# Theory of operation

From GridLAB-D Wiki (UVic shadow)

**Theory of operation** - GridLAB-D core **theory of operation**

## Contents



## Synopsis

**core/exec.c (C code)**

```
STATUS exec_start(void)
```

## Description

During command line processing, GridLAB-D loads one or more GLM files listed and uses the directives in those files to set global variables, set global variables, run scripts, load modules, define classes, create objects, and link to external applications. Once all the objects are created and initialized, the main exec loop is started and the global clock begins to advance.

### Clock operation

GridLAB-D's main processing loop advances the global clock by varying time steps depending on the pending state changes of the object's defined in the model until no object reports that is has a pending state change.

Pending state state changes are called *sync events*. There are two types of sync event:

**hard events**
: these are events that must be handled in order for the simulation to continue. If there are no hard events pending, the simulation will stop.

**soft events**
: these are events that are only handled if they occur before a hard event. If there are only soft events pending, the simulation will stop.

The stoptime global variable is a special hard event if it is not set to NEVER. If the stoptime is set to NEVER, the simulation will only stop when there are no pending hard events. Otherwise the simulation will stop when the stoptime is reached.

### Object ranks

Objects are assigned ranks when they are created. The rank strategy varies depending on the solver being used and the parent-child relationships between the objects. Normally, the modeler only controls the parent-child relationship by either nesting objects when they are defined, or using the name property. Modelers cannot strictly control the rank of objects because modules will generally apply their own ranking rules to ensure a

stable solution.

## Synchronization procedure

Objects are instances of classes. Each class has a synchronization characteristic the determines when it receives synchronization signals from the main exec loop. The sync signals are sent in the following order

- **create** - called once per object
- **init** - called until object confirms successful initiatialization (only called once prior to Hassayampa (Version 3.0))

*do until clock stops*

- **precommit** - called once per timestep before sync; used to setup for new timestep

*do until valid timestep is found*'

- **presync** - called once per pass from the top rank down; used to prepare objects for bottom-up pass
- **sync** - called once per pass from the bottom rank up; used to perform main calculations in objects
- **postsync** - called once per pass from the top rank down; used to complete calculations

- **commit** - called once per timestep after sync; used to lock-in states

- **finalize** - called once per simulation after clock stops
- **term** - called to terminate simulation

## Parallelization and multithreading

Any iterative process that does not have interdependencies can be run in parallel using multiple thread. When the threadcount is not 1, GridLAB-D will create threads to handle a wide variety of loops that can be operated in parallel.

## Realtime operation

When running in run_realtime mode, GridLAB-D limits how fast the clock will advance based on the value of the realtime global variable.

## Internal synchronization events

There are a number of internal synchronization events that take place while the main exec loop is running.

**Before precommit**

- *Link* - links are updated first
- *Instance* - All slave instances are resumed
- *Random* - All random variables are updated
- *Schedule* - All schedules are update
- *Loadshape* - All loadshapes are updated
- *Transforms* - Schedule transforms are updated
- *Enduse* - All enduses are updated
- *Heartbeat* - All heartbeat signals are sent

**Before commit**

- *Transforms* - Non-schedule transforms are updated.
- *Instances* - Slave instances are waited on

**After commit**

- *Scripts* - Script sync events are run.

### Debugging

Debugging the main exec loop can be very challenging. The --debugger command line option can be used to facilitate the process. In addition, copious use of the tape module's recorder object, particularly with intervals set to 0 or -1 can also be very helpful.

## See also

- Guide to Programming GridLAB-D
  - Introduction
    - Developer prerequisites
    - Programming conventions
    - Build/release process
    - Documentation Guide
    - **Theory of operation**
  - Creating a module
    - Module globals
    - Module functions
    - Subsecond processing
    - Import/export
    - Check
    - KML output
    - Example 1
  - Creating a class
    - Class functions
    - Class globals
    - Publishing properties
    - Publishing methods
    - Notifications
    - Load methods ***New in 3.2!***
    - Example 2
  - Special Topics
    - Data types
    - Multithreading
    - Application links
    - Realtime server
    - Graphical user interfaces
    - Troubleshooting messages
    - Example 3
  - Source documentation
    - C/C++ Module API documentation (trunk) (http://gridlab-d.sourceforge.net/doxygen/trunk/group__module__api.html)
    - C/C++ Module API Guide

Retrieved from "http://gridlabd.me.uvic.ca/wiki/index.php?title=Theory_of_operation&oldid=6399"

---

- This page was last modified on 17 April 2013, at 13:29.