**Overview**

This document specifies GPIO hardware IP functionality. This module conforms to the [Comportable guideline for peripheral device functionality](https://github.com/lowRISC/opentitan/blob/master/doc/contributing/hw/comportability/README.md) See that document for integration overview within the broader top level system.

**Features**

* 32 GPIO ports
* Configurable interrupt per GPIO for detecting rising edge, falling edge, or active low/high input
* Two ways to update GPIO output: direct-write and masked (thread-safe) update

**Description**

The GPIO block allows software to communicate through general purpose I/O pins in a flexible manner. Each of 32 independent bits can be written as peripheral outputs in two modes. Each of the 32 bits can be read by software as peripheral inputs. How these peripheral inputs and outputs are connected to the chip IO is not within the scope of this document. See the Comportability Specification for peripheral IO options at the top chip level.

In the output direction, this module provides direct 32b access to each GPIO value using direct write. This mode allows software to control all GPIO bits simultaneously. Alternately, this module provides masked writes to half of the bits at a time, allowing software to affect the output value of a subset of the bits without requiring a read-modify-write. In this mode the user provides a mask of which of the 16 bits are to be modified, along with their new value. The details of this mode are given in the [Programmers Guide](https://github.com/lowRISC/opentitan/tree/master/hw/ip/gpio#programmers-guide) below.

In the input direction, software can read the contents of any of the GPIO peripheral inputs. In addition, software can request the detection of an interrupt event for any of the 32 bits in a configurable manner. The choices are positive edge, negative edge or level detection events. A noise filter is available through configuration for any of the 32 GPIO inputs. This requires the input to be stable for 16 cycles of the module clock before the input register reflects the change and interrupt generation is evaluated. Note that if the filter is enabled and the pin is set to output then there will be a corresponding delay in a change in output value being reflected in the input register.

See the Design Details section for more details on output, input, and interrupt control.

**Theory of Operation**

The block diagram above shows the DATA\_OUT and DATA\_OE registers managed by hardware outside of the auto-generated register file. For reference, it also shows the assumed connections to pads in the top level netlist.

**Design Details**

The GPIO module maintains one 32-bit output register DATA\_OUT with two ways to write to it. Direct write access uses [DIRECT\_OUT](https://github.com/lowRISC/opentitan/blob/master/hw/ip/gpio/doc/registers.md#direct_out), and masked access uses [MASKED\_OUT\_UPPER](https://github.com/lowRISC/opentitan/blob/master/hw/ip/gpio/doc/registers.md#masked_out_upper) and [MASKED\_OUT\_LOWER](https://github.com/lowRISC/opentitan/blob/master/hw/ip/gpio/doc/registers.md#masked_out_lower). Direct access provides full write and read access for all 32 bits in one register.

For masked access the bits to modify are given as a mask in the upper 16 bits of the [MASKED\_OUT\_UPPER](https://github.com/lowRISC/opentitan/blob/master/hw/ip/gpio/doc/registers.md#masked_out_upper) and [MASKED\_OUT\_LOWER](https://github.com/lowRISC/opentitan/blob/master/hw/ip/gpio/doc/registers.md#masked_out_lower) register write, while the data to write is provided in the lower 16 bits of the register write. The hardware updates DATA\_OUT with the mask so that the modification is done without software requiring a Read-Modify-Write.

Reads of masked registers return the lower/upper 16 bits of the DATA\_OUT contents. Zeros are returned in the upper 16 bits (mask field). To read what is on the pins, software should read the [DATA\_IN](https://github.com/lowRISC/opentitan/blob/master/hw/ip/gpio/doc/registers.md#data_in) register. (See [GPIO Input](https://github.com/lowRISC/opentitan/blob/master/hw/ip/gpio/doc/theory_of_operation.md#gpio-input) section below).

The same concept is duplicated for the output enable register DATA\_OE. Direct access uses [DIRECT\_OE](https://github.com/lowRISC/opentitan/blob/master/hw/ip/gpio/doc/registers.md#direct_oe), and masked access is available using [MASKED\_OE\_UPPER](https://github.com/lowRISC/opentitan/blob/master/hw/ip/gpio/doc/registers.md#masked_oe_upper) and [MASKED\_OE\_LOWER](https://github.com/lowRISC/opentitan/blob/master/hw/ip/gpio/doc/registers.md#masked_oe_lower).

The output enable is sent to the pad control block to determine if the pad should drive the DATA\_OUT value to the associated pin or not.

A typical use pattern is for initialization and suspend/resume code to use the full access registers to set the output enables and current output values, then switch to masked access for both DATA\_OUT and DATA\_OE.

For GPIO outputs that are not used (either not wired to a pin output or not selected for pin multiplexing), the output values are disconnected and have no effect on the GPIO input, regardless of output enable values.

**GPIO Input**

The [DATA\_IN](https://github.com/lowRISC/opentitan/blob/master/hw/ip/gpio/doc/registers.md#data_in) register returns the contents as seen on the peripheral input, typically from the pads connected to those inputs. In the presence of a pin-multiplexing unit, GPIO peripheral inputs that are not connected to a chip input will be tied to a constant zero input.

The GPIO module provides optional independent noise filter control for each of the 32 input signals. Each input can be independently enabled with the [CTRL\_EN\_INPUT\_FILTER](https://github.com/lowRISC/opentitan/blob/master/hw/ip/gpio/doc/registers.md#ctrl_en_input_filter) (one bit per input). This 16-cycle filter is applied to both the [DATA\_IN](https://github.com/lowRISC/opentitan/blob/master/hw/ip/gpio/doc/registers.md#data_in) register and the interrupt detection logic. The timing for [DATA\_IN](https://github.com/lowRISC/opentitan/blob/master/hw/ip/gpio/doc/registers.md#data_in) is still not instantaneous if [CTRL\_EN\_INPUT\_FILTER](https://github.com/lowRISC/opentitan/blob/master/hw/ip/gpio/doc/registers.md#ctrl_en_input_filter) is false as there is top-level routing involved, but no flops are between the chip input and the [DATA\_IN](https://github.com/lowRISC/opentitan/blob/master/hw/ip/gpio/doc/registers.md#data_in) register.

The contents of [DATA\_IN](https://github.com/lowRISC/opentitan/blob/master/hw/ip/gpio/doc/registers.md#data_in) are always readable and reflect the value seen at the chip input pad regardless of the output enable setting from DATA\_OE. If the output enable is true (and the GPIO is connected to a chip-level pad), the value read from [DATA\_IN](https://github.com/lowRISC/opentitan/blob/master/hw/ip/gpio/doc/registers.md#data_in) includes the effect of the peripheral's driven output (so will only differ from DATA\_OUT if the output driver is unable to switch the pin or during the delay imposed if the noise filter is enabled).

**Interrupts**

The GPIO module provides 32 interrupt signals to the main processor. Each interrupt can be independently enabled, tested, and configured. Following the standard interrupt guidelines in the [Comportability Specification](https://github.com/lowRISC/opentitan/blob/master/doc/contributing/hw/comportability/README.md), the 32 bits of the [INTR\_ENABLE](https://github.com/lowRISC/opentitan/blob/master/hw/ip/gpio/doc/registers.md#intr_enable) register determines whether the associated inputs are configured to detect interrupt events. If enabled via the various INTR\_CTRL\_EN registers, their current state can be read in the [INTR\_STATE](https://github.com/lowRISC/opentitan/blob/master/hw/ip/gpio/doc/registers.md#intr_state) register. Clearing is done by writing a 1 into the associated [INTR\_STATE](https://github.com/lowRISC/opentitan/blob/master/hw/ip/gpio/doc/registers.md#intr_state) bit field.

For configuration, there are 4 types of interrupts available per bit, controlled with four control registers. [INTR\_CTRL\_EN\_RISING](https://github.com/lowRISC/opentitan/blob/master/hw/ip/gpio/doc/registers.md#intr_ctrl_en_rising) configures the associated input for rising-edge detection. Similarly, [INTR\_CTRL\_EN\_FALLING](https://github.com/lowRISC/opentitan/blob/master/hw/ip/gpio/doc/registers.md#intr_ctrl_en_falling) detects falling edge inputs. [INTR\_CTRL\_EN\_LVLHIGH](https://github.com/lowRISC/opentitan/blob/master/hw/ip/gpio/doc/registers.md#intr_ctrl_en_lvlhigh) and [INTR\_CTRL\_EN\_LVLLOW](https://github.com/lowRISC/opentitan/blob/master/hw/ip/gpio/doc/registers.md#intr_ctrl_en_lvllow) allow the input to be level sensitive interrupts. In theory an input can be configured to detect both a rising and falling edge, but there is no hardware assistance to indicate which edge caused the output interrupt.

**Note #1:** The interrupt can only be triggered by GPIO input. **Note #2:** All inputs are sent through optional noise filtering before being sent into interrupt detection. **Note #3:** All interrupts to the processor are level interrupts as per the Comportability Specification guidelines. The GPIO module, if configured, converts an edge detection into a level interrupt to the processor core.