Ports

Overview

The 28-pin PIC24FJ64GA002 chip on the PIC24 Development Board includes a 16-bit PORTB and an 8-bit PORTA. In addition to each pin's role as either a digital input pin or a digital output pin (e.g., RB0), it can generally be configured to serve another dedicated purpose (e.g., AN2, an input to the analog-to-digital converter) or as a remappable pin (e.g., RP0). In this section we will consider only the factors that affect I/O pin use.

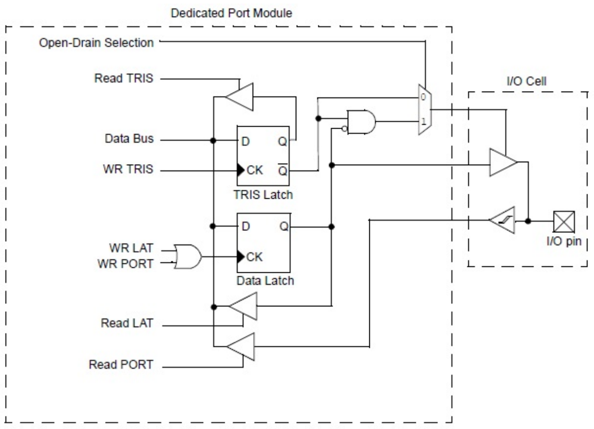


Figure PORT-1 I/O pin circuitry

Before we get started, we need to consider the initialization of the AD1PCFG register. For any pins that can serve as inputs to the analog-to-digital converter, the bits of this register default to making all of these pins serve as analog inputs, disabling their digital role. For the PIC24 Development Board, only bit zero of PORTA (i.e. RA0/AN0) is used in this way. Consequently, we will initialize

```
AD1PCFG = 0xFFFE;             // AN0 = analog input; other ANi = digital
```

The schematic for an I/O pin is shown in Figure PORT-1. It involves the following registers:

- TRISA and TRISB registers determine which pins serve as inputs (via "1") and which as outputs (via "0").
- LATA and LATB registers are used to write to, or read back from, the data latch shown, which controls the I/O pin if the corresponding TRIS bit has been cleared, to make the pin an output.
- PORTA and PORTB registers are used to read back the state of the I/O pin (rather than read back the state of the data latch). A write to PORTA or PORTB writes to the data latch.
- ODCA and ODCB registers control whether an output pin is configured as a normal *totem-pole* output (via "0") or an *open-drain* output (via "1"). A totem-pole output pin includes a transistor that can pull the pin high for a one and another transistor that can pull the pin low for a zero. In contrast, an open-drain output pin includes only the latter transistor that can pull the pin low for a zero. If a one is written to an open-drain output pin, that pin becomes a high impedance, unable to control the pin at all. This configuration is generally used with an external pull-up resistor that will pull the open-drain pin high when a one is written to the pin but which will be overridden by the internal transistor when a zero is written to the pin. The external pull-up resistor must be pulled up with a voltage that does not exceed the supply voltage for the PIC24 chip itself.

Dealing with Individual Pins

When compiling a user program, the first line that the C compiler deals with is

```
#include <p24FJ64GA002.h>
```

Within this file, each bit of any of the above registers is given a name that serves as a Boolean representation for that pin. For example, consider the input from bit 5 of PORTB and its _RB5 representation:

```
        _RB5 = 1 if bit 5 of PORTB is high          whereas         _RB5 = 0 if bit 5 of PORTB is low
```

We can write to and read from individual bits denoted in this way:

```
_LATA2 = 1;                 // Write a one to bit 2 of LATA, the latch register associated with Port A

if (!_RB5) Pushbutton();    // If bit 5 of PORTB is low, call the Pushbutton function
```

LATx vs. PORTx Use

A pin that has been set up as an input, driven from external circuitry, exhibits a state that is unrelated to the state of the data latch for that pin. Consequently, when reading the state of that pin, read the port itself, not the latch.

The use of LATx vs. PORTx for updating the output of a bit of the port presents a subtle complication. Consider the instruction to set a bit of a port (versus setting a bit of the latch). It reads the entire *port* into a "hidden" CPU register (i.e., a register that cannot be accessed by an instruction), sets the appropriate bit of the hidden register, and then copies the result back out to the *latch* register associated with the port. In the process, bits of the latch register (in addition to the bit being set by the instruction) can end up changed from what they were before this operation.

The following rules will keep you out of trouble:
- For port reads or bit tests, use PORTx.
- For port writes or bit writes, use LATx.

Change Notification Feature

Each I/O pin has associated with it a "change notification" role, depicted in Figure PORT-2 and designed to generate an interrupt (if enabled) when the input to this pin changes state. The RB5 pin's change notification function is designated CN27. We can query whether a change on this pin has occurred by enabling the change feature with _CN27IE = 1 and then checking whether _CNIF has been set. To clear the flag, read PORTB and clear _CNIF.

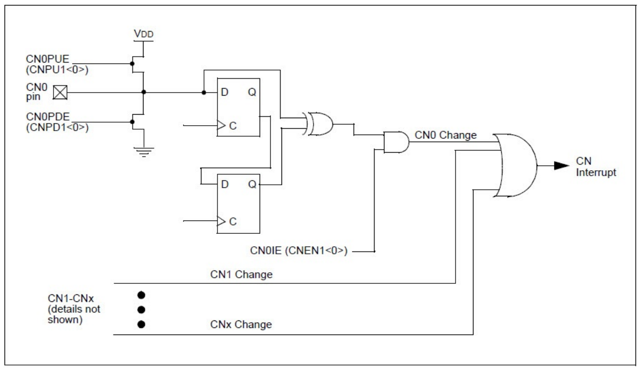
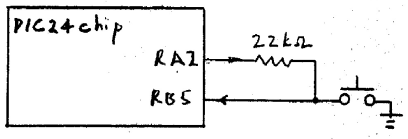


Figure PORT-2 Change notification

Input Pullup or Pulldown Feature

As shown in Figure PORT-3a below, we use the RB5 pin as an input, to sense whether a pushbutton is pressed or released. The circuit uses a 22 kΩ resistor to pull the line high unless the pushbutton is pressed, connecting this line to ground and overriding the effect of the pullup resistor. The pullup resistor is connected to the RA2 output so that when RB5 is not being queried (i.e., most of the time), RA2 can be dropped to zero, eliminating its current draw even though the pushbutton may be pushed.

As shown in Figure PORT-3b, the pushbutton can be connected to RB5 without the need for an external pullup resistor simply by using CN27PUE = 1. Furthermore, when the pushbutton circuit is not being queried (again, most of the time) we can make CN27PUE = 0 to eliminate the current draw that would otherwise occur even though the pushbutton may be pushed.



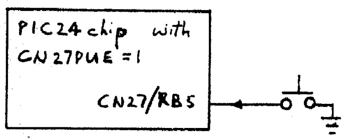


Figure PORT-3 (a) External resistor pullup (b) Internal pullup