# Chapter 34 General-Purpose I/O Ports(GPIO)

## 34.1 Overview

GPIO is a programmable General Purpose Programming I/O peripheral. This component is a APB slave device.GPIO controls the output data and direction of external I/O pads. It also can read back thedata on external pads using memory-mapped registers.

### 34.1.1 Features

GPIOsupports the following features:

- 32 bits APB bus width
- 32 independently configurable signals
- Separate data registers and data direction registers for each signal
- Software control for each signal, or for each bit of each signal
- Configurable interrupt mode for Port A
- Port A has 32 bits

*Notes: Port A 32bits are corresponding to port A/B/C/D 8bits in Chapter1*

## 34.2 Block Diagram

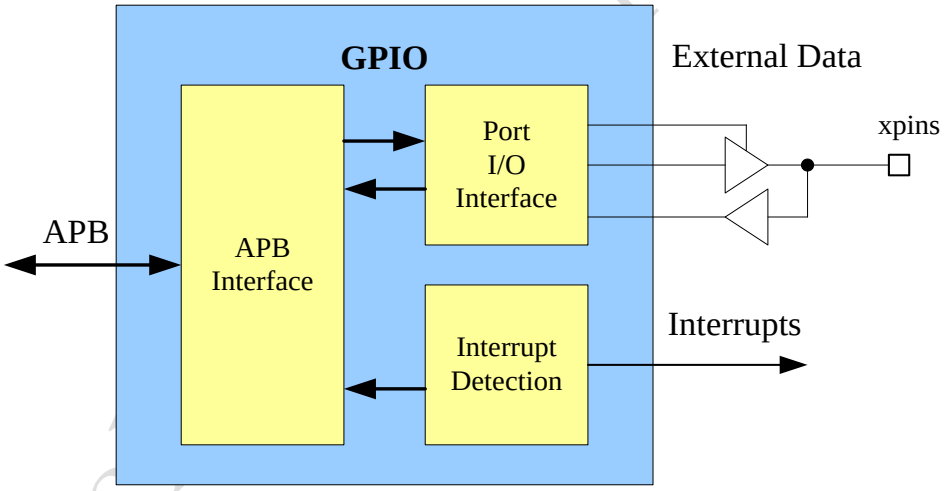


Fig.34-1GPIO block diagram

**Block descriptions:**

**APB Interface**

The APB Interface implements the APB slave operation. It'sdata bus width is 32 bits.

**Port I/O Interface**

External data Interface to or from I/O pads.

**Interrupt Detection**

Interrupt interface to or from interrupt controller.

# 34.3 Function description

## 34.3.1 Operation

### Control Mode(software)

Under software control, the data and direction control for the signal aresourced from the data register (GPIO_SWPORTA_DR) and direction control register (GPIO_SWPORTA_DDR).

The direction of the external I/O pad is controlled by a write to the Porta datadirection register (GPIO_SWPORTA_DDR). The data written to this memory-mapped register gets mappedonto an output signal, GPIO_PORTA_DDR, of the GPIO peripheral. This output signal controls thedirection of an external I/O pad.

The data written to the Porta data register (GPIO_SWPORTA_DR) drives the output buffer of the I/O pad.External data are input on the external data signal, GPIO_EXT_PORTA. Reading the external signal register(GPIO_EXT_PORTA) shows the value on the signal, regardless of the direction. This register is read-only,meaning that it cannot be written from the APB software interface.

### Reading External Signals

The data on the GPIO_EXT_PORTA external signal can always be read. The data on the external GPIO signal is read by an APB read of the memory-mapped register, GPIO_EXT_PORTA.

An APB read to the GPIO_EXT_PORTA register yields a value equal to that which is on the GPIO_EXT_PORTA signal.

### Interrupts

Port A can be programmed to accept external signals as interrupt sources on any of the bits of the signal. The type of interrupt is programmable with one of the following settings:

- Active-high and level
- Active-low and level
- Rising edge
- Falling edge

The interrupts can be masked by programming the GPIO_INTMASK register. The interrupt status can be read before masking (called raw status) and after masking.

The interrupts are combined into a single interrupt output signal, which has the same polarity as the individual interrupts. In order to mask the combined interrupt, all individual interrupts have to be masked. The single combined interrupt does not have its own mask bit.

Whenever Port A is configured for interrupts, the data direction must be set to Input. If the data direction register is reprogrammed to Output, then any pending interrupts are not lost.However, no new interrupts are generated.

For edge-detected interrupts, the ISR can clear the interrupt by writing a 1 to the GPIO_PORTA_EOI register for the corresponding bit to disable the interrupt. This write also clears the interrupt status and raw status registers. Writing to the GPIO_PORTA_EOI register has no effect on level-sensitive interrupts. If level-sensitive interrupts cause the processor to interrupt, then the ISR can poll

the GPIO_INT_RAWSTATUS register until the interrupt source disappears, or it can write to the GPIO_INTMASK register to mask the interrupt before exiting the ISR. If the ISR exits without masking or disabling the interrupt prior to exiting, then the level-sensitive interrupt repeatedly requests an interrupt until the interrupt is cleared at the source.

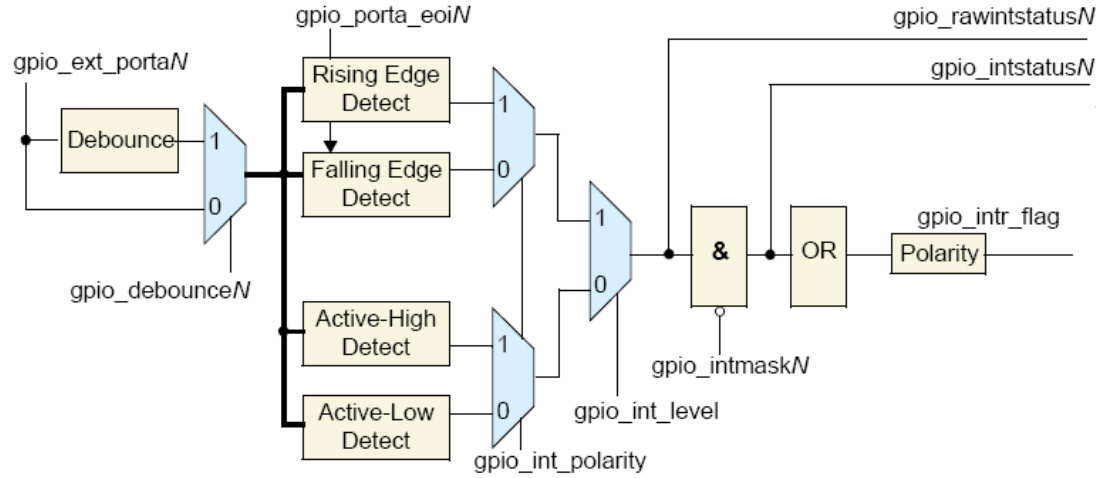


Fig.34-2 GPIO Interrupt RTL Block Diagram

**Debounce operation**

Port A has been configured to include the debounce capability interrupt feature. The external signal can be debounced to remove any spurious glitches that are less than one period of the external debouncing clock.

When input interrupt signals are debounced using a debounce clock(pclk), the signals must be active for a minimum of two cycles of the debounce clock to guarantee that they are registered. Any input pulse widths less than a debounce clock period are bounced. A pulse width between one and two debounce clock widths may or may not propagate, depending on its phase relationship to the debounce clock. If the input pulse spans two rising edges of the debounce clock, it is registered. If it spans only one risingedge, it is not registered.

**Synchronization of Interrupt Signals to the System Clock**

Interrupt signals are internally synchronized to pclk. Synchronization topclk must occur for edge-detect signals. With level-sensitive interrupts, synchronization is optional andunder software control(GPIO_LS_SYNC).

## 34.3.2 Programming

**Programming Considerations**

- Reading from an unused location or unused bits in a particular register always returns zeros. Thereis no error mechanism in the APB.
- Programming the GPIO registers for interrupt capability, edge-sensitive or level-sensitiveinterrupts, and interrupt polarity should be completed prior to enabling the interrupts on PortA inorder to prevent spurious glitches on the interrupt lines to the interrupt controller.
- Writing to the interrupt clear register clears an edge-detected interrupt and has no effect on alevel-sensitive interrupt.

**4 GPIOs' hierarchy in the chip**

GPIO1, GPIO2 are in cpu subsystem, GPIO3 is in peripheral subsystem, and GPIO0 is in alive subsystem.

## 34.4 Register Description

This section describes the control/status registers of the design. Software should read and write these registers using 32-bits accesses. There are 4 GPIOs (GPIO0 ~ GPIO3), and each of them has same register group. Therefore, 4 GPIOs' register groups have 4 different base address.

### 34.4.1 Registers Summary

| Name | Offset | Size | Reset Value | Description |
|---|---|---|---|---|
| GPIO_SWPORTA_DR | 0x0000 | W | 0x00000000 | Port A data register |
| GPIO_SWPORTA_DDR | 0x0004 | W | 0x00000000 | Port A data direction register |
| GPIO_INTEN | 0x0030 | W | 0x00000000 | Interrupt enable register |
| GPIO_INTMASK | 0x0034 | W | 0x00000000 | Interrupt mask register |
| GPIO_INTTYPE_LEVEL | 0x0038 | W | 0x00000000 | Interrupt level register |
| GPIO_INT_POLARITY | 0x003c | W | 0x00000000 | Interrupt polarity register |
| GPIO_INT_STATUS | 0x0040 | W | 0x00000000 | Interrupt status of port A |
| GPIO_INT_RAWSTATUS | 0x0044 | W | 0x00000000 | Raw Interrupt status of port A |
| GPIO_DEBOUNCE | 0x0048 | W | 0x00000000 | Debounce enable register |
| GPIO_PORTA_EOI | 0x004c | W | 0x00000000 | Port A clear interrupt register |
| GPIO_EXT_PORTA | 0x0050 | W | 0x00000000 | Port A external port register |
| GPIO_LS_SYNC | 0x0060 | W | 0x00000000 | Level_sensitive synchronization enable register |

*Notes:Size:**B**- Byte (8 bits) access, **HW**- Half WORD (16 bits) access, **W**-WORD (32 bits) access*

### 34.4.2 Detail Register Description

**GPIO_SWPORTA_DR**
Address: Operational Base + offset (0x0000)
Port A data register

| Bit | Attr | Reset Value | Description |
|---|---|---|---|
| 31:0 | RW | 0x00000000 | gpio_swporta_dr<br>Values written to this register are output on the I/O signals for Port A if the corresponding data direction bits for Port A are set to Output mode.The value read back is equal to the last value written to this register. |

**GPIO_SWPORTA_DDR**
Address: Operational Base + offset (0x0004)
Port A data direction register

| Bit | Attr | Reset Value | Description |
|---|---|---|---|

| Bit | Attr | Reset Value | Description |
|---|---|---|---|
| 31:0 | RW | 0x00000000 | gpio_swporta_ddr<br>Values written to this register independently control the direction of the corresponding data bit in Port A.<br>1'b0: Input (default)<br>1'b1: Output |

**GPIO_INTEN**

Address: Operational Base + offset (0x0030)

Interrupt enable register

| Bit | Attr | Reset Value | Description |
|---|---|---|---|
| 31:0 | RW | 0x00000000 | gpio_int_en<br>Allows each bit of Port A to be configured for interrupts. Whenever a 1 is written to a bit of this register, it configures the corresponding bit on Port A to become an interrupt; otherwise, Port A operates as a normal GPIO signal.<br>Interrupts are disabled on the corresponding bits of Port A if the corresponding data direction register is set to Output.<br>1'b0: Configure Port A bit as normal GPIO signal (default)<br>1'b1: Configure Port A bit as interrupt |

**GPIO_INTMASK**

Address: Operational Base + offset (0x0034)

Interrupt mask register

| Bit | Attr | Reset Value | Description |
|---|---|---|---|
| 31:0 | RW | 0x00000000 | gpio_int_mask<br>Controls whether an interrupt on Port A can create aninterrupt for the interrupt controller by not masking it. Whenever a 1 is written to a bit in this register, it masks the interrupt generation capability for this signal; otherwise interrupts are allowed through.<br>1'b0: Interrupt bits are unmasked (default)<br>1'b1: Mask interrupt |

**GPIO_INTTYPE_LEVEL**

Address: Operational Base + offset (0x0038)

Interrupt level register

| Bit | Attr | Reset Value | Description |
|---|---|---|---|
| 31:0 | RW | 0x00000000 | gpio_inttype_level<br>Controls the type of interrupt that can occur on Port A.<br>1'b0: Level-sensitive (default)<br>1'b1: Edge-sensitive |

**GPIO_INT_POLARITY**
Address: Operational Base + offset (0x003c)
Interrupt polarity register

| Bit | Attr | Reset Value | Description |
|---|---|---|---|
| 31:0 | RW | 0x00000000 | gpio_int_polarity<br>Controls the polarity of edge or level sensitivity that can occur on input of Port A.<br>1'b0: Active-low (default)<br>1'b1: Active-high |

**GPIO_INT_STATUS**
Address: Operational Base + offset (0x0040)
Interrupt status of port A

| Bit | Attr | Reset Value | Description |
|---|---|---|---|
| 31:0 | RO | 0x00000000 | gpio_int_status<br>Interrupt status of Port A |

**GPIO_INT_RAWSTATUS**
Address: Operational Base + offset (0x0044)
Raw Interrupt status of port A

| Bit | Attr | Reset Value | Description |
|---|---|---|---|
| 31:0 | RO | 0x00000000 | gpio_int_rawstatus<br>Raw interrupt of status of Port A (premasking bits |

**GPIO_DEBOUNCE**
Address: Operational Base + offset (0x0048)
Debounce enable register

| Bit | Attr | Reset Value | Description |
|---|---|---|---|
| 31:0 | RW | 0x00000000 | gpio_debounce<br>Controls whether an external signal that is the sourceof an interrupt needs to be debounced to remove anyspurious glitches. Writing a 1 to a bit in this registerenables the debouncing circuitry. A signal must bevalid for two periods of an external clock before it isinternally processed.<br>1'b0: No debounce (default)<br>1'b1: Enable debounce |

**GPIO_PORTA_EOI**
Address: Operational Base + offset (0x004c)
Port A clear interrupt register

| Bit | Attr | Reset Value | Description |
|---|---|---|---|

| Bit | Attr | Reset Value | Description |
|---|---|---|---|
| 31:0 | WO | 0x00000000 | gpio_porta_eoi<br>Controls the clearing of edge type interrupts from Port A. When a 1 is written into a corresponding bit of this register, the interrupt is cleared. All interrupts are cleared when Port A is not configured for interrupts.<br>1'b0: No interrupt clear (default)<br>1'b1: Clear interrupt |

**GPIO_EXT_PORTA**
Address: Operational Base + offset (0x0050)
Port A external port register

| Bit | Attr | Reset Value | Description |
|---|---|---|---|
| 31:0 | RO | 0x00000000 | gpio_ext_porta<br>When Port A is configured as Input, then reading thislocation reads the values on the signal. When the datadirection of Port A is set as Output, reading this location reads the data register for Port A. |

**GPIO_LS_SYNC**
Address: Operational Base + offset (0x0060)
Level_sensitive synchronization enable register

| Bit | Attr | Reset Value | Description |
|---|---|---|---|
| 31:1 | RO | 0x0 | reserved |
| 0 | RW | 0x0 | gpio_ls_sync<br>Writing a 1 to this register results in all level-sensitive interrupts being synchronized to pclk_intr.<br>1'b0: No synchronization to pclk_intr (default)<br>1'b1: Synchronize to pclk_intr |

## 34.5 Interface description

Table 34-1GPIO interface description

| Module Pin | Dir | Pad Name | IOMUX Setting |
|---|---|---|---|
| GPIO0 Interface | | | |
| gpio0_porta[7:0] | I/O | GPIO0_A[7:0] | GRF_GPIO0A_IOMUX[15:0] =16'h0000 |
| gpio0_porta[15:8] | I/O | GPIO0_B[7:0] | GRF_GPIO0B_IOMUX[15:0] =16'h0000 |
| gpio0_porta[23:16] | I/O | GPIO0_C[7:0] | GRF_GPIO0C_IOMUX[15:0] =16'h0000 |
| gpio0_porta[31:24] | I/O | GPIO0_D[7:0] | GRF_GPIO0D_IOMUX[15:0] =16'h0000 |
| GPIO1 Interface | | | |
| gpio1_porta[7:0] | I/O | GPIO1_A[7:0] | GRF_GPIO1A_IOMUX[15:0] =16'h0000 |
| gpio1_porta[15:8] | I/O | GPIO1_B[7:0] | GRF_GPIO1B_IOMUX[15:0] |

| | | | |
|---|---|---|---|
| | | | =16'h0000 |
| gpio1_porta[23:16] | I/O | GPIO1_C[7:0] | GRF_GPIO1C_IOMUX[15:0] =16'h0000 |
| gpio1_porta[31:24] | I/O | GPIO1_D[7:0] | GRF_GPIO1D_IOMUX[15:0] =16'h0000 |
| GPIO2 Interface | | | |
| gpio2_porta[7:0] | I/O | GPIO2_A[7:0] | GRF_GPIO2A_IOMUX[15:0] =16'h0000 |
| gpio2_porta[15:8] | I/O | GPIO2_B[7:0] | GRF_GPIO2B_IOMUX[15:0] =16'h0000 |
| gpio2_porta[23:16] | I/O | GPIO2_C[7:0] | GRF_GPIO2C_IOMUX[7:0]= 8'h00 GRF_GPIO2C_IOMUX2[15:0 ]=16'h0000 |
| gpio2_porta[31:24] | I/O | GPIO2_D[7:0] | GRF_GPIO2D_IOMUX[15:0] =16'h0000 |
| GPIO3 Interface | | | |
| gpio3_porta[7:0] | I/O | GPIO3_A[7:0] | GRF_GPIO3A_IOMUX[15:0] =16'h0000 |
| gpio3_porta[15:8] | I/O | GPIO3_B[7:0] | GRF_GPIO3B_IOMUX[15:0] =16'h0000 |
| gpio3_porta[23:16] | I/O | GPIO3_C[7:0] | GRF_GPIO3C_IOMUX[15:0] =16'h0000 |
| gpio3_porta[31:24] | I/O | GPIO3_D[7:0] | GRF_GPIO3D_IOMUX[15:0] =16'h0000 |

## 34.6 Application Notes

**Steps to set GPIO's direction**

- Write GPIO_SWPORT_DDR[x] as 1 to set this gpio as output direction and Write GPIO_SWPORT_DDR[x] as 0 to set this gpio as input direction.
- Default GPIO's direction is input direction.

**Steps to set GPIO's level**

- Write GPIO_SWPORT_DDR[x] as 1 to set this gpio as output direction.
- Write GPIO_SWPORT_DR[x] as v to set this GPIO's value.

**Steps to get GPIO's level**

- Write GPIO_SWPORT_DDR[x] as 0 to set this gpio as input direction.
- Read from GPIO_EXT_PORT[x] to get GPIO's value

**Steps to set GPIO as interrupt source**

- Write GPIO_SWPORT_DDR[x] as 0 to set this gpio as input direction.
- Write GPIO_INTTYPE_LEVEL[x] as v1 and write GPIO_INT_POLARITY[x] as v2 to set interrupt type
- Write GPIO_INTEN[x] as 1 to enable GPIO's interrupt

Note: Please switch iomux to GPIO mode first!