Использование накопленного за период 2002 – 2005 г. опыта работы с нынешним ПО, опыта его обслуживания и включения в состав автоматических астроклиматических систем.
- Использование накопленного опыта и готовых программных решений, полученных при работе над другими проектами, активное применение шаблонов стандартной библиотеки C++.

Разделение единой программы на независимые компоненты позволяет, при необходимости, освободить ресурсы управляющей машины, перенеся обработку и графику на другую машину или проводя обработку в другое время. Это позволит при обработке использовать более сложные и точные алгоритмы восстановления высотного профиля турбулентности. Также напомним, что нужен был ресурс для работы ПО канала DIMM.

Общая структура ПО показана на Рис. 1. Пунктиром обозначена компонента, относящаяся к DIMM каналу прибора. Реализация этой части ПО также запланировано к осени 2006 г. Проектирование и программирование обеих ядер выполняется так, чтобы использовались не только общие подходы к структуре программ, но и общие программные модули. Это позволяет повысить и эффективность разработки и надежность получаемого программного продукта.

Ядро ПО — Turbina-core(M)

Новая версия управляющей программы для прибора MASS состоит из трех программных компонент и вспомогательных утилит. Для проведения реальных наблюдений требуется,

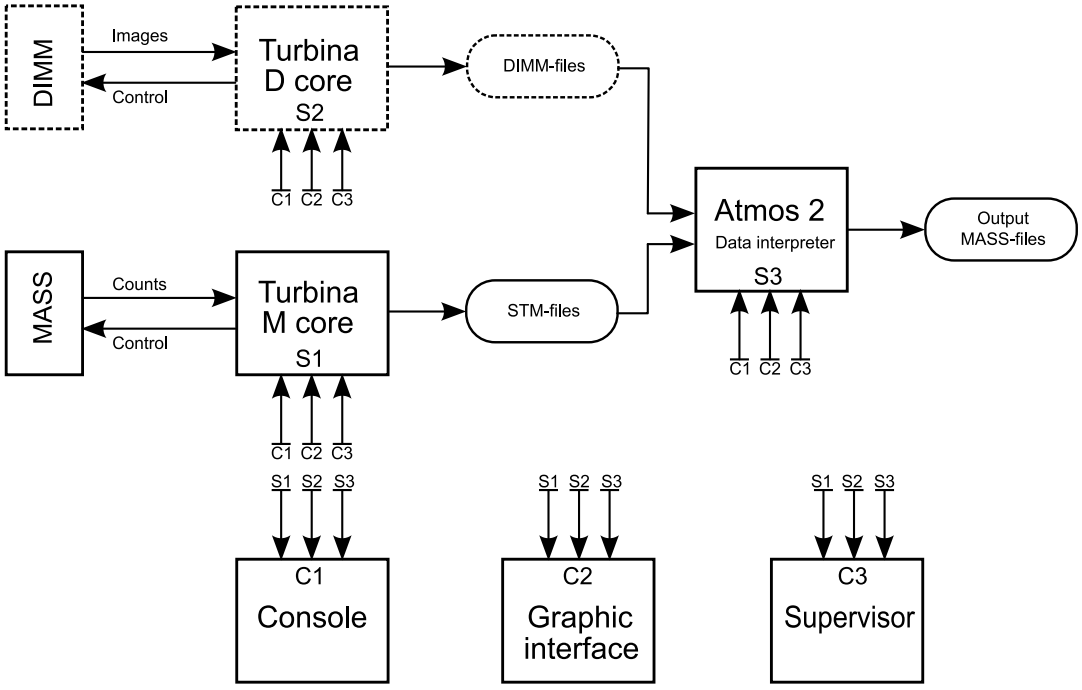

Рис. 1: Структурная схема нового ПО прибора MASS/DIMM. Каждый функциональный компонент — отдельная программа. Взаимодействие осуществляется через сокетные соединения и файлы данных. Пунктиром обозначено ПО DIMM канала.

как минимум, ядро управляющей программы — Turbina-core(M). Эта программная компонента непосредственно взаимодействует с прибором, управляет его работой и получает данные измерений. Данные измерений записываются в выходной файл в расширенном формате файла *.stm. Описание формата данных, записываемых в этот файл, будет приведено далее в разделе .

В отличие от других компонент, ядро должно быть запущено на машине к которой подключен прибор. Работа ядра не зависит от того, работают ли в настоящий момент другие компоненты или нет. Однако, для проведения реальных наблюдений необходимо управление его работой со стороны супервизора (при работе в составе автоматической системы) или модуля графического интерфейса (при ручном управлении наблюдениями). В принципе, любая другая программа или скрипт, обеспечивающие нужный командный протокол, также может инициализировать работу turbina-core и проведение измерений. Для отладочных целей предусмотрена программа-консоль, позволяющая управлять ядром в текстовом режиме, также работающая с ядром через сокетное соединение.

Для повышения устойчивости ядра было минимизировано число функций им выполняемых и приняты следующие ограничения при конкретной реализации:

1. Ядро не поддерживает оригинального инструмента. Если выяснится такая необходимость, требуемые модули будут добавлены позже.
2. В ядро не включается дополнительный инструментарий. Однако, тестовые функции, используемые для контроля функционирования прибора и программы включены. Возможно, что функции, предназначенные для отладочных работ (счетные, статистика

и т.п.), будут добавлены после испытаний ПО в реальных измерениях.

3. По сравнению с существующей версией, число параметров конфигурационного файла сокращено до минимума. Ядро (как и другие программные компоненты ПО) не перезаписывает конфигурационный файл, это может сделать пользователь только вручную.
4. Конфигурационный файл (по умолчанию turbina.cfg) — единственный файл, управляющий работой ядра. В нем содержатся и необходимые данные из файла device.cfg.
5. Значительно усилена роль идентификации модулей (и идентификации прибора в целом), при несоответствии конфигурационного файла и идентификации прибора следует отказ от работы.
6. Хотя параметры, идентифицирующие обстоятельства наблюдений, не нужны для работы ядра, они также включены в конфигурационном файле для перезаписи их в выходной файл без изменений (штамп наблюдений).

Опции запуска ядра Turbina-core(M)

В отличие от первой версии, при запуске Turbina-core(M) в командной строке могут быть указаны опции запуска. Эти опции и пояснения к ним приведены ниже.

`-t my.cfg` — используемый конфигурационный файл. Это позволяет избавиться от копирования конфигурационных файлов.

`-v` — версия, позволит уменьшить путаницу с версиями и при запуске из скрипта отслеживать модификации ПО.

`-d` — отладка, все, в том числе и ранее фатальные, ошибки выдаются на stderr и не приводят к останову выполнения команд даже при отсутствии прибора.

`-r R` — устанавливается скорость R обмена с прибором, отличная от 460Kb/s

`-p N` — номер N серверного порта, перекрывает значение по умолчанию.

Владельцем программы Turbina-core(M) является пользователь `mass:mass` и она запускается от лица этого пользователя. Такое решение должно обезопасить от случайной или злонамеренной интервенции через программу в системные ресурсы.

Размещение ядра Turbina-core(M)

Для работы ядра Turbina-core(M) предполагается использовать ту же структуру директорий, что и для используемой программы Turbina последних версий (начиная с 2.48). Исполняемый код turbina-core (или turbina-M-core) ядра раполагается в /opt/turbina/bin , конфигурационный файл по умолчанию turbina-m.cfg (для отличия от прежнего конфигурационного файла) находится в /opt/turbina/etc.

Остальные файлы — файлы входных данных, для работы ядра не нужны. Поэтому при установке ядра соответствующие директории не создаются. Однако, они уже могут быть созданы ранее для работы предыдущей версии Turbina.

В этой связи замечание: поскольку ядру совершенно не нужно знать, какой объект оно измеряет, входной каталог также не нужен. И команда установки объекта измерения предназначена только для записи имени объекта в выходной файл и должна содержать полную информацию об объекте.

Выходные файлы создаются как и ранее в директории, указанной в конфигурационном файле, по умолчанию: /opt/turbina/data/out/ и /opt/turbina/data/log. Умолчание прописывается в модуле names.h и используется если в файле .turbina.cfg соответствующая информация отсутствует.

Конфигурационный файл и его структура

Для работы ядра нужен только один конфигурационный файл. Часть информации о параметрах прибора и информация об обстоятельствах наблюдений самому ядру не нужна, но нужна для записи в выходной файл для дальнейшей обработки данных. Это может делать только ядро, поскольку предполагается, что остальные компоненты в это время могут быть незапущены. Эта информация хранится в отдельной секции конфигурационного файла как штамп — т.е. ее содержание не анализируется, в конфигурационную структуру не добавляется, а переписывается в выходной файл, после некой преамбулы, генерируемой ядром. Формат записи конфигурационного файла при этом преобразуется в формат выходного файла.

Эта секция называется `General` и содержит подсекцию `Site`. В подсекции содержатся название места и его координаты. Переменная `TimeZone` исключается, поскольку ядро использует только системное время UT. Добавлен обязательный параметр `DeviceNumber`. Дальнейшее изложение основано на сравнении конфигурационного файла оргинальной программы и нового ядра.

Подсекции `Inputs`, `Outputs` и `Socket` исключаются. Пути считаются стандартными и заложены в модуле `names.h`. Это связано в первую очередь с тем, что самому ядру не нужна информация, хранимая ранее в секции `Inputs`. Изменение же расположения выходных файлов должно быть синхронизовано с другими программными компонентами ПО (Atmos2 и Turbina-GUI), что достигается поддержанием стандартной конфигурации директорий для всего ПО. Аналогично решается вопрос и с номером серверного порта.

Секция `Operations` как и прежде содержит настроечные параметры различных типов измерений. Однако она значительно редуцирована и некоторые параметры (например, из подсекции `Common`, которая исключена) перемещены в секцию, содержащую информацию о приборе. Это стало возможным после объединения двух конфигурационных файлов в один.

Параметр `FluxEstimationTime` расположен в подсекции `Tests`. Как показала практика, параметры `SciCountsSave` и `TechCountsSave` – не лучший способ регулирования расширенного вывода. Но реально он бывает нужен на несколько минут, поэтому предусмотрен другой механизм включения/выключения расширенного вывода (А какой?).

```
Section "General"
    SubSection "Site"
        DeviceNumber           MD22     ;device serial number
        SiteName             Moscow     ;observatory or site na
        Longitude          2 07 00      ;longitude of the site:
        Latitude          55 42 14      ;latitude of th
    EndSubSection
EndSection
Section "Operations"
    SubSection "Normal mode"
```

```
        Exposition              1.0     ;micro-exposition time in ms (>=1)
        BaseTime                1.0     ;base time in sec.
        AccumTime                60     ;accumulation time in sec.
    EndSubSection
    SubSection "Background measurement"
        AccumTime                10     ;full time of the measurement
    EndSubSection
    SubSection "Tests"
        FluxEstimationTime        4     ;sec
        ExchangeTestTime         40     ;exchange test duration in
        DetectorTestTime          8     ;
        DetectorLight          0.23     ;reference light level for detector test
        TestCounts    70 120 300 700     ;reference counts into channels
        FluxTolerance          0.2     ;allowed tolerance for real counts
        NonPoisTolerance       0.2     ;allowed tolerance for non
        StatisticLight         0.5     ;light levels for statistical test
        Modulation             0.1     ;modulation amplitude for statisti
    EndSubSection
EndSection
Section "Device"
    SubSection "Bicounter 1"
        Module                  0x01    ;module unique address
        PMTNumberA              AJ4384  ;PMT serial number
        PMTNumberB              AJ4573  ;PMT serial number
        Identification      0x01 0x06 0x6B 0x07
        DiscriminationA         0.4     ;
        DiscriminationB         0.4     ;
        NonLinearityA           11.0    ;
        NonLinearityB           11.0    ;
        NonPoissonA             1.020   ;
        NonPoissonB             1.018   ;
    EndSubSection
    SubSection "Bicounter 2"
        Module                  0x02
        PMTNumberA              AH7806
        PMTNumberB              AJ4567  ;PMT serial number
        Identification      0x01 0x06 0x6B 0x07
        DiscriminationA         0.4     ;
        DiscriminationB         0.4     ;
        NonLinearityA           11.0    ;
        NonLinearityB           11.0    ;
        NonPoissonA             1.040   ;
        NonPoissonB             1.025   ;
    EndSubSection
    SubSection "Channels"
        ChannelA            Counter 2A  ;counter for Channel A
```

```
        ChannelB            Counter 2B  ;counter for Channel B
        ChannelC            Counter 1A  ;counter for Channel C
        ChannelD            Counter 1B  ;counter for Channel D
    EndSubSection
    SubSection "Auxiliary"
        Module                0x10
        Identification      0x01 0x08 0x2a 0x07
        HighVoltage             775     ;default high voltage level, V
        PollingTime               1     ;polling period in s
        IlluminationLevel       0.5     ;default illumination leve
    EndSubSection
EndSection
Section "Optics"
    SubSection "Common"
        Magnification           15.70   ;magnification factor of the device
    EndSubSection
    SubSection "Segment A"
        Inner                   0.00    ;
        Outer                   1.27    ;outer diameter of A mirror, mm
    EndSubSection
    SubSection "Segment B"
        Inner                   1.30    ;inner diameter of B mirror, mm
        Outer                   2.15    ;outer diameter of B mirror, mm
    EndSubSection
    SubSection "Segment C"
        Inner                   2.20    ;inner diameter of C mirror, mm
        Outer                   3.85    ;outer diameter of C mirror, mm
    EndSubSection
    SubSection "Segment D"
        Inner                   3.90    ;inner diameter of D mirror, mm
        Outer                   5.50    ;outer diameter of D mirror, mm
    EndSubSection
EndSection
```

Как видно, значительная часть содержит информацию не о том, как проводить измерения, а о характеристиках конкретного прибора. Часть перечисленных параметров необходима ядру для создания соответствующих объектов. Остальные — для записи в выходной файл для проведения обработки. Однако в выходной файл записывается информация не по счетчикам, а по каналам. Т.е. на основе карты соответствия (подсекция `Channels`) параметры счетчиков преобразуются в параметры каналов. Например, непуассоновость `NonPoissonA` счетчиков `Bicounter 1` транслируется в `NonPoissonC` (См. следующий раздел).

После чтения конфигурационного файла на этапе инициализации вся информация хранится в символьном виде в трехмерном ассоциативном массиве (дереве). К значениям параметров программа имеет прямой доступ по названию секции, назанию подсекции и имени параметра. Преобразование во внутреннее представление осуществляется на этапе инициализации (конструирования) того или иного объекта. Запись в выходной файл осуществляет-

ся непосредственно из конфигурационного дерева. Чтобы избежать рассогласования между текущим значением параметра и его значением в дереве, функции, способные менять параметры (см. команду SET) должны также корректировать содержимое дерева.

Выходные файлы и их краткое описание

Файл результатов измерений: *.stm

Выходной файл данных измерений дополнительно содержит всю необходимую информацию для последующей обработки. Файл открывается в момент инициализации и его имя генерируется в соответствие с действующей договоренностью: до локального полудня — дата предыдущего дня, после — дата полудня. Если этот файл до инициализации не существовал — пишется преамбула, в которой содержится имя выходного файла (для контроля в случае ошибок копирования и редактирования), версия программы turbina-core, время ее компиляции, информация из конфигурационного файла о месте проведения измерений и о номере прибора.

Далее записывается вся необходимая для обработки информация из конфигурационного файла. Как уже было сказано в предыдущем разделе, запись производится в терминах оптических каналов: A, B, C и D.

Собственно данные записываются в форме статистических моментов потоков в этих четырех оптических каналах, определенных по отсчетам ФЭУ, накопленным в течение времени basetime. Дополнительно заметим, что в этом виде записываются данные всех измерений: и измерения фона, и оценки потока произвольного объекта, и тест детекторов. Формат записи идентичен записи в `*.stm` файл существующей программы Turbina. Рассматривается вариант записи дополнительных 4 колонок, содержащих нормированные третьи моменты измерений.

Для разделения данных разных мод (измерений и тестов) записываются M-строки, также как это делается в файле `*.mass` оригинальной программы. Также в виде O-строк записываются данные об объекте наблюдения и в виде P-строк — измененые параметры работы.

Конечно, расширение формата выходного файла приведет к потере полной совместимости ПО разных поколений, однако, преобразование из расширенного формата `*.stm` в формат оригинальных `*.stm` и `*.mass` файлов может быть осуществлено без потерь при помощи достаточно простого скриптового конвертера.

Файл протокола: *.log

В других реализованных проектах нами используется два протокольных файла: файл команд, поступивших серверу в порядке поступления и его реакция на эти команды: `*.cmd` и протокол обмена: `*.log`. Как показывает опыт, иногда трудно определить точное временное соответствие этих двух файлов. Поэтому единственный файл протокола ядра turbina-core содержит в порядке поступления:

1. протокол обмена между сервером и клиентами,
2. протокол обмена с прибором: полный в режиме отладки (см. ниже), иначе, для компактности, — только ошибки обмена,
3. сообщения о внутренних ошибках программы.

Поскольку источников записей в файл протокола несколько (как минимум три), то необходима централизованная система протоколирования команд и ошибок, препятствующая повторной записи одинаковых сообщений. Файл протокола пишется всегда, то есть флаг разрешения/запрета записи протокола аннулирован.

Генерирование файла протокола тесно связано с системой обработки ошибок, возникающих в процессе работы программы. Поскольку ошибки планируется в обязательном порядке дублировать на `stderr` и по запросу передавать внешним компонентам — предполагается во всех этих случаях использовать единый формат сообщения об ошибках.

Строка сообщения об ошибке должна содержать:

1. момент времени появления ошибки с точностью минимум 2 знака после запятой,
2. обобщенный код ошибки для организации автоматической обработки ошибок (простейший случай — для сбора статистик отказов),
3. указание на место появления ошибки,
4. текстовое объяснение характера ошибки (стандартно принятое),

Например: `15:47:13.443 (786) Open turbina.cfg Permission denied` . Эту строку, содержащую последнее запротоколированное сообщение, и предлагается вернуть в ответ на запрос GET ERRMSG от внешней программы. Для этого эта строка, или даже несколько, должны храниться в специальном деке.

Обшая структура и функционирование ядра

Самый верхний логический уровень ядра несёт функции сервера сокетных соединений, инициализируемого в модуле main. Этот сервер работает в основном потоке. После появления запроса на соединение и проверки полномочий потенциального клиента (это предусмотрено, хотя механизм реализации еще не ясен) создается объект-клиент, являющийся изображением внешнего соединения. Дальнейшее взаимодействие сервер ведет с этим объектом. Закрытие соединения возможно только по инициативе внешнего клиента, при этом соответствующий объект-клиент уничтожается.

Команды и запросы, поступившие ядру из различных источников (например, с консоли при помощи утилиты client.tcl или утилиты telnet), передаются сервером специальному разборщику, им обрабатываются в порядке поступления и запускаются на обработку соответствующими специальными функциями. Процедуры выполнения команд SET, GET и STOP быстро-выполняющиеся (мгновенные — время выполнения много меньше тайм-аута, установленного для соединения), поэтому запускаются в основном потоке. Кроме простоты работы, это позволяет также избежать конфликтов установок параметров разными клиентами.

Команды же INIT, PARK, RUN и QUIT связаны с длительными процедурами работы и изменением статуса программы и поэтому их процедуры выполнения запускаются в порождаемых специально для этого новых отделенных потоках. Перед завершением выполнения, эти процедуры в обязательном порядке посылают ответ клиенту с результатом выполнения команды и закрывают свой поток.

Отдельным функциям программы/прибора соответствуют отдельные объекты. Если какая-то одна функция поддерживается разными аппаратными модулями, то она имеет

одинаковый интерфейс в соответствующих им классах (используется механизм наследования/перегрузки и виртуальных функций — полиморфизм). Под функциями в данном контексте понимается как аппаратные, так и функциональные программные модули.

Верхнему уровню программы Turbina-core соответствует объект класса MASS. Этот объект создается динамически при получении команды инициализации. Класс MASS содержит объекты-модули (например, Auxiliary) и объекты-моды (например, Normal). Эти объекты конструируются автоматически при конструировании класса MASS. Парковка приводит к удалению объекта MASS, при этом автоматически вызывается удаление отдельных модулей и мод. При возникновении ошибочной ситуации, препятствующей правильному созданию объекта MASS, также гарантируется автоматическое удаление объектов-модулей и объектов-мод. В результате программа вернется в состояние PARKED.

Приоритет при конструировании объектов принадлежит программе, т.е. при инициализации объекта MASS будут созданы все предусмотренные текущей версией объекты-модули, независимо от того, есть ли они реально в данном приборе. Реальная инициализация (активация) этих объектов управляется конфигурационным файлом, а именно — наличием соответствующей секции. Если такой секции нет, будет создан пустой объект (не активизованный) и вызов любого его публичного метода может привести к ошибке, поэтому каждый метод сам производит проверку своего состояния.

Если же модуль объявлен в конфигурационном файле, отсутствие любого нужного программе параметра или неуспех аппаратной инициализации модуля приведет к генерации исключения, выходу из конструктора объекта MASS и автоматическому разрушению уже созданных объектов. В результате программа вернется в состояние PARKED.

Очевидно, что после запуска, ядро может находится в одном из трех состояний:

1. программа запущена, но еще или уже нет запросов на соединение,
2. соединение установлено, но средства ядра не инициализированы,
3. соединение установлено, произошла полная инициализация функций ядра.

Четвертое состояние: функции ядра инициализированы, но соединение отсутствует (потеряно) не яляется нормальным. При возникновении такой ситуации текущая команда все же завершается, но сразу же после этого выполняется функция PARK и программа возвращается в исходное (первое) состояние.

Ниже приведен список объектов, конструируемых при создании объекта MASS, в порядке их создания.

- `logfile`
- `config`
- `rs485`
- `bicounter[1,2]`
- `auxiliary`
- `channel[A,B,C,D]`
- `normal mode`
- `exchange test`
- `detector test`
- `statistics test`
- `outfile`?

Режимы работы программы

Предусмотрены рабочий и отладочный режимы работы. Последний является более "мягким" по условиям выполнения вариантом первого. В частности,

- Возможно создание объекта-модуля без инициализации прибора.
- Возможна инициализация модуля с несовпадающей идентификацией
- Выполнение команды по возможности доводится до конца независимо от промежуточных ошибок
- Паузы, предусмотренные в программе для достижения аппартурой рабочих характеристик после включения (контрольный свет и высокое напряжение), игнорируются.

Отладочный режим включается опцией запуска программы и отражается специальным комментарием в выходном файле.

Инициализация INIT и парковка PARK

По сравнению с оригинальной Turbina функция INIT значительно расширена. Это сделано для того, чтобы не было необходимости при завершении наблюдений утром выходить из программы и заново запускать ядро вечером, накануне начала наблюдений. В настоящей версии во время инициализации выполняются следующие действия:

1. Текущие сокетные соединения сохраняются.
2. Открывается файл протокола, при необходимости — с новым именем.
3. Происходит открытие конфигурационного файла, его чтение и закрытие.
4. Выполняется открытие устройства и инициализация прибора и отвечающих за его модули объектов.
5. Выполняется инициализация объектов, отвечающих за режимы (моды) работы
6. Происходит открытие выходного файла и запись преамбулы и штампа, если это новый, а не уже существующий, выходной файл.

Повторная инициализация из инициализированного состояния (состояния готовности) невозможна, поскольку перед этим необходимо удалить уже существующие объекты. Удаление же основных объектов программы осуществляется при выполнении команды PARK или завершении работы. Однако можно, в случае необходимости, автоматически вызывать выполнение команды парковки. Открытые устройства и файлы при этом закрываются и переоткрываются. В случае неудачи, т.е. если не удалось прочитать конфигурационный файл, открыть выходные файлы или инициализировать *ни один* модуль прибора, статус системы остаётся PARKED . Как указано выше, в отладочном режиме инициализация модуля возможна и при несовпадении идентификации с данными конфигурационного файла. В случае неудачи процедуры инициализации гарантируется отсутствие состояний, промежуточных между READY и PARKED.

Аналогично, в обратном порядке выполняется парковка:

1. Выключается высокое напряжение и контрольный свет, если они включены.
2. Выполняется закрытие устройства.
3. Выполняется закрытие выходных файлов. Файл протокола закрывается последним.

4. Уничтожаются объекты, созданные во время инициализации.

Запущенная парковка всегда выполняется до конца. Если выполнить необходимые действия с аппаратурой не удалось (погасить свет, выключить высокое), то эти модули сбрасываются посылкой команд RESET. Проблемы при парковке сообщаются клиенту, как положено (см. п.).

При запуске инициализации или парковки (если статус системы это позволяет) возвращается OK WAIT=*waittime* STATUS=BUSY, где waittime – время работы в секундах. По окончании операции возвращается OK STATUS=READY или OK STATUS=PARKED или ERROR STATUS=ERFAT в случае ошибки.

При таком подходе, запуск самой программы не может привести к полной диагностике работоспособности системы. Такую проверку можно организовать посылкой последовательности команд INIT, возможно – набора тестов, и PARK сразу после старта программы. Возможный вариант — включение автоматической стартовой инициализации в программый код с последующей автоматической деинициализацией.

Напомним, что в оригинальной программе автоматическая стартовая инициализация была отключена из-за структурных проблем, возникших после реализации внешнего управления через сокетные соединения. Дополнительным фактором послужило отсутствие внешней команды включения высокого напряжения. Издержки такого решения были очевидны: это запуск измерений без инициализации при интерактивной работе и появление неожиданных проблем при работе в составе автоматической системы.

Команды RUN

Команда RUN запускает процесс измерений потоков прибором в зависимости от параметра команды. Реализовано несколько вариантов этой команды:

1. RUN или RUN NORMAL — запуск измерений мерцаний установленного объекта. Хотя ядру и не важно, какой объект установлен, проверка установки производится, чтобы гарантировать наличие параметров объекта в выходном файле. Окончательный ответ включает грубую оценку нормальных индексов, вычисляемых как $S_i = (disp_i - mean_i)/mean_i$: OK STATUS=READY NORMAL=S_A, S_B, S_C, S_D (индексы выдаются только если команда была дана как RUN NORMAL; в остальном отличий нет).
2. RUN BACKGROUND — запуск измерений, помеченных на выходе как "фон". Проверка значений не производится, но в окончательном ответе средние значения потоков приводятся в форме OK BACKGROUND=B_A, B_B, B_C, B_D STATUS=READY
3. RUN FLUX — запуск измерения потока для целей контроля и вспомогательных операций, например, при центровке звезды. Окончательный ответ содержит средние значения потоков в каналах: OK FLUX=F_A, F_B, F_C, F_D STATUS=READY
4. RUN SCENARIO="20*N+B" — запуск сценария (последовательности мод). В отличие от оригинальной программы сценарии не определяются заранее, а каждый раз передаются как строковое значение параметра SCENARIO. Ответ по окончании работы сценария — OK STATUS=READY.

При начале измерений проверяется, сколько времени прошло с момента включения высокого и, если проводится статистический тест, – с момента включения светодиода. При

необходимости (в рабочем режиме) выдерживается нужная пауза. Суммарное время waittime, потребное на выдержку пауза и выполнение измерения или сценария, возвращается перед выполнением команды как OK WAIT=*waittime* STATUS=BUSY.

К вопросу о процедуре проверки работоспособности прибора (не программы!): последовательность тестов, которые должны быть выполнены после команды INIT, также задаётся извне в виде сценария. Например: INIT, а затем RUN SCENARIO="D+S+2*T".

Команды SET и GET

Команда SET неявно выполняется всегда, когда в строке команды RUN содержится парметр со значением, связанные знаком присвоения. Типичный случай, используемый в программе: RUN SCENARIO="D+S+2*T". Однако предусмотрена отдельная команда установки сценария. В этом случае сценарий является статическим и может быть неоднократно запущен на выполнение командой RUN SCENARIO.

Команда SET OBJECT="...." предназначена для записи в выходной файл O-строки для обозначения измеряемого объекта, хотя сомому ядру эта информация ненужна. Аргумент (значение) этой строки рекомендуется записвать в форме, используемой оригинальной Turbina.

В список команд ядра добавлены команды манипулирования высоким напряжением,

SET HV=ON — включение высокого напряжения,

SET HV=OFF — выключение высокого напряжения,

необходимые, как показал опыт, в случаях потенциальной пересветки. При работе с этими командами нужно исходить из следующих соглашений:

1. при инициализации высокое напряжение не включается,
2. оно включается отдельной командой SET HV=ON или автоматически при запуске измерения, если перед этим не было команды SET HV=OFF,
3. выключается командой SET HV=OFF или по команде PARK,
4. команды установки значения высокого напряжения нет, используется значение из конфигурационного файла.

Измерения показывают, что коэффициент непуассоновости весьма капризный параметр, который в первые минуты сильно отличается от номинального, хотя поток уже достаточно стабилен. Поэтому команда включения должна обеспечивать фиксированную задержку порядка и более 5 мин, достаточную для стабилизации высокого и, главное, стабилизации режима ФЭУ.

Структурно, такая команда включения, требующая значительной задержки, выпадает из группы SET и более логично было бы включить ее в группу RUN. Реализован другой подход. Высокое напряжение включается сразу и внешняя управляющая программа получает ответ о готовности мгновенно. Проблема задержки перекладывается на процедуру запуска измерений, которая должна проверять прошедшее с момента включения время и, если надо, задерживать запуск начала измерений (см. п.). Это больше согласуется и с автоматическим включением высокого, однако это нужно иметь ввиду при планировании измерений. Команда выключения выполняется мгновенно.

Механизм защиты от пересвета постоянно включен. Подразумевается, что основная причина пересвета — случайный посторонний свет ночью или попадание Луны в поле зрения прибора. При обнаружении пересвета высокое, как и прежде, выключается (аппаратно). Если в это время идет измерение или появилась команда на запуск измерения — немедленно возвращается ошибка и внутри программы фиксируется время события. Следующая команда измерений или команда включения высокого инициируют включение высокого. Такая схема делает работу программы в рамках автоматизированной системы и сценариев наблюдений устойчивой к ошибкам пересвета.

Проверить наличие высокого напряжения можно командой GET HV, которая возвращает OK HV=ON или OK HV=OFF. При обнаружении эффекта отключения, можно попытаться включить высокое командой. Отметим, что в случае если пересветка продолжается, включение не произойдет и ответ будет не OK, а ERROR. В случае запуска измерений ответ о неудаче будет послан после определенного числа бузуспешных попыток, разделенных временем ожидания.

В состоянии PARKED попытка выполнения команд RUN, SET и большинства команд GET приводит к ошибке. Разрешено выполнение только комад INIT, PARK, QUIT, GET STATUS, GET IDENT, GET ERROR. Команда GET STATUS возращает текущий статус программы, GET IDENT возвращает имя программы и текущую версию в виде OK IDENT="Turbina-core(M) Vers.N.N". GET ERROR возвращает информацию о последней произошедшей ошибке в виде OK ERROR="2006-12-24 20:01:05.92 (661) 11 пуе - Protocol wrong command".

Предусмотрена команда GET FLUX получения значений последнего измерения вне зависимости от того, в какой моде измерения проводились. Мгновенный ответ на эту команду OK FLUX=1,10,100,1000. Аналогично возможно получить последние значения грубых оценок нормальных индексов мерцания: GET NORMAL (см. выше).

Команды GET OBJECT, GET BACKGROUND также реализованы, а смысл команды GET DATA изменен, поскольку ядро не может предоставить результаты обработки, и теперь эта команда возвращает измеренные статистические моменты.

Ошибки выполнения

При работе Turbina-core(M), как и многих других программ, управляющих аппаратурой, могут появляться ошибки выполнения, связанные с

- допущенными ошибками программирования,
- допущенными ошибками установки и конфигурирования,
- неправильным (ошибочным) воздействием на программу пользователями-клиентами,
- аппаратурными проблемами, связанными с обменом данными с прибором.

Следует отметить, что ошибки, связанные с компиляцией и компоновкой, код программы учесть не в состоянии.

Состав ошибок работы ядра зависит от того, в каком из перечисленных выше состояний находится программа. Соответственно меняются (см. раздел о проколировании) получатели и цели сообщений об этих ошибках. В начальном состоянии 1, сообщения об ошибках выдаются только на `stderr`. В случае, если выполнение программы не может быть продолжено — она аварийно завершает свою работу с ненулевым кодом возврата. Забегая вперед,

заметим, что код возврата не может быть больше 255, т.е. полностью описывать случившую ситуацию. Поэтому код возврата формируется отдельно от кода ошибки и в целом соответствует общепринятым соглашениям (см описание glibc, функция exit()). Естественно, код возврата может быть использован порождающим запуск ядра процессом.

На этапе запуска программы могут возникать следующие ошибки:

- Неверные ключи/значения в командной строке
- Не создано дерево директорий
- Не установлены правильные права доступа к директориям
- Нет конфигурационного файла, указанного в командной строке или по умолчанию.
- Конфигурационный файл не доступен для чтения.
- Ошибка создания серверного сокета.
- Ошибка подключения клиента (по крайне мере, первого).
- Ошибка создания динамических объектов в состоянии 1
- Алгоритмические ошибки программы

В силу краткости перечня функций, выполняемых при старте программы, алгоритмические ошибки крайне маловероятны и относятся в основном к другим фазам работы.

В случае, если программа запущена успешно, то возникновение почти всех ошибок связано с выполнением конкретного запрошенного пользователем действия (команды). Небольшая часть ошибок может быть связана с попытками новых соединений и с маловероятными проблемам на этапе разбора команды внешней программы, что также фиксируется. По отношению к выполнению команды, все ошибки являются “фатальными” (в терминологии протокола обмена Supervisor), то есть однозначно указывают на неуспешность выполнения команды. Момент возникновения ошибки может быть разным – запрошенная процедура может быть не выполнена до конца, а может и завершиться, но с ошибочным результатом.

Обработка ошибок замкнута в пределах выполнения команды, то есть ошибочное состояние системы не сохраняется от команды к команды: ядро “не помнит” о проблемах. Способ передачи информации об ошибках между логическими уровнями зависит от случая и включает как генерацию исключений, так и возврат функцией кода ошибки. Ошибки, делающие дальнейшее выполнение команды невозможным и обнаруженные на нижнем логическом уровне, передаются при помощи исключений, перехват которых идёт на уровне выполнения команды клиента. К выходу из программы не приводят никакие ошибки ни в отладочном, ни рабочем режиме.

В программе принята единая система отождествления ошибок исполнения, базирующаяся на целочисленном ненулевом коде ошибки. Ошибки, связанные с системными вызовами, имеют соответствующее значение errno, на основе которого и формируется расширенный код ошибки. Стандартные errno занимают область значений от 1 до 128, ошибки RS485 драйвера — от 300 до 316. Есть еще определенные в системе ошибки NFS в диапазоне 512 – 528. В случае алгоритмических ошибок и предусмотренных ошибок — код назначен достаточно произвольно.

Специфическим ошибкам, связанным с особенностями работы прибора (электроника MASS канала в данном случае и ПЗС камера и ее драйвер в случае DIMM канала), с особенностями измерительного процесса и со структурой программы присвоены коды из диапазона 600 – 699. Коды равные 0 и 699 отведены для предупреждающих сообщений,

т.е. для ситуаций не мешающих правильной работе прибора, но требующих внимания со стороны операторов.

Каждому коду ошибки соответствует его строковое представление (см. описание файла протокола), которое генерируется специальным модулем. Текстовое представление всегда выводится на `stderr` и записывается в файл протокола, если ядро находится в состоянии 3. В состояниях 2 и 3 внешние программы получают информацию о характере ошибки с помощью команды GET ERROR. Возвращаемое при этом значение строки ERMSG полностью совпадает со последней строкой, выданной на `stderr` и записанной в протокол. Возникновение ошибки может быть зафиксировано анализом результата выполнения команды (ERROR).

После подсоединения клиента (перехода программы в состояние 2) добавляется только один тип ошибки — реакция на синтаксически неправильную команду. Зато, при активизации функций программы, после получения команды INIT, круг возможных источников ошибок расширяется.

Инициализация касается всех компонентов программы – как отвечающих за аппаратуру объектов, так и объектов-мод работы. При создании каждого нового объекта возможны ошибки их создания, при дальнейшем функционировании — ошибки функционирования объекта. Каждый объект в момент инициализации полностью "узнаёт", с какими параметрами придётся работать в дальнейшем и проверяет, определены ли они в конфигурационном файле. Поэтому, при инициализации явные ошибки конфигурации сразу же проявляются. В процессе парковки таких специфических ошибок не возникает.

Ниже приведена объединенная таблица кодов специфических ошибок.

```
ERR_CFG_STRUCT      600     "Invalid structure"
ERR_CFG_MISSED      601     "Missed in configuration"
ERR_CFG_CHANNEL     603     "Invalid channels configuration"
ERR_INI_OBJECT      610     "Is not been created"
ERR_NOT_CONFIGURED  611     "Module is not configured"
ERR_UNKNOWN_ADDRESS 612     "Unknown address"
ERR_CENT_EMPTY      620     "Centering frame is empty"
ERR_CCD_EMPTY       621     "CCD frame is empty"
ERR_NORM_EMPTY      622     "Subframe is empty"
ERR_NORM_SHIF       623     "Center mode wrong data"
ERR_IEEE_BUS        630     "IEEE1394 bus"
ERR_CAM_NOTF        631     "IIDC camera initialization"
ERR_CAM_IDEN        632     "No specified camera found"
ERR_CAM_INIT        633     "IIDC camera initialization"
ERR_CAM_ERR         634     "IIDC camera setting"
ERR_DRIVER_VERSION  640     "Unsupported driver version"
ERR_EXCHANGE        641     "Exchange error"
ERR_NOT_REPLY       642     "Module doesn't respond"
ERR_IDENT           643     "Module has wrong identification"
ERR_TEST            644     "Test failed"
ERR_OVRLIGHT        645     "Overlight! HV is off"
ERR_ALG             650     "Algorithmic Error"
ERR_REG_STRUCT      651     "Invalid regular expression"
```

```
ERR_EMPTY_MODE      652     "Empty scenario run"
ERR_NOTHING_DONE    653     "No measurement done yet"
ERR_PROT_SYNT       660     "Protocol syntax error"
ERR_PROT_WRCOMM     661     "Protocol wrong command"
ERR_PROT_NOKWD      662     "Protocol keyword missed"
ERR_PROT_WRKWD      663     "Protocol wrong keyword"
ERR_PROT_SCEN       664     "Invalid symbol in scenario"
ERR_PROT_NOVAL      665     "This keyword(s) requires value"
ERR_CMD_OPTION      671     "Illegal options"
ERR_STD_EXCP        690     "Std::exception"
ERR_UKN_EXCP        691     "Unknown type exception"
```