

Keys Input Demo

You will learn how to use GPIO as the input by this demo. As shown in Figure 1, you can use the four keys KEY1~KEY4 on the extension board to control four LEDs on/off.

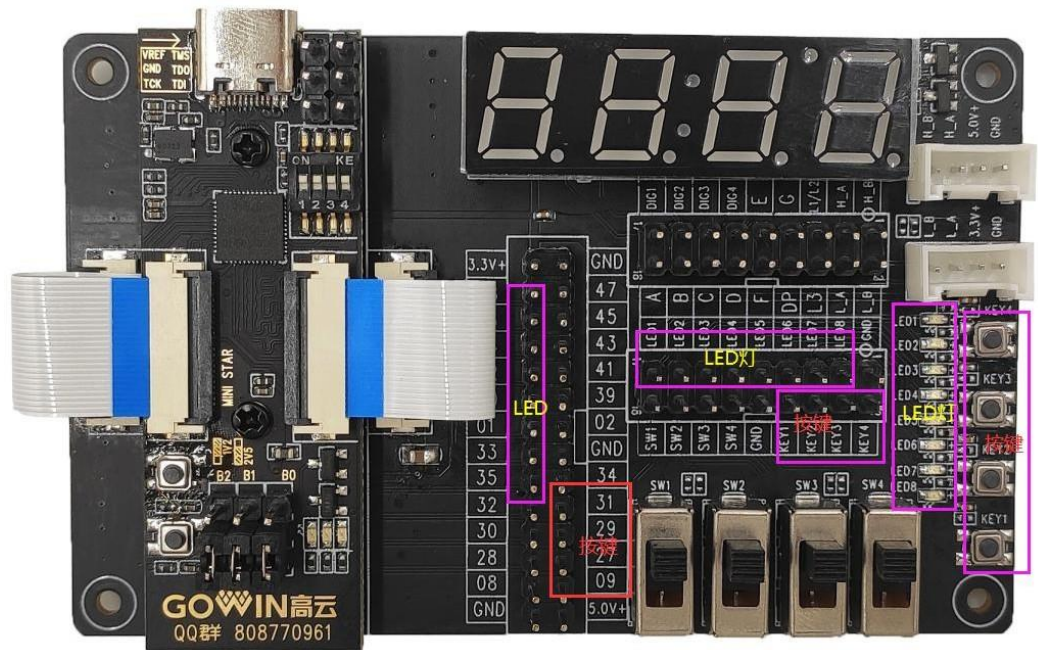


Figure 1 Mini-Star Development Board (GW1NSR-LV4CQN48P) and Extension

This demo introduction includes four parts:

1. GPIO Introduction
2. Hardware Design
3. Software Design
4. Download and Verification

GPIO Introduction

Gowin GW1NSR-LV4CQN48P is used as the platform in this demo. From [DS861, GW1NSR series of FPGA Products Datasheet](#), you can learn the GPIO module and know there is a Cortex-M3 RISC embedded in this chip. The key to this demo is how to control the GPIO as the output. The design information listed below is excerpted from the datasheet.

The SoC microprocessor system communicates with the GPIO block through the AHB bus. The GPIO block interconnects with the FPGA. GPIO provides a 16 bits I/O interface with the following properties:

- Programmable interrupt generation capability. You can configure each bit of the I/O pins to generate interrupts;
- Bit masking support using address values;
- Registers for alternate function switching with pin multiplexing support;
- Thread safe operation by providing separate set and clear addresses for control registers.

The GPIO register is as shown in Table 3-21. The GPIO base address is 0x40010000.

The following table lists GPIO registers.

Table 3-21 GPIO Register

| Name | Base Offset | Type | Data Width | Reset Value | Description |
|------------------------|-------------|------------|------------|-------------|---|
| DATA | 0x0000 | Read/Write | 16 | 0x---- | Data value [15:0] |
| DATAOUT | 0x0004 | Read/Write | 16 | 0x0000 | Data output register value [15:0] |
| OUTENSET | 0x0010 | Read/Write | 16 | 0x0000 | Output enable set [15:0] Write 1: Set the output enable bit. Write 0: No effect. Read 1: Indicates the signal direction as output. Read 0: Indicates the signal direction as input. |
| OUTENCLR | 0x0014 | Read/Write | 16 | 0x0000 | Output enable clear [15:0] |
| ALTFUNCSET | 0x0018 | Read/Write | 16 | 0x0000 | Alternative function set [15:0] Write 1: Sets the ALTFUNC bit. Write 0: No effect. Read 0: GPIO as I/O. Read 1: ALTFUNC Function |
| ALTFUNCCLR | 0x001C | Read/Write | 16 | 0x0000 | Alternative function clear [15:0] |
| INTENSET | 0x0020 | Read/Write | 16 | 0x0000 | Interrupt enable set [15:0] Write 1: Sets the enable bit. Write 0: No effect. Read 0: Interrupt disabled. Read 1: Interrupt enabled. |
| INTENCLR | 0x0024 | Read/Write | 16 | 0x0000 | Interrupt enable clear [15:0] Write 1: Clear the enable bit. Write 0: No effect. Read 0: Interrupt disabled. Read 1: Interrupt enabled. |
| INTYPESET | 0x0028 | Read/Write | 16 | 0x0000 | Interrupt type set [15:0] |
| INTYPECLR | 0x002C | Read/Write | 16 | 0x0000 | Interrupt type clear [15:0] |
| INTPOLSET | 0x0030 | Read/Write | 16 | 0x0000 | Polarity-level, edge interrupt request configuration [15:0] |
| INTPOLCLR | 0x0034 | Read/Write | 16 | 0x0000 | Polarity-level, edge interrupt request configuration [15:0] |
| INTSTATUS/ INTCLEAR | 0x0038 | Read/Write | 16 | 0x0000 | Read interrupt status register Write 1: Clear the interrupt request |

Note!

For the details, you can see 3.11.10 GPIO in [DS861, GW1NSR series of FPGA Products Datasheet](#).

Hardware Design

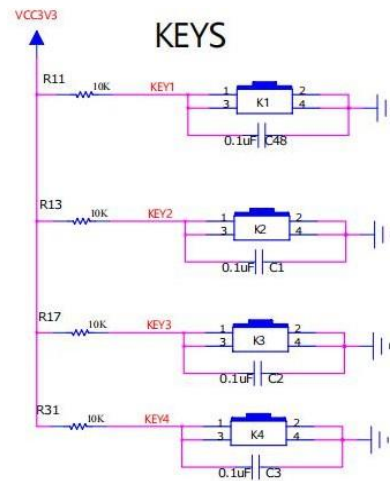


Figure 2 KEY Schematic

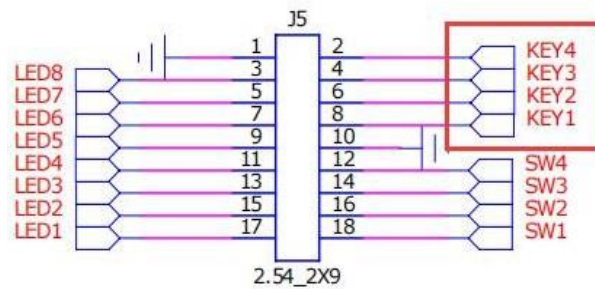


Figure 3 KEY Signal Pins

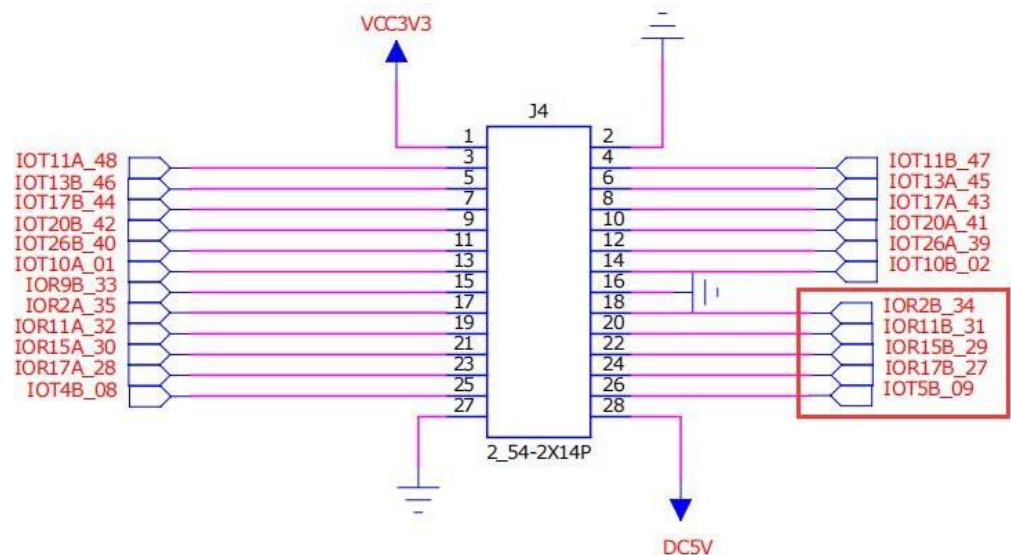


Figure 4 FPGA Pins

In the Figure 2, one end of the KEY is connected to the power supply via the current-limiting resistor, and the other end is connected to the pins in Figure 3 (low level when the key pressed; high level when the key released). Use DuPont wire to connect KEY1~KEY4 signals to FPGA pins such as 20~26 in Figure 4 (pins 1/2/8/9/47/48 have special purpose and are not recommended). The connection of KEY and FPGA is shown in

Figure 5.

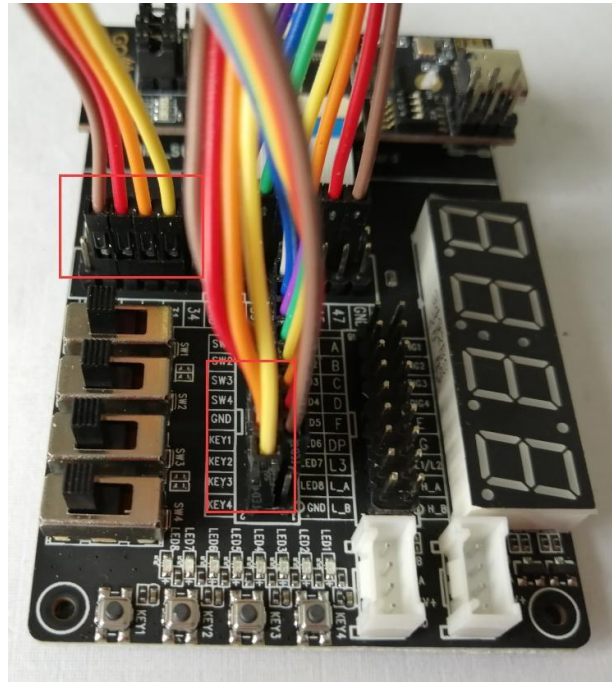


Figure 5 Connection of KEY and FPGA

Software Design

The software design includes two parts: FPGA internal hardware logic and Cortex-M3 software control code, which can be modified on the basis of the led project.

FPGA Internal Logic Design

This HDL code does not need to be modified, and you only need to add the key pin connection. Click "Process" and then select "FloorPlanner" in Figure 6 to open I/O constraints as shown in Figure 7.

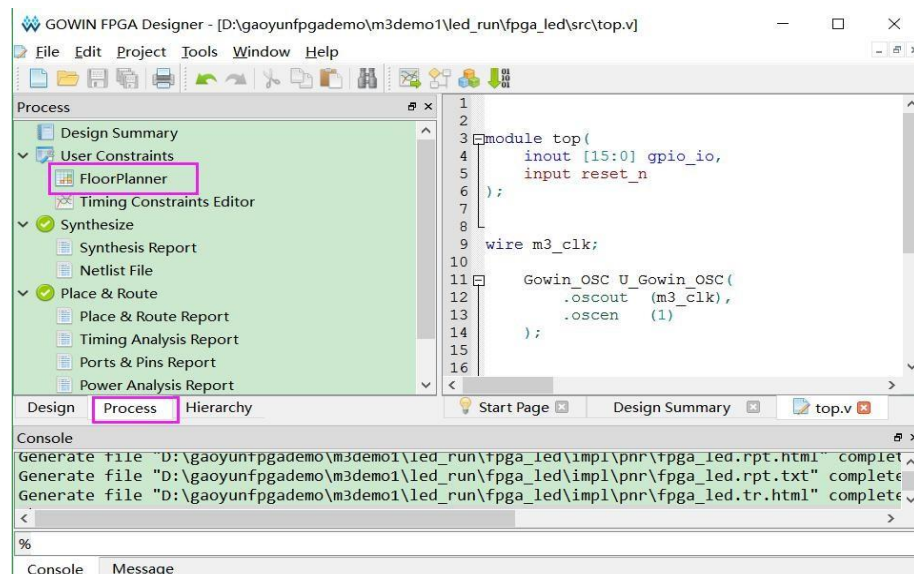


Figure 6 Process View

Click on steps 1-5 on the interface in Figure 7 to set LED, the FPGA pin number and power supply voltage of the KEY according to the connection in Figure 3 and Figure 4, and save the settings when finished.

| Port | Direction | Diff Pair | Location | Bank | Exclusive | IO Type | Drive | Pull Mode | PCI Clamp | Hysteresis | Open Drain | Slew |
|-------------|-----------|-----------|---------------|------|-----------|------------|-------|-----------|-----------|------------|------------|------|
| gpio_io[0] | inout | | 35 | 2 | False | LVC MOS533 | 8 | UP | N/A | NONE | OFF | FA |
| gpio_io[10] | inout | KEY1-2 | 30 | 2 | False | LVC MOS533 | 8 | UP | N/A | NONE | OFF | FA |
| gpio_io[11] | inout | | 34 | 2 | False | LVC MOS533 | 8 | UP | N/A | NONE | OFF | FA |
| gpio_io[12] | inout | | drag or ty... | | False | LVC MOS18 | 8 | UP | N/A | NONE | OFF | FA |
| gpio_io[13] | inout | | drag or ty... | | False | LVC MOS18 | 8 | UP | N/A | NONE | OFF | FA |
| gpio_io[14] | inout | | drag or ty... | | False | LVC MOS18 | 8 | UP | N/A | NONE | OFF | FA |
| gpio_io[15] | inout | | drag or ty... | | False | LVC MOS18 | 8 | UP | N/A | NONE | OFF | FA |
| gpio_io[1] | inout | | 43 | 1 | False | LVC MOS533 | 8 | UP | N/A | NONE | OFF | FA |
| gpio_io[2] | inout | | 33 | 2 | False | LVC MOS533 | 8 | UP | N/A | NONE | OFF | FA |
| gpio_io[3] | inout | | 40 | 1 | False | LVC MOS533 | 8 | UP | N/A | NONE | OFF | FA |
| gpio_io[4] | inout | LED1-8 | 42 | 1 | False | LVC MOS533 | 8 | UP | N/A | NONE | OFF | FA |
| gpio_io[5] | inout | | 44 | 1 | False | LVC MOS533 | 8 | UP | N/A | NONE | OFF | FA |
| gpio_io[6] | inout | | 46 | 1 | False | LVC MOS533 | 8 | UP | N/A | NONE | OFF | FA |
| gpio_io[7] | inout | | 45 | 1 | False | LVC MOS533 | 8 | UP | N/A | NONE | OFF | FA |
| gpio_io[8] | inout | | 31 | 2 | False | LVC MOS533 | 8 | UP | N/A | NONE | OFF | FA |
| gpio_io[9] | inout | KEY3-4 | 29 | 2 | False | LVC MOS533 | 8 | UP | N/A | NONE | OFF | FA |
| reset_n | input | | 20 | 3 | False | LVC MOS18 | N/A | UP | N/A | NONE | N/A | N |

Figure 7 FPGA I/O Constraints

Then click "Synthesize" and "Place & Route" to generate the logic file fpga_led.fs.

Note!

LED1-LED8 corresponds to GPIO[7:0]; KEY1-4 corresponds to GPIO[11:8].

Cortex-M3 Software Control Design

You can modify this design on the basis of the led project. Open Led.uvprojx in the Keil_led\PROJECT folder.

1. Set GPIO register; set GPIO[7: 0] as the output and GPIO[15:8] as the input.

```
GPIO0->OUTENSET = 0x00ff; //IO[15:8]: input IO[7:0]: output
```

2. By setting the GPIO->DATAOUT output data register, you can realize LEDs blinking one by one to test the configuration.

First, write a sub-function led_1() to light up LED1-8 one by one, as shown in Figure 8. Then write a sub-function led_off() to extinguish all LEDs, as shown in Figure 9.

```

47 //led on one by one
48 void led_1()
49 {
50     s = 0xffff; //led: off
51     GPIO0->DATAOUT = s;
52     Delay(833300);
53
54     for(i=0;i<8;i++)
55     {
56         s=s<<1; //led: one bye one on
57         GPIO0->DATAOUT = s;
58         Delay(833300);
59     }
60 }

```

Figure 8 LED Blinks One by One

```

31 //led all off
32 void led_off()
33 {
34     s = 0xffff; //led: off
35     GPIO0->DATAOUT = s;
36     Delay(833300);
37 }

```

Figure 9 LEDs All Off

3. The KEY operation is obtained by reading GPIO->DATA input data register. The GPIO->DATA register is defined as follows.

| Name | Base Offset | Type | Data Width | Reset Value | Description |
|------|-------------|------------|------------|-------------|-------------------|
| DATA | 0x0000 | Read/Write | 16 | 0x--- | Data value [15:0] |

When KEY1 is pressed, it is in low level, that is, the GPIO[11]=0, and the other bits are unknown, marked by x, GPIO->DATA=xxxx_0xxx_xxxx_xxxx; as the LED corresponds to GPIO[7:0], shift GPIO->DATA right by 4 bits to get the result 0000_xxxx_0xxx_xxxx, which will be performed an OR operation 1111_1111_0111_1111, and the result is 0xff7f; send this result to GPIO->DATAOUT output data register, that is, light up a LED corresponding to GPIO[7].

```

key1= (GPIO0->DATA >>4 ) | 0xff7f;
GPIO0->DATAOUT = key1;
Delay(1000);

```

When KEY2 is pressed, it is in low level, that is, the GPIO[10]=0, and the other bits are unknown, marked by x, GPIO->DATA=xxxx_x0xx_xxxx_xxxx; as the LED corresponds to GPIO[7:0], shift GPIO->DATA right by 4 bits to get the result 0000_xxxx_x0xx_xxxx, which will be performed an OR operation 1111_1111_1011_1111, and the result is 0xffbf; send this result to GPIO->DATAOUT output data register, that is, light up a LED corresponding to GPIO[6].

```
key2= (GPIO0->DATA >>4 ) | 0xffbf;
GPIO0->DATAOUT = key2;
Delay(1000);
```

When KEY3 is pressed, it is in low level, that is, the GPIO[9]=0, and the other bits are unknown, marked by x, GPIO->DATA=xxxx_xx0x_xxxx_xxxx; as the LED corresponds to GPIO[7:0], shift GPIO->DATA right by 4 bits to get the result 0000_xxxx_xx0x_xxxx, which will be performed an OR operation 1111_1111_1101_1111, and the result is 0xffdf; send this result to GPIO->DATAOUT output data register, that is, light up a LED corresponding to GPIO[5].

```
key3= (GPIO0->DATA >>4 ) | 0xffdf;
GPIO0->DATAOUT = key3;
Delay(1000);
```

When KEY4 is pressed, it is in low level, that is, the GPIO[8]=0, and the other bits are unknown, marked by x, GPIO->DATA=xxxx_xxx0_xxxx_xxxx; as the LED corresponds to GPIO[7:0], shift GPIO->DATA right by 4 bits to get the result 0000_xxxx_xxx0_xxxx, which will be performed an OR operation 1111_1111_1110_1111, and the result is 0xffef; send this result to GPIO->DATAOUT output data register, that is, light up a LED corresponding to GPIO[4].

```
key4= (GPIO0->DATA >>4 ) | 0xffef;
GPIO0->DATAOUT = key4;
Delay(1000);
```

4. After build, the download file led.bin is generated.

Download and Verification

Use Gowin Software to download, and the running is as shown in Figure 10. The FPGA hardware platform file is fpga_led.fs, and the Cortex-M3 software file is led.bin, so be careful to choose the correct file path and build file.

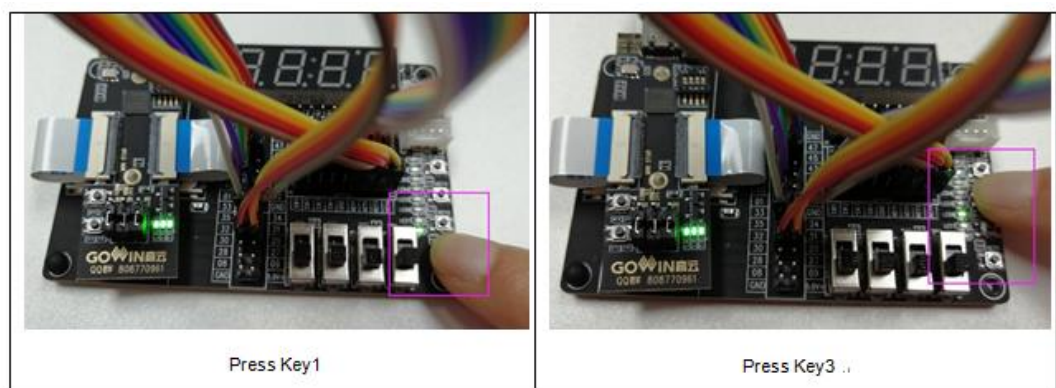


Figure 10 Demo Running