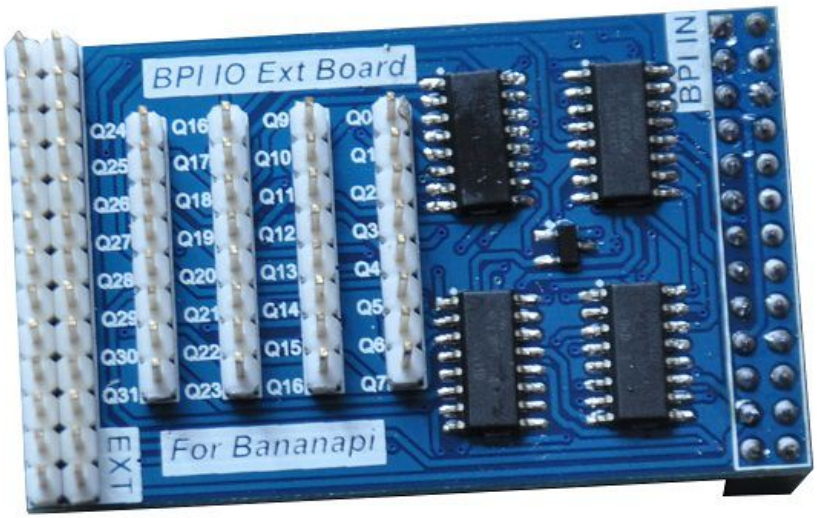
Infinity cascade IO expand module



Infinity cascade, infinity IO

Contact Us

SINOVOIP CO., LIMITED

Company Add: 5/F， Comprehensive Building of Zhongxing industryCity,Chuangye Road,Nanshan District, Shenzhen,Guangdong,China

judyhuang@banana-pi.com

www.banana-pi.com

Version： 1.0

Data： 2014.7.25

Shenzhen SINOVOIP CO., LIMITED Copyright Statement：

The document only descript the information about product however. It can not be guarantee the product function and performance. If the document content or the product feature and tech spec included document be changed, it will be not further notice.

Content in the document might past due. Our cooperation can not promise update these information.

Some information in the document might disable in your local area, that include product and service. You can consult with contact and agency in your local area.

Copyright to the document belong to Shenzhen SINOVOIP CO.,LIMITED. User only can use these content after he or she get authorization from our company or other obligee. However user can not copy paraphrase or creative similar device or product.

The document final right to interpret and document be used final interpret belong Shenzhen SINOVOIP CO.,LIMITED.

More information:
Get more product and support please contact Shenzhen SINOVOIP CO.,LIMITED (www.banana-pi.com)

Attention:

Due to technical requirements components, please do not hand directly connected
Touch. Core board and development system contains static-sensitive devices. Quiet Electrical charge easily accumulate in the human body and the device can not detect possible Damage to equipment, it is recommended to take anti-static measures, it is recommended not to hand. Touch, stored in anti-static effect devices.

Banana Pi Expand Module Serial：

Infinity cascade IO expand module：
This module is designed specifically for the Banana Pi IO expansion modules. The module expand 32 IO, Multiple modules can cascade, infinity cascade， infinity GPIO.

I2C GPIO expand module：
This module is designed specifically for the Banana Pi IO expansion modules. The Module use I2C bus to connect to Banana Pi. The module expand 8 Bidirectional GPIO and wit isolation protection function which can effectively excessive external voltage. There are 8 I2C address, you can choose one of them through setup the jumper. Multiple modules can cascade and maximum cascade 8 modules!

Prototype development module：
The Prototype development module is designed specifically for the Banana Pi. The module suitable enthusiasts and user can weld peripheral to the module； The module expand some amphenol connector and some SMT, so the user can finish prototype test easily.

Berryclip expand module：
The BerryClip module is designed specifically for learning how to use the GPIO of Banana Pi. There are 6 multiple color LED, 1 button and 1 Buzzer on the module.

Berryclip(DIY) expand module：
The module is not the end product, you need weld them by yourself. The function of the module is the same as BerryClip module.

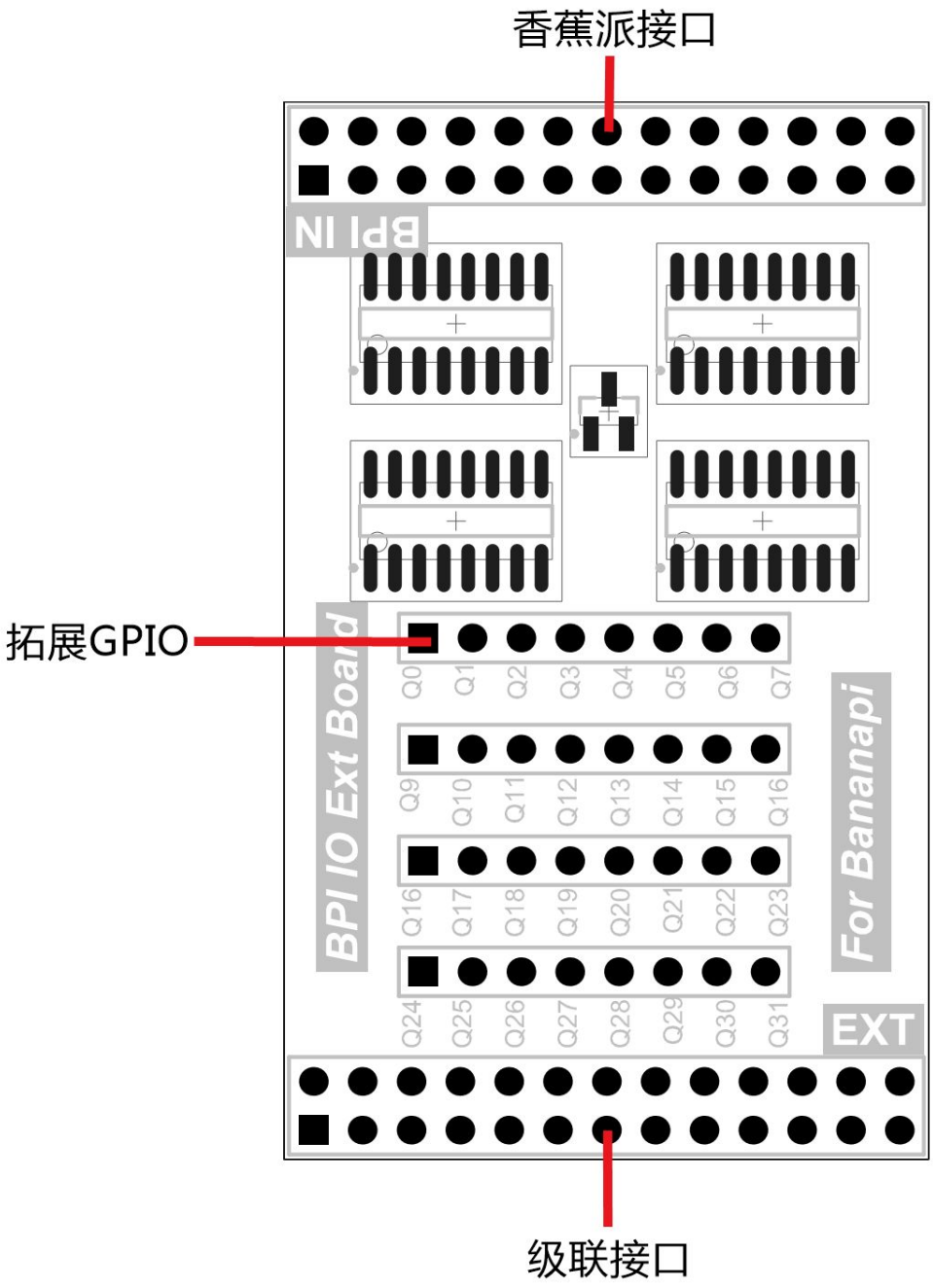
UNO compatibility module：
The module makes Banana Pi compatible with Arduino Uno and many Arduino Shields. The module's GPIO is the same as Arduino Uno and you can choose the voltage of GPIO between 5V or 3V through setup jumper.

T Electric level convert module：
The module expand the GPIO of Banana Pi to breadboard. It convert 3.3V electric to 5V electric level, then the Banana Pi can connect many 5V electric level peripheral.

IO extraction module：

The module expand all of GPIO of Banana Pi to breadboard.

Product Specification：



Product Overview：

This module is designed specifically for the Banana Pi IO expansion modules, which can effectively solve the banana send IO port insufficient. Module uses four 74HC595 chips to expand 32 IO ports. As shown above, the top of the module will expand the IO of Banana Pi again that can be cascaded for more IO expansion modules can theoretically unlimited expansion.

Product Features：

- Expanded 32 GPIO
- Infinity connection for the same module
- Use wiringPi API ,sample code

Port：

- Banana Pi 2X13 port
- Banana Pi 2X13 cascade port
- Q0-Q31 expanded IO port

Product Parameters：

- Working voltage: 2.4V-5V
- IO voltage: 3.3V
- Expanded 32 unidirectional IO
- Connection through SPI
- 100 MHz (typical) shift out frequency
- 8-bit serial input
- 8-bit serial or parallel output
- Specified from -40C to +85C and from -40C to +125C

Typical Application：

- Drive the lattice screen
- Driver numeric display
- Drive matrix LED

Method Of Use：

Insert the module that the silk screen says “BPI IN”. Pay attention don’t make the direction reversed! The correct direction of insert module is above the Banana Pi’s PCB; The 8*4header in the module are Q0-Q31 expanded GPIO, they can connect to the peripheral through the Dupont Line. The header which near silk screen write ”EXT” expand GPIO of Banana Pi, user can cascade the other module or the same module.

More information：

The 74HC595; 74HCT595 are high-speed Si-gate CMOS devices and are pin compatible with Low-power Schottky TTL (LSTTL). They are specified in compliance with JEDEC standard No. 7A.

The 74HC595; 74HCT595 are 8-stage serial shift registers with a storage register and 3-state outputs. The registers have separate clocks.

Data is shifted on the positive-going transitions of the shift register clock input (SHCP).

The data in each register is transferred to the storage register on a positive-going transition of the storage register clock input (STCP). If both clocks are connected together, the shift register will always be one clock pulse ahead of the storage register.

The shift register has a serial input (DS) and a serial standard output (Q7S) for cascading. It is also provided with asynchronous reset (active LOW) for all 8 shift register stages. The storage register has 8 parallel 3-state bus driver outputs. Data in the storage register appears at the output whenever the output enable input (OE) is LOW.

Function table：

Control				Input	Output		Function
SHCP	STCP	OE	MR	DS	Q7S	Qn	
X	X	L	L	X	L	NC	a LOW-level on MRonly affects the shift registers
X	↑	L	L	X	L	L	empty shift register loaded into storage register
X	X	H	L	X	L	Z	shift register clear; parallel outputs in high-impedance OFF-state
↑	X	L	H	H	Q6S	NC	logic HIGH-level shifted into shift register stage 0. Contents of all shift register stages shifted through, e.g. previous state of stage 6 (internal Q6S) appears on the serial output (Q7S).
X	↑	L	H	X	NC	QnS	contents of shift register stages (internal QnS) are transferred to the storage register andparallel output stages
↑	↑	L	H	X	Q6S	QnS	contents of shift register shifted through; previous contents of the shift register is transferred to the storage register and the parallel output stages

H = HIGH voltage state;
L = LOW voltage state;
↑ = LOW-to-HIGH transition;
X = don’t care;
NC = no change;
Z = high-impedance OFF-state.

Timing diagram：

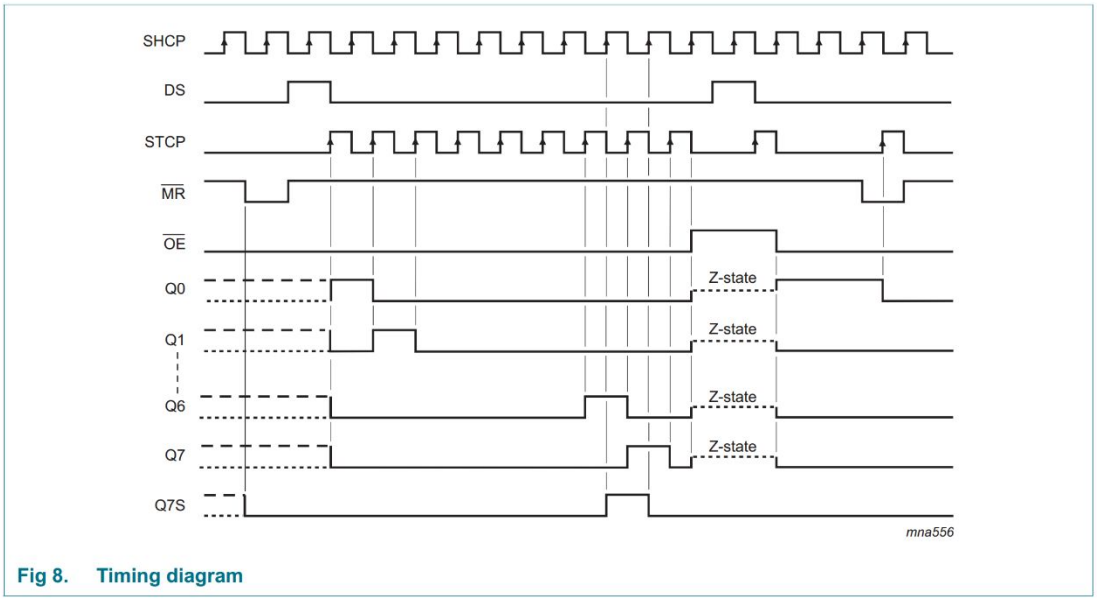
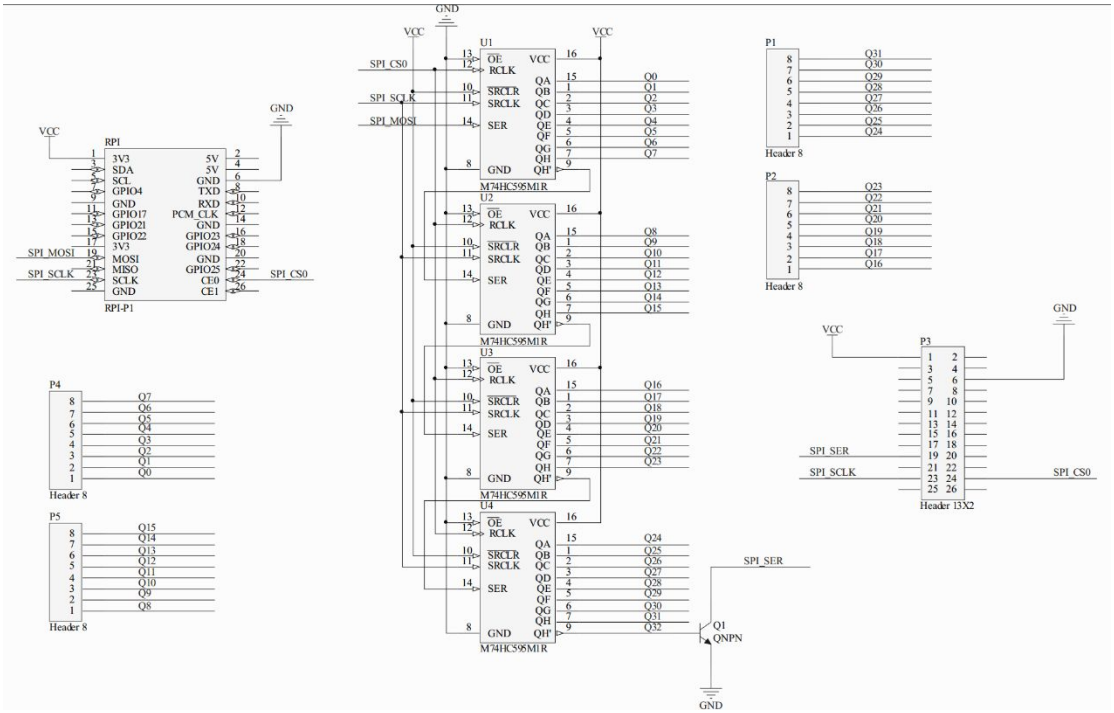


Fig 8. Timing diagram

Schematic diagram：



More information please check：

http://www.nxp.com/products/logic/shift_registers/series/74HC_T_595.html

Example and test code:

```
#include <wiringPi.h>
#include <stdio.h>
 int SER   = 12;
 int RCLK  = 10;
 int SRCLK = 14;
 unsigned char LED[8]={0x01, 0x02, 0x04, 0x08, 0x10, 0x20, 0x40, 0x80};
 void SIPO(unsigned char byte);
 void pulse(int pin);
 void init() {
    pinMode(SER, OUTPUT);
    pinMode(RCLK, OUTPUT);
    pinMode(SRCLK, OUTPUT);
    digitalWrite(SER, 0);
    digitalWrite(SRCLK, 0);
    digitalWrite(RCLK, 0);
 }
 void delayMS(int x) {
   usleep(x * 1000);
 }
 int main (void)
 {
    if (-1 == wiringPiSetup()) {
        printf("Setup wiringPi failed!");
        return 1;
    }
    init();
    int i;
    while(1) {
      for(i = 0; i < 8; i++)
      {
       SIPO(LED[i]);
       pulse(RCLK);
       delayMS(50);
       printf(" i = %d", i);
      }
      printf("\n");
      delayMS(500); // 500 ms
```

```
      for(i = 7; i >= 0; i--)
      {
       SIPO(LED[i]);
       pulse(RCLK);
       delayMS(50);
       printf(" i = %d", i);
      }
      delayMS(500); // 500 ms
    }
    usleep(1000);
    digitalWrite(RCLK, 1);
}
void SIPO(unsigned char byte)
{
    int i;
    for (i=0;i<8;i++)
    {
        digitalWrite(SER,((byte & (0x80 >> i)) > 0));
        pulse(SRCLK);
    }
}
void pulse(int pin)
{
    digitalWrite(pin, 1);
    digitalWrite(pin, 0);
}
```