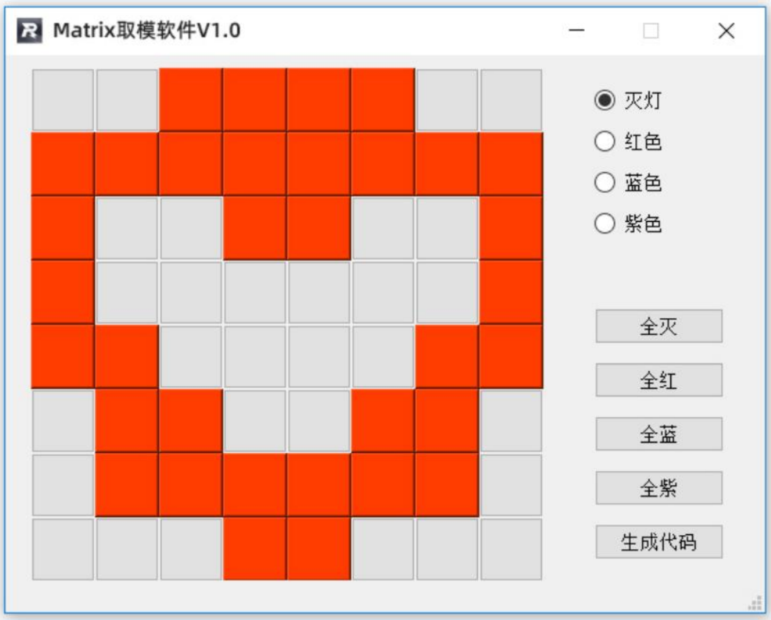
```
            static void Init(uint8_t gcc); static void
            SetGCC(uint8_t gcc); static void
            SetLEDStatus(uint8_t cs, uint8_t sw, IS31FL3733_LED_STATE
state);
            static void SetLEDPWM(uint8_t cs, uint8_t sw, uint8_t value); static void
            SetAllPWM(uint8_t *val);

            /*Below are some
            internal functions*/ static uint8_t ReadCommonReg(uint8_t
            reg_addr); static void WriteCommonReg(uint8_t reg_addr, uint8_t reg_value); static
            void SetLEDMode(uint8_t cs, uint8_t sw, IS31FL3733_LED_MODE
mode);
            static void ConfigABM(IS31FL3733_ABM_NUM n, IS31FL3733_ABM *config); static void
            StartABM();
};
```

Commonly used function name	parameter	range	Parameter interpretation	function
On	/	/	/	open dot matrix
Off	/	/	/	turn off dot matrix
Heat	gcc	0-127	gcc global current control	The dot matrix must be initialized first
SetGCC	gcc	0-127	gcc global current control	Set global current (i.e. control brightness)
SetLEDStatus	cs, sw, state	0-8 / 0-16 / {0,1}	cs horizontal row number sw vertical column number state LED status	Switches that control all LEDs
SetLEDPWM	cs, sw, value	0-8 / 0-16 / 0-255	cs horizontal row number sw vertical column number value LED color value	Control the color value of all LEDs
SetAllPWM	*val	One-dimensional array of size 128	The detailed structure of the array is shown below	set bitmap

Dot matrix modulo and how to use it

In the process of controlling the dot matrix LED screen, we need to make the screen display the content we need. In embedded development, unless you have a packaged display library, you usually need to write a program yourself to control the brightness of each pixel and each LED from the bottom layer, in order to achieve the effect of displaying the content we want.

This means that even if we need to display simple content such as numbers and text, we need to control the brightness of each LED to achieve the corresponding effect. Fortunately, you don't need to implement this control process yourself, just use the modulo software to draw the content to be displayed in a simple graphical way, and then you can get the automatically generated dot matrix code from the modulo software.



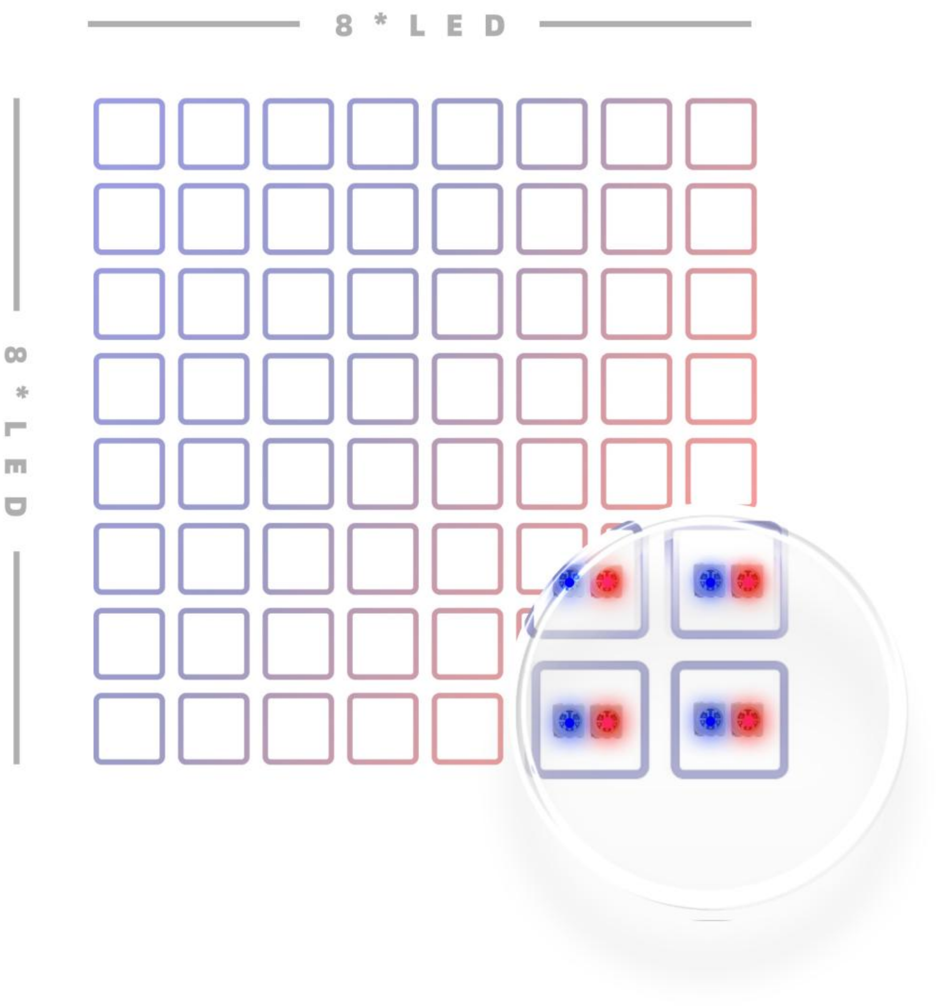
Open matrix.exe under the folder "Lattice Modulo", and then draw the graphics you need. click 生成代码 ,Right now

The corresponding lattice array can be obtained. Copy it into your project and it's ready to use.

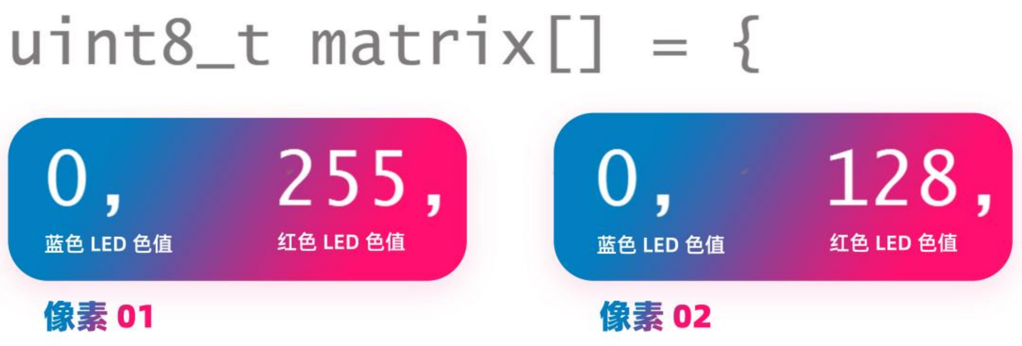
```
uint8_t matrix[] = {
          0,    0,    0,    0, 0, 255, 0, 255, 0, 255, 0, 255, 0, 0, 0,
0,
          0, 255,         0, 255, 0, 255, 0, 255, 0, 255, 0, 255, 0, 255, 0,
255,
          0, 255,         0,    0, 0, 0,          0, 255, 0, 255, 0,          0,    0, 0, 0,
255,
          0, 255,         0,    0, 0, 0,          0,    0,    0,    0,    0,    0,    0, 0, 0,
255,
          0, 255,         0, 255, 0, 0,          0,    0,    0,    0,    0,    0,    0, 255, 0,
255,
          0,    0,    0, 255, 0, 255, 0,               0,    0,    0,    0, 255, 0, 255, 0,
0,
          0,    0,    0, 255, 0, 255, 0, 255, 0, 255, 0, 255, 0, 255, 0,
0,
          0,    0,    0,    0, 0, 0,          0, 255, 0, 255, 0,          0,    0, 0, 0,

0, };
```

The code generated by the modulo software is shown in the figure. Looking confused? In fact, its structure is not that complicated.

Let's take a look at the hardware composition of the dot matrix LED screen.



The dot matrix screen on the Talent expansion module consists of 64 red LEDs and 64 blue LEDs, a total of 128 LEDs. As shown in the figure, two SMD LEDs, one red and one blue, are installed in each grid (pixel) (although they are packaged in the same SMD component).



```
uint8_t matrix[] = {
```

Based on the hardware, we need to use a one-dimensional array with a size of 64 (number of pixels) * 2 (number of colors) = 128 to represent the brightness and darkness of **each LED in each grid.**

From the picture above, you can know that the display of each pixel requires the color values of the blue and red channels for control.

For easy understanding, we can divide the data into 64 groups, each with two color values. These data are arranged from top to bottom and left to right in a one-dimensional array with a size of 128 , respectively representing the LED color value data of 64 pixels.

We can summarize a set of rules:

> 2 * n represents the blue light in the nth pixel **"n belongs to [0, 7]"**

> 2 * n + 1 represents the red light in the nth pixel

```
tt_matrix.SetAllPWM((uint8_t*)matrix);
```

After generating the matrix data you need, call the SetAllPWM function and pass in the matrix data generated by the modulo software to light up the dot matrix screen. (Please remember to initialize the dot matrix screen module before use)

Control Example - Digital Display

```
#include <RMTT_Libs.h>
#include <Wire.h>

RMTT_Matrix tt_matrix;

uint8_t matrix_b3[] = { 0,  0,  0,
                         0, 0, 0, 255,           0, 255,     0,    0,    0,    0, 0, 0,
0,
       0,    0,    0,    0, 255, 0,        0,    0,    0,    0, 255,     0,    0, 0, 0,
0,
       0,    0,    0,    0, 0, 0,          0,    0,    0,    0, 255,     0,    0, 0, 0,
0,
       0,    0,    0,    0, 0, 0, 255,           0, 255,     0,    0,    0,    0, 0, 0,
0,
       0,    0,    0,    0, 0, 0,          0,    0,    0,    0, 255,     0,    0, 0, 0,
0,
       0,    0,    0,    0, 255, 0,        0,    0,    0,    0, 255,     0,    0, 0, 0,
0,
       0,    0,    0,    0, 0, 0, 255,           0, 255,     0,    0,    0,    0, 0, 0,
0,
       0,    0,    0,    0, 0, 0,          0,    0,    0,    0,    0,    0,    0, 0, 0,

0, };

uint8_t matrix_b2[] = { 0,  0,  0,
                         0, 0, 0, 255,           0, 255,     0,    0,    0,    0, 0, 0,
0,
       0,    0,    0,    0, 255, 0,        0,    0,    0,    0, 255,     0,    0, 0, 0,
0,
       0,    0,    0,    0, 0, 0,          0,    0,    0,    0, 255,     0,    0, 0, 0,
0,
       0,    0,    0,    0, 0, 0,          0,    0, 255,     0,    0,    0,    0, 0, 0,
0,
       0,    0,    0,    0, 0, 0, 255,           0,    0,    0,    0,    0,    0, 0, 0,
0,
       0,    0,    0,    0, 255, 0,        0,    0,    0,    0,    0,    0,    0, 0, 0,
0,
       0,    0,    0,    0, 255, 0, 255,         0, 255,     0, 255,     0,    0, 0, 0,
0,
       0,    0,    0,    0, 0, 0,          0,    0,    0,    0,    0,    0,    0, 0, 0,

0, };

uint8_t matrix_b1[] = { 0,  0,  0,
                         0, 0, 0, 255,           0, 255,     0,    0,    0,    0, 0, 0,
0,
```