RTL9310

GPIO

Application Note

Rev. 1.0
27 Apr 2018

Realtek Semiconductor Corp.
No. 2, Innovation Road II, Hsinchu Science Park, Hsinchu 300, Taiwan
Tel.: +886-3-578-0211 Fax: +886-3-577-6047
www.realtek.com

USING THIS DOCUMENT
This document is intended for use by the system engineer when integrating with Realtek switch products. Though every effort has been made to assure that this document is current and accurate, more information may have become available subsequent to the production of this guide. In that event, please contact your Realtek representative for additional information that may help in the development process.

Revision	Release Date	Summary
1.0	2018/04/27	Initial

Contents

Table List

Figure List

1 Overview

RTL9310 supports two kinds of general-purpose input/output (GPIO), one is the internal GPIO directly connected on SoC and the other is external GPIO extended by RTL8231.

Internal GPIO	32 GPIOs
	Interrupt function
External GPIO	37 GPIOs per RTL8231
	RTL8231's 8 strapping pins: GPIO[15,16,17,18,19,20,35,36] only support GPO function
	Interrupt function only supported on the 1st RTL8231

2 Internal GPIO

Each internal GPIO pin can be configured as input pin or output pin. If it is configured as input pin, interrupt can be enabled or not.

Besides that, the internal GPIO pins can be used for other peripherals by configuring the corresponding multiplex mode register(introduced in chapter 2.3).

2.1 General GPIO

RTL9310 supports 32 internal GPIOs, GPIO[31:0], which are divided into 4 sets.

- Port A[7:0]: GPIO[7] ~ GPIO[0]
- Port B[7:0]: GPIO[15] ~ GPIO[8]
- Port C[7:0]: GPIO[23] ~ GPIO[16]
- Port D[7:0]: GPIO[31] ~ GPIO[24]

Table 2-1 is GPIO interface pins.

I:	Input Pin
O:	Output Pin
I/O:	Bi-Direction Input/Output Pin

I_{PU}:	Input Pin With Pull-Up Resistor;
I_{PD}:	Input Pin With Pull-Down Resistor;
O_{PU}:	Output Pin With Pull-Up Resistor;
O_{PD}:	Output Pin With Pull-Down Resistor;

Table 2-1 GPIO Interface Pin

Pin Name	Pin No.	SOC Define	Type	Multiplex Mode	Default
GPIO[0]	AK25	PortA[0]	I/O_{PD}	System LED	System LED
GPIO[1]	AM25	PortA[1]	I/O_{PU}	MDC	GPIO
GPIO[2]	AL25	PortA[2]	I/O_{PU}	MDIO	GPIO
GPIO[3]	AJ23	PortA[3]	I/O_{PU}	JTAG_TMS	JTAG_TMS
GPIO[4]	AK23	PortA[4]	I/O_{PU}	JTAG_TCK	JTAG_TCK
GPIO[5]	AH23	PortA[5]	I/O_{PD}	JTAG_TRST#	JTAG_TRST#
GPIO[6]	AK24	PortA[6]	I/O_{PD}	JTAG_TDI	JTAG_TDI
GPIO[7]	AL24	PortA[7]	I/O_{PD}	JTAG_TDO	JTAG_TDO
GPIO[8]	AL23	PortB[0]	I/O_{PU}	SPI_SCK	GPIO
GPIO[9]	AM23	PortB[1]	I/O_{PU}	SPI_MISO	GPIO
GPO[10]	AL22	PortB[2]	I/O_{PU}	SPI_MOSI	GPIO
GPIO[11]	AK22	PortB[3]	I/O_{PU}	SPI_CS#0	GPIO
GPIO[12]	AH22	PortB[4]	I/O_{PU}	SPI_CS#1	GPIO
GPIO[13]	AK19	PortB[5]	I/O_{PU}	SCL0	GPIO
GPIO[14]	AL20	PortB[6]	I/O_{PU}	SCL1	GPIO
GPIO[15]	AJ19	PortB[7]	I/O_{PU}	SDA0	GPIO
GPIO[16]	AH19	PortC[0]	I/O_{PU}	SDA1	GPIO
GPIO[17]	AG19	PortC[1]	I/O_{PU}	SDA2	GPIO
GPIO[18]	AK20	PortC[2]	I/O_{PU}	SDA3	GPIO
GPIO[19]	AH20	PortC[3]	I/O_{PU}	SDA4	GPIO
GPIO[20]	AG20	PortC[4]	I/O_{PU}	SDA5	GPIO
GPIO[21]	AM21	PortC[5]	I/O_{PU}	SDA6	GPIO
GPIO[22]	AL21	PortC[6]	I/O_{PU}	SDA7	GPIO
GPIO[23]	AK21	PortC[7]	I/O_{PU}	SDA8	GPIO
GPIO[24]	AJ21	PortD[0]	I/O_{PU}	SDA9	GPIO
GPIO[25]	AH21	PortD[1]	I/O_{PU}	SDA10	GPIO
GPIO[26]	AG21	PortD[2]	I/O_{PU}	SDA11	GPIO
GPIO[27]	AH17	PortD[3]	I/O_{PD}	X	GPIO
GPIO[28]	AK17	PortD[4]	I/O_{PU}	X	GPIO

GPIO[29]	AM19	PortD[5]	I/O$_{PU}$	MDC	GPIO
GPIO[30]	AL19	PortD[6]	I/O$_{PU}$	MDIO	GPIO
GPIO[31]	AJ17	PortD[7]	I/O$_{PD}$	LED_SYNC (For 74HC595 application)	GPIO

Note

1. GPIO[5] can not connect with internal pull high device.

2.1.1 Register Setting

Following registers are in RTL9310_SOC_RegisterFile.

GPIO configuration flow:

- Multiplex mode register:

  Used to configure as GPIO via disabling its multiplex function (Explained in chapter 2.3)

- *PABCD_DIR* register(in Table 2-2):

  Used to set the GPIO direction.

- *PABCD_DAT* register(in Table 2-3) :

  Used to access the GPIO data both for input & output.

For example, if configure the GPIO[0] as output with high level voltage, the corresponding register settings is as follows.

MAC_L2_GLOBAL_CTRL2.SYS_LED_EN = 0 (Disable GPIO0 system LED function);

PABCD_DAT.PD_A[0] = 1;

PABCD_DIR.DRC_A[0] = 1;

Table 2-2 PABCD_DIR Register

Bits	Field	Description	Type	Default
31:24	DRC_A[7:0]	Pin direction configuration of Port A. 0=Configured as input pin 1=Configured as output pin Note: PortA[7:0] as GPIO#7~GPIO#0	R/W	00H
23:16	DRC_B[7:0]	Pin direction configuration of Port B. 0=Configured as input pin 1=Configured as output pin Note: PortB[7:0] as GPIO#15~GPIO#8	R/W	00H

15:8	DRC_C[7:0]	Pin direction configuration of Port C. 0=Configured as input pin 1=Configured as output pin Note: PortC[7:0] as GPIO#23~GPIO#16	R/W	00H
7:0	DRC_D[7:0]	Pin direction configuration of Port D. 0=Configured as input pin 1=Configured as output pin Note: PortC[7:0] as GPIO#31~GPIO#24	R/W	00H

Table 2-3 PABCD_DAT Register

Bits	Field	Description	Type	Default
31:24	PD_A[7:0]	Pin data of Port A. 0 : Data=0 1 : Data=1	R/W	00H
23:16	PD_B[7:0]	Pin data of Port B. 0 : Data=0 1 : Data=1	R/W	00H
15:8	PD_C[7:0]	Pin data of Port C. 0 : Data=0 1 : Data=1	R/W	00H
7:0	PD_D[7:0]	Pin data of Port D. 0 : Data=0 1 : Data=1	R/W	00H

2.1.2 SDK Setting

■ **SDK3**

In sdk\system\common\rtcore\ rtcore_init.c, internal GPIO initialization will be done by calling the function drv_gpio_init(unit) as below. User can add their own GPIO configuration code in the function: drv_gpio_init(unit).

```
#if defined(CONFIG_SDK_DRIVER_GPIO)
if ((ret = gpio_probe(unit)) == RT_ERR_OK)
{
        drv_gpio_init(unit);
}
#endif
```

■ **rtk API**

int32 drv_generalCtrlGPIO_pin_init(uint32 unit, drv_generalCtrlGpio_devId_t dev, uint32 pinId, drv_generalCtrlGpio_pinConf_t *pData);

int32 drv_generalCtrlGPIO_direction_set(uint32 unit, drv_generalCtrlGpio_devId_t dev, uint32 pinId, drv_gpio_direction_t direction);

int32 drv_generalCtrlGPIO_dataBit_set(uint32 unit, drv_generalCtrlGpio_devId_t dev, uint32 pinId, uint32 data);

int32 drv_generalCtrlGPIO_dataBit_get(uint32 unit, drv_generalCtrlGpio_devId_t dev, uint32 pinId, uint32 *pData);

Note

1. To configure internal GPIO, parameter “dev” must be assigned as 0.
2. Need to make sure the value of data before setting the direction as output.
3. Need to make sure the pin mode is configured as general GPIO purpose, refer to register setting in chapter 2.3.

2.1.3 Application Example

- Configure Pin: AK25 as the GPIO (GPIO[0]), direction as output while initialization, and then the direction and data of that GPIO can be changed dynamically.

Step 1 Disable GPIO[0] as system LED function.

type	- LED_TYPE_SYS;
enable	- DISABLED;

rtk_led_sysEnable_set(uint32 unit, rtk_led_type_t type, rtk_enable_t enable).

Step 2 Config GPIO[0] direction output, default_value, and interrupt Disable.

dev	- internal GPIO device id should be fixed 0;
pinId	- 0;
pData->direction	- GPIO_DIR_OUT, could be changed in Step3;
pData->default_value	- 1, could be changed in Step 4;
pData->int_gpio.interruptEnable	- GPIO_INT_DISABLE;

drv_generalCtrlGPIO_pin_init(uint32 unit, drv_generalCtrlGpio_devId_t dev, uint32 pinId, drv_generalCtrlGpio_pinConf_t *pData);

Step 3 GPIO[0] direction set.

dev	- internal GPIO device id should be fixed 0;
pinId	- 0;
direction	- GPIO_DIR_OUT GPIO_DIR_IN;

drv_generalCtrlGPIO_direction_set(uint32 unit, drv_generalCtrlGpio_devId_t dev, uint32 pinId, drv_gpio_direction_t direction);

Step 4 GPIO[0] data set.

dev	- internal GPIO device id should be 0;
pinId	- 0;
data	- 1 high level 0 low level;

int32 drv_generalCtrlGPIO_dataBit_set(uint32 unit, drv_generalCtrlGpio_devId_t dev, uint32 pinId, uint32 data);

2.2 Interrupt Function

All internal GPIO pins support interrupt function when set as GPI. The configuration of interrupt enable, mode and status are provided.

2.2.1 Register Setting

Following registers are in RTL9310_SOC_RegisterFile.

Table 2-4 PAB_IMR Register

Bits	Field	Description	Type	Default
31:30	PA7_IM	PortB.7 interrupt mode. 0b00b: Disable interrupt 0b01b: Enable falling edge interrupt 0b10b: Enable rising edge interrupt 0b11b: Enable both falling or rising edge interrupt	RW	0x0
29:28	PA6_IM	PortB.6 interrupt mode.	RW	0x0
…	…	…		
17:16	PA0_IM	PortB.0 interrupt mode.	RW	0x0
15:14	PB7_IM	PortA.7 interrupt mode.	RW	0x0
13:12	PB6_IM	PortA.6 interrupt mode.	RW	0x0
…	…	…		
1:0	PB0_IM	PortA.0 interrupt mode.	RW	0x0

Table 2-5 PCD_IMR Register

Bits	Field	Description	Type	Default
31:30	PC7_IM	PortC.7 interrupt mode. 0b00b: Disable interrupt 0b01b: Enable falling edge interrupt 0b10b: Enable rising edge interrupt 0b11b: Enable both falling or rising edge interrupt	RW	0x0

29:28	PC6_IM	PortC.6 interrupt mode.	RW	0x0
…	…	…		
17:16	PC0_IM	PortC.0 interrupt mode.	RW	0x0
15:14	PD7_IM	PortC.7 interrupt mode.	RW	0x0
13:12	PD6_IM	PortC.6 interrupt mode.	RW	0x0
…	…	…		
1:0	PD0_IM	PortC.0 interrupt mode.	RW	0x0

Table 2-6 PABCD_ISR Register

Bits	Field	Description	Type	Default
31:24	IPS_A[7:0]	Interrupt pending status of port A. Write '1' to clear the interrupt	R/WC	00H
23:16	IPS_B[7:0]	Interrupt pending status of port B. Write '1' to clear the interrupt	R/WC	00H
15:8	IPS_C[7:0]	Interrupt pending status of port C. Write '1' to clear the interrupt	R/WC	00H
7:0	IPS_D[7:0]	Interrupt pending status of port D. Write '1' to clear the interrupt	R/WC	00H

2.2.2 SDK Setting

SDK3

None.

rtk API

int32 drv_generalCtrlGPIO_intrHandler_register(uint32 unit, drv_generalCtrlGpio_devId_t dev, uint32 pinId, drv_generalCtrlGPIO_Isr_cb_f gpioIsrCallback);

int32 drv_generalCtrlGPIO_intrHandler_unregister(uint32 unit, drv_generalCtrlGpio_devId_t dev, uint32 pinId);

int32 drv_generalCtrlGPIO_pin_init(uint32 unit, drv_generalCtrlGpio_devId_t dev, uint32 pinId, drv_generalCtrlGpio_pinConf_t *pData) ;

Note

1. To call the APIs above for internal GPIO, parameter "dev" must be assigned as 0.

2.2.3 Application Example

- Enable GPIO[0] interrupt function and register its ISR.

Step 1 Disable GPIO[0] as system LED function.

type	- LED_TYPE_SYS;
enable	- DISABLED;

rtk_led_sysEnable_set(uint32 unit, rtk_led_type_t type, rtk_enable_t enable).

Step 2 GPIO[0] interrupt request register.

dev	- internal GPIO device id should be fixed 0;
pinId	- 0;
gpioIsrCallback	- interrupt service routing;

int32 drv_generalCtrlGPIO_intrHandler_register(uint32 unit, drv_generalCtrlGpio_devId_t dev, uint32 pinId, drv_generalCtrlGPIO_Isr_cb_f gpioIsrCallback);

Step 3 Config GPIO[0] direction input, and interrupt Enable.

dev	- internal GPIO device id should be fixed 0;
pinId	- 0;
pData->direction	- GPIO_DIR_IN, interrupt request input;
pData->int_gpio.interruptEnable	- GPIO_INT_FALLING_EDGE, GPIO_INT_RISING_EDGE, GPIO_INT_BOTH_EDGE;

drv_generalCtrlGPIO_pin_init(uint32 unit, drv_generalCtrlGpio_devId_t dev, uint32 pinId, drv_generalCtrlGpio_pinConf_t *pData);

2.3 Multiplex Mode

The GPIO pins are shared with peripheral pins and the multiplex mode is configured by registers. (in RTL9310_RegisterFile)

2.3.1 Multiplex Mode of GPIO[0]

GPIO[0] could be used as system LED by the configuration of *SYS_LED_EN* register.

Table 2-7 MAC_L2_GLOBAL_CTRL2 Register – SYS_LED_EN

Bits	Field	Description	Type	Default

8	SYS_LED_EN	When enable System LED, the GPIO[0] change to system LED pin. Strapping pin. 0b0: Disable system LED 0b1: Enable system LED	RW	0x1

Note

1. The default value of SYS_LED_EN is enabled. If wants to enable GPIO[0], must disable system LED.

2.3.2 Multiplex Mode of GPIO[2:1]

GPIO[1] could be used as MDC and GPIO[2] could be used as MDIO via setting *EXT_GPIO_EN* to '1' in *EXT_GPIO_GLB_CTRL*. The MDC/MDIO access RTL8231 to extend GPIO pins.

Table 2-8 EXT_GPIO_GLB_CTRL Register – EXT_GPIO_EN

Bits	Field	Description	Type	Default
8	EXT_GPIO_EN	Enable or Disable extend GPIO Function; this bit will assigned to control GPIO1 and GPIO2 pin; 0b0: disable, GPIO[2:1] will act as GPIO 0b1: enable, GPIO[1] will act as GPIO_MDC and GPIO[2] act as GPIO_MDIO	RW	0x0

2.3.3 Multiplex Mode of GPIO[7:3]

GPIO[7:3] could be used to support EJTAG interface.

GPIO[3]=>JTAG_TMS

GPIO[4]=>JTAG_TCK

GPIO[5]=>JTAG_TRST#

GPIO[6]=>JTAG_TDI

GPIO[7]=>JTAG_TDO

Table 2-9 MAC_L2_GLOBAL_CTRL2 Register

Bits	Field	Description	Type	Default

15	JTAG_SEL	Select GPIO or E-JTAG function
		GPIO[3]=>JTAG_TMS
		GPIO[4]=>JTAG_TCK
		GPIO[5]=>JTAG_TRST#
		GPIO[6]=>JTAG_TDI
		GPIO[7]=>JTAG_TDO
		
		0b0: GPIO
		0b1: E-JTAG
			RW	0x1

2.3.4 Multiplex Mode of GPIO[12:8]

GPIO[12:8] could be used to support master SPI interface.

GPIO8, GPIO9 and GPIO10 work as SCK, MISO and MOSI separately.

GPIO12 works as CS#1.

GPIO11 works as CS#0.

Table 2-10 SPI_CTRL0 Register

Bits	Field	Description	Type	Default
10	GPIO_SPI_SEL	Select GPIO8~GPIO10 works as GPIO or SCK/ MISO/ MOSI. 0b0: works as GPIO8~GPIO10; 0b1: GPIO8 works as SCK, GPIO9 works as MISO, GPIO10 works as MOSI.	RW	0x0
9	GPIO12_CSB1_SEL	Select GPIO12 works as GPIO or CS#1. 0b0: GPIO12; 0b1: CS#1	RW	0x0
8	GPIO11_CSB0_SEL	Select GPIO11 works as GPIO or CS#0. 0b0: GPIO11; 0b1: CS#0	RW	0x0

2.3.5 Multiplex Mode of GPIO[26:13]

GPIO[26:13] could be used to support master I2C interface.

GPIO[26:15] work as SDA[11:0].

GPIO13 works as SCL0 of master I2C #1.

GPIO14 works as SCL1 of master I2C #2.

Table 2-11 I2C_MST_IF_SEL Register

Bits	Field	Description	Type	Default
13:12	GPIO_SCL_SEL	Bit mask[1:0] for selecting GPIO14_GPIO13 works as GPIO or SCL. 0b0: GPIO 0b1: SCL Note: 1. GPIO13 is mapping to SCL0 of controller#1. 2. GPIO14 is mapping to SCL1 of controller#2.	RW	0x0

11:0	GPIO_SDA_SEL	Bit mask[11:0] for selecting GPIO26_GPIO15 work as GPIO or SDA. 0b0: GPIO 0b1: SDA Note: GPIO15_GPIO26 is mapping to SDA0_SDA11.	RW	0x0

2.3.6 Multiplex Mode of GPIO[30:29]

GPIO[30:29] could be used as SMI_SET3 function.

GPIO[29]=>SMI_SET3_MDC

GPIO[30]=>SMI_SET3_MDIO

Table 2-12 MAC_L2_GLOBAL_CTRL2 Register

Bits	Field	Description	Type	Default
17	SMI_SET3_SEL	Select GPIO or SMI_SET3 function GPIO[29]=>SMI_SET3_MDC GPIO[30]=>SMI_SET3_MDIO 0b0: GPIO 0b1: SMI_SET3	RW	0x0

2.3.7 Multiplex Mode of GPIO[31]

GPIO[31] could be used as LED_SYNC(For 74HC595 application) function(Please refer to 9310 LED application note in details).

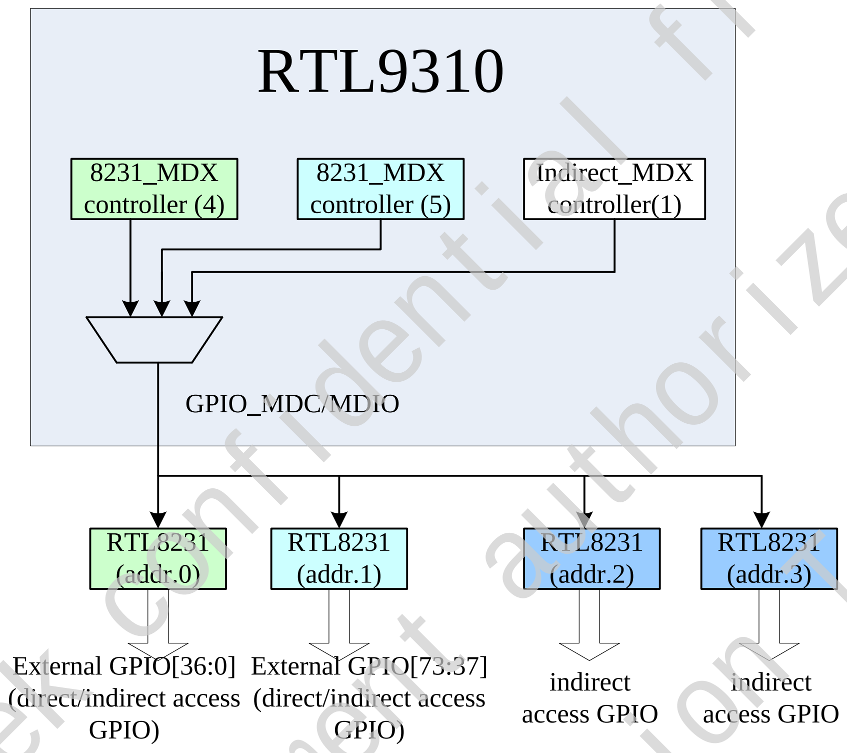
Table 2-13 MAC_L2_GLOBAL_CTRL2 Register

Bits	Field	Description	Type	Default
16	LED_SYNC_SEL	Select GPIO[31] or LED_SYNC function 0b0: GPIO[31] 0b1: LED_SYNC	RW	0x0

3 External GPIO

Internal GPIO[2:1](MDC/MDIO) of RTL9310 could be used to connect RTL8231 to extend GPIO. As shown in below Figure 3-1.

Figure 3-1 RTL9310 supports external GPIO



There are three 8231_MDX controllers to access the RTL8231 in switch. Both the 8231_MDX controller(4) and 8231_MDX controller(5) are turned off in default state, and the Indirect_MDX controller(1) is always on.

The 8231_MDX controller(4) works only for GPIO[36:0] at the RTL8231(addr.0) using the direct access mode,

The 8231_MDX controller(5) only for GPIO[73:37] at the RTL8231(addr.1) using the direct access mode too,

The Indirect_MDX controller(1) can access all of RTL8231 pins(addr.0,addr.1,addr2,addr3,etc) using the indirect access mode.

The main differences between the direct access mode and indirect access mode are as follows.

1> In the direct access mode, user can get the pin status(high or low level) directly, but in indirect access mode, what the result user gets is the 8231 register value with 16bis pin;

2> In the direct access mode, user can trigger ASIC to single read or periodically read automatically; The indirect access mode only supports single read.

Besides that, while configured as periodically read, the direct access mode can support interrupt feature(only GPIO[36:0] controlled by 8231_MDX controller supports);

3.1 General GPIO

RTL8231 totally provides 37 GPIO pins (except for 8 strapping pins, RTL8231's GPIO[15,16,17,18,19,20,35,36] only support GPO function). Figure 3-2 depicts the schematic of RTL8231.

RTL8231 should be configured MIIM mode by strapping pins MOD[0] and Dis_SMI.

RTL8231's address is strapped by strapping pin Addr[0]~Addr[4].

Table 3-1 depicts the RTL8231 MIIM mode GPIO configuration strapping pins.

Figure 3-2 Schematic of RTL8231

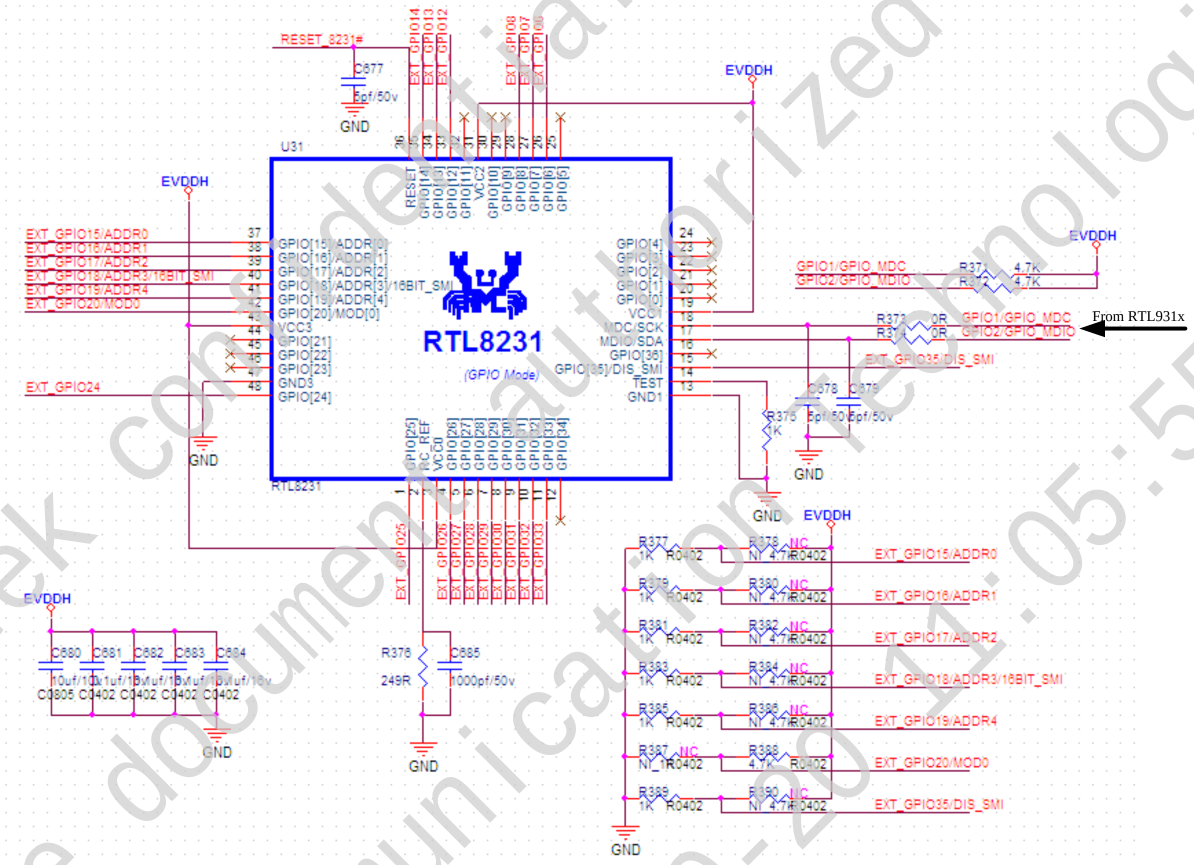


Table 3-1 RTL8231 MIIM mode GPIO configuration strapping pins

Pin Name	Pin No.	Multiplex Mode
GPIO[15]/Addr[0]	37	MIIM Mode:Addr[4:0] is PHY Address.
GPIO[16]/Addr[1]	38	
GPIO[15]/Addr[2]	39	
GPIO[15]/Addr[3]	40	
GPIO[15]/Addr[4]	41	
GPIO[20]/MOD[0]	42	MIIM Mode:Dis_SMI=0 and MOD[0]=1. Selects the MIIM mode when This Strapping Pin is Pulled-High and Dis_SMI is Pulled-Down.

GPIO[35]/Dis_SMI	15	Sets the RTL8231 to MIIM or Shift Register Mode. Pull-up for shift register mode. Pull-down for SMI Slave/MIIM mode. Dis_SMI=1 (pull-high) for shift register mode. Dis_SMI=0 (pull-down) for MIIM mode (initial value is pulled-down).

Note: For RTL8231 shift register mode, refer to RTL9310 LED application note.

3.1.1 Register Setting

Following registers are in RTL9310_RegisterFile.

Internal GPIO[2:1] is used as MDC/MDIO interface by configuration of *EXT_GPIO_GLB_CTRL.EXT_GPIO_EN* to '1'.

Table 3-2 EXT_GPIO_GLB_CTRL Register – EXT_GPIO_EN

Bits	Field	Description	Type	Default
8	EXT_GPIO_EN	Enable or Disable extend GPIO Function; this bit will be assigned to control GPIO1 and GPIO2 pin; 0b0: disable, GPIO[2:1] will act as GPIO 0b1: enable, GPIO[1] will act as GPIO_MDC and GPIO[2] act as GPIO_MDIO	RW	0x0

Direct Access Mode

RTL9310 directly accesses RTL8231 registers through two methods that configured by *EXT_GPIO_MDX4_ACCESS_MODE/EXT_GPIO_MDX5_ACCESS_MODE*. One is continuous periodic directing access, the other is singly periodic directing access when triggered.

For direct access mode, the maximum number of external GPIO is 74 (two RTL8231) , MDX4 controller control GPIO[36:0] which the address of RTL8231 is "0" and MDX5 controller control GPIO[73:37] which the address of RTL8231 is "1".

For example, configure following registers to continuous directly access external GPIO.

- Continuous directly access RTL8231(0)
  - Configure continuous mode

    EXT_GPIO_GLB_CTRL.EXT_GPIO_MDX4_ACCESS_MODE continuous mode
  - Enable polling RTL8231(0)

    EXT_GPIO_GLB_CTRL.EXT_GPIO_MDX4_EN enable
  - Configure RTL8231(0) pin direction

    EXT_GPIO_DIR_CTRL_1.EXT_GPIO_31_0_DIR,

    EXT_GPIO_DIR_CTRL_2.EXT_GPIO_36_32_DIR
  - Configure or get RTL8231(0) pin data

    EXT_GPIO_DATA_CTRL_1.EXT_GPIO_31_0_DATA,

    EXT_GPIO_DATA_CTRL_2. EXT_GPIO_63_32_DATA

Table 3-3 EXT_GPIO_GLB_CTRL Register – EXT_GPIO_MDX4/EXT_GPIO_MDX5

Bits	Field	Description	Type	Default
20:15	EXT_GPIO_MDX4_READY	Ready bits of 8231_MDX controller (4), read from 8231(0) reg1[9:4]	RO	0x0
14:9	EXT_GPIO_MDX5_READY	Ready bits of 8231_MDX controller (5), read from 8231(1) reg1[9:4]	RO	0x0
3	EXT_GPIO_MDX4_ACCESS_MODE	config 8231_MDX controller (4) accessing mode 0b0 : trigger mode 0b1 : continuous mode	RW	0x1
2	EXT_GPIO_MDX5_ACCESS_MODE	config 8231_MDX controller (5) accessing mode 0b0 : trigger mode 0b1 : continuous mode	RW	0x1
1	EXT_GPIO_MDX4_EN	Enable 8231_MDX controller (4) 0b0 : disable 0b1 : enable	RW	0x0
0	EXT_GPIO_MDX5_EN	Enable 8231_MDX controller (5) 0b0 : disable 0b1 : enable	RW	0x0

Table 3-4 EXT_GPIO_TRIG Register

Bits	Field	Description	Type	Default
1	EXT_GPIO_MDX4_TRIG	Trigger 8231_MDX controller (4) in trigger mode 0b0 : Complete 0b1 : Trigger	RW	0x0
0	EXT_GPIO_MDX5_TRIG	Trigger 8231_MDX controller (5) in trigger mode 0b0 : Complete 0b1 : Trigger	RW	0x0

Table 3-5 EXT_GPIO_DIR_CTRL_1 Register

Bits	Field	Description	Type	Default
31:0	EXT_GPIO_31_0_DIR	External GPIO [31:0] direction. 0b0: input 0b1: output	RW	0x0

Table 3-6 EXT_GPIO_DIR_CTRL_2 Register

Bits	Field	Description	Type	Default
31:0	EXT_GPIO_63_32_DIR	External GPIO [63:32] direction. 0b0: input 0b1: output	RW	0x0

Table 3-7 EXT_GPIO_DIR_CTRL_3 Register

Bits	Field	Description	Type	Default
9:0	EXT_GPIO_73_64_DIR	External GPIO [73:64] direction. 0b0: input 0b1: output	RW	0x0

Table 3-8 EXT_GPIO_DATA_CTRL_1 Register

Bits	Field	Description	Type	Default
31:0	EXT_GPIO_31_0_DATA	External GPIO [31:0] data. If EXT_GPIO_DIR=0 (input direction), then EXT_GPIO_31_0_DATA =input data [31:0] If EXT_GPIO_DIR=1 (output direction), then EXT_GPIO_31_0_DATA =output data [31:0]	RW	0x0

Table 3-9 EXT_GPIO_DATA_CTRL_2 Register

Bits	Field	Description	Type	Default
31:0	EXT_GPIO_63_32_DATA	External GPIO [63:32] data. If EXT_GPIO_DIR=0 (input direction), then EXT_GPIO_63_32_DATA =input data [63:32] If EXT_GPIO_DIR=1 (output direction), then EXT_GPIO_63_32_DATA =output data [63:32]	RW	0x0

Table 3-10 EXT_GPIO_DATA_CTRL_3 Register

Bits	Field	Description	Type	Default
9:0	EXT_GPIO_73_64_DATA	External GPIO [73:64] data. If EXT_GPIO_DIR=0 (input direction), then EXT_GPIO_31_0_DATA =input data [73:64] If EXT_GPIO_DIR=1 (output direction), then EXT_GPIO_31_0_DATA =output data [73:64]	RW	0x0

Indirect Access Mode

Configure following registers to indirectly access external GPIO.

Table 3-11 EXT_GPIO_INDRT_ACCESS_CTRL Register

Bits	Field	Description	Type	Default
28	GPIO_RCMD_FAIL	When RTL9310 trigger reading command (GPIO_RWOP=0 & GPIO_CMD=1,this bit clear to 0;When reading command fail to detect turn round bit, this bit is set to 1. Note: GPIO_RCMD_FAIL will be clear to 0 When next GPIO_CMD executing.	RO	0x0

27:12	GPIO_DATA	Input parameters: If RWOP=0 (read), then REG_DATA [15:0]=input data [15:0] If RWOP=1 (write), then REG_DATA [15:0]=output data [15:0]	RW	0x0
11:7	GPIO_REG	Select register number to access. 0x0: access register 0 0x1: access register 1 0x2: access register 2 : : 0x1f: access register 31	RW	0x0
6:2	GPIO_PHYADRR	Select RTL8231's PHY address to access	RW	0x0
1	GPIO_RWOP	Read/write operation. 0b0: read 0b1: Write	RW	0x0
0	GPIO_CMD	Request MAC to access RTL8231 Reg. 0b0: complete access 0b1: execute access Note: When MAC completes access, it will clear this bit.	RW	0x0

RTL8231 Register

The detailed RTL8231 register description for GPIO operation, please refer to RTL8231 datasheet. Bear in mind that all the settings configured into the RTL8231 register will effect only when the enable bit(RTL8231 REG0.Bit1) is set. Before setting the enable bit(RTL8231 REG0.Bit1), please make sure that all the pins initialization have been set correctly to avoid some unexpected phenomenon occurs.

For example, GPIO[3] of RTL8231 is used as reset pin which is low active. So the initial value of RTL8231 GPIO[3] should be set to 1 before setting the enable bit(RTL8231 REG0.Bit1).

3.1.2 SDK Setting

SDK3

If defined macro CONFIG_SDK_RTL8231, in \sdk\system\linux\rtcore\ rtcore_drv.c, rtcore_dev_init () calls drv_rtl8231_init() to enable extend GPIO Function.

```
static int32 __init rtcore_dev_init(void)
{
        #if defined(CONFIG_SDK_RTL8231)
                /* rtl8231 */
                if ((ret = rtl8231_probe(unit)) == RT_ERR_OK)
                        {
                                RT_ERR_CHK_EHDL(drv_rtl8231_init(unit), ret, {RT_INIT_ERR(ret, (MOD_INIT), "unit %u
                                        drv_rtl8231_init fail %d!\n", unit, ret);});
                        }
        #endif
}
```

rtk API

int32 drv_generalCtrlGPIO_dev_init(uint32 unit, drv_generalCtrlGpio_devId_t dev, drv_generalCtrlGpio_devConf_t *pData)

int32 drv_generalCtrlGPIO_pin_init(uint32 unit, drv_generalCtrlGpio_devId_t dev, uint32 pinId, drv_generalCtrlGpio_pinConf_t *pData);

int32 drv_generalCtrlGPIO_devEnable_set(uint32 unit, drv_generalCtrlGpio_devId_t dev, rtk_enable_t enable)

int32 drv_generalCtrlGPIO_direction_set(uint32 unit, drv_generalCtrlGpio_devId_t dev, uint32 pinId, drv_gpio_direction_t direction);

int32 drv_generalCtrlGPIO_dataBit_set(uint32 unit, drv_generalCtrlGpio_devId_t dev, uint32 pinId, uint32 data);

int32 drv_generalCtrlGPIO_dataBit_get(uint32 unit, drv_generalCtrlGpio_devId_t dev, uint32 pinId, uint32 *pData);

Note

1. To configure external GPIO, parameter "dev" can't be 0 (GEN_GPIO_DEV_ID0_INTERNAL).
2. Need to make sure the value of data before setting the direction as output.

3.1.3 Application Example

- Using RTL8231(0) GPIO[1] to output high level.

Step 1 RTL8231(0) device init.

dev	- 1, assign different dev > 0 for different RTL8231
pData->ext_gpio.access_mode	- EXT_GPIO_ACCESS_MODE_MDC;
pData->ext_gpio.address	- 0, RTL8231(0) strapping address is 0;

int32 drv_generalCtrlGPIO_dev_init(uint32 unit, drv_generalCtrlGpio_devId_t dev, drv_generalCtrlGpio_devConf_t *pData).

Step 2 Make sure all pins of RTL8231(0) is in expected direction and data;

Make sure all pins of RTL8231(0) output direction data won't have a bad influence on its link partner.

For example, initialize GPIO[2] output low level, interrupt disabled.

dev	- 1, assign different dev > 0 for different RTL8231;
pinId	- 2
pData->direction	- GPIO_DIR_OUT ;
pData->default_value	- 0;
pData->ext_gpio.interruptEnable	- EXT_GPIO_INT_DISABLE ,

drv_generalCtrlGPIO_pin_init(uint32 unit, drv_generalCtrlGpio_devId_t dev, uint32 pinId, drv_generalCtrlGpio_pinConf_t *pData).

Step 3 Enable the RTL8231(0) device.

dev	- 1, assign different dev > 0 for different RTL8231;
enable	- ENABLED;

int32 drv_generalCtrlGPIO_devEnable_set(uint32 unit, drv_generalCtrlGpio_devId_t dev, rtk_enable_t enable).

Step 4 RTL8231(0) GPIO[1] direction output.

dev	- 1, assign different dev > 0 for different RTL8231;
pinId	- 1;
direction	- GPIO_DIR_OUT;

int32 drv_generalCtrlGPIO_direction_set(uint32 unit, drv_generalCtrlGpio_devId_t dev, uint32 pinId, drv_gpio_direction_t direction);

Step 5 RTL8231(0) GPIO[1] data set high level.

Dev	- 1, assign different dev > 0 for different RTL8231;
pinId	- 1;
data	- 1;

int32 drv_generalCtrlGPIO_dataBit_set(uint32 unit, drv_generalCtrlGpio_devId_t dev, uint32 pinId, uint32 data);

3.2 Interrupt Function

RTL9310 support interrupt function on external GPIO[36:0]. To enable this function, need to enable MDX4 controller (*EXT_GPIO_MDX4_EN*) to directly access 1st RTL8231.

3.2.1 Register Setting

Following registers are in RTL9310_RegisterFile.

When external GPIO[36:0] is setting as GPI, RTL9310 support interrupt function via setting the following registers.

Table 3-12 IMR_EXT_GPIO Register

Bits	Field	Description	Type	Default

0	IMR_EXT_GPIO	Interrupt mask control for GPIOn of external PHY(RTL8231) status change.(n=0~36) 0b0: Does not allow any interrupt 0b1: Allow interrupt	RW	0x0

Table 3-13 ISR_EXT_GPIO Register

Bits	Field	Description	Type	Default
0	ISR_EXT_GPIO	Interrupt pending flag for external GPIOn status change.(n=0~36) 0b0: normal 0b1: interrupt pending	RW1C	0x0

Table 3-14 EXT_GPIO_INTR_MODE Register

Bits	Field	Description	Type	Default
1:0	EXT_GPIO_INTR_MODE	Each GPIOn of external RTL8231 has two bits to decide GPIOn's interrupt mode.(n=0~36) 0x0: reserved. 0x1: falling edge trigger interrupt. 0x2: rising edge trigger interrupt. 0x3: falling or rising edge trigger interrupt.	RW	0x3

3.2.2 SDK Setting

SDK3

None.

rtk API

int32 drv_generalCtrlGPIO_intrHandler_register(uint32 unit, drv_generalCtrlGpio_devId_t dev, uint32 pinId, drv_generalCtrlGPIO_Isr_cb_f gpioIsrCallback);

int32 drv_generalCtrlGPIO_intrHandler_unregister(uint32 unit, drv_generalCtrlGpio_devId_t dev, uint32 pinId);

int32 drv_generalCtrlGPIO_pin_init(uint32 unit, drv_generalCtrlGpio_devId_t dev, uint32 pinId, drv_generalCtrlGpio_pinConf_t *pData) ;

Note

1. To call the APIs above for external GPIO, parameter "dev" can not be assigned as 0.

2. Interrupt function supported on "pinId" 0~36 except 8 strapping pins.

3.2.3 Application Example

- Using RTL8231(0) GPIO[1] as interrupt pin to register ISR.

Step 1	RTL8231(0) device init.	Refer to 3.1.3 Application Example step1~step3
Step 2	RTL8231(0) pin init.	
Step 3	RTL8231(0) device enable.	

Step 4 GPIO[0] interrupt request register.

dev	- 1, assign different dev > 0 for different RTL8231;
pinId	- 1;
gpioIsrCallback	- interrupt service routing;

int32 drv_generalCtrlGPIO_intrHandler_register(uint32 unit, drv_generalCtrlGpio_devId_t dev, uint32 pinId, drv_generalCtrlGPIO_Isr_cb_f gpioIsrCallback);

Step 5 Config RTL8231(0) GPIO[1] direction input, and interrupt Enable.

dev	- 1, assign different dev > 0 for different RTL8231;
pinId	- 1;
pData->direction	- GPIO_DIR_IN, interrupt request input;
pData->ext_gpio.interruptEnable	- EXT_GPIO_INT_FALLING_EDGE, EXT_GPIO_INT_RISING_EDGE, EXT_GPIO_INT_BOTH_EDGE;

drv_generalCtrlGPIO_pin_init(uint32 unit, drv_generalCtrlGpio_devId_t dev, uint32 pinId, drv_generalCtrlGpio_pinConf_t *pData);