

Index

A

Absolute file paths, 53–55
 Access scope, 329–331
 Acquire Semaphore function, 394, 396
 Action engines, 106–114, 298–299
 architecture, 106–107
 counters, 110–111
 data encapsulation compared to, 298–299
 feedback node, 107–110
 kernel sub-VIs, 112–114
 loop implementation, 106–107
 state variable method, 111–114
 Add function, 9–10
 Alley, 90–91
 Append To Log File sub-VI, 54, 56
 Append To Text sub-VI, 43
 Application Directory function, 54–56
 Application Run class, 358
 Application Run Done sub-VI, 75
 Application Run Output sub-VI, 75
 Architecture selection, 441–442
 Arguments, 254–286, 439–440
 array of states for, 268–270
 car wash controller using, 274–286
 clusters used for passing, 257–260
 data highway, 255–257
 disadvantages to using strings, 266–268
 implementation using, 281–286
 improving string-based state machines, 268–273
 loops used for queue states, 268–273
 parsing sub-VIs, 260–262, 276–278
 passing information with, 257
 pros and cons of, 439–440
 queued string state machines, 263–266
 Run Test states, 255–260
 state machines with, 254–273
 string-based VIs, 263–266
 strings used for passing, 260
 time and cycle, 275
 Array Size function, 71–72
 Arrays, 13–15, 268–270
 LabVIEW use of, 13–15
 string-based state machines using, 268–270
 ATM controller, 246–253, 463–468
 accounts file specification, 468
 CLD exam sample, 463–468
 CLD exam solution, 246–253
 controls and indicators, 466
 menu options, 467–468
 message types, 466–467
 modes, 246
 sequence of operation, 464–465
 type definitions, 247–248
 Auto-indexing, 202
 Auto Reset action engine, 128–129
 Auto Reset Time Check method, 127–128
 Auto Reset Timer Kernel sub-VI, 128

Auto Reset Timer With Elapsed Time action engine, 243

B

Block diagram, 3–4
 Blocking action, 389–390
 Boiler controller, 74–86, 289–294, 313–314, 353–367, 449–453
 arguments in, 290–294
 class inheritance and, 357–367
 CLD exam sample, 449–453
 controls and indicators for, 453
 data encapsulation using data highway in, 313–314
 event log file specification, 453
 inheritance and composition for, 353–355
 Load Sequence state, 353–354
 sequence of operations, 449–452
 Step state, 353–354
 string-based, 289–290
 sub-VIs for, 75–86
 Boolean controls, 8, 15, 20–22
 Bottom-up design, 132
 Bridge circuit calibration, 210–211. *See also* Data acquisition
 Build Array function, 157, 159
 Build Option Array sub-VI, 88
 Build Option String Array subVI, 296–297
 Build Option String subVI, 295
 Build Path function, 54–56
 Build Strain Task function, 215–216
 Build Wash Array sub-VI, 226
 Bundle By Name function, 16–18, 148
 Bundle function, 16–17

C

Car In Position sub-VI, 86–87
 Car Wash 2 Time Check sub-VI, 314
 Car Wash Controller, 63–67, 132–150, 178–181, 197–209, 274–286, 397–404, 454–457
 Air Dry state, 148–150, 209
 arguments in, 274–286
 case structures, 135–139
 CLD exam sample, 454–457
 controls and indicators for, 456–457
 Cycle state, 274–275, 278–281
 cycles and timing, 64
 default values, 138–139
 Deluxe Wash button, 203–204
 determination of states, 133–134
 Economy Wash button, 203–204
 event-driven, 178–181
 front panel, 135
 implementation with arguments, 281–286
 Initialize state, 142–144, 197–198
 Lock Selector state, 134, 146–147, 206, 346–347
 main loop, 135–139
 Main Wash state, 148, 150, 208
 non-square icons, 199–201
 operating rules, 64–66
 parsing routines for, 276–278

- producer–consumer, 397–404
- proximity switches, 137
- purchase options, 63–64
- purchase selector, 203–206
- queue reference wire, 197–199
- queued, 197–209
- queues and event structures in, 20–203
- race conditions, 397–404
- Rinse* state, 134, 149–150, 208
- semaphores and, 402–404
- sequence of operation, 455–456
- Shutdown* state, 140–141, 151, 198
- Soap Application* state, 148–149, 207
- state enumerated constant, 97–98, 134–135
- state machine applied to, 132–150, 178–181, 197–209
- state transition diagram of, 66–67
- state transitions for, 134
- STOP button, 139–142, 180–181, 203–206
- Switch When Pressed* buttons, 203–204
- task, 63, 454
- time and cycle arguments, 275
- Underbody Wash* state, 147–149, 206–207
- Unlock Selector* state, 145, 201
- user events and, 400–401
- Wait* state, 134, 145–146, 178–179
- Car Wash Cycle Data Highway sub-VI, 299–301
- Car Wash Cycle Timer Class sub-VI, 309–310
- Car Wash Lamp Map sub-VI, 279–280
- Car Wash Switch Map sub-VI, 278–279
- Car Wash Timer Kernel sub-VI, 121–122
- Car Wash Timer sub-VI, 124
- Car wash timer, 115–124
 - action engine icon and front panel, 117–118
 - feedback node implementation for, 123–124
 - functions of, 116
 - inputs of, 116
 - kernel sub VIs, 121–124
 - loop implementation for, 123
 - outputs of, 117
 - Reset Timer* action, 120, 122
 - state variable approach for, 115–116
 - state variable storage, 123–124
 - Timecheck* action, 118–119, 121
- Car wash 2 controller, 86–89, 125–126, 294–297, 314–315, 348–351, 454–457
 - arguments in, 294–297
 - class inheritance and, 348–351
 - CLD exam sample, 454–457
 - controls and indicators for, 456–457
 - data encapsulation for, 314–315
 - mapping sub-VIs for, 296–297
 - sequence of operation, 455–456
 - sub VIs for, 86–89
 - timers, 125–126, 314–315
- Car Wash2 Timer Kernel sub-VI, 125
- Car Wash2 Timer sub-VI, 126
- Case structures, 20–24
- Certified LabVIEW Developer (CLD) exam, *see* CLD exam
- Character Array To String sub-VI, 435
- Child classes, 317–318, 332–333
- Class diagrams, 318–320
- Class library (.lvclass), 322, 331–332
- Classes, 302–212, 316–347, 440
 - access scope, 329–331
 - child classes, 318, 332–333
 - class library features, 331–332
 - class relationships with, 316–319
 - composition, 316–347
 - creating, 303–304, 320–321
 - data encapsulation using, 302–212
 - data member access sub-VIs, 321–324
 - dynamic dispatch sub-VIs, 319–320, 324–326
 - dynamic loading, 342–345
 - factory method, 338–341
 - inheritance, 316–647
 - LabVIEW use and importance of, 347
 - objects and, 302–212
 - override VIs, 319–320, 333–338
 - patterns, 338–341
 - pros and cons of state machines using, 440
 - static dispatch sub-VIs, 319–320, 327–329
- CLD exam, 231–238, 444–468
 - ATM controller, 246–253, 463–468
 - block diagram documentation, 232–234
 - boiler controller, 449–453
 - car wash controller 2, 454–457
 - controls and indicators, 447, 453, 456–457, 461–462, 466
 - front panel documentation, 232–233
 - functionality, 235–236
 - grading criteria, 231–232
 - practicing for, 238
 - preparing for, 236–237
 - security system, 446–448
 - sprinkler controller, 458–462
 - style, 234–235
 - time management during, 237–238
 - traffic light controller, 444–445
 - VI documentation, 232–233
- Clean Up Folder.vi, 11–13, 25–27, 39–42
- Clear Error Code sub-VI, 226
- Clear Errors function, 227
- Close File function, 46–47
- Close Reference function, 158
- Cluster references, 158–161
- Cluster To Array function, 71–72
- Clusters, 15–18, 20–24, 257–260
 - arguments passed using, 257–260
 - bundling functions, 16–18
 - case structures, 20–24
 - data types, 15
 - error, 15, 20–24
 - unbundling functions, 15–17
- Coercion dots, 7–8
- Composition, 316–347. *See also* Inheritance
- Compound Arithmetic function, 83–84
- Concatenate Strings function, 45–47
- Concatenating tunnel mode, 33–34
- Conditional Acquire Semaphore sub-VI, 404–406
- Conditional Release Semaphore sub-VI, 407
- Conditional tunnel mode, 33–34
- Conditional Write To Log File sub-VI, 357–358
- Connector pane, 17–20
- Consumer loop error, 375–377
- Control references, 154–158, 161–163

- Controls, 4–8, 11–17, 142–144
 - arrays, 13–15
 - Boolean, 8, 15
 - clusters, 15–17
 - enumerated, 12
 - folder paths, 11–12
 - icons as, 5–6
 - reinitializing, 142–144
 - terminals as, 5–7
 - type definitions, 12–13
- Controls[]* property, 158–159, 161–163
- Counters, action engines, 110–111
- Create Enter Button Next State sub-VI, 252–253
- Create Log String sub-VI, 52–53
- Create State and Argument String sub-VI, 433–434
- Create User Event function, 377–379
- Cycle class, 320–347
- Cycle Factory sub-VI, 338–341
- Cycle Time Check method, 328, 330
- D**
- DAQmx Clear Task function, 215
- DAQmx Create Channel function, 214
- DAQmx Create Task function, 214
- DAQmx functions, 212–216
- DAQmx Perform Bridge Offset Nulling Calibration function, 216
- DAQmx Perform Shunt Calibration (Strain) function, 216
- DAQmx Read function, 216
- DAQmx Timing function, 215
- Data accessor methods, 418–419
- Data acquisition (DAQ), 48–49, 210–223
 - bridge circuit calibration using, 210–211
 - Build DAQ Task* state, 218
 - building the task, 214–216
 - Calibrate* state, 220–221
 - DAQmx functions, 212–216
 - functional specification, 212
 - Initialize* state, 217–218
 - LabVIEW use of, 48–49, 212
 - Multitest.vi* development, 217–223
 - Null* state, 220
 - Read strain* state, 221–222
 - read/write configuration cluster to XML file, 228–229
 - Shutdown* state, 217–219
 - state transition diagram for, 212–214
 - strain gage measurement using, 210–211
 - write strain data to output file, 229–230
- Data encapsulation, 298–312
 - action engines compared to, 298–299
 - classes, 302–312
 - data highways and, 299–302
 - methods, 306–309
 - objects, 302–312
 - static dispatches, 304–307
- Data highway, 91, 255–257, 299–302
 - arguments and, 255–257
 - data encapsulation and, 299–302
- Data member access sub-VIs, 321–324
- Data Over Limit sub-VI, 51
- Data value references, 413–418
- Dataflow, 2, 8–11, 42–43
 - execution, 8–11
 - LabVIEW programming and, 2
 - parallel operations of, 11
 - passing data by reference, 42–43
- Deadlock, 404–408
- Default values, 138–139
- Delete Data Value Reference function, 414–415
- Delete From Array function, 290–291
- Delete function, 25–27
- Deposit sub-VI, 251
- Dequeue Element function, 185, 187, 189
- Dequeue Element sub-VI, 421–423
- Dequeue–requeue sequences, 401
- Destroy User Event function, 383
- Diff Status Arrays sub-VI, 129–130
- Disable* property node, 145–146
- Dispatch methods, 304–307, 319–320, 324–329
 - dynamic, 319–320, 324–326
 - static, 304–307, 319–320, 327–329
- Documentation, 11–13, 232–234
 - block diagrams, 232–234
 - CLD exams, 231–232–324
 - front panels, 232–233
 - LabVIEW VIs for, 11–13, 232–233
- Dynamic dispatch sub-VIs, 319–320, 324–326
- Dynamic loading, 342–345
- E**
- Economy Wash button, 203–204
- Enqueue At Opposite End and Set Switch sub-VI, 424–425
- Enqueue Element At Opposite End function, 185, 188, 195
- Enqueue Element function, 185, 187, 192–193, 202
- Enqueue Element sub-VI, 422–423
- Enumerated controls, 12–13, 21–22, 25–26
- Error cluster, 15, 20–24
- Error Cluster From Error Code function, 267
- Error handling, 21–25, 35–39, 42, 385–386
 - error trapping strategy, 35–36
 - error wires, 21–23, 42
 - loops for error detection, 36–38
 - merging errors, 38–39
 - producer–consumer state machines, 385–386
 - sub-VI template for, 24–25
- Error Prone sub-VI, 36, 38–39
- Error Ring constant, 93
- Error shift register, 91, 99
- Error wires, 21–23, 42
- Errors in producer–consumer state machines, 373–377
- Event-driven state machines, 167–181, 372–385, 438–439
 - building, 175–177
 - car wash controller, 178–181
 - data events, 167
 - event structures, 168–175
 - producer–consumer architecture, 372–377
 - pros and cons of, 438–439

- Purchase Selection* control, 178–180
 - Run Test* states, 175–176
 - STOP button, 170–171, 175–176, 180–181
 - time events, 167
 - user events, 167, 377–385
 - Event structures, 168–175, 178–179, 201–206
 - anatomy of, 168–170
 - configuration window, 170–171
 - event queue, 171
 - filter events, 168–169
 - guidelines, 175
 - latched Boolean value changes, 172–173
 - locking the front panel, 174
 - loops and, 171–173
 - notify events, 168–169
 - queues and, 201–206
 - Shutdown* state and, 174
 - Wait* state and, 178–179
- F**
- Factory method, 338–341
 - Feedback nodes, 34–35, 107–110, 123–124
 - action engine, 107–110
 - car wash timer implementation using, 123–124
 - changing direction of, 109
 - initialization, 108–109
 - shift registers and, 34–35
 - File I/O, 43–47
 - File paths, 11–13, 25–27, 53–55
 - absolute, 53–55
 - enumerated controls, 12–13, 25–26
 - LabVIEW documentation from, 11–13
 - relative, 53–55
 - target and destination, 25–27
 - type definitions, 12–13
 - Filter events, 168–169
 - First in, first out (FIFO) concept, 185
 - Flat Sequence Structure, 390–393
 - Flatten To String function, 422
 - Flatten To XML function, 227–228
 - Folder paths, 11–12
 - For loops, 29, 31–33
 - Format Into String function, 52–53
 - Fract/Exp String To Number function, 241–242
 - Front panel, 3–5, 174
- G**
- General Error Handler function, 36–37
 - Generate User Event function, 381–382
 - Get Account sub-VI, 250
 - Get Cycle Object Dynamic Loading VI, 344–345
 - Get Cycle Object sub-VI, 339–340
 - Get Date/Time String function, 52–54
 - Get DBL Arguments sub-VI, 290–291
 - Get File Extension function, 27, 29
 - Get Lamp Refnums sub-VI, 161–162
 - Get LV Class Default Value function, 342–345
 - Get LV Class Path function, 351–352
 - Get Message sub-VI, 249–250
 - Get Message Templates sub-VI, 248
 - Get Option Object method, 350
 - Get Option Station sub-VI, 296
 - Get Position method, 348–349
 - Get Queue and Switch sub-VI, 418–419
 - Get Queue Status function, 409–410
 - Get Signal Name sub-VI, 351–352
 - Get Simulated Data sub-VI, 48–49
 - Get Status Of All Zones sub-VI, 73–74
 - Get Step Factory sub-VI, 367
 - Get Step Object sub-VI, 366
 - Get Switch method, 325–326
 - Get Zone Status sub-VI, 73
 - Global resources, 298–299
 - Glyph menu, 405
 - Greater? function, 80
 - Guard clause, 91, 99–100
- H**
- Has-a relationships, 318–319
 - Hello World.vi, 3–4
- I**
- Icon editor, 199–200, 405
 - Icon/connector pane, 3–4
 - Icons, 5–6, 18–19, 199–201
 - Ignition class, 361–362
 - Ignition Done Arguments sub-VI, 292
 - Ignition Done sub-VI, 79–80
 - Ignition Output sub-VI, 80
 - In Range and Coerce function, 81–82
 - Increment Index sub-VI, 244
 - Index Array function, 88–89
 - Indexing tunnel mode, 33–34
 - Indicators, 4–6
 - Inheritance, 316–647
 - car wash controller with, 345–347
 - child classes, 318, 332–333
 - class hierarchy, 337–338
 - composition and, 316–320
 - dynamic dispatch sub-VIs, 319–320, 324–326
 - factory method, 338–341
 - override VIs, 319–320, 333–338
 - patterns, 338–341
 - static dispatch sub-VIs, 319–320, 327–329
 - Initialize Array function, 71–72
 - Initialize Cycle method, 327, 330
 - Initialize Option method, 349–350
 - Initialize Status Array From Cluster Reference sub-VI, 158–160
 - Initialize Step method, 357
 - Initialize Timer method, 308–309
 - Initialize Zone Array sub-VI, 240–242
 - In-Place Element Structure, 415–417
 - Input area, 91
 - Invoke nodes, 56–57
 - Is-a relationships, 318–319
- K**
- Kernel sub-VIs, 112–114, 121–122
- L**
- LabVIEW, 1–67, 302–312, 322, 331–332, 342–345
 - arrays, 13–15
 - Boolean controls, 8, 15, 20–22

LabVIEW (*continued*)

- case structures, 20–24
 - class library (.lvclass), 322, 331–332
 - classes, 302–212, 342–345
 - clusters, 15–17
 - coercion dots, 7–8
 - controls, 4–8, 11–17
 - data acquisition, 48–49
 - dataflow, 2, 8–11, 42–43
 - enumerated controls, 12
 - error handling, 24–25, 35–39
 - feedback nodes, 34–35
 - file I/O, 43–47
 - file paths, 11–13, 25–27, 53–55
 - front panel controls and indicators, 4–7
 - functional specifications, 61–67
 - graphical nature of, 1–2
 - icons, 5–6
 - invoke nodes, 56–57
 - loops, 27–35, 36–38, 49–51
 - objects, 302–312
 - passing data by reference, 42–43
 - property nodes, 56–57
 - shift registers, 34–35
 - strings, 52–54
 - style elements, 59–60
 - sub-VIs, 17–20, 24–25, 25–27, 38–39, 51–56
 - terminals, 5–7
 - timing, 49–51
 - type definitions, 12–13
 - VIs (virtual instruments), 1–4, 8–13, 15–17, 35–42, 48–49, 57–59, 61–63
- Language, 411–428
- class extension pros and cons, 427–428
 - data accessor methods, 418–419
 - data value references, 413–418
 - LabVIEW extension of, 411–428
 - switched queue references, 412–411, 419–427
- Last Value* tunnel mode, 33
- Latched Boolean value changes, 172–173
- Light Lamps method, 326
- Light Zone Lamp sub-VI, 244
- Lobby, 90–91
- Loops, 27–35, 36–38, 49–51, 106–107, 123, 171–173, 268–273, 368–386, 390–393
- action engine implementation using, 106–107
 - argument queue states using, 268–273
 - car wash timer implementation using, 123
 - code execution by, 27–34
 - error detection using, 36–38
 - event structures and, 171–173
 - feedback nodes, 34–35
 - FOR, 29, 31–33
 - iteration limits, 51
 - latched Boolean value changes, 172–173
 - output tunnels, 33–34
 - producer–consumer state machines, 368–386
 - race conditions and, 390–393
 - shift registers, 34–35
 - stopping parallel, 369–371
 - synchronization, 390–393
 - timing, 49–51

M

- Main case structure, 92, 99–100
- Main error wire, 92
- Main loop, 92
- Make Controls Invisible sub-VI, 155–158
- Match Pattern function, 261–262
- Max & Min function, 293
- Mean function, 222–223
- Mechanical actions, 8
- Merge Errors function, 38–39
- Methods, data encapsulation using, 306–309
- Move function, 25–28
- Move Or Delete sub-VI, 25–27
- Multiply function, 7
- Multitest.vi, 61–63, 97–105, 210–223

N

- New Data Value Reference function, 414
- Next Car Wash State sub-VI, 88
- Next Signal Object sub-VI, 352
- Next Sprinkler State sub-VI, 242–243
- Next Traffic State String sub-VI, 288
- Non-square icons, 199–201
- Not Equal? function, 130, 164
- Notify events, 168–169

O

- Objects, data encapsulation using, 302–312
- Obtain Queue function, 185–186, 192
- Obtain Semaphore Reference function, 394–395
- Obtain Switched Queue Reference sub-VI, 419–420
- One Button Dialog function, 101–102
- Open/Create/Replace File function, 44–45
- Option class, 348–350
- Option Factory sub-VI, 350–351
- Option Time Check method, 350
- Output area, 92
- Output tunnels, 23, 33–34
 - case structures, 23
 - loops, 33–34
- Override VIs, 319–320, 333–338

P

- Parent class, 317
- Parse Name and Time Target.VI, 277–278
- Parse State and Argument Array sub-VI, 290
- Parse State and Arguments sub-VI, 262
- Parse State and Zone Status Array sub-VI, 433–434
- Parse State sub-VI, 276–277
- Parsing sub-VIs, 260–262, 276–278
- Passing information, 42–43, 181–189, 257–260
 - arguments, 257–260
 - clusters used for, 257–260
 - data by reference, 42–43, 188–191
 - data by value, 188–189
 - queued state machines, 188–191
 - strings used for, 260
- Patterns, 338–341, 383–385
 - classes and, 338–341
 - factory, 338–341

- producer-consumer, 383–385
 - user events and, 383–385
 - Polymorphic VIs, 164–165
 - Precision Float function, 8
 - Pre-Purge class, 360–361
 - Pre-Purge Done sub-VI, 79
 - Pre-Purge Output sub-VI, 79
 - Pre-Purge Timer class, 359–360
 - Pre-Purge Timer Done Arguments subVI, 292
 - Pre-Purge Timer Done sub-VI, 77–78
 - Pre-Purge Timer Output sub-VI, 78
 - Preview Queue Element function, 405–407
 - Producer loop error, 374–375
 - Producer-consumer state machines, 368–386, 397–404, 440
 - car wash controller, 397–404
 - error handling, 385–386
 - errors in, 373–377
 - events architecture, 372–377
 - producer-consumer architecture, 371–377
 - producer-consumer pattern with user events, 383–385
 - pros and cons of, 440
 - race conditions, 397–404
 - single-loop state machines compared to, 368–369
 - stopping parallel loops, 369–371
 - user events, 377–383, 385, 400–401
 - Value(Signalling)* property nodes, 374–377, 385
 - Property nodes, 56–57
 - Prove The Pilot class, 362–363
 - Prove The Pilot Done sub-VI, 80–81
 - Prove The Pilot Output sub-VI, 81
 - Purge class, 365–366
 - Purge Done Arguments subVI, 294
 - Purge Done sub-VI, 85
 - Purge Output sub-VI, 85–86
- Q**
- Queue reference wire, 188–189, 193, 197–199
 - Queued state machines, 184–209, 263–266, 268–273, 401, 439.
 - See also* Switched queue references
 - arguments in, 263–266, 268–273
 - building, 191–196
 - car wash controller, 197–209
 - dequeue-requeue sequences, 401
 - event structures and, 201–206
 - first in, first out (FIFO) concept, 185
 - Initialize* state, 192–193, 197–198
 - non-square icons, 199–201
 - passing data by reference, 188–191
 - pros and cons, 439
 - queue functions, 185–190
 - queue reference wire, 188–189, 193, 197–199
 - race conditions, 191, 401
 - Run Test* states, 192–195
 - Shutdown* state, 194–196, 198
 - string-based, 263–266, 268–273
 - Wait* state, 193–195
 - Quotient & Remainder function, 244–245
- R**
- Race conditions, 191, 387–410
 - blocking action, 389–390
 - deadlock, 404–408
 - dequeue-requeue sequences, 401
 - glyph menu, 405
 - guarding against, 393–397
 - living with (acceptance), 408–410
 - loop synchronization and, 390–393
 - producer-consumer car wash controller, 397–404
 - queue functions and, 191
 - semaphores and, 393–398, 402–406
 - sequences causing, 378–393
 - user events and, 400–401
 - Random Number (0–1) function, 48–49
 - Read Done method, 348–349
 - Read Done sub-VI, 323
 - Read From Spreadsheet File function, 241
 - Read From Text File function, 227–228
 - Read/write configuration cluster to XML file, 228–229
 - Recursive File List function, 40–41
 - Register For Events function, 379–381
 - Reinitialize to Default* invoke node, 143–144
 - Reinitializing controls, 142–144
 - Relative file paths, 53–55
 - Release Queue function, 185, 188, 190, 205–206
 - Release Semaphore function, 394, 396–397
 - Release Semaphore Reference function, 396, 398
 - Release Switched Queue Reference sub-VI, 419–420
 - Replace Array Subset function, 251–252
 - Replace Keys With Values sub-VI, 248
 - Reset Lockout class, 359
 - Reset Lockout Done sub-VI, 76
 - Reset Lockout Output sub-VI, 76
 - Reset Timer Highway sub-VI, 299–301
 - Reset Timer Method sub-VI, 120
 - Reset Zone Array sub-VI, 73
 - Right File Type sub-VI, 27–30
 - Round To Nearest function, 76–77
 - Run Car Wash Cycle.VI, 280–281
 - Run Car Wash Option subVI, 297
 - Run Car Wash Option Timer lvclass sub-VI, 315
 - Run class, 364–365
 - Run Cycle method, 328–330
 - Run Done Arguments subVI, 294
 - Run Done sub-VI, 83
 - Run Option method, 350
 - Run Output sub-VI, 84–85
- S**
- Search and Replace String function, 248–249
 - Search ID Array function, 27, 30
 - Security system controller, 71–74, 129–131, 152–165, 288–289, 446–448
 - arguments in, 288–289
 - array of control references for, 161–163
 - array preallocation, 71–74
 - CLD exam sample, 446–448
 - cluster references for, 158–161
 - color-changing indicators, 163
 - controls and indicators for, 447
 - definition of terms, 446–447
 - file logging, 74, 448
 - functional specifications, 152–154
 - local variables for, 154–158
 - operation of, 448

- Security system controller (*continued*)
 - polymorphic sub-VIs for, 164–165
 - states of, 71
 - sub-VIs for, 129–131
- Select function, 104–105
- Semaphores, 393–398, 402–406
- Service Zone sub-VI, 244–245
- Set Display method, 348–349
- Set File Position, 45
- Set Option Display subVI, 296
- Set Zone Colors sub-VI, 160, 162
- Setup Stop or Error sub-VI, 240
- Shift registers, 34–35, 91–92, 98–99, 106–107
 - action engine storage, 106–107
 - error, 91, 99
 - feedback nodes and, 34–35
 - loop storage, 34–35
 - state constant, 92, 98
 - state machines, 91–92, 98–99
 - uninitialized, 35, 106–107
- Shut Down Code sub-VI, 38–39
- Shutdown or Error sub-VI, 408–410
- Signal Map subVI, 288
- Simulated Data Acquisition.vi, 48–49, 53, 57–59
- Split and Queue subVI, 271–273
- Split String and Enqueue sub-VI, 423–424
- Spreadsheet String To Array function, 269–270
- Sprinkler controller, 458–462
 - CLD exam sample, 458–462
 - configuration file, 462
 - controls and indicators for, 461–462
 - sequence of operation, 459–461
- Square function, 9–11
- Start class, 363–364
- Start Done Arguments subVI, 292
- Start Done sub-VI, 81
- Start Output sub-VI, 83
- State and Time Target subVI, 295–296
- State constant, 92, 98
- State machines, 90–105, 132–150, 167–181, 184–209, 254–273, 368–386, 438–441
 - arguments and, 254–273, 439–440
 - building, 97–105
 - car wash controller applications, 132–150, 178–181, 197–209
 - classic, 90–105, 438
 - classes and, 440
 - coupling, 440–441
 - disadvantages to using strings, 266–268
 - elements of, 90–92
 - even-driven, 167–181, 438–439
 - front panel, 98
 - guard clause, 91, 99–100
 - improving string-based, 268–273
 - Initialize state, 101–102, 142–144, 192–193
 - main case structure, 92, 99–100
 - placements of elements in, 96
 - producer–consumer, 368–386, 440
 - pros and cons of, 438–440
 - queued, 184–209, 439
 - queued-string, 263–266
 - Run Test states, 101–103, 175–176, 192–195
 - shift registers, 91–92, 98–99
 - Shutdown state, 100–101, 140–141, 151, 194–196
 - skeleton layout, 98–99
 - state enumerated constant, 97–98, 134–135
 - state identification, 97
 - string-based, 263–273
 - structure of, 92–95
 - Wait state, 102–105, 145–146, 178–179, 193–195
- State shift register, 92, 98
- State transition diagrams, 62–63, 66–67
- State variable method, 111–114, 115–116, 123–134
 - action engine storage, 111–114
 - kernel sub-VIs for, 112–114, 123–124
 - car wash timer storage, 115–116, 123–124
- Static dispatch, 304–307, 319–320, 327–329
 - data encapsulation and, 304–307
 - inheritance and, 319–320, 327–329
 - sub-VIs, 319–320, 327–329
- Step class, 355–357
- Step Done method, 357
- Step Output method, 357
- Stop or Error sub-VI, 140, 197–201
- Stopwatch Kernel sub-VI, 112–114
- Stopwatch Timer action engine, 127
- Stopwatch Timer Kernel sub-VI, 127
- Strain gage measurement, 210–211. *See also* Data acquisition
- String Changed Kernel sub-VI, 313–314
- String Changed.vi, 126
- String Length function, 423
- String Subset function, 436
- String To Array subVI, 269–271
- String To Character Array sub-VI, 435
- String To Path sub-VI, 342–344
- String-based state machines, 263–273
 - disadvantages to, 266–268
 - improving, 268–273
 - queued, 263–266
 - string-based VIs, 263–266
- Strings, 13–15, 52–54, 260–260
 - arguments passes using, 260
 - arrays, 13–15
 - controls and indicators, 13–15
 - parsing sub-VIs, 260–262
 - simulated log of data as, 52–54
- Sub-VIs, 17–20, 24–25, 25–30, 38–39, 51–56, 112–114, 260–262, 276–278, 304–307, 319–329
 - action engines, 112–114
 - appending files, 54–56
 - arguments and, 260–262, 276–278
 - connector pane, 17–20
 - creating, 17–20
 - data member access, 321–324
 - dynamic dispatch, 319–320, 324–326
 - error handling, 24–25, 38–39
 - file paths, 25–27, 53–54
 - inheritance and, 319–329, 333–338
 - kernel, 112–114
 - Move Or Delete, 25–27
 - parsing, 260–262, 276–278
 - Right File Type, 27–30
 - static dispatch, 304–307, 319–320, 327–329

Switch When Pressed buttons, 203–204
 Switched Enqueue At Opposite End sub-VI, 425–426
 Switched Queue class, 412–419

T

Terminals, 5–7
 Testing LabVIEW programs, 61–62
 Time and Proximity Check sub-VI, 119–120
 Time Check method, 306–307
 Timer class, 303–312
 Timing, loops and, 49–51
 To More Generic Class function, 154–155, 157
 To More Specific Class function, 154, 156
 Top-down design, 132
 Traffic light controller, 68–70, 127–129, 288, 313,
 351–353, 444–445
 auto reset timer, 313
 class inheritance and, 351–353
 mapping functions, 288
 operation of, 444–445
 states of, 68–70
 subVIs for, 127–129
 symmetry of, 69–70
 Type Cast function, 87
 Type definitions, 12–13

U

Unbundle By Name function, 15–17, 148
 Unbundle function, 15–16
 Underbody Wash class, 332–336
 Unflatten From String function, 423
 Unflatten From XML function, 229
 Uninitialized shift registers, 35, 106–107
 Unregister For Events function, 382–383
 Update Account File sub-VI, 251
 Update Log Action Engine sub-VI, 225
 Update Log Slowly sub-VI, 429
 Update log sub-VI, 130–131, 225
 User events, 167, 377–385, 400–401
 creating, 377–379
 event-driven state machines, 167
 generating, 381–382
 producer–consumer car wash controller using, 400–401
 producer–consumer pattern using, 383–385
 producer–consumer state machines, 377–386
 race conditions, 400–401
 registering, 379–381
 unregistering, 382–383
 use of, 377–383

V

Value property node, 144
 Value(Signalling) property nodes, 374–377, 385
 Verify Account sub-VI, 250–251
 Virtual instruments, *see* Sub-VIs; VIs
 VIs, 1–4, 8–13, 15–17, 35–42, 48–49, 57–59, 61–63, 319–320,
 333–341
 block diagram, 3–4
 cluster elements, 15–17
 data acquisition, 48–49
 dataflow execution for, 8–11
 development of, 57–59
 documentation and, 11–13
 enumerated controls, 12
 error handling, 35–39, 42
 factory method for, 338–341
 front panel, 3–5
 graphical nature of, 1–2
 icon/connector pane, 3–4
 inheritance and, 319–320, 333–341
 invoke nodes, 56–57
 LabVIEW program as, 1–4
 override, 319–320, 333–338
 property nodes, 56–57
 state transition diagrams for, 62–63
 targeted extensions, 39–41
 testing LabVIEW programs, 61–62
 type definitions, 12–13
 Visible property, 155–158

W

Wait (ms) function, 429–430
 Wait state, 102–105, 134, 145–146, 178–179, 193–195,
 car wash controller, 134, 145–146, 178–179
 queued state machines, 193–195
 state machines, 102–105, 145–146, 178–179, 193–195
 Wait Until Next ms Multiple function, 49–51
 While loops, 29–31
 Wires, 6–7, 19–20
 Withdrawal sub-VI, 251
 Write Done method, 348–349
 Write Done sub-VI, 324
 Write strain data to output file, 229–230
 Write To Boiler Log File sub-VI, 126
 Write To Log File Data Highway sub-VI, 313–314
 Write To Log File sub-VI, 74
 Write To Spreadsheet function, 251, 253
 Write To Text File function, 45–46

