RT568 I/O Re-mapping

V1.0B

JUN 21, 2020

Rafael Micro

8F., No.28, Chenggong 12th St., Zhubei City, Hsinchu County 30264, Taiwan R.O.C.

30264 新竹縣竹北市成功十二街28號8樓

Tel: 886-3-5506258 Fax: 886-3-5506228 www.rafaelmicro.com

Revision History

Revision	Release Date	Author	Summary
1.0	2020/06/21	Ryan	Initial version
1.0B	2020/06/29	Ryan	Chapter 1 description supplement

Abstract

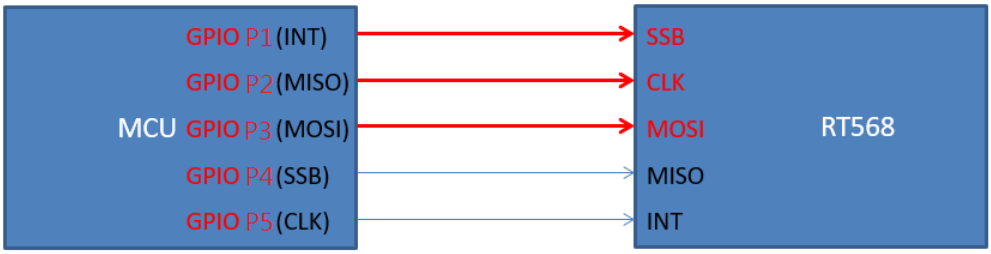
In order to make SiP (System in a Package) easier, users can dynamically configure SPI pins through software.
This file explains how to configure SPI pins dynamically.

1. Brief description of purpose

The purpose of this document can be illustrated by the following diagram, there are many kinds of SPI sorting, we use one example to explain.
The initial definition of the MCU pins is as follows:

MCU GPIO mode	MCU SPI mode	RT568
GPIO P1	INT	SSB
GPIO P2	MISO	CLK
GPIO P3	MOSI	MOSI
GPIO P4	SSB	MISO
GPIO P5	CLK	INT

*In GPIO mode, P1 to P5 represent the names of GPIO pins, and the names may be different in different types of MCUs.

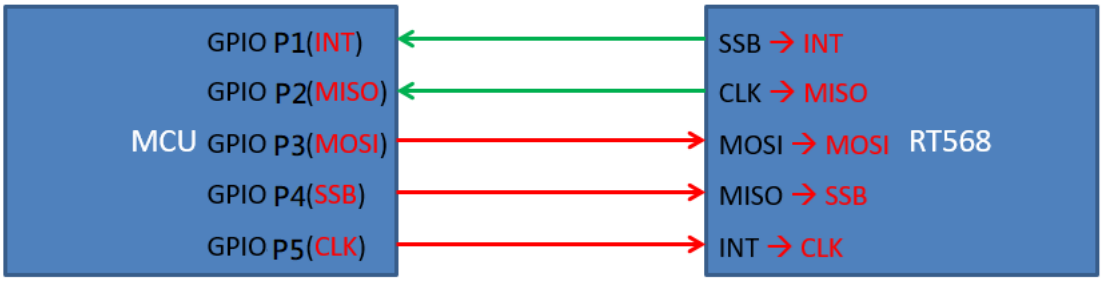


Before I/O remapping

We have observed that the pin definitions do not match, so the user must change the RT568 pin definitions. The RT568 SPI pin definition can be exchanged by changing the register.

Engineers can use the MCU GPO function to combine SPI commands and write to the RT568 register that controls the SPI pin sequence.

When the correct SPI pin sequence is written to RT568, the MCU can be switched from GPIO mode to SPI mode.



After I/O remapping

2. RT568 register configuration

The engineer must know the pin definition registers (GPIO0~GPIO4) of the SiP IC before operating I/O-remapping, the RT568 register configuration is as follows.

Control Registers

Register Address	Name	Description	Power on Value
Reg_248	Bit[2:0] : GPIO0_SEL Bit[5:3] : GPIO1_SEL Bit[7:6] : GOIO2_SEL[1:0]	Refer to GPIO table EN_SDO_INT_Out, 1: enable SDO and INT function pin output 0: Keep SDO and INT function pin input	GPIO0_SEL = 0 GPIO1_SEL = 1 GPIO2_SEL = 2 GPIO3_SEL = 3 GPIO4_SEL = 4 EN_SDO_INT_Out = 0
Reg_249	Bit[0] : GPIO2_SEL[2] Bit[3:1] : GPIO3_SEL Bit[6:4] : GPIO4_SEL Bit[7] : EN_SDO_INT_Out		

GPIO table

RF pin	GPIOx_Sel[2:0]								
	0	1	2	3	4	5	6	7	Initial value (power on)
GPIO 0	SPI_CS	SPI_CLK	SPI_MOSI	SPI_MISO	INT	N/A			SPI_CS
GPIO 1									SPI_CLK
GPIO 2									SPI_MOSI
GPIO 3									SPI_MISO
GPIO 4									INT

3. RT568 I/O re-mapping operation flow

Please follow the steps below to complete I/O re-mapping
[Engineers can also refer to the RF_SpiIoMapping() function.]

*If you cannot find the RF_SpiIoMapping() function, please refer to Appendix 1.

a.

After power on, RT568 RF SPI pins are all set to input mode.

b.

MCU set SPI pins as GPIO type and output mode(GPIO0 ~ GPIO4).
use spiGpioWriteReg() to configure RF SPI pins.
[The spiGpioWriteReg function is in the proting_misc.c file]

*If you cannot find the spiGpioWriteReg() and spiGpioDelay() function, please refer to Appendix 2.

* Write RF Reg_248, Reg_249

* SPI GPIO pin define in “porting_misc.h”. This is by SiP requirement.
For example:

```
#define SPI_CS            PC2
#define SPI_CK            PD3
#define SPI_MOSI          PD2
#define SPI_MISO          PD1
#define DEFAULT_INT       PD0
```

* spiGpioDelay() is to delay 1us.

c.

Change all SPI GPIO pins as output mode, output HIGH and last 10ms.

d.

MCU change to normal SPI pin function.

e.

MCU SPI set Reg_249[7]=1. This step configures RT568 MISO, INT pin as output.

f.

MCU SPI write Reg_53=0xC0, Reg_40=0xC0, wait 25ms for RF LDO power on.

g.

MCU SPI write Reg_53=0x80, wait 10ms for RF chip enable.

Appendix 1

Engineers can refer to the following code to complete I/O remapping. This example demonstrates seven sorting methods as shown in the following table. If SiP sorting is not in this table, the code must be modified according to the same rules.
[The "SPI_IO_ORDER" parameter indicates different cases.]

The following example of 7 case order table

	Case 1	Case 2	Case 3	Case 4	Case 5	Case 6	Case7
GPIO 0	CS	INT	MISO	MOSI	CLK	INT	CS
GPIO 1	CLK	CS	INT	MISO	MOSI	CS	CLK
GPIO 2	MOSI	CLK	CS	INT	MISO	CLK	MISO
GPIO 3	MISO	MOSI	CLK	CS	INT	MISO	MOSI
GPIO 4	INT	MISO	MOSI	CLK	CS	MOSI	INT

```
//SPI IO mapping. Must do this after Power ON
void RT568_SpiIoMapping(void)
{
   //(1) Set all SPI GPIO pin as output. Be careful not to set 5 pins as HIGH, it
   //will trigger HW mapping SPI pin
   SPI_GPIO_Init();   //here set all CLK,CS,MOSI,NISO,INT as output
   SPI_CS=1;
   SPI_CK=0;
   SPI_MOSI=0;
   SPI_MISO=0;
   DEFAULT_INT=0;

   //(2) Write R248, R249 GPIO select
#if (SPI_IO_ORDER==1)
      //Write R248=8'b10,001,000, R249=8'b0,100,011,0
      //GPIO0[2:0]=0 - CS
      //GPIO1[2:0]=1 - CLK
      //GPIO2[2:0]=2 - MOSI
```

```
        //GPIO3[2:0]=3 - MISO
        //GPIO4[2:0]=4 - INT
        CLK_SysTickDelay(50);
        spiGpioWriteReg(248, 0x88);
        CLK_SysTickDelay(50);
        spiGpioWriteReg(249, 0x46);
        CLK_SysTickDelay(50);
#elif (SPI_IO_ORDER==2)
        //Write R248=8'b01,000,100, R249=8'b0,011,010,0
        //GPIO0[2:0]=4 - INT
        //GPIO1[2:0]=0 - CS
        //GPIO2[2:0]=1 - CLK
        //GPIO3[2:0]=2 - MOSI
        //GPIO4[2:0]=3 - MISO
        CLK_SysTickDelay(50);
        spiGpioWriteReg(248, 4 | (0<<3) | ((1&0x03)<<6));
        CLK_SysTickDelay(50);
        spiGpioWriteReg(249, ((1&0x04)>>2) | (2<<1) | (3<<4));
        CLK_SysTickDelay(50);
#elif (SPI_IO_ORDER==3)
        //Write R248=8'b00,100,011, R249=8'b0,010,001,0
        //GPIO0[2:0]=3 - MISO
        //GPIO1[2:0]=4 - INT
        //GPIO2[2:0]=0 - CS
        //GPIO3[2:0]=1 - CLK
        //GPIO4[2:0]=2 - MOSI
        CLK_SysTickDelay(50);
        spiGpioWriteReg(248, 3 | (4<<3) | ((0&0x03)<<6));
        CLK_SysTickDelay(50);
        spiGpioWriteReg(249, ((0&0x04)>>2) | (1<<1) | (2<<4));
        CLK_SysTickDelay(50);
#elif (SPI_IO_ORDER==4)
        //Write R248=8'b00,100,011, R249=8'b0,010,001,0
        //GPIO0[2:0]=2 - MOSI
        //GPIO1[2:0]=3 - MISO
        //GPIO2[2:0]=4 - INT
        //GPIO3[2:0]=0 - CS
        //GPIO4[2:0]=1 - CLK
```

```
        CLK_SysTickDelay(50);
        spiGpioWriteReg(248, 2 | (3<<3) | ((4&0x03)<<6));
        CLK_SysTickDelay(50);
        spiGpioWriteReg(249, ((4&0x04)>>2) | (0<<1) | (1<<4));
        CLK_SysTickDelay(50);
#elif (SPI_IO_ORDER==5)
        //Write R248=8'b00,100,011, R249=8'b0,010,001,0
        //GPIO0[2:0]=1 - CLK
        //GPIO1[2:0]=2 - MOSI
        //GPIO2[2:0]=3 - MISO
        //GPIO3[2:0]=4 - INT
        //GPIO4[2:0]=0 - CS
        CLK_SysTickDelay(50);
        spiGpioWriteReg(248, 1 | (2<<3) | ((3&0x03)<<6));
        CLK_SysTickDelay (50);
        spiGpioWriteReg(249, ((3&0x04)>>2) | (4<<1) | (0<<4));
        CLK_SysTickDelay (50);
#elif (SPI_IO_ORDER==6)     //for SIP
        //Write R248=8'b01,000,100, R249=8'b0,010,011,0
        //GPIO0[2:0]=4 - INT
        //GPIO1[2:0]=0 - CS
        //GPIO2[2:0]=1 - CLK
        //GPIO3[2:0]=3 - MISO
        //GPIO4[2:0]=2 - MOSI
        CLK_SysTickDelay(50);
        spiGpioWriteReg(248, 4 | (0<<3) | ((1&0x03)<<6));
        CLK_SysTickDelay(50);
        spiGpioWriteReg(249, ((1&0x04)>>2) | (3<<1) | (2<<4));
        CLK_SysTickDelay(50);
#elif (SPI_IO_ORDER==7)     //for daughter board
        //Write R248=8'b10,001,000, R249=8'b0,100,011,0
        //GPIO0[2:0]=0 - CS
        //GPIO1[2:0]=1 - CLK
        //GPIO2[2:0]=3 - MISO
        //GPIO3[2:0]=2 - MOSI
        //GPIO4[2:0]=4 - INT
        CLK_SysTickDelay(50);
        spiGpioWriteReg(248, 0 | (1<<3) | ((3&0x03)<<6));
```

```
    CLK_SysTickDelay(50);
    spiGpioWriteReg(249, ((3&0x04)>>2) | (2<<1) | (4<<4));
    CLK_SysTickDelay(50);
#endif
    //(3) Output all pin as HIGH, last 10ms(>1ms). trigger HW take effect
    SPI_GPIO_Init();      //Set all as output
    SPI_CS=1;
    SPI_CK=1;
    SPI_MOSI=1;
    SPI_MISO=1;
    DEFAULT_INT=1;
    CLK_SysTickDelay(10000);   //delay 10ms
    // CLK_SysTickDelay(25000); //RT568 POR_Reset, delay 25ms, wait for
    16M          //stable, then start RT568_Init. Add the line when using HIRC

    //(4) Init GPIO & SPI
    //MCU_GpioIntEnable(); //configure ext_int GPIO pin. Not enable INT!
    MCU_GpioPinInit();    //Set GPIO interrupt pin as input
    MCU_SpiInit();          //Intial SPI pin, change MISO direction as INPUT
    //Tiny_Delay(100);
    //SPI_Open(SPI0, SPI_MASTER, SPI_MODE_0, 8, 1000000); //1M clk

#if (SPI_IO_ORDER==1)
    //set RT568 MISO, INT as output
    SPI_1BYT_SetTx(249, (0x46 | 0x80));
#elif (SPI_IO_ORDER==2)
    //set RT568 MISO, INT as output
    SPI_1BYT_SetTx(249, ((1&0x04)>>2) | (2<<1) | (3<<4) | (1<<7));
#elif (SPI_IO_ORDER==3)
    //set RT568 MISO, INT as output
    SPI_1BYT_SetTx(249, ((0&0x04)>>2) | (1<<1) | (2<<4) | (1<<7));
#elif (SPI_IO_ORDER==4)
    //set RT568 MISO, INT as output
    SPI_1BYT_SetTx(249, ((4&0x04)>>2) | (0<<1) | (1<<4) | (1<<7));
#elif (SPI_IO_ORDER==5)
    //set RT568 MISO, INT as output
    SPI_1BYT_SetTx(249, ((3&0x04)>>2) | (4<<1) | (0<<4) | (1<<7));
#elif (SPI_IO_ORDER==6)
```

```
        //set RT568 MISO, INT as output
        SPI_1BYT_SetTx(249, ((1&0x04)>>2) | (3<<1) | (2<<4) | (1<<7));
#elif (SPI_IO_ORDER==7)
        //set RT568 MISO, INT as output
        SPI_1BYT_SetTx(249, ((3&0x04)>>2) | (2<<1) | (4<<4) | (1<<7));
#endif
//manual control
//To gurantee DC/DC power on when set R40=0x90
SPI_1BYT_SetTx(53, 0xC0);
//enable LDO
SPI_1BYT_SetTx(40, 0xC0);
//Put delay after LDO_enable, or set register may have strange behavior!
Tiny_Delay(25000);
//enable chip
SPI_1BYT_SetTx(53, 0x80);
//Put delay after LDO_enable, or set register may have strange behavior!
Tiny_Delay(10000);
}
```

Appendix 2

```
void spiGpioWriteReg(const unsigned char regAddr, const unsigned char
regData)
{
   // Counter used to clock out the data
   unsigned char SPICount;
   // Define a data structure for the SPI data.
   unsigned char SPIData;
   // Make sure we start with /CS high
   SPI_CS = 1;
   // and CK low
   SPI_CK = 0;
   // Preload the data to be sent with Address
   //SPIData = regAddr;
   // Set /CS low to start the SPI cycle 25nS
   // Although SPIData could be implemented as an "int", resulting in one
   // loop, the routines run faster when two loops are implemented with
   // SPIData implemented as two "char"s.
   SPI_CS = 0;

   spiGpioDelay();

   //Address 1th byte
   SPIData = regAddr & 0x7F;
   // Prepare to clock out the Address byte
   for (SPICount = 0; SPICount < 8; SPICount++)
   {
      // Check for a 1 and set the MOSI line appropriately
      if (SPIData & 0x80)
         SPI_MOSI = 1;
      else
         SPI_MOSI = 0;
      spiGpioDelay();              //delay half clk cycle
      SPI_CK = 1;                  // Toggle the clock line
      spiGpioDelay();
```

```
    SPI_CK = 0;
    SPIData <<= 1;              // Rotate to get the next bit
}                               // and loop back to send the next bit
                                // Repeat for the Data byte
//Address 2nd byte
SPIData = (regAddr & 0x80)>>7;
// Prepare to clock out the Address byte
for (SPICount = 0; SPICount < 8; SPICount++)
{
   // Check for a 1 and set the MOSI line appropriately
   if (SPIData & 0x80)
      SPI_MOSI = 1;
   else
      SPI_MOSI = 0;
   spiGpioDelay();              //delay half clk cycle
   SPI_CK = 1;                  // Toggle the clock line
   spiGpioDelay();
   SPI_CK = 0;
   SPIData <<= 1;               // Rotate to get the next bit
}
//Data
SPIData = regData;              // Preload the data to be sent with Data
// Prepare to clock out the Data
for (SPICount = 0; SPICount < 8; SPICount++)
{
   if (SPIData & 0x80)
      SPI_MOSI = 1;
   else
      SPI_MOSI = 0;
   spiGpioDelay();
   SPI_CK = 1;
   spiGpioDelay();
   SPI_CK = 0;
   SPIData <<= 1;
}
spiGpioDelay();
SPI_CS = 1;
SPI_MOSI = 0;
```

```
}
void spiGpioDelay(void)
{
    CLK_SysTickDelay(1);     //one 1M cycle=1us
}
```