EduCake にて LED マトリックスの使い方

一、 LED マトリックスの原理紹介

86Duino EduCake を使用する前に、単一や LED の数が少ないデジタル/アナログ出力機能制御を導入したが、方法はこれ 1 ピンは LED 制御している、不利な点は、より多くの制御ピンを取る必要があるかの他の機能でプロジェクトまた、これらのピンは便利ではない必要があります。あなたはなど、LED の破片数百人、電球の制御十たい場合は読者が知りたいはず、使用できる任意の方法はありますか？この章では、86Duino EduCake 制御複数の LED マトリクス方式の使用を紹介します、LED マトリックス表示、テキスト表示機能原理の で、鉄道の駅のように LED マーキーを使用して図面を作成する方法を学習します。図 1、図 2 に示すような一般的な市場は、例えば、非常に多様、しかし、今、LED8X8 行列に 2 アノードコモンカソードコモンに分かれた構造を LED マトリクスを購入すること

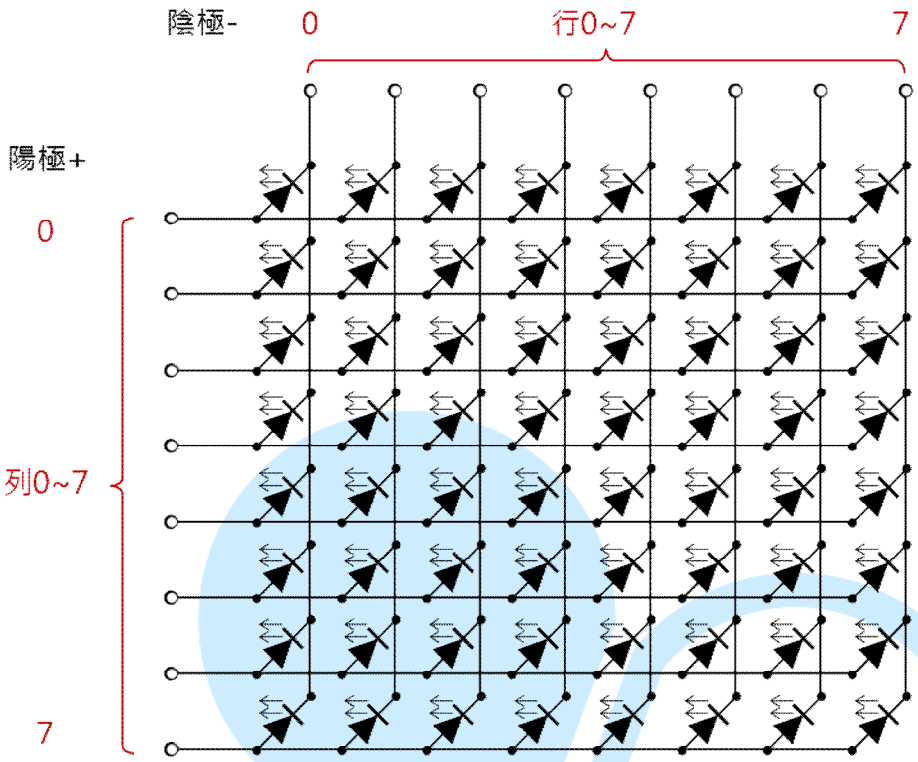


図 1. 共正極 LED マトリックス

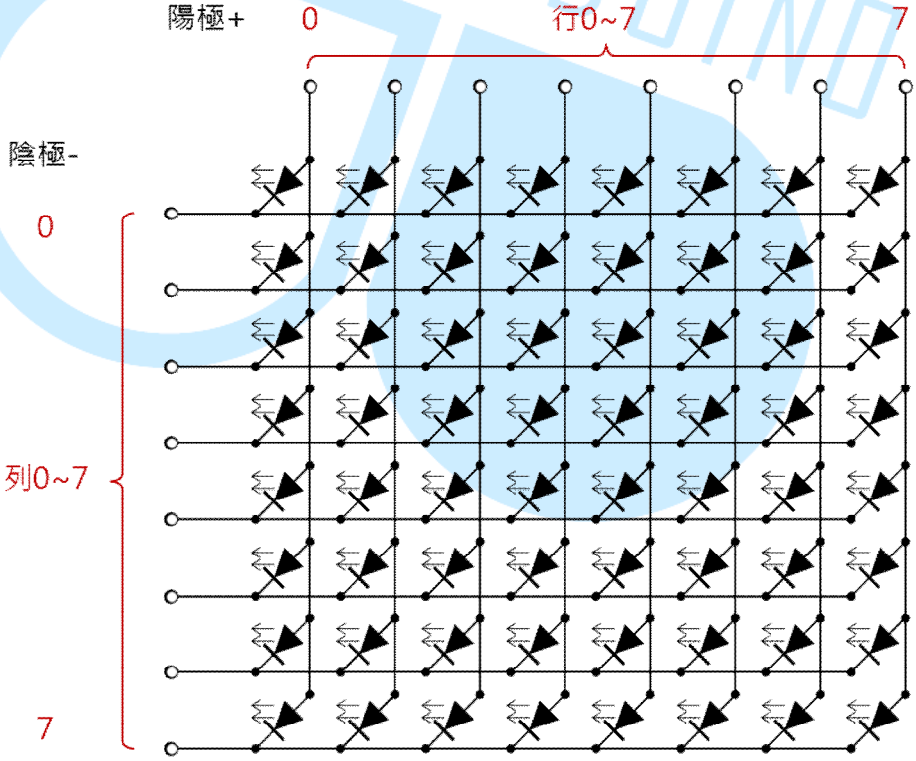


図 2. 共負極の LED マトリックス

実際にはモノクロのみ実際に行列、共通のアノードまたはカソードコモンの LED (発光ダイオードカソードまたはアノードの行全体があると言ったが、同じ点に接続され、上の図では、いわゆる共通のアノードまたは共通カソードから見ることができます異なる方向に配置され、) 違い多色 LED があるでしょう。このアーキテクチャの LED マトリクスの前に、単一のコントロールのようにすることはできません、あなたがコントロールする「スキャン+残像」アプローチを使用する必要があります。図 3 は、いくつかの中で LED ライトの一回行、その後、原則表示画面その全体ピースに応じて、シーケンス内の次の行で置き換えに示すように。

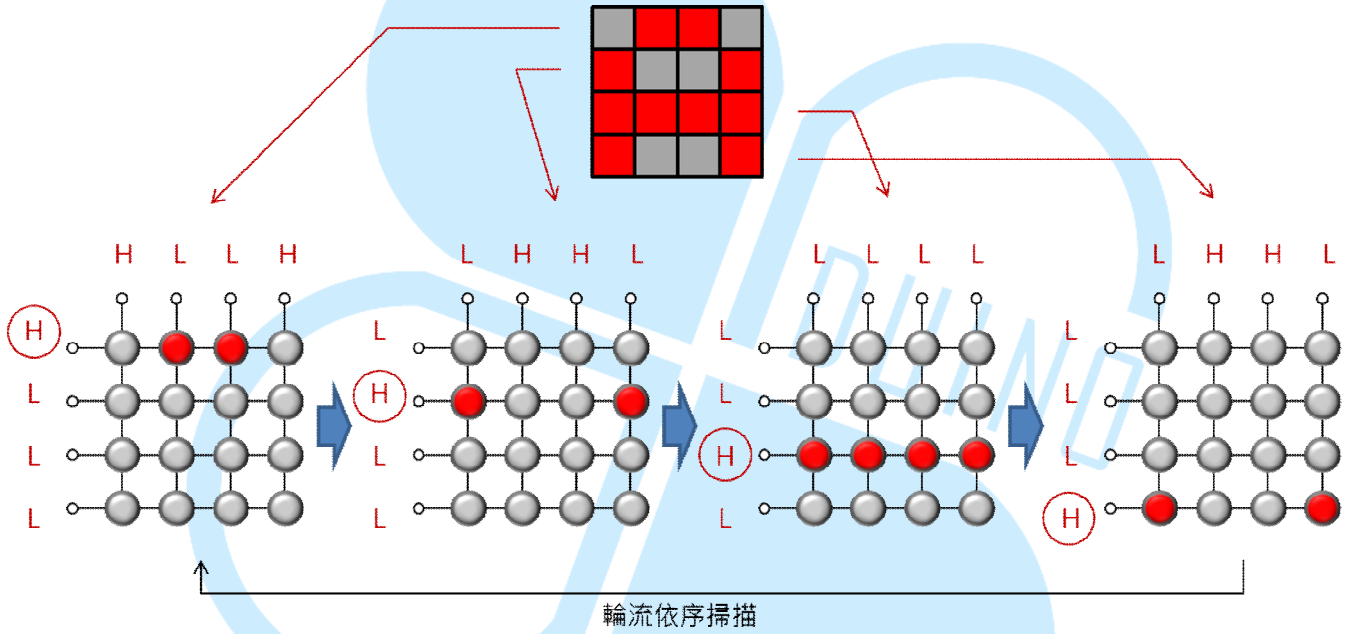


図 3. LED マトリクス スキャンコントローラ

このスキャンコントロールが制御するピンの数を減らすことができ、LED は、単にコントロールの上 86Duino EduCake を使用する場合に実行する必要がある、あなただけにいくつかの抵抗の 16 ピンを必要とする、と 64 ピン·コントロールを使用するよりも効率的ではありませんそれ？プログラムを引き起こすことに加えが複雑になるが、あなたは、複数の LED マトリクスを制御する必要がある場合は、同じ問題がピンが繰り返さ占めるでしょう。しかし、さらに実用化にはまだ不便をたくさん持っている、8X8 LED マトリクスのみ 16 フィートを占有制御します。

幸いなことに、市場では専用の LED 制御 IC が持っている、自動的に上記の作業を「スキャン」するようにユーザーを支援することができ、コードは単に IC の簡単なセットを行い、簡単な LED マトリクス制御を行うことができる。このセクションでは、単一および複数の 8X8 LED マトリクスを制御するために、関係して、LED マトリクス制御 IC で 86Duino EduCake を使用する方法を説明します。ここで、図 4 に示すように MAX7219、配線や構造用の LED 制御 IC モデルを使っています：

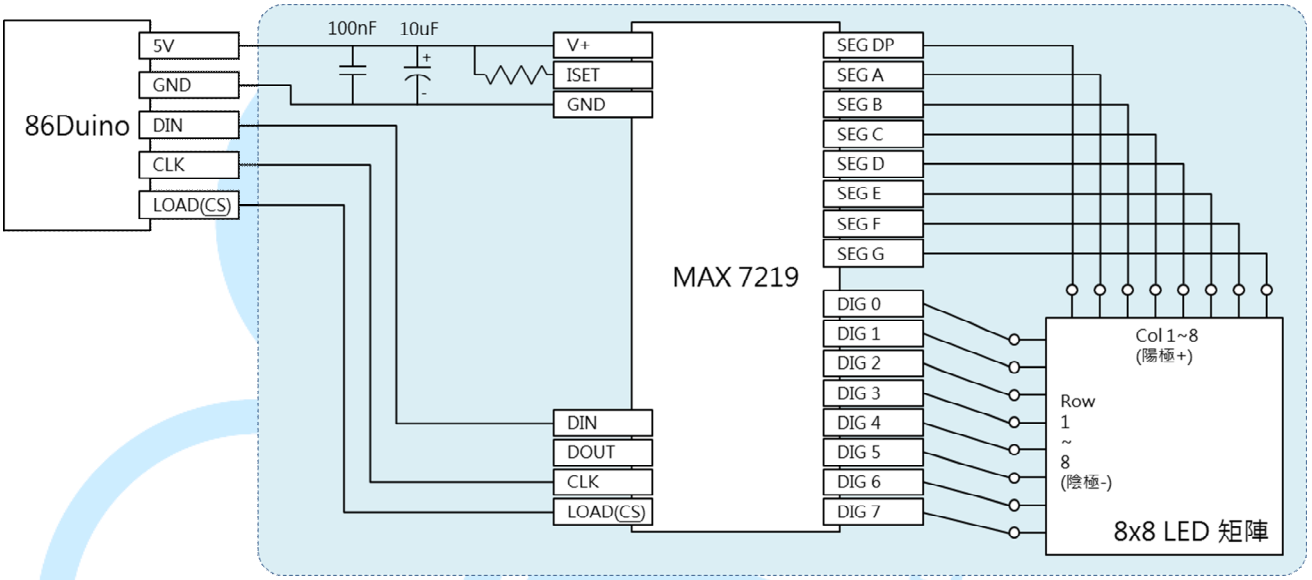


図 4. 86Duino EduCake + MAX7219

読者が配線図を参照することも、市場統合モジュールを購入することができ、チャートからライン MAX7219 と 86Duino EduCake、電源コードのための他の 2 つの間の唯一の 3 つの制御信号を見ることができます) バスケットの図内のコンポーネントがすべて含まれています。かなり合理化された MAX7219 を使用して信号を制御するだけでなく、複数のユニットを直列に接続することができ、スペースマスターコントローラピンを消費しない。図 4 は、複数のモジュールを使用して、この端子を "DIN"次のモジュールピンに接続する必要がある場合、MAX7219 ピン内「DOUT」があることがわかる、第一の残りの部分とピン結合例えば、複数のユニットは図 5 を参照して直列

に接続することができる既存の MAX7219+8X8 LED モジュールへの単位：

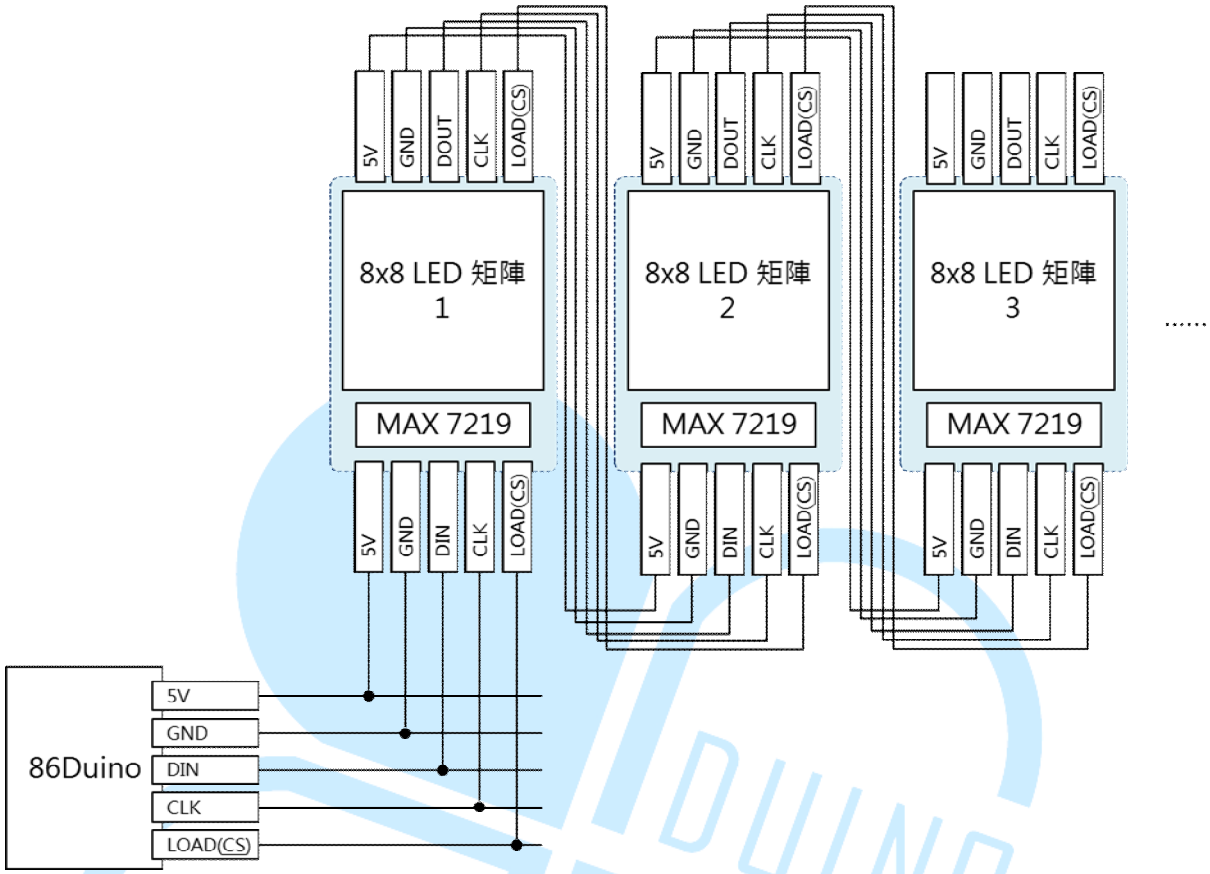
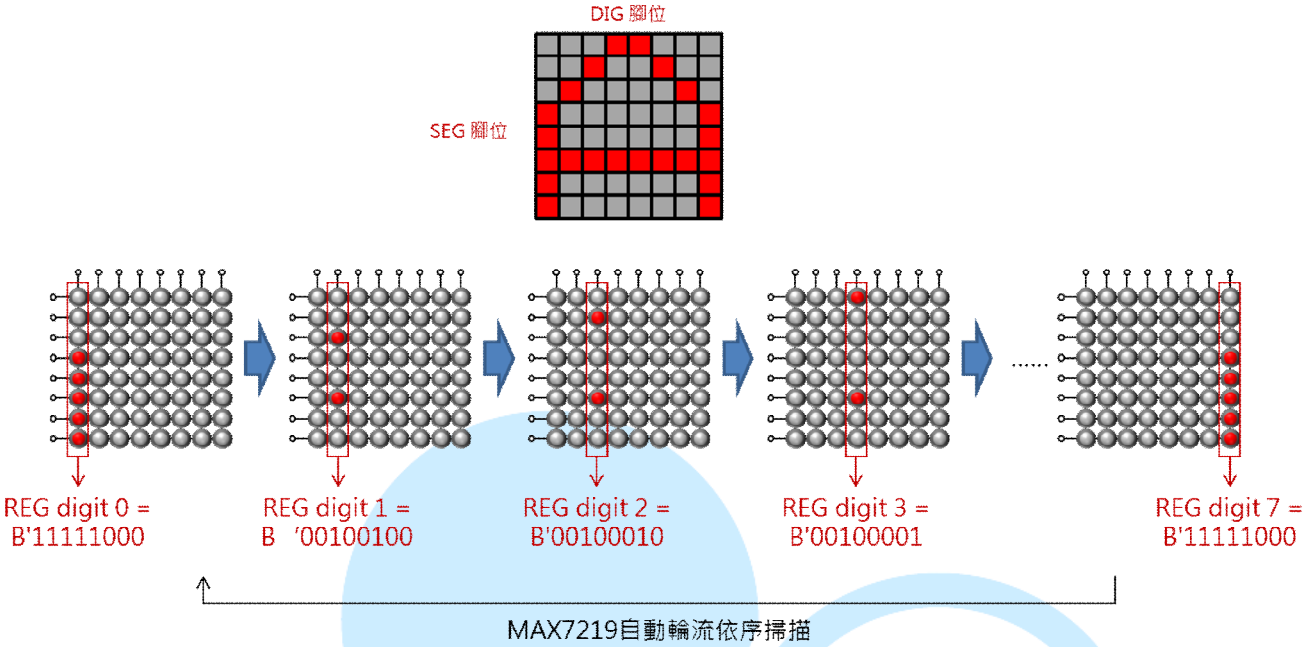


図 5. 86Duino EduCake + MAX7219 接続図

MAX7219 の LED マトリクス制御 IC は、使用するために、これらの 3 つの制御ピンを LOAD、DIN、CLK を使用する必要がある;実際、これは "SPI"通信インターフェースの既知の部分に似ていますが、86Duino EduCake ブレッドボードは、SPI を予約されていないハードウェアのピンソケット (実際の CPU がサポートされていますが、ラインを引いていなかった) が、私たちはまだ、ソフトウェアシミュレーションにより SPI 通信を行うことができます。データシーケンスの送信を担当ピン DIN は、CLK 端子はクロック同期を送信し、LOAD 端子 MAX7219 デバイスが Enable/Disable ラインを可能にするために使用。MAX7219 を使用する場合は、ホストコントローラが IC をいくつかのレジスタ (レジスタ) 書かれたセットを作る、MAX7219 スクラッチパッドを次の表に定義されているの DIN ピンを介して必要があります。

レジスター	アドレス	定義
No-Op	0xX0	なし
Digit 0	0xX1	Digit 0 ピンにマッピングしている行/列データ値
Digit 1	0xX2	Digit 1 ピンにマッピングしている行/列データ値
Digit 2	0xX3	Digit 2 ピンにマッピングしている行/列データ値
Digit 3	0xX4	Digit 3 ピンにマッピングしている行/列データ値
Digit 4	0xX5	Digit 4 ピンにマッピングしている行/列データ値
Digit 5	0xX6	Digit 5 ピンにマッピングしている行/列データ値
Digit 6	0xX7	Digit 6 ピンにマッピングしている行/列データ値
Digit 7	0xX8	Digit 7 ピンにマッピングしている行/列データ値
Decode Mode	0xX9	データ解析モードの起動
Intensity	0xXA	ブライト制御
Scan Limit	0xXB	Digit 0~Digit 7 スキャン範囲の設定
Shutdown	0xXC	LED マトリクスを出力を閉じるかとうかの設定
Display	0xXF	テストモード



Test		

MSB～LSB はそれぞれ SEG DP、SEG A、SEG B、SEG C、D SEG、SEG E、SEG F、LED の SEG G の端子接続から桁リソースセクション 0-7、LED ポジション少し明るく LOW HIGH ため、逆にオフになっている。MAX7219 IC 自家製境界線を使用するために、読者は、LED や電球は、さまざまなピンと情報との対応ことに留意すべきです。MAX7219 ピン名を観察した場合、読者が実際にこの IC は 8X8 に加えて、制御ユニットで使用される見つけるマトリックスの LED だけでなく、一般に呼ばれる 87 セグメントディスプレイ (7 セグメントディスプレイが 8 LED を有するが 制御するために使用することができる。DP、A、B、C、D、E、F、G) ので、読者は LED マトリクスを制御するために使用することを学ぶが、同じ原理は、7 セグメント表示のための行全体の制御に使用することができる。一般的な使用は、唯一のスクラッチパッドの内容を 0～7 桁数字を設定する必要が、彼らはマトリックス LED の表示パターンを制御することができます。ここに選ばれた MAX7219+ LED マトリックスモジュールに 表示時に底部に向かって IC 桁 0 に対応左側の行列の最初の行は、数字 1 は、2 行目の左側に対応する、というように。コンテンツ桁 0 のレジスタは、各ビットラインだけで明るさのモードに対応した 8BIT であり、ボトムアップで モジュールのデフォルトの接続モードを選択している SEG DP SEG A、SEG B、...、 SEG G.図 5 のプロセスに示す原理：上記の「スキャン」の作業 MAX7219 は、自動的に 0-7 の完全なセット、LED マトリクスのパターンを示すことができる桁スクラッチパッドの内容をある限り 完了するために行

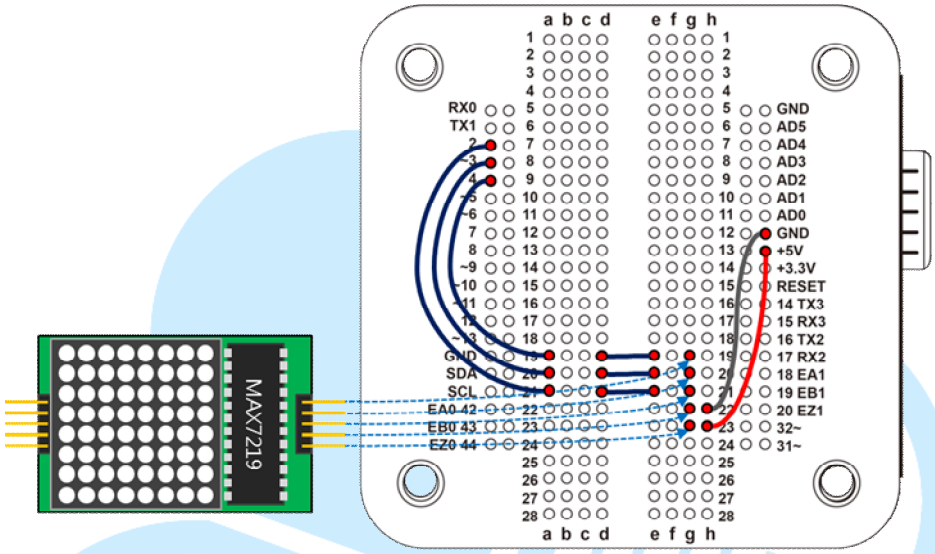
く。もちろん、コンテンツが定期的にパターンを変更するには、アニメーションを表示することができます。読者は他の配線アーキテクチャ LED マトリクスを作るためにした場合、ノートスクラッチパッドデータを取る必要があり、予想通りの LED の位置関係に対応し、パターンが表示されます。詳しく内容は、下記にリンク先を参考ください。

http://datasheets.maximintegrated.com/en/ds/MAX7219-MAX7221.pdf

ここでは実際の MAX7219 制御理論と LED マトリクス表示パターン機能の実用化にそれを行う上で説明したように、私たちは、プログラムを使用してみましょう。

二、第一プログラム – 単一 LED マトリックスの練習 1

一番目の例で 86Duino EduCake と MAX7219 モジュールの使い方を練習する最初のサンプルプログラムは、8X8 の LED マトリクスパターンのグループを次の図に従って配線をしてください。



86Duino Coding IDE にて下記のプログラムコードを入力ください。

```
// 制御ピンの定義
int DIN_pin = 2;
int LOAD_pin = 3;
int CLOCK_pin = 4;

// MAX7219 レジスタのアドレス
byte max7219_REG_noop        = 0x00;
byte max7219_REG_digit0      = 0x01;
byte max7219_REG_digit1      = 0x02;
byte max7219_REG_digit2      = 0x03;
byte max7219_REG_digit3      = 0x04;
byte max7219_REG_digit4      = 0x05;
byte max7219_REG_digit5      = 0x06;
byte max7219_REG_digit6      = 0x07;
byte max7219_REG_digit7      = 0x08;
```

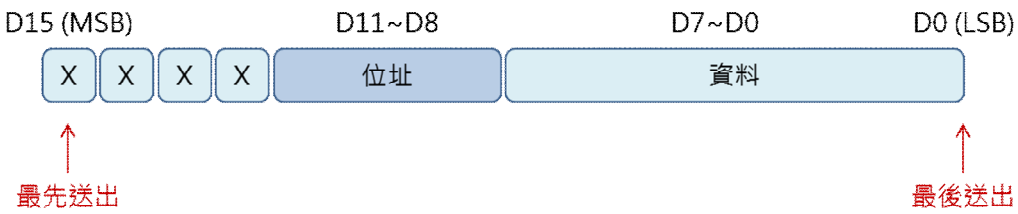
```
byte max7219_REG_decodeMode  = 0x09;
byte max7219_REG_intensity   = 0x0a;
byte max7219_REG_scanLimit   = 0x0b;
byte max7219_REG_shutdown    = 0x0c;
byte max7219_REG_displayTest = 0x0f;

void SPI_SendByte(byte data) {//
SPI インターフェースをまねして byte データを送る
  byte i = 8;
  byte mask;
  while(i > 0) {
    mask = 0x01 << (i - 1);// マスクを作って，一番左側から
    digitalWrite( CLOCK_pin, LOW);// クロック = LOW
    if (data & mask){// マスクにマッピングしている BIT は 0 あるいは 1
を判断する
      digitalWrite(DIN_pin, HIGH);// 1 にマッピングしたら、DIN か
ら HIGH を送る
    }
    else{
      digitalWrite(DIN_pin, LOW);// 1 にマッピングしたら、DIN か
ら LOW を送る    }
    digitalWrite(CLOCK_pin, HIGH);// クロック = HIGH
    --i;// 次の BIT を移る
  }
}

void MAX7219_1Unit(byte reg_addr, byte reg_data) {//
MAX7219 モジュールを制御
  digitalWrite(LOAD_pin, LOW);// 転送する前に、LOAD ピンを
LOW になる
  SPI_SendByte(reg_addr);// まず設定したレジスターアドレスを送り
べき
  SPI_SendByte(reg_data);// 次にデータを送る
  digitalWrite(LOAD_pin,HIGH);// }転送してから LOAD ピンが
HIGH になる
```

```
byte matrixData_8X8[8] = {// グラフデータマトリックス
  B01010101,// 第一行目から順番に
  B10000001,
  B10101010,
  B11111111,
  B00000000,
  B11110000,
  B00001111,
  B11001100
};

void Draw (byte *LED_matrix)// 全画面を描く
{
  MAX7219_1Unit(1, LED_matrix[0]);
  MAX7219_1Unit(2, LED_matrix[1]);
  MAX7219_1Unit(3, LED_matrix[2]);
  MAX7219_1Unit(4, LED_matrix[3]);
  MAX7219_1Unit(5, LED_matrix[4]);
  MAX7219_1Unit(6, LED_matrix[5]);
  MAX7219_1Unit(7, LED_matrix[6]);
  MAX7219_1Unit(8, LED_matrix[7]);
}

void setup ( ) {
  pinMode(DIN_pin, OUTPUT);
  pinMode(CLOCK_pin, OUTPUT);
  pinMode(LOAD_pin, OUTPUT);

  digitalWrite(CLOCK_pin, HIGH);

  // MAX7219 レジスターを初期化する
  MAX7219_1Unit(max7219_REG_scanLimit,  0x07);// 全て
の行列をスキャンすることを設定する
```

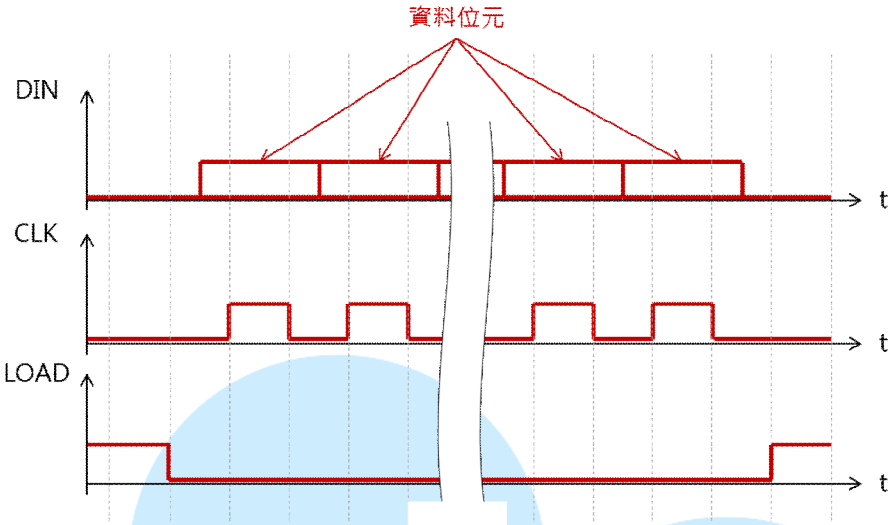
```
    MAX7219_1Unit(max7219_REG_decodeMode, 0x00);// デコードモードを閉じる
    MAX7219_1Unit(max7219_REG_shutdown, 0x01);// シャットダウンモードを閉じる
    MAX7219_1Unit(max7219_REG_displayTest, 0x00); // テストモードを閉じる

    for(int i=1; i<=8; i++) {// まず、全ての LED マトリックスを暗くさせる
      MAX7219_1Unit(i,0);
    }

    MAX7219_1Unit(max7219_REG_intensity, 0x0f);// ブライト範囲を設定する, 0x00 ~ 0x0f

    delay(1000);
  }
    void loop ( ) {
    Draw(matrixData_8X8);
    delay(500);
  }
```

データを書き込んだあと、下記の写真が見られます。

MAX7219 の 0～7 スクラッチパッドの内容を設定するための、このサンプル·プログラム機能は、そのように LED マトリックスは、単一のパターンを表示し、このパターンは、迅速に、LED マトリクスディスプレイモジュール及びデータ定義の方向との関係を同定することができます。この機能は、SPI はデータのバイトを送信するシミュレートするために使用されているプログラムの最初のセットを起動する制御ピン番号だけでなく、すべての MAX7219 のレジスタのアドレス、その後 SPI_SendByte(バイトデータ)、MAX7219_1Unit (バイト REG_ADDR、バイト reg_data) が使用されている特定のデータセットに MAX7219 ユニットを作成します。データビット D0～D7、D8～D11 アドレスに対して、D12～D15 には定義されていない、D15 への配列データ (MSB) は、各データ書き込みレジスタ部 16BITS、一度 MAX7219_1Unit ( ) 関数概略的に示されるように、送信順序の D0 (LSB)



DIN データ線と CLK クロック同期ライン波形図



データは DIN する準備が整うと、CLK が HIGH に変更すると、L になり、負荷線は、転送負荷の前に設定されている間、移動が高くなると終えました。matrixData_8X8 マトリックスはパターンデータを格納するために使用されている[8]、LED0 が消灯し、LED 点灯用である。ドロー ( ) 関数は、それが全体の 8×8 マトリクス画面を描画する必要があるため、そのように構文 MAX7219_1Unit を (使用) は、8 本のデータ線が設定画面パターンを完成するように設定されているます。setup( ) に機能することができるように初期化する必要性を登録 MAX7219 初期化多数のレジスタに加えて、初期値、、と足ビットの I / O モードの使用を設定することに加えて、セットアップ(内側)。ループは、(内部) 500ms の呼び出しドロー ( ) 関数を計時される。ここで、同じ、もちろん、それは残っている静的パターンを推移しているマトリックス表示 matrixData_8X8 コンテンツ情報として用います。

三、第二プログラム – 単一 LED マトリックスの練習 2

```
int shift = 0;
void ShiftDraw(byte *LED_matrix)// シフトで全画面を描く
{
  MAX7219_1Unit(1, LED_matrix[(shift) % 8]);// 第 1 行目のデータを描く
  MAX7219_1Unit(2, LED_matrix[(shift+1) % 8]);// 第 2 行目のデータを描く
  MAX7219_1Unit(3, LED_matrix[(shift+2) % 8]);// 第 3 行目のデータを描く
  MAX7219_1Unit(4, LED_matrix[(shift+3) % 8]);// 第 4 行目のデータを描く
  MAX7219_1Unit(5, LED_matrix[(shift+4) % 8]);// 第 5 行目のデータを描く
  MAX7219_1Unit(6, LED_matrix[(shift+5) % 8]);// 第 6 行目のデータを描く
  MAX7219_1Unit(7, LED_matrix[(shift+6) % 8]);// 第 7 行目のデータを描く
  MAX7219_1Unit(8, LED_matrix[(shift+7) % 8]);// 第 8 行目のデータを描く

  shift++;
  if(shift>=8){// INDEX を 0～7 に循環させる
    shift = 0;
  }
}
```

基本的な原則に 86Duino EduCake 制御 MAX7219 の LED マトリクスを学び、パターンがそれをアニメーションになるように、その後、ほとんど変化しません。読者オープン 86Duino コーディング IDE は、次に、サンプルプログラムに次のコードを追加します。その後、元のループ（）「ドロー（matrixData_8X8）; "に変更」ShiftDraw（matrixData_8X8）は、「左シフトのラインによって、パターンを作ることができるようになります。ここで使用 ShiftDraw（）原則は、各行のインデックス値の輝度情報を LED セットとしてシフトして、インデックス可変シフトを使用することです。一つシフト刻み、こうして matrixData_8X8 行列順次異なる場所に対応するには、変位に到達することができますエフェクトのパターン。（シフト+ N）を使用する必要があり、パターンデータのみ matrixData_8X8 行列インデックスは 0 から 7 の範囲であることに留意されたい実際のインデックス値を計算し、値が間違い

を避けるために 0～7 の範囲に制限される。同じ変数が各シフト 1 の後に、範囲外のチェックは、変数が範囲外の場合も 0 に行く必要がありますので、0 から 7 まで値サイクルシフトを停止することはありません。また matrixData_8X8 行列が異なるパターンに設定されているに試してみるか、他の効果を試すために、異なる屈折率変化を使用することができます。

四、第三プログラム – 単一 LED マトリックスの練習 3

86Duino Coding IDE、「matrixData_8X8」にて下記のプログラムコードを入力ください。

```
// 0
byte matrixData_num_0[8] = {// グラフデータマトリックス
  B00000000,// 第一行目から順番に
  B01111110,
  B10010001,
  B10001001,
  B10001001,
  B10000101,
  B01111110,
  B00000000
};
// 1
byte matrixData_num_1[8] = {// グラフ 1 データマトリックス
  B00000000,// 第一行目から順番に
  B00000000,
  B10000000,
  B10000010,
  B11111111,
  B10000000,
  B00000000,
```

```
  B00000000
  };
  // 2
  byte matrixData_num_2[8] = {// グラフ 1 データマトリックス
    B00000000,// 第一行目から順番に
    B10000110,
    B10000001,
    B11000001,
    B10100001,
    B10010001,
    B10001110,
    B00000000
  };
  // 3
  byte matrixData_num_3[8] = {// グラフ 1 データマトリックス
    B00000000,// 第一行目から順番に
    B01000010,
    B10001001,
    B10001001,
    B10001001,
    B10001001,
    B01110110,
    B00000000
  };
  // 4
  byte matrixData_num_4[8] = {// グラフ 1 データマトリックス
    B00000000,// 第一行目から順番に
    B00110000,
    B00101000,
    B00100100,
    B00100010,
    B11111111,
    B00100000,
```

```
    B00000000
  };
  // 5
  byte matrixData_num_5[8] = {// グラフ 1 データマトリックス
    B00000000,// 第一行目から順番に
    B01001111,
    B10001001,
    B10001001,
    B10001001,
    B10001001,
    B01110011,
    B00000000
  };
  // 6
  byte matrixData_num_6[8] = {// グラフ 1 データマトリックス
    B00000000,// 第一行目から順番に
    B01111110,
    B10001001,
    B10001001,
    B10001001,
    B10001001,
    B01110010,
    B00000000
  };
  // 7
  byte matrixData_num_7[8] = {// グラフ 1 データマトリックス
    B00000000,// 第一行目から順番に
    B00000011,
    B00000001,
    B00000001,
    B11110001,
    B00001001,
    B00000111,
```

```
B00000000
};
// 8
byte matrixData_num_8[8] = {// グラフ 1 データマトリックス
  B00000000,// 第一行目から順番に
  B01110110,
  B10001001,
  B10001001,
  B10001001,
  B10001001,
  B01110110,
  B00000000
};
// 9
byte matrixData_num_9[8] = {// グラフ 1 データマトリックス
  B00000000,// 第一行目から順番に
  B01001110,
  B10010001,
  B10010001,
  B10010001,
  B10010001,
  B01111110,
  B00000000
};
```

setup( )にて加えます。

```
Serial.begin(115200);
```

loop( )を変更します。

```
void loop ( ) {
```

```
        if(Serial.available())// シリアルポートにデータの受取りするかとうかチェックする
        {
          byte num = Serial.read();
          switch(num)// 各々数値を各別に示す
          {
            case '0':
              Draw(matrixData_num_0);
              break;

            case '1':
              Draw(matrixData_num_1);
              break;

            case '2':
              Draw(matrixData_num_2);
              break;

            case '3':
              Draw(matrixData_num_3);
              break;

            case '4':
              Draw(matrixData_num_4);
              break;

            case '5':
              Draw(matrixData_num_5);
              break;

            case '6':
              Draw(matrixData_num_6);
              break;
```

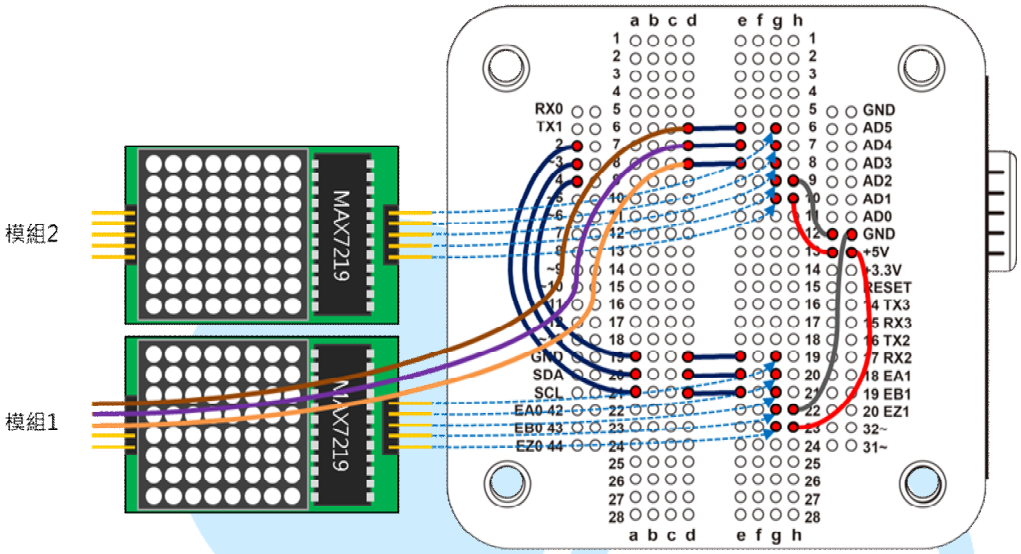
```
            case '7':
                Draw(matrixData_num_7);
                break;

             case '8':
                Draw(matrixData_num_8);
                break;

             case '9':
                Draw(matrixData_num_9);
                break;

             default:
                break;
          }
        }
        delay(100);
    }
```

コンパイルし、アップロードするプログラムを、シリアルモニタを開き、プログラムと同じに設定するノートボーレート、その後は 0 から 9 までの任意の数を入力することができ、86Duino EduCake は LED マトリクスに対応する図形パターンが表示されます接続します。

いくつかの BYTE 行列 matrixData_num_0～matrixData_num_9 は 0 から 9 までのデジタルテキストパターンで追加したプログラムで開始し、ユーザーが数字を入力したときに、このプログラム機能、シリアルモニタ通信に 86Duino EduCake デジタルワードを受信します後、別の文字データ用のスイッチケースを使用して、MAX7219 は LED マトリックス内の対応するテキストパターンを表示します。独自の通信可能な文字を追加して、表示するのです自分の好きなパターンを変更することができます。

五、第四プログラム – マルチ LED マトリックスを制御する 1

以下のような配線をして練習しましょう。



接著請打開 86Duino Coding IDE，在 int CLOCK_pin = 4;之後加入：

```
int MAX7219_units = 2;// MAX7219 使用数量を設定する
```

在 void MAX7219_1Unit( )函式下方加入：

```
    void MAX7219_AllUnit( byte reg_addr, byte reg_data ) {// 全てのMAX2179モジュールに同じくデータを書込みすることを制御する
      digitalWrite( LOAD_pin, LOW );// 転送する前に LOAD ピンを LOW に変更させる
      for (int c =1; c<= MAX7219_units; c++) {
        SPI_SendByte(reg_addr);//設定されたレジスタアドレスを先に送る
        SPI_SendByte(reg_data);// 次にデータを送る
      }
      digitalWrite(LOAD_pin,HIGH);// 転送された後 LOAD ピンを HIGH に変更させる
    }
    void MAX7219_indexUnit( byte unit_index, byte reg_addr, byte reg_data ) {// ある MAX7219 直列回路を制御する
      int c = 0;
      digitalWrite(LOAD_pin, LOW);// 転送する前にLOADピンをLOWに変更させる
      // 直列回路の最尾位モジュールから制御する
      for ( c = MAX7219_units; c > unit_index; c--) {
        SPI_SendByte(0);// NO-OP レジスタ
        SPI_SendByte(0);// データ = 0
      }

      SPI_SendByte(reg_addr);// 設定されたレジスタアドレスを先に送る
      SPI_SendByte(reg_data);// 次にデータを送る

      for ( c = unit_index-1; c >= 1; c--) {
        SPI_SendByte(0);// NO-OP 暫存器
        SPI_SendByte(0);// データ = 0
      }
      digitalWrite(LOAD_pin,HIGH);// 転送された後 LOAD ピンを HIGH に変更させる
    }
```

Draw( )の後に加える：

```
    void Draw_Unit( byte index, byte *LED_matrix )// 特定のユニットグラフを描く
    {
      MAX7219_indexUnit(index, 1, LED_matrix[0]);
      MAX7219_indexUnit(index, 2, LED_matrix[1]);
      MAX7219_indexUnit(index, 3, LED_matrix[2]);
      MAX7219_indexUnit(index, 4, LED_matrix[3]);
      MAX7219_indexUnit(index, 5, LED_matrix[4]);
      MAX7219_indexUnit(index, 6, LED_matrix[5]);
      MAX7219_indexUnit(index, 7, LED_matrix[6]);
      MAX7219_indexUnit(index, 8, LED_matrix[7]);
    }
```

```
    void setup ( ) {
      pinMode(DIN_pin, OUTPUT);
      pinMode(CLOCK_pin, OUTPUT);
      pinMode(LOAD_pin, OUTPUT);

      digitalWrite(CLOCK_pin, HIGH);

      // 全ての MAX7219 レジスターを初期化する
      MAX7219_AllUnit( max7219_REG_scanLimit, 0x07 );//全ての行列をスキャンすることを設定する
      MAX7219_AllUnit( max7219_REG_decodeMode, 0x00 );//デコードモードを閉じる
      MAX7219_AllUnit( max7219_REG_shutdown, 0x01 );//シャットダウンモードを閉じる
      MAX7219_AllUnit( max7219_REG_displayTest, 0x00 ); //不在モードを設定する
```

setup( )の変更内容は

```
    for( int i=1; i<=8; i++ ) {//
まず、全ての LED マトリックスを暗くさせる MAX7219_AllUnit(i,0);
    }

    MAX7219_AllUnit( max7219_REG_intensity, 0x0f );//
ブライト範囲を設定する，0x00 ~ 0x0f

    delay(1000);
}
```

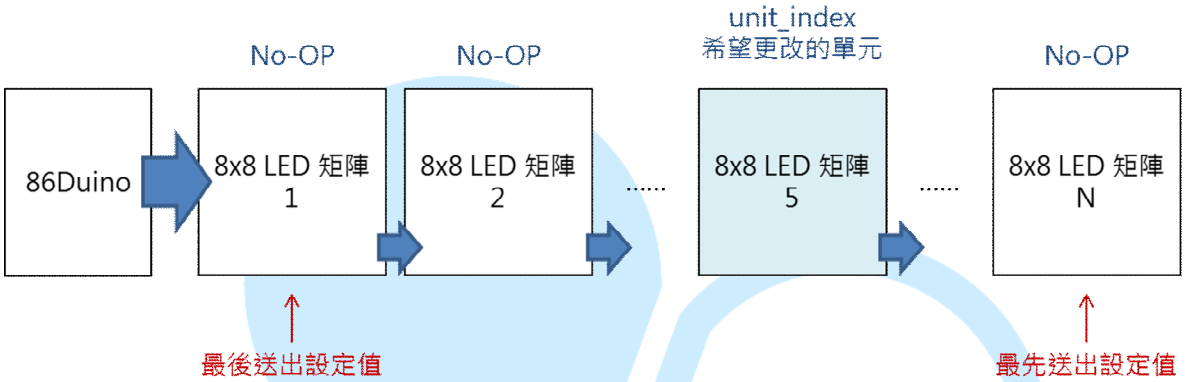
loop( )の変更内容は：

```
void loop () {
  Draw_Unit(1, matrixData_num_0);
  Draw_Unit(2, matrixData_num_1);
  delay(500);
}
```

このサンプル·プログラムは、複数の MAX7219 のモジュール機能のサポートを追加します。「MAX7219_AllUnit(byte reg_addr, byte reg_data)」、「MAX7219_indexUnit(byte unit_index, byte reg_addr, byte reg_data)」、「Draw_Unit(byte index, byte *LED_matrix)」。

前述のように複数のモジュールを直列 MAX7219 およびモジュールの DIN ピンに接続することができるので、モジュール MAX7219 複数の、それがそれぞれのために必要であるように、データは、直列の次のモジュールに DOUT に渡される前にジョブのレジスタ設定のためのモジュール。したがって、この値を設定し、実際に使用するユーザを調節するように必要なモジュールの数に応じて、MAX7219 シリーズを使用して、モジュール数を設定するための新しい変数の整数 MAX7219_units 前面を宣言、ここでは、2 つのモジュールを使用するため、2 に設定されています。MAX7219_AllUnit ( ) LOAD_pin 各 MAX7219 のためのループの使用中旬以降 LOW に設定ように、この機能は、基本的にはほとんどが、ために各 MAX7219 ユニットに対して同じ情報を記述する必要のオリジナル "MAX7219_1Unit ( ) を持つプロセスである同じ値

を書き込むためのレジスタで、その後 LOAD_pin は HIGH にセット。MAX7219_indexUnit とは前の関数 MAX7219 ユニットに「異なるデータ」で設定された。指定されたモジュールを変更したいとの複数バイト unit_index の関数の引数、データを設定する予定が送信されます次のモジュールに渡されるので、N 回送信するために必要なモジュールの数 N が存在する場合、以下に示すように情報を参考ください。



unit_index だけで指定されたモジュールスクラッチパッドデータを変更する必要がありますが、残りはすべきで、それを行う方法を、これの変更ではないですか？「何もしない"スクラッチパッドの前に MAX7219 のレジスタテーブルは便利です。グラフが示すように N シリーズモジュールの最初の 5 つのモジュールを変更する必要があり、最初の 1～4 及び第 6～N 個のモジュールはノーオペレーションに値を書き込むことではなく、ノーオペレーションを書きません必要とされる上記本機の動作に影響を与えます。MAX7219_indexUnit ( ) 関数は、これを実行する過程にある。すべての情報を書くことが HIGH になった後、真ん中には変化がなく、MAX7219_AllUnit (LOW) のように記述されています。LOAD_pin 前と同じで注意してください。Draw_Unit (バイトインデックス、バイト* LED_matrix) この関数は、特定のモジュール内にパターンを描画することですが、他のモジュールのパターンを変更しません。索引セットに対応する MAX7219 のため MAX7219_indexUnit ( ) 関数が使用し、機能を達成するために行うことができます。

セットアップが ( ) そのように直接、元 MAX7219_1Unit に同じ初期化を行う必要とされるため、N 個のモジュールで内部の一部を初期化する代わりに MAX7219_AllUnit ( ) に Draw_Unit (1、matrixData_num_0) の内部で使用されるループ ( ) 、Draw_Unit (2、matrixData_num_1) 、第一のモジュー

ルと直列にパターン番号 0、パターン番号 1 に示された。最初の二つのモジュールを描画します。読者はまた、独自のコンテンツのパターンを変更することができ、セル位置は他の効果を試してプロットました。

三、 第五プログラム – マルチ LED マトリックスを制御する 2

最後のプログラムは、ダイナミックなマーキー効果を追加し、最初の 4 つのプログラムから直接に配線を変更しないでください。

```
// 86Duino EduCake
const unsigned int string_len = 70;
byte matrixData_86Duino_EduCake[string_len] = {//
データマトリクス
  B01110110,// 8
  B10001001,
  B10001001,
  B01110110,
  B00000000,

  B01111110,// 6
  B10001001,
```

```
B10001001,
B01110010,
B00000000,

B11111111,// D
B10000001,
B10000001,
B01111110,
B00000000,

B01111000,// u
B10000000,
B01000000,
B11111000,
B00000000,

B11111010,// i
B00000000,

B11111000,// n
B00010000,
B00001000,
B11110000,
B00000000,

B01110000,// o
B10001000,
B10001000,
B01110000,
B00000000,

B00000000,
```

```
  B11111111,// E
  B10001001,
  B10001001,
  B10000001,
  B00000000,

  B01110000,// d
  B10001000,
  B10001000,
  B11111111,
  B00000000,

  B01111000,// u
  B10000000,
  B01000000,
  B11111000,
  B00000000,

  B01111110,// C
  B10000001,
  B10000001,
  B01100110,
  B00000000,

  B01100100,// a
  B10010100,
  B10010100,
  B11111000,
  B00000000,

  B11111111,// k
  B00100000,
  B01010000,
```

```
    B10001000,
      B00000000,

      B01110000,// e
      B10101000,
      B10101000,
      B10110000,

      B00000000,
      B00000000,
      B00000000
    };

    int shift = 0;
    void ShiftDraw_2Unit(byte *LED_matrix)// シフトで全画面を描く
    {
      // Unit 1
      MAX7219_indexUnit(1, 1, LED_matrix[(shift) % string_len]);//
第 1 行目のデータを作る
      MAX7219_indexUnit(1, 2, LED_matrix[(shift+1) % string_len]);//
第 2 行目のデータを作る
      MAX7219_indexUnit(1, 3, LED_matrix[(shift+2) % string_len]);//
第 3 行目のデータを作る
      MAX7219_indexUnit(1, 4, LED_matrix[(shift+3) % string_len]);//
第 4 行目のデータを作る
      MAX7219_indexUnit(1, 5, LED_matrix[(shift+4) % string_len]);//
第 5 行目のデータを作る
      MAX7219_indexUnit(1, 6, LED_matrix[(shift+5) % string_len]);//
第 6 行目のデータを作る
      MAX7219_indexUnit(1, 7, LED_matrix[(shift+6) % string_len]);//
第 7 行目のデータを作る
      MAX7219_indexUnit(1, 8, LED_matrix[(shift+7) % string_len]);//
第 8 行目のデータを作る
```

```
        // Unit 2
          MAX7219_indexUnit(2,      1,      LED_matrix[(shift+8)      %
string_len]);//第 1 行目のデータを作る
          MAX7219_indexUnit(2,      2,      LED_matrix[(shift+9)      %
string_len]);// 第 2 行目のデータを作る
          MAX7219_indexUnit(2,      3,      LED_matrix[(shift+10)     %
string_len]);// 第 3 行目のデータを作る
       MAX7219_indexUnit(2,      4,      LED_matrix[(shift+11)     %
string_len]);// 第 4 行目のデータを作る
          MAX7219_indexUnit(2,      5,      LED_matrix[(shift+12)     %
string_len]);// 第 5 行目のデータを作る
          MAX7219_indexUnit(2,      6,      LED_matrix[(shift+13)     %
string_len]);// 第 6 行目のデータを作る
          MAX7219_indexUnit(2,      7,      LED_matrix[(shift+14)     %
string_len]);// 第 7 行目のデータを作る
          MAX7219_indexUnit(2,      8,      LED_matrix[(shift+15)     %
string_len]);// 第 8 行目のデータを作る
          shift++;
          if(shift>=string_len){// INDEX を 0～61 にて循環させる
            shift = 0;
          }
        }
```

loop ( )を変更する：

```
        void loop ( )
          ShiftDraw_2Unit(matrixData_86Duino_EduCake);
          delay(100);
        }
```

アップロードプログラムが完了すると、単に一般的な広告マーキー効果として、 "86Duino EduCake“ノンストップサイクル”を見ることができます。ShiftDraw_2Unit ( ) この関数は、基本的には ShiftDraw ( ) ほとんどが、そ

のマトリクス状バイト可変長として使用されているが、このパターンマトリクスの例 70 の長さの合計なので、関数内のインデックス変数のみ 0～ 69 内側のループ、const の unsigned int 型 string_len 書いて、プログラムを修正するために、ここでは、パターン行列の列の長さとして使用され簡単に。読者はより多くの MAX7219 モジュールがある場合、あなたも同じコンセプトでそれを行うことができますこんにちはを拡張します。