



TI-*nspire*[™]

Lua Scripting API Reference Guide

Important Information

Except as otherwise expressly stated in the License that accompanies a program, Texas Instruments makes no warranty, either express or implied, including but not limited to any implied warranties of merchantability and fitness for a particular purpose, regarding any programs or book materials and makes such materials available solely on an "as-is" basis. In no event shall Texas Instruments be liable to anyone for special, collateral, incidental, or consequential damages in connection with or arising out of the purchase or use of these materials, and the sole and exclusive liability of Texas Instruments, regardless of the form of action, shall not exceed the amount set forth in the license for the program. Moreover, Texas Instruments shall not be liable for any claim of any kind whatsoever against the use of these materials by any other party.

© 2023 Texas Instruments Incorporated

The TI-Nspire™ software uses Lua as scripting environment. For copyright and license information, see <http://www.lua.org/license.html>.

The TI-Nspire™ software uses Chipmunk Physics version 5.3.4 as simulation environment. For license information, see <http://chipmunk-physics.net/release/Chipmunk-5.x/Chipmunk-5.3.4-Docs/>.

Microsoft® and Windows® are registered trademarks of Microsoft Corporation in the United States and/or other countries.

Mac OS®, iPad® and OS X® are registered trademarks of Apple Inc.

Unicode® is a registered trademark of Unicode, Inc. in the United States and other countries.

Bluetooth® word mark and logos are registered trademark owned by Bluetooth SIG, Inc.

Actual products may vary slightly from provided images.

Contents

| | |
|---|----------|
| Chapter 1 Standard Libraries | 1 |
| 1.1 Basic Library Functions | 1 |
| 1.1.1 Coroutine Sub-Library | 1 |
| 1.2 Module Library | 1 |
| 1.3 String Library | 1 |
| 1.4 Table Library | 2 |
| 1.5 Math Library | 2 |
| 1.6 Unimplemented Libraries and Functions | 2 |
| Chapter 2 Touch Library | 3 |
| 2.1 Overview | 3 |
| 2.1.1 On-Screen Keyboard and Screen Resize Behavior | 3 |
| 2.1.2 Event Handling | 3 |
| 2.2 Library Functions | 4 |
| 2.2.1 ppi | 4 |
| 2.2.2 xppi | 4 |
| 2.2.3 yppi | 4 |
| 2.2.4 enabled | 4 |
| 2.2.5 isKeyboardAvailable | 5 |
| 2.2.6 isKeyboardVisible | 5 |
| 2.2.7 showKeyboard | 5 |
| Chapter 3 2D Editor Library | 6 |
| 3.1 newRichText | 6 |
| 3.2 createChemBox | 6 |
| 3.3 createMathBox | 7 |
| 3.4 getExpression | 7 |
| 3.5 getExpressionSelection | 7 |
| 3.6 getText | 7 |
| 3.7 hasFocus | 8 |
| 3.8 isVisible | 8 |
| 3.9 move | 8 |
| 3.10 registerFilter | 8 |
| 3.11 resize | 9 |
| 3.12 setBorder | 9 |
| 3.13 setBorderColor | 9 |
| 3.14 setColorable | 9 |
| 3.15 setDisable2DinRT | 9 |
| 3.16 setExpression | 9 |
| 3.17 setFocus | 10 |
| 3.18 setFontSize | 10 |
| 3.19 setMainFont | 10 |
| 3.20 setReadOnly | 11 |
| 3.21 setSelectable | 11 |
| 3.22 setSizeChangeListener | 11 |

| | | |
|------------------------------------|-----------------------|-----------|
| 3.23 | setText | 11 |
| 3.24 | setTextChangeListener | 12 |
| 3.25 | setTextColor | 12 |
| 3.26 | setVisible | 12 |
| 3.27 | setWordWrapWidth | 12 |
| Chapter 4 Class Library | | 13 |
| 4.1 | class | 13 |
| Chapter 5 Clipboard Library | | 14 |
| 5.1 | addText | 14 |
| 5.2 | getText | 14 |
| Chapter 6 Cursor Library | | 15 |
| 6.1 | set | 15 |
| 6.2 | hide | 17 |
| 6.3 | show | 17 |
| Chapter 7 Document Library | | 18 |
| 7.1 | markChanged | 18 |
| Chapter 8 Event Handling | | 19 |
| 8.1 | activate | 20 |
| 8.2 | arrowDown | 20 |
| 8.3 | arrowKey | 20 |
| 8.4 | arrowLeft | 21 |
| 8.5 | arrowRight | 21 |
| 8.6 | arrowUp | 21 |
| 8.7 | charIn | 21 |
| 8.8 | backspaceKey | 21 |
| 8.9 | backTabKey | 21 |
| 8.10 | clearKey | 22 |
| 8.11 | construction | 22 |
| 8.12 | contextMenu | 22 |
| 8.13 | copy | 22 |
| 8.14 | create | 22 |
| 8.15 | createMathBox | 22 |
| 8.16 | cut | 23 |
| 8.17 | deactivate | 23 |
| 8.18 | deleteKey | 23 |
| 8.19 | destroy | 23 |
| 8.20 | enterKey | 23 |
| 8.21 | escapeKey | 23 |
| 8.22 | getFocus | 23 |
| 8.23 | getSymbolList | 24 |
| 8.24 | grabDown | 24 |
| 8.25 | grabUp | 24 |
| 8.26 | help | 24 |
| 8.27 | keyboardDown | 25 |

| | |
|--|-----------|
| 8.28 keyboardUp | 25 |
| 8.29 loseFocus | 25 |
| 8.30 mouseDown | 25 |
| 8.31 mouseMove | 25 |
| 8.32 mouseUp | 25 |
| 8.33 paint | 26 |
| 8.34 paste | 26 |
| 8.35 propertiesChanged | 26 |
| 8.36 resize | 26 |
| 8.37 restore | 26 |
| 8.38 returnKey | 27 |
| 8.39 rightMouseDown | 27 |
| 8.40 rightMouseUp | 27 |
| 8.41 save | 27 |
| 8.42 tabKey | 28 |
| 8.43 timer | 28 |
| 8.44 varChange | 28 |
| Chapter 9 Graphics Library | 29 |
| 9.1 clipRect | 29 |
| 9.2 drawArc | 29 |
| 9.3 drawImage | 29 |
| 9.4 drawLine | 30 |
| 9.5 drawPolyLine | 30 |
| 9.6 drawRect | 30 |
| 9.7 drawString | 30 |
| 9.8 fillArc | 30 |
| 9.9 fillPolygon | 30 |
| 9.10 fillRect | 31 |
| 9.11 getStringHeight | 31 |
| 9.12 getStringWidth | 31 |
| 9.13 setColorRGB | 31 |
| 9.14 setFont | 31 |
| 9.15 setPen | 31 |
| Chapter 10 Image Library | 32 |
| 10.1 new | 32 |
| 10.2 copy | 32 |
| 10.3 height | 32 |
| 10.4 rotate | 32 |
| 10.5 width | 33 |
| Chapter 11 Locale Library | 34 |
| 11.1 name | 34 |
| Chapter 12 Math Library Extension | 35 |
| 12.1 eval | 35 |
| 12.2 evalStr | 36 |
| 12.3 getEvalSettings | 36 |

| | |
|--|-----------|
| 12.4 setEvalSettings | 37 |
| Chapter 13 Module Library | 39 |
| Chapter 14 Platform Library | 40 |
| 14.1 apiLevel | 40 |
| 14.2 hw | 40 |
| 14.3 isColorDisplay | 40 |
| 14.4 isDeviceModeRendering | 41 |
| 14.5 isTabletModeRendering | 41 |
| 14.6 registerErrorHandler | 41 |
| 14.7 window | 41 |
| 14.7.1 height and width | 41 |
| 14.7.2 invalidate | 41 |
| 14.7.3 setBackgroundColor | 42 |
| 14.7.4 setFocus | 42 |
| 14.7.5 getScrollHeight | 42 |
| 14.7.6 setScrollHeight | 42 |
| 14.7.7 displayInvalidatedRectangles | 42 |
| 14.8 withGC | 43 |
| 14.9 getDeviceID | 43 |
| Chapter 15 String Library Extension | 44 |
| 15.1 split | 44 |
| 15.2 uchar | 44 |
| 15.3 usub | 44 |
| 15.4 pack | 44 |
| 15.5 unpack | 45 |
| Chapter 16 Timer Library | 46 |
| 16.1 getMilliSecCounter | 46 |
| 16.2 start | 46 |
| 16.3 stop | 46 |
| Chapter 17 Tool Palette Library | 47 |
| 17.1 register | 47 |
| 17.2 enable | 47 |
| 17.3 enableCut | 48 |
| 17.4 enableCopy | 48 |
| 17.5 enablePaste | 48 |
| Chapter 18 Variable Library | 49 |
| 18.1 list | 49 |
| 18.2 makeNumericList | 49 |
| 18.3 monitor | 49 |
| 18.4 recall | 49 |
| 18.5 recallAt | 50 |
| 18.6 recallStr | 50 |
| 18.7 store | 50 |

| | |
|---|-----------|
| 18.8 storeAt | 50 |
| 18.9 unmonitor | 50 |
| Chapter 19 Physics Library | 51 |
| 19.1 Miscellaneous routines | 51 |
| 19.1.1 INFINITY | 51 |
| 19.1.2 momentForBox | 51 |
| 19.1.3 momentForCircle | 51 |
| 19.1.4 momentForPoly | 52 |
| 19.1.5 momentForSegment | 52 |
| 19.2 Vectors | 52 |
| 19.2.1 Vect | 52 |
| 19.2.2 add | 53 |
| 19.2.3 clamp | 53 |
| 19.2.4 cross | 53 |
| 19.2.5 dist | 54 |
| 19.2.6 distsq | 54 |
| 19.2.7 dot | 54 |
| 19.2.8 eql | 54 |
| 19.2.9 length | 55 |
| 19.2.10 lengthsq | 55 |
| 19.2.11 lerp | 55 |
| 19.2.12 lerpconst | 55 |
| 19.2.13 mult | 56 |
| 19.2.14 near | 56 |
| 19.2.15 neg | 56 |
| 19.2.16 normalize | 56 |
| 19.2.17 normalizeSafe | 57 |
| 19.2.18 perp | 57 |
| 19.2.19 project | 57 |
| 19.2.20 rotate | 57 |
| 19.2.21 rperp | 57 |
| 19.2.22 setx | 58 |
| 19.2.23 sety | 58 |
| 19.2.24 slerp | 58 |
| 19.2.25 slerpconst | 59 |
| 19.2.26 sub | 59 |
| 19.2.27 toangle | 59 |
| 19.2.28 unrotate | 59 |
| 19.2.29 x | 60 |
| 19.2.30 y | 60 |
| 19.3 Bounding Boxes | 60 |
| 19.3.1 BB | 60 |
| 19.3.2 b | 60 |
| 19.3.3 clampVect | 61 |
| 19.3.4 containsBB | 61 |
| 19.3.5 containsVect | 61 |
| 19.3.6 expand | 61 |
| 19.3.7 intersects | 62 |

| | |
|-------------------------|----|
| 19.3.8 l | 62 |
| 19.3.9 merge | 62 |
| 19.3.10 setb | 62 |
| 19.3.11 r | 63 |
| 19.3.12 setl | 63 |
| 19.3.13 setr | 63 |
| 19.3.14 sett | 63 |
| 19.3.15 t | 64 |
| 19.3.16 wrapVect | 64 |
| 19.4 Bodies | 64 |
| 19.4.1 Body | 64 |
| 19.4.2 activate | 64 |
| 19.4.3 angle | 65 |
| 19.4.4 angVel | 65 |
| 19.4.5 applyForce | 65 |
| 19.4.6 applyImpulse | 65 |
| 19.4.7 data | 66 |
| 19.4.8 force | 66 |
| 19.4.9 isRogue | 66 |
| 19.4.10 isSleeping | 66 |
| 19.4.11 local2World | 67 |
| 19.4.12 kineticEnergy | 67 |
| 19.4.13 mass | 67 |
| 19.4.14 moment | 67 |
| 19.4.15 pos | 67 |
| 19.4.16 resetForces | 68 |
| 19.4.17 rot | 68 |
| 19.4.18 setAngle | 68 |
| 19.4.19 setAngVel | 68 |
| 19.4.20 setData | 69 |
| 19.4.21 setForce | 69 |
| 19.4.22 setMass | 69 |
| 19.4.23 setMoment | 70 |
| 19.4.24 setPos | 70 |
| 19.4.25 setPositionFunc | 70 |
| 19.4.26 setTorque | 70 |
| 19.4.27 setVel | 71 |
| 19.4.28 setVelocityFunc | 71 |
| 19.4.29 setVLimit | 71 |
| 19.4.30 setWLimit | 72 |
| 19.4.31 sleep | 72 |
| 19.4.32 sleepWithGroup | 72 |
| 19.4.33 torque | 73 |
| 19.4.34 updatePosition | 73 |
| 19.4.35 updateVelocity | 73 |
| 19.4.36 vel | 74 |
| 19.4.37 vLimit | 74 |
| 19.4.38 wLimit | 74 |

| | |
|----------------------------------|----|
| 19.4.39 world2Local | 74 |
| 19.5 Shapes | 74 |
| 19.5.1 BB | 75 |
| 19.5.2 body | 75 |
| 19.5.3 collisionType | 75 |
| 19.5.4 data | 75 |
| 19.5.5 friction | 75 |
| 19.5.6 group | 76 |
| 19.5.7 layers | 76 |
| 19.5.8 rawBB | 76 |
| 19.5.9 restitution | 76 |
| 19.5.10 sensor | 77 |
| 19.5.11 setCollisionType | 77 |
| 19.5.12 setData | 77 |
| 19.5.13 setFriction | 77 |
| 19.5.14 setGroup | 78 |
| 19.5.15 setLayers | 78 |
| 19.5.16 setRestitution | 78 |
| 19.5.17 setSensor | 78 |
| 19.5.18 setSurfaceV | 79 |
| 19.5.19 surfaceV | 79 |
| 19.6 Circle Shapes | 79 |
| 19.6.1 CircleShape | 79 |
| 19.6.2 offset | 80 |
| 19.6.3 radius | 80 |
| 19.7 Polygon Shapes | 80 |
| 19.7.1 PolyShape | 80 |
| 19.7.2 numVerts | 80 |
| 19.7.3 points | 81 |
| 19.7.4 vert | 81 |
| 19.8 Segment Shapes | 81 |
| 19.8.1 SegmentShape | 81 |
| 19.8.2 a | 82 |
| 19.8.3 b | 82 |
| 19.8.4 normal | 82 |
| 19.8.5 radius | 82 |
| 19.9 Spaces | 82 |
| 19.9.1 Space | 83 |
| 19.9.2 addBody | 83 |
| 19.9.3 addConstraint | 83 |
| 19.9.4 addCollisionHandler | 83 |
| 19.9.5 addPostStepCallback | 84 |
| 19.9.6 addShape | 84 |
| 19.9.7 addStaticShape | 85 |
| 19.9.8 damping | 85 |
| 19.9.9 data | 85 |
| 19.9.10 elasticIterations | 85 |
| 19.9.11 gravity | 85 |

| | | |
|----------|------------------------------|----|
| 19.9.12 | idleSpeedThreshold | 86 |
| 19.9.13 | iterations | 86 |
| 19.9.14 | rehashShape | 86 |
| 19.9.15 | rehashStatic | 86 |
| 19.9.16 | removeBody | 86 |
| 19.9.17 | removeConstraint | 87 |
| 19.9.18 | removeShape | 87 |
| 19.9.19 | removeStaticShape | 87 |
| 19.9.20 | resizeActiveHash | 87 |
| 19.9.21 | resizeStaticHash | 88 |
| 19.9.22 | setDamping | 88 |
| 19.9.23 | setData | 88 |
| 19.9.24 | setElasticIterations | 89 |
| 19.9.25 | setGravity | 89 |
| 19.9.26 | setIdleSpeedThreshold | 89 |
| 19.9.27 | setIterations | 89 |
| 19.9.28 | setSleepTimeThreshold | 90 |
| 19.9.29 | sleepTimeThreshold | 90 |
| 19.9.30 | step | 91 |
| 19.10 | Constraints | 91 |
| 19.10.1 | Damped Rotary Spring | 91 |
| 19.10.2 | Damped Spring | 92 |
| 19.10.3 | Gear Joint | 92 |
| 19.10.4 | Groove Joint | 93 |
| 19.10.5 | Pin Joint | 93 |
| 19.10.6 | Pivot Joint | 94 |
| 19.10.7 | Ratchet Joint | 94 |
| 19.10.8 | Rotary Limit Joint | 94 |
| 19.10.9 | Simple Motor | 95 |
| 19.10.10 | Slide Joints | 95 |
| 19.11 | Arbiters and Collision Pairs | 96 |
| 19.11.1 | # | 96 |
| 19.11.2 | a | 96 |
| 19.11.3 | b | 96 |
| 19.11.4 | bodies | 96 |
| 19.11.5 | depth | 97 |
| 19.11.6 | elasticity | 97 |
| 19.11.7 | friction | 97 |
| 19.11.8 | impulse | 97 |
| 19.11.9 | isFirstContact | 97 |
| 19.11.10 | normal | 98 |
| 19.11.11 | point | 98 |
| 19.11.12 | setElasticity | 98 |
| 19.11.13 | setFriction | 98 |
| 19.11.14 | shapes | 99 |
| 19.11.15 | totalImpulse | 99 |
| 19.11.16 | totalImpulseWithFriction | 99 |
| 19.12 | Shape Queries | 99 |

| | |
|---|------------|
| 19.12.1 pointQuery | 99 |
| 19.12.2 segmentQuery | 100 |
| 19.13 Space Queries | 100 |
| 19.13.1 pointQuery | 100 |
| 19.13.2 pointQueryFirst | 100 |
| 19.13.3 segmentQuery | 101 |
| 19.13.4 segmentQueryFirst | 101 |
| 19.14 SegmentQueryInfo | 102 |
| 19.14.1 hitDist | 102 |
| 19.14.2 hitPoint | 102 |
| Chapter 20 Bluetooth® Smart Library | 103 |
| 20.1 Bluetooth® LE | 103 |
| 20.1.1 addStateListener | 103 |
| 20.1.2 removeStateListener | 104 |
| 20.1.3 pack | 104 |
| 20.1.4 unpack | 104 |
| 20.1.5 Format Specifier for pack and unpack | 104 |
| 20.2 Bluetooth® LE Central | 105 |
| 20.2.1 startScanning | 105 |
| 20.2.2 stopScanning | 106 |
| 20.2.3 isScanning | 107 |
| 20.3 Peripheral Class | 107 |
| 20.3.1 getName | 107 |
| 20.3.2 getState | 107 |
| 20.3.3 connect | 108 |
| 20.3.4 disconnect | 109 |
| 20.3.5 discoverServices | 109 |
| 20.3.6 getServices | 110 |
| 20.4 Service Class | 110 |
| 20.4.1 getUUID | 110 |
| 20.4.2 discoverCharacteristics | 110 |
| 20.4.3 getCharacteristics | 111 |
| 20.5 Characteristic Class | 111 |
| 20.5.1 getUUID | 111 |
| 20.5.2 setValueUpdateListener | 111 |
| 20.5.3 setWriteCompleteListener | 112 |
| 20.5.4 read | 112 |
| 20.5.5 setNotify | 113 |
| 20.5.6 getValue | 113 |
| 20.5.7 write | 113 |
| Chapter 21 Asynchronous Serial Interface | 114 |
| 21.1 require 'asi' | 114 |
| 21.2 addStateListener | 114 |
| 21.3 removeStateListener | 115 |
| 21.4 isScanning | 115 |
| 21.5 startScanning | 115 |
| 21.6 stopScanning | 116 |

| | |
|---|------------|
| 21.7 Port Class | 116 |
| 21.7.1 getName | 116 |
| 21.7.2 getIdentifier | 116 |
| 21.7.3 getState | 116 |
| 21.7.4 setBaudRate | 117 |
| 21.7.5 connect | 117 |
| 21.7.6 disconnect | 118 |
| 21.7.7 setWriteListener | 118 |
| 21.7.8 write | 119 |
| 21.7.9 setReadListener | 119 |
| 21.7.10 setReadTimeout | 120 |
| 21.7.11 read | 120 |
| 21.7.12 getValue | 120 |
| Appendix A Script Compatibility | 121 |
| A.1 Backward and Forward Compatibility | 121 |
| A.1.1 Document Compatibility | 121 |
| A.1.2 Scripting Compatibility | 121 |
| A.2 Creating Scripts for a Future Software Release | 122 |
| A.3 Platform Compatibility | 122 |
| Appendix B Deprecated API Functions and API Behavior | 123 |
| B.1 Image Library | 123 |
| B.2 Platform Library | 123 |
| B.2.1 gc | 123 |
| B.3 Platform Library | 124 |
| B.3.1 drawString Vertical Alignment | 124 |
| B.4 Requested API Level | 124 |
| Index | 125 |

List of Tables

| | |
|--|-----|
| Table 2.1: Gesture to event handler mapping | 3 |
| Table 3.1: 2D editor markup language | 6 |
| Table 20.1: Format specifier for pack and unpack | 105 |
| Table A.1: Mapping between API level and TI-Nspire™ software version | 122 |
| Table A.2: Overview about platform incompatibilities | 122 |

Listings

| | |
|---|----|
| Listing 3.1: Default Values of a new 2D Rich Text Editor | 6 |
| Listing 3.2: Example 1 for <code>getExpressionSelection()</code> | 7 |
| Listing 3.3: Example 2 for <code>getExpressionSelection()</code> | 7 |
| Listing 3.4: Example for <code>D2Editor:registerFilter()</code> | 8 |
| Listing 3.5: Example 1 for <code>D2Editor:setExpression</code> | 10 |
| Listing 3.6: Example 2 for <code>D2Editor:setExpression</code> | 10 |
| Listing 4.1: Class Library Example | 13 |
| Listing 8.1: Event Handler Example | 19 |
| Listing 8.2: Example for <code>getSymbolList</code> | 24 |
| Listing 12.1: Converting a Lua Number to a String to be Used in <code>math.eval()</code> (E Notation) | 35 |
| Listing 12.2: <code>math.evalStr()</code> Returning Result in E Notation | 36 |
| Listing 12.3: <code>math.evalStr()</code> Returning Negative Numbers | 36 |
| Listing 12.4: TI-Nspire™ Software Default Settings Returned by <code>getEvalSettings</code> | 37 |
| Listing 12.5: Calling <code>math.setEvalSettings()</code> using a table with names | 37 |
| Listing 12.6: Calling <code>math.setEvalSettings()</code> using a table with ordinal number | 37 |
| Listing 12.7: Calling <code>math.setEvalSettings()</code> using a table with combined names and numbers | 37 |
| Listing 14.1: Example of Using <code>withGC()</code> to get the Pixel Length and Height of a String | 43 |
| Listing 15.1: Examples for <code>string.usub()</code> | 44 |
| Listing 15.2: Example Showing the use of <code>string.pack()</code> | 45 |
| Listing 15.3: Concatenation of Multiple calls to <code>string.pack()</code> | 45 |
| Listing 15.4: Example Showing the use of <code>string.unpack()</code> | 45 |
| Listing 15.5: Splitting Unpacking into Multiple calls to <code>string.unpack()</code> | 45 |
| Listing 17.1: Registering a Tool Palette | 47 |
| Listing 18.1: Example for Accessing a Matrix via the Variable Library | 49 |
| Listing 19.1: Spherical Linear Interpolation Example | 58 |
| Listing 19.2: Example for <code>physics.Body:setVelocityFunc()</code> | 71 |
| Listing 19.3: The Form of the Callback Table for <code>physics.Space:addCollisionHandler()</code> | 83 |

| | |
|---|-----|
| Listing A.1: Authoring for a Future Software Release for the Example of Touch | 122 |
| Listing B.1: Use of the static GC in <code>platform.apiLevel = '1.0'</code> | 123 |

List of Figures

| | |
|---|-----|
| Figure 8.1: Open Document Sequence Chart | 20 |
| Figure 20.1: Bluetooth® LE Scanning Procedure | 107 |

Chapter 1

Standard Libraries

The TI-Nspire™ software integrates most Lua standard libraries that come with the Lua distribution. This chapter provides an overview about the supported Lua library functions as well as restrictions to these functions.

See the ([Lua 5.1 Reference Manual](#)) for definitions and details of the standard functions.

1.1 Basic Library Functions

For further details, please follow this link to the “[Basic Functions](#)” section in the Lua 5.1 Reference Manual.

| | | | | | |
|--------------------|-------------------|-------------------------|--------|---------|--------------|
| assert | collectgarbage | error | _G | getfenv | getmetatable |
| ipairs | load ¹ | loadstring ¹ | next | pairs | pcall |
| print ² | rawequal | rawget | rawset | select | setfenv |
| setmetatable | tonumber | tostring | type | unpack | _VERSION |
| xpcall | | | | | |

1.1.1 Coroutine Sub-Library

For further details, please follow this link to the “[Coroutine Manipulation](#)” section in the Lua 5.1 Reference Manual. The following functions are defined inside the **coroutine** table. Heavy use of coroutines might be difficult to debug inside the TI-Nspire™ Editor.

| | | | | | |
|--------|--------|---------|--------|------|-------|
| create | resume | running | status | wrap | yeild |
|--------|--------|---------|--------|------|-------|

1
2

1.2 Module Library

The implementation of this module is very limited. Please consult the [Module Library chapter](#) for more details.

1.3 String Library

For further details, please follow this link to the “[String Manipulation](#)” section in the Lua 5.1 Reference Manual.

String routines `lower` and `upper` are not tailored to the current locale. The conversion of strings to **upper** and **lower** case letters operates only on the 26 letters of the Latin alphabet. This restriction also applies to the alphabetic matching patterns (`%a`, `%l`, `%u`, and `%w`) employed by the **find**, **gmatch**, and **match** functions.

| | | | | | | | |
|-------|-------|------|---------|--------|--------|------|-----|
| byte | char | dump | find | format | gmatch | gsub | len |
| lower | match | rep | reverse | sub | upper | | |

¹Please be cautious with the use of **load** and **loadstring**. Lua source code loaded by the use of these functions is not supported in the TI-Nspire™ Editor. This source code cannot be debugged and error messages resulting from functions loaded using **load** and **loadstring** might cause confusing results.

²The output from the **print** function is directed into the console of the TI-Nspire™ Editor only. On any platform where the TI-Nspire™ Editor is not included calls to the **print** function are ignored.

1.4 Table Library

For further details, please follow this link to the [“Table Manipulation”](#) section in the Lua 5.1 Reference Manual.

concat insert maxn remove sort

1.5 Math Library

For further details, please follow this link to the [“Mathematical Functions”](#) section in the Lua 5.1 Reference Manual. The following functions are defined inside the **math** table. Infinite and undefined results will convert to the appropriate TI-Nspire™ representations and cooperate with the TI-Nspire™ math extensions. The reverse conversion of string representation (infinite and undefined) to numerical representation is not supported.

abs acos asin atan atan2 ceil cos cosh
deg exp floor fmod frexp huge ldexp log
log10 max min modf pi pow rad random
randomseed sin sinh sqrt tan tanh

1.6 Unimplemented Libraries and Functions

The following standard Lua libraries are not available in the TI-Nspire™ software:

file io os debug

The following standard functions and standard table entries are not available in the TI-Nspire™ software:

dofile loadfile module package.cpath package.loadlib
package.path package.seeall

Chapter 2

Touch Library

The touch library is added to the TI-Nspire™ platform with **platform.apiLevel = '2.2'**. It is visible on all platforms but may ignore calls to its functions if the platform running the script does not support touch.

The touch library offers a low-level interface, which enables script authors to develop scripts that run on all platforms equally. It also places the effort on the script writer to design and test the script for all different platforms if platform compatibility is desired.

2.1 Overview

The following will give an overview about system features and behavior that script authors need to be aware of to write successful scripts for touch platforms and scripts working well across all TI-Nspire™ platforms.

2.1.1 On-Screen Keyboard and Screen Resize Behavior

The TI-Nspire™ software features two keyboards — ABC and Function keyboard. The user can switch between both keyboards. The default keyboard for the scripting environment is the ABC keyboard.

There are different keyboard modes that might be supported on each touch platform — docked, undocked, and split keyboard. In any mode, no resize event will be sent to the script. If the keyboard is docked, the TI-Nspire™ platform will allow the user to pan the screen allowing access to content behind the keyboard - see [setScrollHeight\(\)](#) for controlling scrolling by the script while a docked keyboard is onscreen. The new [on.keyboardUp\(\)](#) event handler supports the script with the overlapping height of the on-screen keyboard.

Touch platforms usually support undocked and split on-screen keyboards to be panned; therefore, panning of the script is not needed.

2.1.2 Event Handling

All event handling is described [Chapter 8](#). There is no change for touch platforms in **Introduced in platform.apiLevel = '2.2'** except for two new handlers, on-screen keyboard up and down detection --- see [on.keyboardUp\(keyboardOverlapHeight\)](#) and [on.keyboardDown\(\)](#) event handler.

Please see Table 2.1 for the mapping between touch gestures and the existing event handlers.

Table 2.1: Gesture to event handler mapping

| Gesture | "on" handler | Comment |
|------------|--|--|
| Single Tap | on.mouseDown() on.mouseUp() | It should be noted that the gesture recognizer adds a small delay between lifting the finger from the screen and sending the mouseUp event. |
| Double Tap | on.mouseDown() on.mouseUp() on.mouseDown() on.mouseUp() | Likewise due to the gesture recognizer the first mouseUp is received after the second tap is complete. The following down and up are send immediately. |
| Pan | on.mouseDown() on.mouseMove() 's ... on.mouseUp() | Same behavior as on a desktop platform when pressing the mouse button, dragging the mouse and releasing the mouse button again. When running on a non-touch platform, on.mouseMove() can be received while the mouse button is not pressed. |

| | | |
|-----------------|---|--|
| Long Press Move | on.mouseDown() on.mouseMove() 's on.mouseUp() | Same behavior as pan. There is no differentiation possible from the script |
| Other Gestures | on.mouseDown() [on.mouseMove() 's] on.mouseUp() | Will reliably generate a on.mouseDown() and on.mouseUp() event. One or multiple on.mouseMove() might be send. Multi-finger gestures will report coordinates below or between the fingers. |

Note: The behavior of the mapping described in [Table 2.1](#) is slightly different for mouse handler registered with [D2Editor:registerFilter\(\)](#). In case of single and double tap will the first **on.mouseDown()** event be received after the gesture is fully recognized and the finger lifted up from the screen. Similar is true for the pan and long press gesture. The **on.mouseDown()** event is send when either the finger starts moving or the stays without moving for a particular time.

Another important aspect related to event handling is the return value of an event handler. The main use case in **platform.apiLevel = '2.0'** for event handler return values has been lter event handler registered for a 2D Editor - see [D2Editor:registerFilter\(\)](#). Every event handler may return a boolean to indicate if the event has been handled (**true**) or ignored (**false**). If an event handler does not return explicitly a value, the value will default to true. In the context of touch and on-screen keyboard, the return value of **mouseDown** while the keyboard is up plays an important role and can disturb the user experience when used incorrectly. While the keyboard is up, the user can pan the screen to see content behind the keyboard. If **mouseDown** returns true, or has no explicit return statement, the user will be prevented from panning the screen.

2.2 Library Functions

2.2.1 ppi

```
touch.ppi()
```

Returns pixels per inches along the diagonal of the screen. This function is useful to determine the touch target size of touchable objects on the screen.

Introduced in **platform.apiLevel = '2.2'**

2.2.2 xppi

```
touch.xppi()
```

Returns pixels per inches along the x-axis of the screen. This function is useful to determine the touch target size of touchable objects on the screen.

Introduced in **platform.apiLevel = '2.2'**

2.2.3 yppi

```
touch.yppi()
```

Returns pixels per inches along the y-axis of the screen. This function is useful to determine the touch target size of touchable objects on the screen.

Introduced in **platform.apiLevel = '2.2'**

2.2.4 enabled

```
touch.enabled()
```

Returns **true** if the platform supports touch, otherwise **false**. If touch is supported, it is recommended to use the ppi values to calculate touch target sizes.

Introduced in **platform.apiLevel = '2.2'**

2.2.5 isKeyboardAvailable

```
touch.isKeyboardAvailable()
```

Returns **true** if an on-screen keyboard is available on the platform, otherwise **false**.

Introduced in `platform.apiLevel = '2.2'`

2.2.6 isKeyboardVisible

```
touch.isKeyboardVisible()
```

Returns true if any keyboard is visible (docked, undocked, and split keyboards).

Introduced in `platform.apiLevel = '2.2'`

2.2.7 showKeyboard

```
touch.showKeyboard(boolean)
```

Causes the docked ABC keyboard to appear on the screen if no keyboard is currently visible. Default is **true**.

Introduced in `platform.apiLevel = '2.2'`

Chapter 3

2D Editor Library

The Lua 2D editor bindings enable 2D rich text editors to be created and manipulated within the TI-Nspire™ product. 2D rich text editors are created using `newRichText()`. Script authors should be aware that rich text editors may embed annotations in proprietary markup language. Such markup could be embedded from the script by calling `createMathBox()` or `createChemBox()`. Users of the script application may also be able to copy and paste text with other markup information from other TI-Nspire™ applications like Notes. Some information about the markup language used inside the 2D editor is shown in [Table 3.1](#).

Table 3.1: 2D editor markup language

| Description | Markup | Comment |
|--------------------------------|---|--|
| Math Box | <code>"\0el {...}"</code> | Contains a 2D formula. |
| Evaluated Math Box expressions | <code>"\0el {...}>"</code> <code>\0el {...}"</code> | A pair of Math Boxes — formula and evaluated result. |
| Chem Box | <code>"\0chem {...}"</code> | Describes a chemical formula. |
| Other | <code>"\1 ...\"</code> | It is not recommended to utilize this type in scripts as the used markup may change in future releases. But it is recommended that scripts will gracefully handle this type of markup without Lua error. |

3.1 newRichText

```
D2Editor.newRichText()
```

Creates and returns a new 2D rich text editor. Default values are illustrated in [Listing 3.4](#).

Note

The program must [resize](#) the 2D editor before the text editor widget is painted the first time.

Listing 3.1: Default Values of a new 2D Rich Text Editor

```
editor:move(0, 0)
:setBorder(0)
:setBorderColor(0x000000)
:setColorable(false)
:setDisable2DinRT(false)
:setFontSize(<default system size>)
:setMainFont(<default system font>)
:setReadOnly(false)
:setSelectable(true)
:setTextColor(0x000000)
:setVisible(true)
```

Introduced in `platform.apiLevel = '1.0'`

3.2 createChemBox

```
D2Editor:createChemBox()
```

Inserts a Chem Box in the current cursor position of the editor. Returns the text editor object.

Introduced in `platform.apiLevel = '2.0'`

3.3 createMathBox

```
D2Editor:createMathBox()
```

Inserts a Math Box (Expression Box) in the current cursor position of the editor. Returns the text editor object.

Introduced in `platform.apiLevel = '2.0'`

3.4 getExpression

```
D2Editor:getExpression()
```

Returns the contents of the text editor as a UTF-8 encoded string.

Introduced in `platform.apiLevel = '2.0'`

3.5 getExpressionSelection

```
D2Editor:getExpressionSelection()
```

Returns three values: the contents of the text editor as a UTF-8 encoded string, the cursor position as an integer, and the selection start as an integer.

Usage

Cursor and selection positions are the borders between characters, not the position of the characters. The following code snippets serve as examples.

Listing 3.2: Example 1 for `getExpressionSelection()`

```
str = 'This is a test string to see it working.'
d2e, error = D2Editor.newRichText():resize(100, 100)
result, error = d2e.setText(str, 16, 28)
str, pos, sel, error =
d2e:getExpressionSelection()

-- The getExpressionSelection() are results are:
str = 'This is a test string to see it working.'
pos = 16 -- (right before the 's' in "string")
sel = 28 -- (between the two e's in "see")
```

Listing 3.3: Example 2 for `getExpressionSelection()`

```
str = 'This is a test string to see it working.'
d2e, error = D2Editor.newRichText():resize(100, 100)
result, error = d2e.setText(str, 28, 16)
str, pos, sel, error = d2e:getExpressionSelection()

-- The getExpressionSelection() are results are:
str = 'This is a test string to see it working.'
pos = 28 -- (between the two e's in "see")
sel = 16 -- (right before the 's' in "string")
```

Introduced in `platform.apiLevel = '2.0'`

3.6 getText

```
D2Editor:getText()
```

Returns the contents of the text editor as a UTF-8 encoded string.

Introduced in `platform.apiLevel = '1.0'`

3.7 hasFocus

```
D2Editor:hasFocus()
```

Returns true if the editor has focus; otherwise returns false.

Introduced in `platform.apiLevel = '2.0'`

3.8 isVisible

```
D2Editor:isVisible()
```

Returns true if the editor is visible; otherwise returns false.

Introduced in `platform.apiLevel = '2.0'`

3.9 move

```
D2Editor:move(x, y)
```

Sets the parent-relative location of the upper-left corner of the text editor. Both `x` and `y` must be between -32767 and 32767.

Returns the text editor object.

Introduced in `platform.apiLevel = '1.0'`

3.10 registerFilter

```
D2Editor:registerFilter(handlerTable)
```

This routine registers a table of handler functions that can filter events before they are sent to the 2D editor widget, or unregisters if nil is passed.

Returns the text editor object.

The **handlerTable** is a table of event handler functions. Any event described in the section on Event Handling can be filtered by a function in the handler table.

In the example code below, if the user presses Tab in the text editor `ed`, the `tabKey` filter function moves the focus to text editor `ed2`. Events `charIn` and `arrowKey` simply report which key was pressed and then allow the event to pass on through to the text editor.

Listing 3.4: Example for `D2Editor:registerFilter()`

```
-- Create an editor
ed = D2Editor.newRichText()

-- Register filters for events
ed:registerFilter {
  tabKey = function()
    ed2:setFocus()
    return true
  end,
  charIn = function(ch)

    print(ch)
    return false
  end,
  arrowKey = function(key)
```



```
print(key)
return false
end
}
```

Introduced in `platform.apiLevel = '2.0'`

3.11 resize

```
D2Editor:resize(width, height)
```

Changes the width and height of the text editor. Both **width** and **height** must be > 0 and < 32768 .

Returns the text editor object.

Introduced in `platform.apiLevel = '1.0'`

3.12 setBorder

```
D2Editor:setBorder(thickness)
```

Sets the editor's border thickness. The thickness value must be between 0 and 10. Returns the text editor object.

Introduced in `platform.apiLevel = '2.0'`

3.13 setBorderColor

```
D2Editor:setBorderColor(color)
```

Sets the editor's border color. The color value must be between 0 and 16777215 (0x000000 and 0xFFFFFFFF).

Returns the text editor object.

Introduced in `platform.apiLevel = '2.0'`

3.14 setColorable

```
D2Editor:setColorable(boolean)
```

Makes the expression colorable or uncolorable. Returns the text editor object.

Introduced in `platform.apiLevel = '2.0'`

3.15 setDisable2DinRT

```
D2Editor:setDisable2DinRT(boolean)
```

Turns off 2D layout of math input to the text box. Returns the text editor object.

Introduced in `platform.apiLevel = '2.0'`

3.16 setExpression

```
D2Editor:setExpression(text[, cursor[, selection]])
```

Sets the text content of the text editor. The cursor position is set to 1 (beginning of text), -1 (end of text), or a value from 1 to the text length plus 1. Text can be selected by specifying a selection index that indicates the end of the selection. If the

selection = -1, no text is selected. If the cursor < -1 or **selection** < -1, an error is returned. If unspecified, both the cursor and the selection start default to -1. Returns the text editor object.

Note

All backslashes sent to the editor must be doubled. This is in addition to the standard escape rule for special characters. As a result, the string required to get the editor to show **home\stuff\work** is "home\\stuff\\work".

Usage

Cursor and selection positions are the borders between characters, not the character positions. The following code snippet highlights the characters "string to se" and places the cursor before the 's' in "string".

Listing 3.5: Example 1 for D2Editor:setExpression

```
str = 'This is a test string to see it working.'
d2e, error = D2Editor.newRichText():resize(100, 100)
result, error = d2e:setExpression(str, 16, 28)
```

D2Editor output: This is a test |string to see it working.

The following code snippet highlights the characters "string to se" and places the cursor before the second 'e' in "see".

Listing 3.6: Example 2 for D2Editor:setExpression

```
str = 'This is a test string to see it working.'
d2e, error = D2Editor.newRichText():resize(100, 100)
result, error = d2e:setExpression(str, 28, 16)
```

D2Editor output: This is a test string to se|e it working.

Introduced in **platform.apiLevel = '2.0'**

3.17 setFocus

```
D2Editor:setFocus(boolean)
```

Sets the user input focus on the editor if true (the default). This is usually called from the on.getFocus event handler.

Returns the text editor object.

Introduced in **platform.apiLevel = '2.0'**

3.18 setFontSize

```
D2Editor:setFontSize(size)
```

Sets the text font size in the editor. The point size is restricted on the TI-Nspire™ family of handhelds. Choose one of these sizes: 7, 9, 10, 11, 12, 16, or 24. Any font size supported by Windows® or Mac OS® can be used in the desktop software.

Returns the text editor object.

Introduced in **platform.apiLevel = '2.0'**

3.19 setMainFont

```
D2Editor:setMainFont(family, style)
D2Editor:setMainFont(family, style [, fontSize]) -- API Level 2.3
```

Sets the main font family ("serif" or "sansserif") and style ("r", "b", "i", "bi"). The new font size parameter introduced in **platform.apiLevel = '2.3'** is optional.

| Style | Description |
|-------|-----------------|
| r | Regular |
| b | Bold |
| i | Italic |
| bi | Bold and Italic |

Returns the text editor object.

Introduced in platform.apiLevel = '2.0'

Extended in platform.apiLevel = '2.3'

3.20 setReadOnly

```
D2Editor: setReadOnly(boolean)
```

Makes the text editor content modifiable (false) or unmodifiable (true) by the user. If a Boolean value is not specified, defaults to true.

Returns the text editor object.

Introduced in platform.apiLevel = '1.0'

3.21 setSelectable

```
D2Editor: setSelectable(boolean)
```

Makes the text editor content selectable (true) or unselectable (false) by the user. If a Boolean value is not specified, defaults to true.

Returns the text editor object.

Introduced in platform.apiLevel = '1.0'

3.22 setSizeChangeListener

```
D2Editor: setSizeChangeListener(function(editor, w, h))
```

Sets the callback function for when the editor contents exceed the current editor size, when the contents fit on fewer lines, or when the contents fit on a single line of smaller width. This function can then resize the editor appropriately. The callback function should be a void function. It will be passed into the following parameters:

| Parameter | Description |
|-----------|--|
| editor | Editor in which the expression changed size. |
| w | Optimal widget width to t the expression. |
| h | Optimal widget height to t the expression. |

Returns the text editor object.

Info

To remove the listener, call `D2Editor:setSizeChangeListener(nil)`

Introduced in platform.apiLevel = '2.0'

3.23 setText

```
D2Editor: setText(text[, cursor[, selection]])
```

See `setExpression()` for details.

Returns the text editor object.

Introduced in `platform.apiLevel = '1.0'`

3.24 `setTextChangeListener`

```
D2Editor:setTextChangeListener(function(editor))
```

Sets the callback function for when the text expression changes. This function will be passed into the editor object. This allows for processing text input as it occurs.

Returns the text editor object.

Info

To remove the listener, call `D2Editor:setTextChangeListener(nil)`

Introduced in `platform.apiLevel = '2.0'`

3.25 `setTextColor`

```
D2Editor:setTextColor(color)
```

Sets the editor text color. The color value must be between 0 and 16777215 (0x000000 and 0xFFFFFFFF).

Returns the text editor object.

Introduced in `platform.apiLevel = '2.0'`

3.26 `setVisible`

```
D2Editor:setVisible(boolean)
```

Sets the visibility of the text editor. Returns the text editor object.

Introduced in `platform.apiLevel = '2.0'`

3.27 `setWordWrapWidth`

```
D2Editor:setWordWrapWidth(width)
```

Sets the rich text editor word-wrapping width in pixels. Ignored if the editor is in 2D mode. To indicate widget width, sets to 0. To disable wrapping, sets to < 0. The width must be -32767 to 32767.

Note

When word wrapping is disabled, that is the width is < 0, and ellipses are added to cut words, the negative value of the width specifies the margin from the right of the widget before ellipses are used.

Returns the text editor object.

Introduced in `platform.apiLevel = '2.0'`

Chapter 4

Class Library

The class library implements basic object-oriented class definitions.

4.1 class

```
class([parent_class])
```

Returns a new class. If a parent class is specified, the new class inherits the methods of the parent class.

Listing 4.1: Class Library Example

```
Widget = class()  
function Widget:init() ... end  
  
Button = class(Widget)  
function Button:init() ... end
```

With these definitions, when the script calls `Button()`, a new `Button` is created. The `Button:init()` function is called to initialize the button, and the newly minted `Button` object is returned as the function result of the call.

Class `Button` in this example inherits all the methods and class variables defined in class `Widget`.

Class `Button` can override any methods of its parent class.

Introduced in platform.apiLevel = '1.0'

Chapter 5

Clipboard Library

5.1 addText

```
clipboard.addText(string)
```

This routine adds the contents of **string** to the Clipboard as plain text, MIME type "text/plain".

Introduced in `platform.apiLevel = '1.0'`

5.2 getText

```
clipboard.getText()
```

This routine returns the contents of the Clipboard as a string of plain text. If the Clipboard does not contain any text (MIME type "text/plain"), this routine returns nil.

Introduced in `platform.apiLevel = '1.0'`

Chapter 6

Cursor Library

This cursor library controls the appearance of the mouse pointer. The visibility of the cursor can only be controlled on a handheld.












Touch platforms do not support the concept of a mouse cursor, therefore any call to this library will be ignored on touch platforms.





















A good practice is to request the expected cursor appearance within [on.activate\(\)](#). Calls on the cursor library are ignored while deactivated (after [on.deactivate\(\)](#) is received).

6.1 set

```
cursor.set(cursorname)
```

Parameter **cursorname** is a string that contains the name of the cursor shape to use for the mouse pointer. It can be one of the following strings:

| Cursor icon | Cursor name | SmartClick? | Usage Notes |
|---|-------------------------|-------------|--|
|  | "default", "pointer" | N | Used to show the position of the cursor. |
|  | "hand pointer" | Y | Used to indicate that the underlying object can be selected or activated with a click. |
|  | "crosshair" | Y | Used for fine control of a selection – often used to indicate the bounds of a rectangular region selection. |
|  | "hand open" | Y | Indicates that the underlying object can be grabbed. |
|  | "hand closed" | N | Indicates that the underlying object has been grabbed. |
|  | "drag grab" | N | Typically used to indicate that a pan type of operation is in progress. |
|  | "rotation" | Y | Indicates that the underlying object can be rotated and is also used to indicate that the rotation operation is ongoing. |
|  | "translation" | Y | Indicates that the underlying object can be translated and is also used to indicate that the translation operation is ongoing. |
|  | "dilation" | Y | Indicates that the underlying object can be dilated and is also used to indicate that the dilation operation is ongoing. |
|  | "diag resize" | Y | Indicates that a grab at this location will initiate a diagonal resize operation. A hand closed should be used during the resize operation. |
|  | "resize column" | Y | Indicates that a grab at this location will initiate a resize column or horizontal resize operation. A hand closed should be used during the resize operation. |

| Cursor icon | Cursor name | SmartClick? | Usage Notes |
|---|------------------|-------------|---|
|  | "resize row" | Y | Indicates that a grab at this location will initiate a resize row or vertical resize operation. A hand closed should be used during the resize operation. |
|  | "zoom in" | Y | Indicates that a click will result in a zoom in. |
|  | "zoom out" | Y | Indicates that a click will result in a zoom out. |
|  | "zoom box" | Y | Indicates that a click will initiate a zoom box operation. |
|  | "pencil" | Y | Indicates that a click will result in the next step in some kind of construction or drawing operation. |
|  | "hide" | Y | Indicates that a click will hide the underlying object. |
|  | "show" | N | Indicates that a click will show the underlying object. |
|  | "clear" | N | Indicates that a click will delete the underlying object. |
|  | "animate" | Y | Indicates that a click will animate the underlying object. |
|  | "interrogate" | N | |
|  | "text" | N | Indicates that the underlying object is text and that a click will initiate an edit of that text. |
|  | "link select" | N | Used as part of a linking operation to indicate that the underlying object is available to be linked to. |
|  | "unavailable" | N | The current operation is unavailable for the underlying object. |
|  | "wait busy" | N | Used by the system to indicate that the handheld is busy doing work. |
|  | "writing" | N | Deprecated. |
|  | "hollow pointer" | Y | Used when you need a SmartClick mouse pointer. Indicates that the object under the cursor can be selected. |
|  | "arrow" | N | No standard usage. |
|  | "dotted arrow" | Y | No standard usage. |
|  | "excel plus" | Y | Used to indicate the location suitable for initiation of a drag to fill operation. |
|  | "mod label" | Y | Deprecated. |

What is SmartClick?

- SmartClick improves the user experience by making it easier to center click on an object.

How does it work?

- When the application displays a SmartClick cursor it is an indication that the next user operation will likely be a center click. Therefore, when a SmartClick cursor is displayed, any click (button press) on the touchpad will result in a center click event being sent to the active application. In this context the user does not have to worry about positioning the finger in the very center of the touchpad – even clicking on the edge of the touchpad will result in a center click.

Introduced in platform.apiLevel = '1.0'

6.2 hide

```
cursor.hide()
```

This routine hides the mouse pointer on a handheld.

Note: Calls to this routine are ignored if not executed on a handheld.

Introduced in platform.apiLevel = '1.0'

6.3 show

```
cursor.show()
```

This routine makes the mouse pointer visible on a handheld.

Note: Calls to this routine are ignored if not executed on a handheld.

Introduced in platform.apiLevel = '1.0'

Chapter 7

Document Library

7.1 markChanged

```
document.markChanged().
```

This routine marks the current document as changed. The user is prompted to save the TI-Nspire™ document before closing.

Introduced in `platform.apiLevel = '1.0'`

Chapter 8

Event Handling

Script applications respond to external stimuli by implementing event handlers. All the event handlers are grouped in the “on” module.

Example

For example, the application script implements `on.paint(gc)` to be notified when it is time to redraw its window. `on.paint` is passed a graphics context that it can use to call drawing routines on its window.

Listing 8.1: Event Handler Example

```
function on.paint(gc)
  gc.drawLine(...)
  :
end
```

Simplified Open Document Scenario

There are many scenarios that can be discussed in detail. All specifics about the behavior of particular events are discussed as part of the event description.

However the open document scenario will be discussed here to visualize options and the order of the events received. In reality there might be many more events involved due to painting the script in different contexts (page sorter).

Based on the API level of the script, either the routine **on.construction** or **on.create** is called. The idea of **on.construction** is first to separate the definition of variables classes (done in main) from constructing the app; secondly to separate the layout from construction (**on.resize()**). The main issue of **on.create()** is the missing ability to invalidate ([subsection 14.7.2](#)) and the mix of creation and layout. The latter might be impacted in some cases due to the missing capability of requesting to invalidate the screen. Another option is the call to **on.restore()**, which is only done when the script is saved and provided a state table (see [section 8.41](#)). The following figure shows this visually.

In addition, it is also important to understand that a script may not have a size before the **on.resize()** event is received. Calling width or height of the platform window ([subsection 14.7.1](#)) before **on.resize()** may return 0.

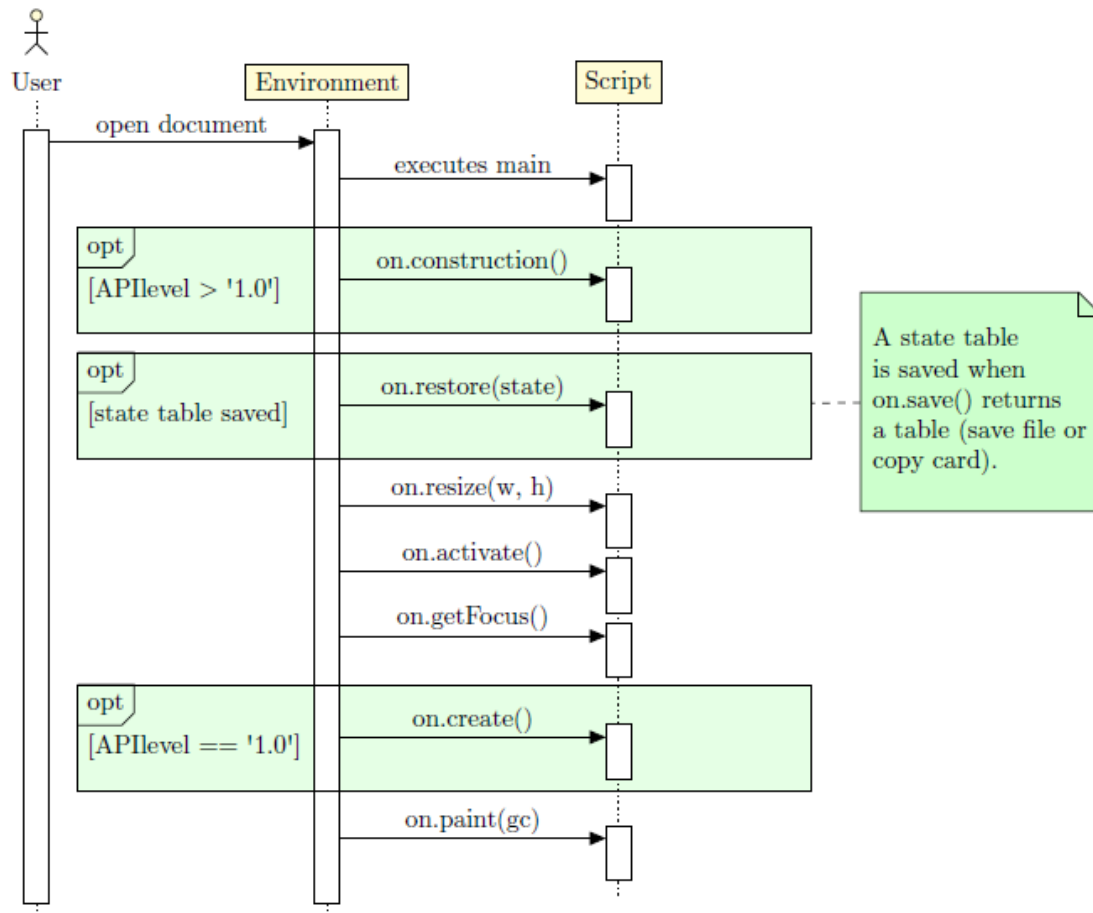


Figure 8.1: Open Document Sequence Chart

8.1 activate

```
on.activate()
```

This routine is called when the script application is activated. The dimensions of the drawing window cannot be initialized at this point, so it is not a good place to create and position graphical elements if they depend on the window size.

Introduced in platform.apiLevel = '1.0'

8.2 arrowDown

```
on.arrowDown()
```

This routine is called when the user presses the down arrow key.

Introduced in platform.apiLevel = '1.0'

8.3 arrowKey

```
on.arrowKey(key)
```

This routine is called when the user presses an arrow key. The **key** parameter may be “up”, “down”, “left”, or “right”. This routine is not called if the script implements a specific arrow key handler (`on.arrowDown` for instance) for the particular arrow key type.

Introduced in `platform.apiLevel = '1.0'`

8.4 arrowLeft

```
on.arrowLeft()
```

This routine is called when the user presses the left arrow key.

Introduced in `platform.apiLevel = '1.0'`

8.5 arrowRight

```
on.arrowRight()
```

This routine is called when the user presses the right arrow key.

Introduced in `platform.apiLevel = '1.0'`

8.6 arrowUp

```
on.arrowUp()
```

This routine is called when the user presses the up arrow key.

Introduced in `platform.apiLevel = '1.0'`

8.7 charIn

```
on.charIn(char)
```

This routine is called when the user types a letter, digit, or other character. The parameter `char` is normally a one-byte string, but because it can contain a UTF-8 encoded character, it may be two or more bytes long. It may also contain the letters of a function name from one of the short-cut keys, such as “sin” from the trig menu.

Introduced in `platform.apiLevel = '1.0'`

8.8 backspaceKey

```
on.backspaceKey()
```

This routine is called when the user presses Backspace on the desktop keyboard or the Del key on the handheld keypad.

Introduced in `platform.apiLevel = '1.0'`

8.9 backTabKey

```
on.backtabKey()
```

This routine is called when the user presses Shift + Tab.

Introduced in `platform.apiLevel = '1.0'`

8.10 clearKey

```
on.clearKey()
```

This routine is called when the user presses the Clear key on the handheld keypad.

Introduced in `platform.apiLevel = '1.0'`

8.11 construction

```
on.construction()
```

This function is guaranteed to fire first before any other event.

Introduced in `platform.apiLevel = '2.0'`

8.12 contextMenu

```
on.contextMenu()
```

This routine is called when the user presses the context Menu key.

Introduced in `platform.apiLevel = '1.0'`

8.13 copy

```
on.copy()
```

This routine is called when the user selects the Copy command either from a menu or by pressing Ctrl + C.

Note

Copy is enabled/disabled by `toolpalette.enableCopy(enable)`.

Introduced in `platform.apiLevel = '1.0'`

8.14 create

```
on.create()
```

For scripts with `platform.apiLevel ≥ '2.0'`, use `on.construction()` instead.

This routine is called after `resize` and before `paint` when the script application is created. The window size and graphics context are valid at this point. The `on.paint` event handler will be called soon after this routine finishes.

It is best to think of this function as an initialization method that fires once automatically.

Introduced in `platform.apiLevel = '2.0'`

8.15 createMathBox

```
on.createMathBox()
```

This routine is called when the user presses Ctrl + M or inserts a Math Box (Expression Box). The implementation for this callback should call the corresponding 2D editor to insert a math box if applicable.

Introduced in `platform.apiLevel = '2.0'`

8.16 cut

```
on.cut()
```

This routine is called when the user selects the Cut command either from a menu or by pressing Ctrl + X.

Note

Cut is enabled/disabled by `toolpalette.enableCut(enable)`.

Introduced in `platform.apiLevel = '1.0'`

8.17 deactivate

```
on.deactivate()
```

This routine is called when the script is deactivated. This happens when the user moves the focus to another page or to another application on the same page.

Introduced in `platform.apiLevel = '1.0'`

8.18 deleteKey

```
on.deleteKey()
```

This routine is called when the user presses the Delete key on the desktop keyboard. This is not the Del key on the handheld keypad.

Introduced in `platform.apiLevel = '1.0'`

8.19 destroy

```
on.destroy()
```

This routine is called just before the script application is deleted. A script app is deleted when it is cut to the Clipboard and when the document that contains it is closed.

Introduced in `platform.apiLevel = '1.0'`

8.20 enterKey

```
on.enterKey()
```

This routine is called when the user presses the Enter key.

Introduced in `platform.apiLevel = '1.0'`

8.21 escapeKey

```
on.escapeKey()
```

This routine is called when the user presses the Esc key.

Introduced in `platform.apiLevel = '1.0'`

8.22 getFocus

```
on.getFocus()
```

This routine is called when the script receives user input focus.

Introduced in `platform.apiLevel = '2.0'`

8.23 getSymbolList

```
on.getSymbolList()
```

This routine is called when the script app symbol list is being serialized to the Clipboard. The script app returns a list of names of variables in the symbol table that it needs to copy with it to the Clipboard. The TI-Nspire™ software copies the names and values of the variables along with the script app. Then when the user pastes the script app in another problem, the system adds the companion variables to the problem symbol table.

As a remark, `on.getSymbolList()` is called when a page containing a script app is copied, but not when a problem containing a script app is copied. This is because the entire symbol table is copied when the problem is copied.

For example, the following function indicates that it needs variable `f1` to be copied with the app to the Clipboard. The value of `f1` will be added to the symbol table when it is pasted into another problem even in another TNS document.

Listing 8.2: Example for `getSymbolList`

```
function on.getSymbolList()  
    return {"f1"}  
end
```

Introduced in `platform.apiLevel = '2.0'`

8.24 grabDown

```
on.grabDown(x, y)
```

This routine is called in these situations:

- When the user presses and holds the Select key on a handheld
- When the user presses Ctrl + Select on a handheld
- When the user presses the middle mouse button over an active card on the desktop

`x` & `y` are always zero

The `grabDown` and `grabUp` events prevent the generation of a `mouseUp` event in all cases. They will be preceded by a `mouseDown` event when generated by pressing and holding the Select key on a device.

Introduced in `platform.apiLevel = '1.0'`

8.25 grabUp

```
on.grabUp(x, y)
```

This routine is called when the mouse button is released while `grab` is in effect.

`x` & `y` are always zero

Introduced in `platform.apiLevel = '1.0'`

8.26 help

```
on.help()
```

This routine is called when the user presses the Help key. On the desktop, the Help key is Ctrl + Shift + ?. On the handheld, it is Ctrl + ?, the control key over the Trig button.

Introduced in `platform.apiLevel = '1.0'`

8.27 keyboardDown

```
on.keyboardDown()
```

This routine is only called on touch-enabled platforms. It indicates that any docked keyboard has been hidden by the user or the script by calling `touch.keyboardShow(false)`

Introduced in `platform.apiLevel = '2.2'`

8.28 keyboardUp

```
on.keyboardUp(keyboardOverlapHeight)
```

This routine is only called on touch-enabled platforms. It indicates that a docked keyboard opened on the screen and may overlap the script content. The parameter `keyboardOverlapHeight` provides the height if an overlap occurs. The return value of this routine controls if user scrolling should be enabled via the pan gesture. If returning `true` user scrolling is enabled otherwise (`false`) scrolling needs to be implemented by the script in terms of alternating the content drawn by `on.paint()`. The default value is `true`.

Introduced in `platform.apiLevel = '2.2'`

8.29 loseFocus

```
on.loseFocus()
```

This routine is called when the script loses user input focus.

Introduced in `platform.apiLevel = '2.0'`

8.30 mouseDown

```
on.mouseDown(x, y)
```

This routine is called when the user clicks the mouse. `x` and `y` are in window-relative pixel coordinates.

Note

This event will NOT be generated if the right mouse button is being held down.

Introduced in `platform.apiLevel = '1.0'`

8.31 mouseMove

```
on.mouseMove(x, y)
```

This routine is called when the user moves the mouse pointer. The mouse button does not have to be pressed to receive these events.

Introduced in `platform.apiLevel = '1.0'`

8.32 mouseUp

```
on.mouseUp(x, y)
```

This routine is called when the user releases the mouse button.

Note

This event will NOT be generated in the following cases:

- The preceding `mouseDown` event was blocked because the right mouse button was down already.

- The preceding mouseDown event was not handled.

Introduced in `platform.apiLevel = '1.0'`

8.33 paint

```
on.paint(gc, x, y, width, height)
```

This routine is called when the script application window needs to be painted. The `gc` graphics context is used in the script code to draw on the window. Additionally it provides the rectangle to be painted. Usually the provided rectangle will match the one provided to `invalidate()` ([subsection 14.7.2](#)) however the system might merge multiple consecutive calls to `invalidate` in one single paint rectangle. This merge optimization varies based on platform and screen resolution.

Introduced in `platform.apiLevel = '1.0'`

Extended in `platform.apiLevel = '2.4'`

8.34 paste

```
on.paste()
```

This routine is called when the user selects the Paste command either from a menu or by pressing Ctrl + V.

Note

Paste is enabled/disabled by `toolpalette.enablePaste(enable)`.

Introduced in `platform.apiLevel = '1.0'`

8.35 propertiesChanged

```
on.propertiesChanged(propertiesTable)
```

This routine is called on property changes. Currently all property changes are propagated as unsolicited events. Future API level may require registration for certain properties.

| Property | Data Format | Description |
|----------|--|---|
| 'locale' | language code, same as locale.name() | Provides the language selected if changed by the user (only supported on the family of handhelds) |

Introduced in `platform.apiLevel = '2.2'`

8.36 resize

```
on.resize(width, height)
```

This routine is called when the script application window changes size. This is a good place to initialize (or relayout) graphical objects based on the window size.

Introduced in `platform.apiLevel = '1.0'`

8.37 restore

```
on.restore(state)
```

This routine is called when the script application is restored from its saved state in a document or when the app is pasted into a document. It is called only if the state was saved with the application when it was previously copied to the Clipboard or saved in a document. See the `on.save` handler.

The parameter **state** is the table that the `on.save` event handler returned.

Warning

Functionality that is not available during initialization is also not usable within `on.restore`. Among the functions that cannot be called are `math.eval` and `platform.isDeviceModeRendering`.

Introduced in `platform.apiLevel = '1.0'`

8.38 returnKey

```
on.returnKey()
```

This routine is called when the user presses the Return key on the handheld keypad.

Introduced in `platform.apiLevel = '1.0'`

8.39 rightMouseDown

```
on.rightMouseDown(x, y)
```

This routine is called when the user clicks the right mouse button. **x** and **y** are in window-relative pixel coordinates.

Note

Only available on the desktop version.

Mouse events are exclusive, which means that a `rightMouseDown` event cannot occur while the left mouse button is being held down and vice versa.

Introduced in `platform.apiLevel = '1.0'`

8.40 rightMouseUp

```
on.rightMouseUp(x, y)
```

This routine is called when the user releases the right mouse button.

Note

Only available on the desktop version.

This event will NOT be generated in the following cases:

- The preceding `rightMouseDown` event was blocked because the left mouse button was already down.
- The preceding `rightMouseDown` event was not handled.

Introduced in `platform.apiLevel = '1.0'`

8.41 save

```
on.save()
```

This routine is called when the script app is saved to the document or copied to the Clipboard. The script should return a table of data needed to properly restore when the `on.restore` event handler is called.

Introduced in `platform.apiLevel = '1.0'`

8.42 tabKey

```
on.tabKey()
```

This routine is called when the user presses the Tab key.

Introduced in `platform.apiLevel = '1.0'`

8.43 timer

```
on.timer()
```

If the script application implements `on.timer`, the system calls this routine each time the timer ticks.

Introduced in `platform.apiLevel = '1.0'`

8.44 varChange

```
on.varChange(varlist)
```

This routine is called when a monitored variable is changed by another application. The **varlist** is a list of variable names whose values were changed. This handler must return a value to indicate if it accepts the new value(s) or vetoes the change.

Valid return values are:

| Value | Brief Description | Comment |
|-------|-------------------|---|
| 0 | Success | The script application accepts the change. |
| -1 | Veto range | The new value is unsatisfactory because it is outside the acceptable range, which is too low or too high. |
| -2 | Veto type | The new value is unsatisfactory because its type cannot be used by the script application. |
| -3 | Veto existence | Another application deleted the variable, and this application needs it. |

Introduced in `platform.apiLevel = '1.0'`

Chapter 9

Graphics Library

A graphics context is a module that has a handle to the script's graphics output window and a library of graphics routines that are used to draw on the window. A graphics context is supplied to the script **on.paint** event handler each time the window needs to be redrawn.

The graphics context employs a pixel-based coordinate system with the origin in the upper left corner of the drawing window.

9.1 clipRect

```
gc:clipRect(op[, x, [y, [width, [height]]]])
```

Sets the clipping rectangle for subsequent graphics operations.

Parameter **op** takes one of the strings "set," "reset," "intersect," or "null".

| Operation | Description |
|-----------|---|
| reset | Sets the clipping rectangle to include the entire window. The remaining parameters are ignored and can be left out. |
| set | Sets the clipping rectangle to the x, y coordinates with the specified width and height. Unspecified parameters default to the system window location and size. |
| intersect | Removed in platform.apiLevel = '2.0'. |
| null | Sets the clipping rectangle to empty. All subsequent graphics commands are ignored. |

Typically the "set" operation is called before drawing, such as for a text string. It is important to call the "reset" operation after drawing the last clipped graphic so that you do not leave a lingering clipping rectangle as a side effect.

Introduced in platform.apiLevel = '1.0'

9.2 drawArc

```
gc:drawArc(x, y, width, height, startAngle, arcAngle)
```

Draws an arc in the rectangle with upper left corner (x,y) and pixel width and height. Both the width and height must be ≥ 0 . The arc is drawn beginning at startAngle degrees and continues for endAngle degrees. Zero degrees points to the right, and 90 degrees points up (standard mathematical practice but worth mentioning since the y axis is inverted).

To draw a circle, the width and height must be equal in length, and the start and end angles must be 0 and 360. If the width and height are different lengths, this routine draws an oval.

Introduced in platform.apiLevel = '1.0'

9.3 drawImage

```
gc:drawImage(imageHandle, x, y)
```

Draws an image at (x, y). The image must have been created by a previous call to [image.new\(...\)](#).

Introduced in platform.apiLevel = '1.0'

9.4 drawLine

```
gc.drawLine(x1, y1, x2, y2)
```

Draws a line from (x1,y1) to (x2,y2).

Introduced in `platform.apiLevel = '1.0'`

9.5 drawPolyLine

```
gc.drawPolyLine({x1, y1, x2, y2, ..., xn, yn})
```

Draws a series of lines connecting the (x, y) points. The polygon is not closed automatically. The first x-y coordinate pair must be repeated at the end of the array of points to draw a closed polygon.

Introduced in `platform.apiLevel = '1.0'`

9.6 drawRect

```
gc.drawRect(x, y, width, height)
```

Draws a rectangle at (x, y) with the given pixel width and height. Both width and height must be ≥ 0 .

Introduced in `platform.apiLevel = '1.0'`

9.7 drawString

```
gc.drawString("text", x, y [,verticalalignment])
```

Draws text on the window beginning at pixel location (x,y). Vertical alignment may be “baseline”, “bottom”, “middle”, or “top”. This aligns the text in the height of the characters’ bounding rectangle.

Prior to `platform.apiLevel = '2.3'` “none” was used to specify unspecified alignment. The vertical alignment “none” has been [deprecated](#). Specifying no alignment defaults to “top” and so does “none”.

Returns the x pixel position after the text.

Introduced in `platform.apiLevel = '1.0'`

Extended in `platform.apiLevel = '2.3'`

9.8 fillArc

```
gc.fillArc(x, y, width, height, startAngle, endAngle)
```

Fills an arc with the preset color. Both width and height must be ≥ 0 . See [setColorRGB](#) to set the fill color.

Introduced in `platform.apiLevel = '1.0'`

9.9 fillPolygon

```
gc.fillPolygon({x1, y1, x2, y2, ... xn, yn})
```

Fills a polygon with the preset color. The array of points bounds the polygon. To set the fill color, see [setColorRGB](#).

Introduced in `platform.apiLevel = '1.0'`

9.10 fillRect

```
gc:fillRect(x, y, width, height)
```

Fills a rectangle with the preset color. Both the width and height must be ≥ 0 . To set the fill color, see [setColorRGB](#).

Introduced in `platform.apiLevel = '1.0'`

9.11 getStringHeight

```
gc:getStringHeight("text")
```

Returns the pixel height of the text. The pixel height is determined by the font setting previously set by a call to `setFont`.

Introduced in `platform.apiLevel = '1.0'`

9.12 getStringWidth

```
gc:getStringWidth("text")
```

Returns the pixel width of text. The pixel width is calculated using the font setting previously set by a call to `setFont`.

Introduced in `platform.apiLevel = '1.0'`

9.13 setColorRGB

```
gc:setColorRGB(red, green, blue)
gc:setColorRGB(0xRRGGBB) -- API Level > '1.0'
```

Sets the color for subsequent draw and fill routines. The red, green, and blue components of the color are values in the range of 0 to 255. Black is 0,0,0 and white is 255,255,255. Alternately, a single value can be passed in. The components of this single value are `blue + 255 * (green + 255 * red)`.

Introduced in `platform.apiLevel = '1.0'`

Extended in `platform.apiLevel = '2.0'`

9.14 setFont

```
gc:setFont(family, style, size)
```

Sets the font for drawing text and measuring text size. Family may be “sansserif” or “serif”. Style may be “r” for regular, “b” for bold, “i” for italic, or “bi” for bold italic.

The point size of the font is restricted on the TI-Nspire™ CX and older handhelds. Choose one of these sizes: 7, 9, 10, 11, 12, or 24. Any font size supported by Windows® or Mac OS® can be used on the desktop software.

Returns the font family, style, and size previously in effect.

Introduced in `platform.apiLevel = '1.0'`

9.15 setPen

```
gc:setPen([thickness[, style]])
```

Sets the pen for drawing lines and borders. Thickness may be “thin”, “medium”, or “thick”. If the thickness is not specified, it defaults to “thin”. The style can be “smooth”, “dotted”, or “dashed”. If the style is not specified, it defaults to “smooth”.

Introduced in `platform.apiLevel = '1.0'`

Chapter 10

Image Library

An “image” object is a container for graphical images, typically small GUI objects such as buttons, arrowheads, and other such graphical adornments.

Starting with `platform.apiLevel = '2.3'` this library has been reworked to image resources rather than images encoded as strings inside the script itself. Please refer to [section B.1](#) on page 163 for details about the deprecated behavior.

10.1 new

```
img = image.new(string) -- API Level < 2.3  
img = image.new(resource) -- API Level 2.3
```

This function returns a new image object from an image resource or string, based on the API level. These two different kinds of image description cannot be mixed within one script. Image resources support alpha blending on all platforms of the TI-Nspire™ product family.

For details about authoring image resources please refer to the Script Editor section in either the teacher or student [TI-Nspire™ software guidebook](#).

Introduced in `platform.apiLevel = '1.0'`

Extended in `platform.apiLevel = '2.3'`

10.2 copy

```
cimage = image:copy(width, height)
```

Returns a copy of the input image scaled to fit the specified pixel width and height. The width and height default to the size of the input image.

Introduced in `platform.apiLevel = '1.0'`

10.3 height

```
h = image:height()
```

Returns the pixel height of the image.

Introduced in `platform.apiLevel = '1.0'`

10.4 rotate

```
rimage = image:rotate(angle)
```

Returns a copy of the input image rotated counterclockwise by **angle** degrees.

Introduced in `platform.apiLevel = '2.0'`

10.5 width

```
w = image.width()
```

Returns the pixel width of the image.

Introduced in platform.apiLevel = '1.0'

Chapter 11

Locale Library

11.1 name

```
locale.name()
```

Returns the name of the current locale. The locale name is a two-letter language code. The language code may be followed by an underscore and two-letter country code.

Introduced in `platform.apiLevel = '1.0'`

Chapter 12

Math Library Extension

In addition to the functions that come with the standard Lua math library, there is an interface to the TI-Nspire™ math server that allows access to the advanced mathematical features of the TI-Nspire™ product.

Note

The TI-Nspire™ math server uses a number of unicode characters. For example, the math server uses Unicode character U+F02F, *i*, UTF-8 character “\239\128\175”, for imaginary numbers and another special character for the exponent for a scientific notation, small capital letter “E”.

See <http://en.wikipedia.org/wiki/UTF-8> for a description of how to convert unicode to UTF-8 and vice versa. See [TI-Nspire™ Reference Guide](#) for a list of unicode characters used in TI-Nspire™ software.

All results from the TI-Nspire™ math server are returned as full-precision expressions. To limit the precision of the result to the display digits, retrieve the current display digits via `math.getEvalSettings()` and apply the appropriate precision before displaying the value returned by the TI-Nspire™ math server.

12.1 eval

```
math.eval(math_expression) -- platform.apiLevel = '2.0'  
math.eval(math_expression [,exact]) -- platform.apiLevel = '1.0'
```

This function sends an expression or command to the TI-Nspire™ math server for evaluation. The input expression must be a string that the TI-Nspire™ math server can interpret and evaluate.

The second parameter, **exact**, (`platform.apiLevel = '1.0'` only) is meaningful only with the Computer Algebra System. If true, it instructs the math server to calculate and return exact numerical results when it can. The default value of `exact` is false, in which case the math server attempts to calculate an approximate result.

Beginning with `platform.apiLevel = '2.0'`, the evaluation is performed using the current document settings, except that all evaluations are performed at full precision in approximate mode. The current document settings can be overridden by [math.setEvalSettings](#).

If the math server evaluates the expression successfully, it returns the results as a fundamental Lua data type. If the math server cannot evaluate the expression because of a syntax, simplification, or semantic error, **eval** returns two results: nil and an error number meaningful to the math server. (The error numbers are documented in the [TI-Nspire™ Reference Guide - Error Codes and Messages for math.eval](#).) If the math server calculates a symbolic result, it cannot be represented as a fundamental Lua type, so **eval** returns nil and the string “incompatible data type.”

Example

To evaluate **f1** for a given value in *x*, the parameter *x* must be converted to a string, and then any embedded “e” must be replaced with Unicode character U+F000.

Listing 12.1:
Converting a Lua Number to a String to be Used in `math.eval()` (E Notation)

```
local mx = tostring(x):gsub("e", string.uchar(0xF000))  
local expr = "f1(" .. mx .. ")"  
return math.eval(expr)
```

Note

Because **math.eval** always does calculations in approximate mode, things like Boolean logic and some conversions will throw an error:

`r,e = math.eval('1 and 2')` returns “Argument must be a Boolean expression or integer” error

`r,e = math.eval("0@>Base10")` returns “Domain Error”

`math.evalStr` works fine in such cases.

Warning

`math.eval` is not available during script initialization.

Introduced in `platform.apiLevel = '1.0'`

Extended in `platform.apiLevel = '2.0'`

12.2 evalStr

```
math.evalStr(math_expression)
```

This function sends an expression or command to the TI-Nspire™ math server for evaluation. The input expression must be a string that the TI-Nspire™ math server can interpret and evaluate. The evaluation is performed using the current document settings, which can be overridden by [math.setEvalSettings](#). NOTE: All evaluations are performed at full precision regardless of the document settings or overrides.

If the math server evaluates the expression successfully, it returns the results as a string. The `evalStr` function returns no result if the math server does not return a calculated result. If the math server cannot evaluate the expression because of a syntax, simplification, or semantic error, `evalStr` returns two results: nil and an error number meaningful to the math server.

Scientific Notation

The evaluation of “10.2^ 20” (document settings in auto mode) returns the following result: 1.4859473959784 20. A closer look at the result string reveals the box character as “\239\128\128”, which is the Unicode character U+F000 – a small capital letter “E” used inside TI-Nspire™ software for the E notation.

Listing 12.2: `math.evalStr()` Returning Result in E Notation

```
result, error = math.evalStr('10.2^20')
firstFive = table.concat({string.byte(result, 1, 5)}, ' ')
lastFive = table.concat({string.byte(result, 15, 20)}, ' ')
print (result, ':', firstFive, '...', lastFive)
```

Listing 12.2 prints:

```
1.4859473959784 20 : 49 46 52 56 53 ... 52 239 128 128 50 48
```

Negative numbers

The evaluation of “2-3” returns “-1”. The result string will be encoded as “\226\136\146\49”. “\226\136\146” is Unicode character U+2212, which is a minus sign.

Listing 12.3: `math.evalStr()` Returning Negative Numbers

```
result, error = math.evalStr('2-3')
print (result, ':', string.byte(result, 1, 10))
```

Listing 12.3 prints:

```
-1 : 226 136 146 498
```

Introduced in `platform.apiLevel = '2.0'`

12.3 getEvalSettings

```
math.getEvalSettings()
```

Returns a table of tables with the document settings that are currently being used by [math.eval](#). These settings are equivalent to the current document settings unless a call has been made to [setEvalSettings](#).

Listing 12.4: TI-Nspire™ Software Default Settings Returned by getEvalSettings

```
{
  {'Display Digits', 'Float6'},
  {'Angle Mode', 'Radian'},
  {'Calculation Mode', 'Auto'},
  {'Real or Complex Format', 'Real'},
  {'Exponential Format', 'Engineering'},
  {'Vector Format', 'Normal'},
  {'Base', 'Decimal'},
  {'Unit System', 'SI'}, }
}
```

Introduced in `platform.apiLevel = '2.0'`

12.4 setEvalSettings

```
math.setEvalSettings(settingStructure)
```

This function is used to override one or more of the current document settings for all subsequent math evaluations performed by [math.eval](#) and [math.evalStr](#). It does not change the document context settings. The setting structure is a table of tables. Each inner table consists of the name of the document setting to override and the name of the value to use instead.

Listing 12.5: Calling math.setEvalSettings() using a table with names

```
settings = {
  {'Unit System', 'Eng/US'},
  {'Calculation Mode', 'Approximate'},
  {'Real or Complex Format', 'Polar'},
  {'Exponential Format', 'Engineering'}
}

math.setEvalSettings(settings)
```

For user convenience, `setEvalSettings` also accepts the ordinal number of the setting to override and the ordinal number of the value to use instead. The ordinal numbers to use correspond to the order of the settings and their values found at File > Settings > Document Settings.

Listing 12.6: Calling math.setEvalSettings() using a table with ordinal number

```
settingsTable = {
  {2, 3},
  {4, 3},
  {6, 3},
  {8, 2}
}

math.setEvalSettings(settingsTable)
```

In fact, `setEvalSettings` accepts any combination of names and ordinal numbers. So the following example is also valid.

Listing 12.7: Calling math.setEvalSettings() using a table with combined names and numbers

```
settings = {
  {3, 'Exact'},
  {'Angle Mode', 2},
  {'Real or Complex Format',
  'Polar'},
  {8, 2}
}

math.setEvalSettings(settings)
```

The function **math.setEvalSettings** may be called at any point in the script app. The modified document settings are used by **math.eval** for all subsequent calls within the script app (unless modified by a subsequent call to **setEvalSettings**).

Precision of Results

All results from the TI-Nspire™ math server are returned as full-precision expressions. If users want to limit the display digits, they must call `math.getEvalSettings()` and apply the appropriate precision before displaying the value returned by the TI-Nspire™ math server.

Introduced in platform.apiLevel = '2.0'

Chapter 13

Module Library

```
require '<library name>'
```

Use **require** to load predefined libraries in TI-Nspire™ software. Please see the following table.

The behavior of **require** is the same as in standard Lua, but the available libraries are restricted. User-defined libraries are not supported.

| Library | Description |
|------------|--|
| color | Table defining colors used in TI-Nspire™ software to color objects using the color picker. |
| physics | Loads the physics module. |
| ble | Basic <i>Bluetooth</i> ® LE Interface |
| bleCentral | <i>Bluetooth</i> ® LE Interface for the central role |

Colors defined in color table:

| | | | | | |
|--------|----------|------------|------------|-----------|--------|
| black | darkgray | gray | mediumgray | lightgray | white |
| navy | blue | brown | red | magenta | orange |
| yellow | green | dodgerblue | | | |

Introduced in `platform.apiLevel = '2.0'`

Chapter 14

Platform Library

Platform specific information is available through the platform library.

14.1 apiLevel

```
platform.apiLevel
```

Uniquely identifies the Script environment. If the script does not request a desired API level it will always default to the API level the script was created with.

Requesting a non-supported API level will result in the highest supported but below the requested API level supported by the TI-Nspire™ software version running the script. But requesting an API level below **platform.apiLevel = '1.0'** will result in the current API level of TI-Nspire™ software version running the script. Please see [section A.1](#) for more details.

Note

- If present, the platform.apiLevel = 'X.X' statement should be in the main part of the script only. It is advisable to place it on the first line of the script.
- Dynamically loaded scripts (**load()** or **loadstring()**) will use the same "platform.apiLevel = 'X.X'" as the main script. Requesting to change the API level within dynamically loaded scripts causes an error.

Introduced in platform.apiLevel = '2.0'¹

Extended in platform.apiLevel = '2.3'

14.2 hw

```
platform.hw()
```

Returns a numeric value that indicates the CPU speed of the host hardware. The higher the number, the faster the hardware.

| level | host hardware |
|-------|--|
| 3 | TI-Nspire™ family of handhelds |
| 7 | Microsoft® Windows®, Mac® and TI-Nspire™ App |

Introduced in platform.apiLevel = '2.0'

14.3 isColorDisplay

```
platform.isColorDisplay()
```

Returns **true** if the display of the host platform is color. Returns **false** if the display is grayscale.

Introduced in platform.apiLevel = '1.0'

¹Please see [section B.4](#) on for details about the original behavior.

14.4 isDeviceModeRendering

```
platform.isDeviceModeRendering()
```

Returns **true** if the script is running on the handheld or in the emulator of the desktop software. Returns **false** if the script is running in the normal view of the desktop software.

Note

platform.isDeviceModeRendering is not available during script initialization or within **on.restore**.

Introduced in platform.apiLevel = '1.0'

14.5 isTabletModeRendering

```
platform.isTabletModeRendering()
```

Returns **true** if the script is running on a tablet supporting touch otherwise **false**.

Introduced in platform.apiLevel = '2.2'

14.6 registerErrorHandler

```
platform.registerErrorHandler(function(lineNumber, errorMessage,  
                                     callStack, locals) ... end)
```

This function sets the error handler callback function for the script. Setting an error handler callback function provides control over what happens when an error is encountered in the script. Returning a **true** value prevents reporting the Error to the user. The script will continue executing on the next event.

Note

The error handler callback function is not called for errors that occur during initialization or within **on.restore**.

Introduced in platform.apiLevel = '2.0'

14.7 window

```
platform.window
```

Returns the window object that the script application currently owns. The window consists of the portion of the page allotted to the script app. Several applications can be visible when the page is arranged in a split layout. Each visible application has its own window.

The window object has several methods of particular interest.

Introduced in platform.apiLevel = '1.0'

14.7.1 height and width

```
platform.window.height()  
platform.window.width()
```

Routines **height()** and **width()** return the pixel height and width respectively of the display window.

Introduced in platform.apiLevel = '1.0'

14.7.2 invalidate

```
platform.window.invalidate(x, y, width, height)
```

This function invalidates a region of the window and forces it to repaint. x and y default to (0, 0) and width and height default to the pixel width and height of the window. The entire window can be forced to repaint with a call to `platform.window.invalidate()`, which allows all parameters to take their default values.

For performance reasons, especially for large screen resolutions, it is advisable to invalidate not all of the screen but the smallest possible region.

Caution

Please make sure for moving objects to invalidate both the old and the new location of the object. In addition, based on the selected pen setting of the graphics library, drawing lines and other shapes may draw to some degree outside of the specified area. The extent of this area around the specified area might vary in addition by platform. Therefore, add some additional space around the invalidate region to avoid drawing artifacts.

Invalidating multiple regions at a time might result in one or multiple calls to `on.paint` depending on the region and the platform. Therefore the implementation of `on.paint` should not make any assumptions about the region to draw, but always draw all of the screen.

Introduced in `platform.apiLevel = '1.0'`

14.7.3 setBackgroundColor

```
platform.window.setBackgroundColor([0xRRGGBB])
```

Sets the background color for the Script Application. If no color is provided, it defaults to none, causing the background color of the Script Application to be the one of the system which is white.

Introduced in `platform.apiLevel = '2.4'`

14.7.4 setFocus

```
platform.window.setFocus(boolean)
```

This function sets the focus to the main window. Any focus of other objects is removed (currently only D2Editor).

Introduced in `platform.apiLevel = '2.0'`

14.7.5 getScrollHeight

```
platform.window.getScrollHeight()
```

This function returns the current scroll height if a docked keyboard is shown or 0 otherwise. Therefore the return value will always be 0 on platforms not supporting touch.

See [touch.isKeyboardAvailable\(\)](#) for details about keyboard availability.

Introduced in `platform.apiLevel = '2.2'`

14.7.6 setScrollHeight

```
platform.window.setScrollHeight()
```

Sets the scroll height if a docked keyboard is shown or is ignored otherwise. The valid range for this function is 0 to `<keyboard overlap height>`. See [on.keyboardUp\(\)](#) for keyboard overlap height.

Introduced in `platform.apiLevel = '2.2'`

14.7.7 displayInvalidatedRectangles

```
platform.window.displayInvalidatedRectangles(boolean)
```

Displays rectangles surrounding the actual invalidated area by the platform. Available for Computer Preview, TI-Nspire™ CX Handheld and TI-Nspire™ CX iPad Apps. No operation on Handheld Preview.

Introduced in `platform.apiLevel = '2.7'`

14.8 withGC

```
platform.withGC(function, ...)
```

Executes `function(... , gc)` within a non-painting graphics context and returns all return values from `function()`. It is used to support layout procedures that measure the width and height of strings outside of the paint context. It is a good practice to separate the layout from the paint routine to enhance the performance of the script. A layout may happen during `on.resize()` and when data is changing based on user interaction or timer expiration. The script should not assume that any state, like a font size, is preserved from one call of `platform.withGC` to the next call of `platform.withGC`.

This graphics context cannot be used to draw.

Listing 14.1: Example of Using withGC() to get the Pixel Length and Height of a String

```
function getHeightWidth(str, gc)
  gc.setFont('serif', 'b', 12) -- Set the font
  width = gc.getStringWidth(str) -- Pixel length of str
  height = gc.getStringHeight(str) -- Pixel height of str
  return height, width
end

height, width = platform.withGC(getHeightWidth, 'Hello World')
```

Introduced in `platform.apiLevel = '2.0'`

14.9 getDeviceID

```
platform.getDeviceID()
```

Returns the Handheld Product ID.

Introduced in `platform.apiLevel = '2.7'`

Chapter 15

String Library Extension

In addition to the standard Lua string functions, a few routines aid handling Unicode strings.

15.1 split

```
string.split(str [,delim])
```

Divides `str` into substrings based on a delimiter, returning a list of the substrings. The default pattern for the delimiter is white space (“%s+”).

Introduced in `platform.apiLevel = '1.0'`

15.2 uchar

```
string.uchar(chnum, ...)
```

Unicode characters can be included in strings by encoding them in UTF-8. This routine converts one or more Unicode character numbers into a UTF-8 string.

Introduced in `platform.apiLevel = '1.0'`

15.3 usub

```
string.usub(str, startpos, endpos)  
or  
str:usub(startpos, endpos)
```

This routine returns a substring of `str`. It is the Unicode version of `string.sub`. It accounts for multi-byte characters encoded in UTF-8.

Caution

This is an expensive routine. It allocates a temporary memory buffer during its operation.

Listing 15.1: Examples for `string.usub()`

```
print(string.usub("abc", 1, 1)) -- prints "a"  
print(string.usub("abc", 2, 2)) -- prints "b"  
print(string.usub("abc", 2, 3)) -- prints "bc"
```

Introduced in `platform.apiLevel = '1.0'`

15.4 pack

```
characteristicValue = string.pack("formatString", ...)
```

Packs one or multiple Lua values into a *Bluetooth*® LE characteristic data value. The number or arguments after the `formatString` must match the number of formats specified inside the `formatString`. The format specifier used to build the `formatString` as specified in [Table 20.1](#) and additional details can be found in [subsection 20.1.5](#).

| Parameter | Type | Description |
|---------------------|------------|---|
| "formatString" | in string | Lists one or multiple formats to be packed |
| ... | in any | The parameter list associated to the format specified |
| characteristicValue | out string | The packed characteristic value to be written. |

Listing 15.2: Example Showing the use of `string.pack()`

```
data = string.pack("bb2b", true, 2, false) -- binary data 1100
data = string.pack("bbr2b", true, false, true) -- binary data 10001
```

If the format is complex and repetitively used across multiple characteristic values it is possible to split the packing of the data into multiple calls to `pack`. Combining the multiple `pack` results into one piece of data can be achieved by string concatenation. Listing 15.3 shows two simple lines which result in the same data value.

Listing 15.3: Concatenation of Multiple calls to `string.pack()`

```
data1 = string.pack('u8u8', 10, 12)
data2 = string.pack('u8', 10) .. string.pack('u8', 12)
```

Introduced in `platform.apiLevel = '2.7'`

15.5 unpack

```
..., remnant = string.unpack("formatString", characteristicValue)
```

Unpacks a *Bluetooth*® LE characteristic data value into one or multiple Lua values. The number of returned values is defined by the format specifiers inside the `formatString`. All supported format specifiers are listed in Table 20.1 and additional details can be found in subsection 20.1.5 .

| Parameter | Type | Description |
|---------------------|------------|--|
| "formatString" | in string | Lists one or multiple formats to be unpacked |
| characteristicValue | in string | The characteristic value read. |
| ... | out any | The parameter list associated to the format specified |
| remnant | out string | The remnant of the characteristicValue if the format did not decode all data, or <code>nil</code> otherwise. |

Listing 15.4: Example Showing the use of `string.unpack()`

```
bool1, number, bool2 = string.unpack("bb2b", data)
bool1, bool2, bool3 = string.unpack("bbr2b", data)
```

Similar to the `pack` function it is possible to split the unpacking of the data into multiple calls to `unpack`. This can be achieved by passing the remnant returned of one call to `unpack` as characteristic value to the next call of `unpack`. Listing 15.5 show the two scenarios.

Listing 15.5: Splitting Unpacking into Multiple calls to `string.unpack()`

```
ten, twelve = string.unpack('u8u8', '\\10\\12')
ten, remnant = string.unpack('u8', '\\10\\12') -- returns 10, '\\12'
twelve = string.unpack('u8', remnant) -- returns 12, nil
```

Introduced in `platform.apiLevel = '2.7'`

Chapter 16

Timer Library

Each script application has one timer at its disposal. The timer resolution depends on the platform. It is about 0.02 second on the handheld. Please be cautious with short timer periods on the handheld.

The script application should implement the **on.timer()** function to respond to timer expiration.

The timer continues to send ticks to the script application even when its window is not visible on the screen.

The timer is stopped automatically when the document containing the script application is closed or if the script application is deleted from the document.

16.1 getMilliSecCounter

```
timer.getMilliSecCounter()
```

Returns the value of the internal millisecond counter. The counter rolls over to zero when it passes 2^{32} milliseconds.

Introduced in `platform.apiLevel = '1.0'`

16.2 start

```
timer.start(period)
```

Starts the timer with the given period in seconds. The period must be ≥ 0.01 (10 ms). If the timer is already running when this routine is called, the timer is reset to the new period.

Introduced in `platform.apiLevel = '1.0'`

Caution

`timer.start()` should not be called when processing an `on.timer()` event unless it is the final statement before the `on.timer()` event completes.

16.3 stop

```
timer.stop()
```

Stops the timer.

Introduced in `platform.apiLevel = '1.0'`

Chapter 17

Tool Palette Library

The tool palette provides a menu from which the user can select commands that invoke functionality of the script app.

17.1 register

```
toolpalette.register(menuStructure)
```

The script app uses this routine to register its tool palette with the TI-Nspire™ framework. The menu structure is a table describing the name of each toolbox, the menus that appear in each tool box, and the function to call when the user invokes the menu item.

The function **toolpalette.register()** can be called once in the top level flow of the script app. Once registered, the tool palette is managed automatically by the TI-Nspire™ framework. Up to 15 toolboxes can be created with up to 30 menu items each.

When the user chooses an item from a tool box, the associated function is called with two parameters: the name of the toolbox and the name of the menu item.

A call to **toolpalette.register()** within the paint context might be ignored and should therefore be avoided.

Beginning with apiLevel '2.0' **toolpalette.register()** can be called multiple times in the program flow to change dynamically at runtime.

Calling **toolpalette.register(nil)** deactivates the toolpalette.

[Listing 17.1](#) demonstrate the layout of a tool palette's menu structure.

Introduced in platform.apiLevel = '1.0'

Extended in platform.apiLevel = '2.0'

Listing 17.1: Registering a Tool Palette

```
menu = {
  {"Mode", -- Tool box "Mode"
   {"Decimal", setDec}, -- Menu item "Decimal" calls setDec()
   {"Hexadecimal", setHex},
   "-", -- Section divider
   {"Signed", setSigned},
   {"Unsigned", setUnsigned},
  },
  {"Boolean",
   {"And", binopAnd},
   {"Or", binopOr},
  },
}
toolpalette.register(menu)
```

17.2 enable

```
toolpalette.enable(toolname, itemname, enable)
```

This routine enables or disables a menu item in the tool palette. Parameter **toolname** is a string containing the name of the top level tool box. Parameter **itemname** is a string containing the name of the menu item. Parameter **enable** is a Boolean value that enables the menu item if true or disables the menu item if false.

This routine returns true if the menu item was properly enabled or disabled. It returns nil if the toolname / itemname pair cannot be found in the registered menu items.

Note

`toolpalette.register()` must be called prior to `toolpalette.enable()`.

Introduced in `platform.apiLevel = '1.0'`

17.3 enableCut

```
toolpalette.enableCut(enable)
```

This routine enables or disables the Edit > Cut menu command. Parameter **enable** is a Boolean value that enables the command if true or disables the menu item if false.

Introduced in `platform.apiLevel = '1.0'`

17.4 enableCopy

```
toolpalette.enableCopy(enable)
```

This routine enables or disables the Edit > Copy menu command. Parameter **enable** is a Boolean value that enables the command if true or disables the menu item if false.

Introduced in `platform.apiLevel = '1.0'`

17.5 enablePaste

```
toolpalette.enablePaste(enable)
```

This routine enables or disables the Edit > Paste menu command. Parameter **enable** is a Boolean value that enables the command if true or disables the menu item if false.

Introduced in `platform.apiLevel = '1.0'`

Chapter 18

Variable Library

A symbol table is used by the TI-Nspire™ math engine to calculate and store variables. This library gives scripts access to the variables stored in the symbol table.

Not all variables in the symbol table have compatible types in Lua, but many important variable types are supported: real and integer numbers, strings, and lists of numbers and strings, matrices (represented in Lua as lists of lists), and boolean constants true and false.

18.1 list

```
var.list()
```

This function returns a list of names of variables currently defined in the symbol table.

Introduced in `platform.apiLevel = '1.0'`

18.2 makeNumericList

```
var.makeNumericList(name)
```

Creates a list in the symbol table with the given **name**. The list is optimized to hold numeric values. Routines **storeAt** and **recallAt** operate much more efficiently on lists that are created with this function.

Usage Note

This function cannot be used to create a numeric matrix. Routines `var.recallAt` and `var.storeAt` documented below will work with matrices but only if they are created by some other means (see [Listing 18.1](#)).

Listing 18.1: Example for Accessing a Matrix via the Variable Library

```
var.store("mat", {{1,2}, {3,4}}) -- creates matrix mat
var.storeAt("mat", 13.3, 1, 1)
val = var.recallAt("mat", 1, 1)
```

Introduced in `platform.apiLevel = '2.0'`

18.3 monitor

```
var.monitor(name)
```

Turns on monitoring of the math variable with given **name**. When another application changes the math variable, this script application's `on.varChange` handler is called. See the description of `on.varChange` below. Any other return value from 0 is an error value.

Introduced in `platform.apiLevel = '1.0'`

18.4 recall

```
var.recall(name)
```

Returns the value of a math variable with the given **name**. If the type of the named variable has no compatible Lua type, then nil and an error message are returned.

Introduced in `platform.apiLevel = '1.0'`

18.5 recallAt

```
var.recallAt(name, col [,row])
```

Recalls a value from a cell of a list or matrix in the symbol table. **col** is a 1-based column number of the matrix or list. **row** is a 1-based row number. **row** is only required when recalling a value from a matrix.

This function is optimized to work with numeric values and normally returns a number. If the value of the recalled cell is not numeric, this function returns nil and an error message string.

Introduced in `platform.apiLevel = '2.0'`

18.6 recallStr

```
var.recallStr(name)
```

Returns the value of a math variable with the given **name** as a string. Some math types have no compatible Lua type but all math types can be represented as a string. If the value cannot be recalled even as a string, this function returns nil and an error message.

Introduced in `platform.apiLevel = '1.0'`

18.7 store

```
var.store(name, value)
```

Stores value as a math variable with the given **name**. If the value cannot be stored, an error message is returned; otherwise, nil is returned.

Introduced in `platform.apiLevel = '1.0'`

18.8 storeAt

```
var.storeAt(name, numericValue, col [, row])
```

Stores a numeric value into an element of a math list or matrix with the given **name**. **col** is a 1-based column number of the matrix or list. **row** is a 1-based row number. **row** is only required when storing a value into a matrix.

The value must be numeric. Any other type raises an error.

New values can be appended to a list by storing to one column past the end of the list. This function is useful particularly as an optimization when adding new values to a list during a simulation.

Returns nil on success or “cannot store” if the value cannot be stored at the given index.

Introduced in `platform.apiLevel = '2.0'`

18.9 unmonitor

```
var.unmonitor(name)
```

Turns off monitoring of the named math variable.

Introduced in `platform.apiLevel = '1.0'`

Chapter 19

Physics Library

This is an interface library to Chipmunk Physics version 5.3.4. For details about this library see <http://chipmunk-physics.net/release/Chipmunk-5.x/Chipmunk-5.3.4-Docs/>.

To use this library the physics module must be loaded: “`require ('physics')`”.

This library is introduced in `platform.apiLevel = '2.0'`.

19.1 Miscellaneous routines

19.1.1 INFINITY

```
infinity = physics.misc.INFINITY()
```

| Parameter | Type | Description |
|-----------|------------|----------------|
| Infinity | out number | Infinity value |

Returns a number representing infinity in the physics engine.

Introduced in `platform.apiLevel = '2.0'`

19.1.2 momentForBox

```
inertia = physics.misc.momentForBox(mass, width, height)
```

| Parameter | Type | Description |
|-----------|------------|------------------------|
| mass | in number | The mass of the box |
| width | in number | The width of the box |
| height | in number | The height of the box |
| inertia | out number | The inertia of the box |

This routine computes the moment of inertia for a solid box. This is a useful helper routine for computing the moment of inertia as an input to the `physics.Body(...)` constructor.

Introduced in `platform.apiLevel = '2.0'`

19.1.3 momentForCircle

```
inertia = physics.misc.momentForCircle(mass, innerRadius,  
                                        outerRadius, offBody)
```

| Parameter | Type | Description |
|-------------|-----------|--------------------------------|
| mass | in number | The mass of the circle |
| innerRadius | in number | The inner radius of the circle |

| Parameter | Type | Description |
|-------------|-----------------|---|
| outerRadius | in number | The outer radius of the circle |
| offset | in physics.Vect | The offset of the circle from the center of gravity |
| inertia | out number | The inertia of the circle |

This routine computes the moment of inertia for a [circle](#). A solid circle has an inner radius of 0. This is a useful helper routine for computing the moment of inertia as an input to the [physics.Body\(...\)](#) constructor.

Introduced in platform.apiLevel = '2.0'

19.1.4 momentForPoly

```
inertia = physics.misc.momentForPoly(mass, vertices, offset)
```

| Parameter | Type | Description |
|-----------|-------------------|--|
| mass | in number | The mass of the polygon |
| vertices | in {physics.Vect} | A list of vertices defining the shape of the polygon |
| offset | in physics.Vect | The offset of the polygon from the center of gravity |
| inertia | out number | The inertia of the polygon |

This routine computes the moment of inertia for a [polygon](#). This is a useful helper routine for computing the moment of inertia as an input to the [physics.Body\(...\)](#) constructor.

Introduced in platform.apiLevel = '2.0'

19.1.5 momentForSegment

```
inertia = physics.misc.momentForSegment(mass, endPointA,
                                         endPointB)
```

| Parameter | Type | Description |
|-----------|-----------------|---|
| mass | in number | The mass of the segment |
| endPointA | in physics.Vect | The point defining one end of the segment |
| endPointB | in physics.Vect | The point defining the other end of the segment |
| inertia | out number | The inertia of the segment |

This routine computes the moment of inertia for a [segment](#). The end points can be in either world or local coordinates. This is a useful helper routine for computing the moment of inertia as an input to the [physics.Body\(...\)](#) constructor.

Introduced in platform.apiLevel = '2.0'

19.2 Vectors

A vector is a 2-dimensional object with x and y components. Its type is `Tl.cpVect`.

19.2.1 Vect

```
vector = physics.Vect(x, y)
vector = physics.Vect(angle)
vector = physics.Vect(vect)
```

| Parameter | Type | Description |
|-----------|------------------|-------------------------------|
| x | in number | The x component of the vector |
| y | in number | The y component of the vector |
| angle | in number | An angle in radians |
| vect | in physics.Vect | A vector |
| vector | out physics.Vect | A vector |

Creates a vector with initial x and y component values. The second form creates a unit vector pointing in direction angle. The third form creates a copy of the input vector.

Introduced in platform.apiLevel = '2.0'

19.2.2 add

```
sum = physics.Vect:add(vec)
```

| Parameter | Type | Description |
|-----------|------------------|--|
| self | in physics.Vect | The input vector |
| vec | in physics.Vect | A vector to add to self |
| sum | out physics.Vect | The vector sum of self and vec |

Returns the vector sum of **self** and **vec**.

The Vect class also implements the addition operator (+). Therefore vectors **v1** and **v2** can be added with the expression **v1 + v2**.

Introduced in platform.apiLevel = '2.0'

19.2.3 clamp

```
clamped = physics.Vect:clamp(len)
```

| Parameter | Type | Description |
|-----------|------------------|--|
| self | in physics.Vect | The input vector |
| len | in number | The maximum length of the vector |
| clamped | out physics.Vect | A new vector with a length no longer than len |

Returns a copy of **self** clamped to length **len**.

Introduced in platform.apiLevel = '2.0'

19.2.4 cross

```
crossprod = physics.Vect:cross(vec)
```

| Parameter | Type | Description |
|-----------|-----------------|--|
| self | in physics.Vect | The input vector |
| vec | in physics.Vect | A vector to cross with self |
| zmag | out number | The z magnitude of the cross product of self and vec |

Returns the z magnitude of the cross product of **self** and **vec**.

Introduced in platform.apiLevel = '2.0'

19.2.5 dist

```
dist = physics.Vect:dist(vec)
```

| Parameter | Type | Description |
|-----------|-----------------|---|
| self | in physics.Vect | The input vector |
| vec | in physics.Vect | A vector used to find the distance from self |
| dist | out number | The distance from self to vec |

Returns the distance between **self** and **vec**.

Introduced in platform.apiLevel = '2.0'

19.2.6 distsq

```
distsq = physics.Vect:distsq(vec)
```

| Parameter | Type | Description |
|-----------|-----------------|---|
| self | in physics.Vect | The input vector |
| vec | in physics.Vect | The vector used to find the distance squared from self |
| distsq | out number | The distance squared from self to vec |

Returns the distance squared between **self** and **vec**. For distance comparison, this routine is faster than **physics.Vect:dist**.

Introduced in platform.apiLevel = '2.0'

19.2.7 dot

```
dotprod = physics.Vect:dot(vec)
```

| Parameter | Type | Description |
|-----------|-----------------|--|
| self | in physics.Vect | The input vector |
| vec | in physics.Vect | The other vector |
| dotprod | out number | The scalar dot product of self and vec |

Returns the scalar dot product of **self** and **vec**.

Introduced in platform.apiLevel = '2.0'

19.2.8 eql

```
isequ = physics.Vect:eql(vec)
```

| Parameter | Type | Description |
|-----------|-----------------|--|
| self | in physics.Vect | The input vector |
| vec | in physics.Vect | The vector against which to compare with self |
| isequ | out boolean | True if the components of self equal the components of vec |

Returns true if the x and y components of **self** equal those of **vec**. Take the usual precautions when comparing floating point numbers for equality.

The Vect class also implements the equal comparison operator (==). Therefore vectors **v1** and **v2** can be compared with the expression **v1 == v2**.

Introduced in platform.apiLevel = '2.0'

19.2.9 length

```
len = physics.Vect:length()
```

| Parameter | Type | Description |
|-----------|-----------------|----------------------------------|
| self | in physics.Vect | The input vector |
| len | out number | The length of vector self |

Returns the magnitude of **self**.

Introduced in platform.apiLevel = '2.0'

19.2.10 lengthsq

```
lensq = physics.Vect:lengthsq()
```

| Parameter | Type | Description |
|-----------|-----------------|--|
| self | in physics.Vect | The input vector |
| lensq | out number | The length squared of vector self |

Returns the length squared of **self**. This routine is faster than Vect:length() when you only need to compare lengths.

Introduced in platform.apiLevel = '2.0'

19.2.11 lerp

```
v = physics.Vect:lerp(vec, f)
```

| Parameter | Type | Description |
|-----------|------------------|--|
| self | in physics.Vect | The input vector |
| vec | in physics.Vect | The other vector |
| f | in number | f is a fractional number from 0 to 1 representing the proportion of distance between self and vec |
| v | out physics.Vect | A vector interpolated between self and vec |

Returns the linear interpolation between **self** and **vec** as a vector. **f** is the fraction of distance between **self** and **vec**.

Note

May not behave as expected for **f** larger than 1.0 or less than 0.

Introduced in platform.apiLevel = '2.0'

19.2.12 lerpconst

```
v = physics.Vect:lerpconst(vec, d)
```

| Parameter | Type | Description |
|-----------|------------------|---|
| self | in physics.Vect | The input vector |
| vec | in physics.Vect | The other vector |
| d | in number | The distance from self to vec to interpolate a new vector |
| v | out physics.Vect | |

Returns a vector interpolated from **self** towards **vec** with length **d**.

Note

May not behave as expected for d larger than 1.0 or less than 0.

Introduced in platform.apiLevel = '2.0'

19.2.13 mult

```
v = physics.Vect:mult(factor)
```

| Parameter | Type | Description |
|-----------|------------------|--------------------------------------|
| self | in physics.Vect | The input vector |
| factor | in number | The value to multiply by self |
| v | out physics.Vect | The resulting scaled vector |

Multiplies a vector by a factor.

Introduced in platform.apiLevel = '2.0'

19.2.14 near

```
isnear = physics.Vect:near(vec, distance)
```

| Parameter | Type | Description |
|-----------|-----------------|---|
| self | in physics.Vect | The input vector |
| vec | in physics.Vect | The value to multiply by self |
| distance | in number | The distance from vec |
| isnear | out boolean | True if self is within distance of vec |

Determines if **self** is near another vector.

Introduced in platform.apiLevel = '2.0'

19.2.15 neg

```
v = physics.Vect:neg()
```

| Parameter | Type | Description |
|-----------|------------------|------------------------------|
| self | in physics.Vect | The input vector |
| v | out physics.Vect | The resulting negated vector |

Returns the negative of **self**.

The Vect class also implements the unary minus operator (**-self**).

Introduced in platform.apiLevel = '2.0'

19.2.16 normalize

```
normvec = physics.Vect:normalize()
```

| Parameter | Type | Description |
|-----------|------------------|---------------------------------|
| self | in physics.Vect | The input vector |
| normvec | out physics.Vect | The resulting normalized vector |

Returns a normalized copy of **self**. The length of a normal vector is 1.

Introduced in `platform.apiLevel = '2.0'`

19.2.17 normalizeSafe

```
normvec = physics.Vect:normalizeSafe()
```

| Parameter | Type | Description |
|-----------|------------------|---------------------------------|
| self | in physics.Vect | The input vector |
| normvec | out physics.Vect | The resulting normalized vector |

Returns a normalized copy of `self`. Protects against division by zero.

Introduced in `platform.apiLevel = '2.0'`

19.2.18 perp

```
perpvec = physics.Vect:perp()
```

| Parameter | Type | Description |
|-----------|------------------|------------------------------------|
| self | in physics.Vect | The input vector |
| perpvec | out physics.Vect | The resulting perpendicular vector |

Returns a vector perpendicular to `self`. (90 degree rotation).

Introduced in `platform.apiLevel = '2.0'`

19.2.19 project

```
pvec = physics.Vect:project(vec)
```

| Parameter | Type | Description |
|-----------|------------------|---|
| self | in physics.Vect | The input vector |
| vec | in physics.Vect | The other vector |
| pvec | out physics.Vect | The vector of <code>self</code> projected onto <code>vec</code> |

Computes the projection of `self` onto another vector.

Introduced in `platform.apiLevel = '2.0'`

19.2.20 rotate

```
rvec = physics.Vect:rotate(vec)
```

| Parameter | Type | Description |
|-----------|------------------|------------------------------|
| self | in physics.Vect | The input vector |
| vec | in physics.Vect | The other vector |
| rvec | out physics.Vect | The resulting rotated vector |

Uses complex multiplication to rotate `self` by `vec`. Scaling will occur if `self` is not a unit vector.

Introduced in `platform.apiLevel = '2.0'`

19.2.21 rperp

```
perpvec = physics.Vect:rperp()
```

| Parameter | Type | Description |
|-----------|------------------|------------------------------------|
| self | in physics.Vect | The input vector |
| perpvec | out physics.Vect | The resulting perpendicular vector |

Returns a vector perpendicular to **self**. (90 degree rotation)

Introduced in `platform.apiLevel = '2.0'`

19.2.22 setx

```
self = physics.Vect:setx(x)
```

| Parameter | Type | Description |
|-----------|------------------|---|
| self | in physics.Vect | The vector to modify |
| x | in number | The new value of the x component of the vector |
| self | out physics.Vect | The input vector is returned as the output |

Changes the value of the **x** component of **self**. Returns **self**.

Introduced in `platform.apiLevel = '2.0'`

19.2.23 sety

```
self = physics.Vect:sety(y)
```

| Parameter | Type | Description |
|-----------|------------------|---|
| self | in physics.Vect | The vector to modify |
| y | in number | The new value of the y component of the vector |
| self | out physics.Vect | The input vector is returned as the output |

Changes the value of the **y** component of **self**. Returns **self**.

Introduced in `platform.apiLevel = '2.0'`

19.2.24 slerp

```
v = physics.Vect:slerp(vec, f)
```

| Parameter | Type | Description |
|-----------|------------------|--|
| self | in physics.Vect | A unit vector |
| vec | in physics.Vect | The other unit vector |
| f | in number | f is a fractional number from 0 to 1 representing the proportion of distance between self and vec |
| v | out physics.Vect | A vector interpolated between self and vec |

Computes a spherical linear interpolation between unit vectors **self** and **vec**. See <http://en.wikipedia.org/wiki/Slerp> for a discussion of the meaning, value, and usage of spherical linear interpolation.

Listing 19.1: Spherical Linear Interpolation Example

```
local vect1 = physics.Vect(math.pi/3) -- unit vector
local vect2 = physics.Vect(math.pi/2) -- unit vector
local result = vect1:slerp(vect2, 0.55)
```

Note

This routine computes meaningful results only when the two inputs are unit vectors. May not behave as expected for f larger than 1.0 or less than 0.

Introduced in platform.apiLevel = '2.0'

19.2.25 slerpconst

```
v = physics.Vect:slerpconst(vec, angle)
```

| Parameter | Type | Description |
|-----------|------------------|--|
| self | in physics.Vect | A unit vector |
| vec | in physics.Vect | The other unit vector |
| angle | in number | The maximum angle between self and vec to interpolate a new vector |
| v | out physics.Vect | |

Returns the spherical linear interpolation from **self** towards **vec**, but by no more than **angle in radians**. See <http://en.wikipedia.org/wiki/Slerp> for a discussion of the meaning, value, and usage of spherical linear interpolation.

Note

This routine computes meaningful results only when the two inputs are unit vectors.

Introduced in platform.apiLevel = '2.0'

19.2.26 sub

```
diff = physics.Vect:sub(vec)
```

| Parameter | Type | Description |
|-----------|------------------|--|
| self | in physics.Vect | The input vector |
| vec | in physics.Vect | A vector to subtract from self |
| diff | out physics.Vect | The vector difference between self and vec |

Returns the vector difference of **self** and **vec**.

The Vect class also implements the subtraction operator (-). Therefore vector **v2** can be subtracted from **v1** with the expression **v1 - v2**.

Introduced in platform.apiLevel = '2.0'

19.2.27 toangle

```
angle = physics.Vect:toangle()
```

| Parameter | Type | Description |
|-----------|-----------------|--------------------------|
| self | in physics.Vect | The input vector |
| angle | out number | The angle of self |

Returns the angle in radians of **self**.

Introduced in platform.apiLevel = '2.0'

19.2.28 unrotate

```
uvec = physics.Vect:unrotate(vec)
```

| Parameter | Type | Description |
|-----------|------------------|--------------------------------|
| self | in physics.Vect | The input vector |
| vec | in physics.Vect | The other vector |
| uvec | out physics.Vect | The resulting unrotated vector |

Inverse of physics.Vect:rotate(vec).

Introduced in platform.apiLevel = '2.0'

19.2.29 x

```
x = physics.Vect:x()
```

| Parameter | Type | Description |
|-----------|-----------------|---|
| self | in physics.Vect | The input vector |
| x | out number | The value of the x component of the vector |

Returns the value of the **x** component of the input vector.

Introduced in platform.apiLevel = '2.0'

19.2.30 y

```
y = physics.Vect:y()
```

| Parameter | Type | Description |
|-----------|-----------------|---|
| self | in physics.Vect | The input vector |
| y | out number | The value of the y component of the vector |

Returns the value of the **y** component of the input vector.

Introduced in platform.apiLevel = '2.0'

19.3 Bounding Boxes

A bounding box is a structure the contains the left, bottom, right, and top edges of a box. Its type is TI.cpBB.

19.3.1 BB

```
bb = physics.BB(l, b, r, t)
```

| Parameter | Type | Description |
|-----------|----------------|---|
| l | in number | left |
| b | in number | bottom |
| r | in number | right |
| t | in number | top |
| bb | out physics.BB | A bounding box with boundaries left, bottom, right, and top |

Returns a new bounding box with the given initial edges.

Introduced in platform.apiLevel = '2.0'

19.3.2 b

```
bottom = physics.BB:b()
```

| Parameter | Type | Description |
|-----------|---------------|-------------------------------------|
| self | in physics.BB | The input bounding box |
| bottom | out number | The bottom edge of the bounding box |

Returns the bottom edge of the bounding box.

Introduced in platform.apiLevel = '2.0'

19.3.3 clampVect

```
cvec = physics.BB:clampVect(vec)
```

| Parameter | Type | Description |
|-----------|------------------|--------------------------------------|
| self | in physics.BB | The input bounding box |
| vec | in physics.Vect | A vector |
| cvec | out physics.Vect | A vector clamped to the bounding box |

Returns a copy of `vec` clamped to the bounding box.

Introduced in platform.apiLevel = '2.0'

19.3.4 containsBB

```
bool = physics.BB:containsBB(other)
```

| Parameter | Type | Description |
|-----------|---------------|--|
| self | in physics.BB | The input bounding box |
| other | in physics.BB | The other bounding box |
| bool | out boolean | True if <code>self</code> completely contains the other bounding box |

Determines if a bounding box contains another bounding box.

Introduced in platform.apiLevel = '2.0'

19.3.5 containsVect

```
bool = physics.BB:containsVect(vec)
```

| Parameter | Type | Description |
|-----------|-----------------|--|
| self | in physics.BB | The input bounding box |
| vec | in physics.Vect | A vector |
| bool | out boolean | True if <code>self</code> contains vector <code>vec</code> |

Determines if a bounding box contains a [vector](#).

Introduced in platform.apiLevel = '2.0'

19.3.6 expand

```
bb = physics.BB:expand(vec)
```

| Parameter | Type | Description |
|-----------|-----------------|--|
| self | in physics.BB | The input bounding box |
| vec | in physics.Vect | A vector |
| bb | out physics.BB | The bounding box self expanded to include vector vec |

Returns the bounding box that contains both **self** and **vec**.

Introduced in platform.apiLevel = '2.0'

19.3.7 intersects

```
bool = physics.BB:intersects(other)
```

| Parameter | Type | Description |
|-----------|---------------|--|
| self | in physics.BB | The input bounding box |
| other | in physics.BB | The other bounding box |
| bool | out boolean | True if self intersects the other bounding box |

Determines if two bounding boxes intersect.

Introduced in platform.apiLevel = '2.0'

19.3.8 l

```
left = physics.BB:l()
```

| Parameter | Type | Description |
|-----------|---------------|-----------------------------------|
| self | in physics.BB | The input bounding box |
| left | out number | The left edge of the bounding box |

Returns the left edge of the bounding box.

Introduced in platform.apiLevel = '2.0'

19.3.9 merge

```
bb = physics.BB:merge(other)
```

| Parameter | Type | Description |
|-----------|----------------|---|
| self | in physics.BB | The input bounding box |
| other | in physics.BB | The other bounding box |
| bb | out physics.BB | The bounding box that contains both self and the other bounding box |

Returns the bounding box that contains both **self** and the other bounding box.

Introduced in platform.apiLevel = '2.0'

19.3.10 setb

```
self = physics.BB:setb(bottom)
```

| Parameter | Type | Description |
|-----------|---------------|------------------------|
| self | in physics.BB | The input bounding box |

| Parameter | Type | Description |
|-----------|----------------|---|
| bottom | in number | The new value for the bottom edge of the bounding box |
| self | out physics.BB | The input bounding box is returned as the output |

Sets the bottom edge of the bounding box to a new **value**. Returns **self**.

Introduced in platform.apiLevel = '2.0'

19.3.11 r

```
right = physics.BB:r()
```

| Parameter | Type | Description |
|-----------|---------------|------------------------------------|
| self | in physics.BB | The input bounding box |
| right | out number | The right edge of the bounding box |

Returns the right edge of the bounding box.

Introduced in platform.apiLevel = '2.0'

19.3.12 setl

```
self = physics.BB:setl(left)
```

| Parameter | Type | Description |
|-----------|----------------|---|
| self | in physics.BB | The input bounding box |
| left | in number | The new value for the left edge of the bounding box |
| self | out physics.BB | The input bounding box is returned as the output |

Sets the left edge of the bounding box to a new **value**. Returns **self**.

Introduced in platform.apiLevel = '2.0'

19.3.13 setr

```
self = physics.BB:setr(right)
```

| Parameter | Type | Description |
|-----------|----------------|--|
| self | in physics.BB | The input bounding box |
| right | in number | The new value for the right edge of the bounding box |
| self | out physics.BB | The input bounding box is returned as the output |

Sets the right edge of the bounding box to a new **value**. Returns **self**.

Introduced in platform.apiLevel = '2.0'

19.3.14 sett

```
self = physics.BB:sett(top)
```

| Parameter | Type | Description |
|-----------|----------------|--|
| self | in physics.BB | The input bounding box |
| top | in number | The new value for the top edge of the bounding box |
| self | out physics.BB | The input bounding box is returned as the output |

Sets the top edge of the bounding box to a new **value**. Returns **self**.

Introduced in `platform.apiLevel = '2.0'`

19.3.15 t

```
top = physics.BB:t()
```

| Parameter | Type | Description |
|-----------|---------------|----------------------------------|
| self | in physics.BB | The input bounding box |
| top | out number | The top edge of the bounding box |

Returns the top edge of the bounding box.

Introduced in `platform.apiLevel = '2.0'`

19.3.16 wrapVect

```
wvec = physics.BB:wrapVect(vec)
```

| Parameter | Type | Description |
|-----------|------------------|--------------------------------------|
| self | in physics.BB | The input bounding box |
| vec | in physics.Vect | A vector |
| wvec | out physics.Vect | A vector wrapped to the bounding box |

Returns a copy of **vec** wrapped to the bounding box.

Introduced in `platform.apiLevel = '2.0'`

19.4 Bodies

A body holds the physical properties (mass, position, rotation, velocity, etc.) of an object. It does not have a [shape](#) until you attach one (or more) to it. Its type is `TI.cpBody`.

19.4.1 Body

```
body = physics.Body(mass, inertia)
```

| Parameter | Type | Description |
|-----------|------------------|---|
| mass | in number | Mass of the body |
| inertia | in number | The inertia of the body |
| body | out physics.Body | A new Body with the supplied mass and inertia |

Returns a new Body with the given mass and moment of inertia.

Use the [provided helper functions](#) to compute the moment of inertia.

Introduced in `platform.apiLevel = '2.0'`

19.4.2 activate

```
self = physics.Body:activate()
```

| Parameter | Type | Description |
|-----------|------------------|--|
| self | in physics.Body | The input Body |
| self | out physics.Body | The input Body is returned as the output |

Activates a sleeping body.

Info

See <http://chipmunk-physics.net/release/Chipmunk-5.x/Chipmunk-5.3.4-Docs/> for an explanation of this routine.

Introduced in platform.apiLevel = '2.0'

19.4.3 angle

```
angle = physics.Body:angle()
```

| Parameter | Type | Description |
|-----------|-----------------|----------------------------------|
| self | in physics.Body | The input Body |
| angle | out number | The angle of the Body in radians |

Returns the angle in radians of the orientation of the body.

Introduced in platform.apiLevel = '2.0'

19.4.4 angVel

```
ave1 = physics.Body:angVel()
```

| Parameter | Type | Description |
|-----------|-----------------|---|
| self | in physics.Body | The input Body |
| ave1 | out number | The angular velocity of the Body in radians per unit time |

Returns the angular velocity of the body in radians per unit time.

Introduced in platform.apiLevel = '2.0'

19.4.5 applyForce

```
self = physics.Body:applyForce(forceVect, rOffset)
```

| Parameter | Type | Description |
|-----------|------------------|---|
| self | in physics.Body | The input Body |
| forceVect | in physics.Vect | A force vector |
| rOset | in physics.Vect | Vector offset of the force relative to the Body |
| self | out physics.Body | The input Body is returned as the output |

Apply force [vector](#) on **self** at a relative offset from the center of gravity.

Introduced in platform.apiLevel = '2.0'

19.4.6 applyImpulse

```
self = physics.Body:applyImpulse(impulseVect, rOffset)
```

| Parameter | Type | Description |
|-------------|------------------|---|
| self | in physics.Body | The input Body |
| impulseVect | in physics.Vect | Impulse force on the Body |
| rOset | in physics.Vect | Vector offset of the force relative to the Body |
| self | out physics.Body | The input Body is returned as the output |

Apply the impulse [vector](#) on `self` at a relative offset from the center of gravity.

Introduced in `platform.apiLevel = '2.0'`

19.4.7 data

```
obj = physics.Body:data()
```

| Parameter | Type | Description |
|-----------|-----------------|--|
| self | in physics.Body | The input Body |
| obj | out Lua object | An object previously set on the Body by the programmer |

Returns the contents of the programmer data eld of the Body.

Introduced in `platform.apiLevel = '2.0'`

19.4.8 force

```
fvec = physics.Body:force()
```

| Parameter | Type | Description |
|-----------|------------------|------------------------------|
| self | in physics.Body | The input Body |
| fvec | out physics.Vect | The force vector on the Body |

Returns the force [vector](#) on the body.

Introduced in `platform.apiLevel = '2.0'`

19.4.9 isRogue

```
bool = physics.Body:isRogue()
```

| Parameter | Type | Description |
|-----------|-----------------|----------------------------------|
| self | in physics.Body | The input Body |
| bool | out boolean | True if the Body is a rogue Body |

Returns true if the Body is a rogue Body, never having been added to the simulation [Space](#).

Info

See <http://chipmunk-physics.net/release/Chipmunk-5.x/Chipmunk-5.3.4-Docs/> for an explanation of rogue bodies.

Introduced in `platform.apiLevel = '2.0'`

19.4.10 isSleeping

```
bool = physics.Body:isSleeping()
```

| Parameter | Type | Description |
|-----------|-----------------|------------------------------|
| self | in physics.Body | The input Body |
| bool | out boolean | True if the Body is sleeping |

Returns true if the body is sleeping.

Info

See <http://chipmunk-physics.net/release/Chipmunk-5.x/Chipmunk-5.3.4-Docs/> for an explanation of rogue bodies.

Introduced in `platform.apiLevel = '2.0'`

19.4.11 local2World

```
wvec = physics.Body:local2World(lvec)
```

| Parameter | Type | Description |
|-----------|------------------|---|
| self | in physics.Body | The input Body |
| lvec | in physics.Vect | A vector relative to the position of the Body |
| wvec | out physics.Vect | A vector in world coordinates |

Converts **lvec** from body-relative coordinates to world coordinates. Returns the converted [vector](#).

Introduced in platform.apiLevel = '2.0'

19.4.12 kineticEnergy

```
ke = physics.Body:kineticEnergy()
```

| Parameter | Type | Description |
|-----------|-----------------|--------------------------------------|
| self | in physics.Body | The input Body |
| ke | out number | The total kinetic energy of the Body |

Returns the kinetic energy of the body..

Introduced in platform.apiLevel = '2.0'

19.4.13 mass

```
m = physics.Body:mass()
```

| Parameter | Type | Description |
|-----------|-----------------|----------------------|
| self | in physics.Body | The input Body |
| m | out number | The mass of the Body |

Returns the mass of the body.

Introduced in platform.apiLevel = '2.0'

19.4.14 moment

```
m = physics.Body:moment()
```

| Parameter | Type | Description |
|-----------|-----------------|-----------------------------------|
| self | in physics.Body | The input Body |
| m | out number | The moment of inertia of the Body |

Returns the moment of inertia of the body.

Introduced in platform.apiLevel = '2.0'

19.4.15 pos

```
p = physics.Body:pos()
```

| Parameter | Type | Description |
|-----------|------------------|--------------------------|
| self | in physics.Body | The input Body |
| p | out physics.Vect | The position of the Body |

Returns the [vector](#) position of the body.

Introduced in platform.apiLevel = '2.0'

19.4.16 resetForces

```
self = physics.Body:resetForces()
```

| Parameter | Type | Description |
|-----------|------------------|--|
| self | in physics.Body | The input Body |
| self | out physics.Body | The input Body is returned as the output |

Zero both the force and torque accumulated on **self**.

Introduced in platform.apiLevel = '2.0'

19.4.17 rot

```
rvec = physics.Body:rot()
```

| Parameter | Type | Description |
|-----------|------------------|---|
| self | in physics.Body | The input Body |
| rvec | out physics.Vect | The unit vector orientation of the Body |

Returns the [vector](#) orientation of the body. This is a unit vector cached from the last calculated angle of the Body.

Introduced in platform.apiLevel = '2.0'

19.4.18 setAngle

```
self = physics.Body:setAngle(angle)
```

| Parameter | Type | Description |
|-----------|------------------|--|
| self | in physics.Body | The input Body |
| angle | in number | The angle of rotation in radians of the Body |
| angle | out physics.Body | The input Body is returned as the output |

Updates the angle of rotation in radians of the body.

Returns the Body.

Introduced in platform.apiLevel = '2.0'

19.4.19 setAngVel

```
self = physics.Body:setAngVel(vel)
```

| Parameter | Type | Description |
|-----------|-----------------|----------------|
| self | in physics.Body | The input Body |

| Parameter | Type | Description |
|-----------|------------|---|
| vel | in number | The angular velocity in radians per unit time of the Body |
| avel | out number | The input Body is returned as the output |

Updates the angular velocity of the body. The angular velocity is in radians per unit time.

Returns the Body.

Introduced in platform.apiLevel = '2.0'

19.4.20 setData

```
self = physics.Body:setData(value)
```

| Parameter | Type | Description |
|-----------|------------------|--|
| self | in physics.Body | The input Body |
| value | in object | A programmer-supplied Lua object |
| self | out physics.Body | The input Body is returned as the output |

Sets the programmer data field of the Body. The programmer can store any Lua object in this field. This is a handy place to store a reference to a simulation object.

Returns the Body.

Introduced in platform.apiLevel = '2.0'

19.4.21 setForce

```
self = physics.Body:setForce(vector)
```

| Parameter | Type | Description |
|-----------|------------------|--|
| self | in physics.Body | The input Body |
| vector | in physics.Vect | The vector of force on the Body |
| self | out physics.Body | The input Body is returned as the output |

Updates the force [vector](#) on the body.

Returns the Body.

Introduced in platform.apiLevel = '2.0'

19.4.22 setMass

```
self = physics.Body:setMass(mass)
```

| Parameter | Type | Description |
|-----------|------------------|--|
| self | in physics.Body | The input Body |
| mass | in number | The mass of the Body |
| self | out physics.Body | The input Body is returned as the output |

Updates the mass of the body.

Returns the Body.

Introduced in platform.apiLevel = '2.0'

19.4.23 setMoment

```
self = physics.Body:setMoment(moment)
```

| Parameter | Type | Description |
|-----------|------------------|--|
| self | in physics.Body | The input Body |
| moment | in number | The moment of inertia of the Body |
| self | out physics.Body | The input Body is returned as the output |

Updates the moment of inertia of the body.

Use the [provided helper functions](#) to compute the moment of inertia.

Returns the Body.

.Introduced in platform.apiLevel = '2.0'

19.4.24 setPos

```
self = physics.Body:setPos(vector)
```

| Parameter | Type | Description |
|-----------|------------------|--|
| self | in physics.Body | The input Body |
| vector | in physics.Vect | The position of the Body |
| self | out physics.Body | The input Body is returned as the output |

Updates the position of the body. Returns the Body.

Returns the Body.

.Introduced in platform.apiLevel = '2.0'

19.4.25 setPositionFunc

```
self = physics.Body:setPositionFunc(func)
```

| Parameter | Type | Description |
|-----------|-----------------------|---|
| self | in physics.Body | The input Body |
| func | in function(body, dt) | A callback function that updates the position of the Body on each time step |
| self | out physics.Body | The input Body is returned as the output |

Sets the position function of the body. The position function must be a function that accepts a Body and a time step value and at some point calls `body:updatePosition` to update the position of the body.

Returns the Body.

.Introduced in platform.apiLevel = '2.0'

19.4.26 setTorque

```
self = physics.Body:setTorque(torque)
```

| Parameter | Type | Description |
|-----------|------------------|--|
| self | in physics.Body | The input Body |
| torque | in number | The torque of the Body |
| self | out physics.Body | The input Body is returned as the output |

Updates the torque on the body. Torque is a numeric magnitude.

Returns the Body.

Introduced in platform.apiLevel = '2.0'

19.4.27 setVel

```
self = physics.Body:setVel(vector)
```

| Parameter | Type | Description |
|-----------|------------------|--|
| self | in physics.Body | The input Body |
| vector | in physics.Vect | The velocity vector of the Body |
| self | out physics.Body | The input Body is returned as the output |

Updates the velocity of the body.

Returns the Body.

Introduced in platform.apiLevel = '2.0'

19.4.28 setVelocityFunc

```
self = physics.Body:setVelocityFunc(func)
```

| Parameter | Type | Description |
|-----------|--------------------------------------|---|
| self | in physics.Body | The input Body |
| func | in function(body, grav, damping, dt) | A callback function that updates the velocity of the Body on each time step |
| self | out physics.Body | The input Body is returned as the output |

Sets the velocity function of the body. The velocity function must be a function that accepts a Body, a gravity [vector](#), a numeric damping factor, and a time step value. The function should call `body:updateVelocity` to adjust the velocity of the body.

Returns the Body.

Listing 19.2: Example for physics.Body:setVelocityFunc()

```
function sampleVelocityFunc(body, gravity, damping, dt)
  local pos = body:pos()
  local sqdist = pos:lengthsq()
  local g = pos:mult(-GravityStrength /
    (sqdist * math.sqrt(sqdist)))
  body:updateVelocity(g, damping, dt)
end

body:setVelocityFunc(sampleVelocityFunc)
```

Introduced in platform.apiLevel = '2.0'

19.4.29 setVLimit

```
self = physics.Body:setVLimit(limit)
```

| Parameter | Type | Description |
|-----------|------------------|--|
| self | in physics.Body | The input Body |
| limit | in number | The maximum speed of the Body |
| self | out physics.Body | The input Body is returned as the output |

Sets the limit for the maximum speed of the body.

Returns the Body.

Introduced in platform.apiLevel = '2.0'

19.4.30 setWLimit

```
self = physics.Body:setWLimit(limit)
```

| Parameter | Type | Description |
|-----------|------------------|--|
| self | in physics.Body | The input Body |
| limit | in number | The maximum angular velocity of the Body |
| self | out physics.Body | The input Body is returned as the output |

Updates the limit of the angular velocity of the body. Angular velocity is in radians per unit time.

Returns the Body.

Introduced in platform.apiLevel = '2.0'

19.4.31 sleep

```
self = physics.Body:sleep()
```

| Parameter | Type | Description |
|-----------|------------------|--|
| self | in physics.Body | The input Body |
| bool | out physics.Body | The input Body is returned as the output |

Puts the Body to sleep.

Info

See <http://chipmunk-physics.net/release/Chipmunk-5.x/Chipmunk-5.3.4-Docs/> for an explanation of sleeping bodies.

Note

The body must be added to a [Space](#) before it can be put to sleep.

Calling this function within a query or callback is not allowed.

Introduced in platform.apiLevel = '2.0'

19.4.32 sleepWithGroup

```
self = physics.Body:sleepWithGroup( [group] )
```

| Parameter | Type | Description |
|-----------|------------------|--|
| self | in physics.Body | The input Body |
| group | in physics.Body | A sleeping body. If this parameter is not supplied, a new group is created |
| bool | out physics.Body | The input Body is returned as the output |

Puts the Body to sleep and adds it to a group of other sleeping bodies.

Info

See <http://chipmunk-physics.net/release/Chipmunk-5.x/Chipmunk-5.3.4-Docs/> for an explanation of this routine.

Note

The body must be added to a [Space](#) before it can be put to sleep.

Calling this function within a query or callback is not allowed.

This routine will raise an exception if **group** is not sleeping.

Introduced in `platform.apiLevel = '2.0'`

19.4.33 torque

```
t = physics.Body:torque()
```

| Parameter | Type | Description |
|-----------|-----------------|------------------------|
| self | in physics.Body | The input Body |
| torque | out number | The torque on the Body |

Returns the torque on the Body.

Introduced in `platform.apiLevel = '2.0'`

19.4.34 updatePosition

```
physics.Body:updatePosition(dt)
```

| Parameter | Type | Description |
|-----------|-----------------|------------------------------|
| self | in physics.Body | The input Body |
| dt | out number | The time interval in seconds |

Updates the position of the body using Euler integration

Info

See <http://chipmunk-physics.net/release/Chipmunk-5.x/Chipmunk-5.3.4-Docs/> for an explanation of this routine.

Introduced in `platform.apiLevel = '2.0'`

19.4.35 updateVelocity

```
physics.Body:updateVelocity(grav, damp, dt)
```

| Parameter | Type | Description |
|-----------|------------------|------------------------------|
| self | in physics.Body | The input Body |
| grav | in physics.Vect | The force of gravity |
| damp | in physics.Vect | The damping factor |
| dt | out physics.Vect | The time interval in seconds |

Updates the velocity of the body using Euler integration.

Info

See <http://chipmunk-physics.net/release/Chipmunk-5.x/Chipmunk-5.3.4-Docs/> for an explanation of this routine.

Introduced in `platform.apiLevel = '2.0'`

19.4.36 vel

```
vvel = physics.Body:vel()
```

| Parameter | Type | Description |
|-----------|------------------|--------------------------|
| self | in physics.Body | The input Body |
| vvel | out physics.Vect | The velocity of the Body |

Returns the [vector](#) velocity of the body.

Introduced in platform.apiLevel = '2.0'

19.4.37 vLimit

```
vmax = physics.Body:vLimit()
```

| Parameter | Type | Description |
|-----------|-----------------|-------------------------------|
| self | in physics.Body | The input Body |
| vmax | out number | The maximum speed of the Body |

Returns the speed limit of the body.

Introduced in platform.apiLevel = '2.0'

19.4.38 wLimit

```
wmax = physics.Body:wLimit()
```

| Parameter | Type | Description |
|-----------|-----------------|---|
| self | in physics.Body | The input Body |
| wmax | out number | The maximum angular velocity of the Body in radians per unit time |

Returns the angular velocity limit of the body. The angular velocity is in radians per unit time.

Introduced in platform.apiLevel = '2.0'

19.4.39 world2Local

```
lvec = physics.Body:world2Local(wvec)
```

| Parameter | Type | Description |
|-----------|------------------|---|
| self | in physics.Body | The input Body |
| wvec | in physics.Vect | A vector in world coordinates |
| lvec | out physics.Vect | A vector relative to the position of the Body |

Converts **wvec** from world coordinates to body-relative coordinates. Returns the converted [vector](#).

Introduced in platform.apiLevel = '2.0'

19.5 Shapes

Shapes contain the surface properties of an object such as how much friction or elasticity it has. All collision shapes implement the following accessor routines.

19.5.1 BB

```
bb = physics.Shape:BB()
```

| Parameter | Type | Description |
|-----------|------------------|---------------------------|
| self | in physics.Shape | The input Shape |
| bb | out physics.BB | Bounding box of the Shape |

Returns the [bounding box](#) of the shape.

Introduced in platform.apiLevel = '2.0'

19.5.2 body

```
body = physics.Shape:body()
```

| Parameter | Type | Description |
|-----------|------------------|------------------------------------|
| self | in physics.Shape | The input Shape |
| body | out physics.Body | The Body associated with the Shape |

Returns the body attached to the shape. If the shape is static, then it will return nil.

Introduced in platform.apiLevel = '2.0'

19.5.3 collisionType

```
coll = physics.Shape:collisionType()
```

| Parameter | Type | Description |
|-----------|------------------|--|
| self | in physics.Shape | The input Shape |
| coll | out number | The programmer-assigned integer collision type |

Returns the integer collision type of the Shape.

Introduced in platform.apiLevel = '2.0'

19.5.4 data

```
obj = physics.Shape:data()
```

| Parameter | Type | Description |
|-----------|------------------|--|
| self | in physics.Shape | The input Shape |
| obj | out Lua object | The programmer-assigned data object assigned to this Shape |

Returns the contents of the programmer data field of the Shape.

Introduced in platform.apiLevel = '2.0'

19.5.5 friction

```
f = physics.Shape:friction()
```

| Parameter | Type | Description |
|-----------|------------------|-----------------|
| self | in physics.Shape | The input Shape |

| Parameter | Type | Description |
|-----------|------------|--|
| f | out number | The coefficient of friction for this Shape |

Returns the friction coefficient of the shape.

Introduced in platform.apiLevel = '2.0'

19.5.6 group

```
g = physics.Shape:group()
```

| Parameter | Type | Description |
|-----------|------------------|---------------------------|
| self | in physics.Shape | The input Shape |
| g | out number | The assigned group number |

Returns the group number of the shape.

Note

The group number is converted to a positive whole number when stored.

Introduced in platform.apiLevel = '2.0'

19.5.7 layers

```
layers = physics.Shape:layers()
```

| Parameter | Type | Description |
|-----------|------------------|--|
| self | in physics.Shape | The input Shape |
| layers | out number | A bitmap of the layers this shape occupies |

Returns the bitmap of layers the shape occupies.

Introduced in platform.apiLevel = '2.0'

19.5.8 rawBB

```
bb = physics.Shape:rawBB()
```

| Parameter | Type | Description |
|-----------|------------------|-------------------------------|
| self | in physics.Shape | The input Shape |
| bb | out physics.BB | The bounding box of the Shape |

Returns the [bounding box](#) of the shape. Only valid after a call to [physics.Shape:BB\(\)](#) or [physics.Space:step\(\)](#).

Introduced in platform.apiLevel = '2.0'

19.5.9 restitution

```
r = physics.Shape:restitution()
```

| Parameter | Type | Description |
|-----------|------------------|------------------------------|
| self | in physics.Shape | The input Shape |
| r | out number | The restitution of the Shape |

Returns the restitution (or elasticity) of the shape.

Introduced in `platform.apiLevel = '2.0'`

19.5.10 sensor

```
s = physics.Shape:sensor()
```

| Parameter | Type | Description |
|-----------|------------------|-------------------------------|
| self | in physics.Shape | The input Shape |
| s | out boolean | True if the Shape is a sensor |

Returns true if the shape is a sensor.

Introduced in `platform.apiLevel = '2.0'`

19.5.11 setCollisionType

```
self = physics.Shape:setCollisionType(collisionType)
```

| Parameter | Type | Description |
|---------------|-------------------|---|
| self | in physics.Shape | The input Shape |
| collisionType | in number | Programmer-defined type of collision |
| self | out physics.Shape | The input Shape is returned as the output |

Assigns a collision type (an integer value of your choosing) to the shape. It is used to determine which handler to call when a collision occurs. Returns **self**.

Introduced in `platform.apiLevel = '2.0'`

19.5.12 setData

```
self = physics.Shape:setData(obj)
```

| Parameter | Type | Description |
|-----------|-------------------|---|
| self | in physics.Shape | The input Shape |
| obj | in Lua object | An object defined by the programmer |
| self | out physics.Shape | The input Shape is returned as the output |

Sets the programmer data field of the Shape. The programmer can store any Lua object in this field. Returns **self**.

Introduced in `platform.apiLevel = '2.0'`

19.5.13 setFriction

```
self = physics.Shape:setFriction(f)
```

| Parameter | Type | Description |
|-----------|-------------------|--|
| self | in physics.Shape | The input Shape |
| f | in number | Coefficient of friction for the surface of the Shape |
| self | out physics.Shape | The input Shape is returned as the output |

Sets the friction coefficient for the shape. Returns **self**.

Note

May not behave as expected for f larger than 1.0 or less than 0.

Introduced in platform.apiLevel = '2.0'

19.5.14 setGroup

```
self = physics.Shape:setGroup(group)
```

| Parameter | Type | Description |
|-----------|-------------------|---|
| self | in physics.Shape | The input Shape |
| group | in number | Group number |
| self | out physics.Shape | The input Shape is returned as the output |

Sets the group (a number defined by the programmer) of the shape. Shapes in the same group do not generate collisions. Returns **self**.

Note

The group number is converted to a positive whole number when stored.

Introduced in platform.apiLevel = '2.0'

19.5.15 setLayers

```
self = physics.Shape:setLayers(layers)
```

| Parameter | Type | Description |
|-----------|-------------------|--|
| self | in physics.Shape | The input Shape |
| layers | in number | A bitmap of integer layer numbers. This implementation permits 32 layers |
| self | out physics.Shape | The input Shape is returned as the output |

Sets the layers that the shape inhabits. Shapes only collide if they are in the same layer. **layers** is an integer bitmap of all the layers that the shape occupies. Returns **self**.

Introduced in platform.apiLevel = '2.0'

19.5.16 setRestitution

```
self = physics.Shape:setRestitution(r)
```

| Parameter | Type | Description |
|-----------|-------------------|---|
| self | in physics.Shape | The input Shape |
| r | in number | The new value for the shape's restitution |
| self | out physics.Shape | The input Shape is returned as the output |

Sets the restitution (or elasticity) of the shape. A value of 0.0 gives no bounce and a value of 1.0 gives a perfect bounce. Returns **self**.

Note

May not behave as expected for r larger than 1.0 or less than 0.

Introduced in platform.apiLevel = '2.0'

19.5.17 setSensor

```
self = physics.Shape:setSensor(bool)
```

| Parameter | Type | Description |
|-----------|-------------------|---|
| self | in physics.Shape | The input Shape |
| bool | in boolean | True if the shape is a sensor |
| self | out physics.Shape | The input Shape is returned as the output |

Determines if the shape is a sensor (true) or not (false). Sensors call [collision handlers](#) but do not generate collisions. Returns `self`.

Introduced in `platform.apiLevel = '2.0'`

19.5.18 setSurfaceV

```
self = physics.Shape:setSurfaceV(vel)
```

| Parameter | Type | Description |
|-----------|-------------------|---|
| self | in physics.Shape | The input Shape |
| velgroup | in physics.Vect | The new vector for the surface velocity |
| self | out physics.Shape | The input Shape is returned as the output |

Sets the surface velocity of the shape. Returns `self`.

Note

The group number is converted to a positive whole number when stored.

Introduced in `platform.apiLevel = '2.0'`

19.5.19 surfaceV

```
sv = physics.Shape:surfaceV()
```

| Parameter | Type | Description |
|-----------|------------------|-----------------------------------|
| self | in physics.Shape | The input Shape |
| sv | out physics.Vect | The surface velocity of the Shape |

Returns the surface velocity [vector](#) of the shape.

Introduced in `platform.apiLevel = '2.0'`

19.6 Circle Shapes

A `CircleShape` is a subclass of [Shape](#). Its type is `TI.cpCircleShape`.

19.6.1 CircleShape

```
cs = physics.CircleShape(body, radius, offset)
```

| Parameter | Type | Description |
|-----------|-------------------------|--|
| body | in physics.Body | A Body or nil |
| radius | in number | The radius of the circle |
| offset | in physics.Vect | The offset of the circle from the Body |
| cs | out physics.CircleShape | A new CircleShape |

Returns a new `CircleShape` with the given [body](#), radius, and offset [vector](#) from the body's center of gravity in body-local coordinates. Specify nil for the body to use the space's static body.

Introduced in platform.apiLevel = '2.0'

19.6.2 offset

```
ovec = physics.CircleShape:offset()
```

| Parameter | Type | Description |
|-----------|------------------------|---------------------------------------|
| self | in physics.CircleShape | The input CircleShape |
| ovec | out physics.Vect | The offset of the shape from the Body |

Returns the offset [vector](#) of the shape from the [body](#)'s center of gravity.

Introduced in platform.apiLevel = '2.0'

19.6.3 radius

```
r = physics.CircleShape:radius()
```

| Parameter | Type | Description |
|-----------|------------------------|-------------------------|
| self | in physics.CircleShape | The input CircleShape |
| r | out number | The radius of the shape |

Returns the radius of the shape.

Introduced in platform.apiLevel = '2.0'

19.7 Polygon Shapes

Polygon shapes are bounded by a set of line segments. The enclosed area of the polygon must be convex and the vertices must be defined in counterclockwise order. Polygon shapes are of type `Tl.cpPolyShape`.

19.7.1 PolyShape

```
ps = physics.PolyShape(body, vertices, offset)
```

| Parameter | Type | Description |
|-----------|-----------------------|--|
| body | in physics.Body | A Body or nil |
| vertices | in {physics.Vect} | The list of vertices that define the boundaries of the polygon defined in counterclockwise order |
| offset | in physics.Vect | The offset of the PolyShape from the Body |
| ps | out physics.PolyShape | A new PolyShape |

Returns a new PolyShape with the given [body](#), table of vertices, and offset from the body's center of gravity. Specify nil for the body to use the space's static body.

Introduced in platform.apiLevel = '2.0'

19.7.2 numVerts

```
nv = physics.PolyShape:numVerts()
```

| Parameter | Type | Description |
|-----------|----------------------|---|
| self | in physics.PolyShape | The input PolyShape |
| nv | out number | The number of vertices in the PolyShape |

Returns the number of vertices in the table of polygon vertices.

Introduced in platform.apiLevel = '2.0'

19.7.3 points

```
points = physics.PolyShape:points()
```

| Parameter | Type | Description |
|-----------|----------------------|---|
| self | in physics.PolyShape | The input PolyShape |
| points | out {physics.Vect} | A table of vertices that define the boundary of the polygon. The vertices are translated to the polygon's current world coordinates |

Returns a copy of the table of vertices defining the bounds of the polygon. The vertices are translated to the polygon's current world coordinates.

Note

When a PolyShape has not been added to a Space, it has no world coordinates. In this case, each vertex returned by physics.PolyShape:points() will have x and y equal to 0.

Introduced in platform.apiLevel = '2.0'

19.7.4 vert

```
v = physics.PolyShape:vert(n)
```

| Parameter | Type | Description |
|-----------|----------------------|---|
| self | in physics.PolyShape | The input PolyShape |
| n | in number | Index of requested vertex inside the table of vertexes describing the polygon |
| v | out physics.Vect | The nth vertex of the polygon. The coordinates of the vector are relative to the shape's Body |

Returns vertex number **n** of the table of vertices defining the bounds of the polygon. If the shape is static, then the vertex values are in world coordinates, otherwise the vertex coordinates are relative to the shape's [body](#). Returns nil if **n** is less than 1 or greater than the number of vertices in the polygon.

Introduced in platform.apiLevel = '2.0'

19.8 Segment Shapes

A segment shape is defined by two end points and a radius. Its type is Tl.cpSegmentShape.

19.8.1 SegmentShape

```
ss = physics.SegmentShape(body, a, b, radius)
```

| Parameter | Type | Description |
|-----------|--------------------------|---|
| body | in physics.Body | A Body or nil |
| a | in physics.Vect | The first end point of the segment. The end point is in coordinates relative to the Body |
| b | in physics.Vect | The second end point of the segment relative to the Body |
| radius | in number | The distance of the border of the segment from the line between the end points of the segment |
| ss | out physics.SegmentShape | A new SegmentShape |

Returns a new SegmentShape with end point vectors **a** and **b**. **radius** defines the thickness of the segment.

Introduced in platform.apiLevel = '2.0'

19.8.2 a

```
avec = physics.SegmentShape:a()
```

| Parameter | Type | Description |
|-----------|-------------------------|------------------------------------|
| self | in physics.SegmentShape | The input SegmentShape |
| avec | out physics.Vect | The first end point of the segment |

Returns the **a** [vector](#) defining one of the end points of the segment.

Introduced in platform.apiLevel = '2.0'

19.8.3 b

```
bvec = physics.SegmentShape:b()
```

| Parameter | Type | Description |
|-----------|-------------------------|-------------------------------------|
| self | in physics.SegmentShape | The input SegmentShape |
| bvec | out physics.Vect | The second end point of the segment |

Returns the **b** [vector](#) defining one of the end points of the segment.

Introduced in platform.apiLevel = '2.0'

19.8.4 normal

```
nvec = physics.SegmentShape:normal()
```

| Parameter | Type | Description |
|-----------|-------------------------|---------------------------------------|
| self | in physics.SegmentShape | The input SegmentShape |
| nvec | out physics.Vect | The unit normal vector of the segment |

Returns the computed [unit normal vector](#) to the segment.

Introduced in platform.apiLevel = '2.0'

19.8.5 radius

```
r = physics.SegmentShape:radius()
```

| Parameter | Type | Description |
|-----------|-------------------------|---------------------------|
| self | in physics.SegmentShape | The input SegmentShape |
| r | out number | The radius of the segment |

Returns the radius of the segment.

Introduced in platform.apiLevel = '2.0'

19.9 Spaces

A physics Space is the basic unit of simulation.

19.9.1 Space

```
s = physics.Space()
```

| Parameter | Type | Description |
|-----------|-------------------|------------------------|
| s | out physics.Space | A new simulation Space |

Returns a new physics simulation Space.

Introduced in `platform.apiLevel = '2.0'`

19.9.2 addBody

```
self = physics.Space:addBody(body)
```

| Parameter | Type | Description |
|-----------|-------------------|---|
| self | in physics.Space | The input simulation Space |
| body | in physics.Body | Adds the Body to the simulation Space |
| self | out physics.Space | The input Space is returned as the output |

Adds a [Body](#) to the Space. Returns `self`.

Introduced in `platform.apiLevel = '2.0'`

19.9.3 addConstraint

```
self = physics.Space:addConstraint(constraint)
```

| Parameter | Type | Description |
|------------|-----------------------|---|
| self | in physics.Space | The input simulation Space |
| constraint | in physics.Constraint | Adds a Constraint to the simulation Space |
| self | out physics.Space | The input Space is returned as the output |

Adds a [Constraint](#) to the Space. Returns `self`.

Introduced in `platform.apiLevel = '2.0'`

19.9.4 addCollisionHandler

```
self = physics.Space:addCollisionHandler(collisionTypeA, collisionTypeB,  
                                         callbacksTable)
```

| Parameter | Type | Description |
|----------------|-----------------------|--|
| self | in physics.Space | The input simulation Space |
| collisionTypeA | in number | Type of first collision |
| collisionTypeB | in number | Type of second collision |
| callbacksTable | in table of functions | A table of functions to call during collision detection and handling |
| self | out physics.Space | The input Space is returned as the output |

Registers a table of callback functions to handle collisions between [shapes](#) of `collisionTypeA` and [shapes](#) of `collisionTypeB`. Listing 19.3 shows the form of the `callbacksTable`.

Listing 19.3: The Form of the Callback Table for `physics.Space:addCollisionHandler()`

```

{
  begin = function(arbiter, space, callbacksTable) ... end,
  preSolve = function(arbiter, space, callbacksTable) ... end,
  postSolve = function(arbiter, space, callbacksTable) ... end,
  separate = function(arbiter, space, callbacksTable) ... end
}

```

If the **begin** handler or **preSolve** handler return false, further collision calculations are bypassed. If they return true, the collision processing proceeds as normal.

It is not necessary to provide handlers for all callback table entries. Default handling will be provided for unspecified handlers.

Returns **self**.

See <http://chipmunk-physics.net/release/Chipmunk-5.x/Chipmunk-5.3.4-Docs/> for an explanation of collision processing and collision handler callbacks.

One important point to note is that these callback handlers must not add or remove [Bodies](#), [Shapes](#), or [Constraints](#) from the Space

See the [post-step callback functions](#) for the right way to remove (or add) objects as the result of a collision.

Introduced in platform.apiLevel = '2.0'

19.9.5 addPostStepCallback

```

self = physics.Space:addPostStepCallback(body|shape|constraint,
                                         function(space, object)
                                         ...end )

```

| Parameter | Type | Description |
|--------------------------------|---|--|
| self | in physics.Space | The input simulation Space |
| body or shape or constraint | in physics.Body or physics.Shape or physics.Constraint | A simulation object that will receive attention after the simulation step |
| function | in function(space, object) | The callback function to run against the simulation object at the end of the simulation step |
| self | out physics.Space | The input Space is returned as the output |

Adds a callback function to be called when the current [step](#) is finished. One callback may be registered per [Body](#), [Shape](#), or [Constraint](#). Only the first callback for a given object is registered. Any attempt to register another callback for the same object is ignored.

Returns **self**.

Introduced in platform.apiLevel = '2.0'

19.9.6 addShape

```

self = physics.Space:addShape(shape)

```

| Parameter | Type | Description |
|-----------|-------------------|---|
| self | in physics.Space | The input simulation Space |
| shape | in physics.Shape | Adds the Shape to the simulation Space |
| self | out physics.Space | The input Space is returned as the output |

Adds a [Shape](#) to the Space. Returns **self**.

Introduced in platform.apiLevel = '2.0'

19.9.7 addStaticShape

```
self = physics.Space:addStaticShape(staticShape)
```

| Parameter | Type | Description |
|-------------|-------------------|---|
| self | in physics.Space | The input simulation Space |
| staticShape | in physics.Shape | Adds the static Shape to the simulation Space |
| self | out physics.Space | The input Space is returned as the output |

Adds a static Shape to the Space. Returns **self**.

Introduced in platform.apiLevel = '2.0'

19.9.8 damping

```
d = physics.Space:damping()
```

| Parameter | Type | Description |
|-----------|------------------|---|
| self | in physics.Space | The input simulation Space |
| d | out number | The amount of damping of the simulation Space |

Introduced in platform.apiLevel = '2.0'

19.9.9 data

```
obj = physics.Space:data()
```

| Parameter | Type | Description |
|-----------|-------------------|---|
| self | in physics.Space | The input simulation Space |
| obj | out Lua object | The object associated with the Space |
| self | out physics.Space | The input Space is returned as the output |

Introduced in platform.apiLevel = '2.0'

19.9.10 elasticIterations

```
iters = physics.Space:elasticIterations()
```

| Parameter | Type | Description |
|-----------|------------------|---|
| self | in physics.Space | The input simulation Space |
| iters | out number | The number of iterations to use in the impulse solver to solve elastic collisions |

Introduced in platform.apiLevel = '2.0'

19.9.11 gravity

```
grav = physics.Space:gravity()
```

| Parameter | Type | Description |
|-----------|------------------|---|
| self | in physics.Space | The input simulation Space |
| grav | out physics.Vect | The gravity force vector applied to all Bodies in the simulation Space. |

Introduced in `platform.apiLevel = '2.0'`

19.9.12 idleSpeedThreshold

```
speed = physics.Space:idleSpeedThreshold()
```

| Parameter | Type | Description |
|-----------|------------------|----------------------------|
| self | in physics.Space | The input simulation Space |
| speed | out number | Threshold speed |

Introduced in `platform.apiLevel = '2.0'`

19.9.13 iterations

```
iters = physics.Space:iterations()
```

| Parameter | Type | Description |
|-----------|------------------|--|
| self | in physics.Space | The input simulation Space |
| iters | out number | The number of iterations the solver takes to update one step of the simulation |

Introduced in `platform.apiLevel = '2.0'`

19.9.14 rehashShape

```
self = physics.Space:rehashShape(shape)
```

| Parameter | Type | Description |
|-----------|-------------------|---|
| self | in physics.Space | The input simulation Space |
| shape | in shape | The shape to rehash |
| self | out physics.Space | The input Space is returned as the output |

Update an individual [static shape](#) that has moved. Returns `self`.

Introduced in `platform.apiLevel = '2.0'`

19.9.15 rehashStatic

```
self = physics.Space:rehashStatic()
```

| Parameter | Type | Description |
|-----------|-------------------|---|
| self | in physics.Space | The input simulation Space |
| self | out physics.Space | The input Space is returned as the output |

Rehashes the shapes in the static spatial hash. You must call this if you move any [static shapes](#) or Chipmunk will not update their collision detection data.

Returns `self`.

Introduced in `platform.apiLevel = '2.0'`

19.9.16 removeBody

```
self = physics.Space:removeBody(body)
```

| Parameter | Type | Description |
|-----------|-------------------|--|
| self | in physics.Space | The input simulation Space |
| body | in physics.Body | A Body to remove from the simulation Space |
| self | out physics.Space | The input Space is returned as the output |

Removes a [Body](#) from the Space. Returns **self**.

Introduced in platform.apiLevel = '2.0'

19.9.17 removeConstraint

```
self = physics.Space.removeConstraint(constraint)
```

| Parameter | Type | Description |
|------------|-----------------------|--|
| self | in physics.Space | The input simulation Space |
| constraint | in physics.Constraint | A Constraint to remove from the simulation Space |
| self | out physics.Space | The input Space is returned as the output |

Removes a [Constraint](#) from the Space. Returns **self**.

Introduced in platform.apiLevel = '2.0'

19.9.18 removeShape

```
self = physics.Space.removeShape(shape)
```

| Parameter | Type | Description |
|-----------|-------------------|---|
| self | in physics.Space | The input simulation Space |
| shape | in physics.Shape | A Shape to remove from the simulation Space |
| self | out physics.Space | The input Space is returned as the output |

Removes a [Shape](#) from the Space. Returns **self**.

Introduced in platform.apiLevel = '2.0'

19.9.19 removeStaticShape

```
physics.Space.removeStaticShape(staticShape)
```

| Parameter | Type | Description |
|-------------|-------------------|--|
| self | in physics.Space | The input simulation Space |
| staticShape | in physics.Shape | A static Shape to remove from the simulation Space |
| self | out physics.Space | The input Space is returned as the output |

Removes a [static Shape](#) from the Space. Returns **self**.

Introduced in platform.apiLevel = '2.0'

19.9.20 resizeActiveHash

```
self = physics.Space.resizeActiveHash(dim, count)
```

| Parameter | Type | Description |
|-----------|-------------------|---|
| self | in physics.Space | The input simulation Space |
| dim | in number | The length of one side of a hash cell. Default is 100 count |
| count | in number | The number of cells in the hash table. Default is 1000 |
| self | out physics.Space | The input Space is returned as the output |

The spatial hash of active [Shapes](#) can be tuned to improve collision detection. **dim** establishes the size of a hash cell (default 100), and **count** sets the number of hash cells (default 1000). **dim** should approximate the side length of a typical Shape. A good rule of thumb is to set **count** to about ten times the number of [Shapes](#) in the space.

.Introduced in platform.apiLevel = '2.0'

19.9.21 resizeStaticHash

```
self = physics.Space:resizeStaticHash(dim, count)
```

| Parameter | Type | Description |
|-----------|-------------------|---|
| self | in physics.Space | The input simulation Space |
| dim | in number | The length of one side of a hash cell. Default is 100 count |
| count | in number | The number of cells in the hash table. Default is 1000 |
| self | out physics.Space | The input Space is returned as the output |

This routine configures the spatial hash of static Shapes. Configure this similarly to `resizeActiveHash` but for [static Shapes](#).

.Introduced in platform.apiLevel = '2.0'

19.9.22 setDamping

Damping drains speed from bodies in the simulation. A value of 0.9 means that each [body](#) will lose 10% of its speed per second. Defaults to 1. This value can be overridden on a per body basis.

```
self = physics.Space:setDamping(d)
```

| Parameter | Type | Description |
|-----------|-------------------|--|
| self | in physics.Space | The input simulation Space |
| d | in number | The new amount of damping for the simulation Space |
| self | out physics.Space | The input Space is returned as the output |

Amount of viscous damping to apply to the Space.

Note

May not behave as expected for `d` larger than 1.0 or less than 0.

.Introduced in platform.apiLevel = '2.0'

19.9.23 setData

```
self = physics.Space:setData(obj)
```

| Parameter | Type | Description |
|-----------|-------------------|---|
| self | in physics.Space | The input simulation Space |
| obj | in Lua object | The data object to be sent |
| self | out physics.Space | The input Space is returned as the output |

The programmer can store any Lua object in this field.

Introduced in platform.apiLevel = '2.0'

19.9.24 setElasticIterations

```
self = physics.Space:setElasticIterations(iters)
```

| Parameter | Type | Description |
|-----------|-------------------|--|
| self | in physics.Space | The input simulation Space |
| iters | in number | The number of iterations to use in the impulse solver to solve elastic collisions. Defaults to 0 |
| self | out physics.Space | The input Space is returned as the output |

Introduced in platform.apiLevel = '2.0'

19.9.25 setGravity

```
self = physics.Space:setGravity(grav)
```

| Parameter | Type | Description |
|-----------|-------------------|--|
| self | in physics.Space | The input simulation Space |
| grav | in physics.Vect | The gravity force vector applied to all Bodies in the simulation Space. Defaults to physics.Vect(0, 0) |
| self | out physics.Space | The input Space is returned as the output |

Global gravity applied to the Space. Can be overridden on a per [body](#) basis by writing custom integration functions

Introduced in platform.apiLevel = '2.0'

19.9.26 setIdleSpeedThreshold

```
self = physics.Space:setIdleSpeedThreshold(speed)
```

| Parameter | Type | Description |
|-----------|-------------------|---|
| self | in physics.Space | The input simulation Space |
| speed | in number | Threshold speed |
| self | out physics.Space | The input Space is returned as the output |

The idleSpeedThreshold is the speed below which a [body](#) is considered to be idle. This value is used to determine when a body can be put to sleep.

Introduced in platform.apiLevel = '2.0'

19.9.27 setIterations

```
self = physics.Space:setIterations(iters)
```

| Parameter | Type | Description |
|-----------|-------------------|--|
| self | in physics.Space | The input simulation Space |
| iters | in number | Number of iterations to refine the accuracy of the solver. Default is 10 |
| self | out physics.Space | The input Space is returned as the output |

This value allows the programmer to control the accuracy of the solver. Default is 10.

Introduced in platform.apiLevel = '2.0'

19.9.28 setSleepTimeThreshold

```
self = physics.Space:setSleepTimeThreshold(sleep)
```

| Parameter | Type | Description |
|-----------|-------------------|--|
| self | in physics.Space | The input simulation Space |
| sleep | in number | The amount of time (seconds) below which time if a Shape has not moved, it is put to sleep |
| self | out physics.Space | The input Space is returned as the output |

Sleep time threshold is used to calculate when a [Body](#) can be put to sleep

Introduced in platform.apiLevel = '2.0'

19.9.29 sleepTimeThreshold

```
sleep = physics.Space:sleepTimeThreshold()
```

| Parameter | Type | Description |
|-----------|------------------|---|
| self | in physics.Space | The input simulation Space |
| sleep | out number | The threshold time used to determine when a Shape can be put to sleep |

Introduced in platform.apiLevel = '2.0'

19.9.30 step

```
self = physics.Space:step(dt)
```

| Parameter | Type | Description |
|-----------|-------------------|--|
| self | in physics.Space | The input simulation Space |
| dt | in number | The length of time (seconds) of one step of the simulation |
| self | out physics.Space | The input Space is returned as the output |

Updates the Space for the given time step **dt**. A xed time step is recommended and increases the efficiency of the contact persistence, requiring an order of magnitude fewer iterations and lower CPU usage.

Returns **self**.

Introduced in platform.apiLevel = '2.0'

19.10 Constraints

All Constraints share common accessors.

| Accessors | Type | Description |
|--------------------------|--------------|--|
| bodyA | physics.Body | The first Body that the Constraint acts on |
| bodyB | physics.Body | The second Body that the Constraint acts on |
| setBiasCoef, biasCoef | number | The fraction of error corrected each step of the simulation. Defaults to 0.1. May not behave as expected for numbers larger than 1.0 or less than 0. |
| setData, data | Lua object | A programmer-defined object |
| impulse | number | Calculated impulse applied by the Constraint in the last simulation step. To convert this to the magnitude of the force, divide by the time step passed to physics.Space:step() |
| setMaxBias, maxBias | number | Maximum speed the Constraint can apply error correction. Defaults to INFINITY |
| setMaxForce, maxForce | number | Magnitude of maximum force the Constraint can use to act on the two Bodies. Defaults to INFINITY |

19.10.1 Damped Rotary Spring

```
spring = physics.DampedRotarySpring(a, b, restAngle,  
stiffness, damping)
```

| Parameter | Type | Description |
|-----------|-----------------------------------|--|
| a | in physics.Body | First Body |
| b | in physics.Body | Second Body |
| restAngle | in number | Relative angle in radians that the Bodies want to maintain |
| stiffness | in number | The spring constant |
| damping | in number | How soft to make the damping of the spring |
| spring | out physics.DampedRotarySpring | New DampedRotarySpring |

Like a damped spring, but works in an angular fashion. **restAngle** is the relative angle in radians that the Bodies want to have, **stiffness** and **damping** work basically the same as on a damped spring.

| Accessors | Type |
|-------------------------|--------|
| setRestAngle, restAngle | number |
| setStiffness, stiffness | number |
| setDamping, damping | number |

Introduced in platform.apiLevel = '2.0'

19.10.2 Damped Spring

```
spring = physics.DampedSpring(a, b, anchr1, anchr2, restLength,
                              stiffness, damping)
```

| Parameter | Type | Description |
|------------|--------------------------|--|
| a | in physics.Body | First Body |
| b | in physics.Body | Second Body |
| anchr1 | in physics.Vect | Anchor point to first Body |
| anchr2 | in physics.Vect | Anchor point to second Body |
| restLength | in number | The distance the spring wants to maintain between its Bodies |
| stiffness | in number | The spring constant |
| damping | in number | How soft to make the damping of the spring |
| spring | out physics.DampedSpring | New DampedSpring |

Defined much like a [SlideJoint](#). **restLength** is the distance the spring wants to be, **stiffness** is the spring constant, and **damping** is how soft to make the damping of the spring.

| Accessors | Type |
|---------------------------|--------------|
| setAnchr1, anchr1 | physics.Vect |
| setAnchr2, anchr2 | physics.Vect |
| setRestLength, restLength | number |
| setStiness, stiness | number |
| setDamping, damping | number |

Introduced in platform.apiLevel = '2.0'

19.10.3 Gear Joint

```
joint = physics.GearJoint(a, b, phase, ratio)
```

| Parameter | Type | Description |
|-----------|-----------------------|---|
| a | in physics.Body | First Body |
| b | in physics.Body | Second Body |
| phase | in number | The initial angular offset in radians of the two Bodies |
| ratio | in number | Ratio of velocities between the two Bodies |
| joint | out physics.GearJoint | New GearJoint |

Keeps the angular velocity ratio of a pair of Bodies constant. ratio is always measured in absolute terms. phase is the initial angular offset of the two [bodies](#).

| Accessors | Type |
|-----------------|--------|
| setPhase, phase | number |
| setRatio, ratio | number |

Introduced in platform.apiLevel = '2.0'

19.10.4 Groove Joint

```
joint = physics.GrooveJoint(a, b, grooveA, grooveB, anchr2)
```

| Parameter | Type | Description |
|-----------|-------------------------|-----------------------------------|
| a | in physics.Body | First Body |
| b | in physics.Body | Second Body |
| grooveA | in physics.Vect | One end point of the groove |
| grooveB | in physics.Vect | The other end point of the groove |
| anchr2 | in physics.Vect | The pivot point of Body b |
| joint | out physics.GrooveJoint | New GrooveJoint |

The groove goes from **grooveA** to **grooveB** on Body **a**, and the pivot is attached to **anchr2** on Body **b**. All coordinates are [body](#) local.

| Accessors | Type |
|---------------------|--------------|
| setAnchr2, anchr2 | physics.Vect |
| setGrooveA, grooveA | physics.Vect |
| setGrooveB, grooveB | physics.Vect |
| grooveN | physics.Vect |

Introduced in platform.apiLevel = '2.0'

19.10.5 Pin Joint

```
joint = physics.PinJoint(a, b, anchr1, anchr2)
```

| Parameter | Type | Description |
|-----------|----------------------|----------------------------|
| a | in physics.Body | First Body |
| b | in physics.Body | Second Body |
| anchr1 | in physics.Vect | The anchor point on Body a |
| anchr2 | in physics.Vect | The anchor point on Body b |
| joint | out physics.PinJoint | New PinJoint |

a and **b** are the two [bodies](#) to connect, and **anchr1** and **anchr2** are the anchor points on those bodies. The distance between the two anchor points is measured when the joint is created. If you want to set a specific distance, use the setter function to override it.

| Accessors | Type |
|-------------------|--------------|
| setAnchr1, anchr1 | physics.Vect |
| setAnchr2, anchr2 | physics.Vect |
| setDist, dist | number |

Introduced in platform.apiLevel = '2.0'

19.10.6 Pivot Joint

```
joint = physics.PivotJoint(a, b, pivot)
joint = physics.PivotJoint(a, b, anchr1, anchr2)
```

| Parameter | Type | Description |
|-----------|------------------------|---------------------------------------|
| a | in physics.Body | First Body |
| b | in physics.Body | Second Body |
| pivot | in physics.Vect | Point of pivot between the two Bodies |
| anchr1 | in physics.Vect | The anchor point on Body a |
| anchr2 | in physics.Vect | The anchor point on Body b |
| joint | out physics.PivotJoint | New PivotJoint |

a and **b** are the two [bodies](#) to connect, and **pivot** is the point in world coordinates of the pivot. Because the pivot location is given in world coordinates, you must have the bodies moved into the correct positions already. Alternatively you can specify the joint based on a pair of anchor points, but make sure you have the bodies in the right place as the joint will fix itself as soon as you start simulating the [Space](#).

| Accessors | Type |
|-------------------|--------------|
| setAnchr1, anchr1 | physics.Vect |
| setAnchr2, anchr2 | physics.Vect |

Introduced in platform.apiLevel = '2.0'

19.10.7 Ratchet Joint

```
joint = physics.RatchetJoint(a, b, phase, ratchet)
```

| Parameter | Type | Description |
|-----------|--------------------------|---|
| a | in physics.Body | First Body |
| b | in physics.Body | Second Body |
| phase | in number | Initial offset in radians |
| ratchet | in number | The distance in radians between clicks of the ratchet |
| joint | out physics.RatchetJoint | New RatchetJoint |

Works like a socket wrench. **ratchet** is the distance between clicks, **phase** is the initial offset to use when deciding where the ratchet angles are.

| Accessors | Type |
|---------------------|--------|
| setAngle, angle | number |
| setphase, phase | number |
| setRatchet, ratchet | number |

Introduced in platform.apiLevel = '2.0'

19.10.8 Rotary Limit Joint

```
joint = physics.RotaryLimitJoint(a, b, min, max)
```

| Parameter | Type | Description |
|-----------|-----------------|-------------|
| a | in physics.Body | First Body |

| Parameter | Type | Description |
|-----------|------------------------------|---|
| b | in physics.Body | Second Body |
| min | in number | The minimum angular distance in radians |
| max | in number | The maximum angular distance in radians |
| joint | out physics.RotaryLimitJoint | New RotaryLimitJoint |

Constrains the relative rotations of two [bodies](#). **min** and **max** are the angular limits in radians. It is implemented so that it is possible for the range to be greater than a full revolution.

| Accessors | Type |
|-------------|--------|
| setMin, min | number |
| setMax, max | number |

Introduced in platform.apiLevel = '2.0'

19.10.9 Simple Motor

```
motor = physics.SimpleMotor(a, b, rate)
```

| Parameter | Type | Description |
|-----------|-------------------------|-------------------------------|
| a | in physics.Body | First Body |
| b | in physics.Body | Second Body |
| rate | in number | The relative angular velocity |
| motor | out physics.SimpleMotor | New SimpleMotor |

Keeps the relative angular velocity of a pair of [bodies](#) constant. **rate** is the desired relative angular velocity.

| Accessors | Type |
|---------------|--------|
| setRate, rate | number |

Introduced in platform.apiLevel = '2.0'

19.10.10 Slide Joints

```
joint = physics.SlideJoint(a, b, anchr1, anchr2, min, max)
```

| Parameter | Type | Description |
|-----------|------------------------|---------------------------------|
| a | in physics.Body | First Body |
| b | in physics.Body | Second Body |
| anchr1 | in physics.Vect | Anchor point to first Body |
| anchr2 | in physics.Vect | Anchor point to second Body |
| min | in number | Minimum distance between Bodies |
| max | in number | Maximum distance between Bodies |
| joint | out physics.SlideJoint | New SlideJoint |

a and **b** are the two [bodies](#) to connect, **anchr1** and **anchr2** are the anchor points on those bodies, and **min** and **max** define the allowed distances of the anchor points.

| Accessors | Type |
|-------------------|--------------|
| setAnchr1, anchr1 | physics.Vect |

| Accessors | Type |
|-------------------|--------------|
| setAnchr2, anchr2 | physics.Vect |
| setMin, min | number |
| setMax, max | number |

Introduced in platform.apiLevel = '2.0'

19.11 Arbiters and Collision Pairs

The Arbiter class encapsulates information about each pair of collisions.

19.11.1

```
count = #physics.Arbiter
```

Returns the number of contact points in this Arbiter.

Introduced in platform.apiLevel = '2.0'

19.11.2 a

```
shape = physics.Arbiter:a()
```

| Parameter | Type | Description |
|-----------|--------------------|---------------------------------------|
| self | in physics.Arbiter | The input Arbiter |
| shape | out physics.Shape | The first Shape in the collision pair |

Returns Shape **a** (the first [shape](#)) in a collision pair.

Introduced in platform.apiLevel = '2.0'

19.11.3 b

```
shape = physics.Arbiter:b()
```

| Parameter | Type | Description |
|-----------|--------------------|--|
| self | in physics.Arbiter | The input Arbiter |
| shape | out physics.Shape | The second Shape in the collision pair |

Returns Shape **b** (the second [shape](#)) in a collision pair.

Introduced in platform.apiLevel = '2.0'

19.11.4 bodies

```
bodyA, bodyB = physics.Arbiter:bodies()
```

| Parameter | Type | Description |
|-----------|--------------------|---------------------------------------|
| self | in physics.Arbiter | The input Arbiter |
| bodyA | out physics.Body | The first Body in the collision pair |
| bodyB | out physics.Body | The second Body in the collision pair |

Returns bodyA and bodyB in the collision pair.

Introduced in platform.apiLevel = '2.0'

19.11.5 depth

```
d = physics.Arbitrator:depth(i)
```

| Parameter | Type | Description |
|-----------|-----------------------|--|
| self | in physics.Arbitrator | The input Arbitrator |
| i | in number | A contact point number |
| d | out number | The penetration depth of the ith contact point |

Returns the penetration depth of the ith contact or nil if i is out of range of the number of contact points.

Introduced in platform.apiLevel = '2.0'

19.11.6 elasticity

```
e = physics.Arbitrator:elasticity()
```

| Parameter | Type | Description |
|-----------|-----------------------|--|
| self | in physics.Arbitrator | The input Arbitrator |
| e | out number | The calculated elasticity of the collision |

Returns the calculated elasticity of this collision pair.

Introduced in platform.apiLevel = '2.0'

19.11.7 friction

```
f = physics.Arbitrator:friction()
```

| Parameter | Type | Description |
|-----------|-----------------------|--|
| self | in physics.Arbitrator | The input Arbitrator |
| f | out number | The calculated friction of the collision |

Returns the calculated friction of this collision pair.

Introduced in platform.apiLevel = '2.0'

19.11.8 impulse

```
ivec = physics.Arbitrator:impulse([friction])
```

| Parameter | Type | Description |
|-----------|-----------------------|---|
| self | in physics.Arbitrator | The input Arbitrator |
| friction | in boolean | If true, the calculated friction is included in the calculation |
| ivec | out physics.Vect | The vector impulse applied to resolve the collision |

Returns the [vector](#) impulse that was applied during this [step](#) to resolve the collision. If **friction** is true (default false), then the calculated friction is taken into account.

Introduced in platform.apiLevel = '2.0'

19.11.9 isFirstContact

```
bool = physics.Arbitrator:isFirstContact()
```

| Parameter | Type | Description |
|-----------|-----------------------|--|
| self | in physics.Arbitrator | The input Arbitrator |
| bool | out boolean | True if this is the first step that the Shapes touched |

Returns true if this is the first [step](#) that the [Shapes](#) touched. This information only persists until a step when the shapes are no longer touching. Once they are no longer touching, this flag is reset.

Introduced in platform.apiLevel = '2.0'

19.11.10 normal

```
nvec = physics.Arbitrator:normal(i)
```

| Parameter | Type | Description |
|-----------|-----------------------|--|
| self | in physics.Arbitrator | The input Arbitrator |
| i | in number | A contact point number |
| nvec | out physics.Vect | Vector normal to the ith contact point |

Returns the collision [normal vector](#) for the ith contact point. Returns nil if i is out of the range of the number of contact points.

Introduced in platform.apiLevel = '2.0'

19.11.11 point

```
pvec = physics.Arbitrator:point(i)
```

| Parameter | Type | Description |
|-----------|-----------------------|---------------------------------------|
| self | in physics.Arbitrator | The input Arbitrator |
| i | in number | A contact point number |
| pvec | out physics.Vect | The position of the ith contact point |

Returns the position of the ith contact point. Returns nil if i is out of the range of the number of contact points.

Introduced in platform.apiLevel = '2.0'

19.11.12 setElasticity

```
self = physics.Arbitrator:setElasticity(e)
```

| Parameter | Type | Description |
|-----------|------------------------|--|
| self | in physics.Arbitrator | The input Arbitrator |
| e | in number | Elasticity of the collision |
| self | out physics.Arbitrator | The input Arbitrator is returned as the output |

Overrides the calculated elasticity of the collision.

Note

May not behave as expected for e larger than 1.0 or less than 0.

Introduced in platform.apiLevel = '2.0'

19.11.13 setFriction

```
self = physics.Arbitrator:setFriction(friction)
```

| Parameter | Type | Description |
|-----------|------------------------|--|
| self | in physics.Arbitrator | The input Arbitrator |
| f | in number | Friction in the collision |
| self | out physics.Arbitrator | The input Arbitrator is returned as the output |

Overrides the calculated friction of the collision.

Note

May not behave as expected for f larger than 1.0 or less than 0.

Introduced in platform.apiLevel = '2.0'

19.11.14 shapes

```
shapeA, shapeB = physics.Arbitrator.shapes()
```

| Parameter | Type | Description |
|-----------|-----------------------|-----------------------------------|
| self | in physics.Arbitrator | The input Arbitrator |
| shapeA | out physics.Shape | The first Shape in the collision |
| shapeB | out physics.Shape | The second Shape in the collision |

Returns shapeA and shapeB in the order they were defined in the [collision handler](#) associated with this Arbitrator.

Introduced in platform.apiLevel = '2.0'

19.11.15 totalImpulse

```
ivec = physics.Arbitrator.totalImpulse()
```

| Parameter | Type | Description |
|-----------|-----------------------|---|
| self | in physics.Arbitrator | The input Arbitrator |
| ivec | out physics.Vect | The vector impulse applied to resolve the collision |

Returns the [vector](#) impulse that was applied during this [step](#) to resolve the collision.

Introduced in platform.apiLevel = '2.0'

19.11.16 totalImpulseWithFriction

```
ivec = physics.Arbitrator.totalImpulseWithFriction()
```

| Parameter | Type | Description |
|-----------|-----------------------|---|
| self | in physics.Arbitrator | The input Arbitrator |
| ivec | out physics.Vect | The vector impulse applied to resolve the collision |

Returns the [vector](#) impulse that was applied during this [step](#) to resolve the collision. The calculated friction is taken into account.

Introduced in platform.apiLevel = '2.0'

19.12 Shape Queries

19.12.1 pointQuery

```
bool = physics.Shape.pointQuery(point)
```

| Parameter | Type | Description |
|-----------|------------------|--|
| self | in physics.Shape | The input Shape |
| point | in physics.Vect | A point |
| bool | out boolean | True if point lies within the bounds of Shape |

Returns true if **point** lies within the [Shape](#).

Introduced in platform.apiLevel = '2.0'

19.12.2 segmentQuery

```
info = physics.Shape:segmentQuery(vecta, vectb)
```

| Parameter | Type | Description |
|-----------|------------------------------|---|
| self | in physics.Shape | The input Shape |
| vecta | in physics.Vect | One end point of the segment |
| vectb | in physics.Vect | The other end point of the segment |
| info | out physics.SegmentQueryInfo | Information about where the segment and Shape intersect. Nil if no intersection |

Checks if the line segment from **vecta** to **vectb** intersects the [Shape](#). Returns a SegmentQueryInfo object with the result of the query or nil if no intersection.

If a segment query starts inside of a shape then the result is somewhat undefined. Circles and polygons will not report a collision with that shape, and segments will report an incorrect point and normal if they do detect a collision with that shape. To get around this deficiency, use a separate point query to determine if the segment query starts inside of a shape.

See the [SegmentQueryInfo](#) methods below for helper routines to convert the results to world coordinates or absolute distance.

Introduced in platform.apiLevel = '2.0'

19.13 Space Queries

19.13.1 pointQuery

```
physics.Space:pointQuery(point, layers, group,
    function(shape) ... end)
```

| Parameter | Type | Description |
|-----------|------------------|---|
| self | in physics.Space | The input Space |
| point | in physics.Vect | A point |
| layers | in number | A bitmap of the layers. Match if shape.layers intersects layers |
| group | in number | The group number to check. Match if Shape is not in group |
| function | function(shape) | A function to call providing each Shape in turn that matches the criteria |

Queries the [Space](#) for all shapes that contain **point** and match **layers** but not in **group**. The **function** is called with each matching [Shape](#). Sensor Shapes are included.

Introduced in platform.apiLevel = '2.0'

19.13.2 pointQueryFirst

```
shape = physics.Space:pointQueryFirst(point, layers, group)
```

| Parameter | Type | Description |
|-----------|------------------|---|
| self | in physics.Space | The input Space |
| point | in physics.Vect | A point |
| layers | in number | A bitmap of the layers. Match if shape.layers intersects layers |
| group | in number | The group number to check. Match if Shape is not in group |

Queries [Space](#) at a point and returns the first [Shape](#) that matches the given **layers** and not in **group**. Returns nil if no Shape was found. Sensor Shapes are ignored.

Introduced in platform.apiLevel = '2.0'

19.13.3 segmentQuery

```
physics.Space.segmentQuery(startvect, stopvect, layers, group,
```

| Parameter | Type | Description |
|-----------|----------------------------|---|
| self | in physics.Shape | The input Shape |
| startvect | in physics.Vect | An end point of the segment |
| stopvect | in physics.Vect | Other end point of the segment |
| layers | in number | A bitmap of the layers. Match if shape.layers inter- sects layers |
| group | in number | The group number to check. Match if object is not in group |
| function | function(shape, t, normal) | A function to call providing each Shape in turn that matches the criteria |

Queries the [Space](#) for all [Shapes](#) that intersect the line segment from **startvect** to **stopvect** and match **layers** and not in **group**. The **function** is called with each matching Shape. Sensor Shapes are included.

The callback function is called with each Shape, proportion of distance along the line segment (a fraction from 0 to 1), and the surface [normal vector](#) of the intersection point of the Shape.

Introduced in platform.apiLevel = '2.0'

19.13.4 segmentQueryFirst

```
info = physics.Space.segmentQueryFirst(startvect, stopvect, layers, group)
```

| Parameter | Type | Description |
|-----------|------------------------------|--|
| self | in physics.Shape | The input Shape |
| startvect | in physics.Vect | An end point of the segment |
| stopvect | in physics.Vect | Other end point of the segment |
| layers | in number | A bitmap of the layers. Match if shape.layers inter- sects layers |
| group | in number | The group number to check. Match if object is not in group |
| info | out physics.SegmentQueryInfo | Information about where the segment and Shape intersect. Nil if no intersection |

Queries [Space](#) along the line segment from **startvect** to **stopvect** and returns the first intersecting [Shape](#) that matches **layers** and not in **group**. Returns a SegmentQueryInfo object with the first Shape that matches the query or nil if no intersection.

Introduced in platform.apiLevel = '2.0'

19.14 SegmentQueryInfo

A SegmentQueryInfo object is a Lua dictionary table with three fields.

| Key | Value |
|-------|---|
| shape | Shape object found in a query. |
| t | Fractional distance (0 .. 1) from the start of the line segment to the intersection of the Shape. |
| n | Surface normal vector of the Shape at the intersection point. |

This object also has the following helper routines that convert information in a SegmentQueryInfo object to world coordinates or an absolute distance along the line segment.

19.14.1 hitDist

```
d = SegmentQueryInfo:hitDist(startvect, stopvect)
```

| Parameter | Type | Description |
|-----------|-----------------------------|--------------------------------|
| self | in physics.SegmentQueryInfo | The input SegmentQueryInfo |
| startvect | in physics.Vect | An end point of the segment |
| stopvect | in physics.Vect | Other end point of the segment |
| d | out physics.Vect | Hit distance |

Returns the absolute distance where the segment first hit the [Shape](#).

Introduced in platform.apiLevel = '2.0'

19.14.2 hitPoint

```
p = SegmentQueryInfo:hitPoint(startvect, stopvect)
```

| Parameter | Type | Description |
|-----------|-----------------------------|--------------------------------|
| self | in physics.SegmentQueryInfo | The input SegmentQueryInfo |
| startvect | in physics.Vect | An end point of the segment |
| stopvect | in physics.Vect | Other end point of the segment |
| p | out physics.Vect | Hit point |

Returns the hit point in world coordinates where the segment between **startvect** and **stopvect** first intersects the [Shape](#).

Introduced in platform.apiLevel = '2.0'

Chapter 20

Bluetooth® Smart Library

The *Bluetooth*® Smart Library enables TI-Nspire™ software running on platforms that support *Bluetooth*® Smart wireless technology to connect to *Bluetooth*® LE devices (Low Energy) supporting the peripheral role. As some of the communication is asynchronous, each asynchronous function providing a result requires a callback to receive responses and events. Responses are the asynchronously provided results values for a request and events are additional state information over a period of time, e.g. the duration of an established connection.

20.1 Bluetooth® LE

The *Bluetooth*® LE Library summarizes all generic functionality related to *Bluetooth*® LE technology offered inside the TI-Nspire™ platform.

20.1.1 addStateListener

```
ble.addStateListener(callback [, object])
```

Registers a *Bluetooth*® LE state-change listener callback. The registration of multiple listener callbacks at the same time is supported. Registered listener callbacks can be removed by calling **removeStateListener**

| Parameter | Type | Description |
|-----------|-------------------|---|
| callback | in function | Callback to receive unsolicited events about <i>Bluetooth</i> ® LE state changes |
| object | in any (optional) | If an object is provided it will be passed as the first parameter to the specified callback function. The object can be of any type except nil . |

Callback Function

```
callback([object,] state)
```

The callback function provided in **addStateListener** will be called for unsolicited *Bluetooth*® LE state changes. This includes switching on/off *Bluetooth*® technology or the OS resetting the *Bluetooth*® stack.

| Parameter | Type | Description |
|-----------|------------------------------|---|
| object | in any (optional) | If an object was provided as a parameter to the function addStateListener , it will be passed as the first parameter to this callback function |
| state | in ble table constant | Please see the following section for details. |

Bluetooth® LE State Constants

The constants described in the following table are part of the **ble** table, e.g. **ble.OFF**.

| Name | Description |
|------------|---|
| ON | <i>Bluetooth</i> ® technology is switched on |
| OFF | <i>Bluetooth</i> ® technology is switched off. This implies that any ongoing scan has been stopped and connected peripherals lose their connection. |

| Name | Description |
|-------------|--|
| RESETTING | The <i>Bluetooth</i> ® stack is resetting. This is an intermittent state and an update will follow. In this state the Lua script should release all <i>Bluetooth</i> ® LE object references as these objects have become invalid and cannot be used anymore. The use of invalidated objects will cause a Lua error. |
| UNSUPPORTED | <i>Bluetooth</i> ® technology is not supported on this platform |

Introduced in `platform.apiLevel = '2.5'`

20.1.2 removeStateListener

```
success = ble.removeStateListener(callback)
```

Removes a registered *Bluetooth*® LE state-change listener callback which was previously registered by calling `addStateListener`.

| Parameter | Type | Description |
|-----------|-------------|---|
| callback | in function | The callback previously registered by calling <code>addStateListener</code> |
| success | out boolean | True if successful, otherwise false if specified listener was never added |

Introduced in `platform.apiLevel = '2.5'`

20.1.3 pack

Note: Applies to pack and unpack.

- This function moved from ble to string.
- This is available at `ble.pack` and `ble.unpack` for apilevels 2.5 and 2.6
- Moved to string.[pack](#) string.[unpack](#) starting apilevel 2.7

Introduced in `platform.apiLevel = '2.5'`

20.1.4 unpack

Note: Applies to pack and unpack.

- This function moved from ble to string.
- This is available at `ble.pack` and `ble.unpack` for apilevels 2.5 and 2.6
- Moved to string.[pack](#) string.[unpack](#) starting apilevel 2.7

Introduced in `platform.apiLevel = '2.5'`

20.1.5 Format Specifier for pack and unpack

The sections contains explanations and additional information to the [Table 20.1](#) . The format specifier `rX`, which means a lower case `r` followed by a number, serves the purpose of skipping bits but does not read or write any data. Therefore the nature of this format is different from any other format and no Lua type is associated to this format. Skipping of bits is only needed if more than one format is used in a row which have all bit alignment.

Bit verses Octet Alignment

An octet is a set of 8 bits which is often also described as a byte. Bit alignment means that the format `"bbn"` will read/write 6 bits inside of 1 octet. These formats read/write a stream of bits. If 8 bits are written, writing just continues, without respect to octet borders. So a nibble (4 bits) can be split across 2 octets, depending on how many bits have been read/written before.

Octet alignment is different. The format `"u12u12"` describes 24 bits of data and could be in theory written into 3 octets if `u12` were bit aligned (`"nnnnnn"`3 octets). But as `u12` is byte aligned the second 12 bit integer will read from/write into a new octet so that 4 octets are used.

Now what does the format "**u12n**" do? An octet-aligned format enforces octet alignment surrounding itself - it reads/writes on the next octet border and enforces a potentially following format on a octet border as well.

Exponent

The *Bluetooth*® LE specification allows exponents for integer types ($1.2 * 10^{\text{exponent}}$). The exponent is not encoded inside the data itself but needs to be used on encoding and decoding of the data. This concept is known as fixed-point number format. If an exponent is allowed for a format but no exponent is specified it defaults to 0 (10⁰). if an exponent is specified read/write of the format will apply the exponent automatically ("**u8e1u8e-1s16e-7**"). Simply add a lower case e followed by a number after the format specifier.

Table 20.1: Format specifier for pack and unpack

| Format Specifier | Data Type Description | Lua Type | Alignment | Exponent |
|------------------|--------------------------------|----------|-----------|----------|
| rX | X number of bits skipped | N/A | bit | N/A |
| b | boolean | boolean | bit | No |
| b2 | 2 bits | number | bit | No |
| n | 4 bits (nibble) | number | bit | No |
| u8 | unsigned integer 8 bits | number | octet | Yes |
| u12 | unsigned integer 12 bits | number | octet | Yes |
| u16 | unsigned integer 16 bits | number | octet | Yes |
| u24 | unsigned integer 24 bits | number | octet | Yes |
| u32 | unsigned integer 32 bits | number | octet | Yes |
| u48 | unsigned integer 48 bits | number | octet | Yes |
| s8 | signed integer 8 bits | number | octet | Yes |
| s12 | signed integer 12 bits | number | octet | Yes |
| s16 | signed integer 16 bits | number | octet | Yes |
| s24 | signed integer 24 bits | number | octet | Yes |
| s32 | signed integer 32 bits | number | octet | Yes |
| s48 | signed integer 48 bits | number | octet | Yes |
| f | IEEE-754 32-bit floating point | number | octet | No |
| fl | IEEE-754 64-bit floating point | number | octet | No |
| S8 | UTF-8 string | string | octet | No |
| S16 | UTF-16 string | string | octet | No |

20.2 Bluetooth® LE Central

20.2.1 startScanning

```
error = bleCentral.startScanning([UUID, ] callback [, object])
```

Scans for *Bluetooth*® LE devices advertising a service with the given service UUID (Universal Unique Identifier) or for *Bluetooth*® LE devices advertising any service if no UUID is provided. Successive calls of **startScanning** automatically stop previous scans. A malformed UUID will cause a Lua error as this is an authoring error and not a run-time error..

| Parameter | Type | Description |
|-----------|---------------------------------------|---|
| UUID | in string (optional) | UUID of the service searched for. The UUID can be provided in 16- or 128-bit format. |
| callback | in function | Callback to receive asynchronously peripherals found, one peripheral at a time |
| object | in any (optional) | If an object is provided it will be passed as the first parameter to the specified callback function. |

| Parameter | Type | Description |
|--------------------|------------------------------|--|
| | | The object can be of any type except <code>nil</code> . |
| <code>error</code> | out string (optional) | If successful <code>nil</code> is returned, an error message otherwise |

Introduced in `platform.apiLevel = '2.5'`

Callback Function

```
callback([object,] peripheral, advertisementData, isConnectable, RSSI)
```

The callback function provided in `startScanning` will be called for every *Bluetooth*® LE device fulfilling the search criteria. The `peripheral` parameter will be a peripheral object representing the found *Bluetooth*® LE device. The same peripheral object might be reported more than once based on peripheral and platform behavior.

| Parameter | Type | Description |
|--------------------------------|-----------------------------|--|
| <code>object</code> | in any (optional) | If an object was provided as a parameter to the function <code>startScanning</code> , it will be passed as the first parameter to this callback function |
| <code>peripheral</code> | in peripheral object | One peripheral found during scanning |
| <code>advertisementData</code> | in table | A table containing the advertisement data of the device.. |
| <code>isConnectable</code> | in boolean | true if the device advertises as connectable, otherwise false |
| <code>RSSI</code> | in Integer value | The received signal strength indicator (RSSI) in dbm. |

Advertisement Data Keys

The constants described in the following table are part of the `ble` table, e.g.

`ble.AD TX POWER LEVEL`.

| Name | Description |
|-----------------------------------|---|
| <code>AD NAME</code> | Same as the device name or a shortened name. Please see subsection 20.3.1 for <code>peripheral.getName()</code> . |
| <code>AD MANUFACTURER DATA</code> | <i>Bluetooth</i> ® technology is switched off. This implies that any A string with the first 2 octets identifying the manufacturer (see Company Identifiers .). The interpretation of any other octet in the string is manufacturer specific. |
| <code>AD SERVICE UUIDS</code> | A list of service UUIDs. This list might be complete or not. |
| <code>AD TX POWER LEVEL</code> | If provided by the device, the sending power level of the device in dBm. Subtracting the RSSI value from the power level can be used to compare the approximate distance of different devices |

Introduced in `platform.apiLevel = '2.5'`

Extended in `platform.apiLevel = '2.6'`

20.2.2 stopScanning

```
bleCentral.stopScanning()
```

Stops scanning for *Bluetooth*® LE devices.

Introduced in `platform.apiLevel = '2.5'`

20.2.3 isScanning

```
bleCentral.isScanning()
```

Returns **true** if a scan for *Bluetooth*® LE devices is ongoing or **false** otherwise.

Introduced in `platform.apiLevel = '2.5'`

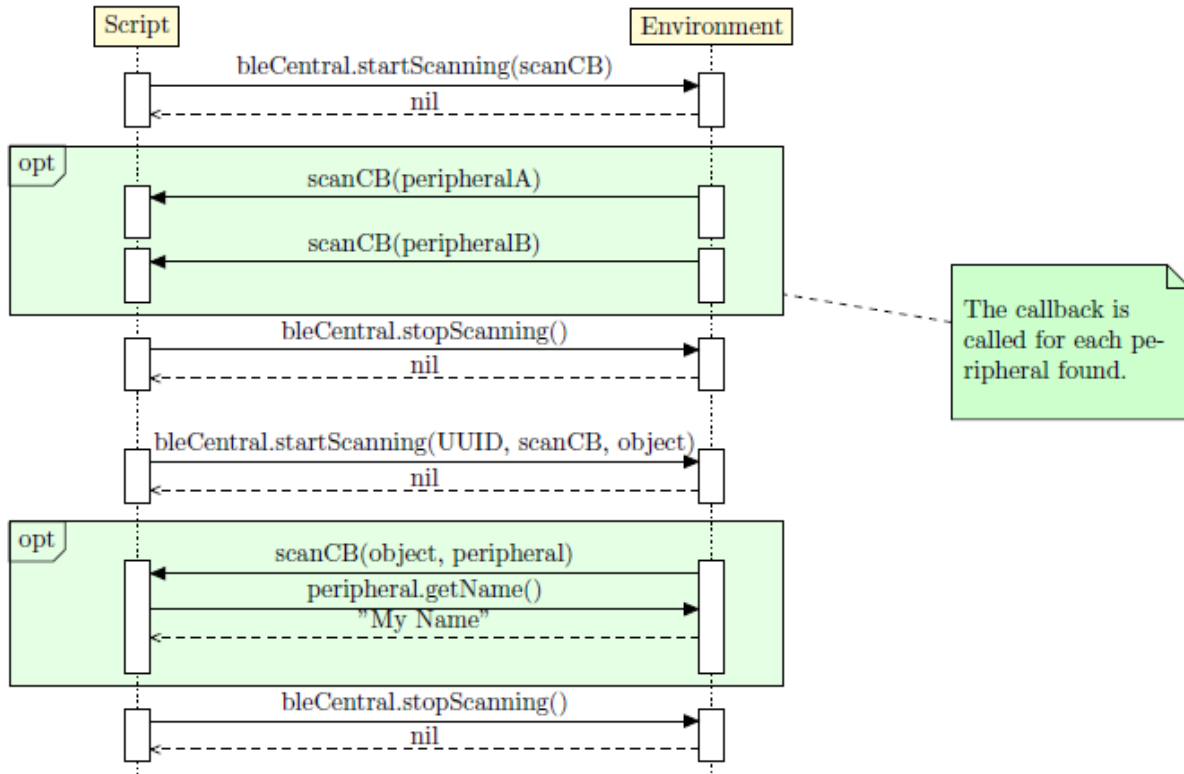


Figure 20.1: Bluetooth® LE Scanning Procedure

20.3 Peripheral Class

20.3.1 getName

```
name = peripheral.getName()
```

Returns the name of the peripheral as a string. The peripheral name is what an Application would typically show to the user. There is no guarantee that two different devices have different names. On the contrary two devices of the same kind and vendor could show the same name until changed by the user. If no name is available at the point of time `nil` will be returned. Changing the name for *Bluetooth*® LE device could be device specific.

Introduced in `platform.apiLevel = '2.5'`

20.3.2 getState

```
state = peripheral.getState()
```

Returns the connection state - disconnected, connecting, connected, disconnecting.

```
function isConnected(peripheral)
    return peripheral:getState() == bleCentral.CONNECTED
end
```

Introduced in `platform.apiLevel = '2.5'`

20.3.3 connect

```
error = peripheral:connect([timeout, ]callback [, object])
```

Requests connection to the *Bluetooth*® LE device represented by the peripheral object. The callback will be called for all events related to the connection state of this peripheral object. An optional timeout can be provided to automatically abort the request after the time specified. After disconnecting or after a failure during the connection procedure, the specified callback will not be referenced anymore.

Caution

Please make sure you disconnect peripheral objects before closing the document. A simple way to do so is by calling `peripheral:disconnect()` from the `on.destroy()` event.

| Parameter | Type | Description |
|-----------|--|--|
| timeout | in number (optional) | If provided, connection request aborts after the specified time in seconds. The specified time can be between 0 and 3600. 0 or less waits forever, anything above 3600 defaults to 3600. |
| callback | in function | Callback to receive asynchronous events about the connection state |
| object | out any (optional) | If an object is provided it will be passed as the first parameter to the specified callback function. The object can be of any type except <code>nil</code> |
| error | out string (optional) | If successful <code>nil</code> is returned, an error message otherwise |

Introduced in `platform.apiLevel = '2.5'`

Extended in `platform.apiLevel = '2.6'`

Callback Function

```
callback([object,] peripheral, event [, error])
```

The callback function provided in `peripheral:connect()` will be called for every event related to the connection state between the *Bluetooth*® LE central and the peripheral. Based on whether an object was provided in the call to `connectPeripheral`, the callback should have three or four parameters.

| Parameter | Type | Description |
|------------|---|--|
| object | in any (optional) | If an object was provided as a parameter to the function <code>startScanning</code> , it will be passed as the first parameter to this callback function |
| peripheral | in peripheral object | A peripheral found during scanning |
| event | in <code>bleCentral</code> table constant | Please see the following section for details |
| error | in string (optional) | If successfully connected or disconnected the parameter will be <code>nil</code> , an error message otherwise (unsuccessful connection or dropped connected) |

Event Constants

The constants described in the following table are part of the `bleCentral` table, e.g. `bleCentral.CONNECTED`.

| Name | Description |
|--------------------------|--|
| CONNECTED | The connection has been successfully established |
| CONNECTING FAILED | The connect procedure failed. A new call of the connect() function is required to retry connecting. |
| DISCONNECTED | The connection has been successfully terminated |

Introduced in `platform.apiLevel = '2.5'`

20.3.4 disconnect

```
peripheral:disconnect()
```

Disconnects the connection with the peripheral object. The callback provided in **connectPeripheral** will be called to confirm completion of the disconnect procedure.

Introduced in `platform.apiLevel = '2.5'`

20.3.5 discoverServices

```
error = peripheral:discoverServices([UUIDs, ] callback [, object])
```

Initiates the services discovery procedure for the peripheral object. The callback will be called once on completion of the procedure. The discovery may complete successfully or fail.

| Parameter | Type | Description |
|-----------|---|--|
| UUID(s) | in strings(s) (optional) | 0 to 10 UUIDs, identifying services to search for. Omit this parameter to search for all services. A UUID can be provided in 16- or 128-bit format. |
| callback | in function | Callback to inform about the completion of the service discovery procedure. A call to getServices() of the peripheral object provides the results. |
| object | out any (optional) | If an object is provided it will be passed as the first parameter to the specified callback function. The object can be of any type except nil . |
| error | out string (optional) | If successful nil is returned, an error message otherwise |

Introduced in `platform.apiLevel = '2.5'`

Callback Function

```
callback([object, ] peripheral [, error])
```

The callback function provided in **discoverServices** will be called once when the services discovery procedure completes. Based on whether an object was provided in the call to **discoverServices**, the callback should have two or three parameters. The error will be **nil** if the procedure completed successfully. Calling **getServices** will retrieve the discovered services.

| Parameter | Type | Description |
|------------|---------------------------------------|---|
| object | in any (optional) | If an object was provided as a parameter to the function discoverServices , it will be passed as the first parameter to this callback function |
| peripheral | in peripheral object | The peripheral object offering the discovered services |
| error | in string (optional) | If successful the parameter will be nil , an error message otherwise |

Introduced in `platform.apiLevel = '2.5'`

20.3.6 getServices

```
table [, error] = peripheral:getServices()
```

Returns a table containing the list of services discovered which can be traversed with the help of the **ipairs** function. An empty table is returned if no services were discovered or if **getServices** gets called before the service discovery procedure completes. In case of an error, **nil** is returned together with an error message.

| Parameter | Type | Description |
|-----------|---------------------------------------|--|
| table | out table | A table containing the discovered services otherwise nil if an error occurred |
| error | out string (optional) | If successful nil is returned, an error message otherwise |

Introduced in `platform.apiLevel = '2.5'`

20.4 Service Class

20.4.1 getUUID

```
UUID = service:getUUID()
```

Returns the UUID of the service as string.

Introduced in `platform.apiLevel = '2.5'`

20.4.2 discoverCharacteristics

```
error = service:discoverCharacteristics([UUIDs, ] callback [, object])
```

Initiates the characteristics discovery procedure for the service object. The callback will be called once on completion of the procedure. The discovery may complete successfully or fail.

| Parameter | Type | Description |
|-----------|--|--|
| UUID(s) | in strings(s) (optional) | 0 to 10 UUIDs, identifying characteristics to search for. Omit this parameter to search for all characteristics. A UUID can be provided in 16- or 128-bit format. |
| callback | in function | Callback to inform about the completion of the characteristics discovery procedure. A call to getCharacteristics() of the service object provides the results. |
| object | in any (optional) | If an object is provided it will be passed as the first parameter to the specified callback function. The object can be of any type except nil |
| error | out string (optional) | If successful nil is returned, an error message otherwise |

Introduced in `platform.apiLevel = '2.5'`

Callback Function

```
callback([object,] service [, error])
```

The callback function provided in **discoverCharacteristics** will be called once when the characteristics discovery procedure completes. Based on whether an object was provided in the call to **discoverCharacteristics**, the callback should have two or three parameters. The error will be **nil** if the procedure completed successfully. Calling **getCharacteristics** will retrieve the discovered characteristics..

| Parameter | Type | Description |
|-----------|----------------------|--|
| object | in any (optional) | If an object was provided as a parameter to the function discoverCharacteristics , it will be passed as the first parameter to this callback function |
| service | in service object | The service object offering the discovered characteristics |
| error | in string (optional) | If successful the parameter will be nil , an error message otherwise |

Introduced in `platform.apiLevel = '2.5'`

20.4.3 getCharacteristics

```
table [, error] = service:getCharacteristics()
```

Returns a table containing the list of characteristics discovered which can be traversed with the help of the **ipairs** function. An empty table is returned if no characteristics were discovered or if **getCharacteristics** gets called before the characteristic discovery procedure completes. In case of an error, **nil** is returned together with an error message. See [discoverCharacteristics \(subsection 20.4.2\)](#).

| Parameter | Type | Description |
|-----------|-----------------------|---|
| table | out table | A table containing the discovered characteristics otherwise nil if an error occurred. |
| error | out string (optional) | error out string (optional) If successful nil is returned, an error message otherwise |

Introduced in `platform.apiLevel = '2.5'`

20.5 Characteristic Class

20.5.1 getUUID

```
UUID = characteristic:getUUID()
```

Returns the UUID of the characteristic as a string.

Introduced in `platform.apiLevel = '2.5'`

20.5.2 setValueUpdateListener

```
characteristic:setValueUpdateListener(callback [, object])
```

Sets or removes the value-update listener callback for read and notification updates. To remove the callback, use **nil** as callback parameter. Once the listener callback is called the result can be retrieved via **getValue()**. This function can be called at any time to update the value update listener callback for a discovered characteristic.

| Parameter | Type | Description |
|-----------|-------------------|--|
| callback | in function | The callback is called once the characteristic value is ready to be retrieved via the getValue() function |
| object | in any (optional) | If a callback is provided, optionally an object can be specified which will be passed as the first parameter to the specified callback function. The object can be of any type except nil . |

Callback Function

```
callback([object,] characteristic [, error])
```

The callback function informs when the characteristic value is ready to be retrieved via the **getValue()** function. The callback should not be used to initiate another read for the same characteristic.

| Parameter | Type | Description |
|----------------|--------------------------|---|
| object | in any (optional) | If an object was provided as a parameter to the function setValueUpdateListener , it will be passed as the first parameter to this callback function |
| characteristic | in characteristic object | The characteristic for which the value can be read |
| error | in string (optional) | If successful the parameter will be nil , an error message otherwise |

Introduced in platform.apiLevel = '2.5'

20.5.3 setWriteCompleteListener

```
characteristic:setWriteCompleteListener(callback [, object])
```

Sets or removes the write-complete listener callback for write requests. To remove the callback, use **nil** as the callback parameter. This callback is only called for write requests and not write commands. The type of the write procedure depends on the boolean value specified when calling **write**. This function can be called at any time to update the write complete listener callback for a discovered characteristic.

| Parameter | Type | Description |
|-----------|-------------------|--|
| callback | in function | Callback to inform about the completion of the write procedure |
| object | in any (optional) | If a callback is provided, optionally an object can be specified which will be passed as the first parameter to the specified callback function. The object can be of any type except nil . |

Callback Function

```
callback([object,] characteristic [, error])
```

The callback function will be called to confirm completion of a write request procedure.

| Parameter | Type | Description |
|----------------|--------------------------|---|
| object | in any (optional) | If an object was provided as a parameter to the function setWriteCompleteListener , it will be passed as the first parameter to this callback function |
| characteristic | in characteristic object | The characteristic for which the value was written |
| error | in string (optional) | If successful the parameter will be nil , an error message otherwise |

Introduced in platform.apiLevel = '2.5'

20.5.4 read

```
error = characteristic:read()
```

Initiates reading the characteristics value. If a listener callback is provided with **setValueUpdateListener** for this characteristic (see [subsection 20.5.2](#)) it will be called once the read operation completes and the result can be retrieved via **getValue()**. There is no guarantee that for every single call to the read function a dedicated call of the callback will happen. The system may decide to combine multiple completions of read requests into a single call of the callback.

| Parameter | Type | Description |
|-----------|-----------------------|--|
| error | out string (optional) | If successful nil is returned, an error message otherwise |

Introduced in platform.apiLevel = '2.5'

20.5.5 setNotify

```
error = characteristic:setNotify(doEnable)
```

Enables or disables continuous notification mode for the value of the characteristic, provided the characteristic of the device supports this feature. If **true** is passed, notifications get enabled, whereas **false** will stop notifications. The notification period, changeability of the notification period and the method of changing the notification period is *Bluetooth*® LE device specific. The peripheral needs to be connected so that a call to **setNotify** can have an effect. If a listener callback is provided with **setValueUpdateListener** for this characteristic (see [subsection 20.5.2](#)) it will be called when a new value can be retrieved via **getValue()**. There is no guarantee that for every single response from the device a dedicated call of the callback will happen. The system may decide to combine multiple as well as completions of read requests into a single call of the callback.

| Parameter | Type | Description |
|-----------|-----------------------|---|
| doEnable | in boolean | true will enable notifications, otherwise false will stop notifications |
| error | out string (optional) | If successful nil is returned, an error message otherwise |

Introduced in platform.apiLevel = '2.5'

20.5.6 getValue

```
value = characteristic:getValue()
```

Returns the Value of the characteristic as string.

Introduced in platform.apiLevel = '2.5'

20.5.7 write

```
error = characteristic:write(data, [isRequest])
```

Sets the **data** string attribute as the new characteristic value. Based on the supported write procedure of the device characteristic, the form of a request or a command might be required or not allowed. If both procedures are supported by the device characteristic, it is an author's choice which procedure to use. There is currently no function to retrieve the characteristic property and it is therefore the author's responsibility to either find out the information about the device characteristic or use trial-and-error. A write request – as opposed to a write command – will send back an information about the success, please see [subsection 20.5.3](#).

| Parameter | Type | Description |
|-----------|-----------------------|---|
| data | in string | Data to be written. Data with zero length is not supported. |
| isRequest | in boolean | If true a write request will be send otherwise a write command |
| error | out string (optional) | If successful nil is returned, an error message otherwise |

Introduced in platform.apiLevel = '2.5'

Chapter 21

Asynchronous Serial Interface

The Asynchronous Serial Interface (ASI) allows Lua authors to interact with the serial ports available on the system where the script is running. The ASI allows, through a serial interface, to perform input and output operations. The ASI is designed to be simple and easy to use within TI-Nspire.

Scripts are able to scan ports available in the system and connect to them. One script can be connected to multiple ports. One port can be connected only to one script. Multiple scripts can be connected to multiple ports within the same document. If more than one script needs to be connected to the same port, it is possible to disconnect from one script and then connect from the other script; for instance, by leveraging `on.loseFocus()` and `on.getFocus()` event handlers.

21.1 require 'asi'

Loads and initializes the ASI library.

21.2 addStateListener

```
error = asi.addStateListener(asiStateCallback [, object])
```

Registers an ASI state-change listener callback. The registration of multiple listener callbacks at the same time is supported. Registered listener callbacks can be removed by calling [removeStateListener](#).

| Parameter | Type | Description |
|------------------|--------------------------|---|
| asiStateCallback | in function | Callback to receive events about ASI state changes. |
| object | in any (optional) | If an object is provided, it will be passed as the first parameter to the specified callback function. The object can be of any type except nil. |
| error | out string (optional) | If successful nil is returned, an error message otherwise. |

Introduced in `platform.apiLevel = '2.7'`.

Callback Function

```
asiStateCallback ([object,] state)
```

The callback function provided in `addStateListener` will be called for ASI state changes.

| Parameter | Type | Description |
|-----------|--------------------------|--|
| object | in any (optional) | If an object was provided as a parameter to the function <code>addStateListener</code> , it will be passed as the first parameter to this callback function. |
| state | in asi table constant | The current ASI state (please see table below). |

ASI State Constants

| Name | Description |
|--------|-------------------------------|
| asi.ON | ASI has started and is ready. |

| Name | Description |
|-----------------|--|
| asi.STARTING | ASI is starting. |
| asi.UNSUPPORTED | ASI is not supported on this platform. |

Introduced in `platform.apiLevel = '2.7'`.

21.3 removeStateListener

```
success = asi.removeStateListener(asiStateCallback)
```

Removes a registered ASI state-change listener callback which was previously registered by calling [addStateListener](#).

| Parameter | Type | Description |
|------------------|-------------|--|
| asiStateCallback | in function | The callback previously registered by calling addStateListener . |
| success | out boolean | If successful true is returned; false if the specified listener was never added. |

Introduced in `platform.apiLevel = '2.7'`.

21.4 isScanning

```
asi.isScanning()
```

Returns true if a scan for ASI ports is ongoing or false otherwise.

Introduced in `platform.apiLevel = '2.7'`.

21.5 startScanning

```
error = asi.startScanning(portFoundCallback [, object])
```

Scans for ASI ports. A second call to `startScanning` while already scanning does not reset the process. If a rescan is desired, call [stopScanning](#) first and then `startScanning` to reset the process.

| Parameter | Type | Description |
|-------------------|-----------------------|--|
| portFoundCallback | in function | Callback to receive ports found, one call per port. |
| object | in any (optional) | If an object is provided, it will be passed as the first parameter to the specified callback function. The object can be of any type except nil. |
| error | out string (optional) | If successful nil is returned, an error message otherwise. |

Introduced in `platform.apiLevel = '2.7'`.

Callback Function

```
portFoundCallback ([object,] port)
```

The callback function provided in `startScanning` will be called for every ASI port found. The port parameter will be a port object representing the port interface found. One call per port found. Ports may be present at the moment of calling `asi.startScanning()` or later added while scanning. Ports are reported only once between `startScanning/stopScanning` cycles.

| Parameter | Type | Description |
|-----------|-------------------|---|
| object | in any (optional) | If an object was provided as a parameter to the function <code>startScanning</code> , it will be passed as the first parameter to this callback function. |

| Parameter | Type | Description |
|-----------|----------------|---------------------------------|
| port | in port object | One port found during scanning. |

Introduced in platform.apiLevel = '2.7'.

21.6 stopScanning

```
asi.stopScanning()
```

Stops scanning for ASI ports. Also resets the list of reported ports. Calling asi.startScanning() again will report all available ports once more.

Introduced in platform.apiLevel = '2.7'.

21.7 Port Class

21.7.1 getName

```
name = port.getName()
```

Returns the name of the port as a string, as given by the platform. Typical examples are the following:

| Platform | Port name |
|--------------|---------------|
| TI-Nspire CX | COM1 |
| | COM2 |
| Windows | COM1 |
| | COM9 |
| | COM12 |
| MacOS | usbmodem14121 |
| | usbmodem00001 |

Introduced in platform.apiLevel = '2.7'.

21.7.2 getIdentifier

```
identifier = port.getIdentifier()
```

Returns the identifier associated to the port as a string, as given by the platform. Typical examples are the following:

| Platform | Port name |
|--------------|-----------------------|
| TI-Nspire CX | COM1 |
| Windows | COM1 |
| | COM9 |
| Mac | /dev/cu.usbmodem14121 |
| | /dev/cu.usbmodem00001 |

Introduced in platform.apiLevel = '2.7'.

21.7.3 getState

```
state = port.getState()
```

Returns the current state of the port as a constant from the asi table.

| State | Description |
|-------------------|---|
| asi.DISCONNECTED | Port is disconnected. |
| asi.CONNECTING | Port is connecting. |
| asi.CONNECTED | Port is connected. |
| asi.DISCONNECTING | Port is disconnecting. |
| asi.INVALID | Port is invalid or no longer present in the system. |

Introduced in `platform.apiLevel = '2.7'`.

21.7.4 setBaudRate

```
self = port:setBaudRate(newBaudRate)
```

Sets the baud rate for the connection. By default connections are established at 115200 bauds. If a different value is desired, the new baud rate must be set before establishing a connecting to the port. Returns self.

| Parameter | Type | Description |
|-------------|-----------------|---|
| port | in port object | The port to modify. |
| newBaudRate | in number | The new valid baud rate (please see table below). |
| self | out port object | The input port is returned as the output. |

Baud Rate Constants

Valid baud rates can be set by using either the asi constant or their numeric value.

| Constant | Value# |
|-----------------------|--------------|
| asi.BAUD_RATE_9600 | 9600 bauds |
| asi.BAUD_RATE_115200 | 115200 bauds |
| asi.BAUD_RATE_DEFAULT | 115200 bauds |

21.7.5 connect

```
error = port:connect(connectionCallback[, object])
```

Sends an asynchronous request for connection to the port. When the request is processed, the result is reported to the specified callback.

| Parameter | Type | Description |
|--------------------|-----------------------|--|
| port | in port object | The port to connect to. |
| connectionCallback | in function | Callback to receive connection events. |
| object | in any (optional) | If an object is provided, it will be passed as the first parameter to the specified callback function. The object can be of any type except nil. |
| error | out string (optional) | If successful nil is returned, an error message otherwise. |

Introduced in `platform.apiLevel = '2.7'`.

Callback Function

```
connectionCallback ([object,] port, event[, error])
```

The callback function provided in connect will be called when the state of the connection to the port changes.

| Parameter | Type | Description |
|-----------|-----------------------|---|
| object | in any (optional) | If an object was provided as a parameter to the function connect, it will be passed as the first parameter to this callback function. |
| port | in port object | The port requesting to connect or disconnect. |
| event | in asi table constant | The connection event (please see table below). |
| error | in string (optional) | If successfully connected or disconnected the parameter will be nil, an error message otherwise. |

Event Constants

| Connection event | Description |
|-----------------------|---|
| asi.CONNECTED | The connection was successful and the port is ready for input/output operations. |
| asi.CONNECTING_FAILED | The connection failed. An error message is received. |
| asi.DISCONNECTED | The port has been disconnected. An error message is received if the port has been removed from the system. |

Introduced in `platform.apiLevel = '2.7'`.

21.7.6 disconnect

```
port.disconnect()
```

Sends an asynchronous request for disconnection from the port. The result will be notified at the callback provided at `port.connect()`.

Introduced in `platform.apiLevel = '2.7'`.

21.7.7 setWriteListener

```
self = port.setWriteListener(writeCallback[, object])
```

Registers a callback for write-complete notifications. The callback is called after a write request. Returns self.

| Parameter | Type | Description |
|---------------|-------------------|--|
| port | in port object | The port to modify. |
| writeCallback | in function | Callback to receive write-complete notifications. |
| object | in any (optional) | If an object is provided, it will be passed as the first parameter to the specified callback function. The object can be of any type except nil. |
| self | out port object | The input port is returned as the output. |

Introduced in `platform.apiLevel = '2.7'`.

Callback Function

```
writeCallback([object , ]port[, error])
```

This callback is called when a write request has been completed. A string is passed in case an error occurred while processing the write request.

| Parameter | Type | Description |
|-----------|----------------------|--|
| object | in any (optional) | If an object was provided as a parameter to the function <code>setWriteListener</code> , it will be passed as the first parameter to this callback function. |
| port | in port object | The port where the data was sent to be written. |
| error | in string (optional) | An error message if an error occurred while processing the write request; nil otherwise. |

21.7.8 write

```
error = port.write(writeData)
```

Sends an asynchronous request for a write operation. When the request is serviced, the `writeCallback` is called to confirm completion, if previously specified with [setWriteListener](#). Returns error if an error occurred.

| Parameter | Type | Description |
|-----------|-----------------------|--|
| port | in port object | The port to write to. |
| writeData | in string | Data to be written. Data with zero length is not supported. |
| error | out string (optional) | If the request is successfully queued up, nil is returned; an error message otherwise. |

Introduced in `platform.apiLevel = '2.7'`.

21.7.9 setReadListener

```
self = port.setReadListener(readCallback[, object])
```

Registers a callback for read notifications. The callback is called after a read request. Returns self.

| Parameter | Type | Description |
|--------------|-------------------|--|
| port | in port object | The port to modify. |
| readCallback | in function | Callback to receive read notifications. The actual value read can be retrieved with <code>port.getValue()</code> . |
| object | in any (optional) | If an object is provided, it will be passed as the first parameter to the specified callback function. The object can be of any type except nil. |
| self | out port object | The input port is returned as the output. |

Introduced in `platform.apiLevel = '2.7'`.

Callback Function

```
readCallback([object , ]port[, error])
```

This callback is called when a read request has been completed. A string is passed in case an error occurred while processing the read request. The actual value read can be retrieved with `port.getValue()`.

| Parameter | Type | Description |
|-----------|-------------------|---|
| object | in any (optional) | If an object was provided as a parameter to the function <code>setReadListener</code> , it will be passed as the first parameter to this callback function. |
| port | in port | The port from where the data was requested to be read. |

| Parameter | Type | Description |
|-----------|----------------------|---|
| | object | |
| error | in string (optional) | An error message if an error occurred while processing the read request; nil otherwise. |

Introduced in `platform.apiLevel = '2.7'`.

21.7.10 `setReadTimeout`

```
self = port:setReadTimeout(newTimeout)
```

Sets the maximum amount of time that the platform should wait for the first byte. This affects the behavior of `read()`. By default the timeout is 1000 milliseconds (1 second). Returns `self`.

| Parameter | Type | Description |
|------------|-----------------|--|
| port | in port object | The port to modify. |
| newTimeout | in number | The read timeout in milliseconds. Must be in the interval [30..3000], either as a numeric value or as a constant (please see table below). |
| self | out port object | The input port is returned as the output. |

Timeout Constant

| Constant | Value# |
|---------------------------------------|--------|
| <code>asi.READ_TIMEOUT_DEFAULT</code> | 1000 |

21.7.11 `read`

```
error = port:read([bytesToRead])
```

Sends an asynchronous request for a read operation. When the request is serviced, the `readCallback` is called to confirm completion, if previously specified with [setReadListener](#). Returns `error` if an error occurred.

| Parameter | Type | Description |
|-------------|-----------------------|--|
| port | in port object | The port to read from. |
| bytesToRead | in number | Amount of bytes to read. Must be in the interval [1..1024]. By default 1024 is used if no amount is specified. |
| error | out string (optional) | If request is successfully queued up, nil is returned; an error message otherwise. |

Introduced in `platform.apiLevel = '2.7'`.

21.7.12 `getValue`

```
value = port:getValue()
```

Retrieves the last data read, as string.

Introduced in `platform.apiLevel = '2.7'`.

Appendix A

Script Compatibility

This Appendix summarizes aspects about different types of compatibility issues and concepts for Lua scripts inside the TI-Nspire™ platform. It supports authoring documents for a mixed environment of TI-Nspire™ software releases and different platforms. Authoring scripts for a higher API level than supported inside a current script development environment is detailed in [section A.2](#).

A.1 Backward and Forward Compatibility

There are two compatibility concepts implemented in the TI-Nspire™ platform. The following sections describe these concepts and their interaction. Understanding both is essential to author documents able to run in environments with mixed TI-Nspire™ software releases. If this is not desired, you can skip section A.1 and continue reading [section A.2](#).

A.1.1 Document Compatibility

This is an old concept of the TI-Nspire™ platform. For every document there are two different TI-Nspire™ release values — the release where the document was “last saved” and a “minimum requested” release. Any TI-Nspire™ release with a lower release number than the “minimum requested” release, blocks opening the document. If the TI-Nspire™ release is at least the “last saved” release level, the document will open without warning.

This concept has been recently enhanced. The “minimum requested” release is now determined dynamically based on the content. This allows a lower minimum release; however, changing the document content may raise the “minimum requested” release dynamically.

Script authors interested in backwards compatibility of scripts need to understand that changing non-scripting content inside the same document as the script might modify the “minimum release.” Currently, there is no better support for the script writer to understand what the “minimum requested” releases is other than opening the document manually with multiple releases of the TI-Nspire™ software.

If the document contains only scripts, the rule is simple. Documents will open, but scripts may fail if the used API level is not supported. The earliest software able to open documents containing scripts is TI-Nspire™ software version 3.1. As an exception, the 3.1 software release only opens documents if all contained scripts are of **platform.apiLevel = '1.0'**

A.1.2 Scripting Compatibility

Scripts written for the TI-Nspire™ platform are by default forward compatible on the particular platforms the script was designed and tested for (platform compatibility will be discussed in [section A.3](#)). The key component to ensure forward compatibility is the API level concept. The API level specifies the scripting interface of a particular TI-Nspire™ software release. The mapping between the software release and its highest supported API level is shown in [Table A.1](#). The highest supported, or current API level of a software release used to create a script, is the default API level for scripts when initially authored. The API level can be changed manually by the author at any time.

Backwards compatibility of scripts can be reached by requesting the API level of the oldest TI-Nspire™ software release that is targeted to run the script. To support software release 3.1, the script would request **platform.apiLevel = '1.0'**.

The requested API level of the script is not guaranteed, as an older TI-Nspire™ software version running the script may not support this API level. In addition, requesting an API level that does not exist or is not supported in the used TI-Nspire™ software version will result to the highest supported API level, but not higher than the requested API level. If the script requires a minimum API level to run successful, it might be the simplest solution to prevent the script from executing. This can be archived via the File Menu in the Script Editor.

As an exception to the outlined API level behavior, requesting an API level below 1.0 will result in the current API level of the software release. Please see [Table A.1](#) for more details. Please see [section A.2](#) for a useful example of requesting an API level that is not supported.

A.2 Creating Scripts for a Future Software Release

There might be times when a new version of TI-Nspire™ software with a higher API Level is released, but it does not contain a development environment. In this case, the new functions of the higher API level must be used conditionally at run time. In addition, the authoring process might become a two-step approach. After first authoring the script and saving the document on the authoring platform, it might be that TI-Nspire™ software marks the document dirty when opened first on the target API level platform. When this happens please save the script on the target platform. Saving the document in such a case will be the second step of the authoring process. Once this second step is completed for a specific document it will be usually not be requested again.

An example of how this can be accomplished for the touch library when developing with OS version 3.2 is shown in [Listing A.1](#). This Lua snippet should be the first section in the script. The touch library is not defined in `platform.apiLevel = '2.0'` but in all future releases.

Table A.1: Mapping between API level and TI-Nspire™ software version

| API Level | Software Version | Comment |
|-----------|------------------|--|
| '1.0' | 3.1 | Initial release supporting Lua scripting. |
| '2.0' | 3.2 | Major update containing physics and many other new binding. |
| '2.2' | 3.4 | Introduction of low-level support for touch platforms. |
| '2.3' | 3.6 | Image resources. |
| '2.4' | 3.7 | Support for background color and painted rectangle. |
| '2.5' | 3.11 | Bluetooth LE. |
| '2.6' | 4.1 | Bluetooth LE added APIs - Advertisement data, RSSI, timeout for connect procedure. |
| '2.7' | 4.2 | Asynchronous Serial Interface. |

Listing A.1: Authoring for a Future Software Release for the Example of Touch

```
platform.apiLevel = '2.2'
iftouch then
  if not touch.enabled then
    functiontouch.enabled() return true end
    functiontouch.isKeyboardAvailable() return true end
  end
else
  touch = {}
  functiontouch.enable() returnfalseend
end
```

A.3 Platform Compatibility

A script author usually prefers to write scripts that are platform independent. Unfortunately this is not true for every feature supported by all platforms. [Table A.2](#) shows the major differences. It is the script authors choice to avoid them, use them on selected platforms only, or try to achieve a seamless user experience across all platforms. In the latter case, authors should test scripts on all platforms.

Table A.2: Overview about platform incompatibilities

| Feature | Desktop | Handheld | Touch Platform |
|----------------|--|--|----------------|
| on.grabDown | Supported, (x, y) == (0, 0) | Supported, same as Desktop if no mouse visible | Not supported |
| on.grabUp | | | |
| on.returnKey() | Not supported | Supported | Supported |
| Context Menu | on.contextMenu() on.rightMouseDown() on.rightMouseUp() | on.contextMenu() | Not supported |

Appendix B

Deprecated API Functions and API Behavior

B.1 Image Library

Before `platform.apiLevel = '2.3'`, images were encoded as strings within the script itself. Only the TI-Nspire™ Script Editor of the software version 3.2 supports authoring images encoded as strings inside the script itself.

The following provides details about the encoding.

The header consists of 20 bytes of data arranged as presented in the following table. All elds are little endian integers.

| Offset | Width (bytes) | Contents |
|--------|---------------|---|
| 0 | 4 | Pixel width of image |
| 4 | 4 | Pixel height of image |
| 8 | 1 | Image alignment (0) |
| 9 | 1 | Flags (0) |
| 10 | 2 | Pad (0) |
| 12 | 4 | The number of bytes between successive raster lines |
| 16 | 2 | The number of bits per pixel (16) |
| 18 | 2 | Planes per bit (1) |

The image pixel data immediately follows the header. Pixels are arranged in rows. Each pixel is a little endian 16-bit integer with ve bits for each color red, green, and blue. The top bit determines if the pixel is drawn. If it is zero (0), the pixel is not drawn. If it is one (1), the pixel is drawn in the RGB color of the remaining 15 bits.

0x8000 is black, 0x801F is blue, 0x83E0 is green, 0xFC00 is red, and 0xFFFF is white.

B.2 Platform Library

B.2.1 gc

```
platform.gc()
```

This function has been replaced by `platform.withGC()`, but if you want to author or modify scripts with `platform.apiLevel = '1.0'` you still need this function.

This graphics context should not be used for drawing purposes because it is not guaranteed to be associated with a window.

[Listing B.1](#) shows an example of using the static graphics context to get the string width and height.

Listing B.1: Use of the static GC in `platform.apiLevel = '1.0'`

```
local gc = platform.gc()
gc:setFont('serif', 'r', 10)
local width = gc:getStringWidth(a_string)
local height = gc:getStringHeight(a_string)
```

Introduced in `platform.apiLevel = '1.0'`

Removed in `platform.apiLevel = '2.0'`

B.3 Platform Library

B.3.1 drawString Vertical Alignment

```
gc:drawString("text", x, y [, verticalalignment])
```

Prior to **platform.apiLevel = '2.3'**, “none” was used to specify unspecified vertical alignment. The vertical alignment “none” has been deprecated. Specifying no alignment defaults to “top” and so does “none”.

Introduced in platform.apiLevel = '1.0'

Extended in platform.apiLevel = '2.3'

B.4 Requested API Level

Prior to TI-Nspire™ software version 3.6 (**platform.apiLevel = '2.3'**), requesting a non-supported API level resulted in the highest API level supported by the TI-Nspire™ software version used to run the script. This behavior has been revised. See [section 14.1](#) for details about the new revised behavior.

Introduced in platform.apiLevel = '2.0'

Extended in platform.apiLevel = '2.3'

Index

2

| | |
|-------------------------------|----|
| 2D Editor Library | 6 |
| createChemBox | 6 |
| createMathBox | 7 |
| getExpressions | 7 |
| getExpressionsSelection | 7 |
| getText | 7 |
| hasFocus | 8 |
| isVisible | 8 |
| move | 8 |
| newRichText | 6 |
| registerFilter | 8 |
| resize | 9 |
| setBorder | 9 |
| setBorderColor | 9 |
| setColorable | 9 |
| setDisable2DinRT | 9 |
| setExpression | 9 |
| setFocus | 10 |
| setFontSize | 10 |
| setMainFont | 10 |
| setReadOnly | 11 |
| setSelectable | 11 |
| setSizeChangeListener | 11 |
| setText | 11 |
| setTextChangeListener | 12 |
| setTextColor | 12 |
| setVisible | 12 |
| setWordWrapWidth | 12 |

A

| | |
|--|---------|
| Appendix A | |
| Backward and Forward Compatibility | 121 |
| Creating Scripts for a Future Software Release | 122 |
| Document Compatibility | 121 |
| Platform Compatibility | 122 |
| Script Compatibility | 121 |
| Scripting Compatibility | 121 |
| Appendix B | |
| Deprecated API Functions and API Behavior | 123 |
| Image Library | 123 |
| Platform Library | 123-124 |
| drawStringVerticleAlignment | 124 |
| gc | 123 |
| Requested API Level | 124 |
| ASI Library | |
| addStateListener | 114-115 |
| Asynchronous Serial Interface | 114 |
| isScanning | 115 |
| Port Class | 116 |
| connect | 117 |
| disconnect | 118 |
| getIdentifier | 116 |
| getName | 116 |
| getState | 116 |
| getValue | 120 |

| | |
|------------------------|-----|
| read | 120 |
| setBaudRate | 117 |
| setReadListener | 119 |
| setReadTimeout | 120 |
| setWriteListener | 118 |
| write | 119 |
| require 'asi' | 114 |
| startScanning | 115 |
| stopScanning | 116 |

B

| | |
|--|-----|
| Bluetooth® Smart Library | 103 |
| Bluetooth® LE | 103 |
| addStateListener | 103 |
| Format Specifier for pack and unpack | 104 |
| pack | 104 |
| removeStateListener | 104 |
| startScanning | 105 |
| unpack | 104 |
| Bluetooth® LE Central | 105 |
| isScanning | 107 |
| stopScanning | 106 |
| Characteristic Class | 111 |
| getUUID | 111 |
| getValue | 113 |
| read | 112 |
| setNotify | 113 |
| setValueUpdatedListener | 111 |
| setWriteCompleteListener | 112 |
| write | 113 |
| Peripheral Class | 107 |
| connect | 108 |
| disconnect | 109 |
| discoverServices | 109 |
| getName | 107 |
| getServices | 110 |
| getState | 107 |
| Service Class | 110 |
| discoverCharacteristics | 110 |
| getCharacteristics | 111 |
| getUUID | 110 |

C

| | |
|-------------------------|----|
| Class Library | 13 |
| class | 13 |
| Clipboard Library | 14 |
| addText | 14 |
| getText | 14 |
| Cursor Library | 15 |
| hide | 17 |
| set | 15 |
| show | 17 |

D

| | |
|------------------------|----|
| Document Library | 18 |
| markChanged | 18 |

E

| | |
|----------------------|----|
| Event Handling | 19 |
| activate | 20 |
| arrowDown | 20 |
| arrowKey | 20 |

| | | | |
|-------------------|----|------------------------------|----|
| arrowLeft | 21 | new | 32 |
| arrowRight | 21 | rotate | 32 |
| arrowUp | 21 | width | 33 |
| backspaceKey | 21 | | |
| backTabKey | 21 | L | |
| charIn | 21 | List of Figures | iv |
| clearKey | 22 | List of Tables | i |
| construction | 22 | Listings List | ii |
| contextMenu | 22 | Locale Library | 34 |
| copy | 22 | name | 34 |
| create | 22 | | |
| createMathBox | 22 | M | |
| cut | 23 | Math Library Extension | 35 |
| deactivate | 23 | eval | 35 |
| deleteKey | 23 | evalStr | 36 |
| destroy | 23 | getEvalSettings | 36 |
| enterKey | 23 | setEvalSettings | 37 |
| escapeKey | 23 | Module Library | 39 |
| getFocus | 23 | | |
| getSymbolList | 24 | P | |
| grabDown | 24 | Physics Library | 51 |
| grabUp | 24 | Arbiters and Collision Pairs | 96 |
| help | 24 | # | 96 |
| keyboardDown | 25 | a | 96 |
| keyboardUp | 25 | b | 96 |
| loseFocus | 25 | bodies | 96 |
| mouseDown | 25 | depth | 97 |
| mouseMove | 25 | elasticity | 97 |
| mouseUp | 25 | friction | 97 |
| paint | 26 | impulse | 97 |
| paste | 26 | isFirstContact | 97 |
| propertiesChanged | 26 | normal | 98 |
| resize | 26 | point | 98 |
| restore | 26 | setElasticity | 98 |
| returnKey | 27 | setFriction | 98 |
| rightMouseDown | 27 | shapes | 99 |
| rightMouseUp | 27 | totalImpulse | 99 |
| save | 27 | totalImpulseWithFriction | 99 |
| tabKey | 28 | Bodies | 64 |
| timer | 28 | activate | 64 |
| varChange | 28 | angle | 65 |
| | | angVel | 65 |
| G | | applyForce | 65 |
| Graphics Library | 29 | applyImpulse | 65 |
| clipRect | 29 | Body | 64 |
| drawArc | 29 | data | 66 |
| drawImage | 29 | force | 66 |
| drawLine | 30 | isRogue | 66 |
| drawPolyLine | 30 | isSleeping | 66 |
| drawRect | 30 | kineticEnergy | 67 |
| drawString | 30 | local2World | 67 |
| fillArc | 30 | mass | 67 |
| fillPolygon | 30 | moment | 67 |
| fillRect | 31 | pos | 67 |
| getStringHeight | 31 | resetForces | 68 |
| getStringWidth | 31 | rot | 68 |
| setColorRGB | 31 | setAngle | 68 |
| setFont | 31 | setAngVel | 68 |
| setPen | 31 | setData | 69 |
| | | setForce | 69 |
| | | setMass | 69 |
| | | setMoment | 70 |
| | | setPos | 70 |
| I | | | |
| Image Library | 32 | | |
| copy | 32 | | |
| height | 32 | | |

| | | | |
|------------------------|----|-----------------------|-----|
| setPositionFunc | 70 | SegmentQueryInfo | 102 |
| setTorque | 70 | hitDist | 102 |
| setVel | 71 | hitPoint | 102 |
| setVelocityFunc | 71 | Shape Queries | 99 |
| setVLimit | 71 | pointQuery | 99 |
| setWLimit | 72 | segmentQuery | 100 |
| sleep | 72 | Shapes | 74 |
| sleepWithGroup | 72 | BB | 75 |
| torque | 73 | Bdata | 75 |
| updatePosition | 73 | body | 75 |
| updateVelocity | 73 | collisionType | 75 |
| vel | 74 | friction | 75 |
| vLimit | 74 | group | 76 |
| wLimit | 74 | layers | 76 |
| world2Local | 74 | rawBB | 76 |
| Bounding Boxes | 60 | restitution | 76 |
| b | 60 | sensor | 77 |
| BB | 60 | setCollisionType | 77 |
| clampVect | 61 | setData | 77 |
| containsBB | 61 | setFriction | 77 |
| containsVect | 61 | setGroup | 78 |
| expand | 61 | setLayers | 78 |
| intersects | 62 | setRestitution | 78 |
| l | 62 | setSensor | 78 |
| merge | 62 | setSurfaceV | 79 |
| r | 63 | surfaceV | 79 |
| setb | 62 | Space Queries | 100 |
| setl | 63 | pointQuery | 100 |
| setr | 63 | pointQueryFirst | 100 |
| sett | 63 | segmentQuery | 101 |
| t | 64 | segmentQueryFirst | 101 |
| wrapVect | 64 | Spaces | 82 |
| Circle Shapes | 79 | addBody | 83 |
| CircleShape | 79 | addCollisionHandler | 83 |
| offset | 80 | addConstraint | 83 |
| radius | 80 | addPostStepCallback | 84 |
| Constraints | 91 | addShape | 84 |
| Damped Rotary Spring | 91 | addStaticShape | 85 |
| Damped Spring | 92 | damping | 85 |
| Gear Joint | 92 | data | 85 |
| Groove Joint | 93 | elasticIterations | 85 |
| Pin Joint | 93 | gravity | 85 |
| Pivot Joint | 94 | idleSpeedThreshold | 86 |
| Ratchet Joint | 94 | iterations | 86 |
| Rotary Limit Joint | 94 | rehashShape | 86 |
| Simple Motor | 95 | rehashStatic | 86 |
| Slide Joints | 95 | removeBody | 86 |
| Miscellaneous routines | 51 | removeConstraint | 87 |
| INFINITY | 51 | removeShape | 87 |
| momentForBox | 51 | removeStaticShape | 87 |
| momentForCircle | 51 | resizeActiveHash | 87 |
| momentForPoly | 52 | resizeStaticHash | 88 |
| momentForSegment | 52 | setDamping | 88 |
| Polygon Shapes | 80 | setData | 88 |
| numVerts | 80 | setElasticIterations | 89 |
| points | 81 | setGravity | 89 |
| PolyShape | 80 | setIdleSpeedThreshold | 89 |
| vert | 81 | setIterations | 89 |
| Segment Shapes | 81 | setSleepTimeThreshold | 90 |
| a | 82 | sleepTimeThreshold | 90 |
| b | 82 | Space | 83 |
| normal | 82 | step | 91 |
| radius | 82 | Vectors | 52 |
| SegmentShape | 81 | add | 53 |
| | | clamp | 53 |

| | |
|------------------------------|----|
| cross | 53 |
| dist | 54 |
| distq | 54 |
| dot | 54 |
| eql | 54 |
| length | 55 |
| lengthsq | 55 |
| lerp | 55 |
| lerpconst | 55 |
| mult | 56 |
| near | 56 |
| neg | 56 |
| normalize | 56 |
| normalizeSafe | 57 |
| perp | 57 |
| project | 57 |
| rotate | 57 |
| rperp | 57 |
| setx | 58 |
| sety | 58 |
| slerp | 58 |
| slerpconst | 59 |
| sub | 59 |
| toangle | 59 |
| unrotate | 59 |
| Vect | 52 |
| x | 60 |
| y | 60 |
| Platform Library | 40 |
| apiLevel | 40 |
| getDeviceID | 43 |
| hw | 40 |
| isColorDisplay | 40 |
| isDeviceModeRendering | 41 |
| isTableModeRendering | 41 |
| registerErrorHandling | 41 |
| window | 41 |
| displayInvalidatedRectangles | 42 |
| getScrollHeight | 42 |
| height and width | 41 |
| invalidate | 41 |
| setBackgroundcolor | 42 |
| setFocus | 42 |
| setScrollHeight | 42 |
| withGC | 43 |

S

| | |
|---------------------------------------|----|
| Standard Libraries | 1 |
| Basic Library Functions | 1 |
| Coroutine Sub-Library | 1 |
| Math Library | 2 |
| Module Library | 1 |
| String Library | 1 |
| Table Library | 2 |
| Unimplemented Libraries and Functions | 2 |
| String Library Extension | 44 |
| pack | 44 |
| split | 44 |
| uchar | 44 |
| unpack | 45 |
| usub | 44 |

T

| | |
|---|----|
| Timer Library | 46 |
| getMilliSecCounter | 46 |
| start | 46 |
| stop | 46 |
| Tool Palette Library | 47 |
| enable | 47 |
| enableCopy | 48 |
| enableCut | 48 |
| enablePaste | 48 |
| register | 47 |
| Touch Library | 3 |
| Library Functions | 4 |
| enabled | 4 |
| isKeyboardAvaliable | 5 |
| isKeyboardVisible | 5 |
| ppi | 4 |
| showKeyboard | 5 |
| xppi | 4 |
| yppi | 4 |
| Overview | 3 |
| Event Handling | 3 |
| On-Screen Keyboard and Screen Resize Behavior | 3 |

V

| | |
|------------------|----|
| Variable Library | 49 |
| list | 49 |
| makeNumericList | 49 |
| monitor | 49 |
| recall | 49 |
| recallAt | 50 |
| recallStr | 50 |
| store | 50 |
| storeAt | 50 |
| unmonitor | 50 |