TLB – Top Level Baseline

application and component template/tool

Norman Kirchner
Norm.Kirchner@NI.com

Top Level?

- A VI that can be considered the primary of a system
  - or major component of a system
- Typically has front panel shown
  - but not a requirement
- Controls flow of application and processing
  - Processing happens within executed states
- Runs for lifetime of application or component

Get Installed

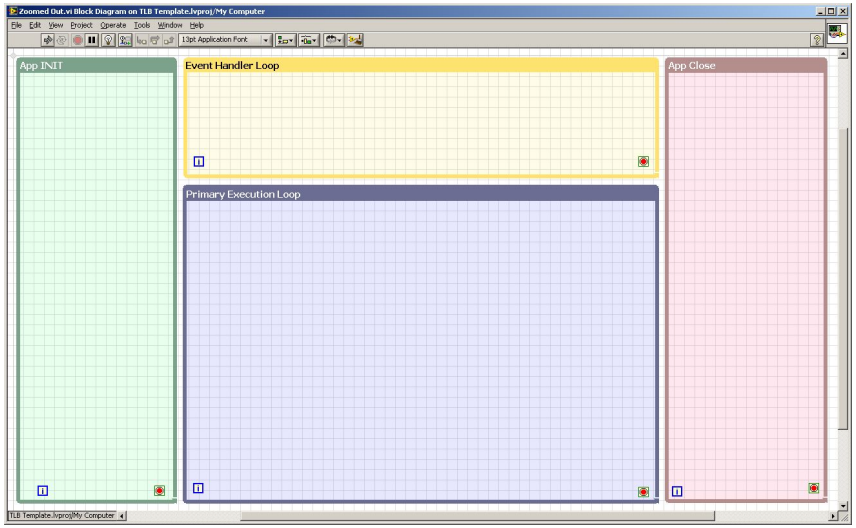
- Install VI Package Manager (free community edition)
  - www.jki.net
- Install package
  - http://lavag.org/topic/13003-tlb-top-level-baseline/
- Restart LV
  - Refreshes tools menu

Get Developing

- Tools>>TLB – New Baseline
  - Follow Interactive Dialog

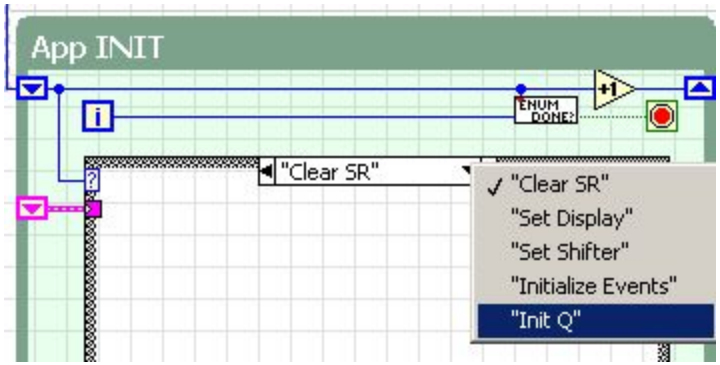
Baseline Architecture

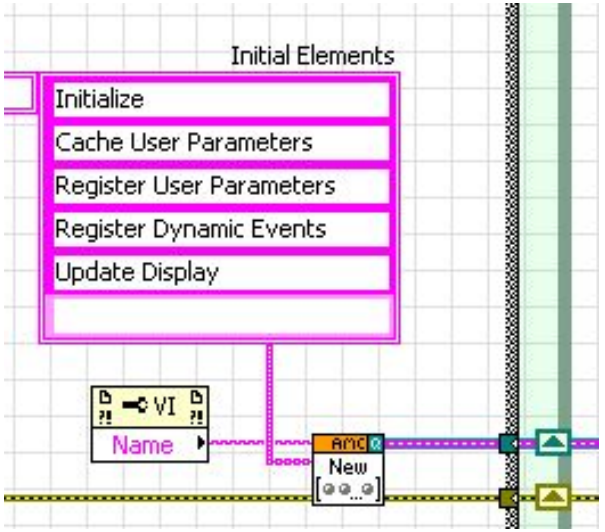
- Application INIT
  - once and only once code
- Event Handler Loop
  - user interaction response
- Primary Execution Loop
  - main flow of program
- App Close
  - close references & shutdown



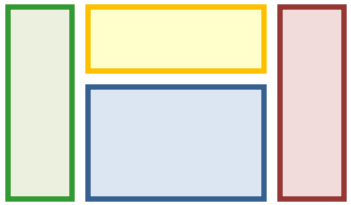
Application INIT

- Sequence of cases
  - primes the application
  - stacked instead of flat for cleanliness
- Not ‘all’ init. code goes here
  - Just once and only once stuff
- Enqueues specific set of states to Primary Execution

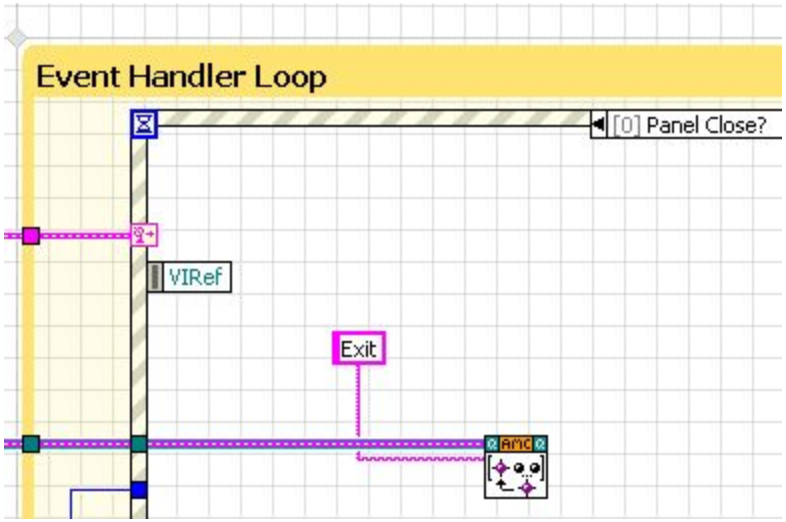




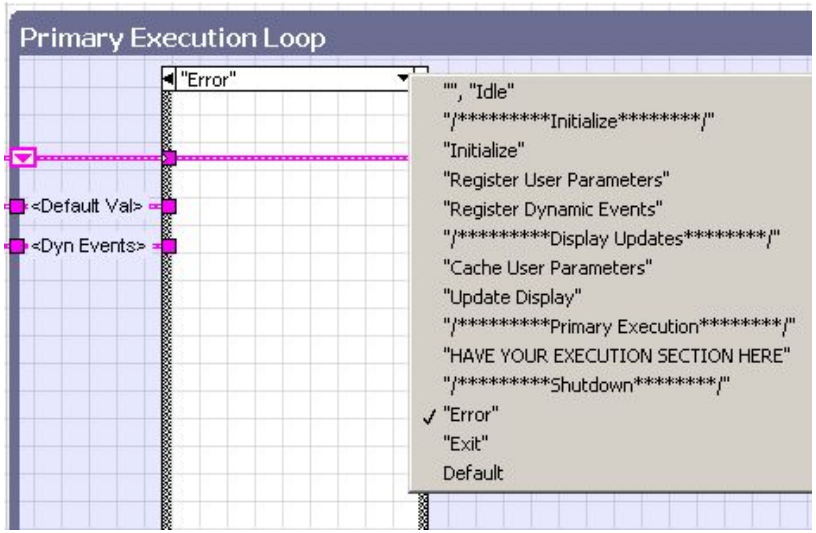
Event Handler



- Responsive to button pushes and value changes
- Pre-defined cases handle window close and shutdown
- Source of queued messages to Primary Execution



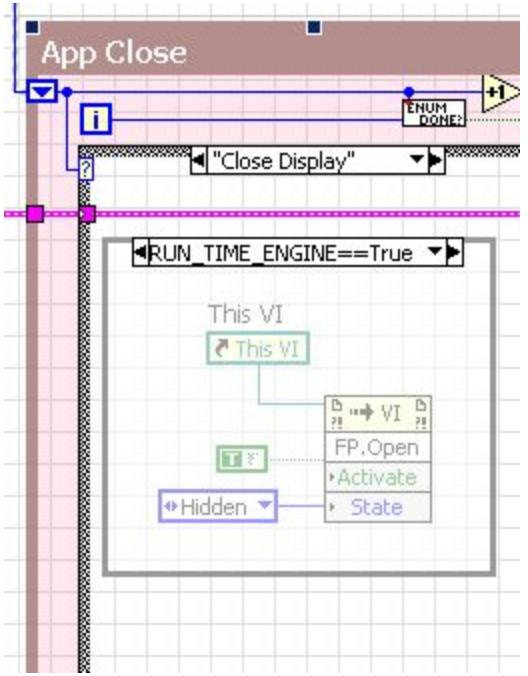
Primary Execution



- Pre-defined cases handle common functions
  - Init
  - Error
- 95% of work happens here
- ‘Default’ case handles state typos

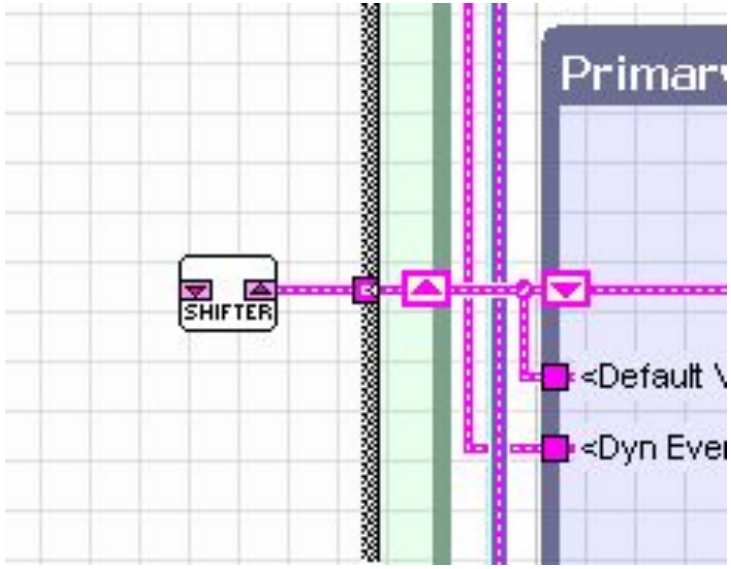
Application Close

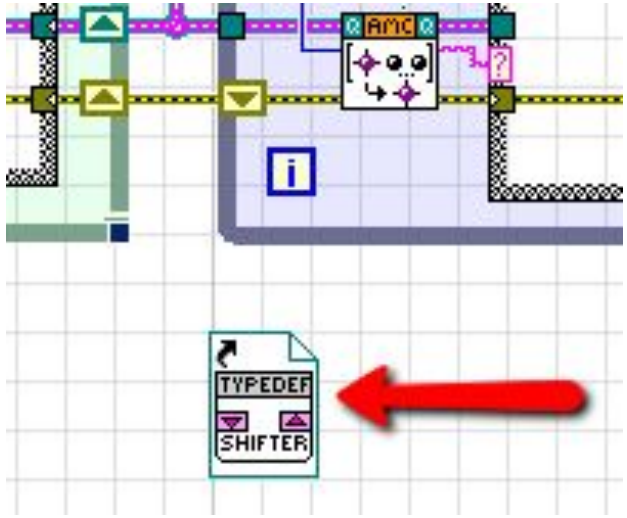
- Handles display consideration if built into EXE
- No real processing should happen here
  - Use ‘Exit’ in Primary Execution



The SHIFTER

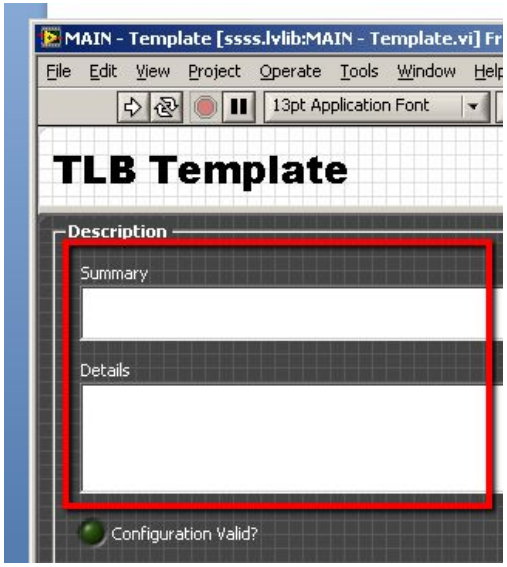
- Master state cluster
  - ALL data associated with application
- NEVER send into sub-VI
- Easily add elements through Typedef shortcut

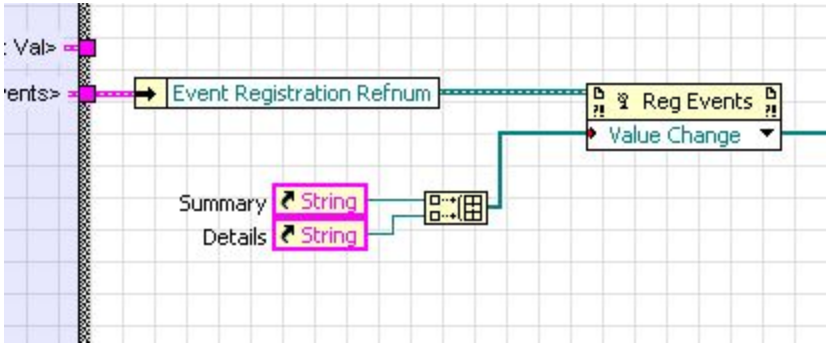


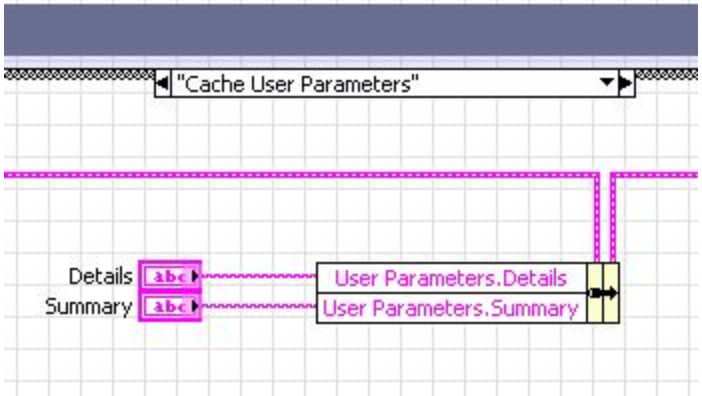


‘User Parameters’

- Scalar inputs on the front panel
  - numerics, strings, etc.
- Register for ‘value change’ event dynamically
  - Prevents event structure bloat
- All terminals in 1 state: Update User Param.
  - Enables all param to be accessible to entire program without need for local Var

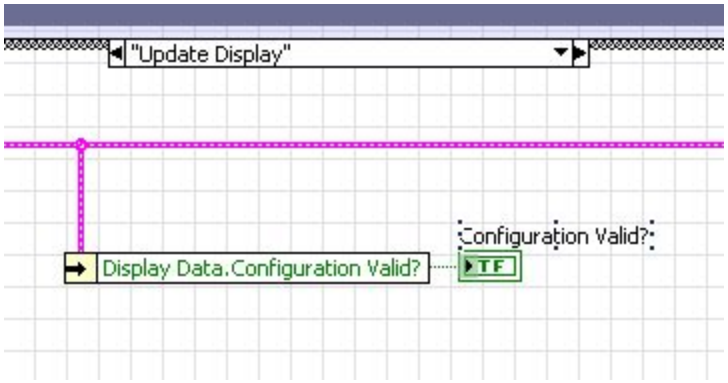






‘Display Data’

- Scalars, plots etc
  - Any thing that needs to get updated on the panel
- Store in shifter to provide data access to entire app
  - Instead of outputting directly to indicator
- Call ‘Update Display’ state to refresh display



Extra Handy Things

- Gracefully shuts down both loops
- Idle state defaults to 1s delay
- Caches current values of controls on startup in case of change while not running
- Graceful error in case of typo
- Interactive or headless execution
  - Your choice
- Error handling allows Stop OR continue

Closing note

- Appropriate for simple and intermediate grade applications
- THIS IS NOT OVERKILL – it has all those things that you end up implementing anyway