Quick Reference Guide for Propeller Assembly
Elements marked with a superscript "s" are also available in Propeller Spin Language.

Directives

ORG	Adjust compile-time cog address pointer	392
FIT	Validate that prev. instructs/data fit entirely in cog	372
RES	Reserve next long(s) for symbol	397

Configuration

CLKSET^s	Set clock mode at run time	361

Process Control

LOCKNEW^s	Check out a new lock	376
LOCKRET^s	Return a lock	376
LOCKCLR^s	Clear a lock by ID	375
LOCKSET^s	Set a lock by ID	377
WAITCNT^s	Pause execution temporarily	411
WAITPEQ^s	Pause exec until pin(s) match designated state(s)	412
WAITPNE^s	Pause exec until pin(s) do not match desig state(s)	413
WAITVID^s	Pause exec until Vid Generator is avail for pixel data	414

Conditions

IF_ALWAYS	Always	369
IF_NEVER	Never	369
IF_E	If equal (Z = 1)	369
IF_NE	If not equal (Z = 0)	369
IF_A	If above (!C & !Z = 1)	369
IF_B	If below (C = 1)	369
IF_AE	If above or equal (C = 0)	369
IF_BE	If below or equal (C | Z = 1)	369
IF_C	If C set	369
IF_NC	If C clear	369
IF_Z	If Z set	369
IF_NZ	If Z clear	369
IF_C_EQ_Z	If C equal to Z	369
IF_C_NE_Z	If C not equal to Z	369
IF_C_AND_Z	If C set and Z set	369
IF_C_AND_NZ	If C set and Z clear	369
IF_NC_AND_Z	If C clear and Z set	369
IF_NC_AND_NZ	If C clear and Z clear	369
IF_C_OR_Z	If C set or Z set	369
IF_C_OR_NZ	If C set or Z clear	369
IF_NC_OR_Z	If C clear or Z set	369
IF_NC_OR_NZ	If C clear or Z clear	369
IF_Z_EQ_C	If Z equal to C	369
IF_Z_NE_C	If Z not equal to C	369
IF_Z_AND_C	If Z set and C set	369
IF_Z_AND_NC	If Z set and C clear	369
IF_NZ_AND_C	If Z clear and C set	369
IF_NZ_AND_NC	If Z clear and C clear	369
IF_Z_OR_C	If Z set or C set	369
IF_Z_OR_NC	If Z set or C clear	369
IF_NZ_OR_C	If Z clear or C set	369
IF_NZ_OR_NC	If Z clear or C clear	369

Cog Control

COGID^s	Get current cog's ID	365
COGINIT^s	Start, or restart, a cog by ID	366
COGSTOP^s	Stop a cog by ID	367

Flow Control

CALL	Jump to addr with intention to return to next instr	360
DJNZ	Decrement value and jump to address if not zero	370
JMP	Jump to address unconditionally	374
JMPRET	Jump to addr with intent to "return" to another addr	374
TJNZ	Test value and jump to address if not zero	409
TJZ	Test value and jump to address if zero	410
RET	Return to stored address	399

Effects

NR	No result (don't write result)	371
WR	Write result	371
WC	Write C status	371
WZ	Write Z status	371

Main Memory Access

RDBYTE	Read byte of main memory	394
RDWORD	Read word of main memory	396
RDLONG	Read long of main memory	395
WRBYTE	Write a byte to main memory	414
WRWORD	Write a word to main memory	416
WRLONG	Write a long to main memory	415

Registers

DIRA^s	Direction Register for 32-bit port A	397
DIRB^s	Direction Register for 32-bit port B (future use)	397
INA^s	Input Register for 32-bit port A (read only)	397
INB^s	Input Register for 32-bit port B (read only) (future)	397
OUTA^s	Output Register for 32-bit port A	397
OUTB^s	Output Register for 32-bit port B (future use)	397
CNT^s	32-bit System Counter Register (read only)	397
CTRA^s	Counter A Control Register	397
CTRB^s	Counter B Control Register	397
FRQA^s	Counter A Frequency Register	397
FRQB^s	Counter B Frequency Register	397
PHSA^s	Counter A Phase Lock Loop (PLL) Register	397
PHSB^s	Counter B Phase Lock Loop (PLL) Register	397
VCFG^s	Video Configuration Register	397
VSCL^s	Video Scale Register	397
PAR^s	Cog Boot Parameter Register (read only)	397

Constants

TRUE^s	Logical true: -1 ($FFFFFFFF)	202
FALSE^s	Logical false: 0 ($00000000)	202
POSX^s	Maximum positive integer: 2,147,483,647	203
NEGX^s	Maximum negative integer: -2,147,483,648	203
PI^s	Floating-point value for PI: ~3.141593	203

Quick Reference Guide for Propeller Assembly

Elements marked with a superscript "s" are also available in Propeller Spin Language.

Common Operations

ABS	Get absolute value of a number	353
ABSNEG	Get negative of number's absolute value	354
NEG	Get negative of a number	386
NEGC	Get a value, or its additive inverse, based on C	386
NEGNC	Get a value or its additive inverse, based on !C	387
NEGZ	Get a value, or its additive inverse, based on Z	389
NEGNZ	Get a value, or its additive inverse, based on !Z	388
MIN	Limit min of unsigned val. to another unsigned val.	379
MINS	Limit min of signed value to another signed value	380
MAX	Limit max of unsigned val. to another unsigned val.	378
MAXS	Limit max of signed value to another signed value	378
ADD	Add two unsigned values	354
ADDABS	Add absolute value to another value	355
ADDS	Add two signed values	356
ADDX	Add two unsigned values plus C	357
ADDSX	Add two signed values plus C	356
SUB	Subtract two unsigned values	403
SUBABS	Subtract an absolute value from another value	404
SUBS	Subtract two signed values	404
SUBX	Subtr unsigned val. plus C frm another unsigned val.	406
SUBSX	Subtr. signed val. plus C from another signed val.	405
SUMC	Sum signed value with another of C-affected sign	406
SUMNC	Sum signed value with another of !C-affected sign	407
SUMZ	Sum signed value with another Z-affected sign	408
SUMNZ	Sum signed value with another of !Z-affected sign	408
MUL	<reserved for future use>	n/a
MULS	<reserved for future use>	n/a
AND	Bitwise AND two values	358
ANDN	Bitwise AND value with NOT of another	359
OR	Bitwise OR two values	392
XOR	Bitwise XOR two values	417
ONES	<reserved for future use>	n/a
ENC	<reserved for future use>	n/a
RCL	Rotate C left into value by specified number of bits	393
RCR	Rotate C right into value by specified number of bits	394
REV	Reverse LSBs of value and zero-extend	399
ROL	Rotate value left by specified number of bits	400
ROR	Rotate value right by specified number of bits	400
SHL	Shift value left by specified number of bits	402
SHR	Shift value right by specified number of bits	402
SAR	Shift value arithmetically right by spec. num. of bits	401
CMP	Compare two unsigned values	362
CMPS	Compare two signed values	362
CMPX	Compare two unsigned values plus C	364
CMPSX	Compare two signed values plus C	364
CMPSUB	Comp unsigned values, subt. 2nd if lesser or equal	363
TEST	Bitwise AND two values to affect flags only	409

MOV	Set a register to a value	380
MOVS	Set a register's source field to a value	382
MOVD	Set a register's destination field to a value	381
MOVI	Set a register's instruction field to a value	381
MUXC	Set discrete bits of a value to the state of C	383
MUXNC	Set discrete bits of a value to the state of !C	384
MUXZ	Set discrete bits of a value to the state of Z	385
MUXNZ	Set discrete bits of a value to the state of !Z	384
HUBOP	Perform a hub operation	373
NOP	No operation, just elapse four cycles	389

Unary Operators

NOTE: All operators shown are constant-expression operators.

+	Positive (+X) unary form of Add	391
-	Negate (-X) unary form of Subtract	391
^^	Square root	391
||	Absolute Value	391
|<	Decode value (0-31) into single-high-bit long	391
>|	Encode long into value (0 - 32) as high-bit priority	391
!	Bitwise: NOT	391
@	Address of symbol	391

Binary Operators

NOTE: All operators shown are constant expression operators.

+	Add	391
-	Subtract	391
*	Multiply and return lower 32 bits (signed)	391
**	Multiply and return upper 32 bits (signed)	391
/	Divide and return quotient (signed)	391
//	Divide and return remainder (signed)	391
#>	Limit minimum (signed)	391
<#	Limit maximum (signed)	391
~>	Shift arithmetic right	391
<<	Bitwise: Shift left	391
>>	Bitwise: Shift right	391
<-	Bitwise: Rotate left	391
->	Bitwise: Rotate right	391
><	Bitwise: Reverse	391
&	Bitwise: AND	391
|	Bitwise: OR	391
^	Bitwise: XOR	391
AND	Boolean: AND (promotes non-0 to -1)	391
OR	Boolean: OR (promotes non-0 to -1)	391
= =	Boolean: Is equal	391
<>	Boolean: Is not equal	391
<	Boolean: Is less than (signed)	391
>	Boolean: Is greater than (signed)	391
=<	Boolean: Is equal or less (signed)	391
=>	Boolean: Is equal or greater (signed)	391