GBasic Language Reference

niosV-RTOS on DE1-SoC on Cyclone V FPGA

Technical Manual Edition
April 2026

ECE243, Computer Organization. Avery lor, Ayan Ali.

Contents

Overview

GBasic is a line-numbered BASIC interpreter, similar to QBasic/GW-BASIC, designed for niosV-RTOS on the DE1-SoC. It features integer-only arithmetic, fixed-point graphics support, and hardware-accelerated GPU commands.

Mirroring classic home-computer BASIC, the language emphasizes short edit-test loops, numbered program lines, immediate visual output, and deterministic integer math. Programs execute one statement at a time via `basic_step()`, then yield to the RTOS so UI and system tasks remain responsive.

Each command entry in this manual follows a practical structure: one-line intent, syntax, and concise behavior notes. Sections are ordered from language fundamentals to graphics and hardware-accelerated commands, so you can move efficiently from syntax to performance details.

Data Types

Data Type	Description
integers only	all variables store 32-bit signed integers
fixed-point graphics	world coordinates use FP_SHIFT=10 (multiply by 1024)
arrays	1D integer arrays via DIM, shared pool of 2048 elements (8 KB)

Variables

Property	Description
single-letter	A–Z (26 variables, case-insensitive)
multi-character	up to 64 total with names like PX, MY, SPEED
automatic creation	variables are created on first use, initialized to 0
reserved outputs	`COLOR r,g,b` → C, `TRANSFORM` → X,Y,Z,W, `PROJECT` → X,Y,Z, `ZPSET` → Z

Program Structure

Rule	Description
line numbers	required for all program lines (e.g. `10 PRINT "HELLO"`)
execution order	runs from lowest to highest line number
program capacity	up to 128 lines, 80 characters per line

Operators

Arithmetic		Comparison & Logical	
+	addition	=	equal
−	subtraction	< >	less / greater than
*	multiplication	<= >=	less/greater or equal
/	integer division	<>	not equal
%	modulo (remainder)	AND	logical and
()	grouping	OR	logical or

Interpreter Data Model

At runtime GBasic maintains a compact set of fixed-size stores. The table below summarizes where program state lives and which parts are shared pools.

Runtime Store	Role and Capacity
Program Lines	stores source lines for execution order (up to 128 lines, 80 chars each)
Variables	scalar working state (A–Z plus named variables, up to 64 total)
Control Stacks	loop/subroutine state for flow control (GOSUB depth 16, FOR depth 8)
Array Pool	shared integer array memory (2048 ints total)
Matrix Slots	transform matrices used by 3D math (8 slots of 4x4 fixed-point values)
Mesh Buffer	geometry working set (up to 256 vertices and 512 faces)
Interaction Model	program lines execute into variables, variables index arrays and feed matrix operations, and matrix/array data support mesh transforms and rendering

Execution Cycle

The interpreter follows a cooperative cycle. Programs advance one step at a time, allowing UI and RTOS tasks to remain responsive.

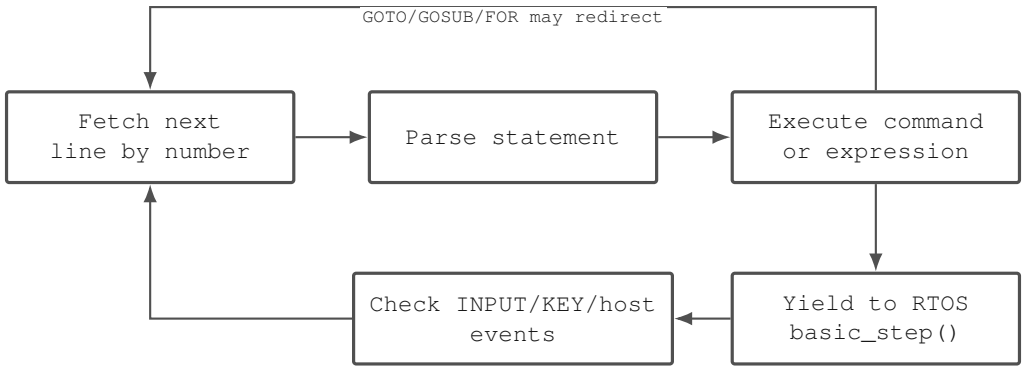

Figure 1: Cooperative interpreter loop integrated with the RTOS scheduler.

Because this cycle yields frequently, long-running graphics or computation loops remain interrupt-friendly as long as program logic avoids blocking input patterns.

Graphics Pipeline Overview

Graphics operations are organized in layers so programs can scale from single-pixel demos to matrix-driven mesh rendering. For best throughput, prefer higher-layer hardware-backed commands when available.

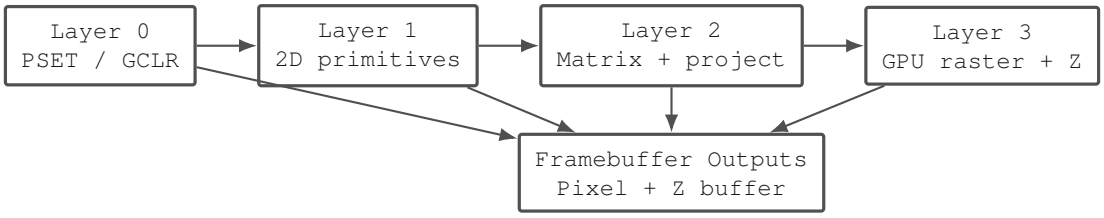

Figure 2: Layered graphics path from basic raster ops to GPU-accelerated depth writes.

Performance Ladder

Use this guide when choosing commands for a new routine. Moving upward generally improves throughput for larger workloads.

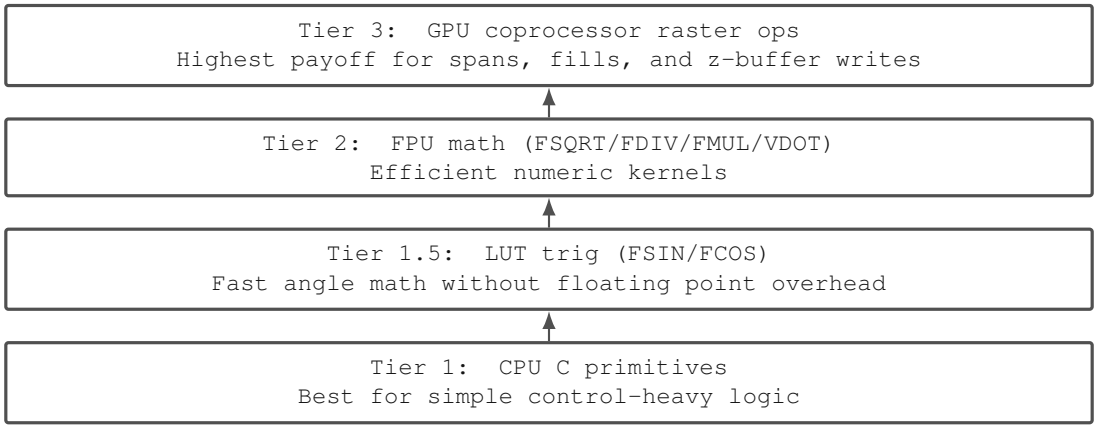

Figure 3: Practical decision ladder for selecting implementation tiers.

Core Commands

Control Flow & Host Bridge

Command	Syntax	Notes
REM	`REM text`	comment line, ignored by interpreter
PRINT	`PRINT expr [; expr ...]`	`;` suppresses space between items; `,` tabs to next 8-column position
LET	`[LET] var = expr`	LET keyword is optional
IF...THEN	`IF cond THEN stmt`	supports inline statements or GOTO targets; comparison: `= < > <= >= <>`; logical: `AND OR`
GOTO	`GOTO linenum`	unconditional jump to line number
GOSUB	`GOSUB linenum`	subroutine call; up to 16 nested levels
RETURN	`RETURN`	return from most recent GOSUB
FOR...NEXT	`FOR v=a TO b [STEP s] ... NEXT v`	counted loop; STEP defaults to 1; up to 8 nested
INPUT	`INPUT ["prompt";] var`	optional prompt with semicolon; stores integer or ASCII value of first character
CLS	`CLS`	clear console output
END	`END`	stop program execution
HOST	`HOST "action", "payload"`	host shell/UI bridge; result in Z (0=success); actions: exec, prefill, mode, close

Arrays

Command	Syntax	Notes
DIM	`DIM name(size) [, ...]`	1D arrays from shared pool (2048 ints, 8 KB); up to 16 arrays; elements init to 0
array access	`name(index) = expr`	0-indexed; out-of-bounds reads return 0, writes silently ignored
ERASE	`ERASE`	reset all arrays and free pool memory

Built-in Functions

General Functions

Function	Description
`KEY()`	non-blocking key poll; returns 0 when no key pending; printable keys return ASCII (letters normalized to uppercase); arrows: Up=1001, Down=1002, Left=1003, Right=1004
`LEN("text")`	length of a string literal
`ASC("text" [, index])`	ASCII code of character at index (default 0); out-of-range returns 0
`VAL("text")`	parse signed integer from a string literal
`POINT(x, y)`	read pixel color from framebuffer
`RGB(r, g, b)`	convert 8-bit RGB to RGB565 color value
`MGET(slot, row, col)`	read one matrix element from a slot
`RND(n)`	pseudo-random integer from 0 to n−1

Mesh Query Functions

Function	Description
`MESHVERTS()`	returns loaded mesh vertex count
`MESHFACES()`	returns loaded mesh face count

Tier 1.5 and Tier 2 Math Functions

These functions provide faster math paths through LUT and FPU-backed operations.

Function	Description
`FSIN(angle)`	LUT-backed sine; returns fixed-point
`FCOS(angle)`	LUT-backed cosine; returns fixed-point
`FSQRT(x)`	hardware square root using niosV FPU
`FDIV(a, b)`	hardware floating-point divide
`FMUL(a, b)`	hardware floating-point multiply
`VDOT(x0,y0,z0, x1,y1,z1)`	vector dot product using FPU

Tier 2 FPU Commands

Command	Notes
`VCROSS x0,y0,z0, x1,y1,z1`	vector cross product; stores result in X, Y, Z
`VNORM x, y, z`	normalize vector; stores result in X, Y, Z as fixed-point

Graphics Commands

Layer 0: Pixel Access

Command	Syntax	Notes
PSET	PSET x, y, color	set a single pixel
GCLR	GCLR [color]	clear entire pixel buffer
GCLROLD	GCLROLD [color]	clear pixel buffer using legacy C primitive path

Layer 1: 2D Primitives

Command	Syntax	Notes
LINE	LINE x0, y0, x1, y1, color	draw line (Bresenham)
RECT	RECT x0, y0, x1, y1, color	draw rectangle outline
CIRCLE	CIRCLE cx, cy, radius, color	draw circle outline
FILLRECT	FILLRECT x0, y0, x1, y1, color	filled rectangle
FILLRECTOLD	FILLRECTOLD x0, y0, x1, y1, color	filled rectangle using legacy C path
FILLCIRCLE	FILLCIRCLE cx, cy, radius, color	filled circle
FILLTRI	FILLTRI x0,y0,x1,y1,x2,y2,color	filled triangle
COLOR	COLOR r, g, b	compute RGB565 and store in variable C

Layer 2: 3D Matrix Operations

Command	Syntax	Notes
MIDENT	MIDENT slot	set matrix to identity
MSET	MSET slot, row, col, value	set matrix element (fixed-point value)
MROTX	MROTX slot, angle	rotation around X axis (integer degrees)
MROTY	MROTY slot, angle	rotation around Y axis
MROTZ	MROTZ slot, angle	rotation around Z axis
MSCALE	MSCALE slot, sx, sy, sz	scale matrix (fixed-point values)
MTRANS	MTRANS slot, tx, ty, tz	translation matrix (fixed-point values)
MMUL	MMUL slotA, slotB, slotOut	matrix multiply (tier 3 hardware)
TRANSFORM	TRANSFORM slot, x, y, z	transform vertex by matrix; stores clip-space result in X, Y, Z, W; input coordinates are fixed-point (world × 1024)
PROJECT	PROJECT slot, x, y, z	transform and project to screen; stores screen coords in X, Y; Z = visibility (1=visible, 0=behind camera)

Mesh Commands

Command	Syntax	Notes
MESHLOAD	MESHLOAD "name"	load mesh from filesystem into internal buffer
MESHGETV	MESHGETV index	load vertex into X, Y, Z variables
MESHGETF	MESHGETF index	load face vertex indices into A, B, C variables
MESHSETV	MESHSETV index, x, y, z	update a mesh vertex using world coordinates
MESHSAVE	MESHSAVE "name"	serialize current mesh and save to filesystem; stores bytes written in Z
MESHDATA	MESHDATA "name", arrayname	read mesh file as raw int16 values into array

Layer 3: GPU Raster Operations

Primary commands below use the hardware-backed path (with internal fallback as needed). Commands ending in OLD force the legacy C primitive path.

Command	Syntax	Notes
HLINE	HLINE x0, x1, y, color	horizontal span (tier 3 hardware)
HLINEOLD	HLINEOLD x0, x1, y, color	horizontal span using legacy C path
VLINE	VLINE x, y0, y1, color	vertical column (tier 3 hardware)
VLINEOLD	VLINEOLD x, y0, y1, color	vertical column using legacy C path
ZBUFCLR	ZBUFCLR [value]	clear z-buffer (tier 3 hardware); defaults to 0xFFFF (max depth)
ZBUFCLROLD	ZBUFCLROLD [value]	clear z-buffer using legacy C path; defaults to 0xFFFF
ZPSET	ZPSET x, y, depth, color	z-buffered pixel write (tier 3 hardware); Z=1 if written, Z=0 if rejected by z-test
ZPSETOLD	ZPSETOLD x, y, depth, color	z-buffered pixel write using legacy C path; Z=1 if written, Z=0 if rejected

Example Programs

Hello World

```
10 PRINT "Hello, World!"
20 END
```

Simple Loop

```
10 FOR I = 1 TO 10
20   PRINT "Count: "; I
30 NEXT I
40 END
```

3D Mesh Rasterizer

```
10 REM === 3D MESH RASTERIZER ===
20 MESHLOAD "MESHTEST"
25 NV = MESHVERTS()
26 NF = MESHFACES()
27 DIM SX(256), SY(256), VIS(256)
30 RX = 0
40 RY = 0
50 GCLR
60 MROTX 0, RX
70 MROTY 3, RY
80 MMUL 3, 0, 4
90 MTRANS 1, 0, 0, 102400
100 MMUL 1, 4, 2
110 REM transform all verts to screen via gpu
120 FOR I = 0 TO NV - 1
130   MESHGETV I
140   PROJECT 2, X, Y, Z
150   SX(I) = X
160   SY(I) = Y
170   VIS(I) = Z
180 NEXT I
190 REM rasterize each face
200 FOR I = 0 TO NF - 1
210   MESHGETF I
220   IF VIS(A) = 0 GOTO 340
230   IF VIS(B) = 0 GOTO 340
240   IF VIS(C) = 0 GOTO 340
250   REM backface cull screen cross product
260   EX = SX(B) - SX(A)
270   EY = SY(B) - SY(A)
280   FX = SX(C) - SX(A)
290   FY = SY(C) - SY(A)
300   IF EX * FY - EY * FX >= 0 GOTO 340
310   BR = 80 + I * 3
315   IF BR > 220 THEN BR = 220
320   FILLTRI SX(A), SY(A), SX(B), SY(B), SX(C), SY(C), RGB(BR, BR, BR + 30)
340 NEXT I
350 PRINT "1=up 2=dn 3=lt 4=rt 0=quit"
360 INPUT N
370 IF N = 1 THEN RX = RX + 15
380 IF N = 2 THEN RX = RX - 15
390 IF N = 3 THEN RY = RY - 15
400 IF N = 4 THEN RY = RY + 15
410 IF N = 0 THEN END
420 GOTO 50
```

Limits & Implementation

Current Constraints

Constraint	Details
1D arrays only	no multi-dimensional arrays (use offset math: `arr(y * W + x)`)
no string variables	string literals are supported in PRINT and string literal functions
no floating-point	all math is integer (use fixed-point scaling)
no user-defined functions	only built-in functions; use GOSUB for subroutines
no line editing	programs loaded from filesystem or entered line-by-line
cooperative execution	programs yield via `basic_step()` for RTOS integration

Fixed-Point Conventions

Convention	Details
world coordinates	multiply by 1024 (FP_ONE) for sub-pixel precision
matrix elements	stored as fixed-point integers (FP_SHIFT=10)
angles	specified in integer degrees (0–359)
colors	RGB565 format (5 bits red, 6 bits green, 5 bits blue)

Memory Constraints

Resource	Limit
program storage	128 lines × 80 chars = 10 KB max
variables	64 × 4 bytes = 256 bytes
arrays	shared pool of 2048 ints (8 KB), up to 16 arrays
stack depth	16 GOSUB levels, 8 FOR loops
matrix slots	8 slots × 4×4 × 4 bytes = 512 bytes
mesh buffer	256 vertices, 512 faces max
filesystem	64 KB RAM filesystem, 64 files max

Execution Model

Property	Details
cooperative multitasking	BASIC programs run via `basic_step()` in RTOS tasks
non-blocking behavior	INPUT pauses execution until user provides input
timer-driven updates	graphics programs wake on timer ticks for smooth animation

Proposed Future Additions

Missing 1990s Accelerator Features

Command	Syntax	Description
BLIT	BLIT sx, sy, w, h, dx, dy	block image transfer
TEXVLINE	TEXVLINE x, y0, y1, u, h, ptr	textured vertical span
TEXHLINE	TEXHLINE x0, x1, y, v, w, ptr	textured horizontal span
COLOR_EXPAND	COLOR_EXPAND src, dst, n, pal	palette expansion
ROP	ROP op, src, dst, count	raster operations (e.g. XOR)

String Support

Feature	Description
string variables	suffix with $ (e.g. NAME$ = "PLAYER")
string functions	LEFT$, RIGHT$, MID$, LEN, CHR$, ASC

Command Reference Summary

Category	Commands / Functions
Control Flow + Host (13)	REM, PRINT, LET, IF…THEN, GOTO, GOSUB, RETURN, FOR…TO…STEP, NEXT, INPUT, CLS, END, HOST
Arrays (3)	DIM, array(index) = expr, ERASE
Built-in Functions (8)	KEY(), LEN("s"), ASC("s"[,i]), VAL("s"), POINT(x,y), RGB(r,g,b), MGET(slot,row,col), RND(n)
Mesh Functions (2)	MESHVERTS(), MESHFACES()
Tier 1.5 / Tier 2 Math (6)	FSIN(angle), FCOS(angle), FSQRT(x), FDIV(a,b), FMUL(a,b), VDOT(x0,y0,z0,x1,y1,z1)
Tier 2 FPU Commands (2)	VCROSS, VNORM
Graphics Layer 0–1 (11)	PSET, GCLR, GCLROLD, LINE, RECT, CIRCLE, FILLRECT, FILLRECTOLD, FILLCIRCLE, FILLTRI, COLOR
Graphics Layer 2 (10)	MIDENT, MSET, MROTX, MROTY, MROTZ, MSCALE, MTRANS, MMUL, TRANSFORM, PROJECT
Graphics Layer 3 (8)	HLINE, HLINEOLD, VLINE, VLINEOLD, ZBUFCLR, ZBUFCLROLD, ZPSET, ZPSETOLD
Mesh Commands (6)	MESHLOAD, MESHGETV, MESHGETF, MESHSETV, MESHSAVE, MESHDATA