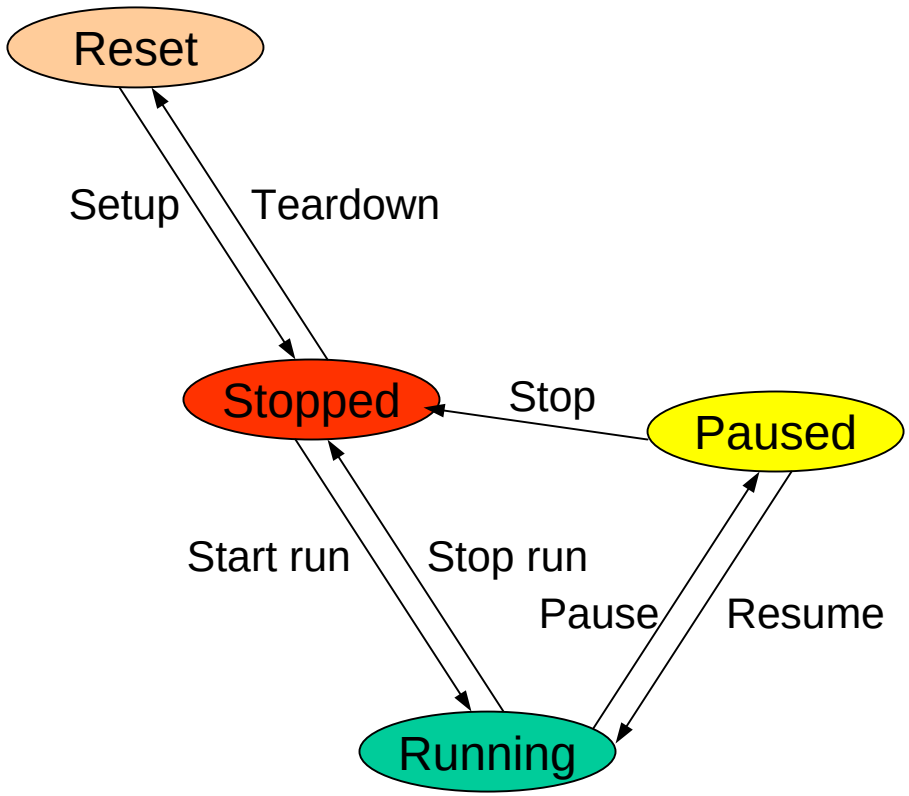
Proposed State Diagram



Anatomy of a Test

- The Run Control process implements the state transition diagram
- A test/application is selected from a Run Control GUI
- Run Control coordinates the test/application's execution
- Standard actions are carried out by Run Control on state transitions
  - Run number assignment on StartRun
  - Trigger enable on StartRun and Resume
  - Trigger disable on StopRun and Pause
- The test/application optionally implements a callback per state transition:
  - Setup – Loads a schema and configuration appropriate to the hardware
  - Teardown – Removes the existing schema and prepares for a new one
  - StartRun – Selects a trigger mask to use
  - StopRun – Generally nothing to do by application?
  - Pause – Generally nothing to do by application?
  - Resume – Generally nothing to do by application?
  - Stop – Generally nothing to do by application?
- The test/application takes data in a separate thread from Run Control

Procedure to Run a Test

- Launch procedure that creates directory and checks (runnable?) system (core (?), test suite) out from CVS
  - cvs update if this is a rerun and the dir & files already exist
- cd to the new directory
- Set the PYTHONPATH correctly to point to core and 3rd party products
- Start Run Control GUI and select a test from the suite
- At completion “cvs add” and “commit” output files
  - Is cvs check-in necessary for all involved files (e.g. core/test python scripts) in case they were modified?
- Tag with test name and date/time
- Update test report database record with this tag
- Optionally (?) remove the created directory after the run?

Inputs for Test Report

- Test suite name
- Name of test that was run
- Date & time of run
- Completion status (Success, Failure (code?), abort)
- Input files used (schema, configuration, command database)
- Output files generated (Message log, event data, analyzed data (plots, etc.))
- Version strings (also entered in message log?)
  - Software
    - Granularity of interest?
  - Hardware
    - All hardware components *should* have a software-readable version register. Alas, some don't. What to do about these?
- CVS tag of software used to run test, input files and output files
    - Remote access issues
- Operator notes string(s)

    - How to capture non-software accessible external inputs, e.g. system clock dialed in to 27 MHz? Operator note is not robust enough?

Test Report Generation

- Inputs for Test Reports form a row in a *local* relational database
  - "local" means local to test stand workstation
- Local database to be periodically uploaded to Oracle at SLAC
- Test configuration cvs repository to be maintained at SLAC?
  - Is this the best way vs local cvs repository?
  - Test configuration control using some other tool?
- Test report will be accessible from the Web