

Programmable Interrupt Controller

Programmable Interrupt Controller (PIC) used to increase the number of interrupt lines a controller can handle. Many controllers' families have only one interrupt input. Our design have 8 interrupt request inputs – **irq[7:0]**. The design can be programmed to mask selected interrupt lines. The design transfers the highest priority, non masked, interrupt toward a data bus **d[7:0]** by interrupt request **irq_pic**. The PIC controlled by signals:

rd, wr, rst, clk. An interrupt acknowledge signal - \overline{INTA} , enable handshaking with a processor and resets the PIC output.



Fig. 1. PIC top view

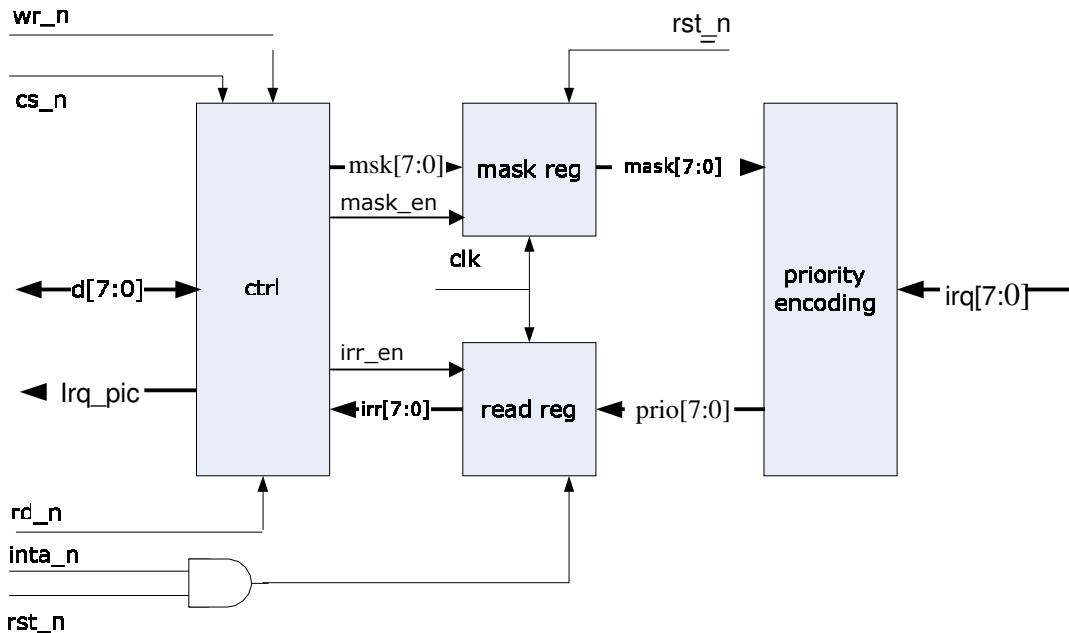


Fig. 2. PIC block diagram

Control block.

The control block determines whether the data bus-**d[7:0]**, will be:

- In input mode – when **wr_n** is low and **cs_n** is low.
- In output mode – when **rd_n** is low and **cs_n** is low.

This block generates **mask_en** and data to the **mask_reg** in write mode on **mask[7:0]**, and receives data from **read_reg** in read mode on **irr[7:0]**. In addition, this block keeps the **read_reg** value by de-assert the **irr_en** each time when the **read_reg** upholds unread value (i.e. **read_reg**≠0), otherwise this block asserted the **irq_pic**. Figure 3 presents the top view of control block.

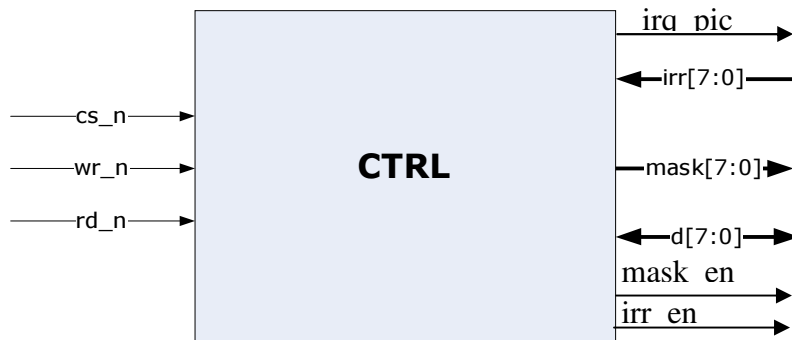


Fig. 3. Control block

Mask register and Read register.

Two register, synchronized by the same clock, each one has an enable input – **en_n**, the reset input – **rst_n**, 8 bit data input **din** and output **dout**.



Fig. 4. Register block

Priority encoder.

Generate a bus on which one bit at the most is active (high logic level). The generated data indicates which interrupt that is not masked is most prior and active. The input to this block are the **irq[7:0]** and **mask[7:0]** from the mask register, and output is **prio[7:0]** (input to read register).

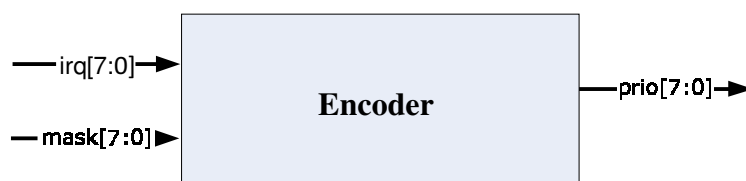


Fig. 5. Priority encoder