A to Z Function and Instruction Reference

Quick-Find Locator...................................................... 262
Alphabetical Listing of Operations.............................. 266
## Quick-Find Locator

This section lists the TI-86 functions and instructions in functional groups along with the page numbers where they are described in this chapter.

### Graphing

<table>
<thead>
<tr>
<th>Function</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Axes(</td>
<td>271</td>
</tr>
<tr>
<td>AxesOff</td>
<td>271</td>
</tr>
<tr>
<td>AxesOn</td>
<td>271</td>
</tr>
<tr>
<td>Circ(</td>
<td>273</td>
</tr>
<tr>
<td>ClDrw</td>
<td>273</td>
</tr>
<tr>
<td>CoordOff</td>
<td>275</td>
</tr>
<tr>
<td>CoordOn</td>
<td>275</td>
</tr>
<tr>
<td>DiffEq</td>
<td>281</td>
</tr>
<tr>
<td>DirFid</td>
<td>282</td>
</tr>
<tr>
<td>DrawDot</td>
<td>285</td>
</tr>
<tr>
<td>DrawF</td>
<td>286</td>
</tr>
<tr>
<td>DrawLine</td>
<td>286</td>
</tr>
<tr>
<td>DrEqu(</td>
<td>287</td>
</tr>
<tr>
<td>DrInv</td>
<td>287</td>
</tr>
<tr>
<td>Line(</td>
<td>314</td>
</tr>
<tr>
<td>RectGC</td>
<td>344</td>
</tr>
<tr>
<td>ZFit</td>
<td>373</td>
</tr>
<tr>
<td>ZInt</td>
<td>374</td>
</tr>
<tr>
<td>ZPrev</td>
<td>375</td>
</tr>
<tr>
<td>ZSqr</td>
<td>376</td>
</tr>
<tr>
<td>ZStd</td>
<td>377</td>
</tr>
<tr>
<td>ZTrig</td>
<td>378</td>
</tr>
</tbody>
</table>

### Lists

<table>
<thead>
<tr>
<th>Function</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>aug()</td>
<td>270</td>
</tr>
<tr>
<td>cSum()</td>
<td>278</td>
</tr>
<tr>
<td>Deltalst()</td>
<td>279</td>
</tr>
<tr>
<td>dimL</td>
<td>282</td>
</tr>
<tr>
<td>®dimL</td>
<td>282</td>
</tr>
<tr>
<td>Fill()</td>
<td>285</td>
</tr>
<tr>
<td>Form()</td>
<td>288</td>
</tr>
<tr>
<td>List entry: { }</td>
<td>316</td>
</tr>
<tr>
<td>livc</td>
<td>316</td>
</tr>
<tr>
<td>prod</td>
<td>338</td>
</tr>
<tr>
<td>Select()</td>
<td>350</td>
</tr>
<tr>
<td>seq()</td>
<td>351</td>
</tr>
<tr>
<td>trotx</td>
<td>359</td>
</tr>
<tr>
<td>Sorty</td>
<td>359</td>
</tr>
<tr>
<td>Sortx</td>
<td>359</td>
</tr>
<tr>
<td>Sorty</td>
<td>359</td>
</tr>
<tr>
<td>Sortx</td>
<td>359</td>
</tr>
<tr>
<td>sum</td>
<td>364</td>
</tr>
<tr>
<td>vclli</td>
<td>369</td>
</tr>
<tr>
<td>ZData</td>
<td>371</td>
</tr>
<tr>
<td>ZData</td>
<td>371</td>
</tr>
<tr>
<td>ZData</td>
<td>371</td>
</tr>
<tr>
<td>ZData</td>
<td>371</td>
</tr>
</tbody>
</table>
### Mathematics, Algebra, and Calculus

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>abs</td>
<td>Absolute value</td>
<td>267</td>
</tr>
<tr>
<td>Addition:</td>
<td>sum</td>
<td>267</td>
</tr>
<tr>
<td>and</td>
<td>Logical AND</td>
<td>268</td>
</tr>
<tr>
<td>angle</td>
<td>Angle entry</td>
<td>269</td>
</tr>
<tr>
<td>Ans</td>
<td>Answer</td>
<td>269</td>
</tr>
<tr>
<td>arc</td>
<td>Arc entry</td>
<td>269</td>
</tr>
<tr>
<td>Assignment:</td>
<td>equal to:</td>
<td>270</td>
</tr>
<tr>
<td>b</td>
<td>Base</td>
<td>271</td>
</tr>
<tr>
<td>Bin</td>
<td>Binary entry</td>
<td>272</td>
</tr>
<tr>
<td>Bin</td>
<td>Equal to:</td>
<td>272</td>
</tr>
<tr>
<td>ClrEnt</td>
<td>Clear entry</td>
<td>273</td>
</tr>
<tr>
<td>ClrTbl</td>
<td>Clear table</td>
<td>273</td>
</tr>
<tr>
<td>conj</td>
<td>Conjugate</td>
<td>275</td>
</tr>
<tr>
<td>cos</td>
<td>Cosine</td>
<td>276</td>
</tr>
<tr>
<td>cos&lt;sup&gt;-1&lt;/sup&gt;</td>
<td>Cosine inverse</td>
<td>276</td>
</tr>
<tr>
<td>cosh</td>
<td>Hyperbolic cosine</td>
<td>277</td>
</tr>
<tr>
<td>cosh&lt;sup&gt;-1&lt;/sup&gt;</td>
<td>Hyperbolic cosine inverse</td>
<td>277</td>
</tr>
<tr>
<td>d</td>
<td>Degree entry</td>
<td>278</td>
</tr>
<tr>
<td>Dec</td>
<td>Decimal entry</td>
<td>278</td>
</tr>
<tr>
<td>Dec</td>
<td>Equal to:</td>
<td>278</td>
</tr>
<tr>
<td>Degree</td>
<td>Degree entry</td>
<td>279</td>
</tr>
<tr>
<td>Degree entry:</td>
<td>Equal to:</td>
<td>279</td>
</tr>
<tr>
<td>der1</td>
<td>1st derivative</td>
<td>280</td>
</tr>
<tr>
<td>der2</td>
<td>2nd derivative</td>
<td>280</td>
</tr>
<tr>
<td>Div, mod</td>
<td>Division, modulus</td>
<td>284</td>
</tr>
<tr>
<td>DMS entry:</td>
<td>Equal to:</td>
<td>285</td>
</tr>
<tr>
<td>DMS</td>
<td>DMS entry</td>
<td>285</td>
</tr>
<tr>
<td>dxDer1</td>
<td>1st derivative</td>
<td>288</td>
</tr>
<tr>
<td>dxNDer</td>
<td>1st derivative</td>
<td>288</td>
</tr>
<tr>
<td>e^</td>
<td>Exponent</td>
<td>288</td>
</tr>
<tr>
<td>eng</td>
<td>ENG entry</td>
<td>290</td>
</tr>
<tr>
<td>Eq</td>
<td>Equation</td>
<td>290</td>
</tr>
<tr>
<td>EqSt</td>
<td>Equation stored</td>
<td>290</td>
</tr>
<tr>
<td>exp</td>
<td>Exponential</td>
<td>294</td>
</tr>
<tr>
<td>Euler</td>
<td>Euler entry</td>
<td>291</td>
</tr>
<tr>
<td>eval</td>
<td>Evaluation</td>
<td>291</td>
</tr>
<tr>
<td>evalF</td>
<td>Evaluation</td>
<td>292</td>
</tr>
<tr>
<td>Fix</td>
<td>Fix</td>
<td>292</td>
</tr>
<tr>
<td>float</td>
<td>Float entry</td>
<td>294</td>
</tr>
<tr>
<td>gcd</td>
<td>Greatest common divisor</td>
<td>299</td>
</tr>
<tr>
<td>EqPSt</td>
<td>Equation prime stored</td>
<td>300</td>
</tr>
<tr>
<td>frac</td>
<td>Fraction</td>
<td>301</td>
</tr>
<tr>
<td>Inter</td>
<td>Intersection</td>
<td>301</td>
</tr>
<tr>
<td>lcm</td>
<td>Least common multiple</td>
<td>305</td>
</tr>
<tr>
<td>less than</td>
<td>Less than</td>
<td>312</td>
</tr>
<tr>
<td>Less than or</td>
<td>Equal to:</td>
<td>312</td>
</tr>
<tr>
<td>ln</td>
<td>Natural logarithm</td>
<td>316</td>
</tr>
<tr>
<td>log</td>
<td>Logarithm</td>
<td>318</td>
</tr>
<tr>
<td>max</td>
<td>Maximum</td>
<td>319</td>
</tr>
<tr>
<td>min</td>
<td>Minimum</td>
<td>320</td>
</tr>
<tr>
<td>mod</td>
<td>Modulus</td>
<td>320</td>
</tr>
<tr>
<td>Multiplication:</td>
<td>Equal to:</td>
<td>321</td>
</tr>
<tr>
<td>nCr</td>
<td>Combination</td>
<td>321</td>
</tr>
<tr>
<td>nDer</td>
<td>Derivative</td>
<td>323</td>
</tr>
<tr>
<td>negation</td>
<td>Negation</td>
<td>323</td>
</tr>
<tr>
<td>Normal</td>
<td>Normal</td>
<td>324</td>
</tr>
<tr>
<td>not</td>
<td>Not equal to</td>
<td>325</td>
</tr>
<tr>
<td>nPr</td>
<td>Permutation</td>
<td>326</td>
</tr>
<tr>
<td>Offset</td>
<td>Offset</td>
<td>326</td>
</tr>
<tr>
<td>shfl</td>
<td>Shift left</td>
<td>353</td>
</tr>
<tr>
<td>shfr</td>
<td>Shift right</td>
<td>353</td>
</tr>
<tr>
<td>sign</td>
<td>Sign</td>
<td>354</td>
</tr>
<tr>
<td>simul</td>
<td>Simulation</td>
<td>354</td>
</tr>
<tr>
<td>sin</td>
<td>Sine</td>
<td>355</td>
</tr>
<tr>
<td>sin&lt;sup&gt;-1&lt;/sup&gt;</td>
<td>Sine inverse</td>
<td>355</td>
</tr>
<tr>
<td>sinh</td>
<td>Hyperbolic sine</td>
<td>356</td>
</tr>
<tr>
<td>sinh&lt;sup&gt;-1&lt;/sup&gt;</td>
<td>Hyperbolic sine inverse</td>
<td>356</td>
</tr>
<tr>
<td>Solver</td>
<td>Solver</td>
<td>358</td>
</tr>
<tr>
<td>sqrt</td>
<td>Square root</td>
<td>360</td>
</tr>
<tr>
<td>Store</td>
<td>Store to</td>
<td>361</td>
</tr>
<tr>
<td>Subtraction:</td>
<td>Equal to:</td>
<td>362</td>
</tr>
<tr>
<td>tan</td>
<td>Tangent</td>
<td>364</td>
</tr>
<tr>
<td>tan&lt;sup&gt;-1&lt;/sup&gt;</td>
<td>Tangent inverse</td>
<td>365</td>
</tr>
<tr>
<td>tanh</td>
<td>Hyperbolic tangent</td>
<td>365</td>
</tr>
<tr>
<td>tanh&lt;sup&gt;-1&lt;/sup&gt;</td>
<td>Hyperbolic tangent inverse</td>
<td>365</td>
</tr>
<tr>
<td>xor</td>
<td>Xor</td>
<td>370</td>
</tr>
<tr>
<td>round</td>
<td>Round</td>
<td>348</td>
</tr>
<tr>
<td>Sci</td>
<td>Scientific notation</td>
<td>349</td>
</tr>
<tr>
<td>shfl</td>
<td>Shift left</td>
<td>353</td>
</tr>
<tr>
<td>shfr</td>
<td>Shift right</td>
<td>353</td>
</tr>
<tr>
<td>sign</td>
<td>Sign</td>
<td>354</td>
</tr>
<tr>
<td>simul</td>
<td>Simulation</td>
<td>354</td>
</tr>
<tr>
<td>sin</td>
<td>Sine</td>
<td>355</td>
</tr>
<tr>
<td>sin&lt;sup&gt;-1&lt;/sup&gt;</td>
<td>Sine inverse</td>
<td>355</td>
</tr>
<tr>
<td>sinh</td>
<td>Hyperbolic sine</td>
<td>356</td>
</tr>
<tr>
<td>sinh&lt;sup&gt;-1&lt;/sup&gt;</td>
<td>Hyperbolic sine inverse</td>
<td>356</td>
</tr>
<tr>
<td>Solver</td>
<td>Solver</td>
<td>358</td>
</tr>
<tr>
<td>sqrt</td>
<td>Square root</td>
<td>360</td>
</tr>
<tr>
<td>Store</td>
<td>Store to</td>
<td>361</td>
</tr>
<tr>
<td>Subtraction:</td>
<td>Equal to:</td>
<td>362</td>
</tr>
<tr>
<td>tan</td>
<td>Tangent</td>
<td>364</td>
</tr>
<tr>
<td>tan&lt;sup&gt;-1&lt;/sup&gt;</td>
<td>Tangent inverse</td>
<td>365</td>
</tr>
<tr>
<td>tanh</td>
<td>Hyperbolic tangent</td>
<td>365</td>
</tr>
<tr>
<td>tanh&lt;sup&gt;-1&lt;/sup&gt;</td>
<td>Hyperbolic tangent inverse</td>
<td>365</td>
</tr>
<tr>
<td>xor</td>
<td>Xor</td>
<td>370</td>
</tr>
</tbody>
</table>
Chapter 20: A to Z Function and Instruction Reference

Matrices

<table>
<thead>
<tr>
<th>Function</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>aug</td>
<td>270</td>
</tr>
<tr>
<td>cnorm</td>
<td>273</td>
</tr>
<tr>
<td>cond</td>
<td>274</td>
</tr>
<tr>
<td>det</td>
<td>281</td>
</tr>
<tr>
<td>dim</td>
<td>281</td>
</tr>
<tr>
<td>➤dim</td>
<td>281</td>
</tr>
<tr>
<td>LU</td>
<td>318</td>
</tr>
<tr>
<td>rAdd</td>
<td>340</td>
</tr>
<tr>
<td>rSwap</td>
<td>348</td>
</tr>
</tbody>
</table>

Programming

<table>
<thead>
<tr>
<th>Function</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Asm</td>
<td>269</td>
</tr>
<tr>
<td>AsmComp</td>
<td>270</td>
</tr>
<tr>
<td>AsmPrgm</td>
<td>270</td>
</tr>
<tr>
<td>CILCD</td>
<td>273</td>
</tr>
<tr>
<td>DelVar</td>
<td>290</td>
</tr>
<tr>
<td>Disp</td>
<td>283</td>
</tr>
<tr>
<td>DispG</td>
<td>283</td>
</tr>
<tr>
<td>DispT</td>
<td>284</td>
</tr>
<tr>
<td>DS&lt;</td>
<td>288</td>
</tr>
<tr>
<td>Else</td>
<td>290</td>
</tr>
<tr>
<td>End</td>
<td>290</td>
</tr>
<tr>
<td>Equal</td>
<td>290</td>
</tr>
<tr>
<td>Equal to</td>
<td>291</td>
</tr>
<tr>
<td>For</td>
<td>297</td>
</tr>
<tr>
<td>Get</td>
<td>299</td>
</tr>
<tr>
<td>InpSt</td>
<td>307</td>
</tr>
<tr>
<td>Input</td>
<td>307</td>
</tr>
<tr>
<td>Prompt</td>
<td>338</td>
</tr>
<tr>
<td>Repeat</td>
<td>345</td>
</tr>
<tr>
<td>Return</td>
<td>345</td>
</tr>
<tr>
<td>Stop</td>
<td>362</td>
</tr>
<tr>
<td>Then</td>
<td>366</td>
</tr>
<tr>
<td>While</td>
<td>369</td>
</tr>
</tbody>
</table>

Statistics

<table>
<thead>
<tr>
<th>Function</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Box</td>
<td>272</td>
</tr>
<tr>
<td>ExpR</td>
<td>293</td>
</tr>
<tr>
<td>fcstx</td>
<td>294</td>
</tr>
<tr>
<td>fcty</td>
<td>294</td>
</tr>
<tr>
<td>Hist</td>
<td>303</td>
</tr>
<tr>
<td>LgstR</td>
<td>313</td>
</tr>
<tr>
<td>LinR</td>
<td>315</td>
</tr>
<tr>
<td>LnR</td>
<td>317</td>
</tr>
<tr>
<td>MBox</td>
<td>319</td>
</tr>
<tr>
<td>OneVar</td>
<td>327</td>
</tr>
<tr>
<td>P2Reg</td>
<td>330</td>
</tr>
<tr>
<td>P3Reg</td>
<td>331</td>
</tr>
<tr>
<td>P4Reg</td>
<td>332</td>
</tr>
<tr>
<td>POff</td>
<td>334</td>
</tr>
<tr>
<td>Plot1</td>
<td>335</td>
</tr>
<tr>
<td>Plot2</td>
<td>335</td>
</tr>
<tr>
<td>Plot3</td>
<td>335</td>
</tr>
<tr>
<td>PwrR</td>
<td>339</td>
</tr>
<tr>
<td>rand</td>
<td>341</td>
</tr>
<tr>
<td>randBin</td>
<td>341</td>
</tr>
<tr>
<td>randInt</td>
<td>342</td>
</tr>
<tr>
<td>randM</td>
<td>342</td>
</tr>
<tr>
<td>randNorm</td>
<td>342</td>
</tr>
<tr>
<td>Scat</td>
<td>349</td>
</tr>
<tr>
<td>Select</td>
<td>350</td>
</tr>
<tr>
<td>SetLEdit</td>
<td>351</td>
</tr>
<tr>
<td>SinR</td>
<td>357</td>
</tr>
<tr>
<td>Sortx</td>
<td>359</td>
</tr>
<tr>
<td>Sorty</td>
<td>359</td>
</tr>
<tr>
<td>StReg</td>
<td>362</td>
</tr>
<tr>
<td>TwoVar</td>
<td>368</td>
</tr>
<tr>
<td>xyline</td>
<td>370</td>
</tr>
</tbody>
</table>
### Strings

<table>
<thead>
<tr>
<th>Function</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Concatenation: +</td>
<td>274</td>
</tr>
<tr>
<td>EqStf</td>
<td>290</td>
</tr>
<tr>
<td>InGth</td>
<td>316</td>
</tr>
<tr>
<td>StEq</td>
<td>361</td>
</tr>
<tr>
<td>String entry: &quot;</td>
<td>363</td>
</tr>
<tr>
<td>sub</td>
<td>363</td>
</tr>
</tbody>
</table>

### Vectors

<table>
<thead>
<tr>
<th>Function</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>cnorm</td>
<td>273</td>
</tr>
<tr>
<td>cross</td>
<td>277</td>
</tr>
<tr>
<td>Cyl</td>
<td>278</td>
</tr>
<tr>
<td>CylV</td>
<td>278</td>
</tr>
<tr>
<td>dim</td>
<td>281</td>
</tr>
<tr>
<td>dim</td>
<td>281</td>
</tr>
<tr>
<td>dot</td>
<td>285</td>
</tr>
<tr>
<td>Fill</td>
<td>295</td>
</tr>
<tr>
<td>liVc</td>
<td>316</td>
</tr>
<tr>
<td>norm</td>
<td>323</td>
</tr>
<tr>
<td>RectV</td>
<td>344</td>
</tr>
<tr>
<td>rnorm</td>
<td>346</td>
</tr>
<tr>
<td>Sph</td>
<td>360</td>
</tr>
<tr>
<td>SphereV</td>
<td>360</td>
</tr>
<tr>
<td>unitV</td>
<td>368</td>
</tr>
<tr>
<td>vclli</td>
<td>369</td>
</tr>
</tbody>
</table>

Vector entry: []
Alphabetical Listing of Operations

All the operations in this section are included in the CATALOG. Non-alphabetic operations (such as +, 1, and >) are listed at the end of the CATALOG. In this A to Z Reference, however, these operations are listed under their alphabetic equivalent (such as addition, factorial, and greater than).

You always can use the CATALOG to select an operation and paste it to the home screen or to a command line in the program editor. You also can use the specific keystrokes, menus, or screens listed in this section.

† Indicates menus or screens that paste the operation’s name only if you are in the program editor. In most cases, you can use these menus or screens from the home screen to perform the operation interactively, without pasting the name.

‡ Indicates menus or screens that are valid only from the program editor’s main menu. From the home screen, you cannot use these menus or screens to select an operation.

The syntax for some operations uses brackets [ ] to indicate optional arguments. If you use an optional argument, do not enter the brackets.
**abs**

MATH NUM menu
CPLX menu
MATRX CPLX menu
VECTR CPLX menu

**abs**

**abs realNumber** or **abs (realExpression)**

Returns the absolute value of **realNumber** or **realExpression**.

**abs (complexNumber)**

Returns the magnitude (modulus) of **complexNumber**.

**abs (real,imaginary)** returns \(\sqrt{\text{real}^2 + \text{imaginary}^2}\).

**abs (magnitude,angle)** returns **magnitude**.

**abs list**

**abs matrix**

**abs vector**

Returns a list, matrix, or vector in which each element is the absolute value of the corresponding real or complex element in the argument.

**Addition: +**

**numberA + numberB**

Returns the sum of two real or complex numbers.

**number + list**

Returns a list in which a real or complex **number** is added to each element of a real or complex **list**.

In **RectC** complex number mode:

\( (2,5) + (5,9) \) \( (7,14) \)

\( 4 + (1,2,3) \) \( (5,6,7) \)

\( 3 + (1,7,(2,1)) \) \( ((4,0),(10,0),(5,1)) \)
Chapter 20: A to Z Function and Instruction Reference

\[ \text{listA} + \text{listB} \]
\[ \text{matrixA} + \text{matrixB} \]
\[ \text{vectorA} + \text{vectorB} \]

Returns a list, matrix, or vector that is the sum of the corresponding real or complex elements in the arguments. The two arguments must have the same dimension.

For information about adding two strings, refer to Concatenation on page 274.

and

BASE BOOL menu

\[ \text{integerA and integerB} \]

Compares two real integers bit by bit. Internally, both integers are converted to binary. When corresponding bits are compared, the result is 1 if both bits are 1; otherwise, the result is 0. The returned value is the sum of the bit results.

For example, \( 78 \) and \( 23 = 6 \).

\[
\begin{align*}
78 & = 1001110b \\
23 & = 0010111b \\
0000110b & = 6
\end{align*}
\]

You can enter real numbers instead of integers, but they are truncated automatically before the comparison.
### angle

**CPLX menu**
**MATRX CPLX menu**
**VECTR CPLX menu**

**angle**

- **angle** (*complexNumber*)
  - Returns the polar angle of *complexNumber*, adjusted by $\pi$ in the 2nd quadrant or $-\pi$ in the 3rd quadrant. The polar angle of a real number is always 0.
  - **angle** (*real*, *imaginary*) returns $\tan^{-1}(*imaginary*/*real*)$.
  - **angle** (*magnitude* $\angle$ *angle*) returns *angle*, $-\pi < \text{angle} \leq \pi$.

**angle** *complexList*
**angle** *complexMatrix*
**angle** *complexVector*

- Returns a list, matrix, or vector in which each element is the polar angle of the corresponding element in the argument.
- If *complexVector* has only two real elements, the returned value is a real number, not a vector.

### Ans

- **Ans**
  - $\text{Ans}$
  - Returns the last answer.

### arc

- **arc** (*expression*, *variable*, *start*, *end*)
  - Returns the length along *expression* with respect to *variable*, from *variable* = *start* to *variable* = *end*.

### Asm

- **Asm** (*assemblyProgramName*)
  - Executes an assembly language program. For more information, refer to Chapter 16.
### Chapter 20: A to Z Function and Instruction Reference

**AsmComp**

| **AsmComp(AsmComp(AsciiAssemblyPrgmName,HexAssemblyPrgmName)** |
| --- | --- |
| **Chapter 20: A to Z Function and Instruction Reference** |
| Compiles an assembly language program written in ASCII and stores the hex version. The compiled hex version, which uses about half the storage space of the ASCII version, cannot be edited. |
| When you execute the ASCII version, the TI-86 compiles it each time. To speed up execution, use AsmComp( to compile the ASCII version once and then execute the hex version each time you want to run the program. |

**AsmPrgm**

<table>
<thead>
<tr>
<th><strong>AsmPrgm</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Must be used as the first line of an assembly language program.</td>
</tr>
</tbody>
</table>

**Assignment:**

<table>
<thead>
<tr>
<th><strong>equationVariable = expression</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Stores expression to equationVariable, without evaluating expression. (If you use STO to store an expression to a variable, the expression is evaluated and then the result is stored.)</td>
</tr>
</tbody>
</table>

**aug(**

<table>
<thead>
<tr>
<th><strong>aug(listA,listB)</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Returns a list consisting of listB appended (concatenated) to the end of listA. The lists can be real or complex.</td>
</tr>
</tbody>
</table>
aug(matrixA, matrixB)

Returns a matrix consisting of matrixB appended as new columns to the end of matrixA. The matrices can be real or complex. Both must have the same number of rows.

aug(matrix, vector)

Returns a matrix consisting of vector appended as a new column to the end of matrix. The arguments can be real or complex. The number of rows in matrix must equal the number of elements in vector.

\[
\begin{bmatrix}
1 & 2 & 3 \\
4 & 5 & 6 \\
\end{bmatrix}
\]

\[
\begin{bmatrix}
7 & 8 \\
9 & 10 \\
\end{bmatrix}
\]

\[
\begin{bmatrix}
1 & 2 & 3 & 7 & 8 \\
4 & 5 & 6 & 9 & 10 \\
\end{bmatrix}
\]

Axes(† GRAPH VARS menu)

Axes(xAxisVariable, yAxisVariable)

Specifies the variables plotted for the axes in DifEq graphing mode. The xAxisVariable or yAxisVariable can be t, Q1 through Q9, or Q’1 through Q’9.

AxesOff† graph format screen

AxesOff

Turns off the graph axes.

AxesOn† graph format screen

AxesOn

Turns on the graph axes.

\[
\text{integer}b
\]

Designates a real integer as binary, regardless of the number base mode setting.

In Dec number base mode:

\[
10b \quad \text{ENTER} \quad 2
\]

\[
10b + 10 \quad \text{ENTER} \quad 12
\]
Bin
	mode screen
Sets binary number base mode. Results are displayed with the b suffix. In any number base mode, you can designate an appropriate value as binary, decimal, hexadecimal, or octal by using the b, d, h, or o designator, respectively, from the BASE TYPE menu.

Bin
BASE CONV menu

number→Bin
list→Bin
matrix→Bin
vector→Bin

Returns the binary equivalent of the real or complex argument.

In Bin number base mode:

10+Bu10c+10d ENTER 100011b

In Dec number base mode:

2→B ENTER 16
Ans→Bin ENTER 10000b

Starting with a ZStd graph screen:

{1,2,3,4,9}→XL ENTER
{1,2,3,4,5,9}→FL ENTER
{1,1,1,4,1,1}→yMin ENTER
{1,1,1,4,1,1}→yMin ENTER

Uses frequencies of 1.

Uses the data in built-in variables xStat and fStat. These variables must contain valid data of the same dimension; otherwise, an error occurs.
**Circl(**

† GRAPH DRAW menu

`Circl(x,y,radius)`

Draws a circle with center `(x,y)` and `radius` on the current graph.

Starting with a `ZStd` graph screen:

```
ZStd:Circl(1,2,7) ENTER
```

---

**ClDrw**

† GRAPH DRAW menu
† STAT DRAW menu

`ClDrw`

Clears all drawn elements from the current graph.

---

**ClLCD**

‡ program editor
I/O menu

`ClLCD`

Clears the home screen (LCD).

---

**ClrEnt**

MEM menu

`ClrEnt`

Clears the contents of the Last Entry storage area.

---

**ClTbl**

‡ program editor
I/O menu

`ClTbl`

Clears all values from the current table if `Indpnt: Ask (IAsk`, page 304) is set.

---

**cnorm**

MATRX MATH menu

`cnorm matrix`

Returns the column norm of a real or complex `matrix`. For each column, `cnorm` sums the absolute values (magnitudes of complex elements) of the elements in that column and returns the largest of those column sums.

```
[[1,-2,3]
 [4,5,-6]] → MAT [ENTER]
[[1,2,3]]
 [4 5 -6]]
```

```
cnorm MAT [ENTER]
```

```
9
```
**cnorm vector**

Returns the sum of the absolute values of the real or complex elements in `vector`.  

```
[-1,2,-3]→VEC ENTER  [-1 2 -3]
cnorm VEC ENTER  6
```

**Concatenation: +**

```
stringA + stringB
```

Returns a string consisting of `stringB` appended (concatenated) to the end of `stringA`.  

```
"your name:"→STR ENTER
your name:  "Enter "→STR ENTER
Enter your name:
```

**cond**

**MATRX MATH menu**

Returns the condition number of a real or complex `squareMatrix`, which is calculated as:

```
cnorm squareMatrix * cnorm squareMatrix⁻¹
```

The condition number indicates how well-behaved `squareMatrix` is expected to be for certain matrix functions, particularly inverse. For a well-behaved matrix, the condition number is close to 1.  

```
log(cond squareMatrix) indicates the number of digits that may be lost due to round-off errors in computing the inverse.
```

For a matrix with no inverse, `cond` returns an error.  

```
[[1,0,0][0,1,0][0,0,1]]→MAT1
[[1 0 0]
 [0 1 0]
 [0 0 1]]
cond MAT1 ENTER  1
log (Ans) ENTER  0
[[1,2,3][4,5,6][7,8,9]]→MAT2
[[1 2 3]
 [4 5 6]
 [7 8 9]]
cond MAT2 ENTER  1.8614
log (Ans) ENTER  14.2552725051
```
### conj

**CPLX menu**

**MATRX CPLX menu**

**VECTR CPLX menu**

**conj**

**conjugate**

**Returns the complex conjugate of complexNumber.**

In **RectC** mode, **conj** (real, imaginary) returns (real, -imaginary).

In **PolarC** mode, **conj** (magnitude, angle) returns (magnitude, -angle), \(-\pi < \text{angle} \leq \pi\).

**conj complexList**

**conj complexMatrix**

**conj complexVector**

Returns a complex list, matrix, or vector in which each element is the complex conjugate of the original.

### CoordOff

**† graph format screen**

**CoordOff**

Turns off cursor coordinates so they are not displayed at the bottom of a graph.

### CoordOn

**† graph format screen**

**CoordOn**

Displays cursor coordinates at the bottom of a graph.

---

**In RectC complex number mode:**

<table>
<thead>
<tr>
<th>conj (3,4)</th>
<th>conj (3+2i)</th>
</tr>
</thead>
<tbody>
<tr>
<td>conj (3,4)</td>
<td>conj (3-i)</td>
</tr>
<tr>
<td>conj (3+2i)</td>
<td>conj (3+i)</td>
</tr>
<tr>
<td>conj (3,4)</td>
<td>conj (3+2i)</td>
</tr>
</tbody>
</table>

**In PolarC complex number mode:**

<table>
<thead>
<tr>
<th>conj (3,4) ( \langle 5\angle 92.72\degree \rangle )</th>
<th>conj (( \sqrt{2} ), (3,4)) [ENTER]</th>
</tr>
</thead>
<tbody>
<tr>
<td>conj (3,4) ( \langle 5\angle 92.72\degree \rangle )</td>
<td>conj (( \sqrt{2} ), (3,4)) [ENTER]</td>
</tr>
</tbody>
</table>

\( \langle 1.41421356237\angle 1.5\degree \rangle \)
**COS**

**COS angle or COS (expression)**
- Returns the cosine of *angle* or *expression*, which can be real or complex.
- An angle is interpreted as degrees or radians according to the current angle mode. In any angle mode, you can designate an angle as degrees or radians by using the ° or ′ designator, respectively, from the MATH ANGLE menu.

**COS list**
- Returns a list in which each element is the cosine of the corresponding element in *list*.

**COS squareMatrix**
- Returns a square matrix that is the matrix cosine of *squareMatrix*. The matrix cosine corresponds to the result calculated using power series or Cayley-Hamilton Theorem techniques. This is *not* the same as simply calculating the cosine of each element.

**COS⁻¹**

**COS⁻¹ number or COS⁻¹ (expression)**
- Returns the arccosine of *number* or *expression*, which can be real or complex.

**COS⁻¹ list**
- Returns a list in which each element is the arccosine of the corresponding element in *list*. 
### cosh

- **MATH HYP menu**

  - **cosh**
    - **cosh number or cosh (expression)**
      - Returns the hyperbolic cosine of `number or expression`, which can be real or complex.
      - Example: `cosh(1.2)`
        - Input: `cosh 1.2` (ENTER)
        - Output: `1.81065556732`

  - **cosh list**
    - Returns a list in which each element is the hyperbolic cosine of the corresponding element in `list`.
    - Example: `cosh {0,1.2}`
      - Input: `cosh {0,1.2}` (ENTER)
      - Output: `{1 1.81065556732}`

### cosh⁻¹

- **MATH HYP menu**

  - **cosh⁻¹**
    - **cosh⁻¹ number or cosh⁻¹ (expression)**
      - Returns the inverse hyperbolic cosine of `number or expression`, which can be real or complex.
      - Example: `cosh⁻¹(1)`
        - Input: `cosh⁻¹ 1` (ENTER)
        - Output: `0`

  - **cosh⁻¹ list**
    - Returns a list in which each element is the inverse hyperbolic cosine of the corresponding element in `list`.
    - Example: `cosh⁻¹ {1,2.1,3}`
      - Input: `cosh⁻¹ {1,2.1,3}` (ENTER)
      - Output: `{0 1.37285914424 1.7...`

### cross()

- **VECTR MATH menu**

  - **cross(vectorA, vectorB)**
    - Returns the cross product of two real or complex vectors, where:
      - `cross([a,b,c],[d,e,f]) = [bf-cd ae-bd]`
    - Both vectors must have the same dimension (either 2 or 3 elements). A 2-D vector is treated as a 3-D vector with 0 as the third element.
    - Example: `cross([1,2,3],[4,5,6])`
      - Input: `cross([1,2,3],[4,5,6])` (ENTER)
      - Output: `[-3 6 -3]`
    - Example: `cross([1,2],[3,4])`
      - Input: `cross([1,2],[3,4])` (ENTER)
      - Output: `[0 0 -2]`
### cSum

**LIST OPS menu**

- **cSum(list)**
  - Returns a list of the cumulative sums of the real or complex elements in `list`, starting with the first element.

  *Example:*
  - `cSum({1,2,3,4})`
  - `{10,20,30}`
  - `{10,20,30}`

### Cyl

**VECTR OPS menu**

- **Cyl**
  - Displays a 2- or 3-element real `vector` result in cylindrical form, `[r θ z]`, even if the display mode is not set for cylindrical (`CyIV`).

  *Examples:*
  - `{(-2,0)Cyl}`
  - `{2∠3.14159265359 0}`
  - `{(-2,0,1)Cyl}`
  - `{2∠3.14159265359 1}`

### CyIV

**† mode screen**

- **CyIV**
  - Sets cylindrical vector coordinate mode (`[r θ z]`).

  *Example:*
  - In `CyIV` vector coordinate mode and *Radian* angle mode:
  - `{3,4,5}`
  - `{5∠927295218002 5}`

### d

**BASE TYPE menu**

- **d**
  - Designates a real `number` as decimal, regardless of the number base mode setting.

  *Examples:*
  - In `Bin` number base mode:
    - `10d` [ENTER] 1010b
    - `10d+10` [ENTER] 1100b

### Dec

**† mode screen**

- **Dec**
  - Sets decimal number base mode. In any number base mode, you can designate an appropriate value as binary, decimal, hexadecimal, or octal by using the `b`, `d`, `h`, or `c` designator, respectively, from the BASE TYPE menu.

  *Example:*
  - In `Dec` number base mode:
    - `10+10b+Fh+10c` [ENTER] 35
**Dec**

BASE CONV menu

- `number→Dec`
- `list→Dec`
- `matrix→Dec`
- `vector→Dec`

Returns the decimal equivalent of the real or complex argument.

In **Hex** number base mode:

- `2+F {ENTER} 1Eh`
- `Ans→Dec {ENTER} 30d`
- `[A,B,C,D,E]→Dec {ENTER} [1d 1ld 12d 13d 14d]`

**Degree**

† mode screen

- **Sets degree angle mode.**

Degree entry: °

MATH ANGLE menu

- `number°` or `(expression)°`

Designates a real `number` or `expression` as degrees, regardless of the angle mode setting.

- `list°`

Designates each element in `list` as degrees.

**Deltalist(**

LIST OPS menu

(Deltal shows on menu)

- `Deltalist(list)`

Returns a list containing the differences between consecutive real or complex elements in `list`. This subtracts the first element in `list` from the second element, the second from the third, and so on. The resulting list is always one element shorter than `list`. 
DelVar\((variable)\)

Deletes the specified user-created \(variable\) from memory.

You cannot use \texttt{DelVar()} to delete a program variable or built-in variable.

\[ \text{DelVar}(A) \]

\texttt{DelVar(A)} \hspace{2em} \texttt{Done}

\texttt{ERROR 14 UNDEFINED}

\[ \text{DelVar}(\text{A}+2) \]

\texttt{(A+2)} \hspace{2em} \texttt{Done}

\texttt{ERROR 14 UNDEFINED}

\textbf{der1(} \textit{expression, variable, value)}

Returns the first derivative of \textit{expression} with respect to \textit{variable} at the real or complex \textit{value}.

\[ \text{der1}(x^3,x,5) \]

\[ \texttt{der1(x^3,x,5)} \hspace{2em} 75 \]

\textbf{der1(} \textit{expression, variable)}

Uses the current value of \textit{variable}.

\[ \text{der1}(x^3,x) \]

\[ \texttt{der1(x^3,x)} \hspace{2em} 27 \]

\textbf{der1(} \textit{expression, variable, list)}

Returns a list containing the first derivatives at the values specified by the elements in \textit{list}.

\[ \text{der1}(x^3,x,{5,3}) \]

\[ \texttt{der1(x^3,x,{5,3})} \hspace{2em} \{75 \text{ 27}\} \]

\textbf{der2(} \textit{expression, variable, value)}

Returns the second derivative of \textit{expression} with respect to \textit{variable} at the real or complex \textit{value}.

\[ \text{der2}(x^3,x,5) \]

\[ \texttt{der2(x^3,x,5)} \hspace{2em} 30 \]

\textbf{der2(} \textit{expression, variable)}

Uses the current value of \textit{variable}.

\[ \text{der2}(x^3,x) \]

\[ \texttt{der2(x^3,x)} \hspace{2em} 18 \]

\textbf{der2(} \textit{expression, variable, list)}

Returns a list containing the second derivatives at the values specified by the elements in \textit{list}.

\[ \text{der2}(x^3,x,{5,3}) \]

\[ \texttt{der2(x^3,x,{5,3})} \hspace{2em} \{30 \text{ 18}\} \]
**det**

MATRX MATH menu

Returns the determinant of `squareMatrix`. The result is real for a real matrix, complex for a complex matrix.

\[ \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \]

\[ \text{det} \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \] \[ \begin{bmatrix} 1 \\ 2 \end{bmatrix} \]

**dim**

MATRX OPS menu

VECTR OPS menu

Returns a list containing the dimensions (number of rows and columns) of a real or complex matrix.

\[ \begin{bmatrix} 2 & 7 & 1 \\ 8 & 0 & 1 \end{bmatrix} \]

\[ \text{dim} \begin{bmatrix} 2 & 7 & 1 \\ 8 & 0 & 1 \end{bmatrix} \] \[ \{2, 3\} \]

Returns the length (number of elements) of a real or complex vector.

\[ \text{dim} \{8, 0, 1\} \] \[3\]

**DifEq**

† mode screen

Sets differential equation graphing mode.

\[ \begin{bmatrix} 2 & 7 & 0 \\ 8 & 0 & 1 \end{bmatrix} \]

\[ \text{dim} \begin{bmatrix} 2 & 7 & 0 \\ 8 & 0 & 0 \end{bmatrix} \] \[ \begin{bmatrix} 2 & 7 \\ 8 & 0 \end{bmatrix} \] \[ \{3, 3\} \] \[ \text{dim} \begin{bmatrix} 2 & 7 & 0 \\ 8 & 0 & 0 \end{bmatrix} \] \[ \begin{bmatrix} 2 & 7 \\ 8 & 0 \end{bmatrix} \]

If `matrixName` does not exist, creates a new matrix with the specified dimensions and fills it with zeros.

If `matrixName` exists, redimensions that matrix to the specified dimensions. Existing elements within the new dimensions are not changed; elements outside the new dimensions are deleted. If additional elements are created, they are filled with zeros.
#ofElements\textbf{dim }\textit{vectorName}

If \textit{vectorName} does not exist, creates a new vector with the specified \#ofElements and fills it with zeros.

If \textit{vectorName} exists, redimensions that vector to the specified \#ofElements. Existing elements within the new dimension are not changed; elements outside the new dimension are deleted. If additional elements are created, they are filled with zeros.

\textbf{dimL}

\textbf{LIST OPS menu}

\textbf{dimL list}

Returns the length (number of elements) of a real or complex list.

\textbf{dirFld}

\[\text{DirFld}^\dagger\text{ graph format screen (scroll down to second screen)}\]

\[\text{DirFld}^\dagger\text{ graph format screen (scroll down to secondscreen)}\]

\[\text{DirFld}^\dagger\text{ graph format screen (scroll down to secondscreen)}\]

\[\text{DirFld}^\dagger\text{ graph format screen (scroll down to secondscreen)}\]

\[\text{DirFld}^\dagger\text{ graph format screen (scroll down to secondscreen)}\]

\[\text{DirFld}^\dagger\text{ graph format screen (scroll down to secondscreen)}\]

\[\text{DirFld}^\dagger\text{ graph format screen (scroll down to secondscreen)}\]
Chapter 20: A to Z Function and Instruction Reference

**Disp**

† program editor

I/O menu

**Disp** valueA, valueB, valueC, ...

Displays each value. The values can include strings and variable names.

**Disp**

Displays the home screen.

---

**DispG**

† GRAPH menu

† program editor

I/O menu

**DispG**

Displays the current graph.

Function names are case-sensitive. Use y1, not Y1.

To select from a list of window variable names, press [2nd] [CATLG-VARS] [MORE] [MORE] [FS].

Program segment in **Func** graphing mode:

```
10→x ENTER 10
Disp x^3+3x-6 ENTER 1024
"Hello"→STR ENTER
Disp STR+", Jan" ENTER
Hello, Jan
```

```
10→x
Disp x^3+3x-6

"Hello"→STR
Disp STR+", Jan"
Hello, Jan
```

```
Program segment in Func graphing mode:

\( y_1 = 4\cos x \)
\( x_{\text{Min}}: 10 \)
\( x_{\text{Max}}: 5 \)
\( y_{\text{Min}}: 5 \)
\( y_{\text{Max}}: 5 \)
```

```
:DispG
```

```
```

---

```
```
DispT

Displays the table.

Function names are case-sensitive. Use y1, not Y1.

DispT

Program segment in Func graphing mode:

\[ y1 = 4 \cos x \]
\[ \text{DispT} \]

DispT

Program segment in Func graphing mode:

\[ y1 = 4 \cos x \]
\[ \text{DispT} \]

Division: /

- \( \text{numberA/numberB} \) or \( (\text{expressionA})/(\text{expressionB}) \)
  - Returns one argument divided by another. The arguments can be real or complex.
  - \( \frac{-98}{4} \) \text{ ENTER} \quad -24.5
  - \( \frac{-98}{(4 \times 1)} \) \text{ ENTER} \quad -8.1666666667

- \( \text{number/list or (expression)/list} \)
  - Returns a list in which each element is \text{number} or \text{expression} divided by the corresponding element in \text{list}.
  - \( \frac{100}{(10,25,2)} \) \text{ ENTER} \quad \{10 \ 4 \ 50\}

- \( \text{list/number or list/(expression)} \)
  - Returns a list or vector in which each element of \text{list} or \text{vector} is divided by \text{number} or \text{expression}.
  - \( \frac{[120,92,8]}{4} \) \text{ ENTER} \quad \{30 \ 23 \ 2\}

- \( \text{vector/number or vector/(expression)} \)
  - Returns a list or vector in which each element of \text{list} or \text{vector} is divided by \text{number} or \text{expression}.
  - \( \frac{[8,1,(5,2)]}{2} \) \text{ ENTER} \quad \{(4,0), (5,0), (2.5,1,\ldots\}

- \( \text{listA/listB} \)
  - Returns a list in which each element of \text{listA} is divided by the corresponding element of \text{listB}. The lists must have the same dimension.
  - \( \frac{[1,2,3]}{[4,5,6]} \) \text{ ENTER} \quad \{0.25 \ 0.4 \ 0.5\}

- \( \text{listA/listB} \)
  - Returns a list in which each element of \text{listA} is divided by the corresponding element of \text{listB}. The lists must have the same dimension.
  - \( \frac{[1,2,3]}{[4,5,6]} \) \text{ ENTER} \quad \{0.25 \ 0.4 \ 0.5\}
DMS entry: ’
MATH ANGLE menu

In a trig calculation, the result of a DMS entry is treated as degrees in the Degree angle mode only. It is treated as radians in Radian angle mode.

Designates the entered angle is in DMS format. degrees (≤ 999,999), minutes (< 60), and seconds (< 60, may have decimal places) must be entered as real numbers, not as variable names or expressions.

Do not use º and ’ symbols to specify degrees and seconds. For example, 5º59’ is interpreted as implied multiplication of 5° * 59' according to the current angle mode setting.

54°32’30”
In Degree angle mode:
cos 54°32’30” [ENTER] 0.580110760699
In Radian angle mode:
cos 54°32’30” [ENTER] -0.422502666138

Do not use the following notation; in Degree angle mode:
5°59’ [ENTER] 295

angle→DMS
MATH ANGLE menu

Displays angle in DMS format. The result is shown in degrees’minutes’seconds” format, even though you use degrees’minutes’seconds’ to enter a DMS angle.

In Degree angle mode:
45.371°→DMS [ENTER] 45°22’15.6”
54°32’30”→2 [ENTER] 109.083333333
Ans→DMS [ENTER] 109°5’0”

dot(vectorA,vectorB)
VECTR MATH menu

Returns the dot product of two real or complex vectors.

dot([a,b,c],[d,e,f]) returns a*d+b*e+c*f.

dot(vectorA,vectorB) [ENTER] 32

DrawDot
† graph format screen

Sets dot graphing format.

DrawDot [ENTER]
<table>
<thead>
<tr>
<th><strong>DrawF</strong></th>
<th><strong>DrawF expression</strong></th>
<th><strong>In Func graphing mode:</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>GRAPH DRAW menu</td>
<td>Draws expression (in terms of $x$) on the current graph.</td>
<td>ZStd:DrawF 1.25 $x \cos x$ [ENTER]</td>
</tr>
</tbody>
</table>

**DrawLine**<br>† graph format screen<br>

**DrawLine**<br>Sets connected line graphing format.
DrEqu(† GRAPH menu

To enter the ‘ character for the Q’ variables, use the CHAR MISC menu.

DrEqu(xAxisVariable,yAxisVariable,xList,yList,tList)

In DiffEq graphing mode, draws the solution to a set of differential equations stored in the Q’ variables specified by xAxisVariable and yAxisVariable. If direction fields are off (FldOff is selected), the initial values must be stored also.

After the solution is drawn, DrEqu waits for you to move the cursor to a new initial value and press ENTER to draw the new solution.

You then are prompted to press Y (to specify another initial value) or N (to stop).

For the last-drawn solution, the x, y, and t values (beginning at their initial values) are stored to xList, yList, and tList, respectively.

DrEqu(xAxisVariable,yAxisVariable)

Does not store x, y, and t values for the solution.

DrInv

GRAPH DRAW menu

DrInv expression

Draws the inverse of expression by plotting x values on the y-axis and y values on the x-axis.

In DiffEq graphing mode, starting with a ZStd graph screen:

Q’1=Q2:Q’2=Q1 ENTER Done
0→ tMin:Q11:0→ Q12 ENTER 0
DrEqu(Q1,Q2,XL,YL,TL) ENTER

Move the cursor to a new initial value.

Press N to stop graphing. You can then examine XL, YL, and TL.

In Func graphing mode:

ZStd:DrInv 1.25 x cos x ENTER

Press N to stop graphing. You can then examine XL, YL, and TL.
Chapter 20: A to Z Function and Instruction Reference

DS<(  
‡ program editor
CTL menu

:DS<(variable,value)
:command-if-variable=value
:commands

Decrement variable by 1. If the result is < value, skips
command-if-variable=value.
If the result is ≥ value, then command-if-variable=value
is executed.
variable cannot be a built-in variable.

dxDer1  
† mode screen

Sets der1 as the current differentiation type. der1
differentiates exactly and calculates the value for each
function in an expression. It is more accurate than
dxNDer, but more restrictive in that only certain
functions are valid in the expression.

dxNDer  
† mode screen

Sets nDer as the current differentiation type. nDer
differentiates numerically and calculates the value for
an expression. It is less accurate than dxDer1, but less
restrictive in the functions that are valid in the
expression.

e^  

2nd [e^]

e^0 [ENTER] 1

 Returns e raised to power or expression. The argument
can be real or complex.
**e^list**

Returns a list in which each element is \( e \) raised to the power specified by the corresponding element in \( list \).

\[ e^{1,0,.5} \]

```
\{2.71828182846 1 1.6...
```

**e^squareMatrix**

Returns a square matrix that is the matrix exponential of \( squareMatrix \). The matrix exponential corresponds to the result calculated using power series or Cayley-Hamilton Theorem techniques. This is not the same as simply calculating the exponential of each element.

\[ e^{\begin{bmatrix}
1 & 2 \\
3 & 4 \\
\end{bmatrix}} \]

```
\begin{bmatrix}
2.06527633610 & 2.06527633610 \\
1.26832065307 & 1.26832065307 \\
\end{bmatrix}
```

**eigVc**

**eigVc squareMatrix**

Returns a matrix containing the eigenvectors for a real or complex \( squareMatrix \), where each column in the result corresponds to an eigenvalue. The eigenvectors of a real matrix may be complex. Note that an eigenvector is not unique; it may be scaled by any constant factor. TI-86 eigenvectors are normalized.

\[ eigVc \begin{bmatrix}
1 & 2 \\
3 & 4 \\
\end{bmatrix} \]

```
\begin{bmatrix}
\{.800906446592,0\} & \{-.484028886343,0\} & \{-.352512270699,0\}
\end{bmatrix}
```

**eigVI**

**eigVI squareMatrix**

Returns a list of the eigenvalues of a real or complex \( squareMatrix \). The eigenvalues of a real matrix may be complex.

\[ eigVI \begin{bmatrix}
1 & 2 \\
3 & 4 \\
\end{bmatrix} \]

```
\{-4.40941084667,0\} ...
```
Else
‡ program editor
CTL menu

Refer to syntax information for If, beginning on page 305. See the If:Then:Else:End syntax.

End
‡ program editor
CTL menu

End

Identifies the end of a While, For, Repeat, or If-Then-Else loop.

Eng
† mode screen

Eng

Sets engineering notation mode, in which the power-of-10 exponent is a multiple of 3.

In Eng notation mode:
123456789 ENTER  123.456789E6

In Normal notation mode:
123456789 ENTER  123456789

Eq→St(equationVariable,stringVariable)
STRNG menu

Converts the contents of equationVariable to a string and stores it to stringVariable. Be sure to specify an equation variable, not an equation.
To create an equation variable, use an equal sign (=) to define the variable. For example, enter A=B*C, not B*C→A.

Example of = treated as N, where 4=6+1 is evaluated as 4-(6+1):

4=6+1 ENTER  -3

For true/false comparison, use == instead:

4==6+1 ENTER  0

Equal: =

Refer to syntax information for Assignment on page 270.

If you use = in an expression in which the first argument is not a variable name at the beginning of a line, the = is treated as -(.)

Example of = treated as -(.), where 4=6+1 is evaluated as -(6+1):

4=6+1 ENTER  -3

For true/false comparison, use == instead:

4==6+1 ENTER  0
Equal to: ==

等于，**==**

**TEST menu**

The **==** operator is used to compare arguments, while = is used to assign a value or expression to a variable.

Tests whether the condition `argumentA == argumentB` is true or false. Numbers, matrices, and vectors can be real or complex. If complex, the magnitude (modulus) of each element is compared. Strings are case-sensitive.

- If true (`argumentA = argumentB`), returns 1.
- If false (`argumentA ≠ argumentB`), returns 0.

<table>
<thead>
<tr>
<th>expression</th>
<th>result</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>2+2==2+2</code></td>
<td>1</td>
</tr>
<tr>
<td><code>2+(2==2)+2</code></td>
<td>5</td>
</tr>
<tr>
<td><code>[1,2]==[3-2,1+3]</code></td>
<td>1</td>
</tr>
<tr>
<td><code>&quot;A&quot;==&quot;a&quot;</code></td>
<td>0</td>
</tr>
<tr>
<td><code>{1,5,9}=={1,-6,9}</code></td>
<td><code>{1,0,1}</code></td>
</tr>
</tbody>
</table>

**Euler**

**† graph format screen (scroll down to second screen)**

**Euler**

In **DifEq** graphing mode, uses an algorithm based on the Euler method to solve differential equations. Typically, **Euler** is less accurate than **RK** but finds the solutions much quicker.

**eval**

**MATH MISC menu**

**eval xValue**

Returns a list containing the y values of all defined and selected functions evaluated at a real xValue.

Remember that built-in equation variables y1 and y2 are case-sensitive:

<table>
<thead>
<tr>
<th>expression</th>
<th>result</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>y1=x^3+x+5</code></td>
<td>Done</td>
</tr>
<tr>
<td><code>y2=2 x</code></td>
<td>Done</td>
</tr>
<tr>
<td><code>eval 5</code></td>
<td><code>{135 10}</code></td>
</tr>
</tbody>
</table>
evalF(expression, variable, value)

Returns the value of expression evaluated with respect to variable at a real or complex value.

Example:

\[ \text{evalF}(x^3 + x + 5, x, 5) \]

\[ 135 \]

evalF(expression, variable, list)

Returns a list containing the values of expression evaluated with respect to variable at each element in list.

Example:

\[ \text{evalF}(x^3 + x + 5, x, \{3, 5\}) \]

\[ \{35, 135\} \]

Exponent: E

number \( \text{E} \) power or \( (\text{expressionA}) \text{E} (\text{expressionB}) \)

Returns a real or complex number raised to the power of 10, where power is a real integer such that \(-999 < \text{power} < 999\). Any expressions must evaluate to appropriate values.

Example:

\[ 12.3456789 \times 10^{5} \]

\[ 1234567.89 \]

\[ (1.78/2.34) \times 10^{2} \]

\[ 76.0683760684 \]

list \( \text{E} \) power or \( \text{listE} (\text{expression}) \)

Returns a list in which each element is the corresponding element in list raised to the power of 10.

Example:

\[ \text{listE} (6.34, 854.6) \times 10^{3} \]

\[ \{6340, 854600\} \]
**ExpR**

**STAT CALC menu**

Built-in equation variables such as $y_1$, $r_1$, and $x_1$ are case-sensitive. Do not use $Y1$, $R1$, and $XT1$.

**ExpR**

\[\text{ExpR } xList, yList, \text{frequencyList}, \text{equationVariable} \]

Fits an exponential regression model ($y = ab^x$) to real data pairs in $xList$ and $yList$ ($y$ values must be > 0) and frequencies in $\text{frequencyList}$. The regression equation is stored to $\text{equationVariable}$, which must be a built-in equation variable such as $y_1$, $r_1$, and $x_1$.

Values used for $xList$, $yList$, and $\text{frequencyList}$ are stored automatically to built-in variables $xStat$, $yStat$, and $fStat$, respectively. The regression equation is stored also to built-in equation variable $\text{RegEq}$.

**ExpR**

\[\text{ExpR } xList, yList, \text{equationVariable} \]

Uses frequencies of 1.

**ExpR**

\[\text{ExpR } xList, yList, \text{frequencyList} \]

Stores the regression equation to $\text{RegEq}$ only.

**ExpR**

\[\text{ExpR } xList, yList \]

Uses frequencies of 1, and stores the regression equation to $\text{RegEq}$ only.

**ExpR**

\[\text{ExpR } \text{equationVariable} \]

Uses $xStat$, $yStat$, and $fStat$ for $xList$, $yList$, and $\text{frequencyList}$, respectively. These built-in variables must contain valid data of the same dimension; otherwise, an error occurs. The regression equation is stored to $\text{equationVariable}$ and $\text{RegEq}$.

In Func graphing mode:

\[
\begin{align*}
\{1,2,3,4,5\} & \rightarrow L1 \ \ \ \text{ENTER} \\
\{1,20,55,230,742\} & \rightarrow L2 \ \ \ \text{ENTER} \\
\text{ExpR } L1, L2, y1 & \ \ \ \text{ENTER}
\end{align*}
\]

\[
\text{Plot1}(L1, L2) \ \ \ \text{ENTER} \quad \text{Done}
\]

\[
\text{ZData} \ \ \ \text{ENTER}
\]

---

In the context of the TI-86 calculator:

\[
\begin{align*}
\text{ExpR } L1, L2, y1 & \ \ \ \text{ENTER}
\end{align*}
\]

This command fits an exponential regression model to the data pairs in lists $L1$ and $L2$, where $L2$ contains $y$ values greater than 0. The regression equation is stored in the built-in equation variables $y1$, $r1$, and $x_1$. Values used for $xList$, $yList$, and $\text{frequencyList}$ are stored automatically to the built-in variables $xStat$, $yStat$, and $fStat$, respectively. The regression equation is also stored to the built-in equation variable $\text{RegEq}$. In Func graphing mode, the command is executed with the specified lists, and the regression equation is plotted and stored according to the graph settings.
ExpR

Uses \texttt{xStat}, \texttt{yStat}, and \texttt{fStat}, and stores the regression equation to \texttt{RegEq} only.

\textbf{Factorial: !}

\texttt{MATH PROB menu}

\begin{align*}
\text{number!} & \quad \text{or} \quad \text{(expression)!} \\
6! & \quad \text{ENTER} \quad 720 \\
12.5! & \quad \text{ENTER} \quad 1710542068.32 \\
\end{align*}

\text{Returns the factorial of a real integer or non-integer, where } 0 \leq \text{integer} \leq 449 \text{ and } 0 \leq \text{non-integer} \leq 449.9. \text{ For a non-integer, the Gamma function is used to find the factorial. An expression must evaluate to an appropriate value.}

\textit{list!}

\text{Returns a list in which each element is the factorial of the corresponding element in list.}

\texttt{fcstx}

\texttt{† STAT menu}

\textit{fcstx yValue}

Based on the current regression equation (\texttt{ReqEq}), returns the forecasted \texttt{x} at a real \texttt{yValue}.

\texttt{fcsty}

\texttt{† STAT menu}

\textit{fcsty xValue}

Based on the current regression equation (\texttt{ReqEq}), returns the forecasted \texttt{y} at a real \texttt{xValue}.
**Chapter 20: A to Z Function and Instruction Reference**

**Fill**

**LIST OPS menu**

**MATRX OPS menu**

**VECTR OPS menu**

- **Fill(number,listName)**
- **Fill(number,matrixName)**
- **Fill(number,vectorName)**

Replaces each element in an existing listName, matrixName, or vectorName with a real or complex number.

- `{3,4,5}`
- `L1`
- `b` `{3 4 5}`
- `Fill((3,4),L1)`
- `{3 4 5}`
- `L1` `b` `{3 4 5}`

**Fix**

† mode screen

**Fix** or **Fix(expression)**

Sets fixed decimal mode for integer number of decimal places, where 0 ≤ integer ≤ 11. An expression must evaluate to an appropriate integer.

- `Fix 3` `n/2` `π/2`
- `Done` 
- `In Radian angle mode:
- `Fix 11` `sin (π/6)`
- `Done`
- `Float` `sin (π/6)`
- `Done`
- `Float` `Done`
- `Float` `.5`

**FldOff**

† graph format screen (scroll down to second screen)

**FldOff**

In **DifEq** graphing mode, turns off the slope and direction fields. To turn on slope fields, use **SlpFid**. To turn on direction fields, use **DirFid**.

- `In Radian angle mode:`
- `Fix 11` `sin (π/6)`
- `Done`
- `Float` `Done`
- `Float` `.5`
### Chapter 20: A to Z Function and Instruction Reference

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>**fMax(<strong><em>expression</em>, <em><strong>variable</strong></em>, <em><strong>lower</strong></em>, <em><strong>upper</strong></em>)</strong></td>
<td>Returns the value at which a local maximum of <em>expression</em> with respect to <em>variable</em> occurs, between real <em>lower</em> and <em>upper</em> values for <em>variable</em>. The tolerance is controlled by the built-in variable <em>tol</em>, whose default is 1E-5. To view or set <em>tol</em>, press <strong>2nd</strong> [MEM] [F4] to display the tolerance editor.</td>
<td><code>fMax(sin x, x, ς, π)</code> &lt;br&gt; 1.57079632598</td>
</tr>
<tr>
<td>**fMin(<strong><em>expression</em>, <em><strong>variable</strong></em>, <em><strong>lower</strong></em>, <em><strong>upper</strong></em>)</strong></td>
<td>Returns the value at which a local minimum of <em>expression</em> with respect to <em>variable</em> occurs, between real <em>lower</em> and <em>upper</em> bounds for <em>variable</em>. The tolerance is controlled by the built-in variable <em>tol</em>, whose default is 1E-5. To view or set <em>tol</em>, press <strong>2nd</strong> [MEM] [F4] to display the tolerance editor.</td>
<td><code>fMin(sin x, x, ς, π)</code> &lt;br&gt; -1.57079632691</td>
</tr>
<tr>
<td>**fnInt(<strong><em>expression</em>, <em><strong>variable</strong></em>, <em><strong>lower</strong></em>, <em><strong>upper</strong></em>)</strong></td>
<td>Returns the numerical function integral of <em>expression</em> with respect to <em>variable</em>, between real <em>lower</em> and <em>upper</em> bounds for <em>variable</em>. The tolerance is controlled by the built-in variable <em>tol</em>, whose default is 1E-5. To view or set <em>tol</em>, press <strong>2nd</strong> [MEM] [F4] to display the tolerance editor.</td>
<td><code>fnInt(x^2, x, 0, 1)</code> &lt;br&gt; 0.333333333333</td>
</tr>
</tbody>
</table>

**FnOff**<br>† GRAPH VARS menu  <br> Deselects the specified equation function numbers.
**FnOff**

Deselects all equation function numbers.

**FnOn**

† GRAPH VARS menu

Selects the specified equation function numbers, in addition to any others already selected.

**FnOn**

Selects all equation function numbers.

---

**For(**

‡ program editor

CTL menu

Executes the commands in loop iteratively, where the number of repetitions is controlled by variable. The first time through the loop, variable = begin. At the End of the loop, variable is incremented by step. The loop is repeated until variable > end. If you do not specify step, the default is 1.

You can specify values such that begin > end. If so, be sure to specify a negative step.
**Form(***

**LIST OPS menu**

**Form**(*"formula",listName*)

Generates the contents of *listName* automatically, based on the attached *formula*. If you express *formula* in terms of a list, you can generate one list based on the contents of another.

The contents of *listName* are updated automatically if you edit *formula* or edit a list referenced in *formula*.

```
{1,2,3,4}∥L1 ENTER (1 2 3 4)
Form("10*L1",L2) ENTER Done
L2 ENTER (10 20 30 40)
(5,10,15,20)∥L1 ENTER (5 10 15 20)
L2 ENTER (50 100 150 200)
Form("L1/5",L2) ENTER Done
L2 ENTER (1 2 3 4)
```

**fPart**

**MATH NUM menu**

**fPart** *number* or **fPart**(expression)

Returns the fractional part of a real or complex *number* or expression.

```
fPart 23.45 ENTER .45
fPart (-17.268 ENTER) -.08
```

**fPart** list

**fPart** matrix

**fPart** vector

Returns a list, matrix, or vector in which each element is the fractional part of the corresponding element in the specified argument.

```
[[1,-23.45],[-99.5,47.15]]∥MAT ENTER
[[1 -23.45]
 [-99.5 47.15 ]

fPart MAT ENTER [[0 -.45]
 [.5 .15 ]
```

**Frac**

**MATH MISC menu**

**number**∥**Frac**

Displays a real or complex *number* as its rational equivalent, a fraction reduced to its simplest terms.

If *number* cannot be simplified or if the denominator is more than four digits, the decimal equivalent is returned.

```
1/3+2/7 ENTER .619047619048
Ans∥Frac ENTER 13/21
```
list->Frac
matrix->Frac
vector->Frac
Returns a list, matrix, or vector in which each element is the rational equivalent of the corresponding element in the argument.

Func
† mode screen

Func
Sets function graphing mode.

gcd(
MATH MISC menu

gcd(integerA,integerB)
Returns the greatest common divisor of two nonnegative integers.

gcd(listA,listB)
Returns a list in which each element is the gcd of the two corresponding elements in listA and listB.

Get(
† program editor
I/O menu

Get(variable)
Gets data from a CBL or CBR System or another TI-86 and stores it to variable.
getKy

Returns the key code for the last key pressed. If no key has been pressed, getKy returns 0. Refer to the TI-86 key code diagram in Chapter 16.

Program:

```
PROGRAM: CODES
:Lbl TOP
:getKy KEY
:While KEY==0
 :getKy KEY
 :End
 :Disp KEY
 :Goto TOP
```

To break the program, press $\uparrow$ and then $\ast$.

Goto label

Transfers (branches) program control to the label specified by an existing Lbl instruction.

Program segment:

```
:0→TEMP:1→J
:Lbl TOP
:TEMP+J→TEMP
:If J<10
 :Then
 : J+1→J
 : Goto TOP
 :End
 :Disp TEMP
 ::
```

Greater than: >

Tests whether the condition is true or false. The arguments must be real numbers.

- If true ($numberA > numberB$), returns 1.
- If false ($numberA \leq numberB$), returns 0.

<table>
<thead>
<tr>
<th>Expression</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>2&gt;0</td>
<td>ENTER</td>
</tr>
<tr>
<td>88&gt;123</td>
<td>ENTER</td>
</tr>
<tr>
<td>-5&gt;-5</td>
<td>ENTER</td>
</tr>
<tr>
<td>(20<em>5/2)&gt;(18</em>2)</td>
<td>ENTER</td>
</tr>
</tbody>
</table>
number > list

Returns a list of 1s and/or 0s to indicate if number is > the corresponding element in list.

> {1, -6, 10} ENTER

listA > listB

Returns a list of 1s and/or 0s to indicate if each element in listA is > the corresponding element in listB.

{1, 5, 9} > {1, -6, 10} ENTER

Greater than or equal to: ≥

TEST menu

numberA ≥ numberB or (expressionA) ≥ (expressionB)

Tests whether the condition is true or false. The arguments must be real numbers.

- If true (numberA ≥ numberB), returns 1.
- If false (numberA < numberB), returns 0.

number ≥ list

Returns a list of 1s and/or 0s to indicate if number is ≥ the corresponding element in list.

≥ {1, -6, 10} ENTER

listA ≥ listB

Returns a list of 1s and/or 0s to indicate if each element in listA is ≥ the corresponding element in listB.

≥ {1, 5, 9} ≥ {1, -6, 10} ENTER

GridOff

† graph format screen

GridOff

Turns off grid format so that grid points are not displayed.
### GridOn

Turns on grid format so that grid points are displayed in rows and columns corresponding to the tick marks on each axis.

### GrStl(

**CATALOG**

Sets the graph style for `function#`. For `graphStyle#`, specify an integer from 1 through 7:

- `1 = \` (line)
- `2 = \` (thick)
- `3 = \` (above)
- `4 = \` (below)
- `5 = \` (path)
- `6 = \` (animate)
- `7 = `. (dot)

Depending on the graphing mode, some graph styles may not be available.

### h

**BASE TYPE menu**

Designates a real integer as hexadecimal, regardless of the number base mode setting.

### Hex

Sets hexadecimal number base mode. Results are displayed with the h suffix. In any number base mode, you can designate an appropriate value as binary, decimal, hexadecimal, or octal by using the b, d, h, or o designator, respectively, from the BASE TYPE menu. To enter hexadecimal numbers A through F, use the BASE A-F menu. Do not use [ALPHA] to type a letter.
### Hex

**BASE CONV menu**

- **number > Hex**
- **list > Hex**
- **matrix > Hex**
- **vector > Hex**

Returns the hexadecimal equivalent of the real or complex argument.

#### In Bin number base mode:

- \(01001110\) \(\rightarrow\) \(00011000\) \(\rightarrow\) \(8\)h
- \(Ans\) > Hex \(\rightarrow\) \(100,101,110\) \(\rightarrow\) Hex \(\rightarrow\) \(04\)h \(5\)h \(6\)h

### Hist

**† STAT DRAW menu**

**Hist xList, frequencyList**

Draws a histogram on the current graph, using the real data in *xList* and the frequencies in *frequencyList*.

**Hist xList**

Uses frequencies of 1.

**Hist**

Uses the data in built-in variables *xStat* and *fStat*. These variables must contain valid data of the same dimension; otherwise, an error occurs.

Starting with a ZStd graph screen:

- \([1,2,3,4,6,7] \rightarrow XL \rightarrow 123467\)
- \([1,6,4,2,3,5] \rightarrow FL \rightarrow 164235\)
- \(0 \rightarrow \text{Min:} O \rightarrow \text{yMin} \rightarrow 0\)
- Hist XL, FL \rightarrow

```
[1,1,2,2,3,3,3,3,3,3,4,4,5,5,5,7,7] \rightarrow XL \rightarrow [1,1,2,2,3,3,3,3,3,3,4,4,5,5,5,7,7] \rightarrow XL \rightarrow
```

```
CDrw:Hist XL \rightarrow
```
Horiz

† GRAPH DRAW menu

Horiz yValue

Draws a horizontal line on the current graph at yValue.

In a ZStd graph screen:
Horiz 4.5 ENTER

---

IAAsk

CATALOG

IAAsk

Sets the table so that the user can enter individual
values for the independent variable.

IAuto

CATALOG

IAuto

Sets the table so that the TI-86 generates the
independent-variable values automatically, based on
values entered for TblStart and ΔTbl.

Ident

MATRIX OPS menu

Ident dimension

Returns the identity matrix of dimension rows x
dimension columns.

Ident 4 ENTER

[[1 0 0 0]
[0 1 0 0]
[0 0 1 0]
[0 0 0 1]]
If

$‡$ program editor

CTL menu

<table>
<thead>
<tr>
<th>If condition</th>
<th>command-if-true</th>
<th>commands</th>
</tr>
</thead>
</table>

If condition is true, executes command-if-true. Otherwise, skips command-if-true. The condition is true if it evaluates to any nonzero number, or false if it evaluates to zero.

To execute multiple commands if condition is true, use If:Then:End instead.

Program segment:

```
:If x<0
:Disp "x is negative"
:End
```

<table>
<thead>
<tr>
<th>If condition</th>
<th>Then</th>
<th>commands-if-true</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>End</th>
<th>commands</th>
</tr>
</thead>
</table>

If condition is true (nonzero), executes commands-if-true from Then to End. Otherwise, skips commands-if-true and continues with the next command following End.

Program segment:

```
:If x<0
:Then
  :Disp "x is negative"
  :abs(x)►x
:End
```
:If condition
:Then
:commands-if-true
:Else
:commands-if-false
:End
:commands

If condition is true (nonzero), executes commands-if-true from Then to Else and then continues with the next command following End.

If condition is false (zero), executes commands-if-false from Else to End and then continues with the next command following End.

imag (complexNumber)

Returns the imaginary (nonreal) part of complexNumber. The imaginary part of a real number is always 0.

imag (real, imaginary) returns imaginary.
imag (magnitude, angle) returns magnitude sin angle.

imag complexList
imag complexMatrix
imag complexVector

Returns a list, matrix, or vector in which each element is the imaginary part of the original argument.
**InpSt**

‡ program editor  
I/O menu

**InpSt** _promptString, variable_

Pauses a program, displays _promptString_, and waits for the user to enter a response. The response is stored to _variable_ always as a string. When entering the response, the user should not enter quotation marks.

To prompt for a number or expression instead of a string, use **Input**.

**InpSt** _variable_

Displays ? as the prompt.

---

**Input**

‡ program editor  
I/O menu

**Input** _promptString, variable_

Pauses a program, displays _promptString_, and waits for the user to enter a response. The response is stored to _variable_ in the form in which the user enters it.

- A number or expression is stored as a number or expression.
- A list, vector, or matrix is stored as a list, vector, or matrix.
- An entry enclosed in " marks is stored as a string.

**Input** _variable_

Displays ? as the prompt.
Input

Pauses a program, displays the graph screen, and lets the user update $x$ and $y$ (or $r$ and $\theta$ in PolarGC graph format) by moving the free-moving cursor. To resume the program, press [ENTER].

Input "CBLGET",variable

Receives list data sent from a CBL or CBR System and stores it to variable on the TI-86. Use this "CBLGET" syntax for both CBL and CBR.

You can receive data also by using Get( as described on page 299.

int

MATH NUM menu

int number  or  int (expression)  int 23.45  ENTER  23

Returns the largest integer $\leq$ number or expression. The argument can be real or complex.

For a negative non-integer, int returns the integer that is one less than the integer part of the number. To return the exact integer part, use iPart instead.

int list

[[1.25,-23.45],[-99,47.15]]$\rightarrow$MAT

int matrix

[[1.25,-23.45]

[-99  47.15 ]]

int vector

[[1  -24]

[-99  47 ]]
### inter(

<table>
<thead>
<tr>
<th>inter( (x_1,y_1,x_2,y_2,x\text{Value}))</th>
<th>Calculates the line through points ((x_1,y_1)) and ((x_2,y_2)) and then interpolates or extrapolates a (y) value for the specified (x\text{Value}).</th>
</tr>
</thead>
<tbody>
<tr>
<td>Using points (3,5) and (4,4), find the (y) value at (x=1):</td>
<td>(\text{inter}(3,5,4,4,1)) ENTER 7</td>
</tr>
<tr>
<td>(\text{inter}(x_1,y_1,x_2,y_2,y\text{Value}))</td>
<td>Interpolates or extrapolates an (x) value for the specified (y\text{Value}). Notice that points ((x_1,y_1)) and ((x_2,y_2)) must be entered as ((y_1,x_1)) and ((y_2,x_2)).</td>
</tr>
<tr>
<td>Using points ((-4,-7)) and ((2,6)), find the (x) value at (y=10):</td>
<td>(\text{inter}(-4,-7,2,6,10)) ENTER 3.84615384615</td>
</tr>
</tbody>
</table>

### Inverse: \(^{-1}\)

<table>
<thead>
<tr>
<th>(\text{number}^{-1}) or (\text{expression}^{-1})</th>
<th>Returns 1 divided by a real or complex (\text{number}), where (\text{number} \neq 0).</th>
</tr>
</thead>
<tbody>
<tr>
<td>5 (^{-1}) ENTER</td>
<td>.2</td>
</tr>
<tr>
<td>((10\times6)^{-1}) ENTER</td>
<td>.01666666667</td>
</tr>
<tr>
<td>(\text{list}^{-1})</td>
<td>Returns a list in which each element is 1 divided by the corresponding element in (\text{list}).</td>
</tr>
<tr>
<td>([-.5,10,2/8]^{-1}) ENTER</td>
<td>([-2 \times 1.4])</td>
</tr>
<tr>
<td>(\text{squareMatrix}^{-1})</td>
<td>Returns an inverted (\text{squareMatrix}), where det (\neq 0).</td>
</tr>
<tr>
<td>(\text{[[1,2][3,4]]}^{-1}) ENTER</td>
<td>([[3,2] [1.5 - .5]])</td>
</tr>
</tbody>
</table>

### iPart

<table>
<thead>
<tr>
<th>iPart (\text{number}) or iPart ((\text{expression}))</th>
<th>Returns the integer part of (\text{number or expression}). The argument can be real or complex.</th>
</tr>
</thead>
<tbody>
<tr>
<td>iPart 23.45 ENTER</td>
<td>23</td>
</tr>
<tr>
<td>iPart -23.45 ENTER</td>
<td>-23</td>
</tr>
</tbody>
</table>
iPart list, iPart matrix, iPart vector

Returns a list, matrix, or vector in which each element is the integer part of the corresponding element in the specified argument.

\[
\begin{bmatrix}
1.25, & 23.45 \\
99.5, & 47.15
\end{bmatrix}
\]  
\[\text{iPart MAT ENTER} \]
\[\begin{bmatrix}
1, & 23 \\
99, & 47
\end{bmatrix}
\]

IS>(

‡ program editor  CTL menu

\[\text{IS>}(\text{variable}, \text{value})\]
\[\text{xcommand-if-variablesvalue}\]
\[\text{xcommands}\]

Increments variable by 1. If the result is \(>\) value, skips command-if-variablesvalue.
If the result is \(\leq\) value, then command-if-variablesvalue is executed.

variable cannot be a built-in variable.

LabelOff

‡ graph format screen

Turns off axes labels.

LabelOn

‡ graph format screen

Turns on axes labels.
Lbl
‡ program editor
CTL menu

Lbl label

Creates a label of up to eight characters. A program can use a Goto instruction to transfer control (branch) to a specified label.

InpSt stores input as a string, so be sure to store a string to the password variable.

Program segment, assuming a correct password has already been stored to the password variable:

:Lbl Start
:InpSt "Enter password:",PSW
:If PSW≠password
:Goto Start
:Disp "Welcome"

lcm(integerA, integerB)

MATH MISC menu

Returns the least common multiple of two nonnegative integers.

lcm(5, 2)
\[ \text{lcm}(5,2) = 10 \]
lcm(6, 9)
\[ \text{lcm}(6,9) = 18 \]
lcm(18, 33)
\[ \text{lcm}(18,33) = 198 \]

LCust(item#, "title", item#, "title", ...)

‡ program editor
CTL menu

LCust loads (defines) the TI-86's custom menu, which is displayed when the user presses CUSTOM. The menu can have up to 15 items, shown in three groups of five items. For each item#/title pair:

• item# — integer from 1 through 15 that identifies the item's position in the menu. The item numbers must be specified in order, but you can skip numbers.

• "title" — string with up to 8 characters (not counting the quotes) that will be pasted to the current cursor location when the item is selected. This can be a variable name, expression, function name, program name, or any text string.

Program segment:


After executed and when the user presses CUSTOM:
Less than: <

TEST menu

\(\text{numberA} < \text{numberB} \) or \((\text{expressionA}) < (\text{expressionB})\)

Tests whether the condition is true or false. The arguments must be real numbers.

- If true \((\text{numberA} < \text{numberB})\), returns 1.
- If false \((\text{numberA} \geq \text{numberB})\), returns 0.

2<0 \(\text{ENTER}\) 0
88<123 \(\text{ENTER}\) 1
-5<5 \(\text{ENTER}\) 0
\((20+5/2) < (18+3)\) \(\text{ENTER}\) 1

number<list

Returns a list of 1s and/or 0s to indicate if \(\text{number}\) is < the corresponding element in \(\text{list}\).

1<(1,-6,10) \(\text{ENTER}\) \{0 0 1\}

listA < listB

Returns a list of 1s and/or 0s to indicate if each element in \(\text{listA}\) is < the corresponding element in \(\text{listB}\).

\{(1,5,9) < (1,-6,10) \(\text{ENTER}\) \{0 0 1\\}

Less than or equal to: \(\leq\)

TEST menu

\(\text{numberA} \leq \text{numberB} \) or \((\text{expressionA}) \leq (\text{expressionB})\)

Tests whether the condition is true or false. The arguments must be real numbers.

- If true \((\text{numberA} \leq \text{numberB})\), returns 1.
- If false \((\text{numberA} > \text{numberB})\), returns 0.

2\leq0 \(\text{ENTER}\) 0
88\leq123 \(\text{ENTER}\) 1
-5\leq5 \(\text{ENTER}\) 1
\((20+5/2) \leq (18+3)\) \(\text{ENTER}\) 1

number\leq list

Returns a list of 1s and/or 0s to indicate if \(\text{number}\) is \(\leq\) the corresponding element in \(\text{list}\).

1\leq(1,-6,10) \(\text{ENTER}\) \{1 0 1\}

listA \leq listB

Returns a list of 1s and/or 0s to indicate if each element in \(\text{listA}\) is \(\leq\) the corresponding element in \(\text{listB}\).

\{(1,5,9) \leq (1,-6,10) \(\text{ENTER}\) \{1 0 1\\}

LgstR

STAT CALC menu

Built-in equation variables such as \( y_1 \), \( r_1 \), and \( xt1 \) are case-sensitive. Do not use \( Y1 \), \( R1 \), and \( XT1 \).

LgstR returns a tolMet value that indicates if the result meets the TI-86's internal tolerance.
- If \( \text{tolMet} = 1 \), the result is within the internal tolerance.
- If \( \text{tolMet} = 0 \), the result is outside the internal tolerance, although it may be useful for general purposes.

\[ \text{LgstR} \{ \text{iterations}, \text{xList}, \text{yList}, \text{frequencyList}, \text{equationVariable} \} \]

Fits a logistic regression model \( y = a/(1 + be^{cx} + d) \) to real data pairs in \( \text{xList} \) and \( \text{yList} \) and frequencies in \( \text{frequencyList} \). The regression equation is stored to \( \text{equationVariable} \), which must be a built-in equation variable such as \( y_1 \), \( r_1 \), and \( xt1 \). The equation's coefficients always are stored as a list to built-in variable \( PRegC \).

The number of \( \text{iterations} \) is optional. If omitted, 64 is the default. A large number of \( \text{iterations} \) may produce more accurate results but may require longer calculation times. A smaller number may produce less accurate results but with shorter calculation times.

Values used for \( \text{xList} \), \( \text{yList} \), and \( \text{frequencyList} \) are stored automatically to built-in variables \( \text{xStat} \), \( \text{yStat} \), and \( \text{fStat} \), respectively. The regression equation is stored also to built-in equation variable \( \text{RegEq} \).

\[ \text{In Func graphing mode:} \]

\[ \{1,2,3,4,5,6\} \rightarrow \text{L1 ENTER} \]
\[ \{1,1.3,2.5,3.5,4.5,4.8\} \rightarrow \text{L2 ENTER} \]
\[ \text{LgstR L1,L2,y1 ENTER} \]

\[ \text{Plot1(L1,L2)} \quad \text{Done} \]

\[ \text{ZData ENTER} \]

\[ \text{LgstR \{iterations, xList, yList, frequencyList, equationVariable\}} \]

Uses frequencies of 1.

\[ \text{LgstR \{iterations, xList, yList, frequencyList\}} \]

Stores the regression equation to \( \text{RegEq} \) only.

\[ \text{LgstR \{iterations, xList, yList\}} \]

Uses frequencies of 1, and stores the regression equation to \( \text{RegEq} \) only.
LgstR [iterations,equationVariable]
Uses xStat, yStat, and fStat for xList, yList, and frequencyList, respectively. These built-in variables must contain valid data of the same dimension; otherwise, an error occurs. The regression equation is stored to equationVariable and RegEq.

LgstR [iterations]
Uses xStat, yStat, and fStat, and stores the regression equation to RegEq only.

Line(† GRAPH DRAW menu)
Line(x1,y1,x2,y2)
Draws a line from point (x1,y1) to (x2,y2).

In Func graphing mode and a ZStd graph screen:
Line(-2,-7,9,8) ENTER
LinR  
STAT CALC menu

Built-in equation variables such as $y_1$, $r_1$, and $xt_1$ are case-sensitive. Do not use $Y1$, $R1$, and $XT1$.

LinR $xList, yList, frequencyList, equationVariable$

Fits a linear regression model ($y = a + bx$) to real data pairs in $xList$ and $yList$ and frequencies in $frequencyList$. The regression equation is stored to $equationVariable$, which must be a built-in equation variable such as $y_1$, $r_1$, and $xt_1$.

Values used for $xList$, $yList$, and $frequencyList$ are stored automatically to built-in variables $xStat$, $yStat$, and $fStat$, respectively. The regression equation is stored also to built-in equation variable $RegEq$.

LinR $xList, yList, equationVariable$

Uses frequencies of 1.

LinR $xList, yList, frequencyList$

Stores the regression equation to $RegEq$ only.

LinR $xList, yList$

Uses frequencies of 1, and stores the regression equation to $RegEq$ only.

LinR $equationVariable$

Uses $xStat$, $yStat$, and $fStat$ for $xList$, $yList$, and $frequencyList$, respectively. These built-in variables must contain valid data of the same dimension; otherwise, an error occurs. The regression equation is stored to $equationVariable$ and $RegEq$.

In Func graphing mode:

```
{1, 2, 3, 4, 5, 6}→L1 ENTER
{4.5, 4.6, 6.7, 5.8, 5.8, 7.8}→L2 ENTER
LinR L1, L2, y1 ENTER
```

```
LinReg
y=ax+b
\[a = 3.2133333\]
\[b = 3.57142857\]
\[r^2 = 0.9454755\]
\[n = 6\]
```

```
Plot1(L1, L2) ENTER  Done
ZData ENTER
```

```
LinReg
```

```
LinR y1
```

```
Graph
```

```
Plot1
```

```
ZData
```

```
Done
```
LinR

Uses \textit{xStat}, \textit{yStat}, and \textit{fStat}, and stores the regression equation to \textit{RegEq} only.

List entry: \{ \}

\textbf{Li\textasciitilde\textasciitilde vc}

\textbf{In}

\textbf{lngth}

\textbf{INR}

\textbf{Def}ines a list in which each element is a real or complex number or variable.

\{element1,element2, \ldots \} \rightarrow \{1,2,3\} \rightarrow L1 \rightarrow \{1 2 3\}

\textbf{In RectC} complex number mode:

\{3,(2,4),8\} \rightarrow L2 \rightarrow \{(3,0) (2,4) (16,0)\}

\textbf{Li\textasciitilde\textasciitilde vc list}

Returns a vector converted from a real or complex list.

\textbf{In number or In (expression)}

Returns the natural logarithm of a real or complex number or expression.

\textbf{In list}

Returns a list in which each element is the natural logarithm of the corresponding element in \textit{list}.

\textbf{lngth string}

Returns the length (number of characters) of \textit{string}. The character count includes spaces but not quotation marks.

\textbf{lngth "The answer is:" \rightarrow 14}

\textbf{ln 2 \rightarrow .69314718056}

\textbf{ln (36.4/3) \rightarrow 2.4959648597}

\textbf{In RectC complex number mode:}

\textbf{ln -3 \rightarrow (1.09861228867,3.141...}

\textbf{ln (2,3) \rightarrow \{.69314718056 1.0986...}

\textbf{lngth STR \rightarrow 14}

\textbf{The answer is:" \rightarrow STR \rightarrow \text{Enter} \rightarrow 14}

\textbf{The answer is:

\textbf{lngth STR \rightarrow 14}

\textbf{The answer is:

\textbf{lngth STR \rightarrow 14}
LnR

STAT CALC menu

Built-in equation variables such as y1, r1, and xt1 are case-sensitive. Do not use Y1, R1, and XT1.

LnR xList,yList,frequencyList,equationVariable

Fits a logarithmic regression model (y=a+b ln x) to the real data pairs in xList and yList (x values must be > 0) and frequencies in frequencyList. The regression equation is stored to equationVariable, which must be a built-in equation variable such as y1, r1, and xt1.

Values used for xList, yList, and frequencyList are stored automatically to built-in variables xStat, yStat, and fStat, respectively. The regression equation is stored also to built-in equation variable RegEq.

LnR xList,yList,equationVariable

Uses frequencies of 1.

LnR xList,yList,frequencyList

Stores the regression equation to RegEq only.

LnR xList,yList

Uses frequencies of 1, and stores the regression equation to RegEq only.

LnR equationVariable

Uses xStat, yStat, and fStat for xList, yList, and frequencyList, respectively. These built-in variables must contain valid data of the same dimension; otherwise, an error occurs. The regression equation is stored to equationVariable and RegEq.
LnR

Uses xStat, yStat, and fStat, and stores the regression equation to RegEq only.

log

\[ \text{log } \text{number or log (expression)} \]

Returns the logarithm of a real or complex number or expression, where:

\[ 10^{\text{logarithm}} = \text{number} \]

log list

Returns a list in which each element is the logarithm of the corresponding element in list.

\[ \text{LU(matrix,LMatrixName, UMatrixName, pMatrixName)} \]

Calculates the Crout LU (lower-upper) decomposition of a real or complex matrix. The lower triangular matrix is stored in lMatrixName, the upper triangular matrix in uMatrixName, and the permutation matrix (which describes the row swaps done during the calculation) in pMatrixName. 

\[ l\text{MatrixName} \times u\text{MatrixName} = p\text{MatrixName} \times \text{matrix} \]
Matrix entry:

```
[[row1][row2] ... ]
```

Defines a matrix entered row-by-row in which each element is a real or complex number or variable. Enter each [row] as [element,element, ... ].

```
[1,2,3]
[4,5,6]
```

### max()

**MATH NUM menu**

```
max(numberA,numberB)
```

Returns the larger of two real or complex numbers.

```
max(2.3,1.4)
```

```
2.3
```

```
max(list)
```

Returns the largest element in list.

```
max({1,9,\pi/2,e^2})
```

```
9
```

```
max(listA,listB)
```

Returns a list in which each element is the larger of the corresponding elements in listA and listB.

```
max({1,10},{2,9})
```

```
{2 10}
```

### MBox

**STAT DRAW menu**

```
MBox xList,frequencyList
```

Draws a modified box plot on the current graph, using the real data in xList and the frequencies in frequencyList.

```
MBox xList
```

Uses frequencies of 1.

```
MBox
```

Uses the data in built-in variables xStat and fStat. These variables must contain valid data of the same dimension; otherwise, an error occurs.
Menu

‡ program editor
CTL menu

Generates a menu of up to 15 items during program execution. Menus are displayed as three groups of five items. For each item:

- item# — integer from 1 through 15 that identifies this item's position in the menu.
- "title" — text string that will be displayed for this item on the menu. Typically, use from 1 through 5 characters; additional characters may not be seen on the menu.
- label — valid label to which program execution will branch when the user selects this item.

Example when executed:

```
Radius:
Area is: 78.5398163397
```

Program segment:

```
:Lbl A
:Input "Radius:",RADIUS
:Disp "Area is:",π*RADIUS²
:Menu(1,"Again",A,5,"Stop",B)
:Lbl B
:Disp "The End"
```

min

MATH NUM menu

Returns the smaller of two real or complex numbers.

```
min(3,-5) ENTER -5
min(-5.2,-5.3) ENTER -5.3
min(5,2+2) ENTER 4
```

Returns the smallest element in list.

```
min((1,3,-5)) ENTER -5
```

Returns a list in which each element is the smaller of the corresponding elements in listA and listB.

```
min((1,2,3),(3,2,1)) ENTER [1 2 1]
```

mod

MATH NUM menu

Returns numberA modulo numberB. The arguments must be real.

```
mod(7,0) ENTER 7
mod(7,3) ENTER 1
mod(-7,3) ENTER 2
mod(7,-3) ENTER -2
mod(-7,-3) ENTER -1
```
**mRAdd(**

**MATRX OPS menu**

**mRAdd(number, matrix, rowA, rowB)**

Returns the result of a “multiply and add row” matrix operation, where:

a. rowA of a real or complex matrix is multiplied by a real or complex number.

b. The results are added to (and then stored in) rowB.

**Multiplication: * (×)**

**numberA * numberB**

Returns the product of two real or complex numbers.

- **number * list** or **list * number**
- **number * matrix** or **matrix * number**
- **number * vector** or **vector * number**

Returns a list, matrix, or vector in which each element is number multiplied by the corresponding element in list, matrix, or vector.

**listA * listB**

Returns a list in which each element of listA is multiplied by the corresponding element of listB. The lists must have the same dimension.

**matrix * vector**

Returns a vector in which matrix is multiplied by vector. The number of columns in matrix must equal the number of elements in vector.
matrixA * matrixB

Returns a matrix in which matrixA is multiplied by matrixB. The number of columns in matrixA must equal the number of rows in matrixB.

\[
\begin{bmatrix}
2 & 2 \\
3 & 4 \\
\end{bmatrix} \text{MAT}\text{A ENTER} \quad \begin{bmatrix}
1 & 2 & 3 \\
4 & 5 & 6 \\
\end{bmatrix} \text{MAT}\text{B ENTER} \\
\begin{bmatrix}
10 & 14 & 18 \\
19 & 26 & 33 \\
\end{bmatrix} \text{MAT}\text{A MAT}\text{B ENTER}
\]

multR( number, matrix, row )

Returns the result of a “row multiplication” matrix operation, where:

a. The specified row of a real or complex matrix is multiplied by a real or complex number.

b. The results are stored in the same row.

\[
\begin{bmatrix}
5 & 3 & 1 \\
2 & 0 & 4 \\
3 & -1 & 2 \\
\end{bmatrix} \text{MAT ENTER} \quad \text{multR(5, MAT, 2) ENTER} \quad \begin{bmatrix}
5 & 3 & 1 \\
10 & 20 & \\
3 & -1 & 2 \\
\end{bmatrix}
\]

nCr

Returns the number of combinations of items (n) taken number (r) at a time. Both arguments must be real nonnegative integers.

\[
5 \text{ nCr 2 ENTER} \quad 10
\]
**nDer(**

**CALC menu**

To view or set the value for \( \delta \), press \( \text{Ô} \) \( \text{DEM} \) \( \text{Ô} \) to display the tolerance screen.

**nDer(expression,variable,value)**

Returns an approximate numerical derivative of `expression` with respect to `variable` evaluated at a real or complex `value`. The approximate numerical derivative is the slope of the secant line through the points:

\[(value - \delta, f(value - \delta)) \text{ and } (value + \delta, f(value + \delta))\]

As the step value \( \delta \) gets smaller, the approximation usually gets more accurate.

For \( \delta = 0.001 \):

```
\text{nDer(x^3,x,5)} \quad 75\,000001
```

For \( \delta = 1 \times 10^{-4} \):

```
\text{nDer(x^3,x,5)} \quad 75
```

**nDer(expression,variable)**

Uses the current value of `variable`.

```
5\times \quad \text{nDer(x^3,x)} \quad 75
```

**Negation: \(-\)**

- `number` or `- (expression)`
- `list`
- `matrix`
- `vector`

Returns the negative of the real or complex argument.

```
-2+5 \quad 3
-(2+5) \quad -7
-(0,-5,5) \quad \{0, 5, -5\}
```

**norm**

**MATRX MATH menu**

**VECTR MATH menu**

Returns the Frobenius norm of a real or complex `matrix`, calculated as:

\[\sqrt{\text{sum}(|real|^2 + |imaginary|^2)}\]

where the sum is over all elements.

```
[[1, 2], [-3, 4]] \quad \text{norm MAT} \quad 5.4772557505
```

}\[
\]
norm \textit{vector}

Returns the length of a real or complex \textit{vector}, where:

norm [a,b,c] returns $\sqrt{a^2+b^2+c^2}$.

\textbf{norm} \textit{number} or \textbf{norm} (\textit{expression})

Returns the absolute value of a real or complex \textit{number} or \textit{expression}, or of each element in \textit{list}.

\textbf{Normal}

\textit{†} mode screen

\textbf{Normal}

Sets normal notation mode.

In \textbf{Eng} notation mode:

123456789 \hspace{1em} ENTER \hspace{1em} 123.456789\times10^6

In \textbf{Sci} notation mode:

123456789 \hspace{1em} ENTER \hspace{1em} 1.23456789\times10^8

In \textbf{Normal} notation mode:

123456789 \hspace{1em} ENTER \hspace{1em} 123456789
not integer

Not BASE BOOL menu

Returns the one’s complement of a real integer. Internally, integer is represented as a 16-bit binary number. The value of each bit is flipped (0 becomes 1, and vice versa) for the one’s complement.

For example, not 78:

\[
78 = 0000000001001110\text{b} \\
1111111110110001\text{b} \quad \text{(one’s complement)}
\]

Sign bit; 1 indicates a negative number

To find the magnitude of a negative binary number, determine its two’s complement (take the one’s complement and then add 1). For example:

\[
1111111110110001\text{b} = \text{one’s complement of 78} \\
0000000001001110\text{b} \quad \text{(one’s complement)} \\
+ 0000000000000001\text{b} \\
0000000001011110\text{b} = 79 \quad \text{(two’s complement)}
\]

Therefore, not 78 = -79.

You can enter real numbers instead of integers, but they are truncated automatically before the comparison.
Not equal to: ≠

TEST menu

numberA ≠ numberB
matrixA ≠ matrixB
vectorA ≠ vectorB
stringA ≠ stringB

Tests whether the condition argumentA ≠ argumentB is true or false. Numbers, matrices, and vectors can be real or complex. If complex, the magnitude (modulus) of each element is compared. Strings are case-sensitive.

• If true (argumentA ≠ argumentB), returns 1.
• If false (argumentA = argumentB), returns 0.

listA ≠ listB

Returns a list of 1s and/or 0s to indicate if each element in listA is ≠ the corresponding element in listB.

nPr

MATH PROB menu

items nPr number

Returns the number of permutations of items (n) taken number (r) at a time. Both arguments must be real nonnegative integers.

O

BASE TYPE menu

integer o

Designates a real integer as octal, regardless of the number base mode setting.

In Dec number base mode:
100 (ENTER) 8
100+16 (ENTER) 18
Oct

† mode screen

Sets octal number base mode. Results are displayed with the \( \_o \) suffix. In any number base mode, you can designate an appropriate value as binary, decimal, hexadecimal, or octal by using the \( b, d, h, \) or \( o \) designator, respectively, from the BASE TYPE menu.

Oct

In Oct number base mode:

\[
10+10b+fh+10d \rightarrow 43o
\]

Oct

BASE CONV menu

- \( n\)umber \( \rightarrow \) Oct
- \( l\)ist \( \rightarrow \) Oct
- \( m\)atrix \( \rightarrow \) Oct
- \( v\)ector \( \rightarrow \) Oct

Returns the octal equivalent of the real or complex argument.

In Dec number base mode:

\[
\begin{align*}
2+8 \rightarrow 16 \\
\text{Ans} \rightarrow \text{Oct} \rightarrow 20o \\
(7,8,9,10) \rightarrow \text{Oct} \rightarrow \{7o, 10o, 11o, 12o\}
\end{align*}
\]

OneVar

STAT CALC menu

(OneVa shows on menu)

OneVar \( x\)List, frequencyList

Performs one-variable statistical analysis using real data points in \( x\)List and frequencies in \( f\)requencyList.

The values used for \( x\)List and \( f\)requencyList are stored automatically to built-in variables \( x\)Stat and \( f\)Stat, respectively.

OneVar \( x\)List

Uses frequencies of 1.

\[
\begin{align*}
\{0,1,2,3,4,5,6\} \rightarrow \text{XL} \rightarrow \{0\ 1\ 2\ 3\ 4\ 5\ 6\} \\
\text{OneVar XL} \rightarrow \text{XL}
\end{align*}
\]

OneVar

\( x\)List, \( f\)requencyList

\[
\begin{align*}
\{0,1,2,3,4,5,6\} \rightarrow \text{XL} \rightarrow \{0\ 1\ 2\ 3\ 4\ 5\ 6\} \\
\text{OneVar XL} \rightarrow \text{XL}
\end{align*}
\]

\[
\begin{align*}
1-\text{Var} \rightarrow \text{Stats} \\
x^2=21 \\
x=2.91 \\
x=1.682469 \\
x=2 \\
x=0
\end{align*}
\]

Scroll down to see more results.
**OneVar**

Uses **xStat** and **fStat** for **xList** and **frequencyList**. These built-in variables must contain valid data of the same dimension; otherwise, an error occurs.

**or**

<table>
<thead>
<tr>
<th>BASE BOOL menu</th>
</tr>
</thead>
</table>

**integerA or integerB**

Compares two real integers bit by bit. Internally, both integers are converted to binary. When corresponding bits are compared, the result is 1 if either bit is 1; the result is 0 only if both bits are 0. The returned value is the sum of the bit results.

For example, 78 or 23 = 95.

\[
\begin{align*}
78 & = 1001110_b \\
23 & = 0010111_b \\
1011111_b & = 95
\end{align*}
\]

You can enter real numbers instead of integers, but they are truncated automatically before the comparison.
**Outpt(row, column, string)**

Displays *string* beginning at *row* and *column*, where 1 ≤ *row* ≤ 8 and 1 ≤ *column* ≤ 21.

**Outpt(row, column, value)**

Displays *value* beginning at the specified *row* and *column*.

**Outpt("CBLSEND", listName)**

Sends the contents of *listName* to the CBL or CBR System.

You can send data also by using **Send** as described on page 350.

---

**Program segment:**

```
:ClrLCD
:For(i,1,8)
  :Outpt(i,randInt(1,21),"A")
:End
```

**Example result after execution:**

```
A A A
A A A
A A A
```

---
**P2Reg**

**STAT CALC menu**

Built-in equation variables such as \( y_1 \), \( r_1 \), and \( xt_1 \) are case-sensitive. Do not use \( Y_1 \), \( R_1 \), and \( XT_1 \).

**P2Reg \( xList,yList,frequencyList,equationVariable \)**

Performs a second order polynomial regression using real data pairs in \( xList \) and \( yList \) and frequencies in \( frequencyList \). The regression equation is stored to \( equationVariable \), which must be a built-in equation variable such as \( y_1 \), \( r_1 \), and \( xt_1 \). The equation's coefficients always are stored as a list to built-in variable \( PRegC \).

Values used for \( xList \), \( yList \), and \( frequencyList \) are stored automatically to built-in variables \( xStat \), \( yStat \), and \( fStat \), respectively. The regression equation is stored also to built-in equation variable \( RegEq \).

**P2Reg \( xList,yList,equationVariable \)**

Uses frequencies of 1.

**P2Reg \( xList,yList,frequencyList \)**

Stores the regression equation to \( RegEq \) only.

**P2Reg \( xList,yList \)**

Uses frequencies of 1, and stores the regression equation to \( RegEq \) only.

**P2Reg \( equationVariable \)**

Uses \( xStat \), \( yStat \), and \( fStat \) for \( xList \), \( yList \), and \( frequencyList \), respectively. These built-in variables must contain valid data of the same dimension; otherwise, an error occurs. The regression equation is stored to \( equationVariable \) and \( RegEq \).
Chapter 20: A to Z Function and Instruction Reference

P2Reg
Uses xStat, yStat, and fStat, and stores the regression equation to RegEq only.

P3Reg
STAT CALC menu

Built-in equation variables such as y1, r1, and xt1 are case-sensitive. Do not use Y1, R1, and XT1.

P3Reg xList, yList, frequencyList, equationVariable
Performs a third order polynomial regression using real data pairs in xList and yList and frequencies in frequencyList. The regression equation is stored to equationVariable, which must be a built-in equation variable such as y1, r1, and xt1. The equation’s coefficients always are stored as a list to built-in variable PRegC.

Values used for xList, yList, and frequencyList are stored automatically to built-in variables xStat, yStat, and fStat, respectively. The regression equation is stored also to built-in equation variable RegEq.

P3Reg xList, yList, equationVariable
Uses frequencies of 1.

P3Reg xList, yList, frequencyList
Stores the regression equation to RegEq only.

P3Reg xList, yList
Uses frequencies of 1, and stores the regression equation to RegEq only.
P3Reg \textit{equationVariable} \\
Uses \textit{xStat}, \textit{yStat}, and \textit{fStat} for \textit{xList}, \textit{yList}, and \textit{frequencyList}, respectively. These built-in variables must contain valid data of the same dimension; otherwise, an error occurs. The regression equation is stored to \textit{equationVariable} and \textit{RegEq}.

P3Reg \\
Uses \textit{xStat}, \textit{yStat}, and \textit{fStat}, and stores the regression equation to \textit{RegEq} only.

\textbf{P4Reg} \texttt{STAT CALC menu} \\
\textit{Built-in equation variables such as} \texttt{y1}, \texttt{r1}, and \texttt{xt1} \textit{are case-sensitive. Do not use} \texttt{Y1}, \texttt{R1}, and \texttt{XT1}. \\
P4Reg \textit{xList,yList,frequencyList,equationVariable} \\
Performs a fourth order polynomial regression using real data pairs in \textit{xList} and \textit{yList} and frequencies in \textit{frequencyList}. The regression equation is stored to \textit{equationVariable}, which must be a built-in equation variable such as \texttt{y1}, \texttt{r1}, and \texttt{xt1}. The equation's coefficients always are stored as a list to built-in variable \textit{PRegC}. \\
Values used for \textit{xList}, \textit{yList}, and \textit{frequencyList} are stored automatically to built-in variables \textit{xStat}, \textit{yStat}, and \textit{fStat}, respectively. The regression equation is stored also to built-in equation variable \textit{RegEq}.

P4Reg \textit{xList,yList,equationVariable} \\
Uses frequencies of 1.

P4Reg \textit{xList,yList,frequencyList} \\
Stores the regression equation to \textit{RegEq} only.

In \textit{Func} graphing mode:
\begin{verbatim}
{\{-2,-1,0,1,2,3,4,5,6\}→L1 ENTER \{-2 -1 0 1 2 3 4 5 6\}}
{\{4,3,1,2,3,2,2,4,6\}→L2 ENTER \{4 3 1 2 3 2 2 4 6\}}
P4Reg L1,L2,y1 ENTER
\end{verbatim}
**P4Reg** \( xList,yList \)

Uses frequencies of 1, and stores the regression equation to `RegEq` only.

**P4Reg** `equationVariable`

Uses `xStat`, `yStat`, and `fStat` for `xList`, `yList`, and `frequencyList` respectively. These built-in variables must contain valid data of the same dimension; otherwise, an error occurs. The regression equation is stored to `equationVariable` and `RegEq`.

**P4Reg**

Uses `xStat`, `yStat`, and `fStat`, and stores the regression equation to `RegEq` only.

---

**Param**

† mode screen

Sets parametric graphing mode.

**Pause**

‡ program editor

<table>
<thead>
<tr>
<th><code>Pause string</code></th>
<th>Program segment:</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>Pause value</code></td>
<td>:Input &quot;Enter x:&quot;,x</td>
</tr>
<tr>
<td><code>Pause list</code></td>
<td>:( y_1 = x^2 - 6 )</td>
</tr>
<tr>
<td><code>Pause matrix</code></td>
<td>:Disp &quot;y_1 is:&quot;,&quot;y_1&quot;</td>
</tr>
<tr>
<td><code>Pause vector</code></td>
<td>:Pause &quot;Press ENTER to graph&quot;</td>
</tr>
</tbody>
</table>

Displays the specified argument and then suspends program execution until the user presses **ENTER**.
Pause
Suspends program execution until the user presses [ENTER].

Percent: %
MATH MISC menu

<table>
<thead>
<tr>
<th>number% or (expression)%</th>
<th>5% [ENTER]</th>
<th>.05</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>5%*(200 [ENTER])</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>(10+5)*200 [ENTER]</td>
<td>30</td>
</tr>
</tbody>
</table>

pEval(coefficientList, xValue)
MATH MISC menu

Returns the value of a polynomial (whose coefficients are given in coefficientList) at xValue.

Evaluate y=2x^2+2x+3 at x=5:
pEval([2,2,3],5) [ENTER] 63

PlOff
STAT PLOT menu

PlOff [1,2,3]
Deselects the specified stat plot numbers.

PlOff
Deselects all stat plot numbers.

PlOn
STAT PLOT menu

PlOn [1,2,3]
Selects the specified stat plot numbers, in addition to any plot numbers that are already selected.

PlOn
Selects all stat plot numbers.
† STAT PLOT menu

The syntax and descriptions to the right refer to Plot1(), but they apply as well to Plot2() and Plot3().

**Scatter plot**

\[ \text{Plot1}(1, x\text{ListName}, y\text{ListName}, \text{mark}) \]

Defines and selects the plot using real data pairs in \( x\text{ListName} \) and \( y\text{ListName} \).

The optional \( \text{mark} \) specifies the character used to plot the points. If you omit \( \text{mark} \), a box is used.

\( \text{mark}: \begin{cases} 1 = \text{box} (\square) & 2 = \text{cross} (+) & 3 = \text{dot} (*) \end{cases} \)

- \( \text{xyLine plot} \)
  - \( \text{Plot1}(2, x\text{ListName}, y\text{ListName}, \text{mark}) \)

- **Modified box plot**
  - \( \text{Plot1}(3, x\text{ListName}, 1 \text{ or frequencyListName}, \text{mark}) \)
  - \( \text{Plot1}(3, x\text{ListName}, 1 \text{ or frequencyListName}) \)
  - \( \text{Plot1}(3, x\text{ListName}) \)

Defines and selects the plot using real data points in \( x\text{ListName} \) with the specified frequencies. If you omit \( 1 \text{ or frequencyListName} \), frequencies of 1 are used.

- **Histogram**
  - \( \text{Plot1}(4, x\text{ListName}, 1 \text{ or frequencyListName}) \)
  - \( \text{Plot1}(4, x\text{ListName}) \)

- **Box plot**
  - \( \text{Plot1}(5, x\text{ListName}, 1 \text{ or frequencyListName}) \)
  - \( \text{Plot1}(5, x\text{ListName}) \)
Chapter 20: A to Z Function and Instruction Reference

### Pol

Pol

Sets polar graphing mode.

<table>
<thead>
<tr>
<th>complexNumber &gt; Pol</th>
<th>Displays complexNumber in polar form (magnitude\angle), regardless of the complex number mode.</th>
</tr>
</thead>
<tbody>
<tr>
<td>list &gt; Pol</td>
<td>Returns a list, matrix, or vector in which each element of the argument is displayed in polar form.</td>
</tr>
<tr>
<td>matrix &gt; Pol</td>
<td></td>
</tr>
<tr>
<td>vector &gt; Pol</td>
<td></td>
</tr>
</tbody>
</table>

### PolarC

PolarC

Sets polar complex number mode (magnitude\angle).

<table>
<thead>
<tr>
<th>magnitude\angle</th>
<th>Used to enter complex numbers in polar form. The angle is interpreted according to the current angle mode.</th>
</tr>
</thead>
<tbody>
<tr>
<td>PolarC complex number mode:</td>
<td></td>
</tr>
<tr>
<td>Radian angle mode and PolarC complex number mode:</td>
<td></td>
</tr>
</tbody>
</table>

### PolarGC

PolarGC

Displays graph coordinates in polar form.
**poly**

Returns a list containing the real and complex roots of a polynomial whose coefficients are given in `coefficientList`.

\[ a_nx^n + \ldots + a_2x^2 + a_1x + a_0 = 0 \]

Find the roots of \(2x^3 - 8x^2 - 14x + 20 = 0\):

\[
poly \{2,-8,-14,20\} \rightarrow \{5\ -2\ 1\} \]

---

**Power:** ^

\[ \text{number}^\text{power} \text{ or } (\text{expression})^\text{expression} \]

Returns \(\text{number}\) raised to \(\text{power}\). The arguments can be real or complex.

\[
4^2 \rightarrow 16 \\
2^{-5} \rightarrow 0.03125
\]

\[
\{2,3,4\}^{3,4,5} \rightarrow \{8\ 81\ 1024\}
\]

---

**Power of 10:** 10^/sup

\[ 10^\text{power} \text{ or } 10^{\text{expression}} \]

Returns 10 raised to \(\text{power}\) or \(\text{expression}\), which can be real or complex.

\[
10^{1.5} \rightarrow 31.6227766017 \\
10^{-2} \rightarrow 0.01
\]
10^list

Returns a list in which each element is 10 raised to the power specified by the corresponding element in list.

prod

LIST OPS menu
MATH MISC menu

prod list

Returns the product of all real or complex elements in list.

Prod {1,2,4,8} ENTER

64

Prod {2,7,-8} ENTER

-112

Prompt

‡ program editor
I/O menu
(Prompt shows on menu)

Prompts the user to enter a value for variable A, then variable B, and so on.

Program segment:

:Prompt A,B,C

PtChg

‡ GRAPH DRAW menu

PtChg(x,y)

Reverses the point at graph coordinates (x,y).

PtChg(-6,2)

PtOff

‡ GRAPH DRAW menu

PtOff(x,y)

Erases the point at graph coordinates (x,y).

PtOff(3,5)

PtOn

‡ GRAPH DRAW menu

PtOn(x,y)

Draws the point at graph coordinates (x,y).

PtOn(3,5)
**PwrR**

**STAT CALC menu**

Built-in equation variables such as \( y_1, r_1, \) and \( xt_1 \) are case-sensitive. Do not use \( Y1, R1, \) and \( XT1 \).

_Fits a power regression model (\( y=ax^b \)) to positive real data pairs in \( x_{list} \) and \( y_{list} \), using frequencies in \( frequency_{list} \). The regression equation is stored to \( equation_{variable} \), which must be a built-in equation variable such as \( y_1, r_1, \) and \( xt_1 \).

Values used for \( x_{list}, y_{list} \), and \( frequency_{list} \) are stored automatically to built-in variables \( x_{stat}, y_{stat} \), and \( f_{stat} \), respectively. The regression equation is stored also to built-in equation variable \( RegEq \).

_PwrR \( x_{list}, y_{list}, equation_{variable} \)_

Uses frequencies of 1.

_PwrR \( x_{list}, y_{list}, frequency_{list} \)_

Stores the regression equation to \( RegEq \) only.

_PwrR \( x_{list}, y_{list} \)_

Uses frequencies of 1, and stores the regression equation to \( RegEq \) only.

_PwrR \( equation_{variable} \)_

Uses \( x_{stat}, y_{stat} \), and \( f_{stat} \) for \( x_{list}, y_{list} \), and \( frequency_{list} \), respectively. These built-in variables must contain valid data of the same dimension; otherwise, an error occurs. The regression equation is stored to \( equation_{variable} \) and \( RegEq \)._
PwrR

Uses xStat, yStat, and fStat, and stores the regression equation to RegEq only.

PxChg(row,column)

Reverses the pixel at (row, column), where 0 ≤ row ≤ 62 and 0 ≤ column ≤ 126.

PxChg(10,95)

PxOff(row,column)

Erases the pixel at (row, column), where 0 ≤ row ≤ 62 and 0 ≤ column ≤ 126.

PxOff(10,95)

PxOn(row,column)

Draws the pixel at (row, column), where 0 ≤ row ≤ 62 and 0 ≤ column ≤ 126.

PxOn(10,95)

PxTest(row,column)

Returns 1 if the pixel at (row, column) is on, 0 if it is off; 0 ≤ row ≤ 62 and 0 ≤ column ≤ 126.

Assuming the pixel at (10,95) is already on:

PxTest(10,95) ENTER 1

rAdd(matrix,rowA,rowB)

Returns a matrix in which rowA of a real or complex matrix is added to (and stored in) rowB.

\[
\begin{bmatrix}
5 & 3 & 1 \\
2 & 0 & 4 \\
3 & -1 & 2 \\
\end{bmatrix}
\]

rAdd(MAT,2,3) ENTER \[
\begin{bmatrix}
5 & 3 & 1 \\
2 & 0 & 4 \\
5 & -1 & 6 \\
\end{bmatrix}
\]
### Radian

**MATH ANGLE menu**

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>† 2nd [MODE]</code></td>
<td>Sets radian angle mode.</td>
</tr>
</tbody>
</table>

**Radian entry:** †

In Radian angle mode:

- `sin(\pi/2)`
- `sin 90`

<table>
<thead>
<tr>
<th>Expression</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>sin(\pi/2)</code></td>
<td>1</td>
</tr>
<tr>
<td><code>sin 90</code></td>
<td>0.893996663601</td>
</tr>
</tbody>
</table>

**number** or **(expression)**

Designates a real number or expression as radians, regardless of the angle mode setting.

In Degree angle mode:

- `cos(\pi/2)`
- `cos(\pi/2)`

<table>
<thead>
<tr>
<th>Expression</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>cos(\pi/2)</code></td>
<td>0.999624216659</td>
</tr>
<tr>
<td><code>cos(\pi/2)</code></td>
<td>0</td>
</tr>
</tbody>
</table>

**list**

Designates each element in a real list as radians.

In Degree angle mode:

- `cos({\pi/2,\pi})`

<table>
<thead>
<tr>
<th>Expression</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>cos({\pi/2,\pi})</code></td>
<td>{0, -1}</td>
</tr>
</tbody>
</table>

### rand

**MATH PROB menu**

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>rand</code></td>
<td>Returns a random number between 0 and 1.</td>
</tr>
</tbody>
</table>

To control a random number sequence, first store an integer seed value to `rand` (such as `0→rand`).

**randBin**

**MATH PROB menu**

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>randBin(#ofTrials, probabilityOfSuccess, #ofSimulations)</code></td>
<td>Returns a list of random integers from a binomial distribution, where <code>#ofTrials</code> ≥ 1 and <code>0 ≤ probabilityOfSuccess ≤ 1</code>. The <code>#ofSimulations</code> is an integer ≥ 1 that specifies the number of integers returned in the list.</td>
</tr>
</tbody>
</table>

A seed value stored to `rand` also affects `randBin`.

**randBin(#ofTrials, probabilityOfSuccess)**

<table>
<thead>
<tr>
<th>Expression</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>randBin(5,.2)</code></td>
<td>{0, 3, 2}</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Expression</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>rand</code></td>
<td>{0.943597402492}</td>
</tr>
<tr>
<td><code>rand</code></td>
<td>{0.146687829222}</td>
</tr>
<tr>
<td><code>0→rand</code></td>
<td><code>randBin(5,.2)</code> {0.943597402492}</td>
</tr>
<tr>
<td><code>0→rand</code></td>
<td><code>randBin(5,.2)</code> {0.943597402492}</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Expression</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>randBin(#ofTrials, probabilityOfSuccess)</code></td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Expression</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>0→rand</code></td>
<td><code>randBin(5,.2)</code></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Expression</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>rand</code></td>
<td>{0.943597402492}</td>
</tr>
<tr>
<td><code>rand</code></td>
<td>{0.146687829222}</td>
</tr>
<tr>
<td><code>0→rand</code></td>
<td><code>randBin(5,.2)</code> {0.943597402492}</td>
</tr>
<tr>
<td><code>0→rand</code></td>
<td><code>randBin(5,.2)</code> {0.943597402492}</td>
</tr>
</tbody>
</table>

### randBin(#ofTrials, probabilityOfSuccess, #ofSimulations)

<table>
<thead>
<tr>
<th>Expression</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>randBin(5,.2,3)</code></td>
<td>{0, 3, 2}</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Expression</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>rand</code></td>
<td>{0.943597402492}</td>
</tr>
<tr>
<td><code>rand</code></td>
<td>{0.146687829222}</td>
</tr>
<tr>
<td><code>0→rand</code></td>
<td><code>randBin(5,.2)</code> {0.943597402492}</td>
</tr>
<tr>
<td><code>0→rand</code></td>
<td><code>randBin(5,.2)</code> {0.943597402492}</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Expression</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>randBin(#ofTrials, probabilityOfSuccess)</code></td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Expression</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>0→rand</code></td>
<td><code>randBin(5,.2)</code></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Expression</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>rand</code></td>
<td>{0.943597402492}</td>
</tr>
<tr>
<td><code>rand</code></td>
<td>{0.146687829222}</td>
</tr>
<tr>
<td><code>0→rand</code></td>
<td><code>randBin(5,.2)</code> {0.943597402492}</td>
</tr>
<tr>
<td><code>0→rand</code></td>
<td><code>randBin(5,.2)</code> {0.943597402492}</td>
</tr>
</tbody>
</table>
randInt(lower,upper,#ofTrials)

Returns a list of random integers bound by the specified integers, lower ≤ integer ≤ upper. The #ofTrials is an integer ≥ 1 that specifies the number of integers returned in the list.

A seed value stored to rand also affects randInt().

randInt(lower,upper)

Returns a single random integer.

randM(rows,columns)

Returns a rows × columns matrix filled with random one-digit integers (-9 to 9).

randNorm(mean,stdDeviation,#ofTrials)

Returns a list of random numbers from a normal distribution specified by mean and stdDeviation. The #ofTrials is an integer ≥ 1 that specifies how many numbers are returned. Each returned number could be any real number, but most will be within the interval: [mean−3(stdDeviation), mean+3(stdDeviation)].

A seed value stored to rand also affects randNorm().

randNorm(mean,stdDeviation)

Returns a single random number.
### Chapter 20: A to Z Function and Instruction Reference

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>RcGDB</strong></td>
<td><strong>graphDataBaseName</strong>&lt;br&gt;Restores all settings stored in <code>graphDataBaseName</code>. For a list of settings, refer to <code>StGDB</code> on page 361.</td>
</tr>
<tr>
<td><strong>RcPic</strong></td>
<td><strong>pictureName</strong>&lt;br&gt;Displays the current graph and adds the picture stored in <code>pictureName</code>.</td>
</tr>
<tr>
<td><strong>real</strong></td>
<td><strong>complexNumber</strong>&lt;br&gt;Returns the real part of <code>complexNumber</code>. &lt;br&gt;<strong>real</strong> (real,imaginary) returns real. &lt;br&gt;<strong>real</strong> (magnitude,angle) returns magnitude * \cos (angle).</td>
</tr>
<tr>
<td><strong>Rec</strong></td>
<td><strong>complexNumber</strong>&lt;br&gt;Displays <code>complexNumber</code> in rectangular form <code>(real,imaginary)</code> regardless of the complex number mode.</td>
</tr>
</tbody>
</table>

---

**In Radian angle mode:**
- real (3,4) ENTER 3
- real (3\angle4) ENTER -1.96093086259

**In PolarC complex number mode:**
- \(\sqrt{2}\) ENTER (1.41421356237\angle1.570\_)
- ans\*rec ENTER (0,1.41421356237)
**complexList**\texttt{Rec}  
**complexMatrix**\texttt{Rec}  
**complexVector**\texttt{Rec}  

Returns a list, matrix, or vector in which each element of the argument is displayed in rectangular form.

---

**RectC**  
† mode screen

Sets rectangular complex number mode ($\text{real}, \text{imaginary}$).

---

**RectGC**  
† graph format screen

Displays graph coordinates in rectangular form.

---

**RectV**  
† mode screen

Sets rectangular vector coordinate mode [x y z].

---

**ref**

MATRX OPS menu

Returns the row-echelon form of a real or complex matrix. The number of columns must be greater than or equal to the number of rows.

---

**In PolarC** complex number mode:

\[
\left[(3\angle\pi/6),\sqrt{-2}\right], \left[(3\angle.523598775598),\sqrt{-2}\right], \left[(2.59807621135,1.5)\right].
\]

---

**In RectC** complex number mode:

\[
\sqrt{-2}, (0,1.41421356237).
\]

---

**In RectV** vector coordinate mode:

\[
3\times[4\angle5], [3.40394622556 \ -11.5…
\]

---

\[
[[4,5,6],[7,8,9]]\text{MAT} \ (\text{ENTER}) \quad [[4,5,6], [7,8,9]]
\]

ref \text{MAT} \ (\text{ENTER})  

\[
[[1.14285714286, 1…
\]

---
Repeat
‡ program editor
CTL menu
(Repeat shows on menu)

\[\text{:Repeat condition}
: \text{commands-to-repeat}
: \text{:End}
: \text{commands}\]

Executes \text{commands-to-repeat} until \text{condition} is true.

Return
‡ program editor
CTL menu
(Return shows on menu)

Return
In a subroutine, exits the subroutine and returns to the calling program. In the main program, stops execution and returns to the home screen.

\[\text{Program segment in the calling program:}\]
\[\text{:Input "Diameter:",DIAM}
\text{:Input "Height:",HT}
\text{:AREACIRC}
\text{:VOL=AREA\*HT}
\text{:Disp "Volume =",VOL}
\]

\[\text{AREACIRC subroutine program:}\]
\[\text{PROGRAM:AREACIRC}
\text{:RADIUS=DIAM/2}
\text{:AREA=π\*RADIUS}^2
\text{:Return}\]

RK
‡ graph format screen
(scroll down to second screen)

RK
In \text{DifEq} graphing mode, uses an algorithm based on the Runge-Kutta method to solve differential equations. Typically, RK is more accurate than Euler but takes longer to find the solutions.
**rnorm**

**MATRX MATH menu**

**rnorm** Returns the row norm of a real or complex matrix. For each row, **rnorm** sums the absolute values (magnitudes of complex elements) of all elements on that row. The returned value is the largest of the sums.

**rnorm** Returns the largest absolute value (or magnitude) in a real or complex vector.

**Root:** $^x\sqrt{}$

**MATH MISC menu**

$x^\theta root^\cdot\sqrt{number}$ or $x^\theta root^\cdot\sqrt{(expression)}$

Returns the $x^{th}$ root of number or expression. The arguments can be real or complex.

$x^\theta root^\cdot\sqrt{list}$

Returns a list in which each element is the $x^{th}$ root of the corresponding element in list.

$x^\theta rootList^\cdot\sqrt{list}$

Returns a list in which each element is the root specified by the corresponding elements in $x^\theta rootList$ and list.
**rotL**

BASE BIT menu

rotL  integer

Returns a real integer with bits rotated one to the left. Internally, integer is represented as a 16-bit binary number. When the bits are rotated left, the leftmost bit rotates to the rightmost bit.

\[
\text{rotL} \ 0000111100001111_\text{b} \rightarrow 00111000011110_\text{b}
\]

rotL is not valid in Dec number base mode. To enter hexadecimal numbers A through F, use the BASE A-F menu. Do not use [ALPHA] to type a letter.

**rotR**

BASE BIT menu

rotR  integer

Returns a real integer with bits rotated one to the right. Internally, integer is represented as a 16-bit binary number. When the bits are rotated right, the rightmost bit rotates to the leftmost bit.

\[
\text{rotR} \ 0000111100001111_\text{b} \rightarrow 1000011110000111_\text{b}
\]

rotR is not valid in Dec number base mode. To enter hexadecimal numbers A through F, use the BASE A-F menu. Do not use [ALPHA] to type a letter.
**round(**

MATH NUM menu

**round(**

**round(number)**

Returns a real or complex number rounded to the specified number of decimals (0 to 11). If number of decimals is omitted, number is rounded to 12 decimal places.

**round(list,#ofDecimals)**

**round(matrix,#ofDecimals)**

**round(vector,#ofDecimals)**

Returns a list, matrix, or vector in which each element is the rounded value of the corresponding element in the argument. #ofDecimals is optional.

**rref**

MATRX OPS menu

**rref matrix**

Returns the reduced row-echelon form of a real or complex matrix. The number of columns must be greater than or equal to the number of rows.

**rSwap**

MATRX OPS menu

**rSwap(matrix,rowA,rowB)**

Returns a matrix with rowA of a real or complex matrix swapped with rowB.
**Scatter**

† STAT DRAW menu
(Scatte shows on menu)

**Scatter** *xList, yList*

Draws a scatter plot on the current graph, using the real data pairs in *xList* and *yList*.

**Sci**

† mode screen

**Sci**

Sets scientific notation display mode.

In **Sci** notation mode:

123456789 ENTER 1.23456789\(\times10^8\)

In **Normal** notation mode:

123456789 ENTER 123456789
Select\((xListName,yListName)\)

If a scatter plot or xyline plot is currently selected and plotted on the graph screen, you can select a subset (range) of those data points. The selected data points are stored to \(xListName\) and \(yListName\).

Select\((xListName,yListName)\) displays the current graph screen and starts an interactive session during which you select a range of data points.

a. Move the cursor to the leftmost (left bound) point of the range you want to select and press \(\text{ENTER}\).

b. Then move the cursor to the rightmost (right bound) point of the range you want to select and press \(\text{ENTER}\).

A new stat plot of \(xListName\) and \(yListName\) replaces the plot from which you selected the points.

Send\((\text{listName})\)

Sends the contents of \(\text{listName}\) to the CBL or CBR System.

\[\begin{align*}
\{1,2,3,4,5\} & \rightarrow L1 \quad \text{ENTER} \\
\{9,6,4,-1,2,5,7,10\} & \rightarrow L1 \quad \text{ENTER} \\
\{7,6,2,1,3,6,7,9\} & \rightarrow L2 \quad \text{ENTER}
\end{align*}\]
**seq(***expression, variable, begin, end, step**)

Retuns a list containing a sequence of numbers created by evaluating expression from variable = begin to variable = end in increments of step.

**seq(expression, variable, begin, end)**

Uses a step of 1.

**SeqG**

Sets sequential graphing format, in which selected functions are plotted one at a time.

**SetLEdit**

Removes all lists from the list editor and then stores one or more ListNames in the specified order, starting with column 1.

Removes all lists from the list editor and stores built-in lists xStat, yStat, and fStat in columns 1 through 3, respectively.
Shade(lowerFunc, upperFunc, xLeft, xRight, pattern, patternRes)

Draws lowerFunc and upperFunc in terms of x on the current graph and shades the area bounded by lowerFunc, upperFunc, xLeft, and xRight. The shading style is determined by pattern (1 through 4) and patternRes (1 through 8).

pattern:

1 = vertical (default)  
2 = horizontal  
3 = negative-slope 45°  
4 = positive-slope 45°

patternRes (resolution):

1 = every pixel (default)  
2 = every 2nd pixel  
3 = every 3rd pixel  
4 = every 4th pixel  
5 = every 5th pixel  
6 = every 6th pixel  
7 = every 7th pixel  
8 = every 8th pixel

Shade(lowerFunc, upperFunc)

Sets xLeft and xRight to xMin and xMax, respectively, and uses the defaults for pattern and patternRes.

In Func graphing mode:

Shade(x-2, x^3-8 x, -5, 1, 2, 3) [ENTER]

ClDrw:Shade(x^3-8, x, -2) [ENTER]
**shftL**

BASE BIT menu

`shftL integer`

Returns a real *integer* with bits shifted one to the left. Internally, *integer* is represented as a 16-bit binary number. When the bits are shifted left, the leftmost bit is dropped and 0 is used as the rightmost bit.

\[
\text{shftL} \ 0000111100001111b = 0001111000011110b
\]

`shftL` is not valid in *Dec* number base mode. To enter hexadecimal numbers A through F, use the BASE A-F menu. Do not use [ALPHA] to type a letter.

**shftR**

BASE BIT menu

`shftR integer`

Returns a real *integer* with bits shifted one to the right. Internally, *integer* is represented as a 16-bit binary number. When the bits are shifted right, the rightmost bit is dropped and 0 is used as the leftmost bit.

\[
\text{shftR} \ 0000111100001111b = 0000111100001111b
\]

`shftR` is not valid in *Dec* number base mode. To enter hexadecimal numbers A through F, use the BASE A-F menu. Do not use [ALPHA] to type a letter.
ShwSt

Displays the results of the most recent stat calculation.

sign

MATH NUM menu

sign number or sign (expression)

Returns -1 if the argument is < 0, 1 if it is > 0, or 0 if it is = 0. The argument must be real.

sign list

Returns a list in which each element is -1, 1, or 0 to indicate the sign of the corresponding element in list.

SimulG

† graph format screen

Sets simultaneous graphing format, in which all selected functions are plotted at the same time.

simult(squareMatrix, vector)

Returns a vector containing the solutions to a system of simultaneous linear equations that have the form:

\[
\begin{align*}
& a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + ... = b_1 \\
& a_{21}x_1 + a_{22}x_2 + a_{23}x_3 + ... = b_2 \\
& a_{31}x_1 + a_{32}x_2 + a_{33}x_3 + ... = b_3 \\
\end{align*}
\]

Each row in squareMatrix contains the \( a \) coefficients of an equation, and vector contains the \( b \) constants.

Solve the following for \( x \) and \( y \):

\[
\begin{align*}
3x - 4y &= 7 \\
x + 6y &= 6 \\
\end{align*}
\]

\[
\begin{bmatrix}
3, 4
\end{bmatrix}
\]

\[
\begin{bmatrix}
1, 6
\end{bmatrix}
\]

\[
\begin{bmatrix}
3, -4
\end{bmatrix}
\]

\[
\begin{bmatrix}
1, 6
\end{bmatrix}
\]

\[
\begin{bmatrix}
7, 6
\end{bmatrix}
\]

\[
\begin{bmatrix}
7, 6
\end{bmatrix}
\]

The solution is \( x=3 \) and \( y=5 \).
\( \sin \)  

\( \sin \) \text{angle} \text{ or} \ \sin (\text{expression}) 

Returns the sine of \text{angle} \text{ or} \ \text{expression}, which can be real or complex.

An angle is interpreted as degrees or radians according to the current angle mode. In any angle mode, you can designate an angle as degrees or radians by using the ° or ′ designator, respectively, from the MATH ANGLE menu.

\( \sin \) \text{list} 

Returns a list in which each element is the sine of the corresponding element in \text{list}.

\( \sin \) \text{squareMatrix} 

Returns a square matrix that is the matrix sine of \text{squareMatrix}. The matrix sine corresponds to the result calculated using power series or Cayley-Hamilton Theorem techniques. This is not the same as simply calculating the sine of each element.

\( \sin^{-1} \) \text{number} \text{ or} \ \sin^{-1} (\text{expression}) 

Returns the arcsine of \text{number} \text{ or} \ \text{expression}, which can be real or complex.

\( \sin^{-1} \) \text{list} 

Returns a list in which each element is the arcsine of the corresponding element in \text{list}. 

\text{In Radian angle mode:} 
\begin{align*} 
\sin \pi/2 & \quad \text{ENTER} \quad 0 \\
\sin (\pi/2) & \quad \text{ENTER} \quad 1 \\
\sin 45° & \quad \text{ENTER} \quad .707106781187 
\end{align*}

\text{In Degree angle mode:} 
\begin{align*} 
\sin 45 & \quad \text{ENTER} \quad .707106781187 \\
\sin (\pi/2)° & \quad \text{ENTER} \quad 1 
\end{align*}

\text{In Radian angle mode:} 
\begin{align*} 
\sin \{0, \pi/2, \pi\} & \quad \text{ENTER} \quad \{0 \ 1 \ 0\} \\
\sin \{0,30,90\} & \quad \text{ENTER} \quad \{0 \ .5 \ 1\} 
\end{align*}

\text{In Degree angle mode:} 
\begin{align*} 
\sin^{-1} .5 & \quad \text{ENTER} \quad .523598775598 \\
\sin^{-1} \{0.5\} & \quad \text{ENTER} \quad \{0 \ .523598775598\} \\
\sin^{-1} 1 & \quad \text{ENTER} \quad 90 
\end{align*}
### sinh

<table>
<thead>
<tr>
<th>sinh number or sinh (expression)</th>
<th>sinh 1.2 ENTER</th>
<th>1.50946135541</th>
</tr>
</thead>
</table>

Returns the hyperbolic sine of `number` or `expression`, which can be real or complex.

<table>
<thead>
<tr>
<th>sinh list</th>
<th>sinh (0,1.2) ENTER</th>
<th>{0 1.50946135541}</th>
</tr>
</thead>
</table>

Returns a list in which each element is the hyperbolic sine of the corresponding element in `list`.

### sinh⁻¹

<table>
<thead>
<tr>
<th>sinh⁻¹ number or sinh⁻¹(expression)</th>
<th>sinh⁻¹ 1 ENTER</th>
<th>.88137358702</th>
</tr>
</thead>
</table>

Returns the inverse hyperbolic sine of `number` or `expression`, which can be real or complex.

<table>
<thead>
<tr>
<th>sinh⁻¹ list</th>
<th>sinh⁻¹ (1,2,1,3) ENTER</th>
<th>{.88137358702 1.4874...}</th>
</tr>
</thead>
</table>

Returns a list in which each element is the inverse hyperbolic sine of the corresponding element in `list`. |
**SinR**

**STAT CALC menu**

**Built-in equation variables** such as \( y_1, r_1, \) and \( xt_1 \) are case-sensitive. Do not use \( Y_1, R_1, \) and \( XT_1 \).

If you specify a period, the TI-86 may find a solution more quickly or it may find a solution when one would not have been found otherwise.

**SinR** \([\text{iterations},] \text{xList}, \text{yList}, [\text{period}], \text{equationVariable}\)**

Attempts to fit a sinusoidal regression model \( y = a \sin(bx + c) + d \) to real data pairs in \( x\text{List} \) and \( y\text{List} \), using an optional estimated period. The regression equation is stored to \( \text{equationVariable} \), which must be a built-in equation variable such as \( y_1, r_1, \) and \( xt_1 \). The equation’s coefficients are always stored as a list to built-in variable \( \text{PRegC} \).

\( \text{iterations} \) is optional; it specifies the maximum number of times (1 through 16) the TI-86 will attempt to find a solution. If omitted, 8 is used. Typically, larger values result in better accuracy but longer execution times, and vice versa.

If you omit the optional \( \text{period} \), the difference between values in \( x\text{List} \) should be equal and in sequential order. If you specify \( \text{period} \), the differences between \( x \) values can be unequal.

Values used for \( x\text{List} \) and \( y\text{List} \) are stored automatically to built-in variables \( x\text{Stat} \) and \( y\text{Stat} \), respectively. The regression equation is stored also to built-in equation variable \( \text{RegEq} \).

The output of **SinR** is always in radians, regardless of the angle mode setting.

**SinR** \([\text{iterations},] \text{xList}, \text{yList}, [\text{period}]\)

Stores the regression equation to \( \text{RegEq} \) only.

---

\[
\text{seq}(x, x, 1, 361, 30) \rightarrow \text{L1} \quad \text{( ENTER )}
\]

\[
\begin{align*}
1 & \quad 31 & \quad 61 & \quad 91 & \quad 121 & \quad 151 & \ldots \\
5.5 & \quad 8 & \quad 11.5 & \quad 15 & \quad 19 & \quad 19.5 & \quad 17 & \quad 14.5 & \quad 12.5 & \quad 8.5 & \quad 5.5 \rightarrow \text{L2} \quad \text{( ENTER )}
\end{align*}
\]

\[
\text{SinR} \quad \text{L1}, \text{L2}, \text{y1} \quad \text{( ENTER )}
\]

\[
\text{PRegC}
\]

\[
\text{Eq} = a \sin(bx + c) + d
\]

\[
\begin{align*}
5.5 & \quad 8 \quad 11.5 & \quad 15 & \quad 19 & \quad 19.5 & \quad 17 & \quad 14.5 & \quad 12.5 & \quad 8.5 & \quad 5.5
\end{align*}
\]

\[
\text{SinReg}
\]

\[
\begin{align*}
x \quad \text{Stat} & \quad \text{L1} \\
5.5 & \quad 8 & \quad 11 & \quad 11.5 & \quad 15 & \quad 19 & \quad 19.5 & \quad 17 & \quad 14.5 & \quad 12.5 & \quad 8.5 & \quad 5.5
\end{align*}
\]

\[
\text{Plot1(L1, L2)} \quad \text{( ENTER )} \quad \text{Done}
\]

\[
\text{ZData} \quad \text{( ENTER )}
\]

---

seq(x, x, 1, 361, 30)→L1 \( \text{( ENTER )} \)

\( \begin{align*} 
1 & \quad 31 & \quad 61 & \quad 91 & \quad 121 & \quad 151 & \ldots \\
(5.5, 8, 11.5, 16.5, 19, 19.5, 17, \\
14.5, 12.5, 8.5, 5.5) & \rightarrow & \text{L2} \quad \text{( ENTER )}
\end{align*} \)

\( \text{SinR} \quad \text{L1}, \text{L2}, \text{y1} \quad \text{( ENTER )} \)

\( \text{SinReg} \)

\( x \cos(bx+c)+d \)

\( \text{Eq} = 5.770225779 \times 1 \cdot 0162… \)

\( \text{Plot1(L1, L2)} \quad \text{( ENTER )} \quad \text{Done} \)

\( \text{ZData} \quad \text{( ENTER )} \)
**SinR [iterations,] equationVariable**

Uses xStat and yStat for xList and yList, respectively. These built-in variables must contain valid data of the same dimension; otherwise, an error occurs. The regression equation is stored to equationVariable and RegEq.

**SinR [iterations]**

Uses xStat and yStat, and stores the regression equation to RegEq only.

---

**SlpFld**

† graph format screen (scroll down to second screen)

In DiffEq graphing mode, turns on slope fields. To turn off direction and slope fields, use FldOff.

---

**Solver(equation,variable,guess,{lower,upper})**

Solves equation for variable, given an initial guess and lower and upper bounds within which the solution is sought. equation can be an expression, which is assumed to equal 0.

**Solver(equation,variable,guess)**

Uses -1e99 and 1e99 for upper and lower, respectively.

**Solver(equation,variable,{guessLower,guessUpper})**

Uses the secant line between guessLower and guessUpper to start the search. Solver will still search for a solution outside of this range.

If \( y=5 \), solve \( x^3+y^2=125 \) for \( x \). You guess the solution is approximately 4:

\[
\begin{align*}
5 \rightarrow y \quad \text{ENTER} \\
\text{Solver}(x^3+y^2=125,x,4) \quad \text{ENTER} \\
\text{Done} \\
x \quad \text{ENTER} \\
4.64158883361
\end{align*}
\]
sortA
LIST OPS menu

**SortA**  list

Returns a list in which the real or complex elements of list are sorted in ascending order.

```
{5,8,-4,0,-6}➔L1 \( \rightarrow 5 \ 8 \ -4 \ 0 \ -6 \)
SortA L1 \( \rightarrow -6 \ -4 \ 0 \ 5 \ 8 \)
```

sortD
LIST OPS menu

**SortD**  list

Returns a list in which the real or complex elements of list are sorted in descending order.

```
{5,8,-4,0,-6}➔L1 \( \rightarrow 5 \ 8 \ -4 \ 0 \ -6 \)
SortD L1 \( \rightarrow 8 \ 5 \ 0 \ -4 \ -6 \)
```

Sortx
LIST OPS menu

**Sortx**  xListName,yListName,frequencyListName

In ascending order of x elements, sorts real or complex x and y data pairs and, optionally, their frequencies in xListName, yListName, and frequencyListName. The lists' contents are updated to reflect the changes.

```
(3,1,2)➔XL \( \rightarrow 3 \ 1 \ 2 \)
Sortx XL,YL \( \rightarrow \text{Done} \)
XL \( \rightarrow 1 \ 2 \ 3 \)
YL \( \rightarrow 8 \ -4 \ 0 \)
```

Sortx uses built-in variables xStat and yStat for xListName and yListName, respectively. These built-in variables must contain valid data of the same dimension; otherwise, an error occurs.

Sorty
LIST OPS menu

**Sorty**  xListName,yListName,frequencyListName

In ascending order of y elements, sorts real or complex x and y data pairs and, optionally, their frequencies in xListName, yListName, and frequencyListName. The lists' contents are updated to reflect the changes.

```
(3,1,2)➔XL \( \rightarrow 3 \ 1 \ 2 \)
Sorty XL,YL \( \rightarrow \text{Done} \)
XL \( \rightarrow 2 \ 3 \ 1 \)
YL \( \rightarrow -4 \ 0 \ 8 \)
```

The lists' contents are updated to reflect the changes.
Sorty

Uses built-in variables \texttt{xStat} and \texttt{yStat} for \texttt{xListName} and \texttt{yListName}, respectively. These built-in variables must contain valid data of the same dimension; otherwise, an error occurs.

\textbf{Sph}

\texttt{vector}\texttt{Sph}

Displays a 2- or 3-element \texttt{vector} as spherical coordinates in \([r \leq \theta \leq 0]\) or \([r \leq \theta \leq \phi]\) form, respectively, even if the display mode is not set for spherical (\texttt{SphereV}).

\texttt{SphereV}

Sets spherical vector coordinate mode \([r \leq \theta \leq \phi]\).

\textbf{Square:} \(2\)

\texttt{list}^2

Returns a real or complex argument multiplied by itself. To square a negative number, enclose it in parentheses. A \texttt{squareMatrix} multiplied by itself is not the same as simply squaring each element.

\textbf{Square root:} \(\sqrt{}\)

\texttt{number} or \texttt{\sqrt{expression}}

Returns the square root of \texttt{number} or \texttt{expression}, which can be real or complex.
\( \sqrt{\text{list}} \) Returns a list in which element is the square root of the corresponding element in \( \text{list} \).

\( \sqrt{\{-2, 25\}} \) \( \rightarrow \) \( \{0, 1.41421356237\} \) (…)

\( \text{St\ Eq} \)

**STRING menu**

\( \text{St\ Eq}(\text{stringVariable},\text{equationVariable}) \)

Converts \( \text{stringVariable} \) to a number, expression, or equation, and stores it in \( \text{equationVariable} \).

To convert the string and retain the same variable name, you can set \( \text{equationVariable} \) equal to \( \text{stringVariable} \).

If you use \text{Input} instead of \text{InpSt} here, the entered expression is evaluated at the current value of \( x \) and the result (not the expression) is stored.

\( "5" \rightarrow x:6 \) \( \rightarrow \) \text{error 10 DATA TYPE}

\( "5" \rightarrow x: \text{St Eq}(x, x):6 \) \( \rightarrow \) \text{enter} \( 30 \)

Program segment:

\( :\text{InpSt} \) "Enter y1(x):",\text{STR}
\( :\text{St Eq}((\text{STR}, y1)) \)
\( :\text{Input} \) "Enter x:“,\text{x}
\( :\text{Disp} "\text{Result is:}", y1(x) \)

You cannot store a string directly to a built-in equation variable.

\( \text{StGDB} \)

**GRAPH menu**

\( \text{StGDB} \text{graphDataBaseName} \)

Creates a graph database (GDB) variable that contains the current:

- Graphing mode, graph format settings, and range variables.
- Functions in the equation editor, whether they are selected, and their graph styles.

To restore the database and recreate the graph, use \( \text{RcGDB} \) (page 343).
Stop

‡ program editor CTL menu

Ends program execution and returns to the home screen.

Program segment:

:Stop

Use N=999, not N=999.

Store to variable: ➔


number ➔ variable or (expression) ➔ variable
string ➔ variable
list ➔ variable
vector ➔ variable
matrix ➔ variable

Stores the specified argument to variable.

StPic

† GRAPH menu

StPic pictureName

Stores a picture of the current graph screen to pictureName.

StReg(

STAT CALC menu

StReg(variable)

Stores the most recently calculated regression equation to variable. This lets you save a regression equation by storing it to any variable as opposed to a built-in equation variable.

{1,2,3,4,5} ➔ L1 ENTER
(1,2345)

{1,20,55,230,742} ➔ L2 ENTER
(1 20 55 230 742)

ExpR L1,L2:StReg(EQ) ENTER

Done

8 ➔ x ENTER

Rec(EQ) ENTER

Recalls the equation. Then ENTER evaluates it at the current value of x.

.411389487080597*4.7879605684671^x ENTER

113620.765451
String entry: " 

"string"

Defines a string. When you display a string, it is left-justified on the screen.

Strings are interpreted as text characters, not numbers. For example, you cannot perform a calculation with strings such as "4" or "A*B". To convert between string variables and equation variables, use EqSt( and StEq( as described on pages 290 and 361, respectively.

sub( string, begin, length )

Returns a new string that is a subset of string, starting at character number begin and continuing for the specified length.

Subtraction: −

numberA − numberB

Returns the value of numberB subtracted from numberA. The arguments can be real or complex.

list − number

Returns a list in which number is subtracted from each element of list. The arguments can be real or complex.
listA - listB
matrixA - matrixB
vectorA - vectorB

Returns a list, matrix, or vector that is the result of each element in the second argument subtracted from the corresponding element in the first argument. The two real or complex arguments must have the same dimension.

\[
\begin{align*}
\{5,7,9\} - \{4,5,6\} & \rightarrow \{1,2,3\} \\
[[5,7,9][11,13,15]] - [[4,5,6][7,8,9]] & \rightarrow [[1,2,3][4,5,6]] \\
[5,7,9] - [1,2,3] & \rightarrow [4,5,6]
\end{align*}
\]

sum
MATH MISC menu
LIST OPS menu

sum list

Returns the sum of all real or complex elements in list.

\[
\begin{align*}
\text{sum } \{1,2,4,8\} & \rightarrow 15 \\
\text{sum } \{2,7,-8,0\} & \rightarrow 1
\end{align*}
\]

tan

\begin{align*}
\text{tan } \text{angle} & \quad \text{or} \quad \text{tan } (\text{expression}) \\
\text{In Radian angle mode:} \\
\tan \frac{\pi}{4} & \rightarrow 0 \\
\tan \left(\frac{\pi}{4}\right) & \rightarrow 1 \\
\tan 45^\circ & \rightarrow 1 \\
\text{In Degree angle mode:} \\
\tan 45^\circ & \rightarrow 1 \\
\tan \left(\frac{\pi}{4}\right)^\circ & \rightarrow 1
\end{align*}

\[
\begin{align*}
\text{tan list} \\
\text{In Degree angle mode:} \\
\tan \{0,45,60\} & \rightarrow \{0 \ 1 \ 1.73205080757\}
\end{align*}
\]

\[
\begin{align*}
\text{tan} \{0,45,60\} & \rightarrow \{0 \ 1 \ 1.73205080757\}
\end{align*}
\]
### Chapter 20: A to Z Function and Instruction Reference

#### $\tan^{-1}$

**$\tan^{-1}$ number or $\tan^{-1}$ (expression)**

Returns the arctangent of *number* or *expression*, which can be real or complex.

In **Radian** angle mode:

$\tan^{-1} \cdot 5 \text{ ENTER} \quad 0.463647609001$

In **Degree** angle mode:

$\tan^{-1} \ 1 \text{ ENTER} \quad 45$

In **Radian** angle mode:

$\tan^{-1} \ {0,0.2,0.5} \text{ ENTER} \quad \{0 \ 0.19739555985 \ 0.463\ldots\}$

#### $\tanh$

**$\tanh$**

**MATH HYP** menu

**$\tanh$ number or $\tanh$ (expression)**

Returns the hyperbolic tangent of *number* or *expression*, which can be real or complex.

$\tanh \ 1.2 \text{ ENTER} \quad 0.833654607012$

**$\tanh$ list**

Returns a list in which each element is the hyperbolic tangent of the corresponding element in *list*.

$\tanh \ {0,1.2} \text{ ENTER} \quad \{0 \ 0.833654607012\}$

#### $\tanh^{-1}$

**$\tanh^{-1}$**

**MATH HYP** menu

**$\tanh^{-1}$ number or $\tanh^{-1}$ (expression)**

Returns the inverse hyperbolic tangent of *number* or *expression*, which can be real or complex.

$\tanh^{-1} \ 0 \text{ ENTER} \quad 0$

In **RectC** complex number mode:

$\tanh^{-1} \ {0,2.1} \text{ ENTER} \quad \{(0,0) \ (0,0.51804596584\ldots)\}$

**$\tanh^{-1}$ list**

Returns a list in which each element is the inverse hyperbolic tangent of the corresponding element in *list*.
TanLn(expression, xValue)

**Graph DRAW menu**

Draws expression on the current graph and then draws a tangent line at xValue.

In **Func** graphing mode and **Radian** angle mode:

```
ZTrig:TanLn(cos x, π/4)
```

---

Text(row, column, string)

**† GRAPH DRAW menu**

Writes a text string on the current graph beginning at pixel (row, column), where 0 ≤ row ≤ 57 and 0 ≤ column ≤ 123.

Text at the bottom of the graph may be covered by a displayed menu. To remove the menu, press [CLEAR].

Program segment in **Func** graphing mode and a **ZStd** graph screen:

```
:Y1=X sin X
:Text(0,70, "Y1=X sin X")
```

When executed:

---

Then

**† program editor**

**CTL menu**

Refer to syntax information for **If**, beginning on page 305. See the **If:Then:End** and **If:Then:Else:End** syntax.
Trace
† GRAPH menu
Displays the current graph and lets the user trace a function. From a program, press [ENTER] to stop tracing and continue with the program.

Transpose: T
MATRX MATH menu
Returns a transposed real or complex matrix in which element row, column is swapped with element column, row of matrix. For example:

\[
\begin{bmatrix}
  a & b \\
  c & d
\end{bmatrix}
\]

\[
\begin{bmatrix}
  a & c \\
  b & d
\end{bmatrix}
\]

For complex matrices, the complex conjugate of each element is taken.

\[
\begin{bmatrix}
  1 & 2 \\
  3 & 4
\end{bmatrix}
\]

\[
\begin{bmatrix}
  1 & 3 \\
  2 & 4
\end{bmatrix}
\]

\[
\begin{bmatrix}
  1,2,3 \\
  4,5,6 \\
  7,8,9
\end{bmatrix}
\]

\[
\begin{bmatrix}
  1,4,7 \\
  2,5,8 \\
  3,6,9
\end{bmatrix}
\]

\[
\begin{bmatrix}
  (1,2) & (1,1) \\
  (3,2) & (4,3)
\end{bmatrix}
\]

\[
\begin{bmatrix}
  (1,2) & (1,1) \\
  (3,2) & (4,3)
\end{bmatrix}
\]

\[
\begin{bmatrix}
  (1,-2) & (3,-2) \\
  (1,-1) & (4,-3)
\end{bmatrix}
\]
TwoVar

STAT CALC menu
(TwoVa shows on menu)

TwoVar \( xList, yList, frequencyList \)
Performs two-variable statistical analysis on the real data pairs in \( xList \) and \( yList \), using the frequencies in \( frequencyList \).
Values used for \( xList \), \( yList \), and \( frequencyList \) are stored automatically to the built-in variables \( xStat \), \( yStat \), and \( fStat \), respectively.

TwoVar \( xList, yList \)
Uses frequencies of 1.

TwoVar
Uses \( xStat \), \( yStat \), and \( fStat \) for \( xList \), \( yList \), and \( frequencyList \). These built-in variables must contain valid data of the same dimension; otherwise, an error occurs.

unitV

VECTR MATH menu

unitV \( \text{vector} \)
Returns a unit vector of a real or complex \( \text{vector} \), where:

\[
\text{unitV} [a, b, c] \text{ returns } \left[ \frac{a}{\text{norm}}, \frac{b}{\text{norm}}, \frac{c}{\text{norm}} \right]
\]
and
\[
\text{norm} \text{ is } \sqrt{a^2+b^2+c^2}.
\]
### vc→li

**LIST OPS menu**

**VECTR OPS menu**

**vc→li vector**

Returns a real or complex vector converted to a list.

- `vc→li [2,7,-8,0]` \( \rightarrow \) \( \{2 \, 7 \, -8 \, 0\}\)
- \( (vc→li [2,7,-8,0])^2 \rightarrow \{4 \, 49 \, 64 \, 0\}\)

**Vector entry:** \[ \]

- \(2n\) \{ i \} and \(2n\) \{ i \}

**Vert**

† **GRAPH DRAW menu**

**Vert xValue**

Draws a vertical line on the current graph at \(xValue\).

- \([4,5,6]\) \(\rightarrow \) \(\text{VEC} \rightarrow \{4 \, 5 \, 6\}\)
- \([5,(2\pi/4)]\) \(\rightarrow \) \(\text{VEC} \rightarrow \{(5\varpi) \, (2.785398163\ldots)\}\)

In **PolarC** complex number mode:

- \([5,(2\pi/4)]\) \(\rightarrow \) \(\text{VEC} \rightarrow \{(5\varpi) \, (2.785398163\ldots)\}\)

**While**

‡ **program editor**

**CTL menu**

**While condition commands-while-true**

Executes commands-while-true as long as condition is true.

- Program segment:
  - `:While J\leq20`
  - `:TEMP+1/J\rightarrow TEMP`
  - `:J+1\rightarrow J`
  - `:Disp "Reciprocal sums to 20",TEMP`
  - `:`
xor
BASE BOOL menu

`integerA xor integerB`

Compares two real integers bit by bit. Internally, both integers are converted to binary. When corresponding bits are compared, the result is 1 if either bit (but not both) is 1; the result is 0 if both bits are 0 or both bits are 1. The returned value is the sum of the bit results.

For example, `78 xor 23 = 89`.

78 = 1001110
23 = 0010111
1011001 = 89

You can enter real numbers instead of integers, but they are truncated automatically before the comparison.

In Dec number base mode:
78 xor 23 [ENTER] 89

In Bin number base mode:
1001110 xor 10111 [ENTER] 89d

xyline
† STAT DRAW menu

`xyline xList,yList`

Draws a line plot on the current graph, using the real data pairs in xList and yList.

`xyline` uses the data in built-in variables xStat and yStat. These variables must contain valid data of the same dimension; otherwise, an error occurs.
ZData
† GRAPH ZOOM menu

Adjusts the window variable values based on the currently defined statistical plots so that all stat data points will be plotted, and then updates the graph screen.

In Func graphing mode:
{1,2,3,4}→XL ENTER (1 2 3 4)
{2,3,4,5}→YL ENTER (2 3 4 5)
Plot1(XL,YL) ENTER Done
ZStd ENTER

ZData ENTER
ZDecm

† GRAPH ZOOM menu

Sets the window variable values such that Δx=Δy=1, and then updates the graph screen with the origin centered on the screen.

xMin=-6.3  yMin=-3.1
xMax=6.3    yMax=3.1
xScl=1      yScl=1

One of the benefits of ZDecm is that you can trace in .1 increments.

In Func graphing mode:
y1=x sin x  [ENTER]  Done
ZStd  [ENTER]

If you trace the graph above, x values start at 0 and increment by .1587301587.
ZDecm  [ENTER]

If you trace this graph, the x values increment by .1.
ZFit
† GRAPH ZOOM menu

Recalculates yMin and yMax to include the minimum and maximum y values of the selected functions between the current xMin and xMax, and then updates the graph screen.
This does not affect xMin and xMax.

In Func graphing mode:
y1=x^2-20 [ENTER]
ZStd [ENTER]

ZIn
† GRAPH ZOOM menu

Zooms in on the part of the graph centered around the current cursor location.
Zoom factors are set by the values of built-in variables xFact and yFact; the default is 4 for both factors.

In Func graphing mode:
y1=x \cdot \sin x [ENTER]
ZStd [ENTER]
ZInt

† GRAPH ZOOM menu

Sets the window variable values so that each pixel is an integer in all directions (Δx=Δy=1), sets \( xScl=yScl=10 \), and then updates the graph screen.

The current cursor location becomes the center of the new graph.

One of the benefits of ZInt is that you can trace in whole number increments.

In Func graphing mode:

\[
y_1=\text{der1}(x^2-20, x) \quad \boxed{\text{ENTER}} \quad \text{Done}
\]

\[
\text{ZStd} \quad \boxed{\text{ENTER}}
\]

If you trace the graph above, \( x \) values start at 0 and increment by \( 158.7301587 \).

\[
\text{ZInt} \quad \boxed{\text{ENTER}}
\]

If you trace this graph, \( x \) values increment by 1.
**ZOut**

† GRAPH ZOOM menu

Zooms out to display more of the graph, centered around the current cursor location.

Zoom factors are set by the values of built-in variables `xFact` and `yFact`; the default is 4 for both factors.

In **Func** graphing mode:

```
y1=x \sin x \quad \text{ENTER}
ZStd \quad \text{ENTER}
```

**ZPrev**

† GRAPH ZOOM menu

Replots the graph using the window variable values of the graph that was displayed before you executed the previous **ZOOM** instruction.
ZRcl
† GRAPH ZOOM menu

Sets the window variables to values stored previously in the user-defined zoom-window variables, and then updates the graph screen.

To set user-defined zoom-window variables, either:

- Press [GRAPH] [ MORE ] [ MORE ] [ MORE ] [ ENTER] (ZSTO) to store the current graph’s window variables.
- or –
- Store the applicable values to the zoom-window variables, whose names begin with z followed by the regular window variable name. For example, store a value for xMin to zxMin, yMin to zyMin, etc.

ZSqr
† GRAPH ZOOM menu

Sets the window variable values to produce “square” pixels where Δx=Δy, and then updates the graph screen.

The center of the current graph (not necessarily the axes intersection) becomes the center of the new graph.

In other types of zooms, squares may look like rectangles and circles may look like ovals. Use ZSqr for a more accurate shape.

In Func graphing mode:

\[
y_1 = \sqrt{(8^2-x^2)}: y_2 = y_1 \quad \text{ENTER} \quad \text{Done}
\]

ZStd \ ENTER

ZSqr \ ENTER
ZStd

† GRAPH ZOOM menu

Sets the window variables to the standard default values, and then updates the graph screen.

**Func** graphing mode:
- xMin = -10
- yMin = -10
- xMax = 10
- yMax = 10
- xScl = 1
- yScl = 1

**Pol** graphing mode:
- θMin = 0
- θMax = 6.28318530718 (2π)
- qMin = -10
- qMax = 10
- qStep = 0.130899693899… (π/24)
- xScl = 1
- yScl = 1

**Param** graphing mode:
- tMin = 0
- tMax = 6.28318530718 (2π)
- tStep = 0.130899693899… (π/24)
- xScl = 1
- yScl = 1

**DifEq** graphing mode:
- tMin = 0
- tMax = 6.28318530718 (2π)
- tStep = 0.130899693899… (π/24)
- xScl = 1
- tScl = 1
- difTol = .001

In **Func** graphing mode:
- y1 = x sin x
- ZStd
- Done

**Example:**

```
ZStd

In Func graphing mode:
- y1=x sin x
- ZStd
- Done
```

![Graph of y1=x sin x](image)
ZTrig
† GRAPH ZOOM menu

Sets the window variables to preset values appropriate for plotting trig functions in Radian angle mode ($\Delta x=\pi/24$), and then updates the graph screen.

- $x_{\text{Min}}=8.24668071567$
- $y_{\text{Min}}=-4$
- $x_{\text{Max}}=8.24668071567$
- $y_{\text{Max}}=4$
- $x_{\text{Scl}}=1.5707963267949$ ($\pi/2$)
- $y_{\text{Scl}}=1$

In Func graphing mode:

$y_1=\sin x$ (ENTER)

ZStd (ENTER)

Done

ZTrig (ENTER)