Categorical Listing of Propeller Assembly Language:

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

Directives

ORG	Adjust compile-time cog address pointer.
FIT	Validate that previous instructions/data fits entirely in cog.
RES	Reserve next long(s) for symbol.

Configuration

_CLKMODE^s	Application-defined clock mode (read-only).
_CLKFREQ^s	Application-defined clock frequency (read-only).
CLKSET^s	Set clock mode and clock frequency.
_XINFREQ^s	Application-defined external clock frequency (read-only).
_STACK^s	Application-defined start of stack (read-only).
RCFAST^s	Constant for _CLKMODE: internal fast oscillator.
RCSLOW^s	Constant for _CLKMODE: internal slow oscillator.
XINPUT^s	Constant for _CLKMODE: external clock/oscillator.
XTAL1^s	Constant for _CLKMODE: external low-speed crystal.
XTAL2^s	Constant for _CLKMODE: external medium-speed crystal.
XTAL3^s	Constant for _CLKMODE: external high-speed crystal.
PLL1X^s	Constant for _CLKMODE: external frequency times 1.
PLL2X^s	Constant for _CLKMODE: external frequency times 2.
PLL4X^s	Constant for _CLKMODE: external frequency times 4.
PLL8X^s	Constant for _CLKMODE: external frequency times 8.
PLL16X^s	Constant for _CLKMODE: external frequency times 16.

Cog Control

COGID^s	Get current cog's ID (0-7).
COGINIT^s	Start, or restart, a cog by ID.
COGSTOP^s	Stop a cog by ID.

Process Control

LOCKNEW^{s}	Check out a new semaphore.
LOCKRET^{s}	Return a semaphore.
LOCKCLR^{s}	Clear a semaphore by ID.
LOCKSET^{s}	Set a semaphore by ID.
WAITCNT^{s}	Wait for System Counter to reach a value.
WAITPEQ^{s}	Wait for pin(s) to be equal to value.
WAITPNE^{s}	Wait for pin(s) to be not equal to value .
WAITVID^{s}	Wait for video sync and deliver next color/pixel group.

Flow Control

IF_ALWAYS	Always.
IF_NEVER	Never.
IF_E	If equal (Z = 1).
IF_NE	If not equal (Z = 0).
IF_A	If above (!C & !Z = 1).
IF_B	If below (C = 1) .
IF_AE	If above or equal (C = 0).
IF_BE	If below or equal (C | Z = 1).
IF_C	If C set; p.
IF_NC	If C clear; p.
IF_Z	If Z set; p.
IF_NZ	If Z clear; p.
IF_C_EQ_Z	If C equal to Z.
IF_C_NE_Z	If C not equal to Z.
IF_C_AND_Z	If C set and Z set.
IF_C_AND_NZ	If C set and Z clear.
IF_NC_AND_Z	If C clear and Z set.
IF_NC_AND_NZ	If C clear and Z clear.
IF_C_OR_Z	If C set or Z set.
IF_C_OR_NZ	If C set or Z clear.
IF_NC_OR_Z	If C clear or Z set.
IF_NC_OR_NZ	If C clear or Z clear.
IF_Z_EQ_C	If Z equal to C.

IF_Z_NE_C	If Z not equal to C.
IF_Z_AND_C	If Z set and C set.
IF_Z_AND_NC	If Z set and C clear.
IF_NZ_AND_C	If Z clear and C set.
IF_NZ_AND_NC	If Z clear and C clear.
IF_Z_OR_C	If Z set or C set.
IF_Z_OR_NC	If Z set or C clear.
IF_NZ_OR_C	If Z clear or C set.
IF_NZ_OR_NC	If Z clear or C clear.
CALL	Jump to address with intention to return to next instruction.
DJNZ	Decrement D and jump to address if not zero.
JMP	Jump to address unconditionally.
JMPRET	Jump to address with intention to "return" to another address.
TJNZ	Test D and jump to address if not zero.
TJZ	Test D and jump to address if zero.
RET	Return to stored address.

Result Control

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

Main Memory Access

RDBYTE	Read main memory byte into D, zero extended.
RDWORD	Read main memory word into D, zero extended.
RDLONG	Read main memory long into D.
WRBYTE	Write byte in D to main memory byte.
WRWORD	Write word in D to main memory word.
WRLONG	Write long in D to main memory long.

Instructions

NOP	No operation, just wait one instruction cycle.
ABS	Set D to absolute S.
ABSNEG	Set D to negative of absolute S.
NEG	Set D to –S.
NEGC	Set D to either –S (if C) or S (if !C).
NEGNC	Set D to either S (if C) or -S (if !C).
NEGZ	Set D to either –S (if Z) or S (if !Z).
NEGNZ	Set D to either S (if Z) or -S (if !Z).
MIN	Store lesser of D and S into D (unsigned).
MINS	Store lesser of D and S into D (signed).
MAX	Store greater of D and S into D (unsigned).
MAXS	Store greater of D and S into D (signed).
ADD	Add unsigned S into D.
ADDABS	Add absolute S into D.
ADDS	Add signed S into D.
ADDX	Add unsigned, extended S+C into D.
ADDSX	Add signed, extended S+C into D.
SUB	Subtract unsigned S from D.
SUBABS	Subtract absolute S from D.
SUBS	Subtract signed S from D.
SUBX	Subtract unsigned, extended S+C from D.
SUBSX	Subtract signed, extended S+C from D.
SUMC	Sum either –S (if C) or S (if !C) into D.
SUMNC	Sum either S (if C) or -S (if !C) into D.
SUMZ	Sum either –S (if Z) or S (if !Z) into D.
SUMNZ	Sum either S (if Z) or -S (if !Z) into D.
MUL	<reserved for future use>.
MULS	<reserved for future use>.
AND	Bitwise AND S into D.
ANDN	Bitwise AND !S into D.
OR	Bitwise OR S into D.
XOR	Bitwise XOR S into D.
ONES	<reserved for future use>.

ENC	<reserved for future use>.
RCL	Rotate C left into D by S bits.
RCR	Rotate C right into D by S bits.
REV	Reverse 32 – S[4..0] bottom bits in D and zero extend.
ROL	Rotate D left by S bits.
ROR	Rotate D right by S bits.
SHL	Shift D left by S bits.
SHR	Shift D right by S bits.
SAR	Shift D arithmetically right by S bits.
CMP	Compare unsigned D to S.
CMPS	Compare signed D to S.
CMPX	Compare unsigned, extended D to S+C.
CMPSX	Compare signed, extended D to S+C.
CMPSUB	Compare D to S, if D => S then subtract S from D.
TEST	Binary AND S with D to affect flags only.
MOV	Copy S into D.
MOVS	Copy S bits into D’s Source Field (S[8..0] into D[8..0]).
MOVD	Copy S bits into D’s Destination Field (S[8..0] into D[17..9]).
MOVI	Copy S bits into D’s Instruction Field (S[8..0] into D[31..23]).
MUXC	Copy C to bits in D with S as mask.
MUXNC	Copy !C to bits in D with S as mask.
MUXZ	Copy Z to bits in D with S as mask.
MUXNZ	Copy !Z to bits in D with S as mask.
HUBOP	Hub operation; template for RDBYTE, CLKSET, etc.

Registers

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

Constants

TRUE^{s}	Logical True: -1 ($FFFFFFFF).
FALSE^{s}	Logical False: 0 ($00000000).
POSX^{s}	Maximum positive integer: 2,147,483,647 ($7FFFFFFF).
NEGX^{s}	Maximum negative integer: -2,147,483,648 ($80000000).
PI^{s}	Floating point value for PI: ~3.141593 ($40490FDB).

Unary Operators

NOTE: All operators shown are constant-expression operators.

+	Positive (+X) unary form of Add.
-	Negate (-X); unary form of Subtract.
^^	Square root.
||	Absolute Value.
|<	Decode value (0-31) into single-high-bit long.
>|	Encode long into value (0 - 32) as high-bit priority.
!	Bitwise: NOT.
NOT	Boolean: NOT (promotes non-0 to -1).
@	Address of symbol.

Binary Operators

NOTE: All operators shown are constant expression operators.

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