Style Rules Checklist

This checklist lists all the style rules presented throughout The LabVIEW Style Book. Use this when performing code reviews, as discussed in section 10.1.2, “Manual Checklist,” of Chapter 10, “Code Reviews.” Edit the contents to customize a style convention for your organization. You can also use it as a quick look-up reference when discussing style with peers.

Chapter 2

2.1 Maintain a LabVIEW project journal

2.2 Write a requirements specification document

2.3 Maintain good LabVIEW style throughout the proof of concepts

2.4 Document your LabVIEW options and back up the LabVIEW.ini file

2.5 Develop reusable SubVIs

2.6 Make reusable libraries accessible from the LabVIEW palettes

2.7 Place reusable templates in the LabVIEW\templates folder

2.8 Maintain an organized repository on disk

2.9 Create an LabVIEW source folder hierarchy that reflects your application’s architecture

2.10 Create the folder hierarchy before you begin coding

2.11 Organize LabVIEW source files into cohesive project libraries, where appropriate

2.12 Create unique and intuitive source filenames

2.13 Do not abbreviate filenames

2.14 Never use LabVIEW’s default filenames

2.15 Identify the top-level VIs

2.16 Follow your organization’s CM Rules

2.17 Avoid moving source files on disk

Chapter 3

3.1 Group related controls using decorations, spacing, tabs, and clusters

3.2 Apply symmetry and spacing to front panel objects

3.3 Size similar objects the same

3.4 Maximize the top-level VI panels for industrial applications

3.5 Size dialog VI panels much less than full screen

3.6 Center dialog VI panels

3.7 Use LabVIEW’s dialogs for desktop applications, avoid them for industrial applications

3.8 Use system controls for desktop dialogs; use 3D controls for industrial dialogs

3.9 Enlarge and center the objects of an industrial GUI VI in proportion to their importance

3.10 Limit the quantity of information displayed on a GUI VI panel

3.11 Avoid overlapping visible objects

3.12 Hide the toolbar

3.13 Include your company logo for a professional appearance

3.14 Use default appearance for subVI panels, objects, and most text

3.15 Arrange controls to resemble the connector assignments

3.16 Resize the panel for a snug fit

3.17 Minimize front panel text

3.18 Delete template instructions immediately after edits are performed

3.19 Apply consistent fonts and capitalization

3.20 Choose only one font, and vary the size, boldness, and color to obtain multiple styles

3.21 Use succinct, intuitive control labels and embedded text

3.22 Apply 13-point black Application font for most subVI panels

3.23 Provide default values and units in parentheses at the end of owned labels

3.24 Combine bold text labels with plain text parentheses, for control labels of commercial subVIs

3.25 Maximize the contrast between text color and background color

3.26 Use large text size for command buttons and critical data

3.27 Allow extra space between labels and objects for multiplatform applications

3.28 Apply color judiciously

3.29 Create a color theme

3.30 Follow universal conventions for green, yellow, and red

3.31 Leave the panel and objects of subVIs gray

3.32 Keep the color schemes simple and time limited

3.33 Limit the number of controls that are visible and enabled at any one time

3.34 Restrict the range of all controls to values that are relevant to the application

3.35 Set the Data Range property of numeric controls

3.36 Use ring or enumerated controls over string controls when feasible

3.37 Set the Tabbing Order and use the Key Focus property to aid navigation

3.38 Customize the runtime menus for top-level VIs

3.39 Always include a Help menu or button

3.40 Be consistent!

Chapter 4

4.1 Use 1280 1024 display resolution

4.2 Leave the background color white­

4.3 Use a high object density

4.4 Limit the diagram size to one visible screen, or limit scrolling to one direction

4.5 Create a multilayer hierarchy of subVIs

4.6 Modularize top-level diagrams with subVIs

4.7 Modularize the high-level subVIs with lower level subVIs

4.8 Do not create subVIs just to save space

4.9 Avoid trivial subVIs containing few nodes

4.10 Create a meaningful icon and cohesive description for every subVI!

4.11 Minimize wire bends; eliminate kinks and loops

4.12 Maintain even spacing of parallel wires

4.13 Tunnel wires through left and right borders of structures

4.14 Do not wire through structures unnecessarily

4.15 Never obstruct the view of wires and nodes

4.16 Limit wire lengths such that source and destination are visible on one screen

4.17 Never use local and global variables for wiring convenience

4.18 Label long wires and wires from hidden source terminals

4.19 Place unwired front panel terminals in a consistent location

4.20 Modularize wires of related data into clusters

4.21 Save clusters as type definitions

4.22 Always flow data from left to right

4.23 Propagate the error cluster

4.24 Avoid array and cluster coercions

4.25 Create controls and constants from a terminal’s context menu

4.26 Disable dots at wire junctions

4.27 Avoid Sequence structures unless required

4.28 Avoid nesting beyond three layers

4.29 Use write local variables for initializing control values

4.30 Use global variables for simple data sharing between parallel loops or VIs

4.31 Use a Sequence structure to order operations if no data dependency exists

4.32 Use only Flat Sequence structures when required

4.33 Avoid polling variables within continuous loops

4.34 Avoid variables if wires are feasible

4.35 Use shift registers over local and global variables

4.36 Group most shift registers near the top of the loop

4.37 Label wires exiting the left shift register terminal

4.38 Use looped Case structures over Sequence structures

Chapter 5

5.1 Have fun creating icons

5.2 Create a unique and meaningful icon for every VI

5.3 Never use LabVIEW’s default icons

5.4 Save VIs with subVI icons visible instead of terminals

5.5 Use a black border

5.6 Combine a glyph with color and text for best style

5.7 Choose universally recognized glyphs

5.8 Use 8- or 10-point small fonts for most text

5.9 Choose a unified style for related VIs

5.10 Budget your time in proportion to intended VI reuse

5.11 Use textual foreground, colored background, and a black border for fastest results

5.12 Contrast the text and background colors

5.13 Choose a color convention for the type of VI

5.14 Create an icon template for related VIs

5.15 Reuse one glyph, color scheme, and font for related VIs

5.16 Copy graphics

5.17 Avoid text and graphics not universally understood on international icons

5.18 Choose connectors and terminal assignments that promote clear wiring and proper data flow

5.19 Select a pattern with extra terminals

5.20 Choose a unified pattern for related VIs

5.21 Use the 4x2x2x4 connector pattern for most VIs

5.22 Assign controls to left terminals, indicators to right terminals

5.23 Never cross wire stubs in the Context Help window

5.24 Specify terminal assignments resembling the panel layout

5.25 Assign error clusters to bottom left and right terminals

5.26 Assign references and I/O names to top left and right terminals

5.27 Choose left and right vertical edge connector terminals for high priority inputs and outputs

5.28 Choose top and bottom horizontal edge connector terminals for lower priority inputs and outputs

5.29 Specify required priority for critical inputs and outputs

5.30 Specify optional priority for inputs and outputs that are normally not used

Chapter 6

6.1 Choose controls that simplify the operation of the panel

6.2 Choose memory efficient data types

6.3 Choose controls and data types that facilitate consistent data structures throughout an application

6.4 Configure an appropriate default value for each control

6.5 Enter control descriptions

6.6 Save custom controls as strict type definitions

6.7 Create arrays and clusters that associate related data

6.8 Use Booleans if two states are logical opposites

6.9 Assign names that identify the TRUE and FALSE value behavior

6.10 Use command buttons for action, slide switches for parameter settings

6.11 Label the TRUE and FALSE states of slide and toggle switches

6.12 Avoid using buttons or switches as indicators, and LEDs as controls

6.13 Use I32 representation for integers and DBL for floating-point numbers

6.14 Use automatic formatting unless a specific format is required

6.15 Show radix for hex, octal, or binary data

6.16 Use enums liberally throughout your applications

6.17 Save enums as type definitions

6.18 Avoid string controls on GUI VI panels unless required

6.19 Use enum, ring, and path controls in place of string controls where possible

6.20 Keep the Browse button visible for path controls on GUI VI panels

6.21 Use arrays for multivalued data items; use clusters for grouping multiple distinct items

6.22 Use arrays to store large or dynamic length data sets

6.23 Enter descriptions for array and cluster shells and control elements

6.24 Use alignment tools to keep clusters neat and compact

6.25 Save all clusters as type definitions

6.26 Always use Bundle and Unbundle By Name

6.27 Avoid clusters for interactive controls with Dialog VIs

6.28 Organize complex data using nested data structures

6.29 Avoid manipulating nested data during critical tasks

6.30 Limit the size of arrays by initializing to maximum length

Chapter 7

7.1 All VIs must trap and report the errors returned from error terminals

7.2 Trap errors by propagating the error cluster among the error terminals

7.3 Trap errors from all iterations of loops

7.4 Disable indexing of errors with continuous loops

7.5 Trap all errors from all nodes that have error terminals

7.6 Report errors using a dialog and/or log file

7.7 Use General Error Handler VI over Simple Error Handler VI

7.8 Implement an error log file for application deployment

7.9 Suppress dialog error reporting for unattended or remote operation

7.10 Avoid subVIs with built-in error reporting

7.11 Maintain user-defined error codes within an XML file

7.12 Use negative codes for I/O device errors, and positive codes for warnings

7.13 Skip most subVI diagrams on error using an Error Case Structure

7.14 Use unwired defaults over constants for output tunnels of Error case

7.15 Use the SubVI with Error Handling template

7.16 Error trapping is required for nodes that perform I/O operations, recommended for nodes that contain error terminals, and optional for diagrams that do not contain nodes with error terminals

7.17 Tunnel the error cluster near the bottom of structures

7.18 Leave automatic error handling disabled

Chapter 8

8.1 Use multiple criteria for the loop condition

8.2 Use a Timed Loop for highly precise or complex timing, and use a While Loop otherwise

8.3 Include a delay within continuous While Loops

8.4 Avoid polling GUI objects

8.5 Use the Value Change event for most GUI controls

8.6 Place control terminals within their Value Change event case

8.7 Resize the Event Data Node to hide unused terminals

8.8 Avoid continuous timeout events

8.9 Use a state machine design pattern in most VIs of medium or greater complexity

8.10 Derive the application’s primary states from the specification or design document

8.11 Divide the primary states into additional states

8.12 Use an enumerated type definition for the case selector

8.13 Minimize code external to the Case structure

8.14 Include states for Initialize, Idle, Shutdown, and Blank

8.15 Avoid timeout with the Enqueue and Dequeue Element

8.16 Use queues, shared variables, or RT FIFOs for parallel loop messaging

8.17 Prioritize loops using delays or thread priorities

8.18 Size parallel loops to the same width and align vertically

8.19 Minimize space between loops

8.20 Label each loop in the top left corner

8.21 Use the Call By Reference Node over the Run method

8.22 Choose standard connector terminal assignments

8.23 Maintain all components within a dedicated directory

8.24 Pass data between components with shift registers

8.25 Assign one input and one output terminal as type variant

8.26 Display plug-in panels within subpanels

8.27 Create a parallel loop for each cohesive parallel task

8.28 Include loops for Event Handling, Main State Machine, Hardware I/O, and Error Handling

8.29 Pass control references into subVIs via a type-defined cluster

8.30 Avoid subVI from selection with continuous loops

8.31 Keep the Event-Handling Loop at the top level

8.32 Keep high speed display updates at the top level

Chapter 9

9.1 Keep front panel terminal labels visible on the diagram

9.2 Apply free label comments in select locations

9.3 Label every subdiagram of every multidiagram structure

9.4 Label algorithms, constants, and Call Library Function Nodes

9.5 Use default 13-point plain black application font for all diagram text

9.6 Leave notes for the development team

9.7 Use enumerated data types with Case structures

9.8 Create the icons and descriptions as you develop your source code

9.9 Provide online documentation with deployed applications

Chapter 10

10.1 Enforce your organization’s style convention using code reviews

10.2 Utilize a combination of self-reviews and peer reviews for best results

10.3 Perform a self-review prior to every peer review

10.4 Use the VI Analyzer to automate tedious inspections

10.5 Customize the VI Analyzer’s test criteria

10.6 Customize the rank for each test according to the priority of each rule

10.7 Use a manual checklist to perform a comprehensive style review

10.8 Perform at least one peer review per project

10.9 Involve the following people in a peer review: project manager, lead developer, experienced peer, and an inexperienced peer

10.10 Bring the requirements specification and style rules checklist to the peer review

10.11 Designate a neutral party to take notes on desired changes

10.12 Do not modify the functional source code during the peer review