Welcome to E-XFL.COM

## What is "Embedded - Microcontrollers"?

"Embedded - Microcontrollers" refer to small, integrated circuits designed to perform specific tasks within larger systems. These microcontrollers are essentially compact computers on a single chip, containing a processor core, memory, and programmable input/output peripherals. They are called "embedded" because they are embedded within electronic devices to control various functions, rather than serving as standalone computers. Microcontrollers are crucial in modern electronics, providing the intelligence and control needed for a wide range of applications.

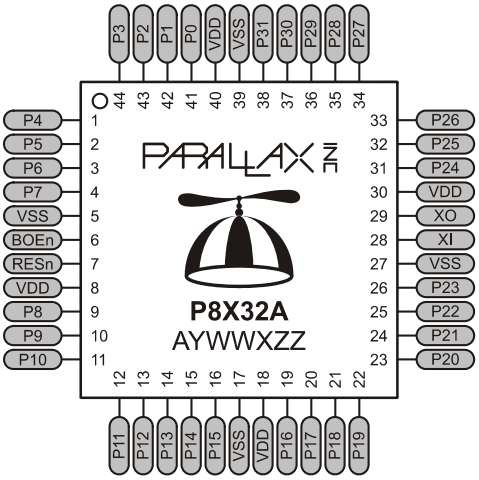
## Applications of "Embedded - Microcontrollers"

| Details | |
|---|---|
| Product Status | Active |
| Core Processor | - |
| Core Size | 32-Bit 8-Core |
| Speed | 80MHz |
| Connectivity | - |
| Peripherals | - |
| Number of I/O | 32 |
| Program Memory Size | 32KB (32K x 8) |
| Program Memory Type | ROM |
| EEPROM Size | - |
| RAM Size | 32K x 8 |
| Voltage - Supply (Vcc/Vdd) | - |
| Data Converters | - |
| Oscillator Type | Internal |
| Operating Temperature | - |
| Mounting Type | Surface Mount |
| Package / Case | 44-LQFP |
| Supplier Device Package | 44-LQFP (10x10) |
| Purchase URL | https://www.e-xfl.com/product-detail/parallax/p8x32a-q44 |

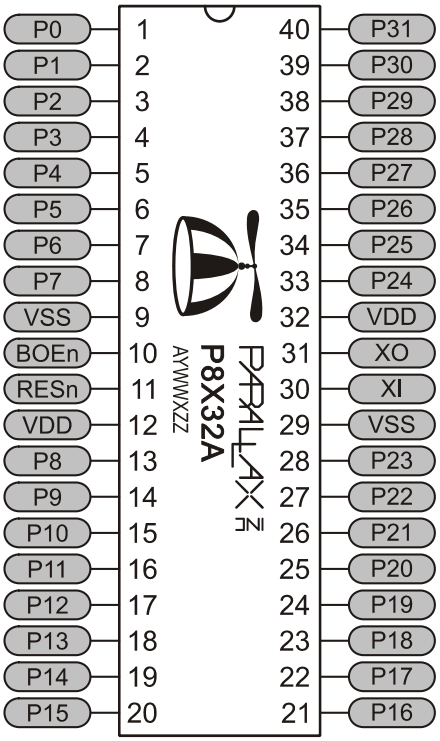
## Table of Contents

# 2.0 CONNECTION DIAGRAMS

## 2.1. Pin Assignments




**LQFP and QFN Packages**




**DIP Package**

## 2.2. Pin Descriptions

**Table 2: Pin Descriptions**

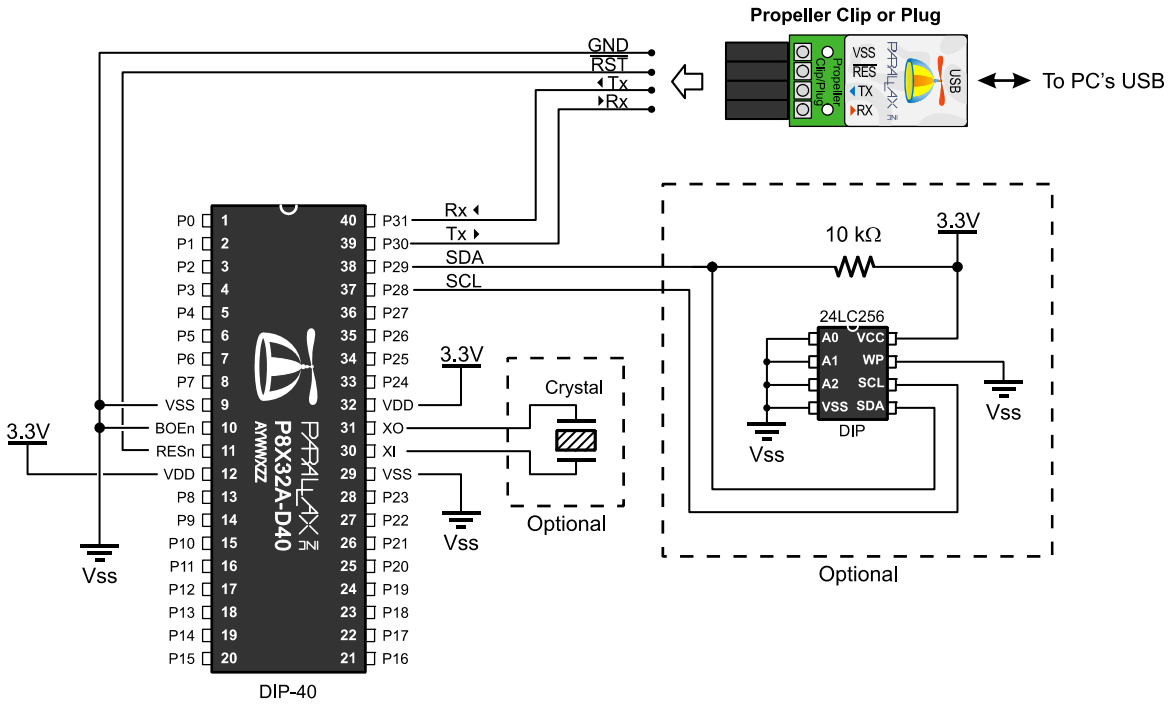
| Pin Name | Direction | Description |
|---|---|---|
| P0 – P31 | I/O | General purpose I/O Port A. Can source/sink 40 mA each at 3.3 VDC. CMOS level logic with threshold of ≈ ½ VDD or 1.6 VDC @ 3.3 VDC.<br><br>The pins shown below have a special purpose upon power-up/reset but are general purpose I/O afterwards.<br>P28 - I2C SCL connection to optional, external EEPROM.<br>P29 - I2C SDA connection to optional, external EEPROM.<br>P30 - Serial Tx to host.<br>P31 - Serial Rx from host. |
| VDD | --- | 3.3 volt power (2.7 – 3.6 VDC) |
| VSS | --- | Ground |
| BOEn | I | Brown Out Enable (active low). Must be connected to either VDD or VSS. If low, RESn becomes a weak output (delivering VDD through 5 kΩ) for monitoring purposes but can still be driven low to cause reset. If high, RESn is CMOS input with Schmitt Trigger. |
| RESn | I/O | Reset (active low). When low, resets the Propeller chip: all cogs disabled and I/O pins floating. Propeller restarts 50 ms after RESn transitions from low to high. |
| XI | I | Crystal Input. Can be connected to output of crystal/oscillator pack (with XO left disconnected), or to one leg of crystal (with XO connected to other leg of crystal or resonator) depending on CLK Register settings. No external resistors or capacitors are required. |
| XO | O | Crystal Output. Provides feedback for an external crystal, or may be left disconnected depending on CLK Register settings. No external resistors or capacitors are required. |

## 2.3. Typical Connection Diagrams

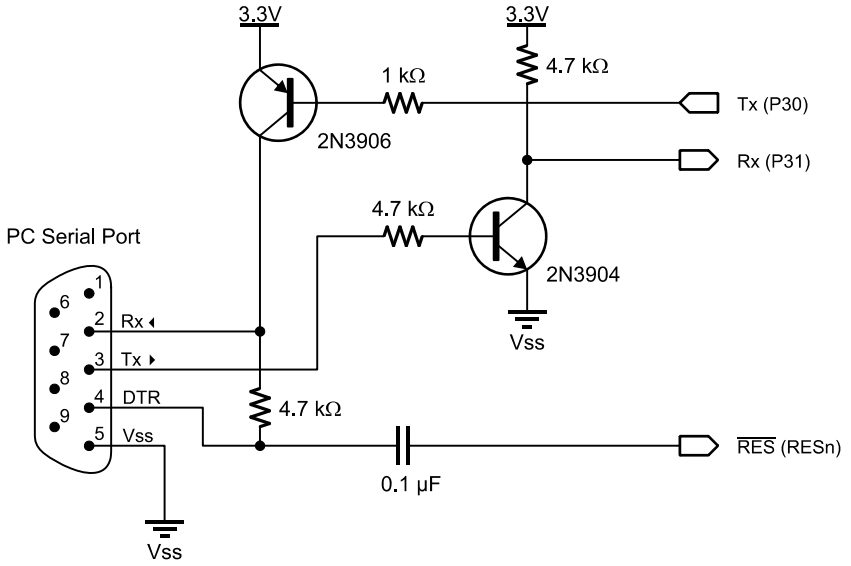
### 2.3.1. Propeller Clip or Propeller Plug Connection - Recommended

Note that the connections to the external oscillator and EEPROM, which are enclosed in dashed lines, are optional.

Propeller Clip, Stock #32200; Propeller Plug, Stock #32201. The Propeller Clip/Plug schematic is available for download from www.parallax.com.




### 2.3.2. Alternative Serial Port Connection



# 3.0 OPERATING PROCEDURES

## 3.1. Boot-Up Procedure

Upon power-up, or reset:

1. The Propeller chip's internal RC oscillator begins running at 20 kHz, then after a 50 ms reset delay, switches to 12 MHz. Then the first processor (Cog 0) loads and runs the built-in Boot Loader program.
2. The Boot Loader performs one or more of the following tasks, in order:
   a. Detects communication from a host, such as a PC, on pins P30 and P31. If communication from a host is detected, the Boot Loader converses with the host to identify the Propeller chip and possibly download a program into global RAM and optionally into an external 32 KB EEPROM.
   b. If no host communication was detected, the Boot Loader looks for an external 32 KB EEPROM on pins P28 and P29. If an EEPROM is detected, the entire 32 KB data image is loaded into the Propeller chip's global RAM.
   c. If no EEPROM was detected, the boot loader stops, Cog 0 is terminated, the Propeller chip goes into shutdown mode, and all I/O pins are set to inputs.
3. If either step 2a or 2b was successful in loading a program into the global RAM, and a suspend command was not given by the host, then Cog 0 is reloaded with the built-in Spin Interpreter and the user code is run from global RAM.

## 3.2. Run-Time Procedure

A Propeller Application is a user program compiled into its binary form and downloaded to the Propeller chip's RAM or external EEPROM. The application consists of code written in the Propeller chip's Spin language (high-level code) with optional Propeller Assembly language components (low-level code). Code written in the Spin language is interpreted during run time by a cog running the Spin Interpreter while code written in Propeller Assembly is run in its pure form directly by a cog. Every Propeller Application consists of at least a little Spin code and may actually be written entirely in Spin or with various amounts of Spin and assembly. The Propeller chip's Spin Interpreter is started in Step 3 of the Boot Up Procedure, above, to get the application running.

Once the boot-up procedure is complete and an application is running in Cog 0, all further activity is defined by the application itself. The application has complete control over things like the internal clock speed, I/O pin usage, configuration registers, and when, what and how many cogs are running at any given time. All of this is variable at run time, as controlled by the application.

## 3.3. Shutdown Procedure

When the Propeller goes into shutdown mode, the internal clock is stopped causing all cogs to halt and all I/O pins are set to input direction (high impedance). Shutdown mode is triggered by one of the three following events:

1. VDD falling below the brown-out threshold (~2.7 VDC), when the brown out circuit is enabled,
2. the RESn pin going low, or
3. the application requests a reboot (see the `REBOOT` command in the Propeller Manual).

Shutdown mode is discontinued when the voltage level rises above the brown-out threshold and the RESn pin is high.

# 4.0 SYSTEM ORGANIZATION

## 4.1. Shared Resources

There are two types of shared resources in the Propeller: 1) common, and 2) mutually-exclusive. Common resources can be accessed at any time by any number of cogs. Mutually-exclusive resources can also be accessed by any number of cogs, but only by one cog at a time. The common resources are the I/O pins and the System Counter. All other shared resources are mutually-exclusive by nature and access to them is controlled by the Hub. See Section 4.4 on page 7.

## 4.2. System Clock

The System Clock (shown as "CLOCK" in Figure 1, page 1) is the central clock source for nearly every component of the Propeller chip. The System Clock's signal comes from one of three possible sources:

- The internal RC oscillator (~12 MHz or ~20 kHz)
- The XI input pin (either functioning as a high-impedance input or a crystal oscillator in conjunction with the XO pin)
- The Clock PLL (phase-locked loop) fed by the XI input

The source is determined by the CLK register's settings, which is selectable at compile time and reselectable at run time. The Hub and internal Bus operate at half the System Clock speed.

## 4.9. Cog Counters

Each cog has two counter modules: CTRA and CTRB. Each counter module can control or monitor up to two I/O pins and perform conditional 32-bit accumulation of its FRQ register into its PHS register on every clock cycle.

Each counter module also has its own phase-locked loop (PLL) which can be used to synthesize frequencies up to 128 MHz.

With a little setup or oversight from the cog, a counter can be used for:

- frequency synthesis
- frequency measurement
- pulse counting
- pulse measurement
- multi-pin state measurement
- pulse-width modulation
- duty-cycle measurement
- digital-to-analog conversion
- analog-to-digital conversion

For some of these operations, the cog can be set up and left in a free-running mode. For others, it may use WAITCNT to time-align counter reads and writes within a loop, creating the effect of a more complex state machine.

Note that for a cog clock frequency of 80 MHz, the counter update period is a mere 12.5 ns. This high speed, combined with 32-bit precision, allows for very dynamic signal generation and measurement.

The design goal for the counter was to create a simple and flexible subsystem which could perform some repetitive task on every clock cycle, thereby freeing the cog to perform some computationally richer super-task. While the counters have only 32 basic operating modes, there is no limit to how they might be used dynamically through software. Integral to this concept is the use of the WAITPEQ, WAITPNE, and WAITCNT instructions, which can event-align or time-align a cog with its counters.

Each counter has three registers:

### 4.9.1. CTRA / CTRB – Control register

The CTR (CTRA and CTRB) register selects the counter's operating mode. As soon as this register is written, the new operating mode goes into effect. Writing a zero to CTR will immediately disable the counter, stopping all pin output and PHS accumulation.

**Table 4: CTRA and CTRB Registers**

| 31 | 30..26 | 25..23 | 22..15 | 14..9 | 8..6 | 5..0 |
|---|---|---|---|---|---|---|
| - | CTRMODE | PLLDIV | - | BPIN | - | APIN |

The CTRMODE field selects one of 32 operating modes for the counter, conveniently written (along with PLLDIV) using the MOVI instruction. These modes of operation are listed in Table 6 on page 11.

**Table 5: PLLDIV Field**

| PLLDIV | %000 | %001 | %010 | %011 | %100 | %101 | %110 | %111 |
|---|---|---|---|---|---|---|---|---|
| Output | VCO / 128 | VCO / 64 | VCO / 32 | VCO / 16 | VCO / 8 | VCO / 4 | VCO / 2 | VCO / 1 |

PLLDIV selects a PLL output tap and may be ignored if not used.

The PLL modes (%00001 to %00011) cause FRQ-to-PHS accumulation to occur every clock cycle. This creates a numerically-controlled oscillator (NCO) in PHS[31], which feeds the counter PLL's reference input. The PLL will multiply this frequency by 16 using its voltage-controlled oscillator (VCO). For stable operation, it is recommended that the VCO frequency be kept within 64 MHz to 128 MHz. This translates to an NCO frequency of 4 MHz to 8 MHz.

The PLLDIV field of the CTR register selects which power-of-two division of the VCO frequency will be used as the final PLL output. This affords a PLL range of 500 kHz to 128 MHz.

BPIN selects a pin to be the secondary I/O. It may be ignored if not used and may be written using the MOVD instruction.

APIN selects a pin to be the primary I/O. It may be ignored if not used and may be written using the MOVS instruction.

### 4.9.2. FRQA / FRQB – Frequency register

FRQ (FRQA and FRQB) holds the value that will be accumulated into the PHS register. For some applications, FRQ may be written once, and then ignored. For others, it may be rapidly modulated.

### 4.9.3. PHSA / PHSB – Phase register

The PHS (PHSA and PHSB) register can be written and read via cog instructions, but it also functions as a free-running accumulator, summing the FRQ register into itself on potentially every clock cycle. Any instruction writing to PHS will override any accumulation for that clock cycle. PHS can only be read through the source operand (same as PAR, CNT, INA, and INB). Beware that doing a read-modify-write instruction on PHS, like "ADD PHSA, #1", will cause the last-written value to be used as the destination operand input, rather than the current accumulation.

VGA mode, each 8-bit color value is written to the pins specified by the VGroup and VPins field. For VGA typically the 8 bits are grouped into 2 bits per primary color and Horizontal and Vertical Sync control lines, but this is up to the software and application of how these bits are used. For composite video each 8-bit color value is composed of 3 fields. Bits 0-2 are the luminance value of the generated signal. Bit 3 is the modulation bit which dictates whether the chroma information will be generated and bits 4-7 indicate the phase angle of the chroma value. When the modulation bit is set to 0, the chroma information is ignored and only the luminance value is output to pins. When the modulation bit is set to 1 the luminance value is modulated ± 1 with a phase angle set by bits 4-7. In order to achieve the full resolution of the chroma value, PLLA should be set to 16 times the modulation frequency (in composite video this is called the color-burst frequency). The PLLB of the cog is used to generate the broadcast frequency; whether this is generated depends on if PLLB is running and the values of VMode and VPins.

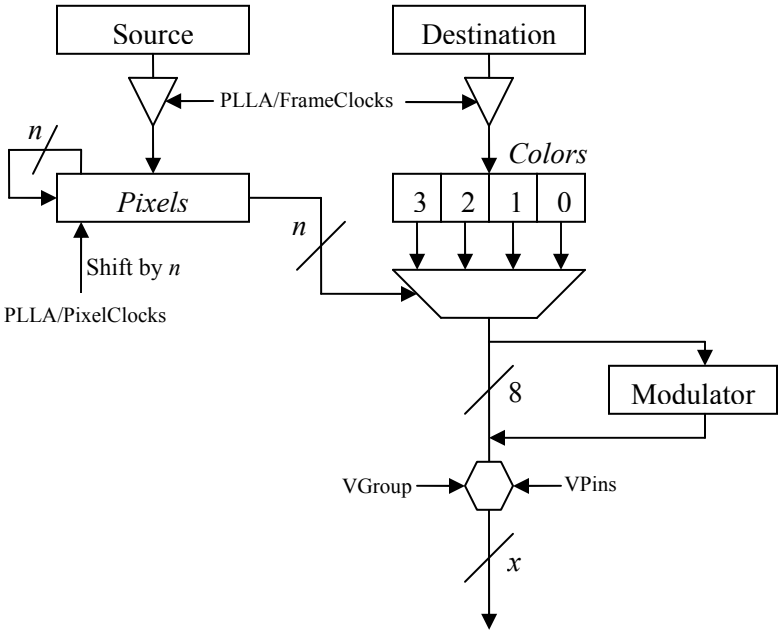
The *Pixels* parameter describes the pixel pattern to display, either 16 pixels or 32 pixels depending on the color depth configuration of the Video Generator. When four-color mode is specified, *Pixels* is a 16x2 bit pattern where each 2-bit pixel is an index into *Colors* on which data pattern should be presented to the pins. When two-color mode is specified, *Pixels* is a 32x1 bit pattern where each bit specifies which of the two color patterns in the lower 16 bits of *Colors* should be output to the pins. The Pixel data is shifted out least significant bits (LSB) first.

When the FrameClocks value is greater than 16 times the PixelClocks value and 4-color mode is specified, the two most significant bits are repeated until FrameClocks PLLA cycles have occurred. When FrameClocks value is greater than 32 times PixelClocks value and 2-color mode is specified, the most significant bit is repeated until FrameClocks PLLA cycles have occurred. When FrameClocks cycles occur and the cog is not in a **WAITVID** instruction, whatever data is on the source and destination busses at the time will be fetched and used. So it is important to be in a **WAITVID** instruction before this occurs.

While the Video Generator was created to display video signals, its potential applications are much more diverse. The Composite Video mode can be used to generate phase-shift keying communications of a granularity of 16 or less and the VGA mode can be used to generate any bit pattern with a fully settable and predictable rate.

Figure 6 is a block diagram of how the VGA mode is organized. The two inverted triangles are the load mechanism for *Pixels* and *Colors; n* is 1 or 2 bits depending on the value of CMode. The inverted trapezoid is a 4-way 8-bit multiplexer that chooses which byte of *Colors* to output. When in composite video mode the Modulator transforms the byte into the luminance and chroma signal and outputs the broadcast signal. VGroup steers the 8 bits to a block of output pins and outputs to those pins which are set to 1 in VPins; this combined functionality is represented by the hexagon.

**Figure 6: Video Generator**



## 4.11. CLK Register

The CLK register is the System Clock configuration control; it determines the source and characteristics of the System Clock. It configures the RC Oscillator, Clock PLL, Crystal Oscillator, and Clock Selector circuits (See the Block Diagram, page 1). It is configured at compile time by the _CLKMODE declaration and is writable at run time through the CLKSET command. Whenever the CLK register is written, a global delay of ~75 µs occurs as the clock source transitions.

Whenever this register is changed, a copy of the value written should be placed in the Clock Mode value location (which is BYTE[4] in Main RAM) and the resulting master clock frequency should be written to the Clock Frequency value location (which is LONG[0] in Main RAM) so that objects which reference this data will have current information for their timing calculations.

Use Spin's CLKSET command when possible (see sections 6.3 and 6.4) since it automatically updates all the above-mentioned locations with the proper information.

**Table 13: Valid Clock Modes**

| Valid Expression | CLK Reg. Value | Valid Expression | CLK Reg. Value |
|---|---|---|---|
| RCFAST | 0_0_0_00_000 | XTAL1 + PLL1X | 0_1_1_01_011 |
| | | XTAL1 + PLL2X | 0_1_1_01_100 |
| RCSLOW | 0_0_0_00_001 | XTAL1 + PLL4X | 0_1_1_01_101 |
| | | XTAL1 + PLL8X | 0_1_1_01_110 |
| XINPUT | 0_0_1_00_010 | XTAL1 + PLL16X | 0_1_1_01_111 |
| | | XTAL2 + PLL1X | 0_1_1_10_011 |
| XTAL1 | 0_0_1_01_010 | XTAL2 + PLL2X | 0_1_1_10_100 |
| XTAL2 | 0_0_1_10_010 | XTAL2 + PLL4X | 0_1_1_10_101 |
| XTAL3 | 0_0_1_11_010 | XTAL2 + PLL8X | 0_1_1_10_110 |
| | | XTAL2 + PLL16X | 0_1_1_10_111 |
| XINPUT + PLL1X | 0_1_1_00_011 | XTAL3 + PLL1X | 0_1_1_11_011 |
| XINPUT + PLL2X | 0_1_1_00_100 | XTAL3 + PLL2X | 0_1_1_11_100 |
| XINPUT + PLL4X | 0_1_1_00_101 | XTAL3 + PLL4X | 0_1_1_11_101 |
| XINPUT + PLL8X | 0_1_1_00_110 | XTAL3 + PLL8X | 0_1_1_11_110 |
| XINPUT + PLL16X | 0_1_1_00_111 | XTAL3 + PLL16X | 0_1_1_11_111 |

**Table 14: CLK Register Fields**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | RESET | PLLENA | OSCENA | OSCM1 | OSCM2 | CLKSEL2 | CLKSEL1 | CLKSEL0 |

| RESET | Effect |
|---|---|
| 0 | Always write '0' here unless you intend to reset the chip. |
| 1 | Same as a hardware reset – reboots the chip. |

| PLLENA | Effect |
|---|---|
| 0 | Disables the PLL circuit. |
| 1 | Enables the PLL circuit. The PLL internally multiplies the XIN pin frequency by 16. OSCENA must be '1' to propagate the XIN signal to the PLL. The PLL's internal frequency must be kept within 64 MHz to 128 MHz – this translates to an XIN frequency range of 4 MHz to 8 MHz. Allow 100 µs for the PLL to stabilize before switching to one of its outputs via the CLKSEL bits. Once the OSC and PLL circuits are enabled and stabilized, you can switch freely among all clock sources by changing the CLKSEL bits. |

| OSCENA | Effect |
|---|---|
| 0 | Disables the OSC circuit |
| 1 | Enables the OSC circuit so that a clock signal can be input to XIN, or so that XIN and XOUT can function together as a feedback oscillator. The OSCM bits select the operating mode of the OSC circuit. Note that no external resistors or capacitors are required for crystals and resonators. Allow a crystal or resonator 10 ms to stabilize before switching to an OSC or PLL output via the CLKSEL bits. When enabling the OSC circuit, the PLL may be enabled at the same time so that they can share the stabilization period. |

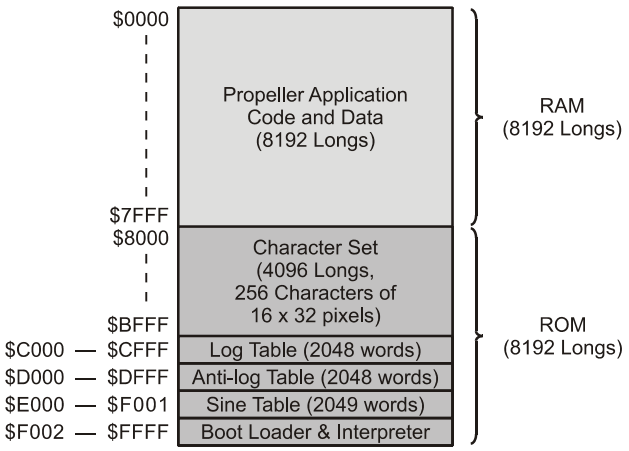
| OSCM1 | OSCM2 | XOUT Resistance | XIN and XOUT Capacitance | Frequency Range |
|---|---|---|---|---|
| 0 | 0 | Infinite | 6 pF (pad only) | DC to 80 MHz Input |
| 0 | 1 | 2000 Ω | 36 pF | 4 MHz to 16 MHz Crystal/Resonator |
| 1 | 0 | 1000 Ω | 26 pF | 8 MHz to 32 MHz Crystal/Resonator |
| 1 | 1 | 500 Ω | 16 pF | 20 MHz to 60 MHz Crystal/Resonator |

| CLKSEL2 | CLKSEL1 | CLKSEL0 | Master Clock | Source | Notes |
|---|---|---|---|---|---|
| 0 | 0 | 0 | ~12 MHz | Internal | No external parts (8 to 20 MHz) |
| 0 | 0 | 1 | ~20 kHz | Internal | No external parts, very low power (13-33 kHz) |
| 0 | 1 | 0 | XIN | OSC | OSCENA must be '1' |
| 0 | 1 | 1 | XIN × 1 | OSC+PLL | OSCENA and PLLENA must be '1' |
| 1 | 0 | 0 | XIN × 2 | OSC+PLL | OSCENA and PLLENA must be '1' |
| 1 | 0 | 1 | XIN × 4 | OSC+PLL | OSCENA and PLLENA must be '1' |
| 1 | 1 | 0 | XIN × 8 | OSC+PLL | OSCENA and PLLENA must be '1' |
| 1 | 1 | 1 | XIN × 16 | OSC+PLL | OSCENA and PLLENA must be '1' |

# 5.0 MEMORY ORGANIZATION

## 5.1. Main Memory

The Main Memory is a block of 64 K bytes (16 K longs) that is accessible by all cogs as a mutually-exclusive resource through the Hub. It consists of 32 KB of RAM and 32 KB of ROM. Main memory is byte, word and long addressable. Words and longs are stored in little endian format; least-significant byte first.




### 5.1.1. Main RAM

The 32 KB of Main RAM is general purpose and is the destination of a Propeller Application either downloaded from a host or from the external 32 KB EEPROM.

### 5.1.2. Main ROM

The 32 KB of Main ROM contains all the code and data resources vital to the Propeller chip's function: character definitions, log, anti-log and sine tables, and the Boot Loader and Spin Interpreter.

### 5.1.3. Character Definitions

The first half of ROM is dedicated to a set of 256 character definitions. Each character definition is 16 pixels wide by 32 pixels tall. These character definitions can be used for video generation, graphical LCD's, printing, etc.

The character set is based on a North American / Western European layout, with many specialized characters added and inserted. There are connecting waveform and schematic building-block characters, Greek characters commonly used in electronics, and several arrows and bullets. (A corresponding Parallax True-Type Font is installed with and used by the Propeller Tool software, and is available to other Windows applications.)

The character definitions are numbered 0 to 255 from left-to-right, then top-to-bottom, per Figure 7 below. They are arranged as follows: Each pair of adjacent even-odd characters is merged together to form 32 longs. The first character pair is located in $8000-$807F. The second pair occupies $8080-$80FF, and so on, until the last pair fills $BF80-$BFFF.

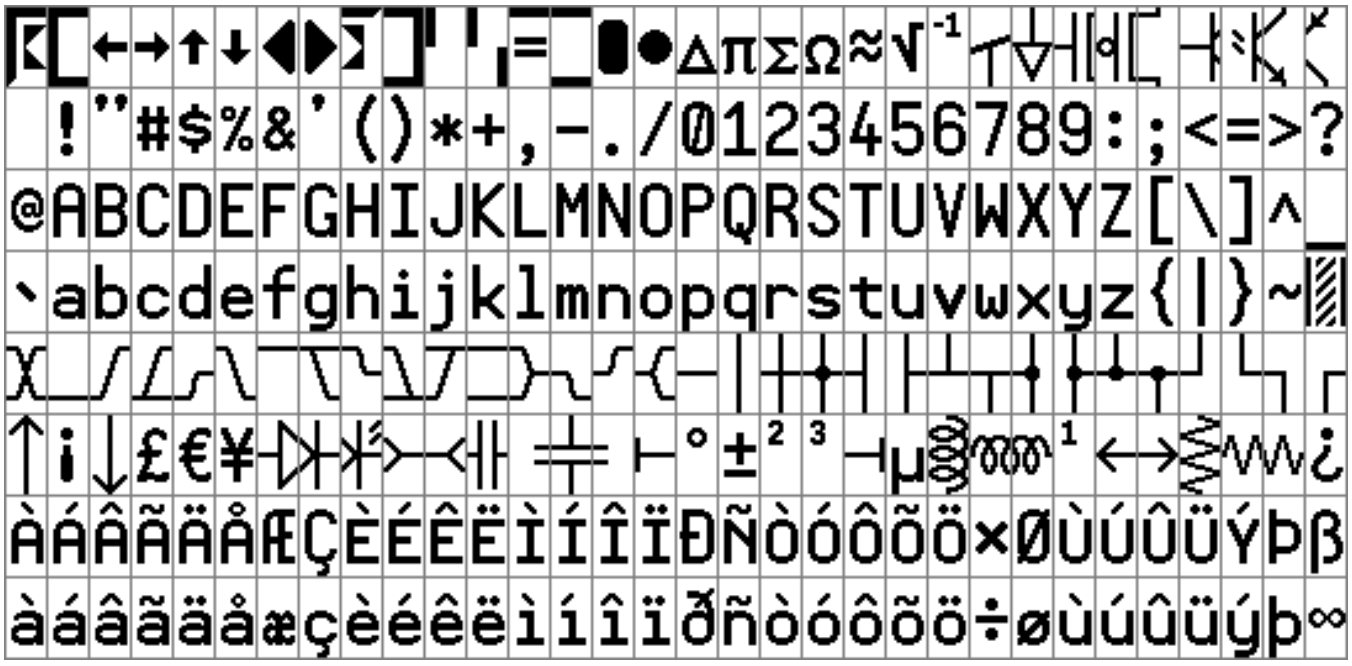


**Figure 7: Propeller Font Character Set**

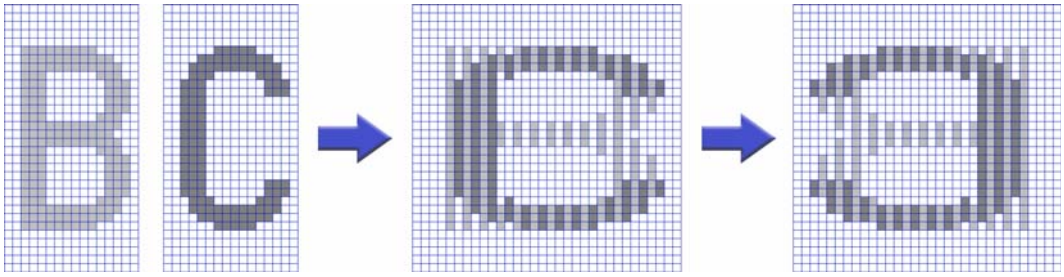


**Figure 8**

**Propeller Character Interleaving**

As shown in Figure 8, The character pairs are merged row-by-row such that each character's 16 horizontal pixels are spaced apart and interleaved with their neighbors' so that the even character takes bits 0, 2, 4, ...30, and the odd character takes bits 1, 3, 5, ...31. The leftmost pixels are in the lowest bits, while the rightmost pixels are in the highest bits. This forms a long for each row of pixels in the character pair. 32 such longs, building from top row down to bottom, make up the complete merged-pair definition. The definitions are encoded in this manner so that a cog's video hardware can handle the merged longs directly, using color selection to display either the even or the odd character.

Some character codes have inescapable meanings, such as 9 for Tab, 10 for Line Feed, and 13 for Carriage Return. These character codes invoke actions and do not equate to static character definitions. For this reason, their character definitions have been used for special four-color characters. These four-color characters are used for drawing 3-D box edges at run-time and are implemented as 16 x 16 pixel cells, as opposed to the normal 16 x 32 pixel cells. They occupy even-odd character pairs 0-1, 8-9, 10-11, and 12-13.

### 5.1.4. Math Function Tables

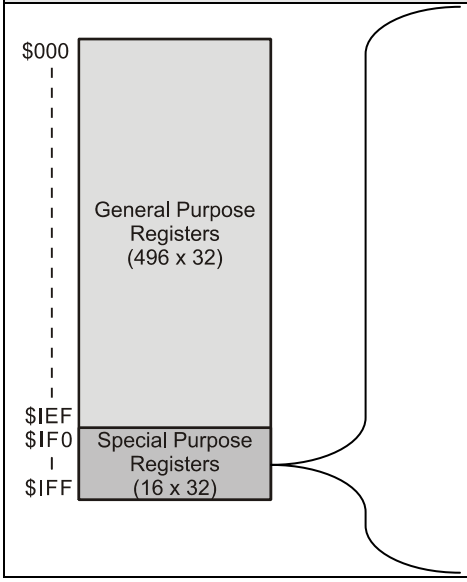
Base-2 Log and Anti-Log tables, each with 2048 unsigned words, facilitate converting values to and from exponent form to facilitate some operations; see the Propeller Manual for access instructions. Also, a sine table provides 2049 unsigned 16-bit sine samples spanning 0° to 90° inclusively (0.0439° resolution).

## 5.2. Cog RAM

As stated in Section 4.3, the Cog RAM is used for executable code, data, and variables, and the last 16 locations serve as interfaces to the System Counter, I/O pins, and local cog peripherals (see Table 15). Cog RAM is long-addressable only.

When a cog is booted up, locations 0 ($000) through 495 ($1EF) are loaded sequentially from Main RAM / ROM and its special purpose locations, 496 ($1F0) through 511 ($1FF), are cleared to zero. Each Special Purpose register may be accessed via its physical address, its predefined name, or indirectly in Spin via a register array variable `SPR` with an index of 0 to 15, the last four bits of the register's address.

**Table 15: Cog RAM Special Purpose Registers**

| Cog RAM Map | Address | Name | Type | Description |
|---|---|---|---|---|
|  | $1F0 | PAR | Read-Only[1] | Boot Parameter |
| | $1F1 | CNT | Read-Only[1] | System Counter |
| | $1F2 | INA | Read-Only[1] | Input States for P31 - P0 |
| | $1F3 | INB | Read-Only[1] | Input States for P63- P32[3] |
| | $1F4 | OUTA | Read/Write | Output States for P31 - P0 |
| | $1F5 | OUTB | Read/Write | Output States for P63 – P32[3] |
| | $1F6 | DIRA | Read/Write | Direction States for P31 - P0 |
| | $1F7 | DIRB | Read/Write | Direction States for P63 - P32[3] |
| | $1F8 | CTRA | Read/Write | Counter A Control |
| | $1F9 | CTRB | Read/Write | Counter B Control |
| | $1FA | FRQA | Read/Write | Counter A Frequency |
| | $1FB | FRQB | Read/Write | Counter B Frequency |
| | $1FC | PHSA | Read/Write[2] | Counter A Phase: |
| | $1FD | PHSB | Read/Write[2] | Counter B Phase |
| | $1FE | VCFG | Read/Write | Video Configuration |
| | $1FF | VSCL | Read/Write | Video Scale |



**Note 1:** Only accessible as a source register (i.e. `MOV` *Dest, Source*).
**Note 2:** Only readable as a Source Register (i.e. `MOV` *Dest, Source*); read-modify-write not possible as a Destination Register.
**Note 3:** Reserved for future use.

## 6.2. Math and Logic Operators

**Table 17: Math and Logic Operators**

| Level[1] | Operator | | Constant Expressions[3] | | Is Unary | Description |
|---|---|---|---|---|---|---|
| | Normal | Assign[2] | Integer | Float | | |
| Highest (0) | -- | always | | | ✓ | Pre-decrement (--X) or post-decrement (X--). |
| | ++ | always | | | ✓ | Pre-increment (++X) or post-increment (X++). |
| | ~ | always | | | ✓ | Sign-extend bit 7 (~X) or post-clear to 0 (X~). |
| | ~~ | always | | | ✓ | Sign-extend bit 15 (~~X) or post-set to -1 (X~~). |
| | ? | always | | | ✓ | Random number forward (?X) or reverse (X?). |
| | @ | never | ✓ | | ✓ | Symbol address. |
| | @@ | never | | | ✓ | Object address plus symbol. |
| 1 | + | never | ✓ | ✓ | ✓ | Positive (+X); unary form of Add. |
| | - | if solo | ✓ | ✓ | ✓ | Negate (-X); unary form of Subtract. |
| | ^^ | if solo | ✓ | ✓ | ✓ | Square root. |
| | \|\| | if solo | ✓ | ✓ | ✓ | Absolute value. |
| | \|< | if solo | ✓ | | ✓ | Bitwise: Decode 0 – 31 to long w/single-high-bit. |
| | >\| | if solo | ✓ | | ✓ | Bitwise: Encode long to 0 – 32; high-bit priority. |
| | ! | if solo | ✓ | | ✓ | Bitwise: NOT. |
| 2 | <- | <-= | ✓ | | | Bitwise: Rotate left. |
| | -> | ->= | ✓ | | | Bitwise: Rotate right. |
| | << | <<= | ✓ | | | Bitwise: Shift left. |
| | >> | >>= | ✓ | | | Bitwise: Shift right. |
| | ~> | ~>= | ✓ | | | Shift arithmetic right. |
| | >< | ><= | ✓ | | | Bitwise: Reverse. |
| 3 | & | &= | ✓ | | | Bitwise: AND. |
| 4 | \| | \|= | ✓ | | | Bitwise: OR. |
| | ^ | ^= | ✓ | | | Bitwise: XOR. |
| 5 | * | *= | ✓ | ✓ | | Multiply and return lower 32 bits (signed). |
| | ** | **= | ✓ | | | Multiply and return upper 32 bits (signed). |
| | / | /= | ✓ | ✓ | | Divide (signed). |
| | // | //= | ✓ | | | Modulus (signed). |
| 6 | + | += | ✓ | ✓ | | Add. |
| | - | -= | ✓ | ✓ | | Subtract. |
| 7 | #> | #>= | ✓ | ✓ | | Limit minimum (signed). |
| | <# | <#= | ✓ | ✓ | | Limit maximum (signed). |
| 8 | < | <= | ✓ | ✓ | | Boolean: Is less than (signed). |
| | > | >= | ✓ | ✓ | | Boolean: Is greater than (signed). |
| | <> | <>= | ✓ | ✓ | | Boolean: Is not equal. |
| | == | === | ✓ | ✓ | | Boolean: Is equal. |
| | =< | =<= | ✓ | ✓ | | Boolean: Is equal or less (signed). |
| | => | =>= | ✓ | ✓ | | Boolean: Is equal or greater (signed). |
| 9 | NOT | if solo | ✓ | ✓ | ✓ | Boolean: NOT (promotes non-0 to -1). |
| 10 | AND | AND= | ✓ | ✓ | | Boolean: AND (promotes non-0 to -1). |
| 11 | OR | OR= | ✓ | ✓ | | Boolean: OR (promotes non-0 to -1). |
| Lowest (12) | = | always | n/a[3] | n/a[3] | | Constant assignment (CON blocks). |
| | := | always | n/a[3] | n/a[3] | | Variable assignment (PUB/PRI blocks). |

[1] Precedence level: higher-level operators evaluate before lower-level operators. Operators in same level are commutable; evaluation order does not matter.
[2] Assignment forms of binary (non-unary) operators are in the lowest precedence (level 12).
[3] Assignment forms of operators are not allowed in constant expressions.

| Spin Command | Returns Value | Description |
|---|---|---|
| ((IF ¦ IFNOT)) *Condition(s)*<br>→! *IfStatement(s)*<br>⟨ELSEIF *Condition(s)*<br>→! *ElseIfStatement(s)*⟩...<br>⟨ELSEIFNOT *Condition(s)*<br>→! *ElseIfStatement(s)*⟩...<br>⟨ELSE<br>→! *ElseStatement(s)*⟩ | | Test condition(s) and execute block of code if valid.<br><br>IF and ELSEIF each test for TRUE. IFNOT and ELSEIFNOT each test for FALSE. |
| INA ⟨[*Pin(s)*]⟩ | ✓ | Input register for 32-bit ports A. |
| LOCKCLR (*ID*) | ✓ | Clear semaphore to false and get its previous state; TRUE or FALSE. |
| LOCKNEW | ✓ | Check out new semaphore and get its ID; 0-7, or -1 if none were available. |
| LOCKRET (*ID*) | | Return semaphore back to semaphore pool, releasing it for future LOCKNEW requests. |
| LOCKSET (*ID*) | ✓ | Set semaphore to true and get its previous state; TRUE or FALSE. |
| LONG *Symbol* ⟨[*Count*]⟩ | | Declare long-sized symbol in VAR block. |
| ⟨*Symbol*⟩ LONG *Data* ⟨[*Count*]⟩ | | Declare long-aligned and/or long-sized data in DAT block. |
| LONG [*BaseAddress*] ⟨[*Offset*]⟩ | ✓ | Read/write long of main memory. |
| LONGFILL (*StartAddress*, *Value*, *Count*) | | Fill longs of main memory with a value. |
| LONGMOVE (*DestAddress*, *SrcAddress*, *Count*) | | Copy longs from one region to another in main memory. |
| LOOKDOWN (*Value*:*ExpressionList*) | ✓ | Get the one-based index of a value in a list. |
| LOOKDOWNZ (*Value*:*ExpressionList*) | ✓ | Get the zero-based index of a value in a list. |
| LOOKUP (*Index*:*ExpressionList*) | ✓ | Get value from a one-based index position of a list. |
| LOOKUPZ (*Index*:*ExpressionList*) | ✓ | Get value from a zero-based index position of a list. |
| NEXT | | Skip remaining statements of REPEAT loop and continue with the next loop iteration. |
| OBJ<br>*Symbol* ⟨[*Count*]⟩:"*Object*" ⟨ ↳ *Symbol* ⟨[*Count*]⟩: "*Object*"⟩... | | Declare symbol object references. |
| OUTA ⟨[*Pin(s)*]⟩ | ✓ | Output register for 32-bit port A. Default is 0 (ground) upon cog startup. |
| PAR | ✓ | Cog Boot Parameter register. |
| PHSA | ✓ | Counter A Phase Lock Loop (PLL) register. |
| PHSB | ✓ | Counter B Phase Lock Loop (PLL) register. |
| PRI *Name* ⟨(*Par* ⟨,*Par*⟩...)⟩ ⟨:*RVal*⟩ ⟨| *LVar* ⟨[*Cnt*]⟩⟩ ⟨,*LVar* ⟨[*Cnt*]⟩⟩...<br>*SourceCodeStatements* | | Declare private method with optional parameters, return value and local variables. |
| PUB *Name* ⟨(*Par* ⟨,*Par*⟩...)⟩ ⟨:*RVal*⟩ ⟨| *LVar* ⟨[*Cnt*]⟩⟩ ⟨,*LVar* ⟨[*Cnt*]⟩⟩...<br>*SourceCodeStatements* | | Declare public method with optional parameters, return value and local variables. |
| QUIT | | Exit from REPEAT loop immediately. |
| REBOOT | | Reset the Propeller chip. |
| REPEAT ⟨*Count*⟩<br>→! *Statement(s)* | | Execute code block repetitively, either infinitely, or for a finite number of iterations. |
| REPEAT *Variable* FROM *Start* TO *Finish* ⟨STEP *Delta*⟩<br>→! *Statement(s)* | | Execute code block repetitively, for finite, counted iterations. |
| REPEAT ((UNTIL ¦ WHILE)) *Condition(s)*<br>→! *Statement(s)* | | Execute code block repetitively, zero-to-many conditional iterations. |
| REPEAT<br>→! *Statement(s)*<br>((UNTIL ¦ WHILE)) *Condition(s)* | | Execute code block repetitively, one-to-many conditional iterations. |
| RESULT | ✓ | Return value variable for PUB/PRI methods. |
| RETURN ⟨*Value*⟩ | ✓ | Exit from PUB/PRI method with optional return *Value*. |
| ROUND (*FloatConstant*) | ✓ | Round floating-point constant to the nearest integer at compile-time, in any block. |
| SPR [*Index*] | ✓ | Special Purpose Register array. |
| STRCOMP (*StringAddress1*, *StringAddress2*) | ✓ | Compare two strings for equality. |
| STRING (*StringExpression*) | ✓ | Declare in-line string constant and get its address. |

## 6.4. Propeller Assembly Instruction Table

The Propeller Assembly Instruction Table lists the instruction's 32-bit opcode, outputs and number of clock cycles. The opcode consists of the instruction bits (**iiiiii**), the "effect" status for the Z flag, C flag, result and indirect/immediate status (**zcri**), the conditional execution bits (**cccc**), and the destination and source bits (**ddddddddd** and **sssssssss**). The meaning of the Z and C flags, if any, is shown in the **Z Result** and **C Result** fields; indicating the meaning of a 1 in those flags. The Result field (**R**) shows the instruction's default behavior for writing (1) or not writing (0) the instruction's result value. The **Clocks** field shows the number of clocks the instruction requires for execution.

- `0 1` Zeros (0) and ones (1) mean binary 0 and 1.
- `i` Lower case "i" denotes a bit that is affected by immediate status.
- `d s` Lower case "d" and "s" indicate destination and source bits.
- `?` Question marks denote bits that are dynamically set by the compiler.
- `---` Hyphens indicate items that are not applicable or not important.
- `..` Double-periods represent a range of contiguous values.

| iiiiii zcri cccc ddddddddd sssssssss | Instruction | Description | Z Result | C Result | R | Clocks |
|---|---|---|---|---|---|---|
| 000000 000i 1111 ddddddddd sssssssss | **WRBYTE** D,S | Write D[7..0] to main memory byte S[15..0] | - | - | 0 | 8..23 * |
| 000000 001i 1111 ddddddddd sssssssss | **RDBYTE** D,S | Read main memory byte S[15..0] into D (0-extended) | Result = 0 | - | 1 | 8..23 * |
| 000001 000i 1111 ddddddddd sssssssss | **WRWORD** D,S | Write D[15..0] to main memory word S[15..1] | - | - | 0 | 8..23 * |
| 000001 001i 1111 ddddddddd sssssssss | **RDWORD** D,S | Read main memory word S[15..1] into D (0-extended) | Result = 0 | - | 1 | 8..23 * |
| 000010 000i 1111 ddddddddd sssssssss | **WRLONG** D,S | Write D to main memory long S[15..2] | - | - | 0 | 8..23 * |
| 000010 001i 1111 ddddddddd sssssssss | **RDLONG** D,S | Read main memory long S[15..2] into D | Result = 0 | - | 1 | 8..23 * |
| 000011 000i 1111 ddddddddd sssssssss | **HUBOP** D,S | Perform hub operation according to S | Result = 0 | - | 0 | 8..23 * |
| 000011 0001 1111 ddddddddd ------000 | **CLKSET** D | Set the global CLK register to D[7..0] | - | - | 0 | 8..23 * |
| 000011 0011 1111 ddddddddd ------001 | **COGID** D | Get this cog number (0..7) into D | ID = 0 | 0 | 1 | 8..23 * |
| 000011 0001 1111 ddddddddd ------010 | **COGINIT** D | Initialize a cog according to D | ID = 0 | No cog free | 0 | 8..23 * |
| 000011 0001 1111 ddddddddd ------011 | **COGSTOP** D | Stop cog number D[2..0] | Stopped ID = 0 | No Cog Free | 0 | 8..23 * |
| 000011 0011 1111 ddddddddd ------100 | **LOCKNEW** D | Checkout a new LOCK number (0..7) into D | ID = 0 | No lock free | 1 | 8..23 * |
| 000011 0001 1111 ddddddddd ------101 | **LOCKRET** D | Return lock number D[2..0] | ID = 0 | No lock free | 0 | 8..23 * |
| 000011 0001 1111 ddddddddd ------110 | **LOCKSET** D | Set lock number D[2..0] | ID = 0 | Prior lock state | 0 | 8..23 * |
| 000011 0001 1111 ddddddddd ------111 | **LOCKCLR** D | Clear lock number D[2..0] | ID = 0 | Prior lock state | 0 | 8..23 * |
| 000100 001i 1111 ddddddddd sssssssss | MUL D,S | Multiply unsigned D[15..0] by S[15..0] | Result = 0 | - | 1 | future |
| 000101 001i 1111 ddddddddd sssssssss | MULS D,S | Multiply signed D[15..0] by S[15..0] | Result = 0 | - | 1 | future |
| 000110 001i 1111 ddddddddd sssssssss | ENC D,S | Encode magnitude of S into D, result = 0..31 | Result = 0 | - | 1 | future |
| 000111 001i 1111 ddddddddd sssssssss | ONES D,S | Get number of 1's in S into D, result = 0..31 | Result = 0 | - | 1 | future |
| 001000 001i 1111 ddddddddd sssssssss | **ROR** D,S | Rotate D right by S[4..0] bits | Result = 0 | D[0] | 1 | 4 |
| 001001 001i 1111 ddddddddd sssssssss | **ROL** D,S | Rotate D left by S[4..0] bits | Result = 0 | D[31] | 1 | 4 |
| 001010 001i 1111 ddddddddd sssssssss | **SHR** D,S | Shift D right by S[4..0] bits, set new MSB to 0 | Result = 0 | D[0] | 1 | 4 |
| 001011 001i 1111 ddddddddd sssssssss | **SHL** D,S | Shift D left by S[4..0] bits, set new LSB to 0 | Result = 0 | D[31] | 1 | 4 |
| 001100 001i 1111 ddddddddd sssssssss | **RCR** D,S | Rotate carry right into D by S[4..0] bits | Result = 0 | D[0] | 1 | 4 |
| 001101 001i 1111 ddddddddd sssssssss | **RCL** D,S | Rotate carry left into D by S[4..0] bits | Result = 0 | D[31] | 1 | 4 |
| 001110 001i 1111 ddddddddd sssssssss | **SAR** D,S | Shift D arithmetically right by S[4..0] bits | Result = 0 | D[0] | 1 | 4 |
| 001111 001i 1111 ddddddddd sssssssss | **REV** D,S | Reverse 32–S[4..0] bottom bits in D and 0-extend | Result = 0 | D[0] | 1 | 4 |
| 010000 001i 1111 ddddddddd sssssssss | **MINS** D,S | Set D to S if signed (D < S) | S = 0 | Signed (D < S) | 1 | 4 |
| 010001 001i 1111 ddddddddd sssssssss | **MAXS** D,S | Set D to S if signed (D => S) | S = 0 | Signed (D < S) | 1 | 4 |
| 010010 001i 1111 ddddddddd sssssssss | **MIN** D,S | Set D to S if unsigned (D < S) | S = 0 | Unsigned (D < S) | 1 | 4 |
| 010011 001i 1111 ddddddddd sssssssss | **MAX** D,S | Set D to S if unsigned (D => S) | S = 0 | Unsigned (D < S) | 1 | 4 |
| 010100 001i 1111 ddddddddd sssssssss | **MOVS** D,S | Insert S[8..0] into D[8..0] | Result = 0 | - | 1 | 4 |
| 010101 001i 1111 ddddddddd sssssssss | **MOVD** D,S | Insert S[8..0] into D[17..9] | Result = 0 | - | 1 | 4 |
| 010110 001i 1111 ddddddddd sssssssss | **MOVI** D,S | Insert S[8..0] into D[31..23] | Result = 0 | - | 1 | 4 |
| 010111 001i 1111 ddddddddd sssssssss | **JMPRET** D,S | Insert PC+1 into D[8..0] and set PC to S[8..0] | Result = 0 | - | 1 | 4 |
| 010111 000i 1111 --------- sssssssss | **JMP** S | Set PC to S[8..0] | Result = 0 | - | 0 | 4 |

### 6.4.1. Assembly Conditions

| Condition | Instruction Executes |
|---|---|
| IF_ALWAYS | always |
| IF_NEVER | never |
| IF_E | if equal (Z) |
| IF_NE | if not equal (!Z) |
| IF_A | if above (!C & !Z) |
| IF_B | if below (C) |
| IF_AE | if above/equal (!C) |
| IF_BE | if below/equal (C \| Z) |
| IF_C | if C set |
| IF_NC | if C clear |
| IF_Z | if Z set |
| IF_NZ | if Z clear |
| 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 |

### 6.4.2. Assembly Directives

| Directive | Description |
|---|---|
| FIT ⟨*Address*⟩ | Validate previous instr/data fit below an address. |
| ORG ⟨*Address*⟩ | Adjust compile-time cog address pointer. |
| ⟨*Symbol*⟩ RES ⟨*Count*⟩ | Reserve next long(s) for symbol. |

### 6.4.3. Assembly Effects

| Effect | Results In |
|---|---|
| WC | C Flag modified |
| WZ | Z Flag modified |
| WR | Destination Register modified |
| NR | Destination Register not modified |

### 6.4.4. Assembly Operators

Propeller Assembly code can contain constant expressions, which may use any operators that are allowed in constant expressions. The table (a subset of Table 17) lists the operators allowed in Propeller Assembly.

| Operator | Description |
|---|---|
| + | Add |
| + | Positive (+X); unary form of Add |
| - | Subtract |
| - | Negate (-X); unary form of Subtract |
| * | Multiply and return lower 32 bits (signed) |
| ** | Multiply and return upper 32 bits (signed) |
| / | Divide (signed) |
| // | Modulus (signed) |
| #> | Limit minimum (signed) |
| <# | Limit maximum (signed) |
| ^^ | Square root; unary |
| \|\| | Absolute value; unary |
| ~> | Shift arithmetic right |
| \|< | Bitwise: Decode value (0-31) into single-high-bit long; unary |
| >\| | Bitwise: Encode long into value (0 - 32) as high-bit priority; unary |
| << | Bitwise: Shift left |
| >> | Bitwise: Shift right |
| <- | Bitwise: Rotate left |
| -> | Bitwise: Rotate right |
| >< | Bitwise: Reverse |
| & | Bitwise: AND |
| \| | Bitwise: OR |
| ^ | Bitwise: XOR |
| ! | Bitwise: NOT; unary |
| AND | Boolean: AND (promotes non-0 to -1) |
| OR | Boolean: OR (promotes non-0 to -1) |
| NOT | Boolean: NOT (promotes non-0 to -1); unary |
| == | 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) |
| @ | Symbol address; unary |

# 7.0 ELECTRICAL CHARACTERISTICS

## 7.1. Absolute Maximum Ratings

Stresses in excess of the absolute maximum ratings can cause permanent damage to the device. These are absolute stress ratings only. Functional operation of the device is not implied at these or any other conditions in excess of those given in the remainder of Section 0. Exposure to absolute maximum ratings for extended periods can adversely affect device reliability.

| Table 18: Absolute Maximum Ratings | |
|---|---|
| Ambient temperature under bias | -55 °C to +125 °C |
| Storage temperature | -65 °C to +150 °C |
| Voltage on $V_{dd}$ with respect to $V_{ss}$ | -0.3 V to +4.0 V |
| Voltage on all other pins with respect to $V_{ss}$* | -0.3 V to ($V_{dd}$ + 0.3 V) |
| Total power dissipation | 1 W |
| Max. current out of $V_{ss}$ pins | 300 mA |
| Max. current into $V_{dd}$ pins | 300 mA |
| Max. DC current into an input pin with internal protection diode forward biased | ±500 µA |
| Max. allowable current per I/O pin | 40 mA |
| ESD (Human Body Model) Supply pins | 3 kV |
| ESD (Human Body Model) all non-supply pins | 8 kV |

***Note: I/O pin voltages with respect to Vss may be exceeded if internal protection diode forward bias current is not exceeded.**

## 7.2. DC Characteristics

(Operating temperature range: -55° C < $T_a$ < +125° C unless otherwise noted)

| Symbol | Parameter | Conditions | Min | Typ* | Max | Units |
|---|---|---|---|---|---|---|
| $V_{dd}$ | Supply Voltage | | 2.7 | - | 3.6 | V |
| $V_{ih}$, $V_{il}$ | Logic High<br>Logic Low | | 0.6 $V_{dd}$<br>$V_{ss}$ | | $V_{dd}$<br>0.3 $V_{dd}$ | V<br>V |
| $I_{il}$ | Input Leakage Current | $V_{in}$ = $V_{dd}$ or $V_{ss}$ | -1.0 | | +1.0 | µA |
| $V_{oh}$ | Output High Voltage | $I_{oh}$ = 10 mA, $V_{dd}$ = 3.3 V | 2.85 | | | V |
| $V_{ol}$ | Output Low Voltage | $I_{ol}$ = 10 mA, $V_{dd}$ = 3.3 V | | | 0.4 | V |
| $I_{BO}$ | Brownout Detector Current | | | 3.8 | | µA |
| I | Quiescent Current | RESn = 0V, BOEn = $V_{dd}$, $P_0$-$P_{31}$=0V | | 600 | | nA |

***Note: Data in the Typical ("Typ") column is $T_a$ = 25 °C unless otherwise stated.**

## 7.3. AC Characteristics

(Operating temperature range: -55°C < $T_a$ < +125°C unless otherwise noted)

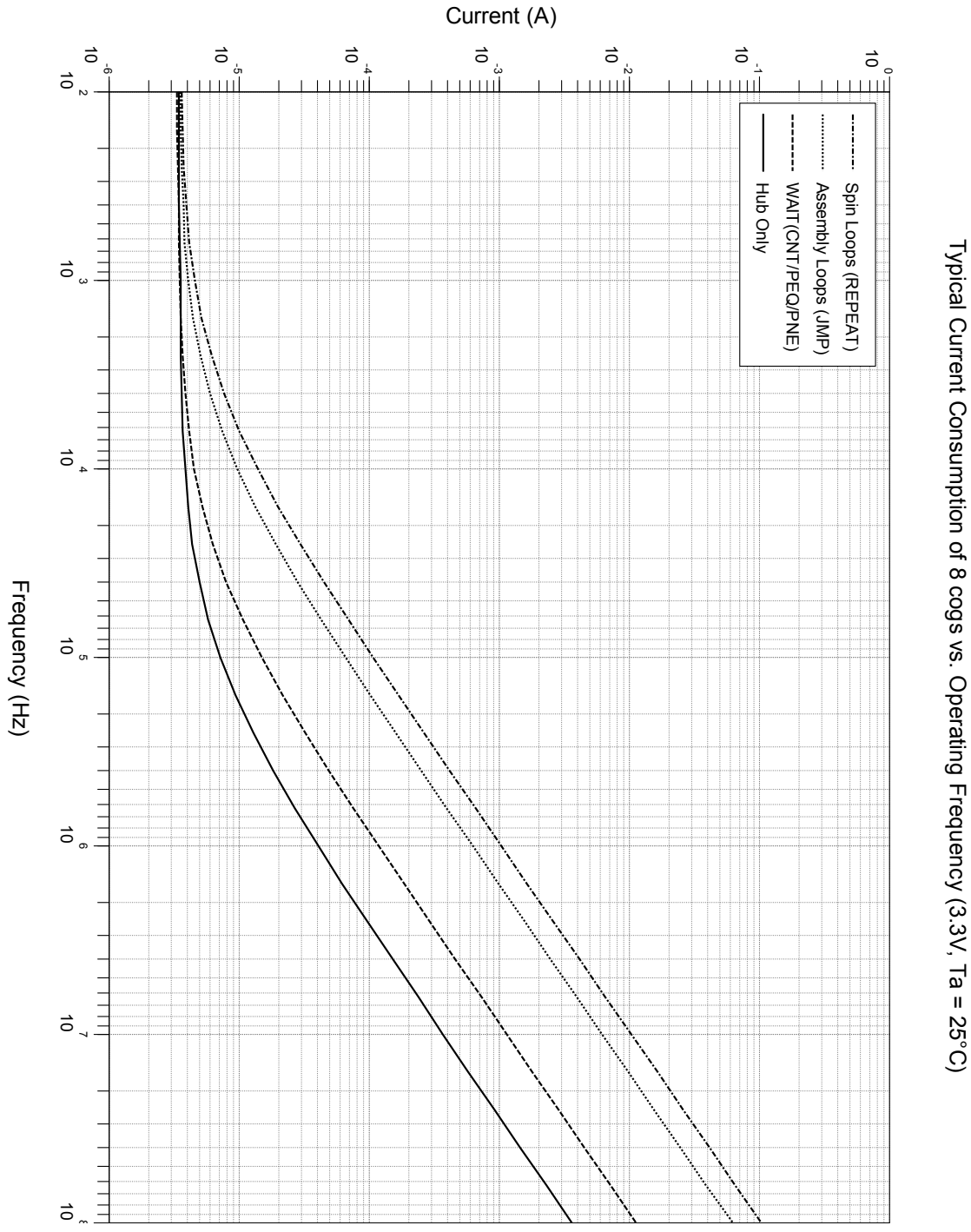
| Symbol | Parameter | Min | Typ* | Max | Units | Condition |
|---|---|---|---|---|---|---|
| $F_{osc}$ | External XI Frequency | DC | - | 80 | MHz | |
| | Oscillator Frequency | DC<br>13<br>8<br>4 | -<br>20<br>12<br>- | 80<br>33<br>20<br>8 | MHz<br>kHz<br>MHz<br>MHz | Direct drive (no PLL)<br>RCSLOW<br>RCFAST<br>Crystal using PLL |
| $C_{in}$ | Input Capacitance | | 6 | - | pF | |

***Note: Data in the Typical ("Typ") column is $T_a$ = 25 °C unless otherwise stated.**

# 8.0 CURRENT CONSUMPTION CHARACTERISTICS

## 8.1. Typical Current Consumption of 8 Cogs

This figure shows the typical current consumption of the Propeller under various operating conditions duplicated across all cogs. Brown out circuitry and the Phase-Locked Loop were disabled for the duration of the test. Current consumption is substantially constant over the operational temperature range.

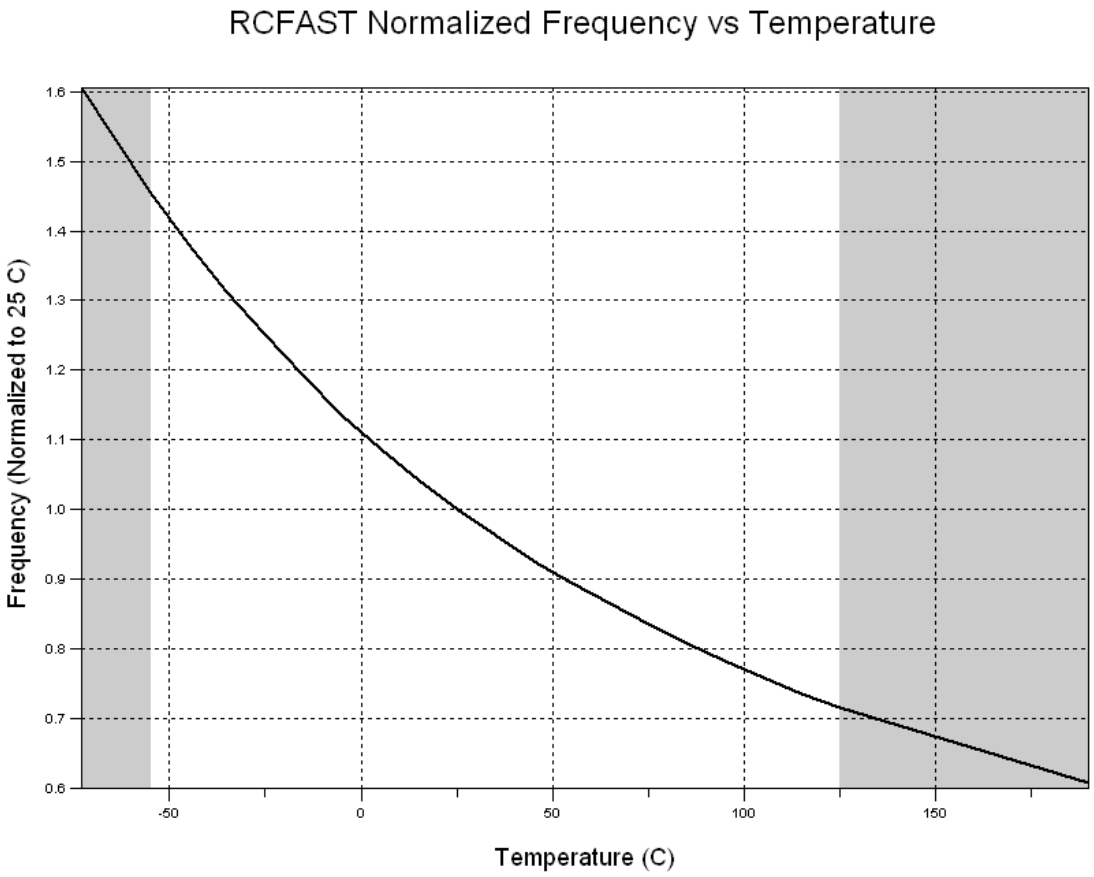



Typical Current Consumption of 8 cogs vs. Operating Frequency (3.3V, Ta = 25°C)

# 9.0 TEMPERATURE CHARACTERISTICS

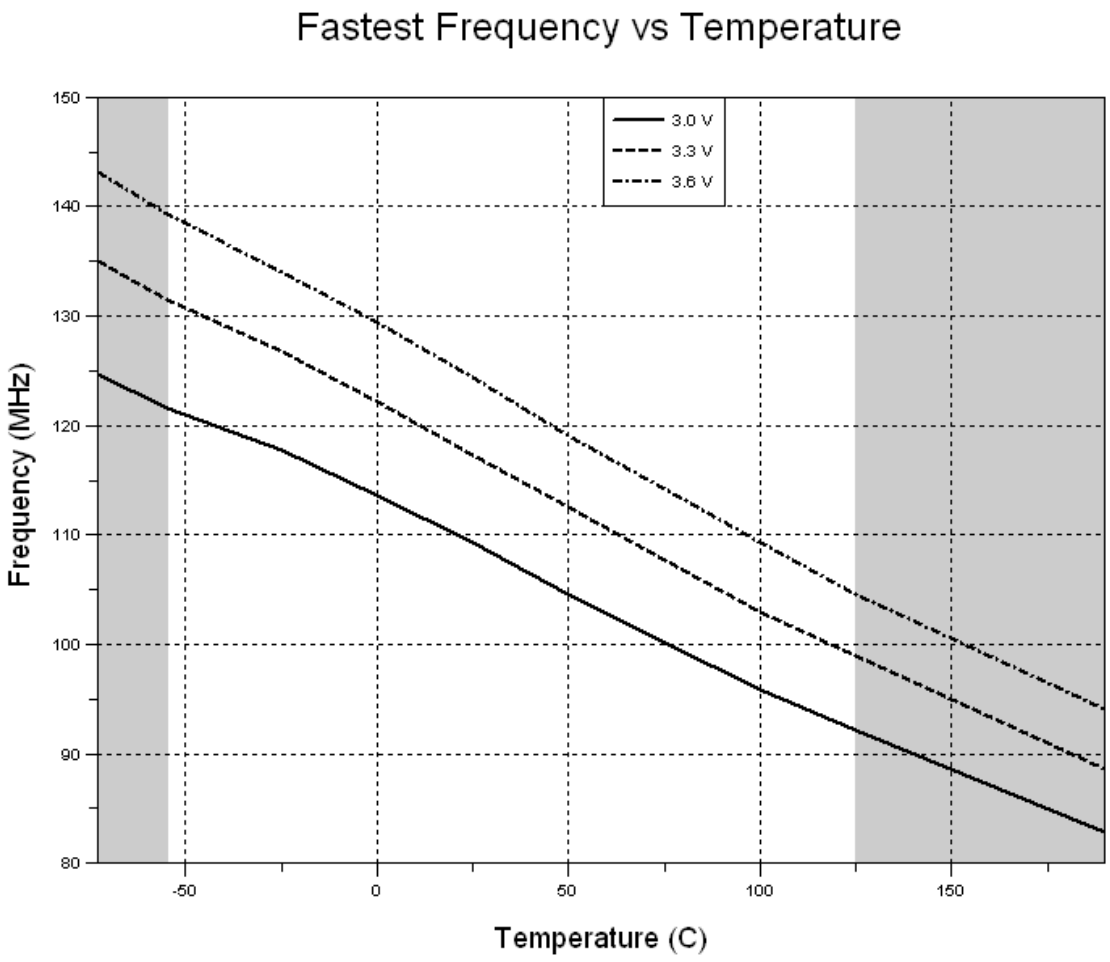
## 9.1. Internal Oscillator Frequency as a Function of Temperature

While the internal oscillator frequency is variable due to process variation, the rate of change as a function of temperature when normalized provides a chip invariant ratio which can be used to calculate the oscillation frequency when the ambient temperature is other than 25 °C (the temperature to which the graph was normalized). The absolute frequency at 25 °C varied from 13.26 to 13.75 MHz in the sample set. The section of the graph which has a white background is the military range of temperature; the sections in grey represent data which is beyond military temperature specification.



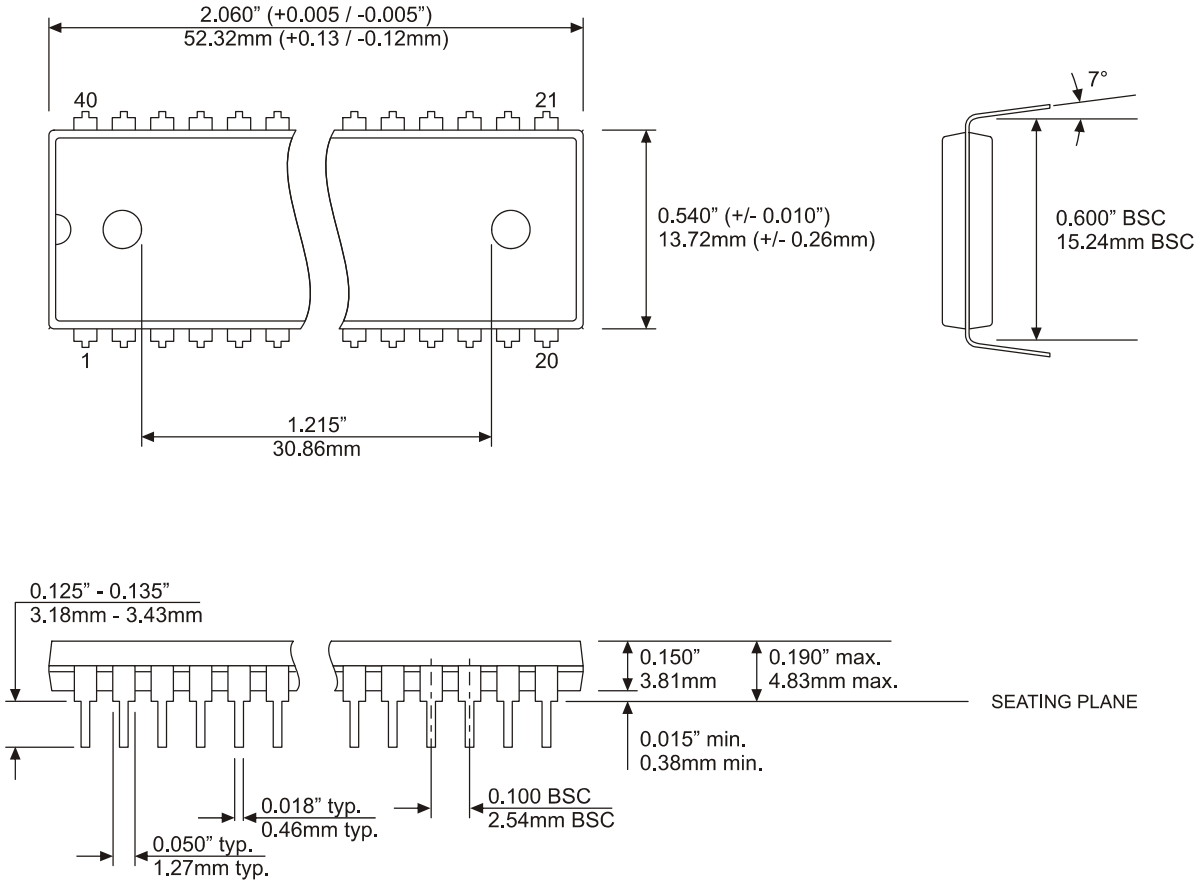
## 9.2. Fastest Operating Frequency as a Function of Temperature

The following graph represents a small sample average of a Propeller chip's fastest operating range. The test was performed in a forced air chamber using code run on all eight cogs, multiple video generators, and counter modules. A frequency was considered successful if the demo ran without fault for one minute. The curves represent an aggressive testing procedure (averaged, forced air, one minute time limit); therefore the designer must de-rate the curve to arrive at a stable frequency for a particular application. Again the grayed regions represent temperatures beyond the military temperature range.

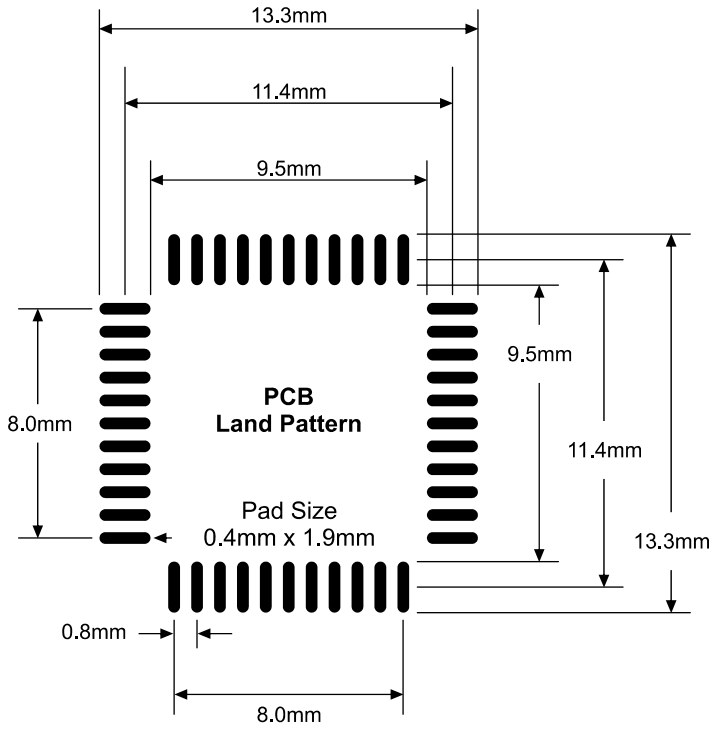


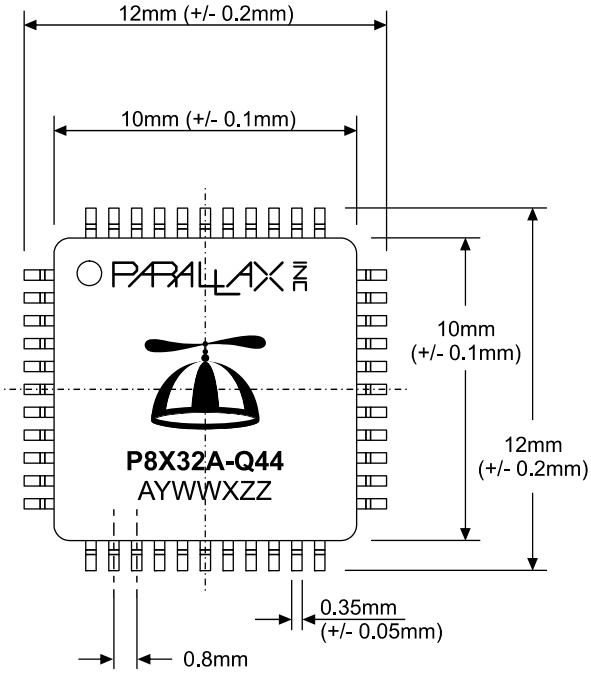
# 10.0 PACKAGE DIMENSIONS

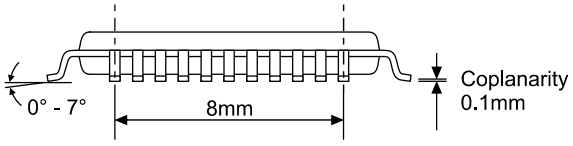
## 10.1. P8X32A-D40 (40-pin DIP)

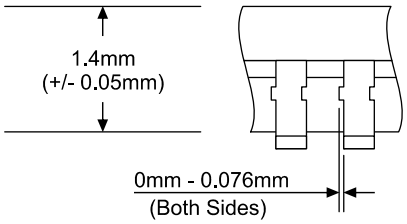


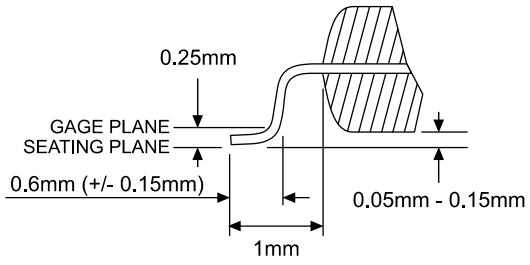
## 10.2. P8X32A-Q44 (44-pin LQFP)



# 11.0 MANUFACTURING INFO

## 11.1. Reflow Peak Temperature

| Package Type | Reflow Peak Temp. |
|---|---|
| DIP | 255+5/-0 °C |
| LQFP | 255+5/-0 °C |
| QFN | 255+5/-0 °C |

## 11.2. Green/RoHS Compliance

All Parallax Semiconductor Propeller P8X32A chip models are certified Green/RoHS Compliant. RoHS, Green, and ISO certificates are available online at www.parallaxsemiconductor.com.

# 12.0 REVISION HISTORY

### 12.1.1. Changes for Version 1.1:

Section 10.3: P8X32A-M44 (44-pin QFN). Image replaced to add stencil pattern diagram. New section inserted: 4.8 Assembly Instruction Execution Stages. Contact Information updated.

### 12.1.2. Changes for Version 1.2:

Section 6.4: Modified table entries for ADD, ADDABS, ADDS, ADDSX, ADDX, CMP, CMPS, CMPSX, CMPX, COGID, COGINIT, COGSTOP, LOCKCLR, LCOKNEW, LOCKRET, LOCKSET, MAX, MAXS, MIN, MINS, SUB, SUBABS, SUBS, SUBSX, SUBX, SUMC, SUMNC, SUMNZ, SUMZ, TEST, TJNZ, TJZ. Section 4.5 updated. Section 5.1: new sentence added at end of paragraph. Section 5.2: new sentence added at end of first paragraph.

### 12.1.3. Changes for Version 1.3

Throughout: updated logo and contact information for Parallax Inc., dba Parallax Semiconductor. Section 7.1: footnote added to Table 18: Absolute Maximum Ratings.

### 12.1.4. Changes for Version 1.4

Section 1.0 changes: 1.3: Key Features and Benefits revised; former sections 1.4 , 1.6 removed. Section 4.4: updated all references to hub timing and replaced both timing diagrams. Section 4.8: reference to hub timing updated. Section 6.4: timing for hub instructions and WAITxxx instructions revised. Former Section 7.0: Propeller Demo Board schematic removed.

# Parallax Semiconductor Contact Information

**Parallax Semiconductor**
**599 Menlo Drive**
**Rocklin, CA 95765**
**USA**

**Phone: (916) 632-4664**
**Fax: (916) 624-8003**

**sales@parallaxsemiconductor.com**
**support@parallaxsemiconductor.com**
**www.parallaxsemiconductor.com**
**http://obex.parallax.com**