Chipsmall Limited consists of a professional team with an average of over 10 year of expertise in the distribution of electronic components. Based in Hongkong, we have already established firm and mutual-benefit business relationships with customers from,Europe,America and south Asia,supplying obsolete and hard-to-find components to meet their specific needs.

With the principle of “Quality Parts,Customers Priority,Honest Operation,and Considerate Service”,our business mainly focus on the distribution of electronic components. Line cards we deal with include Microchip,ALPS,ROHM,Xilinx,Pulse,ON,Everlight and Freescale. Main products comprise IC,Modules,Potentiometer,IC Socket,Relay,Connector.Our parts cover such applications as commercial,industrial, and automotives areas.

We are looking forward to setting up business relationship with you and hope to provide you with the best service and solution. Let us make a better world for our industry!

Contact us

Tel: +86-755-8981 8866 Fax: +86-755-8427 6832
Email & Skype: info@chipsmall.com Web: www.chipsmall.com
Address: A1208, Overseas Decoration Building, #122 Zhenhua RD., Futian, Shenzhen, China

PARALLAX SEMICONDUCTOR
www.parallaxsemiconductor.com
sales@parallaxsemiconductor.com
support@parallaxsemiconductor.com
phone: 916-632-4664 • fax:916-624-8003

Propeller Manual v1.1 Supplement & Errata

This document tracks the technical content changes between Propeller Manual v1.1 and Propeller Manual v1.2. Note that the pagination did not change with this revision, so page numbers below refer to both versions. **Additions/ changes are shown in blue.**

In addition, all significant changes noted here have been marked with comments on the web PDF of the Propeller Manual Version 1.2.

Page 16: Table 1-2: Specifications

RAM/ROM Organization	**Main RAM:** Long (32-bit), Word (16-bit), or Byte (8-bit) addressable **Cog RAM: Long (32-bit) addressable only**

Page 23: First paragraph, last two sentences now read:

The Cog RAM **holds a copy of the Spin Interpreter when executing Spin code, or** is used for executable code, data, **and** variables **when executing Propeller Assembly code. The** last 16 locations serve as interfaces to the System Counter, I/O pins, and local cog peripherals.

Page 23: Table 1-3 Cog RAM Special Purpose Registers

$1F4	OUTA	Read/Write	Output States for P**31**–P0, p. **Error! Bookmark not**

Page 24: First paragraph under Hub, last two sentences:

Hub instructions, the Propeller Assembly instructions that access mutually exclusive resources, require **8** cycles to execute but they first need to be synchronized to the start of the hub access window. It takes up to 15 cycles (**16** minus 1, if we just missed it) to synchronize to the hub access window plus **8** cycles to execute the hub instruction, so hub instructions take from **8** to **23** cycles to complete.

Page 24: New second paragraph inserted under Hub:

Propeller Assembly hub instructions include CLKSET, **p.**Error! Bookmark not defined.**;** COGID, **p.** Error! Bookmark not defined.**;** COGINIT, **p.** Error! Bookmark not defined.**;** COGSTOP, **p.** Error! Bookmark not defined.**;** HUBOP, **p.** Error! Bookmark not defined. LOCKCLR, **p.** Error! Bookmark not defined.**;** LOCKNEW, **p.** Error! Bookmark not defined.**;** LOCKRET, **p.** Error! Bookmark not defined.**;** LOCKSET, **p.** Error! Bookmark not defined.**;** RDBYTE, **p.** Error! Bookmark not defined.**;** RDLONG, **p.** Error! Bookmark not defined.**;** RDWORD, **p.** Error! Bookmark not defined.**;** WRBYTE, **p.** Error! Bookmark not defined. WRLONG, **p.** Error! Bookmark not defined.**;** **and** WRWORD, **p.** Error! Bookmark not defined..

Page 24: Last paragraph under Hub, last sentence:

The hub instruction executes immediately (**8** cycles) leaving an additional **8** cycles for other instructions before the next hub access window arrives.

Page 25: Replaced Figure 1-3: Cog-Hub Interaction – Best Case Scenario and Figure 1-4: Cog-Hub Interaction – Worst Case Scenario

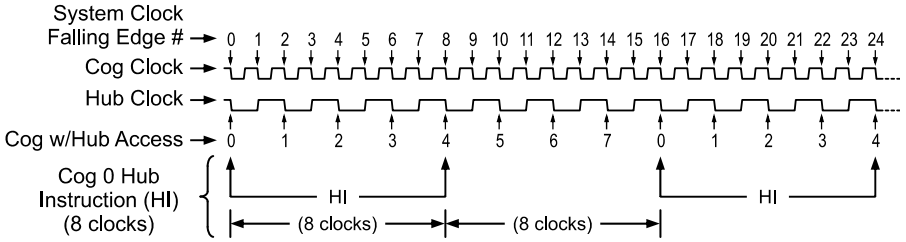

Figure 0-1: Cog-Hub Interaction – Best Case Scenario

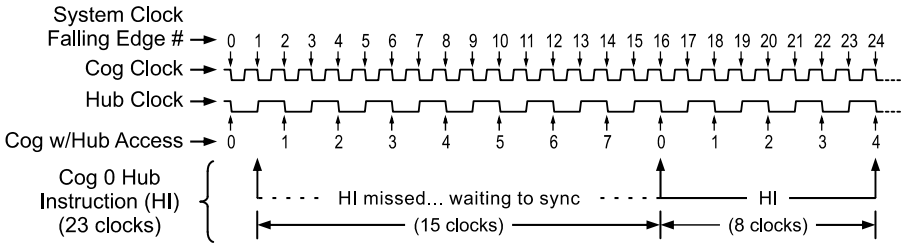

Figure 0-2: Cog-Hub Interaction – Worst Case Scenario

Page 25: First paragraph, second and third sentences:

The cog waits until the next hub access window (15 cycles later) then the hub instruction executes (**8** cycles) for a total of **23** cycles for that hub instruction. Again, there are **8** additional cycles after the hub instruction for other instructions to execute before the next hub access window arrives.

Page 30: Second paragraph under Locks now reads:

The lock bits **exist inside the Hub (outside of Main RAM and Cog RAM) but are globally accessible by every cog** through the hub instructions: LOCKNEW, LOCKRET, LOCKSET, and LOCKCLR. **These special instructions provide** an effective control mechanism **because only one cog at a time can execute them** and **the LOCKSET and LOCKCLR instructions each perform a read and write as one atomic operation. The LOCKNEW and LOCKRET instructions provide a centralized way for** cogs **to** check out **a lock for a new purpose, or** return **a lock that is no longer in use.** See Error! Reference source not found., **Error! Bookmark not defined.**; Error! Reference source not found., **Error! Bookmark not defined.**; Error! Reference source not found., **Error!**

Bookmark not defined.; and `Error! Reference source not found.`, **Error! Bookmark not defined.** for more information.

Pages 43, 144, 158-159, 249, 326:

The "**~>**" operator is now referred to as "**Bitwise** Shift arithmetic right."

Page 68, Table 2 3: Clock Mode Setting Constants

XTAL3	500 Ω	16 pF	External high-speed crystal (20 MHz to **60** MHz)

Page 73: Using CNT Section

Moved second paragraph in the Using CNT section; it now appears as the second paragraph following the first code example.

Page 76: COGINIT syntax definition for *COGID* now reads:

- ***CogID*** is the ID (0 – 7) of the cog to start, or restart. A *CogID* **value of 8 – 15** results in the next available cog being started, if possible; **like a COGNEW command.**

Page 78: COGNEW Explanation, first paragraph divided into two paragraphs with new material:

`COGNEW` starts a new cog and runs either a Spin method or a Propeller Assembly routine within it. **For Spin methods, the new cog's RAM is loaded with the Spin Interpreter from the Main ROM, which then fetches and executes the Spin code tokens from the application image in Main RAM. For Propeller Assembly, the targeted assembly code itself is loaded from the application image in Main RAM into the cog's RAM for execution.**

If **the launch is** successful...

Page 85: First paragraph, second sentence:

`CON` must start in column 1 (the leftmost column) of the line it is on and we recommend the lines following be indented by at least **two spaces for readability.**

Page 103: New heading and paragraphs added to DAT section:

Variables in Propeller Assembly

The assembly example above includes two symbols used as variables, `Delay` and `Time`. Since these are defined as reserved memory (see `RES` directive, page Error! Bookmark not defined.) **they are not initialized; upon launch of the assembly code, they will contain whatever value happened to be in the register they**

occupy in Cog RAM. Assembly variables are defined this way when it is the intention of the code to initialize them during run time.

To create a compile-time, initialized assembly variable instead, use the DAT block's LONG data declaration (page Error! Bookmark not defined.**). It can be used in place of RES, and upon launch of the assembly code, will load the value following it into the register that the variable occupies in Cog RAM.**

If using a mix of LONG-based and RES-based symbols, make sure to define all RES-based symbols last. See Error! Reference source not found. **on page** Error! Bookmark not defined.**, for more information.**

Page 128: LONG Explanation, numbered list:

2) declare **and initialize** long-aligned, and/or long-sized, data in a DAT block, or

Page 129: First sentence under Long Data Declaration (Syntax 2)

In DAT blocks, syntax 2 of LONG is used to declare **and initialize** long-aligned, and/or long-sized data that is compiled as constant values in main memory.

Page 131, starting under first code example, until next header:

The next line, `long[@MyList][0] := Temp + $01234567`, writes a long-sized value to main memory. It sets the value at main memory address $20 to $**012400B7**. The address $20 was calculated from the address of the symbol MyList ($20) plus long offset 0 (0 bytes).

```
long[@MyList][0] ➡ long[$20][0] ➡ long[$20 + (0*4)] ➡
long[$20]
```

The value $**012400B7** was derived from the current value of Temp plus $**01234567**; $BB50 + $01234567 equals $**012400B7**.

Page 141: Second paragraph under Explanation, second sentence.

OBJ must start in column 1 (the leftmost column) of the line it is on and we recommend the lines following be indented by at least **two spaces for readability**.

Page 148: Second example code block, last line:

```
WakeUp      = Reset | Initialize
```

Page 181: PRI syntax element explanations

- ***SourceCodeStatements*** is one or more lines of executable source code that perform the function of the method; **usually indented by at least two spaces for readability**.

Page 182: PUB syntax element explanations

- ***LocalVar*** is a name for a local variable (optional). ***LocalVar*** must **not have the same symbol name as any VAR or CON symbol**, but other methods may also use the same symbol name.
- ***SourceCodeStatements*** is one or more lines of executable source code that perform the function of the method; **usually indented by at least two spaces for readability**.

Page 183: Last paragraph in Public Method Declaration section

All executable statements that belong to a PUB method appear underneath its declaration**. By convention these lines are indenteded by two spaces for readability, but this is not required for functionality. (However, indention is critical for certain command blocks, such as REPEAT and IF.)**

Page 188: REPEAT syntax element explanation

- ***Statement(s)*** is an optional block of one or more lines of code to execute repeatedly. Omitting ***Statement(s)*** is rare, but may be useful in syntax 3 and 4 if ***Condition(s)*** achieves the needed effects**. Indention is required for all statements in the block.**

Page 210: VAR syntax element explanation

- ***Symbol*** is the desired name for the variable**, and must be unique to the object.**

Page 210: Second paragraph added to Explanation section

In Spin, all variables defined in VAR blocks are initialized to zero. Though it is rarely implemented, it is possible to force these variables to initialize to a non-zero value upon the next boot up by including code that updates the application's global variable area in the external EEPROM.

Page 210: Last sentence in first paragraph under Variable Declarations (Syntax 1)

VAR must start in column 1 (the leftmost column) of the line it is on and the lines following it **are usually** indented by at **least two spaces for readability**.

Page 218: First line of code in CON block example:

```
_clkmode = xtal1                'Set for slow crystal
```

Page 252, Opcode Table

Opcode Table "Clocks" column; value range is **8..23**

Pages 254-255: Propeller Assembly Instruction Master Table

In "Clocks" column, the value range is **8..23** for the following rows: CLKSET, COGID, COGINIT, COGSTOP, HUBOP, LOCKCLR, LOCKNEW, LOCKRET, LOCKSET, RDBYTE, RDLONG, RDWORD, WRBYTE, WRLONG, WRWORD.

Page 255, Propeller Assembly Instruction Master Table

WAITCNT, WAITPEQ, WAITPNE rows "Clocks" column value is **6+**.

Page 255, Propeller Assembly Instruction Master Table

WAITVID row "Clocks" column value is **4+**[5].

Page 256: Note 1, sentences one through five:

Hub instructions require **8** to **23** clock cycles to execute depending on the relation between the cog's hub access window and the instruction's moment of execution. The Hub provides a hub access window to each cog every 16 clocks. Because each cog runs independently of the Hub, it must sync to the Hub when executing a hub instruction. The first hub instruction in a sequence will take from 0 to 15 clocks to sync up to the hub access window, and **8** clocks afterwards to execute; thus the **8** to **23** (15 + **8**) clock cycles to execute. After the first hub instruction, there will be **8** (16 – **8**) free clocks before a subsequent hub access window arrives for that cog; enough time to execute two 4-clock instructions without missing the next hub access window.

Page 256: Note 1, new sentence at the end.

Propeller Assembly hub instructions include CLKSET, p. Error! Bookmark not defined.**; COGID, p.** Error! Bookmark not defined.**; COGINIT, p.** Error! Bookmark not defined.**; COGSTOP, p.** Error! Bookmark not defined.**; HUBOP, p.** Error! Bookmark not defined. **LOCKCLR, p.** Error! Bookmark not defined.**; LOCKNEW, p.** Error! Bookmark not defined.**; LOCKRET, p.** Error! Bookmark not defined.**; LOCKSET, p.** Error! Bookmark not defined.**; RDBYTE, p.** Error! Bookmark not defined.**; RDLONG, p.** Error! Bookmark not defined.**; RDWORD, p.** Error! Bookmark not defined.**; WRBYTE, p.** Error! Bookmark not defined.**; WRLONG, p.** Error! Bookmark not defined.**; and WRWORD, p.** Error! Bookmark not defined..

Page 256: added Note 5

> **Note 5: WAITVID consumes 4 clocks itself; however, complete data handoff requires 7 clocks (6 at some frequencies) between frames. The combination of CTRA PLL frequency and VSCL FrameClocks must provide an effective 7 (or 6) system clocks.**

Multiple pages: for the Propeller Assembly Hub instructions

The Opcode Table "Clocks" column value is **8..23**. Also, an Explanation paragraph will read:

> "(Instruction) is a hub instruction. Hub instructions require **8** to **23** clock cycles to execute..."

Propeller Assembly hub instructions include CLKSET, p.**Error! Bookmark not defined.**; COGID, p. **Error! Bookmark not defined.**; COGINIT, p. **Error! Bookmark not defined.**; COGSTOP, p. **Error! Bookmark not defined.**; HUBOP, p. **Error! Bookmark not defined.** LOCKCLR, p. **Error! Bookmark not defined.**; LOCKNEW, p. **Error! Bookmark not defined.**; LOCKRET, p. **Error! Bookmark not defined.**; LOCKSET, p. **Error! Bookmark not defined.**; RDBYTE, p. **Error! Bookmark not defined.**; RDLONG, p. **Error! Bookmark not defined.**; RDWORD, p. **Error! Bookmark not defined.**; WRBYTE, p. **Error! Bookmark not defined.**; WRLONG, p. **Error! Bookmark not defined.**; and WRWORD, p. **Error! Bookmark not defined.**.

Page 268: Second paragraph under Explanation, first sentence.

> **Propeller Assembly** does not use a call stack, so the return address must be stored in a different manner.

Page 300: Second paragraph under Explanation.

> **Propeller Assembly** does not use a call stack, so the return address of a call-type operation must be stored in a different manner.

Multiple Pages:

WAITCNT, WAITPEQ, WAITPNE Opcode Table, "Clocks" column value is **6+** .

Page 371: WAITVID row "Clocks" column value is 4+[1]. Footnote added to table:

> **WAITVID consumes 4 clocks itself; however, complete data handoff requires 7 clocks (6 at some frequencies) between frames. The combination of CTRA PLL frequency and VSCL FrameClocks must provide an effective 7 (or 6) system clocks.**

Page 379: Reserved Word List Table

Added a # symbol to ENC, indicating it is reserved for future use.

Page 380, second paragraph below first code listing, third sentence:

To remedy this**, there is a** special CMPSUB *D*, *S* instruction