1. General Purpose Input/Output (GPIO)

The QN9020/1 processor provides 31/15 highly-multiplexed general-purpose I/O (GPIO) pins for use in generating and capturing application-specific input and output signals. Each pin can be programmed as an output, an input, or as bidirectional for certain alternate functions (that override the value programmed in the GPIO direction registers). When programmed as an input, a GPIO can also serve as an interrupt source. All GPIO pins are configured as inputs with pull-up during the assertion of all resets, and they remain inputs until configured otherwise. In addition, select special-function GPIO pins serve as bidirectional pins where the I/O direction is driven from the respective unit (overriding the GPIO direction register).

To minimize power consumption, configure all unused GPIOs as outputs/digital input and connect to GND.

GPIO pins may have alternate input and output functions. A pin may serve either as GPIO or as an alternate function, but not as both at the same time.

1.1 Features

- Programmable interrupt generation capability
- Registers for alternate function switching with pin multiplexing support
- Inputs are sampled using a double flip-flop to avoid meta-stability issues.
- All ports have programmable internal pull-up/pull-down/high-z.
- As output, the GPIOs can be individually cleared or set.

1.2 Functional Description

The GPIO signals operate as either general-purpose I/O or as one of their alternate functions.
This section describes operation in both modes.

1.2.1 GPIO Operation as Application-Specific GPIO

Use the GPIO Pin Output Configuration registers OUTENABLESET and OUTENABLECLR to program the GPIO pins as inputs or outputs. For a pin configured as an output, write to the Data Output Register DATAOUT to set the pin value. If a pin is configured as an input, the programmed output state occurs when the pin is reconfigured as an output.
To validate the state of a GPIO pin, read the GPIO Pin Output Enable register OUTENABLESET. To get the Value of a GPIO pin, can read the Data Value register DATA. Software can read these two registers at any time, even if the pin is configured as an output.

To detect either a rising or a falling edge on each GPIO pin programmed as an digital input, use the GPIO Intererrupt Type Set register INTTYPESET. To program these edge-detects to generate interrupts, use the Intererrupt Enable register INTENSET, and use INTSTATUS to read IRQ status.

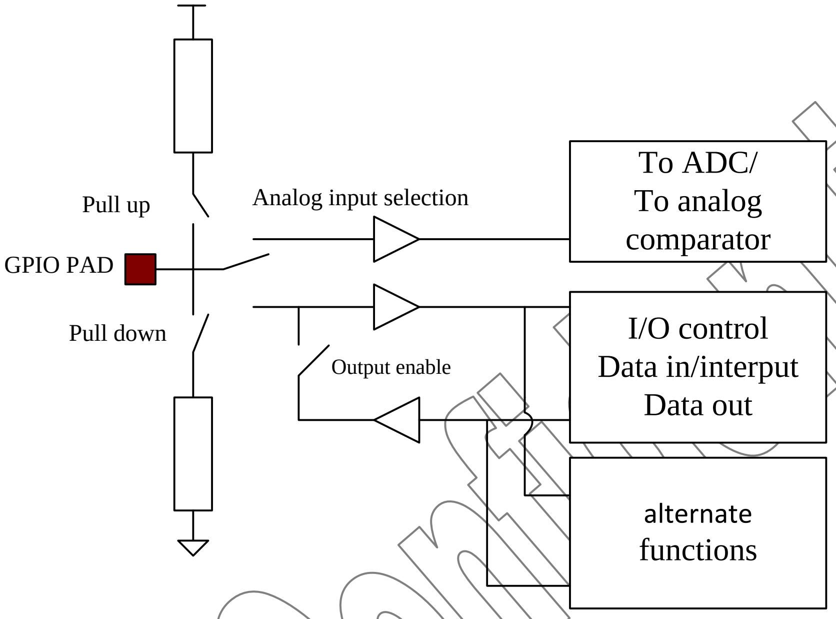
Most GPIO pins are multiplexed with alternate functions of the QN902x processor. Certain modes within the serial controllers require extra pins. These functions are externally available through specific GPIO pins, and their use is described in the following paragraphs. Details on alternate functions are provided in end of the section.

This MASKBYTExxTOxx register masks the read and/or write accesses to the masked DATAOUT register. Only bits set to 1 in the MASK register enable the corresponding bits in the masked registers to be changed or their value to be read.

Setting any mask bit to 1 allows the pin output to be changed by write operations to the pin’s DATAOUT registers. The current state of the pin can be read from the OUTENABLESET registers and the current value of the DATAOUT registers can be read.

Setting any mask bit to 0 allows write operations to the pin’s DATAOUT registers to have no effect on the pin’s output level. Read operations return 0 regardless of the pin’s level or the value of the DATAOUT register.

The figure following shows a block diagram of a single GPIO pin.



1.2.2 GPIO Operation as Alternate Function

GPIO pins can have as many as three alternate input and three alternate output functions. If a GPIO is used for an alternate function, then it cannot be used as a GPIO at the same time. When using an alternate function of a GPIO signal, first configure the alternate function and then enable the corresponding unit. Also, disable the unit prior to changing the alternate function signals of the GPIO.

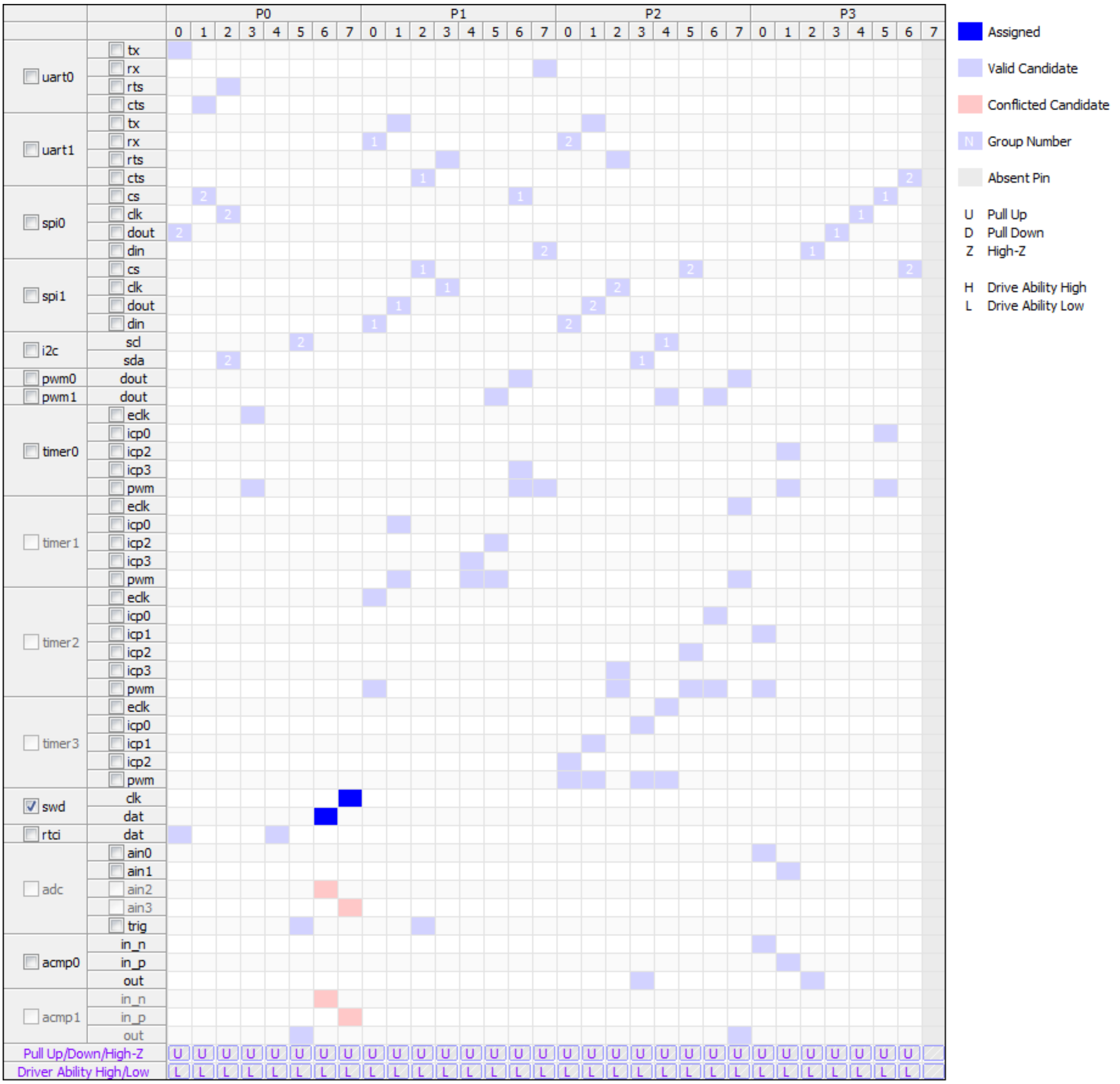
For example, P0_2 can be configured as I2CSDA if it is configured for alternate function and it is configured as an input. Since this is a special-function bidirectional GPIO, I2CSDA functions as an input or as an output depending on how this signal is configured in the i2c controller, not how it is configured in the GPIO direction register. To configure I2CSDA as an output on P0_2, the GPIO Direction register must be configured as an *input*, the GPIO Alternate Function register must select alternate function and the i2c

Control register 1 must configure I2CSDA as an output. If however, P0_2 was configured as an output and Alternate Function was configured in the Alternate Function register, P0_2 would function as SPICLK0. The other combinations of direction and alternate function selection for this signal function normally.

The table follows shows the alternate mapping of all the GPIOs.

PIN	0	1	10	11	Test port
P0_0	GPIO0	UART0_TXD(O)	SPI0_DAT(I/O)	RTCI(I)	Test_pin[3]
P0_1	GPIO1	NC	SPI0_CS0(I/O)	UART0_CTSn(I)	NC
P0_2	GPIO2	I2C_SDA(I/O)	SPI0_CLK(I/O)	UART0_RTSn(O)	NC
P0_3	GPIO3	RADIO_EN(O)	CLKOUT0(O)	TIMER0_eclk(I/O)	Test_pin[2]
P0_4	GPIO4	NC	CLKOUT1(O)	RTCI(I)	NC
P0_5	GPIO5	I2C_SCL(I/O)	ADCT(I)	ACMP1_O(O)	NC
P0_6	SW_DAT	GPIO6	AIN2(AI)	ACMP1-(AI)	Test_pin[1]
P0_7	SW_CLK	GPIO7	AIN3(AI)	ACMP1+(AI)	Test_pin[0]
P1_0	GPIO8	SPI1_DIN(I)	UART1_RXD(I)	TIMER2_eclk(I/O)	Test_pin[8]
P1_1	GPIO9	SPI1_DAT(I/O)	UART1_TXD(O)	TIMER1_0(I/O)	Test_pin[7]
P1_2	GPIO10	SPI1_CS0(I/O)	UART1_CTSn(I)	ADCT(I)	Test_pin[6]
P1_3	GPIO11	SPI1_CLK(I/O)	UART1_RTSn(O)	CLKOUT1(O)	Test_pin[5]
P1_4	GPIO12	RDYN(O)	NC	TIMER1_3(I/O)	NC
P1_5	GPIO13	RADIO_EN(O)	PWM1(O)	TIMER1_2(I/O)	NC
P1_6	GPIO14	SPI0_CS1_O(O)	PWM0(O)	TIMER0_3(I/O)	NC
P1_7	GPIO15	UART0_RXD(I)	SPI0_DIN (I)	TIMER0_o(O)	Test_pin[4]
P2_0	GPIO 16	SPI1_DIN(I)	UART1_RXD(I)	TIMER3_2(I/O)	NC
P2_1	GPIO 17	SPI1_DAT(I/O)	UART1_TXD(O)	TIMER3_1(I/O)	NC
P2_2	GPIO 18	SPI1_CLK(I/O)	UART1_RTSn(O)	TIMER2_3(I/O)	NC
P2_3	GPIO 19	I2C_SDA(I/O)	ACMP0_O(O)	TIMER3_0(I/O)	Test_pin[12]
P2_4	GPIO 20	I2C_SCL(I/O)	PWM1(O)	TIMER3_eclk(I/O)	Test_pin[11]
P2_5	GPIO 21	SPI1_CS1_O(O)	NC	TIMER2_2(I/O)	NC
P2_6	GPIO 22	Antenna_O (O)	PWM1(O)	TIMER2_0(I/O)	Test_pin[10]
P2_7	GPIO 23	ACMP1_O(O)	PWM0(O)	TIMER1_eclk(I/O)	Test_pin[9]
P3_0	GPIO 24	TIMER2_1(I/O)	AIN0(AI)	ACMP0-(AI)	Test_pin[14]
P3_1	GPIO 25	TIMER0_2(I/O)	AIN1(AI)	ACMP0+(AI)	Test_Pin[13]
P3_2	GPIO 26	SPI0_DIN (I)	NC	ACMP0_O(O)	Test_Pin[1]
P3_3	GPIO 27	SPI0_DAT(I/O)	CLKOUT0(O)	NC	Test_Pin[0]
P3_4	GPIO 28	SPI0_CLK(I/O)	NC	NC	NC
P3_5	GPIO 29	SPI0_CS0(I/O)	INT_FM(I)	TIMER0_0(I/O)	NC
P3_6	GPIO 30	SPI1_CS0(I/O)	UART1_CTSn(I)	NC	NC

User can QnDriverTools in SDK to configure GPIO:



		P0								P1								P2								P3							
		0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7
☐ uart0	☐ tx	▨																															░
	☐ rx																▨																░
	☐ rts			▨																													░
	☐ cts		▨																														░
☐ uart1	☐ tx										▨								▨														░
	☐ rx									1								2															░
	☐ rts												▨							▨													░
	☐ cts											1																				2	░
☐ spi0	☐ cs		2													1															1		░
	☐ clk			2																										1			░
	☐ dout	2																											1				░
	☐ din																2											1					░
☐ spi1	☐ cs											1											2									2	░
	☐ clk												1							2													░
	☐ dout										1								2														░
	☐ din									1								2															░
☐ i2c	scl						2															1											░
	sda			2																	1												░
☐ pwm0	dout															▨									▨								░
☐ pwm1	dout														▨							▨		▨									░
☐ timer0	☐ eclk				▨																												░
	☐ icp0																														▨		░
	☐ icp2																										▨						░
	☐ icp3															▨																	░
	☐ pwm				▨											▨	▨										▨				▨		░
☐ timer1	☐ eclk																								▨								░
	☐ icp0										▨																						░
	☐ icp2														▨																		░
	☐ icp3													▨																			░
	☐ pwm										▨			▨	▨										▨								░
☐ timer2	☐ eclk									▨																							░
	☐ icp0																							▨									░
	☐ icp1																									▨							░
	☐ icp2																						▨										░
	☐ icp3																			▨													░
	☐ pwm									▨										▨			▨	▨		▨							░
☐ timer3	☐ eclk																					▨											░
	☐ icp0																				▨												░
	☐ icp1																		▨														░
	☐ icp2																	▨															░
	☐ pwm																	▨	▨		▨	▨											░
☑ swd	clk								■																								░
	dat							■																									░
☐ rtci	dat	▨				▨																											░
☐ adc	☐ ain0																									▨							░
	☐ ain1																										▨						░
	☐ ain2							🟥																									░
	☐ ain3								🟥																								░
	☐ trig						▨					▨																					░
☐ acmp0	in_n																									▨							░
	in_p																										▨						░
	out																				▨							▨					░
☐ acmp1	in_n							🟥																									░
	in_p								🟥																								░
	out						▨																		▨								░
Pull Up/Down/High-Z		U	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U	░
Driver Ability High/Low		L	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L	░

■	Assigned
▨	Valid Candidate
🟥	Conflicted Candidate
N	Group Number
░	Absent Pin
U	Pull Up
D	Pull Down
Z	High-Z
H	Drive Ability High
L	Drive Ability Low

1.2.3 Programming flow

When reset, all GPIO is digital input (pull-up). Port I/O initialization consists of the following steps:

Step1. Configure all pins, in turn, as Application-Specific GPIO or as Alternate Function.
Step2. Configure all pins's direction, input or output.
Step3. Configure the alternate fuction's pins selection.
Step4. Configure pin's driver ability, high or low.
Step5. Configure pins as Pull-down, Pull-high or Reserved.

Step6. Enable corresponding alternate fuction unit.

To minimize power consumption, configure all unused GPIOs as digital input with low driver ability and Pull-down or Reserved, and connect to GND.

1.3 Register Description

1.3.1 Register Map

The GPIO register base address is 0x50000000.

Table 1 Register Map

Offset	Name	Description
000h	DATA	Data value [31:0]: Read Sampled at pin. Read back value is going through double flip-flop synchronization logic with a delay of two cycles.
004h	DATAOUT	Data output Register value [31:0]: Write output data. Read Current value of data output register.
010h	OUTENABLESET	Output enable set [31:0]: Write: 1 Set the output enable bit. Configure the pin as output. 0 No effect. Read: 0 Indicate the signal direction as input. 1 Indicate the signal direction as output.
014h	OUTENABLECLR	Output enable clear [31:0]: Write: 1 Clear the output enable bit. Configure the pin as input. 0 No effect. Read: 0 Indicate the signal direction as input. 1 Indicate the signal direction as output.
018h	reserved	reserved
01Ch	reserve	reserve
020h	INTENSET	Interrupt enable set [31:0]: Write 1 Set the enable bit. 0 No effect. Read back 0 Interrupt disabled. 1 Interrupt enabled.
024h	INTENCLR	Interrupt enable clear [31:0]: Write 1 Clear the enable bit. 0 No effect. Read back 0 Interrupt disabled. 1 Interrupt enabled.
028h	INTTYPESET	Interrupt type set [31:0]: Write 1 Set the interrupt type bit. 0 No effect.

		Read back 0 For LOW or HIGH level. 1 For falling edge or rising edge.
02Ch	INTTYPECLR	Interrupt type clear [31:0]: Write 1 Clear the interrupt type bit. 0 No effect. Read back 0 For LOW or HIGH level. 1 For falling edge or rising edge.
030h	INTPOLSET	Polarity-level, edge IRQ configuration [31:0]: Write 1 Set the interrupt polarity bit. 0 No effect. Read back 0 For LOW level or falling edge. 1 For HIGH level or rising edge.
034h	INTPOLCLR	Polarity-level, edge IRQ configuration [31:0]: Write 1 Clear the interrupt polarity bit. 0 No effect. Read back 0 For LOW level or falling edge. 1 For HIGH level or rising edge.
038h	INTSTATUS	IRQ status clear Register [31:0] Write 1 To clear the interrupt request. 0 No effect. Read back [31:0] IRQ status Register.
400h~7FCh	MASKBYTE7TO0	Bits [9:2] of the address value are used as enable bit mask for the access [31:8] Not used. Write is ignored, and read as 0. [7:0] Data for lower byte access, with [9:2] of address value use as enable mask for each bit.
800h~BFCh	MASKBYTE15TO8	Bits [9:2] of the address value are used as enable bit mask for the access: [31:16],[7:0] Not used. Write is ignored, and read as 0. [15:8] Data for lower byte access, with [9:2] of address value use as enable mask for each bit.
000h~3FCh	MASKBYTE23TO16	Bits [9:2] of the address value are used as enable bit mask for the access: [31:24],[15:0] Not used. Write is ignored, and read as 0. [23:16] Data for lower byte access, with [9:2] of address value use as enable mask for each bit.
400h~7FCh	MASKBYTE31TO24	Bits [9:2] of the address value are used as enable bit mask for the access: [23:0] Not used. Write is ignored, and read as 0. [31:24] Data for lower byte access, with [9:2] of address value use as enable mask for each bit.

1.3.2 Register Description

Table 2 DATA

Bit	Type	Reset	Symbol	Description
31-0	RWH	-	DATA[31-0]	Data value

Table 3 DATAOUT

Bit	Type	Reset	Symbol	Description
31-0	RWH	0	DATAOUT[31-0]	Data output Register value

Table 4 OUTENABLESET

Bit	Type	Reset	Symbol	Description
31-0	RW1	0	OUTENABLESET[31-0]	Output enable set

Table 5 OUTENABLECLR

Bit	Type	Value	Symbol	Description
31-0	RW1	0	OUTENABLECLR[31-0]	Output enable clear

Table 6 INTENSET

Bit	Type	Reset	Symbol	Description
31-0	RW1	0	INTENSET[31-0]	Interrupt enable set

Table 7 INTENCLR

Bit	Type	Reset	Symbol	Description
31-0	RW1	0	INTENCLR[31-0]	Interrupt enable clear

Table 8 INTTYPESET

Bit	Type	Reset	Symbol	Description
31-0	RW1	0	INTTYPESET[31-0]	Interrupt type set

Table 9 INTTYPECLR

Bit	Type	Reset	Symbol	Description
31-0	RW1	0	INTTYPECLR[31-0]	Interrupt type clear

Table 10 INTPOLSET

Bit	Type	Reset	Symbol	Description
31-0	RW1	0	INTPOLSET[31-0]	Polarity-level, edge IRQ configuration

Table 11 INTPOLCLR

Bit	Type	Reset	Symbol	Description
31:0	RW1	0	INTPOLCLR[31-0]	Polarity-level, edge IRQ configuration [31-0]

Table 12 INTSTATUS

Bit	Type	Reset	Symbol	Description
31-0	RWH	0	INTSTATUS[31-0]	Write one to clear interrupt request