CHAPTER 7

Digital Input and Output

Digital inputs and outputs (DIO) are the simplest interfaces between the PIC32 and other electronics, sensors, and actuators. The PIC32 has many DIO pins, each of which normally has one of two states: high (1) or low (0). These states usually correspond to 3.3 V or 0 V.[1] The DIO peripheral also handles change notification (CN) interrupts, which happen when the input changes on at least one of up to 19 digital inputs.

7.1 Overview

The PIC32 offers many DIO pins, arranged into "ports" B through G on the PIC32MX795F512H. The pins are labeled Rxy, where x is the port letter and y is the pin number. For example, pin 5 of port B is named RB5. (We often omit the R, referring to pin RB5 as B5, when there is no possibility of confusion.) Port B is a full 16-bit port, with pins 0-15, and port B can also be used for analog input (Chapter 10). Other ports have a smaller number of pins, not necessarily sequentially numbered; for example, port G has RG2, RG3, and RG6-RG9. All pins labeled Rxy can be used for input or output, except for RG2 and RG3, pins that are shared with USB communication lines and are input only. For more details on the available pin numbers, see Section 1 of the Data Sheet.

The PIC32 has two types of digital outputs: buffered and open drain. Usually, outputs are buffered and can drive the associated pin to either 0 V or 3.3 V. Some pins, however, can be configured with open drains. Their pins can be driven to 0 V or to a high impedance state often called "floating" or "high-Z." When floating, the output is effectively disconnected, allowing you to attach an external "pull-up" resistor from the output to a positive voltage. Then, when the output floats, the external pull-up resistor connects the pin to the positive voltage (Figure 7.1). The positive voltage must be less than either 3.3 V or 5 V, depending on the pin (see, e.g., Figure 2.1 and Table 2.2 to determine which pins can tolerate 5 V).

A pin configured as an output should not source or sink more than about 10 mA. For example, it would be a mistake to connect a digital output to a 10 Ω resistor to ground. In this case,

[1] Technically, there are tolerances associated with these voltages. According to the Electrical Characteristics section of the Data Sheet, input voltages below about 0.5 V are treated as logic low, and input voltages above 2.1 V are treated as logic high.

Embedded Computing and Mechatronics with the PIC32 Microcontroller. http://dx.doi.org/10.1016/B978-0-12-420165-1.00007-X

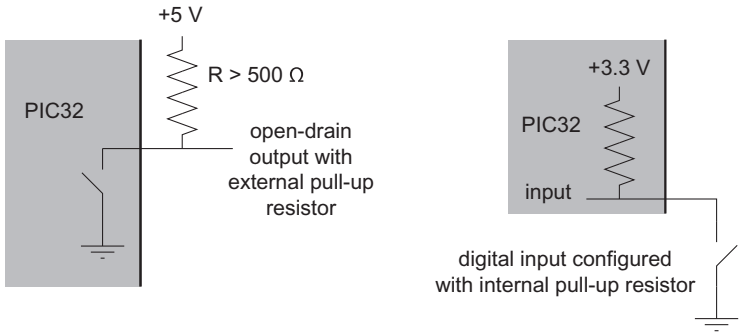

Figure 7.1
Left: A digital output configured as an open-drain output with external pull-up resistor to 5 V. (This should only be done for 5 V tolerant pins.) The resistor should allow no more than 10 mA to flow into the PIC32 when the PIC32 holds the output low. If the LAT bit controlling the pin has the value 1, the internal switch is open, and the output reads 5 V. If the bit is a 0, internal switch is closed and the output reads 0 V. Right: A digital input configured with an internal pull-up resistor allows a simple open-close switch to yield digital high when the switch is open and digital low when the switch is closed.

creating a digital high output of 3.3 V would require 3.3 V/10 Ω = 330 mA, which a digital output cannot source. Be careful; trying to source or sink too much current from a pin may damage your PIC32!

An input pin will read low, or 0, if the input voltage is close to zero, and it will read high, or 1, if it is close to 3.3 V. Some pins tolerate inputs up to 5 V. Some input pins, those that can also be used for "change notification" (labeled CNy, y = 0 to 18, spread out over several of the ports), can be configured with an internal pull-up resistor to 3.3 V. If configured this way, the input will read "high" if it is disconnected (Figure 7.1). This is useful for interfacing with simple buttons, which either connect the input to ground or leave it floating (e.g., the USER button in Figure 2.3).[2] Otherwise, if an input pin is not connected to anything, we cannot be certain what the input will read.

Input pins have fairly high input impedance—very little current will flow in or out of an input pin. Therefore, connecting an external circuit to an input pin should have little effect on the behavior of the external circuit.

Up to 19 inputs can be configured as change notification inputs. When enabled, the change notification pins generate interrupts if their digital input state changes. The ISR must then read from the ports configured with change notification, or else future input changes will not result

[2] According to the PIC32MX575/675/695/775/795 Family Silicon Errata and Data Sheet Clarification, an internal pull-up resistor is not guaranteed to make *external* devices read the pin as high, and even the PIC32 may not read the pin as high if an external load causes it to source more than 50 μA.

in an interrupt. The ISR can compare the new port values to the previous values to determine which specific input has changed.

Microchip recommends that unused digital I/O pins be configured as outputs and driven to a logic low state, though this is not required. All pins are configured as inputs by default. This safety feature prevents the PIC32 from imposing unwanted voltages on attached circuitry before your program begins executing. The pins on port B are shared with the with the analog-to-digital converter (ADC) and default to analog inputs, unless explicitly set as digital pins.

7.2 Details

TRISx, x = B to G These tri-state SFRs determine which port x DIO pins are inputs and which are outputs. Bit y corresponds to the port's pin y (i.e., Rxy). Bits can be accessed individually by using TRISxbits.TRISxy. For example, TRISDbits.TRISD5 = 0 makes RD5 an output (0 = Output), and TRISDbits.TRISD5 = 1 makes RD5 an input (1 = Input). Bits of TRISx default to 1 on reset.

LATx, x = B to G A write to the latch chooses the output for pins configured as digital outputs. (Pins configured as inputs ignore the write.) Bit y correspond to that port's pin (i.e., Rxy). For example, if TRISD = 0x0000, making all port D pins outputs, then LATD = 0x00FF sets pins RD0-RD7 high and other RD pins low. Individual pins can be referenced using LATxbits.LATxy, where y is the pin number. For example, LATDbits.LATD11 = 1 sets pin RD11 high. A write of 1 to an open-drain output sets the output to floating, while a write of 0 makes the output to low.

PORTx, x = B to G PORTx returns the current digital value for DIO pins on port x configured as inputs. Bit y corresponds to pin Rxy. Individual pins can be accessed as PORTxbits.RDy; for example, PORTDbits.RD6 returns the digital input for RD6.

ODCx, x = B to G The open-drain configuration SFR determines whether outputs are open drain or not. Individual bits can be accessed using ODCxbits.ODCxy. For example, if TRISbits.TRISD8 = 0, making RD8 an output, then ODCDbits.ODCD8 = 1 configures RD8 as an open-drain output, and ODCDbits.ODCD8 = 0 configures RD8 as a typical buffered output. The reset default for all bits is 0.

AD1PCFG The pin configuration bits in this register determine whether port B's pins are analog or digital inputs. See Chapter 10 for information about analog inputs.

AD1PCFG⟨x⟩, or AD1PCFGbits.PCFGx, x = 0 to 15, controls whether pin ANx (equivalently RBx) is an analog input: 0 = analog input, 1 = digital pin.

Each of the lower 16 bits of this SFR correspond to a port B pin. On reset, they are zero, meaning that all port B pins are analog inputs by default. Therefore, to use a port B pin as a digital input, you must explicitly set the appropriate pin in AD1PCFG.

For example, to use port B, pin 2 (RB2) as a digital input, set AD1PCFGbits.PCFG2 (AD1PCFG⟨2⟩) to one. This extra configuration step applies only to port B because the other ports do not overlap with analog inputs.

CNPUE Change notification pull-up enable allows you to enable an internal pull-up resistor on the change notification pins (CN0-CN18). Each bit in CNPUE⟨18:0⟩, when set, enables the pull-up, and when clear, disables it. Bit x corresponds to pin CNx. Individual bits can be accessed using CNPUEbits.CNPUEx. For example, if CNPUEbits.CNPUE2 = 1, then CN2/RB0 has the internal pull-up resistor enabled, and if CNPUEbits.CNPUE2 = 0, then it is disabled. The reset default for all bits is 0 so the pull-up resistors are disabled. An internal pull-up resistor can be convenient because it ensures that the input pin is in a known state when disconnected.

Latches vs. ports: What is the difference between the latch (LATx) and port (PORTx) SFRs? The PORTx SFRs correspond to voltages on the pins while LATx SFRS correspond to what the pin should output if configured as an output. When you read from LATx you are actually reading the last value you wanted to put out on the port, not the pins' actual state. Therefore, to read pins, use PORTx, and to write digital outputs, use LATx.

Using the INV, CLR, and SET SFRs vs. directly manipulating bits: To directly set a bit, say bit 4 of LATF, you could use either `LATFSET = 0b10000` (equivalently `LATFSET = 0x10`) or you could use `LATFbits.LATF4 = 1`, based on the bit field names in the `p32mx795f512h.h` file. The former approach is atomic—it executes in a single assembly statement. The latter approach causes the CPU to copy LATF to a CPU register, set bit 4 of the CPU's copy, and write the result back to LATF. While this takes more assembly commands, we generally recommend that you use this approach instead of using the SFR's SET, CLR, and INV registers. The resulting syntax is clearer (essentially self-documenting by the name of the bit field) and less error prone.

7.2.1 Change Notification

Change notification interrupts can be generated on pins CN0-CN18 and are triggered when the input state on the pin changes. The relevant SFRs are:

CNPUE Change notification pull-up enable. See above (it is listed there because pull-ups can be useful even if no change notification is used).

CNCON The change notification control SFR enables CN interrupts if CNCON⟨15⟩ (CNCONbits.ON) equals 1. The default is 0.

CNEN A particular pin CNx can generate a change notification interrupt if CNEN⟨x⟩ (CNENbits.CNENx) is 1. Otherwise it is not included in the change notification.

The relevant interrupt bit fields and constants are:

IFS1⟨0⟩ or IFS1bits.CNIF: Interrupt status flag for change notification, set when the interrupt is pending.

IEC1⟨0⟩ or IEC1bits.CNIE: Interrupt enable flag for change notification, set to enable the interrupt.

IPC6⟨20:18⟩ or IPC6bits.CNIP: Change notification interrupt priority.
IPC6⟨17:16⟩ or IPC6bits.CNIS: Change notification sub-priority.
Vector Number: The change notification uses interrupt vector 26 or `_CHANGE_NOTICE_VECTOR`, as defined in `p32mx795f512h.h`.

A recommended procedure for enabling the CN interrupt:

1. Write an ISR using the vector `_CHANGE_NOTICE_VECTOR` (26). It should clear IFS1bits.CNIF and read the pins involved in the CN scan to re-enable the interrupt.
2. Disable all interrupts using `__builtin_disable_interrupts()`.
3. Set CNENbits.CNENx to one for each pin x that you want included in the change notification, and set CNCONbits.ON to one.
4. Choose the interrupt priority IPC6bits.CNIP and subpriority IPC6bits.CNIS. The priority should match that used in the definition of the ISR.
5. Clear the interrupt flag status IFS1bits.CNIF.
6. Enable the CN interrupt by setting IEC1bits.CNIE to one.
7. Enable interrupts using `__builtin_enable_interrupts()`.

7.3 Sample Code

Our first program, `simplePIC.c`, demonstrated the use of two digital outputs (to control two LEDs) and one digital input (the USER button).

The example below configures the following inputs and outputs:

- Pins RB0 and RB1 are digital inputs with internal pull-up resistors enabled.
- Pins RB2 and RB3 are digital inputs without pull-up resistors.
- Pins RB4 and RB5 are buffered digital outputs.
- Pins RB6 and RB7 are open-drain digital outputs.
- Pins AN8-AN15 are analog inputs.
- RF4 is a digital input with an internal pull-up resistor.
- Change notification is enabled on pins RB0 (CN2), RF4 (CN17), and RF5 (CN18). Since both ports B and F are involved in the change notification, both ports must be read inside the ISR to allow the interrupt to be re-enabled. The ISR toggles one of the NU32 LEDs to indicate that a change has been noticed on pin RB0, RF4, or RF5.

Code Sample 7.1 `DIO_sample.c`**. Digital Input, Output, and Change Notification.**

```
#include "NU32.h"          // constants, funcs for startup and UART

volatile unsigned int oldB = 0, oldF = 0, newB = 0, newF = 0; // save port values

void __ISR(_CHANGE_NOTICE_VECTOR, IPL3SOFT) CNISR(void) { // INT step 1
  newB = PORTB;           // since pins on port B and F are being monitored
```

```
  newF = PORTF;           // by CN, must read both to allow continued functioning
                          // ... do something here with oldB, oldF, newB, newF ...
  oldB = newB;            // save the current values for future use
  oldF = newF;
  LATBINV = 0xF0;         // toggle buffered RB4, RB5 and open-drain RB6, RB7
  NU32_LED1 = !NU32_LED1; // toggle LED1
  IFS1bits.CNIF = 0;      // clear the interrupt flag
}

int main(void) {
  NU32_Startup(); // cache on, min flash wait, interrupts on, LED/button init, UART init

  AD1PCFG = 0x00FF;       // set B8-B15 as analog in, 0-7 as digital pins
  TRISB = 0xFF0F;         // set B4-B7 as digital outputs, 0-3 as digital inputs
  ODCBSET = 0x00C0;       // set ODCB bits 6 and 7, so RB6, RB7 are open drain outputs
  CNPUEbits.CNPUE2 = 1;   // CN2/RB0 input has internal pull-up
  CNPUEbits.CNPUE3 = 1;   // CN3/RB1 input has internal pull-up
  CNPUEbits.CNPUE17 = 1;  // CN17/RF4 input has internal pull-up
                          // due to errata internal pull-ups may not result in a logic 1

  oldB = PORTB;           // bits 0-3 are relevant input
  oldF = PORTF;           // pins of port F are inputs, by default
  LATB = 0x0050;          // RB4 is buffered high, RB5 is buffered low,
                          // RB6 is floating open drain (could be pulled to 3.3 V by
                          // external pull-up resistor), RB7 is low

  __builtin_disable_interrupts(); // step 1: disable interrupts
  CNCONbits.ON = 1;               // step 2: configure peripheral: turn on CN
  CNENbits.CNEN2 = 1;             // Use CN2/RB0 as a change notification
  CNENbits.CNEN17 = 1;            // Use CN17/RF4 as a change notification
  CNENbits.CNEN18 = 1;            // Use CN18/RF5 as a change notification

  IPC6bits.CNIP = 3;              // step 3: set interrupt priority
  IPC6bits.CNIS = 2;              // step 4: set interrupt subpriority
  IFS1bits.CNIF = 0;              // step 5: clear the interrupt flag
  IEC1bits.CNIE = 1;              // step 6: enable the CN interrupt
  __builtin_enable_interrupts();  // step 7: CPU enabled for mvec interrupts

  while(1) {
      ;                           // infinite loop
  }
  return 0;
}
```

7.4 Chapter Summary

- The PIC32 has several DIO ports, labeled with the letters B through G. Only port B has all 16 pins. Almost every pin can be configured as a digital input or a digital output. Some outputs can be configured to be open-drain.
- By default, port B inputs are configured as analog inputs. To use these pins as digital inputs you must set the corresponding bits in AD1PCFG to one.
- Nineteen pins can be configured as change notification pins (CN0-CN18). For those pins that are enabled as change notification pins, any change of input state generates an interrupt. To re-enable the interrupt, the ISR should clear the IRQ flag and read the ports with pins involved in the change notification.

- The change notification pins offer an optional internal pull-up resistor so that the input registers as high when it is left floating. These pull-up resistors can be used regardless of whether change notification is enabled for the pin. The internal pull-up resistor allows simple interfacing with push-buttons, for example.

7.5 Exercises

1. True or false? If an input pin is not connected to anything, it always reads digital low.
2. You are configuring port B to receive analog and digital inputs and to write digital output. Here is how you would like to configure the port. (Pin x corresponds to RBx.)
   - Pin 0 is an analog input.
   - Pin 1 is a "typical" buffered digital output.
   - Pin 2 is an open-drain digital output.
   - Pin 3 is a "typical" digital input.
   - Pin 4 is a digital input with an internal pull-up resistor.
   - Pins 5-15 are analog inputs.
   - Pin 3 is monitored for change notification, and the change notification interrupt is enabled.

   Questions:
   a. Which digital pin would most likely have an external pull-up resistor? What would be a reasonable range of resistances to use? Explain what factors set the lower bound on the resistance and what factors set the upper bound on the resistance.
   b. To achieve the configuration described above, give the eight-digit hex values you should write to AD1PCFG, TRISB, ODCB, CNPUE, CNCON, and CNEN. (Some of these SFRs have unimplemented bits 16-31; write 0 for those bits.)

Further Reading

PIC32 family reference manual. Section 12: I/O ports. (2011). Microchip Technology Inc.