PARALLAX

Web Site: www.parallax.com
Forums: forums.parallax.com
Sales: sales@parallax.com
Technical: support@parallax.com

Office: 916-624-8333
Fax: 916-624-8003
Sales: 888-512-1024
Tech Support: (888) 997-8267

ColorPAL Rev. B (#28380): *Color and Ambient Light Sensor and Color Generator*

General Description

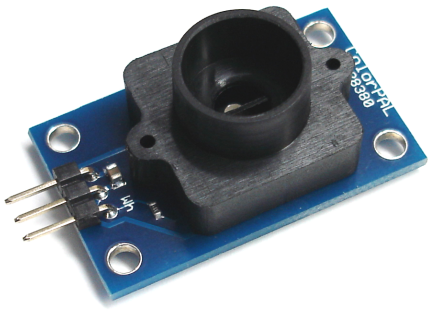
The ColorPAL is a miniature color and light sensor which, through its RGB LED, doubles as a color generator.

Features

- Detects a full range of colors and outputs data as RGB (Red/Green/Blue) components.
- Detects broad-spectrum ambient light with sensitivity down to 44µW/cm^2 per lsb.
- Generates 24-bit color using onboard RGB LED.
- Plugs into servo headers (with optional cable) or wireless breadboards.
- Single-pin interface uses a simple serial protocol to define and initiate color detection and generation.
- Color detection and generation details handled by onboard microcontroller.
- Onboard EEPROM for saving custom color detection and generation programs.
- Autorun feature permits running a pre-designated EEPROM program with only a power supply.

Out of the Box

What's Included



ColorPAL Module

What You Need to Provide

- BASIC Stamp 2 or better or a Propeller, and a carrier board, such as Parallax's Board of Education (BOE) or Propeller Activity Board.
- Optional, but desirable: "servo" extension cable (e.g. Parallax #800-00043).

Quick Start (Color Sensing)

1. Hook up your ColorPAL as shown below in the **Installation** section, using P15 for the signal.
2. Download the color match software zip file from the ColorPAL product page on parallax.com.
3. Load the program **ColorPAL_sense.bs2** into your BASIC Stamp and RUN it.
4. Close the DEBUG window.
5. Start the program **TCS3200_ColorPAL_match.exe** on your PC.
6. Follow the instructions included with the color match zip file.

Quicker Start (Color Sensing)

1. Hook up your ColorPAL as shown below in the **Installation** section, using P15 for the signal.
2. Download the BASIC Stamp program, **ColorPAL_mimic.bs2**, and RUN it in your BASIC Stamp.
3. Calibrate on black and white subjects, using the "b" and "w" keys.
4. Sample a color using the "s" key, then look at the LED to see that color being mimicked.

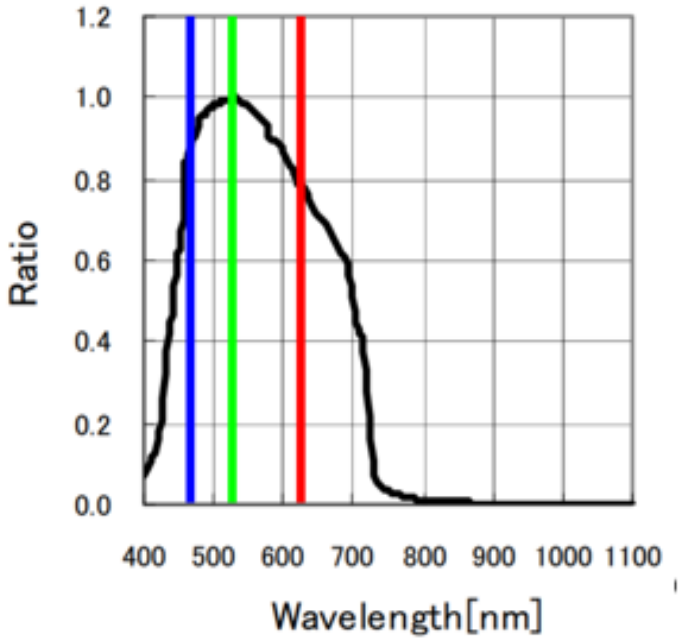
Quickest Start (LED Demo)

1. Hook up your ColorPAL as shown below in the **Installation** section, using P15 for the signal.
2. Download the BASIC Stamp program, **ColorPAL_transit.bs2**, and RUN it in your BASIC Stamp.
3. Hold the ColorPAL above a white sheet of paper, and watch as the LED's colors change, blending from one to another. **CAUTION:** *The LED is much too bright and concentrated to stare at directly. Use the above technique to view the color changes.*

Principle of Operation

The ColorPAL uses an RGB LED to illuminate a sample, one color at a time, along with a broad-spectrum light-to-voltage converter to measure the light reflected back. The amount of light reflected from the sample under illumination from each red, green, and blue LED can be used to determine the sample's color.

The light sensor used in the ColorPAL is a Rohm (www.rohm.com) BH1680FVC, which has the following spectral sensitivity curve (taken from the BH1680FVC datasheet and superimposed with the LED wavelengths:



The sensor outputs a current proportional to *all* the light that it sees, weighted by the above curve, and converted to a voltage via a 5.1K load resistor. Therefore, when a subject is illuminated with a red LED only, it will respond with a voltage proportional to the red component of the subject's color, and similarly with blue and green. When there is ambient light mixed in with the LED's illumination, its effect can be eliminated by sampling first without any LEDs turned on, then subtracting this reading, in turn, from each of the red, green, and blue components. This reference measurement should be taken before each color measurement to eliminate any effects from varying ambient conditions. In the paragraphs that follow, it will be assumed that an ambient reference is taken and subtracted from each measurement discussed.

Because the LED and sensor sit next to each other on the ColorPAL's circuit board, and because the plastic snorkel itself reflects some light back (primarily from its threads), the light response from a completely black subject will be non-zero. For this reason, the black response for each color component has to be determined experimentally, so that it, too, can be subtracted from the overall response. The three components thus measured (with an ambient reference subtracted), **Kr**, **Kg**, and **Kb**, are known as the "black reference". A black reference is typically obtained only once before each measurement session.

The BH1680FVC light sensor does not respond equally to the red, green, and blue LEDs, and those LEDs don't put out equal amounts of light at the red, green, and blue wavelengths. So, in addition to the ambient reference and black reference, it is also necessary to take a "white reference", wherein the ColorPAL is presented with a completely white surface. Again, after subtracting the ambient lighting, the red, green, and blue components of this reference will be called, **Wr**, **Wg**, and **Wb**.

Now we have a way of determining a subject's actual color as a percentage of the difference between the white and black references. This percentage can be expressed as a value between 0 (0%) and 255 (100%), as follows for red, say:

$$\mathbf{Cr = 255 \cdot (Ur - Kr) / (Wr - Kr)}\text{ , where}$$

Ur is the uncorrected (except for ambient) reading, and **Cr** is the corrected reading.

Because the ColorPAL uses an integrated RGB LED to emit the sampling colors, it is also capable of generating a wide range of colors in the visible spectrum by means of its onboard microcontroller, which pulse-width modulates the LED segments to produce 24-bit RGB color.

Comparison to Rev. A ColorPAL

The Rev. A ColorPAL used an AMS (TAOS) TSL13T light-to-voltage detector as its sensor. That detector has been discontinued. The Rev. B ColorPAL uses a Rohm BH1680FVC light-to-current detector instead. Its current output is converted to a voltage by means of a 5.1K load resistor. This substitution has entailed some differences between the programming and performance of the Rev. A and Rev. B ColorPALs, as shown in the table below:

Feature or Capability	Rev. A ColorPAL	Rev. B ColorPAL
Response to **v**ersion command	01	02
Sensor output range to onboard ADC	0V – 5V	0V – 2.9V
Programmable sensitivity setting	Low or high (**l** or **h**)	High only. **l** and **h** commands are accepted but ignored.
Dark-corrected RGB response to white reference (**m** command)	**R**280, **G**215, **B**325, approx. (varies unit-to-unit)	**R**235, **G**295, **B**310, approx. (varies unit-to-unit)
Total RGB response time (**m** command)	~22 msec	~62 msec

Note: The **blue** LED output for named colors (**A** - **Z**) is now half of what it was for the Rev. A ColorPAL. This also applies to the blue value used for RGB sensing. This has the benefit of more truly representing the named colors and also keeping the enhanced blue response of the Rohm sensor from skewing color readings too much before a white balance is applied.

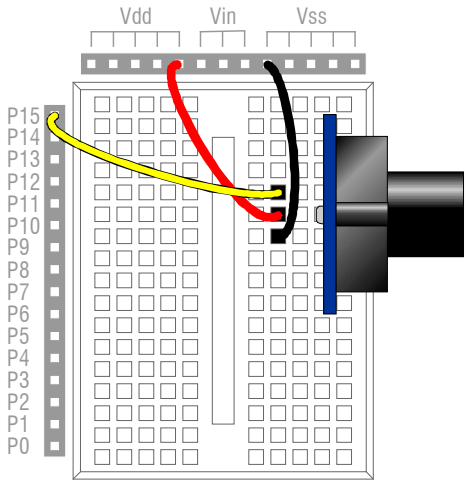
Comparison to TCS3200-DB

The ColorPAL and TCS3200-DB (Parallax #28302) are both capable of detecting colors in the RGB color space. The ColorPAL is designed as an inexpensive sensor for hobby and educational use, whereas the TCS3200-DB also finds use in professional and OEM color detection applications. The following chart will help to illustrate the similarities and differences between the two devices, as an aid to selecting the proper one for a given application. The more stars in a given column, the more applicable or desirable the feature will be.

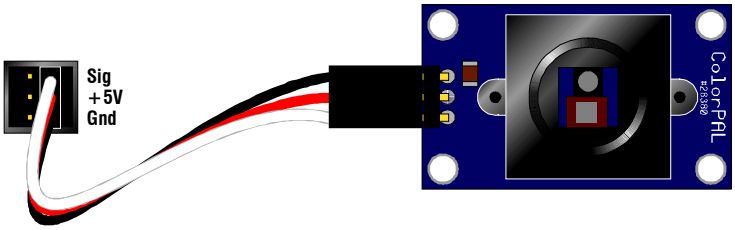
Feature or Capability	ColorPAL	TCS3200-DB
Price	★★★★★	★★★
Color detection accuracy with normal reflective subjects	★★★★	★★★★★
Color detection accuracy with fluorescent (e.g. Day-Glo, Astrobrite) reflective subjects	★★	★★★★★
Color detection on very glossy surfaces	★★★	★★★★★
Color detection of radiant subjects (e.g. LEDs, CRTs)	Not possible	★★★★★
Color sensor output	Serial I/O	Frequency
Pins needed for interface (including +5V and ground)	3	6
Color detection resolution (per RGB component)	Up to 8 bits, using onboard 10-bit ADC	8 bits or more, limited only by sample time
Compatibility with Parallax's PC color-matching software	★★★★★	★★★★★
Accurate color detection requires ambient light correction	Automatic	Yes
Accurate color detection requires white balance	Yes	Yes
Accurate color detection requires black balance	Maybe	No
Color sensing averaged over spot diameter of:	0.47" (12mm)	0.14" (3.5mm)
Color generation	★★★★★	Not possible
On-board programmable sensing and generation sequences	★★★★★	None

Installation

The ColorPAL requires three connections: regulated +5V supply, ground, and open-collector serial data. It can be plugged directly into a BOE, for example, like so:



Using the Parallax "servo" extension cable, it can also be plugged into a servo header:



IMPORTANT: When connecting to a servo header on the BOE or Propeller board, make sure that the header is jumpered for Vdd or +5V and *not* Vin!

Programming

Communication with the ColorPAL takes place using serial I/O, transmitting and receiving at between 2400 and 7200 baud, using a **non-inverted**, **open-drain** protocol. The ColorPAL includes a pullup resistor to Vdd, so you do not need to apply one externally. Because of the open-drain protocol, *the pin used to communicate with the ColorPAL should always be configured as an input, except when being driven low.* Also, when starting up, you should wait for this pin to be pulled high by the ColorPAL before trying to send it any commands. This is particularly important when connecting it to one of the BOEs or Propeller Activity Board's servo headers, since the three-way power switch will start your PBASIC or Spin/C/Blockly program before powering up the ColorPAL.

The pin and baudmode settings for the BS2 on pin 15 at 7200 baud, for example, would be:

```
sio   PIN 15
baud  CON 119 + 32768
```

The ColorPAL has several modes of operation:

- **Direct**: Commands are received and executed immediately.
- **Buffering**: Commands are received and buffered for future execution.
- **Executing**: Commands which were buffered and/or saved in EEPROM are being executed.

There are three different ways to reset a ColorPAL:

- **Powerup:** When first powered up, the ColorPAL begins executing the commands saved in its internal EEPROM at location **00**. In a new, unprogrammed ColorPAL, this program simply exits into Direct mode.
- **Short Break:** A low logic level applied for about 7 milliseconds will reset the ColorPAL and begin execution as above for Powerup.
- **Long Break:** A low logic level applied for 80 milliseconds or more will reset the ColorPAL and enter Direct mode. This is the only way to grab the ColorPAL's attention when it's running a program.

Here is an example of a reset routine that will always cause the ColorPAL to enter Direct mode:

```
Reset:
  LOW sio                   'Pull sio low to eliminate any residual charge.
  INPUT sio                 'Return pin to input.
  DO UNTIL sio : LOOP       'Wait for pin to be pulled high by ColorPAL.
  LOW sio                   'Pull pin low.
  PAUSE 80                  'Keep low for 80ms to enter Direct mode.
  INPUT sio                 'Return pin to input.
  PAUSE 10                  'Pause another 10ms
  RETURN
```

Direct Commands

To begin programming the ColorPAL, you will need to be in Direct mode. In this mode, the ColorPAL will accept any of four different commands. These are:

- **= (Equal sign):** Begin buffering commands.
- **! (Exclamation point):** Begin executing the buffered code.
- **# (Pound sign):** Save the buffered code to an address in EEPROM.
- **+ (Plus sign):** Receive a "network" address.

=	**Begin Buffering Commands**
!	**Execute Buffered Commands**

Any programming (i.e. color-related) commands that the ColorPAL executes must first be buffered before they can be executed. Here is a typical buffer-and-execute sequence that causes the RGB LED to light up red:

```
SEROUT sio, baud, ["= R !"]
```

Note: Any blanks embedded *within a buffered command sequence* (i.e. after the "=" and before a "!" or "#") are optional and can be added to enhance program readability. Adding them affects neither the speed of a program nor the amount of memory it requires. However, ***do not embed blanks in a direct command sequence!*** Doing so will cause the ColorPAL's automatic baud rate detection to fail.

#nn	**Save Buffered Program to EEPROM**

Saving a program to EEPROM requires an address to be supplied in hexadecimal and ranging from **00** to **3F**. *All* numerical arguments passed to the ColorPAL are done so as two-digit hexadecimal constants (**00** to **FF**). To store the "light up red" program to EEPROM location zero (for automatic startup), say, you'd do this:

```
SEROUT sio, baud, ["= R #00"]
```

This, along with the prior example, also illustrates that both the execute (!) and save (#) commands will immediately exit Buffer mode and enter Direct mode.

IMPORTANT: If the program you save starts automatically (i.e. is stored at location **00**) and outputs data, it will do so at an uncalibrated baud rate, somewhere around 4800 baud. The reason is that the ColorPAL calibrates its baud rate from commands sent to it in Direct mode. Lacking any such commands

when autostarting, it has to pick a default value. Because the ColorPAL uses an RC oscillator for its timebase, it's hard to predict with any degree of accuracy what the actual output baud rate will be.

There are two ways around this. One is to restart the stored program manually by entering Direct mode and issuing an Execute command. The ColorPAL will pick up the desired baud rate from this.

The other option is to auto-detect the baud rate at which the ColorPAL is transmitting and use it instead. Here's a PBASIC program that will do that. It will work with any BASIC Stamp, except the BS2px:

```
' {$STAMP BS2}
' {$PBASIC 2.5}

sio     PIN 15

baud    VAR Word
pwidth  VAR Word
i       VAR Byte

PAUSE 100
baud = $ffff

FOR i = 0 TO 255
  PULSIN sio, 0, pwidth
  baud = baud MAX pwidth
NEXT

baud = baud << 1 - 20 | 32768
```

+nn	**Address Networked ColorPALs**

While in Direct mode (and only then) a plus sign (+) followed by two hexadecimal digits will be accepted as a "network" address. Multiple ColorPALs can be paralleled and addressed individually in Direct mode. Each ColorPAL can be assigned an eight-bit unit number, which is saved at EEPROM location **3F** and loaded when the ColorPAL resets. When a non-zero network address is received in Direct mode via the **+** command, it is ANDed with the ColorPAL's unit number. If the result is zero, the ColorPAL will ignore further direct commands until either a **00** address or one which, when ANDed to its unit number, yields a non-zero value. The **00** address can thus be thought of as a broadcast address, to which all ColorPALs will respond. A new, unprogrammed ColorPAL comes with a unit number of **00**, which means it will respond *only* to address **00**.

To change a ColorPAL's unit number, simply create a one-instruction program, with the instruction being one that takes a number (i.e. **p** or **t**), and save it to EEPROM location **3E**. That will place the number in location **3F** and will be effective *after the next and subsequent resets*. Here's an example:

```
SEROUT sio, baud, ["= p03 #3E"]
```

This programs the ColorPAL to have unit number **03** (%00000011), which means it will respond to any address with one or both of its least-significant bits set.

IMPORTANT: Network addressing works *only* in Direct mode. It cannot be used to select which of two or more ColorPALs already running programs gets to output color data, for example. It *can* be used, however, to select which of several units can begin code execution.

When a ColorPAL comes out of reset, any commands sent without prepending an address are assumed to be broadcast and won't be blocked. That way, it's easy simply to ignore the networking feature if you don't want to use it.

Here's some example code that speaks to two ColorPALs, one with address **01**; the other, **02**. Unit 01 will display the colors green, blue, and red in that sequence; unit 02, blue, green, and red. Both display red because they are addressed using the broadcast address **00**. Note that there are *no spaces* in the sub-sequence, **!+02=**. This is because those three commands are all sent in Direct mode, which does not permit spaces.

```
DO
  SEROUT sio, baud, ["+01= G !+02= B !"] : PAUSE 1000
  SEROUT sio, baud, ["+01= B !+02= G !"] : PAUSE 1000
  SEROUT sio, baud, ["+00= R !"] : PAUSE 1000
LOOP
```

Program Commands

While the ColorPAL is in Buffering mode, you can enter the commands that actually do something, like light up the LEDs or take a color sample. Here are the commands that it can accept.

— LED Commands —

WARNING! The light from the RGB LED is very concentrated and can be *very* bright. Just as you would not stare at the sun (and for the same reason), **do not stare directly at the LED.**

rnnnnnn	**Display RGB Color on LED**

A lowercase **r** followed by six hex digits will cause the LED to display the selected color components, with **00** being fully "off", **FF** being fully "on", and numbers in between representing various intermediate intensities. To display the color orange, for example, you can send the following command:

```
SEROUT sio, baud, ["= rC04000 !"]
```

This causes the red LED segment to light up at 75% intensity (**C0**) and the green segment to light at 25% intensity (**40**). The blue segment remains "off" (**00**).

A - Z	**Display Named Color on LED**

The *uppercase* letters represent predefined color names. These are, in order by hue (with letter designators in **boldface**, along with their hexadecimal equivalents):

Color	Hex	Color	Hex
Red	**FF0000**	ro**S**e	**E00010**
Tangerine	**E02000**	**P**ink	**C00020**
Orange	**C04000**	orchi**D**	**A00030**
Flame	**A06000**	**M**agenta	**800040**
Yellow	**808000**	**A**methyst	**600050**
c**H**artreuse	**60A000**	**V**iolet	**400060**
Lime	**40C000**	**I**ndigo	**200070**
Emerald	**20E000**	**B**lue	**000080**
Green	**00FF00**	s**K**y	**002070**
Jade	**00E010**	a**Z**ure	**004060**
ma**N**ganese	**00C020**	t**U**rquoise	**006050**
a**Q**ua	**00A030**	**C**yan	**008040**
X LED Off	**000000**	**W**hite	**55552A**

For example, to display the color "jade", you'd send the following:

```
SEROUT sio, baud, ["= J !"]
```

tnn	Set Inter-color Transition Time

By default, when colors are displayed in sequence, it's done one color after the other without any kind of transition. This behavior can be modified by setting a non-zero transition time between colors. The two-hex-digit argument accompanying this command establishes the ***rate*** at which one color blends into the next during a transition. The overall transition time depends upon how dissimilar the two colors are to begin with. Colors whose RGB components differ by a lot will take more time in their transition than those whose components are closer. Here's an example of red blending into blue:

```
SEROUT sio, baud, ["= R tC0 B !"]
```

— Light Sensing Commands —

s	Sample Output from Light Sensor	Returns:	nnn

The ColorPAL incorporates a 10-bit analog-to-digital converter to read the voltage output from the BH1680FVC light sensor. When the **s**ample command is executed, it takes a reading then outputs it as three hexadecimal digits. Here is how you might take a green color reading, subtracting the ambient light, for example:

```
Red   VAR Word
Grn   VAR Word
Blu   VAR Word
Amb   VAR Word
...
SEROUT sio, baud, ["= X s !"]
SERIN sio, baud, [HEX3 Amb]
SEROUT sio, baud, ["= G s !"]
SERIN sio, baud, [HEX3 Grn]
Grn = Grn - Amb
```

m	Multi-sample RGB colors	Returns:	nnnnnnnnn

This macro function performs an ambient light measurement, then individual measurements with each of the red, green, and blue LEDs, subtracting the ambient reading from each. It then outputs three 3-digit hex numbers representing the ambient-corrected red, green, and blue readings. Here's an example:

```
Red   VAR Word
Grn   VAR Word
Blu   VAR Word
...
SEROUT sio, baud, ["= m !"]
SERIN sio, baud, [HEX3 Red, HEX3 Grn, HEX3 Blu]
```

h	Select High Sensitivity
l	Select Low Sensitivity

These commands were used in the ColorPAL Rev. A to select the sensitivity of the color readings. They are deprecated in the ColorPAL Rev. B, since the unit is preprogrammed to operate only in high-sensitivity mode. The **h** and **l** commands are still accepted, but they have no effect and are not echoed.

— Program Flow and Miscellaneous Commands —

pnn	Pause

The **p**ause command takes a two-hex-digit argument and pauses execution for time approximately equal to **nn** · **5ms**. Here's an example that displays yellow, cyan, and magenta each for one second, then fades to black:

```
SEROUT sio, baud, ["= Y pC8 C pC8 M pC8 t50 X !"]
```

(nn	Begin a Program Loop
)	End a Program Loop

This command pair defines a loop within a program. The "begin" command takes a two-hex-digit argument, which tells how many times the loop is to be executed. If this amount is **00** the loop is executed infinitely. Loops can be nested. Here's an example that alternates between red and green ten times, finishing with a transition to blue:

```
SEROUT sio, baud, ["= (0A R p64 G p64 ) t50 B !"]
```

>nn	Call a Subroutine in EEPROM

Programs saved with the **#nn** direct command can be called as subroutines from a running program. The two-hex-digit argument (**00** – **3F**) specifies the EEPROM address of the program to call. In this example, the program from the prior example is called as a subroutine, then its blue finale slowly fades to black:

```
SEROUT sio, baud, ["= (0A R p64 G p64 ) t50 B #20"]
PAUSE 100
SEROUT sio, baud, ["= >20 tFF X !"]
```

The PAUSE statement is in there because the micro requires some time to store the program. Without it, the following SEROUT would occur too soon, and the ColorPAL would ignore it.

?nn	Set Random Deviation

Sometimes it's nice to mix things up a little for certain color displays (e.g. flickering candles). The set random deviation command (**?**) lets you do just that. The parameter following the question mark is kept as the current random deviate. Once it's been established, any subsequent parameter to the **p**ause, **t**ransition-time, **r**gb-color, and **(**loop**)** commands will be affected by randomization. It works as follows:

1. If the parameter is even, it is taken as a minimum, to which a random number between 0 and (deviate – 1) is added. The sum cannot go above **FE**.

2. If the parameter is odd, it is taken as a maximum, to which a random number between 0 and (deviate – 1) is subtracted. The sum cannot go below **01**.

To turn off randomization, just do a **?00**. In the following example, completely random colors blend together *ad infinitum*:

```
SEROUT sio, baud, ["= t40 (00 ?FF r000000 ?00 p11) !"]
```

There are a few things to point out here: 1) the **t40** is not randomized, since it occurs before the "**?FF**"; 2) the loop constant remains at **00** because it's loaded only once, when the loop begins, and placed on the stack; and 3) the **p11** is not randomized, since it occurs after the **?00**.

Because this example produces so many unsaturated (i.e. whitish) colors, it's pretty boring. Here's an example that will produce fully-saturated colors at the end of each transition, since one component is always held at nearly zero:

```
SEROUT sio, baud, ["= t10 (00 ?FF (07 r808001) (07 r800180) (07 r018080) )
!"]
```

v	**Get Firmware Version Number**	**Returns:**	**nn**

The **v**ersion command allows you to get the version number of the ColorPAL's firmware. Version numbers are output as two hex digits, begin at **01**, and will increment by one with each new version. Version changes are expected to be infrequent. The following program reads and prints the current version number:

```
Version VAR Byte

SEROUT sio, baud, ["= v !"]
SERIN sio, baud, [HEX2 version]
DEBUG "Current version is: ", DEC version, "."
```

"other"	**Echo Miscellaneous Character**	**Returns:**	"other"

Any ASCII character not mentioned here will be buffered and simply echoed when the program runs. This can be handy for synchronizing the BASIC Stamp to a free-running ColorPAL's output. For example, you could program the ColorPAL's EEPROM to sense and output a continuous stream of color data, like so:

```
SEROUT sio, baud, ["= (00 $ m) #00"]
```

Then your program can simply sync on the "$" in the data stream to catch the next available sequence of color data, thus:

```
SERIN sio, baud, [WAIT("$"), HEX3 red, HEX3 grn, HEX3 blu]
```

Be sure to read the important note in the "Save Buffered Program to EEPROM" section regarding baud rates.

Program Limitations

The space available for user programs in the ColorPAL is limited by the onboard micro's RAM and EEPROM space. The space available for buffering programs is 40 bytes. The amount of space required in the buffer for each command is one byte, plus one byte for every two-digit numerical parameter it requires. The space available in the EEPROM is 64 bytes, including the byte at the end that determines the unit number. Spaces are not buffered and do not count toward the memory occupied by a program. Deeply- or incorrectly-nested subroutine calls and loops may cause the stack to encroach upon the end of the buffer space in RAM. Each call adds one byte to the stack; each loop, two bytes. There is no error

checking for excess nesting or unterminated loops. It is the programmer's responsibility to ensure that programs are correctly structured.

Color Matching Program

The Parallax website includes a PC color-matching program that you can use to demo your ColorPAL. The zip file includes the exe file for the PC, a BASIC Stamp program for the ColorPAL, and a couple BASIC Stamp programs for the TCS3200-DB, along with some documentation.

Schematic