Folders

Project related

VI’s

Functions

### LabVIEW at start Options

* 1. Front Panel - Control Style for New VIs - Silver Style
  2. Block Diagram - General - Place front panel terminals as icons (uncheck)

### Prepare folders for the project

Main - name of the objective

* 1. Modules
     1. Templates
        1. API
        2. SubVIs
        3. Controls
        4. QueueThings
     2. Other modules (just the folder name)
  2. User Events
     1. Controls

### Create Project - save in the Main Folder

### In the Project - add Folder (Snapshot) Main Folder

### Create the User Event Abort VI (it is a FGV inside an Error Case)

* 1. Save in the User Events Folder
  2. You must also create the enum Type Def Control - UEA\_Action (Create, Get, Destroy)
  3. Save the Control in the Controls Folder inside User Events Folder
  4. In the Create mode use Create User Event type Boolean
  5. In the Get mode just bypass the User Event Refnum
  6. In the Destroy mode use Destroy User Event and create a constant in the Refnun
  7. If an error occurs, clean the error and use a constant in the Refnum

### Create Main VI in the Main Folder (save in the same location)

* 1. In the Main VI create errors in and out
  2. Also the User Event Abort VI - 2 instances in the beginning and ending (modes Create and Destroy)
  3. A big Case Structure for No Error/Error
  4. Document the code - Front Panel and Block Diagram

### Transform each Module Folder into Library inside the Project

* 1. Don’t make the Template Folder into Library
  2. Save each Library in the Matching Folder

### Create Main\_Module in the Template

1. 2 loops in parallel

### Create FGV\_Queue in the QueueThings Folder with enum Get, Set, Release (save as Type Def Control)

* 1. The FGV must have the Release Queue function in the release, but only control and indicator in the other modes
  2. The cluster (the enum Modes and a Variant) used in the queue, must also be saved as Type Def Control (it will not be used in the FGV\_Queue, but as it will be needed to create the Control and Indicator of the queue - using Obtain Queue it is a must - delete it later - it will be used inside the Pre-Loop to do the first Set queue)
  3. Don’t forget to draw the icon, document code and write the VI description

### Create the Dequeue VI and Enqueue VI, both should be in the QueueThings Folder

* 1. Don’t forget to draw icons, document codes and write the VI’s description

### Create the Pre-Loop VI, there will be all the code that needs to run before the loops

* 1. For the default, it’s needed at least the Event Registration Refnum (Get the Event of the User Event Abort VI) and the creation of the module\_Queue (use the name from the Main\_Module VI to use in the Obtain Queue and Set this in the FGV\_Queue)
  2. If it is needed, start the Queue creating a Start VI that will put the Start mode in the Queue.
  3. The Pre-Loop VI can be put in the SubVIs Folder

### In the Main\_Module, the Event Structure needs to show the Event Dynamic Registration and have the Abort Event (from the User Event Abort) handled by the Event Case

* 1. In the Abort Event Case must have an End VI/Stop VI (that will send the End Mode in a cluster through the Queue)
  2. And a Stop for the While Loop

### Also in the Main\_Module, in the second While Loop an error treatment is needed as soon as the loop starts

* 1. A case structure for error with the Dequeue VI in the No Error case, and a cluster with the mode error and a clear error in the Error case
  2. An unbundle by name and a case structure with the possible modes
  3. The End mode will end the while loop

### Delete the Template from the Project, copy and paste the content folder into the Module. Copy and paste them to the Project inside the library

### Read through the exam and identify the API’s needed

* 1. Do the enum’s and VI’s
  2. Don’t forget documentation

### Make the Interfaces as shown in the sketch

### Start checking the requirements

* 1. Don’t forget documentation

### Review