Using The Parallax ‘ColorPAL’ Colour Sensor and Colour Generator Module with PICAXE Microcontrollers

(ColorPAL = Parallax Part #28380)

1. Introduction

Previously Revolution Education (Rev Ed) sold the AXE045 PICAXE Colour Sensor module.
This was a reasonably sophisticated module for its price. It used two white LED’s for illumination of the object and had a TAOS TCS230 sensor chip which incorporated an 8 by 8 array of photodiodes. Of these 64 photodiodes, they are broken into four groups where 16 are covered by blue filters, 16 have red filters, 16 have green filters and 16 have no filter. It still required the external controller to select which group of photodiode was to be checked. Unfortunately, the TAOS TCS230 colour sensor is no longer manufactured and Rev Ed ceased selling the AXE045 PICAXE Colour Sensor module.

Rev Ed now promotes the Lego NXT Colour sensor which utilises a single RGB LED and a phototransistor. This module utilises the i2c protocol for communications and since all of the latest PIACXE chips include i2c communications it provides a good replacement.

If one does a Google search for colour sensor projects, there are a number of sites, in particular for hobby related Robots, where others have posted their home made colour sensors. Many of there use three separate LED’s for the red, green and blue colours with a light dependant resistor (LDR) as the receptor for colour intensity. One problem with the use of CdS LDR’s as a light level sensor is the time it takes for the LDR to stabilise when the level of illumination changes. This results is a project which is only suitable for relatively slow operation. One project founf on the internet uses a TSL2561 luminous intensity sensor which could be seen as an improvement, The TSL2561 is currently still available for US$12-50ea.

An alternative to the above options is the Parallax “ColorPAL” module which combines a Colour Sensor and a Colour Generator into one module. This module is available for US$19-99.

The remainder of this tutorial covers the connection and use of the ColorPAL module. Considering the price, for most hobby and educational purposes the ColorPAL module will be sufficiently useful.

2. ColorPAL module

The ColorPAL module incorporates a single RGB LED and a TAOS TSL13T light to voltage converter. There is also an on-module microcontroller to handle receipt of commands sent by serial communications and to read the voltage level from the TAOS TSL13T for each colour and send these back in serial format.

Physically the ColorPAL module is as shown in the image below.

The ColorPAL module includes a “snorkel” with a divider for part of the depth of the snorkel. The RGB LED is mounted in one half and the TSL13T light sensor is mounted in the other half.

The ColorPAL utilises a single data pin for bidirectional communications. The output of data is via an open collector output and the module includes a pull-up resistor so that in many cases no further components are required.

The on-board microcontroller incorporates an interpreter which accepts simple commands, typically each command is one character. The commands can either be stores as small programs to make the ColorPAL perform as an independent device, or by putting the ColorPAL into “direct” mode the instruction sequence can be sent and executed when the “execute” command is sent.

To use the ColorPAL as a colour sensor, it is necessary to send instructions to the ColorPAL module to make it illuminate each of the red, green and blue LED’s in turn and measure the light level under each colour, then finally send the RGB colour data back via the serial data pin.
While the ColorPAL transmits the RGB data as 12-bits per colour, in fact only the lower 10-bits are used.

In addition to use as a colour sensor, there are commands that permit the RGB LED to be used as a colour generator 24-bit resolution and thus capable of 16.7 million colours.

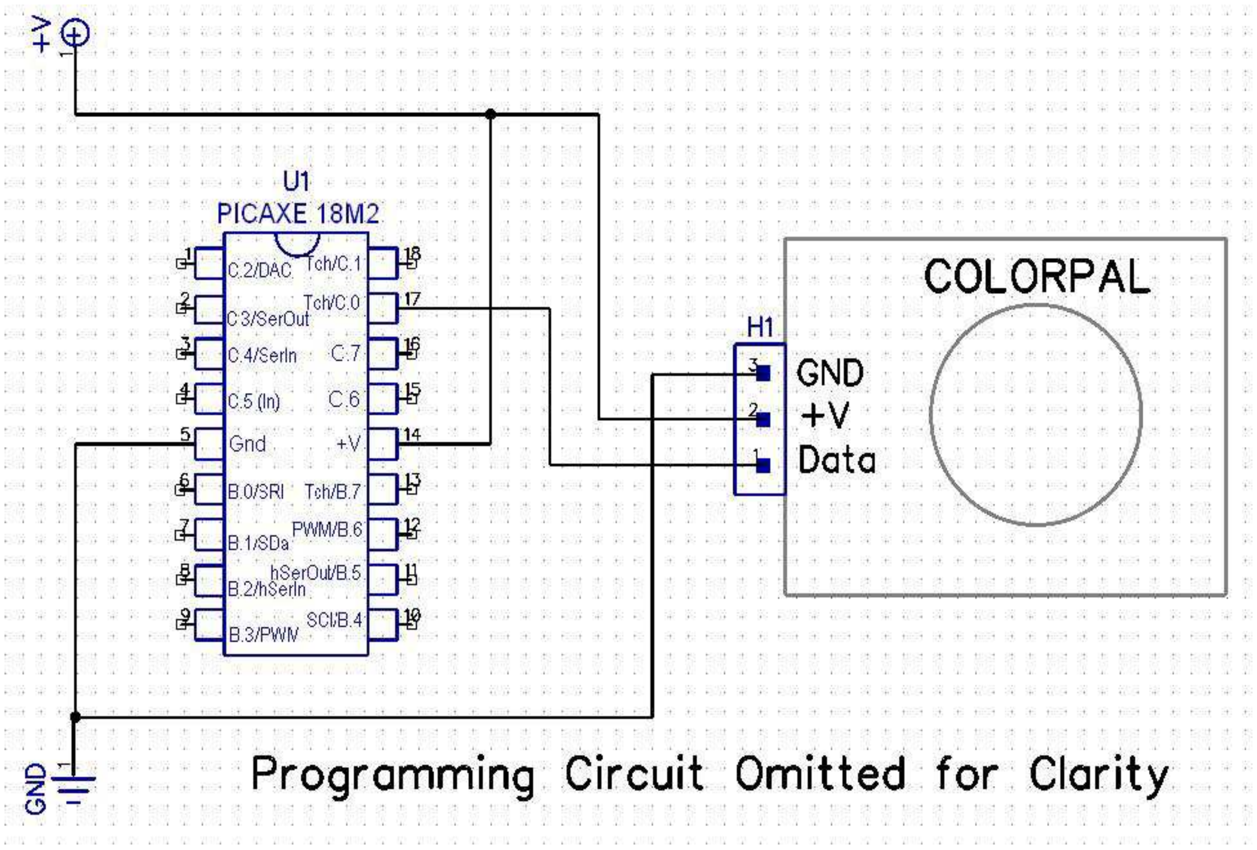
The ColorPAL utilises auto-detect for the serial baudrate and will work with a non inverted serial data signal from 2400 to 7200 baud.

3. Hardware Connection to a PICAXE

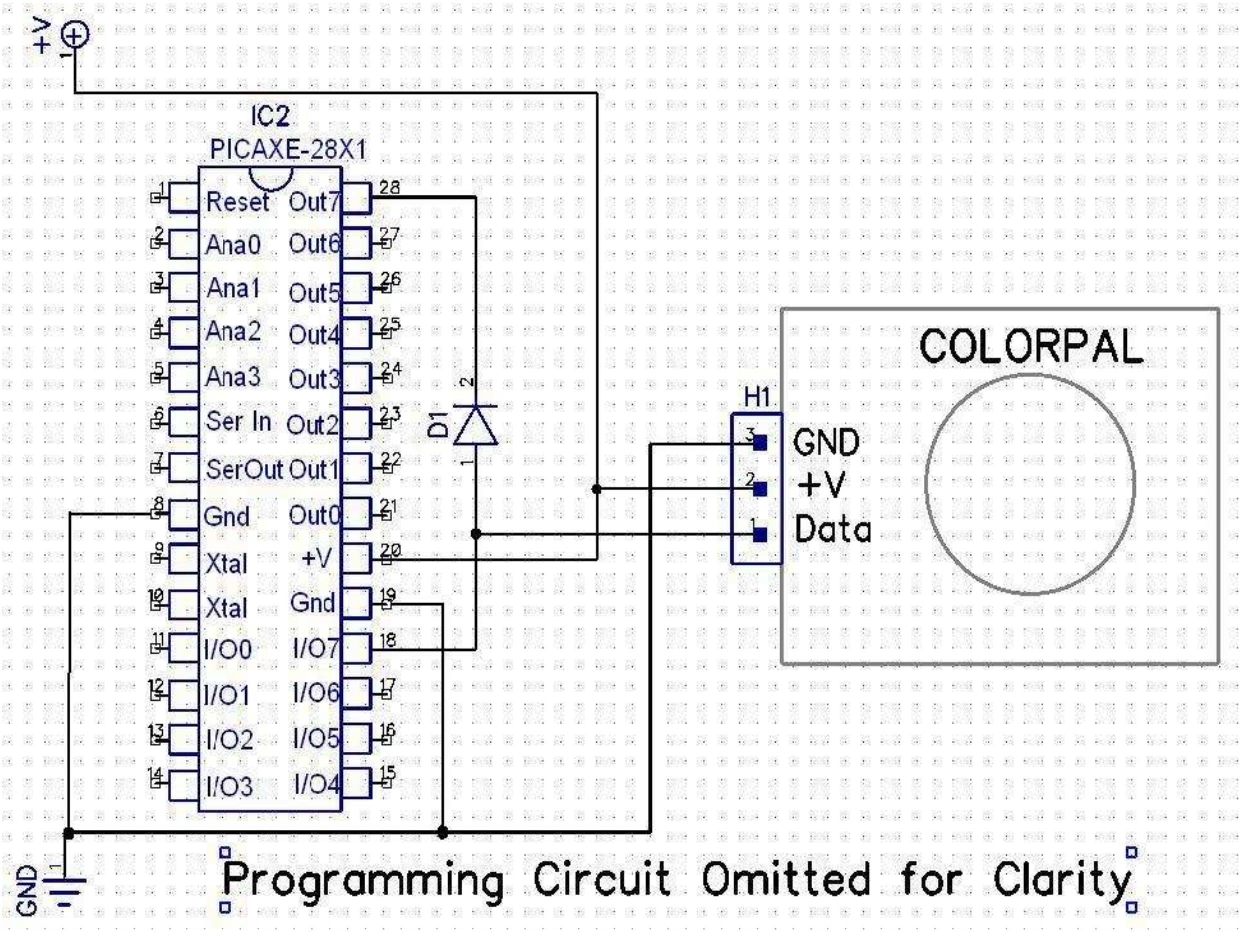
Connection of the ColorPAL to a PICAXE is very simple. For those new M2 and X2 PICAXE chips with bi-directional IO pins, it is a simple case of connecting the ColorPAL to any available bi-directional IO pin.

While the ColorPAL datasheet text mentions the inclusion of a pull-up resistor, strangely this is not shown on the schematic provided on the datasheet. But what is shown on the datasheet is the inclusion of a 330 Ohm resistor in series on the bidirectional serial data communications line. This is good as there is no need for a separate resistor to protect the PICAXE IO in the case of a program error or wiring error which might otherwise result in a short circuit leading to failure of the PIACXE chip or the ColorPAL module.

The following schematic diagram shows the required connections between an M2 or X2 **PICAXE** part and the ColorPAL module



For most of the earlier PICAXE chips from the "M" and "X1" series, most pins are pre-defined as single direction with separate groups for inputs and outputs. To cater for this situation, only the inclusion of a single diode is necessary. The following schematic diagram shows the slight variance needed to allow operation with one of the earlier PICAXE chips.



The diode is used to make the PICAXE output "appear"/perform as an open collector output. When the PICAXE output pin is set to a high state, the diode blocks interaction with the data line, but when the output is low, the data line is pulled to the low state.

4. Principal of Operation

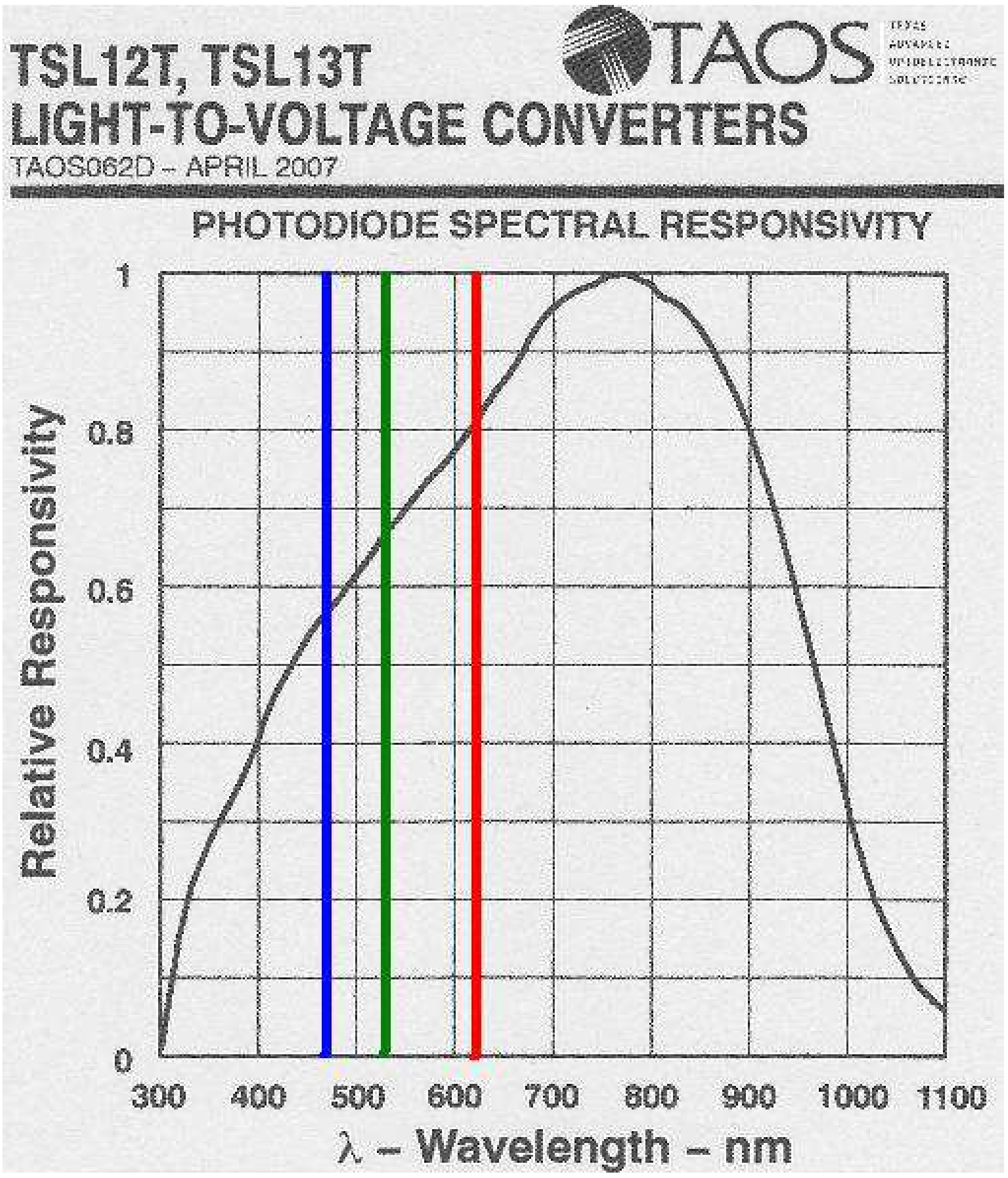
The TAOS TSL13T Light to voltage converter has a spectral response that varies with the wavelength of the light. The peak response is at approximately 780 nm. The responsively drops away at lower and higher wavelengths. The following spectral responsivity chart is taken from the TAOS TSK13T datasheet and has the wavelengths for the RGB LED superimposed to show the relative levels at each of the three LED colours. This chart provides an indication as to the voltage level that might be expected from the TSL13T when each colour within the RGB LED is illuminated.

What Parallax does not provide is the actual illumination output details for the three LEDs forming the RGB LED. As such, the actual output levels received from the TSL13T can differ significantly from what the TAOS chart may imply.

By way of example:

- When placing the ColorPAL module against a piece of matt white photocopy paper the received RGB values from the colorPAL were Red=270, Green=330, and Blue=415.
- When placing the ColorPAL module against a piece of glossy white photo paper the received RGB values from the colorPAL were Red=280, Green=360, and Blue=460.

As can be seen, the blue level is reported with a higher value when the light is reflected from a white surface where the RGB colour levels would typically be considered to be equal, suggesting the Blue LED has a higher output intensity than the red (and green) LED's.



5. Calibration

As can be observed from the TSL13T spectral curve and the data received from the ColorPAL when it is placed against white paper of different types there may be a need to perform a form of calibration if we seek consistent relative output levels.

While the ColorPAL can certainly be used without any calibration, the magnitude of the RGB components is not consistent with expectations in terms of RGB values for a Black or White surface.

To have values available for interpretation which are the same for White and, if desired also for Black, the user should first perform a calibration or “White Balance” measurement to determine scaling factors which will result in red, green and blue having the same value for a white object.

The variable when a black object is scanned for colour is not as great as seen for a white object but if one wishes to achieve to optimal scaling then a similar “Black Balance” should be considered.

The author’s observations are that the ambient light level might be totally ignored, particularly if the lighting is fluorescent as the ColorPAL has (by the datasheet) very poor colour detection under fluorescent lighting. If the user is working under other types of lighting, then an ambient colour balance reading may also be useful.

To use the separate RGB values return from the White Balance and where used the Black Balance values, the following formula can be used separate for the red, Green and Blue components.
To maximise the scale range within the capability of a byte variable, the corrected reading will have a value between 0 (as 0%) and 255 (as 100%).

Cr = 255 * (Ur –Bk) / (Wh – Bk)

Where:

Cr	=	The Corrected reading (scaled form 0 to 255)
Ur	=	The Uncorrected reading being the new reading from the ColorPAL which is to be scaled
Wh	=	The White colour balance reading
Bk	=	The Black colour balance where used, otherwise omit this parameter

6. PICAXE Programming

While this tutorial provides details for using the Parallax ColorPAL module with the **PICAXE** range of microcontroller, it is not intended to be a complete replacement for the Parallax datasheet. Readers are encouraged to also download and read the ColorPAL datasheet.

The ColorPAL module has several modes of operation:

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

On power-up, the ColorPAL begins executing an program previously stored in its internal EEPROM. This tutorial does not cover using the internal EEPROM for a stand alone device and the reader is directed to the ColorPAL datasheet.

To place the ColorPAL into Direct mode to accept commands from the PICAXE microcontroller it is necessary to perform a reset by pulling the data line low for not less than 80 ms.

6.1 Resetting for Direct mode with PICAXE M2 and X2 Parts

The following subroutine can be used to reset the ColorPAL and force it into Direct Mode read to accept commands:

```
; -----[ I/O Definitions ]-----------------------------------------------
; Serial I/O must be open-drain. The ColorPAL includes an internal pull-up.
SYMBOL sio   = C.0              ; Pin used for Serial I/O com with the ColorPAL.
SYMBOL sinp = pinC.0            ; Same pin but definition as an input for tests.

; ----- Subroutine to reset the colorPAL module read for use
; CPReset: Sends a long break to reset ColorPAL and enter direct command mode.

CP_Reset:
   LOW sio                        ; Pull sio low to eliminate any residual charge.
   INPUT sio                  ; Return pin to input.
   DO UNTIL sinp = 1 : 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
```

6.2 Resetting for Direct mode with PICAXE M and X1 Parts

```
; -----[ I/O Definitions ]-----------------------------------------------
; Serial I/O must be open-drain. The ColorPAL includes an internal pull-up.
; Don’t forget to include a diode in the output pin line as per the schematics in section 3.
SYMBOL sinp   = 7              ; Pin used for Serial Input com with the ColorPAL.
SYMBOL sipp   = pin7           ; same input pin but ID when used for pin tests
SYMBOL sout   = 3              ; Pin used for Serial Output com with the ColorPAL

; ----- Subroutine to reset the colorPAL module read for use
; Reset: Sends a long break to reset ColorPAL and enter direct command mode.

CP_Reset:
   LOW sout                    ;Pull sio low to eliminate any residual charge.
   HIGH sout                   ;Return the data line to 'open collector' state
   DO UNTIL sipp = 1 : LOOP    ;Wait for pin to be pulled high by ColorPAL.
   LOW sout                    ;Pull data pin pin low.
   PAUSE 80                    ;Keep low for 80ms to enter Direct mode.
   HIGH sout                   ;Return the data line to 'open collector' state
   PAUSE 10                    ;Pause another 10ms
   RETURN
```

6.3 Baud Rate

As mentioned in Section 2, the ColorPAL uses a serial communications baud rate between 2400 and 7200 baud with non-inverting logic. The ColorPAL auto-detects the baud rate first used after a reset. As 7200 baud is non-standard for the **PICAXE** chips, either 2400 or 4800 baud can be used without the need to try and determine an alternate baud rate parameter for 7200 baud. Within the examples provided in this tutorial, 4800 baud as been used.

The ColorPAL uses an open collector output and the data line has a pull-up resistor.
Therefore, with an M2 or X2 part, the **PICAXE** program will normally keep the data pin as an input unless sending commands to the ColorPAL. In this manner, the usual requirement to make the serial communications pin high when True (non-inverting) mode is used.

For the M and X1 **PICAXE** parts, a diode is used on the output pin since it cannot be changed to an input when not sending serial data (the 08M is an exception). The diode makes the output appear to be an open collector output when the output pin is put into a high state.

6.4 Direct Mode Commands

There are a number of Direct mode commands.

- = (equal sign) ➔ Begin storing commands in the programming buffer.
- ! (exclamation) ➔ Begin execution of the commands stored in the programming buffer
- # (hash symbol) ➔ Save buffer contents to an address in EEPROM
- + (plus sign) ➔ Receive a ‘network’ address

The first two commands mentioned above are those which will typically be used for most simple applications.

Those commands for setting up a ‘Network’ address and saving the instructions in the buffer to EEPROM are not covered further in this tutorial. The reader should refer to the Parallax ColorPAL datasheet if wishing to use these features.

6.5 Program Commands

While the ColorPAL is in buffering mode (after receiving an ‘=’ command), there are a range of program commands that can be placed in the buffer ready for use when the execution command is sent.

Data returned from the ColorPAL is in hexadecimal (base16) format with each nybble ASCII encoded.

The following is a list of the available commands:

6.5.1 LED related commands:

“**A**” to “**Z**” (uppercase letters)
These commands each represent one of 26 pre-defined colours to be displayed by the RGB LED. For example, “R” will set the RGB LED to the colour **red**, “G” will set the RGB LED to **green**, and “B” will set the RGB LED to the colour **blue**.
For the full range of pre-defined colours please refer to the ColorPAL datasheet.

“**r**” (lower case r)
The command followed by six hex digits in the format **rrggbb** will cause the RGB LED to display the selected colour components where 00 I off and FF is fully on.

“**t**”lower case t)
By default when colours are displayed in sequence, this is done without a transition. This command when used with a non-zero 2-digit hex value parameter sets the rate at which one colour blends to the next. The time duration or increment is not specified in the ColorPAL datasheet and is overall time is dependant upon the closeness or dissimilarity between to two colours.

6.5.2 Light Sensing Commands:

“**s**” (lowercase s)
Sample the output form the TSL13T light sensor using a 10-bit analog to digital converter (ADC). The returned value is in the form of three ASCII encoded hex digits (0-9 and A-F).

Notes:
1. while the returned value is in the form of 3 hex digits which can in theory represent $000 to $FFF (0 to 4095), the ADC is only 10-bit conversion so that the actual received value will be in the range $000 to $3FF (0 to 1023).
2. The commands should only be used with the three ‘pure’ colours R, G and B. This is because the LED output for colour shades uses PWM and the TSL13T sensor is fast enough and does not average the ‘seen’ colour in the PWM period. With pure colours the LED is constantly on.

“**m**” (lowercase m)
This is a macro function which performs an ambient light measurement then the individual measurements with each of the red, green and blue LEDs in turn subtracting the ambient reading from each and outputting the RGB results as three 3-digit ASCII encoded hex numbers in the format **rrrgggbbb**.

“**h**” (lowercase h)
This command the high sensitivity mode where the output is referenced to 1.1 Volts instead of 5 Volts. As it multiplies the sensitivity by more than 4.5, it should only be used for colour sensing in very low light conditions.

“**l**” (lowercase L)
This command selects the low/normal sensitivity mode with a 5 Volt reference.

6.5.3 Miscellaneous commands:

"**p**" (lowercase p)
The pause command which is followed by two hex digits ("nn"). The duration of the pasue is approximately equal to nn * 5 ms. For example if nn = "C8" ($C8 = 200) then the pause will be for 1 second.

"**(**" (left bracket)
Command to commence a loop. The bracket is followed by a two digit hex value which indicates how many times to execute the loop. If the value is "00" then the loop executes forever.

"**)**" (right bracket)
End of a program loop

"**>**" (greater than symbol)
Calls a program in EEPROM which has previously been saved with the # command.
Refer to the ColorPAL datasheet for further details.

"**?**" (question mark)
This command is followed by a two hex-digit value ("nn") and is used to set a random deviation to buffered program. Refer to the ColorPAL datasheet for further details.

"**v**" (lower case v)
This command allows you to fetch the ColorPAL firmware **v**ersion number which is output as two ASCII encoded hexadecimal (hex) numbers beginning with "**01"**.

"Other" characters
Any other ASCII character not listed above will be placed in the buffer and simply echoed when the program is executed.
For example, using a "**$**" character can act as a qualifier for reading RGB data in from the ColorPAL using the **PICAXE** SERIN command. This ensures that the SERIN command waits until a "$" character is received before accepting data as the RGB values thereby keeping the PICAXE serial data receival synchronised with the ColorPAL output.

7. Example PICAXE Programs

The following example programs are provided as basic guidance to using the ColorPAL.
The reader will need to ensure the IO pin assignments match those used in the readers project.

7.1 Simple program for M2 or X2 PICAXE

This program resets the Parallax ColorPAL and puts it into multi-sample colour sensing mode, then in a continuous loop fetches the serial RGB data and finally displays the received RGB colour values on the Programming Editor Terminal Screen.

```
; ============================================================================
;
;   File...... ColorPAL_sense
;   Purpose... ColorPAL driver for TSL13T based ColorPAL to test colour sensing.
;   Author.... WestAust55
;   Started... 22 Mar 2012
;   Updated... n/a
;
; ============================================================================
; -----[ Program Description ]--------------------------------------------

; This program sets up the Parallax ColorPAL colour sensor in multi-sample mode
; then in a continuous loop fetches the serial RGB data and finally displays
; the received RGB colour values on the Programming Editor Terminal Screen.

#PICAXE 18M2                    ; change to PICAXE chip being sued
#No_Data

; -----[ I/O Definitions ]-----------------------------------------------
; Serial I/O must be open-drain. The ColorPAL includes an internal pull-up.
SYMBOL sio   = C.0              ; Pin used for Serial I/O com with the ColorPAL.
SYMBOL sinp = pinC.0            ; Same pin but definition as an input for tests.

; -----[ Constants ]-----------------------------------------------------


; -----[ Variables ]-----------------------------------------------------

SYMBOL red  = w2    ; Reconstructed RGB values received from ColorPAL.
SYMBOL grn  = w3
SYMBOL blu  = w4

SYMBOL rd1  = b10  ; Each of the raw three colour bytes for RGB as received
SYMBOL rd2  = b11
SYMBOL rd3  = b12
SYMBOL gn1 = b13
SYMBOL gn2 = b14
SYMBOL gn3 = b15
SYMBOL bl1  = b16
SYMBOL bl2  = b17
SYMBOL bl3  = b18

SYMBOL temp1 = b19 ; Temp values used to pass the raw data to subroutine
SYMBOL temp2 = b20 ; for construction of each could as a single word variable
SYMBOL temp3 = b21
SYMBOL tword = w11 ; w11 = b23:b22

; -----[ Initialization ]------------------------------------------------
PAUSE 1000            ; give the PE terminal window time to set up
SETFREQ m8            ; set the PICAXE to 8 MHz Clock speed
GOSUB CP_Reset        ; reset the ColorPAL and enter direct command mode.
GOSUB GetVersn        ; fetch the ColorPAL version number. Format is 2-digit hex (eg 00, 01, 02 ...)

; -----[ Main Program Code ]---------------------------------------------

SEROUT sio, T4800_8, ("= (00 $ m) !") ; instruct the colorPAL to use multi-sample RGB mode forever
                    ; the '$' is sent verbatim before each multimode sample data so we can use that
                    ; in the SERIN command as a qualifier to get serial data stream timing right
DO
  SERIN sio, T4800_8, ("$"), rd1, rd2, rd3, gn1, gn2, gn3, bl1, bl2, bl3  ; Receive the raw RGB data back
  temp1 = rd1 : temp2 = rd2 : temp3 = rd3 : GOSUB HEXtoDEC : red = tword  ; convert raw red data to one word
  temp1 = gn1 : temp2 = gn2 : temp3 = gn3 : GOSUB HEXtoDEC : grn = tword  ; convert raw green data to one word
  temp1 = bl1 : temp2 = bl2 : temp3 = bl3 : GOSUB HEXtoDEC : blu = tword  ; convert raw blue data to one word
  SERTXD (#rd1, " ", #rd2, " ", #rd3, " ==> Red = ", #red, cr,lf) ; Output Red   data to the Terminal window
  SERTXD (#gn1, " ", #gn2, " ", #gn3, " ==> Grn = ", #grn, cr,lf)    ; Output Green data to the Terminal window
  SERTXD (#bl1, " ", #bl2, " ", #bl3, " ==> Blu = ", #blu, cr,lf, cr,lf) ; Output Blue  data to the Terminal window
  PAUSE 2000
LOOP
```

```
; -----[ Subroutines ]---------------------------------------------------

; ----- Subroutine to reset the colorPAL module read for use
; Reset: Sends a long break to reset ColorPAL and enter direct command mode.

CP_Reset:
  LOW sio                          ;Pull sio low to eliminate any residual charge.
  INPUT sio                         ;Return pin to input.
  DO UNTIL sinp = 1 : 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

; ---------------------------------------------------------------------------
; ----- Subroutine to fetch the version number from the ColorPAL module
GetVersn:
  SEROUT sio, T4800_8, ("= v !")
  SERIN  sio, T4800_8, temp1, temp2
  SERTXD ("ColorPAL Version = ", temp1, temp2, cr, lf, cr, lf)
  pause 1000
  RETURN
; ---------------------------------------------------------------------------
; ----- Subroutine to combine three ASCII encoded hexadecimal digits into a single value
HEXtoDEC:
If temp1 > "9" Then : temp1 = temp1 + 9 : End If
If temp2 > "9" Then : temp2 = temp2 + 9 : End If
If temp3 > "9" Then : temp3 = temp3 + 9 : End If
temp1 = temp1 & $0F
temp2 = temp2 & $0F
temp3 = temp3 & $0F
tword = temp1 * $10 + temp2 * $10 + temp3
RETURN
; ---------------------------------------------------------------------------
```

7.2 Program for M2 or X2 PICAXE With White Colour Balance

```
; ============================================================================
;
;   File...... ColorPAL_sense and Scale
;   Purpose... ColorPAL driver for TSL13T ColorPAL to test colour sensing.
;   Author.... WestAust55
;   Started... 21 Mar 2012
;   Updated... n/a
;
; ============================================================================

; -----[ Program Description ]---------------------------------------------

; This program sets up the Parallax ColorPAL color sensor then does a white balance reading
; before going into a continuous loop where at 2 second intervals, the program
; activates a single multi-colour scan, fetches the serial RGB data and finally displays
; on the Programming Editor Terminal Screen:
; (a) the received raw RGB colour values data, and
; (b) the rescaled RGB data with each colour on a scale of 0 to 255 using the white balance as 255.


#PICAXE 18M2
#No_Data


; -----[ I/O Definitions ]-----------------------------------------------
; Serial I/O must be open-drain. The ColorPAL includes an internal pull-up.
SYMBOL sio   = C.0              ; Pin used for Serial I/O com with the ColorPAL.
SYMBOL sinp = pinC.0            ; Same pin but definition as an input for tests.

; -----[ Constants ]-----------------------------------------------------


; -----[ Variables ]-----------------------------------------------------
; byte variables b0 and b1 are free

SYMBOL red  = w1                ; Reconstructed RGB values received from ColorPAL.
SYMBOL grn  = w2
SYMBOL blu  = w3

SYMBOL rd1  = b8                ; Each of the raw three colour bytes for RGB as received
SYMBOL rd2  = b9
SYMBOL rd3  = b10
SYMBOL gn1  = b11
SYMBOL gn2  = b12
SYMBOL gn3  = b13
SYMBOL bl1   = b14
SYMBOL bl2   = b15
SYMBOL bl3   = b16

SYMBOL temp1 = b17              ; Temp values used to pass the raw data to subroutine
SYMBOL temp2 = b18              ; for construction of each could as a single word variable
SYMBOL temp3 = b19
SYMBOL tword = w10              ; w10 = b21:b20

SYMBOL whred = w11              ; White reference components as R G B
SYMBOL whgrn = w12
SYMBOL whblu = w13
SYMBOL bkred = 25            ; Black reference components as RGB - these read once and saved.
SYMBOL bkgrn = 30
SYMBOL Bkblu = 37

; -----[ Initialization ]-----------------------------------------------
Init:
  PAUSE 1000           ; give the PE terminal window time to set up
  SETFREQ m8           ; set the PICAXE to 8 MHz Clock speed
  GOSUB CP_Reset  ; reset the ColorPAL and enter direct command mode.
  GOSUB GetVersn    ; fetch the ColorPAL version number. Format is 2-digit hex (eg 00, 01, 02 ...)
  GOSUB Calibrate

; -----[ Main Program Code ]--------------------------------------------------

Main:
  DO
   SEROUT sio, T4800_8, ("= (01 $ m) !") ; instruct the colorPAL to use multi-sample RGB mode forever
                              ; the '$' is sent verbatim before each multimode sample data so we can use that
                              ; in the SERIN command as a qualifier to get serial data stream timing right
   GOSUB FetchRGB

   SERTXD (#rd1, " ", #rd2, " ", #rd3, " ==> Red = ", #red, cr,lf) ; Output Red   data to the Terminal window
   SERTXD (#gn1, " ", #gn2, " ", #gn3, " ==> Grn = ", #grn, cr,lf) ; Output Green data to the Terminal window
   SERTXD (#bl1, " ", #bl2, " ", #bl3, " ==> Blu = ", #blu, cr,lf, cr,lf) ; Output Blue  data to the Terminal window

   GOSUB RGBCorrect
   SERTXD ("The rescaled RGB data is: Red = ", #red," Grn = ", #grn," Blu = ", #blu,cr,lf,cr,lf)

   PAUSE 2000
  LOOP

; -----[ Subroutines ]----------------------------------------------------

; ----- Subroutine to reset the ColorPAL module read for use
; Reset: Sends a long break to reset ColorPAL and enter direct command mode.

CP_Reset:
  LOW sio                        ;Pull sio low to eliminate any residual charge.
  INPUT sio                      ;Return pin to input.
  DO UNTIL sinp = 1 : 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

; ---------------------------------------------------------------------
; ----- Subroutine to fetch the version number from the ColorPAL module
GetVersn:
  SEROUT sio, T4800_8, ("= v !")
  SERIN sio, T4800_8, temp1, temp2
  SERTXD ("ColorPAL Version = ", temp1, temp2, cr, lf, cr, lf)
  pause 1000
  RETURN
```

```
; -------------------------------------------------------------------------
; ----- Subroutine to combine three ASCII encoded hexadecimal digits into a single value
HEXtoDEC:
  IF temp1 > "9" THEN : temp1 = temp1 + 9 : ENDIF
  IF temp2 > "9" THEN : temp2 = temp2 + 9 : ENDIF
  IF temp3 > "9" THEN : temp3 = temp3 + 9 : ENDIF
  temp1 = temp1 & $0F
  temp2 = temp2 & $0F
  temp3 = temp3 & $0F
  tword = temp1 * $10 + temp2 * $10 + temp3
  RETURN
; -------------------------------------------------------------------------
; subroutine to wait for '$' character and then read in the 9 bytes of RGB colour data
FetchRGB:
  SERIN sio, T4800_8, ("$"), rd1, rd2, rd3, gn1, gn2, gn3, bl1, bl2, bl3  ; Receive the raw RGB data back
  temp1 = rd1 : temp2 = rd2 : temp3 = rd3 : GOSUB HEXtoDEC : red = tword  ; covert raw red data to one word
  temp1 = gn1 : temp2 = gn2 : temp3 = gn3 : GOSUB HEXtoDEC : grn = tword  ; covert raw green data to one word
  temp1 = bl1 : temp2 = bl2 : temp3 = bl3 : GOSUB HEXtoDEC : blu = tword  ; covert raw blue data to one word
  RETURN
; -------------------------------------------------------------------------
; Subroutine to perform a White balance calibration
Calibrate:
  SERTXD ("Place a sheet of non-glossy white paper in front of the ColorPAL",cr,lf)
  SERTXD ("the ColorPAL and then enter a 'W'",cr,lf)
  SERRXD b0
  IF b0 <> "W" THEN Calibrate ; loop and wait until command received to read while reference data
  SEROUT sio, T4800_8, ("= (01 $ m) !")       ; perform a single multi-colour scan
  GOSUB FetchRGB
  whred = red                          ; save the White reference RGB components
  whgrn = grn
  whblu = blu
  SERTXD (#red, " ", #grn, " ", #blu, cr,lf,cr,lf)
  RETURN
; -------------------------------------------------------------------------
; Subroutine to rescale Red, Green and Blue colour data relative to white for 0 to 255 for each colour
RGBCorrect:
  IF red > bkred THEN           ; prevent roll over if reading is less than the black balance reading
    tword =whred - bkred / 2 :   red = red - bkred * 128 / tword MAX 255 ; calculate rescaled value
  ELSE
          red = 0
  ENDIF
  IF grn > bkgrn THEN
    tword =whgrn - bkgrn / 2 :   grn = grn - bkgrn * 128 / tword MAX 255
  ELSE
    grn = 0
  ENDIF
  IF blu > bkblu THEN
    tword =whblu - bkblu / 2 :   blu = blu - bkblu * 128 / tword MAX 255
  ELSE
    blu = 0
  ENDIF
  RETURN
  ; -------------------------------------------------------------------------
```

7.3 Simple program for M or X1 PICAXE

This program resets the Parallax ColorPAL and puts it into multi-sample colour sensing mode, then in a continuous loop fetches the serial RGB data and finally displays the received RGB colour values on the Programming Editor Terminal Screen.

```
; ===========================================================================
;
;   File...... ColorPAL_sense
;   Purpose... ColorPAL driver for TSL13T ColorPAL to test colour sensing.
;   Author.... WestAust55
;   Started... 22 Mar 2012
;   Updated... n/a
;
; ===========================================================================

; -----[ Program Description ]---------------------------------------------

; This program sets up the Parallax ColorPAL colour sensor in multi-sample mode
; then in a continuous loop fetches the serial RGB data and finally displays
; the received RGB colour values on the Programming Editor Terminal Screen.

#PICAXE 40X1
#No_Data


; -----[ I/O Definitions ]-----------------------------------------------
; Serial I/O must be open-drain. The ColorPAL includes an internal pullup.
SYMBOL sinp = 7              ; Pin used for Serial Input com with the ColorPAL.
SYMBOL sipp = pin7  ; same input pin but ID when used for pin tests
SYMBOL sout = 3              ; Pin used for Serial Output com with the ColorPAL

; -----[ Constants ]-----------------------------------------------------



; -----[ Variables ]-----------------------------------------------------

SYMBOL red  = w2             ; Reconstructed RGB values received from ColorPAL.
SYMBOL grn  = w3
SYMBOL blu  = w4

SYMBOL rd1  = b10            ; Each of the raw three colour bytes for RGB as received
SYMBOL rd2  = b11
SYMBOL rd3  = b12
SYMBOL gn1  = b13
SYMBOL gn2  = b14
SYMBOL gn3  = b15
SYMBOL bl1  = b16
SYMBOL bl2  = b17
SYMBOL bl3  = b18

SYMBOL temp1 = b19           ; Temp values used to pass the raw data to subroutine
SYMBOL temp2 = b20           ; for construction of each could as a single word variable
SYMBOL temp3 = b21
SYMBOL tword = w11           ; w11 = b23:b22


; -----[ Initialization ]------------------------------------------------
PAUSE 1000                   ; give the PE terminal window time to set up
```

```
SETFREQ m8          ; set the PICAXE to 8 MHz Clock speed
GOSUB CP_Reset      ; reset the ColorPAL and enter direct command mode.
GOSUB GetVersn      ; fetch the ColorPAL version number. Format is 2-digit hex (eg 00, 01, 02 ...)


; -----[ Main Program Code ]-------------------------------------------------

SEROUT sout, T4800_8, ("= (00 $ m) !") ; instruct the colourPAL to use multi-sample RGB mode
forever
                    ; the '$' is sent verbatim before each multimode sample data so we can use that
                    ; in the SERIN command as a qualifier to get serial data stream timing right
DO
  SERIN sinp, T4800_8, ("$"), rd1, rd2, rd3, gn1, gn2, gn3, bl1, bl2, bl3  ; Receive the raw RGB data back
  temp1 = rd1 : temp2 = rd2 : temp3 = rd3 : GOSUB HEXtoDEC : red = tword  ; covert raw red data to one word
  temp1 = gn1 : temp2 = gn2 : temp3 = gn3 : GOSUB HEXtoDEC : grn = tword  ; covert raw green data to one word
  temp1 = bl1 : temp2 = bl2 : temp3 = bl3 : GOSUB HEXtoDEC : blu = tword  ; covert raw blue data to one word
  SERTXD (#rd1, " ", #rd2, " ", #rd3, " ==> Red = ", #red, cr,lf) ; Output Red   data to the Terminal window
  SERTXD (#gn1, " ", #gn2, " ", #gn3, " ==> Grn = ", #grn, cr,lf) ; Output Green data to the Terminal window
  SERTXD (#bl1, " ", #bl2, " ", #bl3, " ==> Blu = ", #blu, cr,lf, cr,lf) ; Output Blue  data to the Terminal window
  PAUSE 2000
LOOP

; -----[ Subroutines ]-------------------------------------------------

; ----- Subroutine to reset the colorPAL module read for use
; Reset: Sends a long break to reset ColorPAL and enter direct command mode.

CP_Reset:
  LOW sout                            ;Pull sio low to eliminate any residual charge.
  HIGH sout                           ;Return the data line to 'open collector' state
  DO UNTIL sipp = 1 : LOOP  ;Wait for pin to be pulled high by ColorPAL.
  LOW sout                            ;Pull data pin pin low.
  PAUSE 80    ;Keep low for 80ms to enter Direct mode.
  HIGH sout                           ;Return the data line to 'open collector' state
  PAUSE 10              ;Pause another 10ms
  RETURN

; ------------------------------------------------------------------------
; ----- Subroutine to fetch the version number from the ColorPAL module
GetVersn:
  SEROUT sout, T4800_8, ("= v !")
  SERIN  sinp, T4800_8, temp1, temp2
  SERTXD ("ColorPAL Version = ", temp1, temp2, cr, lf, cr, lf)
  pause 1000
  RETURN
; ------------------------------------------------------------------------
; ----- Subroutine to combine three ASCII encoded hexadecimal digits into a single value
HEXtoDEC:
If temp1 > "9" Then : temp1 = temp1 + 9 : End If
If temp2 > "9" Then : temp2 = temp2 + 9 : End If
If temp3 > "9" Then : temp3 = temp3 + 9 : End If
temp1 = temp1 & $0F
temp2 = temp2 & $0F
temp3 = temp3 & $0F
tword = temp1 * $10 + temp2 * $10 + temp3
RETURN
; ------------------------------------------------------------------------
```