Tom Aarts
Nov 3 2021

Now that there is a basic pin for the mcp23017 there are design points to decide. Admittedly some are these are based upon how I implemented the existing ‘Module’ example.

Table of Contents

Device contains pins.

The chip contains a list of its pins.
Reading the data sheet, details are documented based on the chips registers where user code updates a register and that may update an output pin. The documentation does not imply user code directly access a pin. I think the new example code directly accessing the Pin is not how this should be written. I think the user code should access the chip and the chip access the PIN when required.

So rather than pin.high(), chip.pinHigh(pinNumber).

Other pin and device attributes

There are attributes specific to IN/OUT pins, and a few specific to the CHIP. Each of the three groups can be implemented following the pattern in the example

Device INIT reads HW

Upon initialization the chip could read the existing device register values and populate the pin list with the proper IN or OUT pin. The actual pin code should runtime validate the pin direction is correct, but populating this list means a chip.prettyPrint() method could display the complete details.

Reset Example app

Provide separate example app that drives a SOC gpio to reset the chip.

Example Interrupt Monitor

Supply a separate ‘app’ example that monitors for an input pin configured to interrupt. This should reference the GPINTEN DEFVAL and INTCON pin-IOCON.ODR pin-IOCON.INTPOL pin IOCON.MIRRO only to let the reader know various configuration possibilities exist. A single actual example is all that is needed.