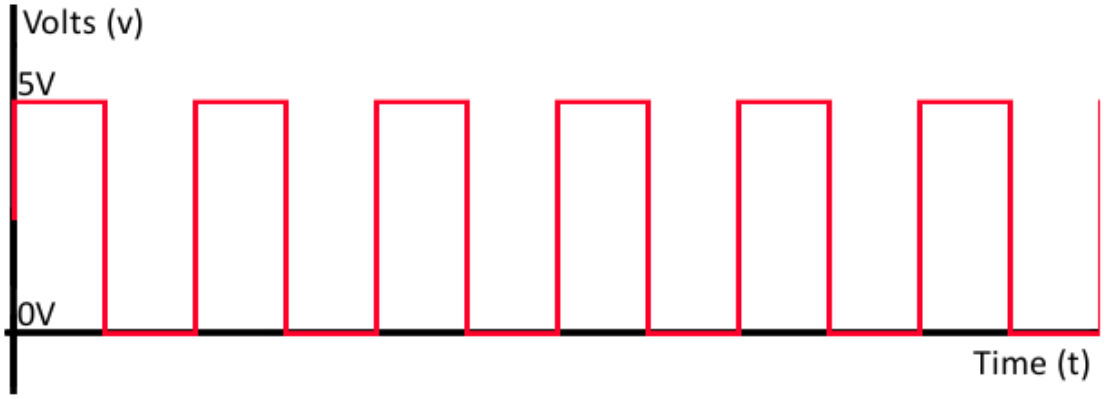
Digital I/O on Microchip's dsPIC33EP512GP504 Processor

Prepared by Sean Collins
August 2018

Datasheets

In order to find information about the processor you are using, you can read its corresponding "datasheet". A datasheet is a document published by the manufacturer of a component, and it provides both technical specifications as well as instructions about how to work with the component. For example, we can find the datasheet for the dsPIC33EP512GP504 at the following page:
https://www.microchip.com/wwwproducts/en/dsPIC33EP512GP504

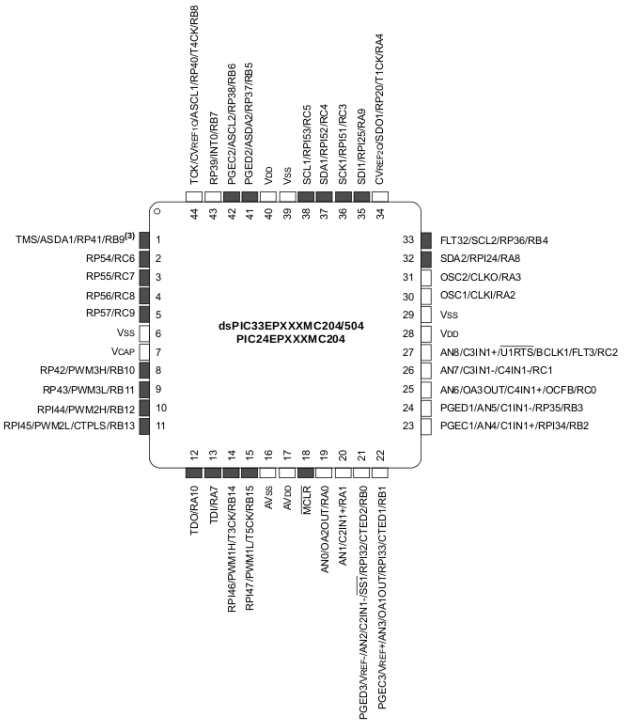
Digital I/O

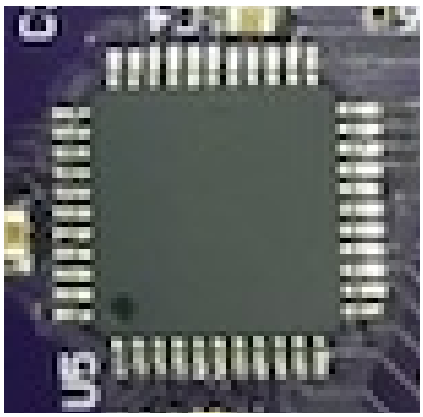


When we refer to "digital signals", we are talking about electrical signals that take on 1 of 2 possible values (LOW or HIGH). Usually, these values take the form of a low voltage (~0 V) or a high voltage (~5 V).

Pins

A processor can output or input this kind of signal using its pins. Check out page 11 in the datasheet for the "Pin Diagram" to the left, below. It indicates the purposes of each of the metal pins, which you can see protruding from the processor to the right, below. Some of the pins (e.g. pin 44, pin 21, etc…) serve multiple different purposes. It is up to the software to specify which purpose/mode is active at any given time. For this discussion, we will be using the pins as interfaces for the input and output of digital signals.

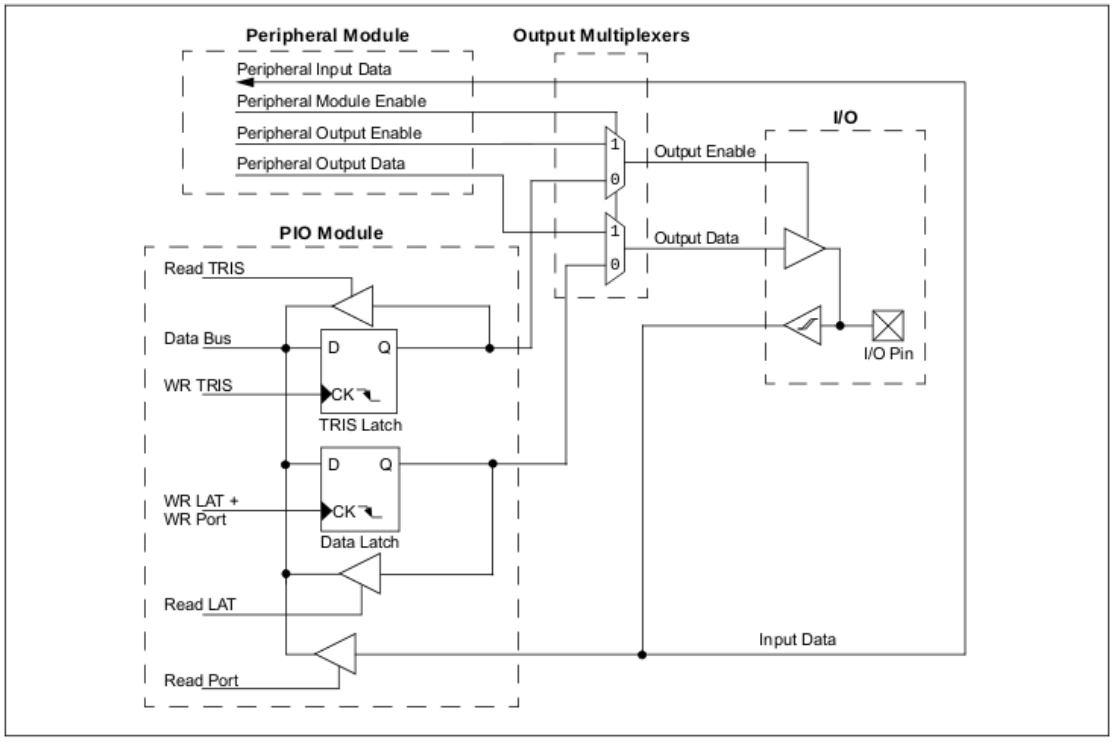




Ports

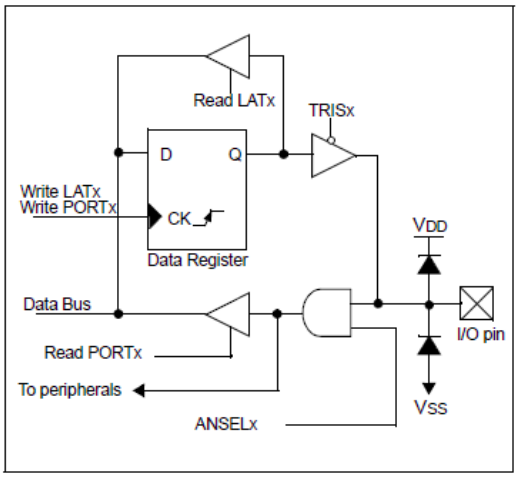
While the term “pins” refers to the actual wires that serve as either inputs or outputs to the processor, the input/output “ports” (described in Section 11, page 173 of datasheet) are the modules that programmers work with in order to control input and output. In other words, the ports are the interface between the program and the physical pins. Figure 11-1 in the datasheet illustrates the structure of a port on a typical Microchip processor:

FIGURE 11-1: BLOCK DIAGRAM OF A TYPICAL SHARED PORT STRUCTURE



This diagram illustrates the fact that some I/O pins are shared by an IO module and a peripheral. A “peripheral” is basically a module that provides the processor with some kind of support or capability. For example, UART and I2C peripherals provide processors with circuitry that can communicate with other devices. As the datasheet describes, peripheral modules usually take precedence over input and output. In other words, for shared ports, if the peripheral is enabled, it will not be possible to use the port for general purpose input and output. By default, peripherals are disabled, so we do not need to worry about them for now. Here is a simpler diagram, which ignores peripherals:

FIGURE 12-1: GENERIC I/O PORT OPERATION



The latch components are made out of “flip-flops”, which are electrical components that are fundamental to modern computer systems. For this discussion, we will ignore the specific internals of flip-flops. The important concept to know is that flip-flops enable an electrical circuit to hold onto a “state.” In other words, they are the bits which make up the registers that software reads and writes in order to control the port.

Configuring a Pin as Input / Output

- Write 0 to TRIS bit for output
- Write 1 to TRIS bit for input

Outputting a Digital Signal

- Write either 1 or 0 to LAT bit

Inputting a Digital Signal

- Read the PORT bit (when the TRIS bit is 1) to sample the signal being applied to the pin

Coding Digital I/O in MPLAB X

Configuration Bits

Programs for Microchip processors must initialize a set of bits that are called “Configuration Bits.” These bits allow the programmer to customize the processor’s behavior. Fortunately, MPLAB X provides a tool with which we can automatically generate code that takes care of configuration. For now, we can use the default code. Access the configuration code by selecting **Production → Set Configuration Bits**. After the tool opens, click the “Generate Source Code to Output.” You can then copy the code from the Output window and paste it at the beginning of the file.

(Note: If you prefer, you can paste the configuration bits into a separate header file called “config.h”, or something similar. Then, #include this header above your main function.)

Processor-Specific Header Files

You will also need to add the following header files, which define several useful macros and constants.

```
#include <libpic30.h>
#include <p33EP512GP504.h>
```

Example Code

```
// Header file containing generated configuration bits
#include <config.h>

// Processor specific header files
#include <libpic30.h>
#include <p33EP512GP504.h>

int main () {
        // Setup Pin B15 as an output
        TRISBbits.TRISB15 = 0;

        // Write LOW to Pin B15
        LATBbits.LATB15 = 0;

        return 0;
}
```