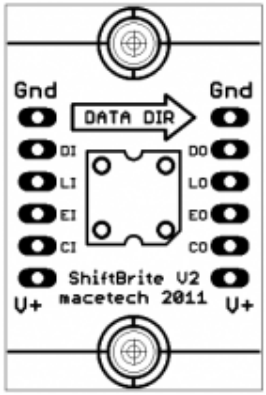
macetech documentation

ShiftBrite 2.0

A ShiftBrite 2.0 [http://macetech.com/store/index.php?main_page=product_info&cPath=1&products_id=1] has an RGB LED and a small controller chip, the Allegro A6281. The A6281 provides 10-bit PWM and 7-bit current control for each of the red, green, and blue LEDs. It uses a simple clocked serial interface to receive a 10-bit brightness value for each color, resulting in over a billion possible colors. Each input is buffered and output on the other side of the module. This allows each ShiftBrite to repeat the signal to the next, allowing longer cable runs between elements without excessive loading of microcontroller I/O pins. The six input and six output pins are 0.1" headers spaced 0.6 inches apart, allowing a ShiftBrite to fit on a breadboard or wide DIP IC socket for testing.

ShiftBrite elements feature current control and automatic overtemperature control (an overheating channel driver will shut off until it has cooled). Each channel can also be adjusted with a separate current control register, for fine tuning of each LED if close brightness matching is necessary. The integrated voltage regulator powers the internal logic, allowing a single 5.5 to 9 volt supply rail to power the ShiftBrite chain.

Features



Controller:	Allegro A6281
LED Brightness:	8000mcd per color
LED Viewing Angle:	140 Degrees
Current:	60mA maximum (all channels on)
Power Supply:	5.5V to 9V DC
PCB Size:	0.8 x 1.2 inches
Pin Spacing:	0.1 inches
Pin Pitch:	0.6 inches (wide DIP)

Power Connections

The **V+** and **GND** pins power both the LED and the control chip. ShiftBrites require up to 60mA per module when all LEDs are active. The supply voltage should be kept between 5.5 and 9 volts. I have had good results with 6V and 7.5V power supplies.

As is true of most digital electronics, if you connect power backwards or short contacts with metal objects or other modules on the chain, you WILL destroy the logic chip. Please triple-check all cables and make sure the modules are not in contact with anything conductive BEFORE applying power.

Data In / Data Out

The **DI** (Data In) pin carries the actual control information into the ShiftBrite. It is the input to an internal 32-bit shift register. Every time data is shifted into the controller, the binary value on the **DI** pin is placed in Bit 0 of the shift register, and the value in Bit 31 overflows out the **DO** (Data Out) pin to the next ShiftBrite in the chain. Data is shifted in using MSB (most significant bit first).

Clock In / Clock Out

The **CI** (Clock In) pin controls the shifting process. Each time the **CI** pin is sent to logic high and low, data is shifted into the **DI** pin and out of the **DO** pin. The **CI** signal is passed through the ShiftBrite to the **CO** (Clock Output) pin, so the next ShiftBrite can receive the bits from the **DO** line.

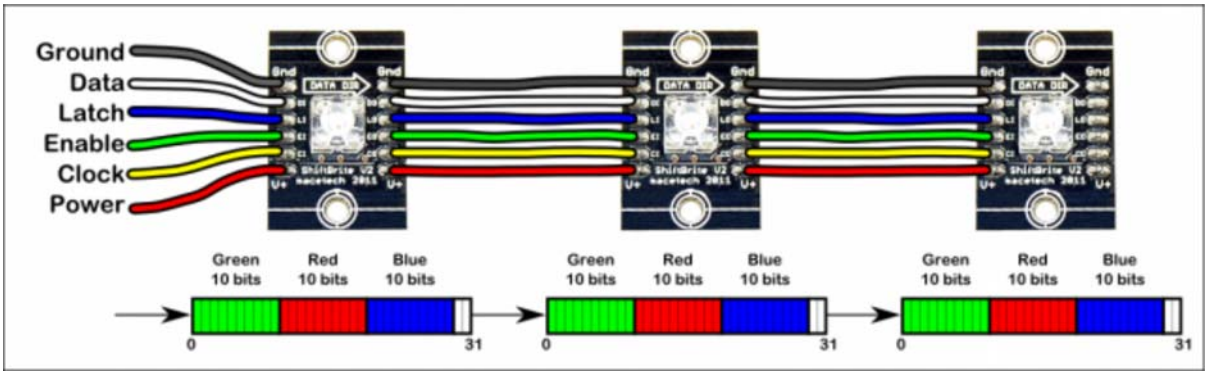
Latch In / Latch Out

The **LI** (Latch Input) pin causes the ShiftBrite to accept whatever is in its shift register as a new command. If you send the **LI** pin high and then low after 32 clocks, the first ShiftBrite in the chain has all new data from the **DI** pin. The second ShiftBrite contains whatever was already in the first ShiftBrite, and so on. To command all ShiftBrites in a chain, you must toggle the **LI** pin after you have shifted data to all ShiftBrites; 32 clock cycles times the number of ShiftBrites in the chain. The **LI** pin passes through to the **LO** (Latch Output) pin.

Enable In / Enable Out

The **EI** (Enable Input) turns the entire chain on and off. If it is sent to logic high, then it will blank all ShiftBrites. When **EI** is low, all ShiftBrites will display the colors specified previously. The **EI** pin passes through to the **EO** pin.

Command Format



0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
PWM Counter 0										PWM Counter 1										PWM Counter 2										0	X
Dot Correction 0							Clock Mode		X	Dot Correction 1							X	X	X	Dot Correction 2						X	ATB	ATB	1	X

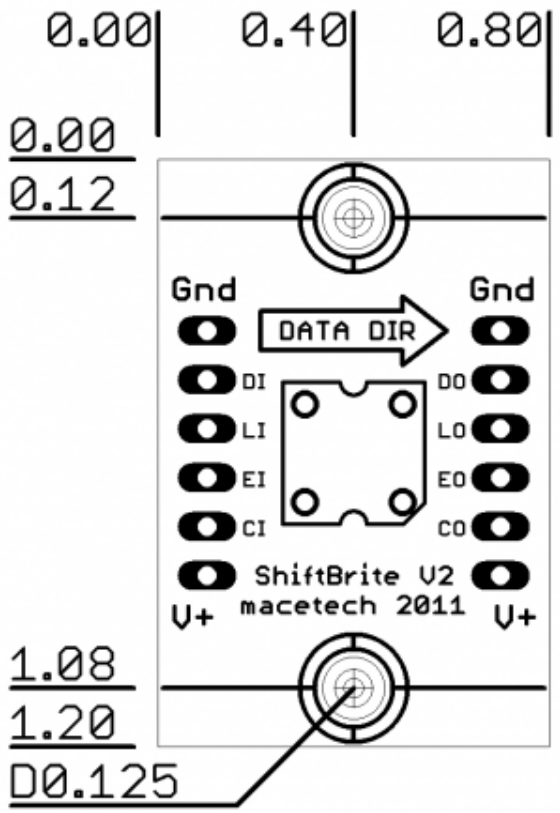
Each color has 10 bits of the 32 bit command word, as shown in the above diagram. The 10 bits allow PWM (pulse width modulation) control of LED brightness to 1024 possible levels. This is only true if the remaining 2 bits are set to zero. Otherwise, the ShiftBrite enters a command mode that allows current control and other parameters. This should always be set as shown in the code below, unless you have a specific reason to do otherwise. Additional information is available at the bottom of page 7 in the A6281 datasheet [http://www.allegromicro.com/en/Products/Part_Numbers/6281/6281.pdf].

Essentially, when Bit 30 is **1** the ShiftBrite enters command mode, the last row of the table above. You can set a 7-bit current control (dot correction) value for each LED, 0 to 127 corresponds to about 33% to 100% power. The Clock Mode setting lets you change the PWM frequency or use an external source; it is normally set to 00 which is an 800KHz internal PWM (please refer to the A6281 datasheet for other settings if needed). The bits marked "X" have no effect on the device. The bits marked "ATB" are for Allegro internal testing, but have the side effect of stopping the output buffers. That means if you accidentally write an ATB due to noise, you must write to each ShiftBrite's command register to be able to rewrite the next one.

Since the control mode can set the ShiftBrite to unusable modes, it is a good idea to write the control register often. Most of our example code writes it each time the color values are written.

Dimensions

All dimensions are in inches.



Code Example

The following Arduino code illustrates using the built-in SPI hardware for faster updates, though it is not really needed for only two ShiftBrites. This code should flash two ShiftBrites (or ShiftBars, or MegaBrites) alternating red and blue. If a ShiftBrite Shield or Shifty VU Shield is used, you should only have to plug in the cables and attach an external power supply, taking care that the ShiftBrite signal names are matched to the shield signal names.

Once you have this code working, you should be able to use it as a basis for any ShiftBrite application. Simply adjust the NumLEDs definition at the beginning of the program, and then write your desired color values from 0 to 1023 in the LEDChannels array. Once all the colors are in the array, call the WriteLEDArray() function and the ShiftBrites should switch to their new colors.

```
#define clockpin 13 // CI
#define enablepin 10 // EI
#define latchpin 9 // LI
#define datapin 11 // DI

#define NumLEDs 2

int LEDChannels[NumLEDs][3] = {0};
int SB_CommandMode;
int SB_RedCommand;
int SB_GreenCommand;
int SB_BlueCommand;

void setup() {

   pinMode(datapin, OUTPUT);
   pinMode(latchpin, OUTPUT);
   pinMode(enablepin, OUTPUT);
   pinMode(clockpin, OUTPUT);
   SPCR = (1<<SPE)|(1<<MSTR)|(0<<SPR1)|(0<<SPR0);
```

```
   digitalWrite(latchpin, LOW);
   digitalWrite(enablepin, LOW);


}

void SB_SendPacket() {

    if (SB_CommandMode == B01) {
     SB_RedCommand = 120;
     SB_GreenCommand = 100;
     SB_BlueCommand = 100;
    }

    SPDR = SB_CommandMode << 6 | SB_BlueCommand>>4;
    while(!(SPSR & (1<<SPIF)));
    SPDR = SB_BlueCommand<<4 | SB_RedCommand>>6;
    while(!(SPSR & (1<<SPIF)));
    SPDR = SB_RedCommand << 2 | SB_GreenCommand>>8;
    while(!(SPSR & (1<<SPIF)));
    SPDR = SB_GreenCommand;
    while(!(SPSR & (1<<SPIF)));

}

void WriteLEDArray() {

    SB_CommandMode = B00; // Write to PWM control registers
    for (int h = 0;h<NumLEDs;h++) {
          SB_RedCommand = LEDChannels[h][0];
          SB_GreenCommand = LEDChannels[h][1];
          SB_BlueCommand = LEDChannels[h][2];
          SB_SendPacket();
    }

    delayMicroseconds(15);
    digitalWrite(latchpin,HIGH); // latch data into registers
    delayMicroseconds(15);
    digitalWrite(latchpin,LOW);

    SB_CommandMode = B01; // Write to current control registers
    for (int z = 0; z < NumLEDs; z++) SB_SendPacket();
    delayMicroseconds(15);
    digitalWrite(latchpin,HIGH); // latch data into registers
    delayMicroseconds(15);
    digitalWrite(latchpin,LOW);

}

void loop() {

   LEDChannels[0][0] = 1023;
   LEDChannels[0][1] = 0;
   LEDChannels[0][2] = 0;

   LEDChannels[1][0] = 0;
   LEDChannels[1][1] = 0;
   LEDChannels[1][2] = 1023;

   WriteLEDArray();
   delay(200);

   LEDChannels[0][0] = 0;
   LEDChannels[0][1] = 0;
   LEDChannels[0][2] = 1023;

   LEDChannels[1][0] = 1023;
   LEDChannels[1][1] = 0;
   LEDChannels[1][2] = 0;
```

```
    WriteLEDArray();
    delay(200);


}
```