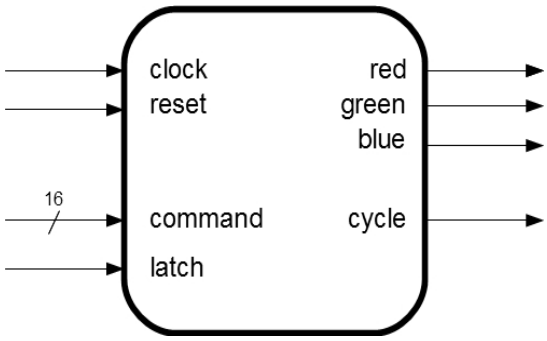
PERDYN90312™

MIBAM RGB LED CONTROLLER

IP Reference Manual

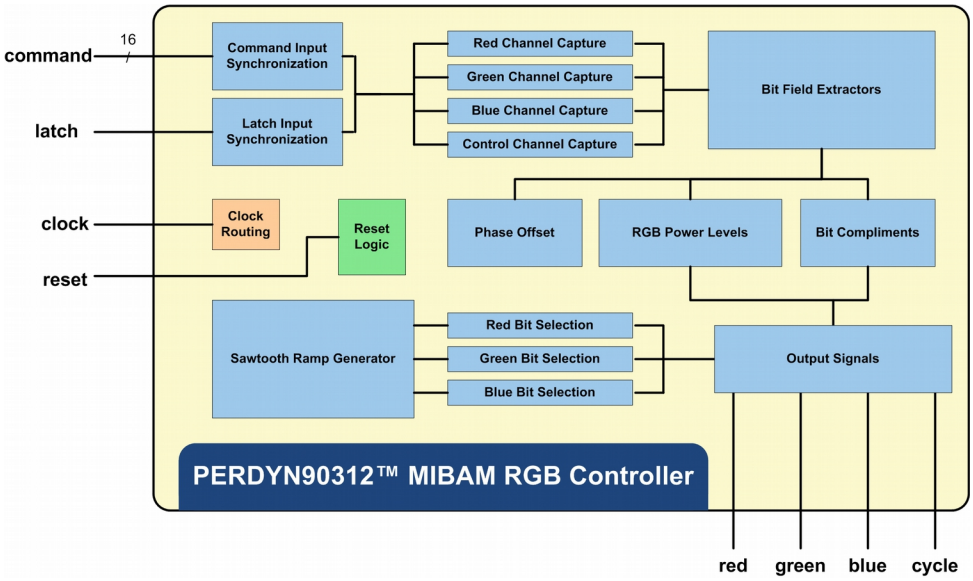
Overview

MIBAM, or mirror-image bit angle modulation, is a method of controlling the average duty cycle of a drive signal using a time period that uniquely corresponds to each bit position of the drive value. MIBAM is a great alternative to traditional PWM drive techniques due to the patent restrictions that have arisen with PWM red/green/blue (RGB) LED drives. The mirror-image functionality solves linearity issues present in previous bit-angle modulations, and the phase control allows power supply load balancing for large LED arrays.



Key Features

- ✔ **Easily controls RGB LED intensity and color using any microcontroller**
- ✔ **Provides 12-bit resolution for each color channel**
- ✔ **Unique MIBAM modulation technique circumvents PWM / RGB LED patent issue**
- ✔ **Linear, flicker-free individual level control of red, green, and blue drive channels**
- ✔ **Phase control allows banks of LED's to draw power at separate times**
- ✔ **Includes parallel interface and SPI (serial peripheral interface) wrapper to connect to any micro-controller or other host**
  Output polarity control for interface to any drive circuitry
- ✔ **Custom interface wrappers available**
- ✔ **Verilog™ RTL synthesizeable in any vendor fabric**



Introduction

RGB (red/green/blue) LED's are everywhere. From billboard signs to architectural lighting to dramatic theatrical presentations, the blending of these additive primary colors creates a color gamut of near infinite variability. Yellow, cyan, magenta, and a multitude of other colors (including white) are created by varying the intensity of each of the three additive primary colors.

Traditionally LED brightness has been controlled using pulse-width modulation (PWM), a technique that uses a fixed frequency with a varying pulse width based on the magnitude of the value to be output. While this is effective from an engineering standpoint, there has been legal precedence set against using PWM to control RGB LED's. (*For additional details review US Patent 7462997 and other relevant patents and court rulings.*)

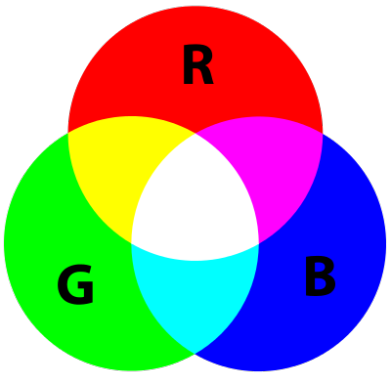


Image 1: *Red, Green, and Blue are combined to create any color in the spectrum. White light is created when all three power levels are equal.*

For this reason, many companies are switching to using Bit-Angle Modulation (BAM), a technique that uses a fixed frequency output with on times representing the weight of each bit in the binary representation of the value to be output. MIBAM, or Mirror-Image Bit Angle Modulation, is an enhanced version of Bit-Angle Modulation that eliminates flicker and provides an effective, trouble-free method of RGB LED control.

The MIBAM drive technique is in the public domain, and is highly suitable (from both an engineering and a legal standpoint) for driving RGB LED's to generate one or more colors. In addition, the PERDYN90312™ MIBAM IP Core allows precise phase adjustment, allowing large banks of LED's to have staggered pulse 'on' times to distribute the power supply load more evenly.

As a stand-alone peripheral, the PERDYN90312 modulates the red, green, and blue color channel intensity to produce the RGB color blend set by the host, requiring no additional control or CPU time from the host.

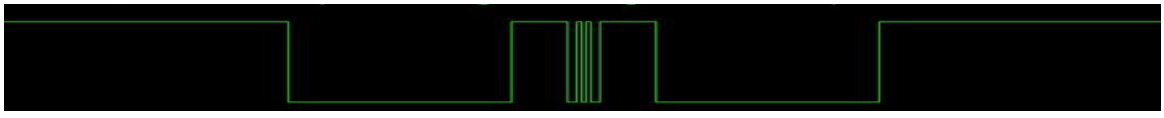


Image 2: *A single MIBAM pulse waveform. Each bit has a specified weight (period of time), and the channel's output is high if that bit is a binary '1'. Notice that the two halves form a mirror image of one another.*

As illustrated in Image 2 above, each of the twelve bits of a primary (red, green, or blue) color's intensity level has a corresponding weight, or period of time for which it controls the output. This value of this weight corresponds to the bit position, for example bit 1 has twice the weight of bit 0, and bit 2 has four times the weight of bit 0. As seen in Table 1 at right, Bit 11 (the most significant bit) has a weight of 2048. Notice that a bit's weight is calculated by the formula:

$$\textbf{weight} = 2^n$$

Weight is the *period* of time the MIBAM signal will output that bit, and *n* is the bit position (ranging from 0 to 11).

Also, notice that the twelve bit sequence is listed twice: first from most-significant bit (MSB) to least-significant bit (LSB), and then from LSB to MSB. This mirror-imaged output results in a flicker-free signal when the output is used to drive a LED color channel. The duration of the twenty-four positions of the MIBAM output are displayed in Image 3 below, where it can be seen that bit 11 (represented by positions 1 and 24) occupies half of the total cycle period, with each bit progressing toward the LSB occupies half of the bit preceding it.

Bit Number	Position	Weight
11	1	2048
10	2	1024
9	3	512
8	4	256
7	5	128
6	6	64
5	7	32
4	8	16
3	9	8
2	10	4
1	11	2
0	12	1
0	13	1
1	14	2
2	15	4
3	16	8
4	17	16
5	18	32
6	19	64
7	20	128
8	21	256
9	22	512
10	23	1024
11	24	2048

Table 1: *The MIBAM output position and relative duration (weight) of all twelve bits.*

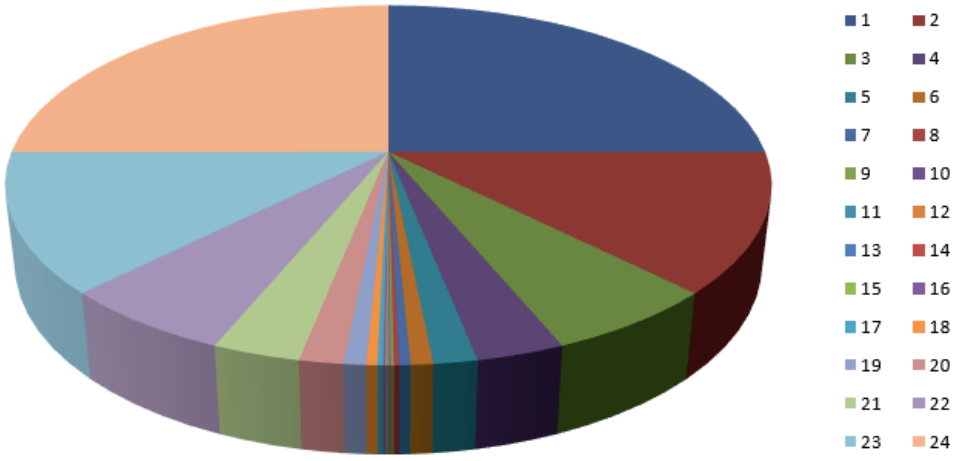


Image 3: *Relative duration of each of the twelve input bits and their mirror image (24 bit positions in total)*

Core Operation

One or more instances of the PERDYN90312 peripheral may be used alongside a host micro-controller or microprocessor, with each one controlling a chain of RGB (red/green/blue) LED's. This relieves the processor of the timing, pin, and processing overheads involved in controlling each channel directly, while also freeing the microcontroller's PWM peripherals for other uses. Each instance of the PERDYN90312 peripheral independently controls the power level of the red, green, and blue color channels associated with the connected driver.

Color channel power levels for the three (RGB) colors are loaded using the 'Command' and 'Latch' inputs. The command input is 16 bits wide, and is captured on the rising edge of the Latch signal. The two uppermost bits of the command input's value (the 'command word') select one of four Registers for writing, as shown in Table 2 below. The first three are the primary color channels, including the Red Intensity Register (binary '00'), the Green Intensity Register (binary '01'), and the Blue Intensity Register (binary '10'). Lastly, when the two uppermost bits are binary '11', the Control Register is written to. For example, if the hexadecimal value 0x0FFF (binary '0000111111111111') is latched into the command input, this would select the Red Intensity Register (bits 15 & 14), set the red Drive Polarity to '0' (bit 13) and set all the Intensity Level bits to '1' (bits 11 through 0). Table 2 below details the function of all bit positions for these four control registers.

Bit Number:	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Red Intensity Register	0	0	Drive Polarity	Reserved	Intensity Level											
Green Intensity Register	0	1	Drive Polarity	Reserved	Intensity Level											
Blue Intensity Register	1	0	Drive Polarity	Reserved	Intensity Level											
Control Register	1	1	RGB Enable	Phase Offset												

Table 2: *Values written to the Command port are stored to one of four registers, selected by bits 15 & 14 of the sixteen-bit command word.*

Each of the three color channels (red, green, and blue) have a twelve bit resolution (bits 11 to 0 in table above) and therefore may be loaded with a value ranging from 0 (0x000 hexadecimal) to 4,095 (0xFFF hexadecimal). This value represents the average power that will be delivered to the LED string connected to a color channel. Setting this value to 0 would turn off that color channel, while setting it to 4095 would select the maximum power level. Values between these two represent a linear power level ramp; for example sending 2,048 would set that channel to a power level of 50% (though not necessarily a 50% brightness level; see your LED datasheet for perceived intensity details).

Setting the 'RGB Enable' bit of the Control Register to '0' deactivates all three color output channels regardless of their intensity setting.

The 'Polarity' bit of each color channel's Intensity Register selects whether the output is active high (when set to 1) or active low (when set to 0). The requirements of your LED drive circuit will dictate whether this should be set high or low.

Additionally, the 'Phase Offset' field of the Control Register specifies a time delay of the outputs. This adjustment is useful when multiple instances of the PERDYN90312 peripheral are used so that the LED strings load the power supply in a balanced manner.

Interfaces

The PERDYN90312 IP has eight interface ports, as described below. Two ports (command, latch) allow the host to control and configure the peripheral's parameters. Three ports (red, green, blue) are pulsed outputs that control your external LED drive circuitry. The 'cycle' output indicates the phase of the MIBAM output. The 'clock' input controls the internal timing of the peripheral, and the 'reset' input initializes the peripheral to a known state.

Interface	Width	Direction	Polarity	Description
command	16	Input	Active-High	Control Word Data
latch	1	Input	Active-High	Control Word Capture Signal
red	1	Output	*Selectable	Red Color Channel MIBAM Signal
green	1	Output	*Selectable	Green Color Channel MIBAM Signal
blue	1	Output	*Selectable	Blue Color Channel MIBAM Signal
cycle	1	Output	Active-High	Output Signal Phase Markers
clock	1	Input	Active-High	Peripheral Clock
reset	1	Input	Active-Low	Peripheral Reset

Table 3: *PERDYN90312 MIBAM Peripheral Interface Ports*

'command' is a 16-bit wide data input port through which the PERDYN90312 peripheral is controlled. Commands placed onto the 'command' input port set the color channel intensity, phase offset, and other features of the peripheral (see Table 2). Data placed onto the 'command' port is not captured until the latch port is activated. Data for the 'command' port may come from a host microcontroller as parallel I/O, from a state machine within the programmable logic, a SPI (serial peripheral interface) using the PERDYN90312's SPI Wrapper, or a custom interface wrapper designed to receive data from your chosen bus or interface type.

'latch' is a one-bit, active high input that causes the PERDYN90312 peripheral to capture the command word currently on the 'command' input pins. This input is synchronized to the peripheral clock, and must be active for the period of at least three peripheral clock cycles to be considered valid.

'red' is a one-bit output that sends the MIBAM signal for the red color channel. The average power level contained in this signal is determined by the value currently stored in the Red Intensity Register. This output drives the input of your high-current driver for the red LED's.

'green' is a one-bit output that sends the MIBAM signal for the green color channel. The average power level contained in this signal is determined by the value currently stored in the Green Intensity Register. This output drives the input of your high-current driver for the green LED's.

'blue' is a one-bit output that sends the MIBAM signal for the blue color channel. The average power level contained in this signal is determined by the value currently stored in the Blue Intensity Register. This output drives the input of your high-current driver for the blue LED's.

'cycle' is a one-bit output that shows the beginning and mid-point of the MIBAM periods. The 'cycle' output signal is one clock period in duration, and is useful for comparing phase relationships in systems using multiple PERDYN90312 instances with varying phase offsets. This output is typically used for system development and is not connected to the LED drive circuitry.

'clock is a one-bit input that synchronizes all peripheral operation. Changes to the input clock frequency will have a direct effect on the MIBAM operational frequency. The PERDYN90312 core is designed to receive a 1MHz input clock signal. If an extra clock PLL circuit is available in the programmable logic device implementing the MIBAM peripheral, it is recommended to use it to generate this 1MHz clock (in phase with the primary system clock). If a PLL is not available, a programmable clock divider block (included with PERDYN90312 IP) may be used.

'reset' is a one-bit input that halts peripheral operation and returns the data registers to their reset condition as shown in the table below. Note that this input is active-low, and must be held high during ordinary operation.

Item	Reset State
Red Intensity Register	0x0000
Green Intensity Register	0x0000
Blue Intensity Register	0x0000
Control Register	0x0000
Red Output Pin	Low
Green Output Pin	Low
Blue Output Pin	Low
Cycle Output Pin	Low

Table 4: *Upon reset, the registers and output pins are set to their reset states.*

A Simple Configuration Example

Consider the following typical configuration of the PERDYN90312 IP core:

Control Register: RGB drive enabled, zero phase offset

Red Register: Intensity: 0x008 (decimal 8), polarity active-high

Green Register: Intensity: 0x080 (decimal 128), polarity active-high

Blue Register: Intensity: 0x800 (decimal 2048), polarity active-high

Following the bit selection shown in Table 2, the Red, Green, Blue, and Control Registers should be loaded with these values:

Control Register:	0xE000	*(binary 1110 0000 0000 0000)*
Red Register:	0x2008	*(binary 0010 0000 0000 1000)*
Green Register:	0x6080	*(binary 0110 0000 1000 0000)*
Blue Register:	0xA800	*(binary 1010 1000 0000 0000)*

Image 4 below illustrates the loading of the four values above into their various registers via the testbench command input, *command_tb*. Note that the testbench's latch signal, *latch_tb*, strobes high once for each data register latch. The entire testbench code may be found in the Testbench section further in this document.

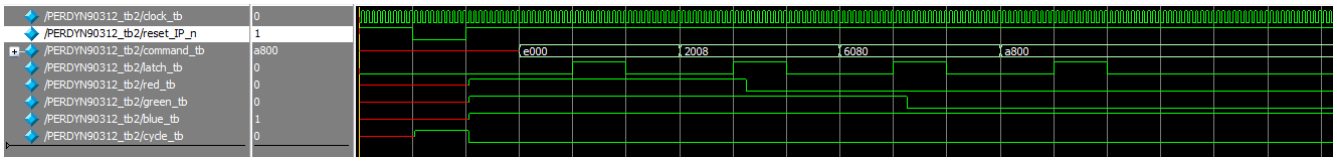


Image 4: Control and RGB Register Setup Example

Following the initialization of these registers, the PERDYN90312 IP core begins driving the red, green, blue output signals to the appropriate on times, as shown in Image 5, below. Observe that the red output (*red_tb*) is on a very small percent of the time (8 of 4096 counts), while the blue channel (*blue_tb*) is on half of the time (2048 of 4096 counts) just as we specified in the color register intensity settings above. Notice that the cycle output (*cycle_tb*) pulses twice per color channel output cycle, once at the start and again at the middle of the cycle.

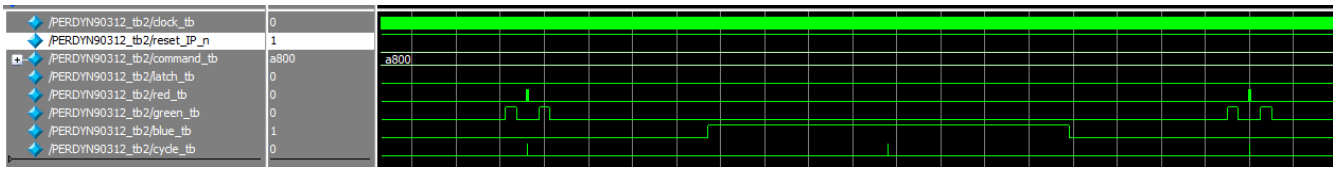


Image 5: Generated Output Signals

Microcontroller Peripheral

While some applications utilize the PERDYN90312 IP in a FPGA-only application, others utilize it as a microcontroller peripheral. In this configuration, a host microcontroller controls the PERDYN90312 IP that is synthesized inside a low-cost CPLD or FPGA IC. Because the native PERDYN90312 IP interface requires 17 pins (16 for Command input, 1 for Latch input) it is often desirable to utilize an interface wrapper to reduce the IO pin requirements on the host microcontroller. Nearly all micro-controllers made today contain a SPI (Serial Peripheral Interface) port, making it an ideal option for controlling one or more PERDYN90312 IP cores from the host microcontroller, as shown in Image 6 below.

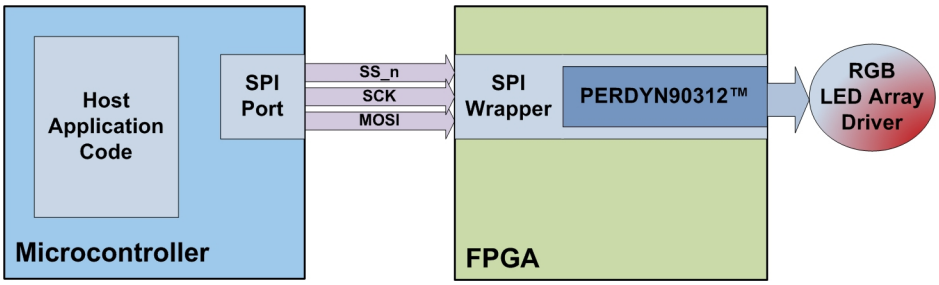


Image 6: SPI Interface to PERDYN90312 IP Core

The SPI interface utilizes three pins for communication:

- **MOSI** *(Master Out, Slave In)*
- **SCK** *(serial data shift clock)*
- **SS_n** *(active-low slave select signal)*

Writing to the PERDYN90312 IP using the SPI Wrapper requires that the host bring the SS_n pin low, followed by clocking the sixteen data bits out via the MOSI and SCK pins, and finally returning the SS_n pin high. SPI peripheral configuration should be for SPI Mode 3 (CPOL = 1, CPHA = 1).

The SPI interface wrapper is included with the PERDYN90312 IP. Other interface wrappers, such as I2C, APB3, Wishbone, etc. may be constructed by the user. Alternately, Peripheral Dynamics can create a custom interface to suit your needs.

Testbench

```
module PERDYN90312_tb;                          // Testbench that demonstrates the PERDYN90312 IP core

        reg clock_tb;                           // testbench generated clock
        reg reset_IP_n;                         // testbench generated reset
        reg [15:0] command_tb;                  // instruction sent to IP core
        reg latch_tb;                           // instruction latch (command_t)
        wire red_tb,green_tb,blue_tb,cycle_tb;  // outputs from IP core

        initial
                begin
                clock_tb = 0;
                reset_IP_n = 1;
                latch_tb = 0;

                #100 reset_IP_n = 0;
                #100 reset_IP_n = 1;

                #100 command_tb <= 16'hE000;    // control
                #100 latch_tb = 1;
                #100 latch_tb = 0;

                #100 command_tb <= 16'h2008;    // red
                #100 latch_tb = 1;
                #100 latch_tb = 0;

                #100 command_tb <= 16'h6080;    // green
                #100 latch_tb = 1;
                #100 latch_tb = 0;

                #100 command_tb <= 16'hA800;    // blue
                #100 latch_tb = 1;
                #100 latch_tb = 0;
                end

        always
                #5 clock_tb = ~clock_tb;        // generate IP clock


        PERDYN90312 U1 (                        // instantiate PERDYN90312 core
                .clock(clock_tb),
                .reset(reset_IP_n),
                .command(command_tb),
                .latch(latch_tb),
                .red(red_tb),
                .green(green_tb),
                .blue(blue_tb),
                .cycle(cycle_tb)
                );


endmodule
```


IMPORTANT NOTICE

Peripheral Dynamics reserves the right to change minor performance and/or implementation specifications without notice. Customers are advised to obtain the latest versions of product specifications, which should be considered when evaluating a product's appropriateness for a particular use.

BY USING THIS PRODUCT, CUSTOMER AGREES THAT IN NO EVENT SHALL PERIPHERAL DYNAMICS, ITS PARENT COMPANY, SUBSIDIARIES, OR PARTNERS BE LIABLE FOR ANY DIRECT, INDIRECT, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES AS A RESULT OF THE PERFORMANCE, OR FAILURE TO PERFORM, OF THIS PRODUCT. PERIPHERAL DYNAMICS MAKES NO OTHER WARRANTIES, EXPRESSED OR IMPLIED, INCLUDING ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

PERIPHERAL DYNAMICS PRODUCTS ARE NOT AUTHORIZED FOR USE IN LIFE SUPPORT DEVICES OR SYSTEMS. Life support devices or systems are those which are intended to support or sustain life and whose failure to perform can be reasonably expected to result in a significant injury or death to the user.