Port Expander (MCP23S17) with PIC32 example

Sean Carroll (BS ECE/CS ’17) [swc63]

I started this mini-project to learn about the port expander and to get some familiarity with how students might use the port expander in their projects.

NOTE: To avoid future confusion in the lectures/labs, I’ve renamed all the ports on the port expander from A/B to Y/Z, that way all code written and loaded on the PIC32 won’t confuse the ports A/B on the device and the ports A/B on the port expander.

Main function:

The two LEDs are lit if the corresponding switches are switched to an ON state, otherwise they are unlit.

Wiring overview:

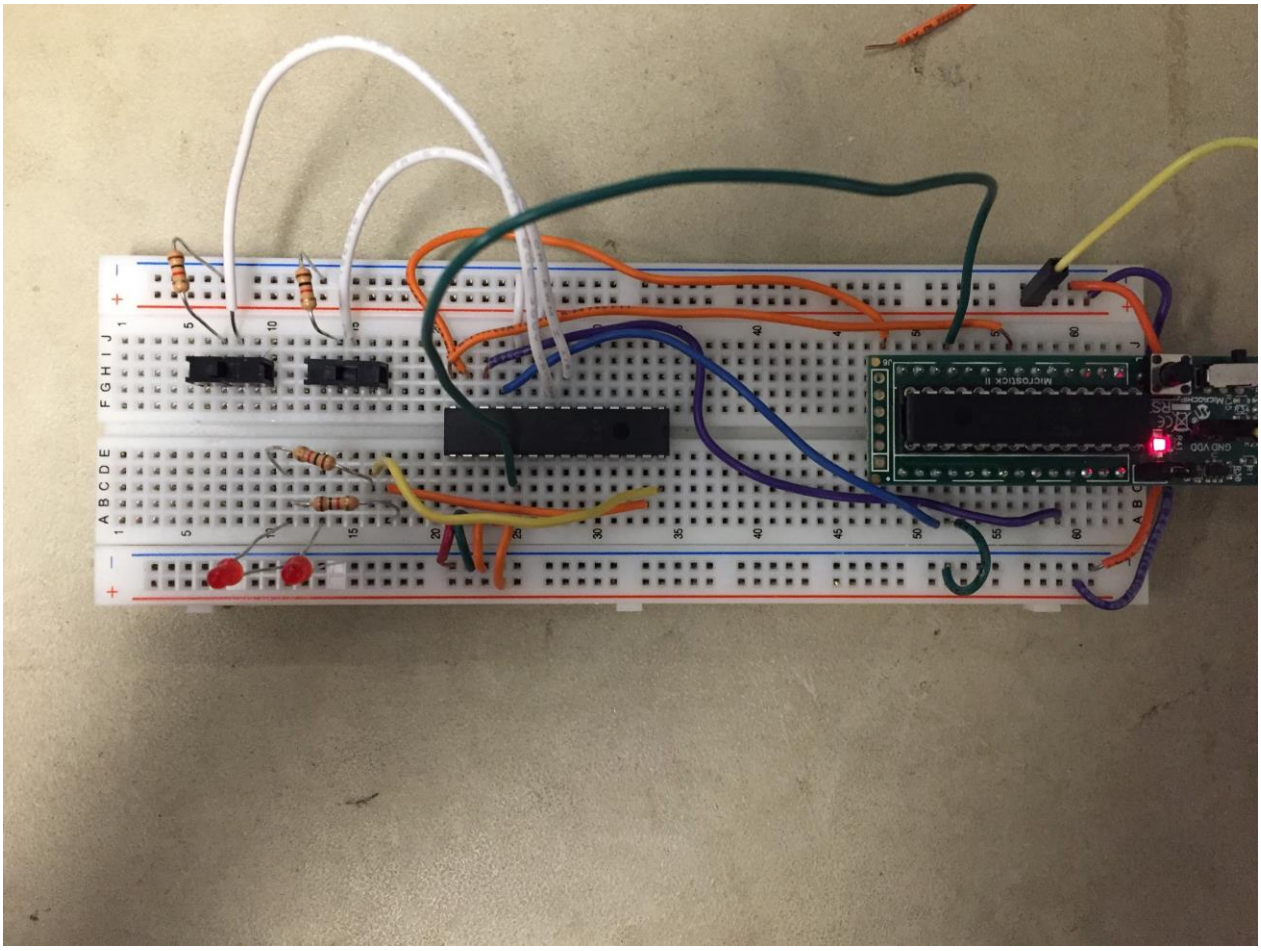
My first task was to light up LEDs with different pins on the port expander (PE), using SPI to communicate between the PIC32 and the PE. I used SPI channel 2 for this example. The pins and ports used in this example are:

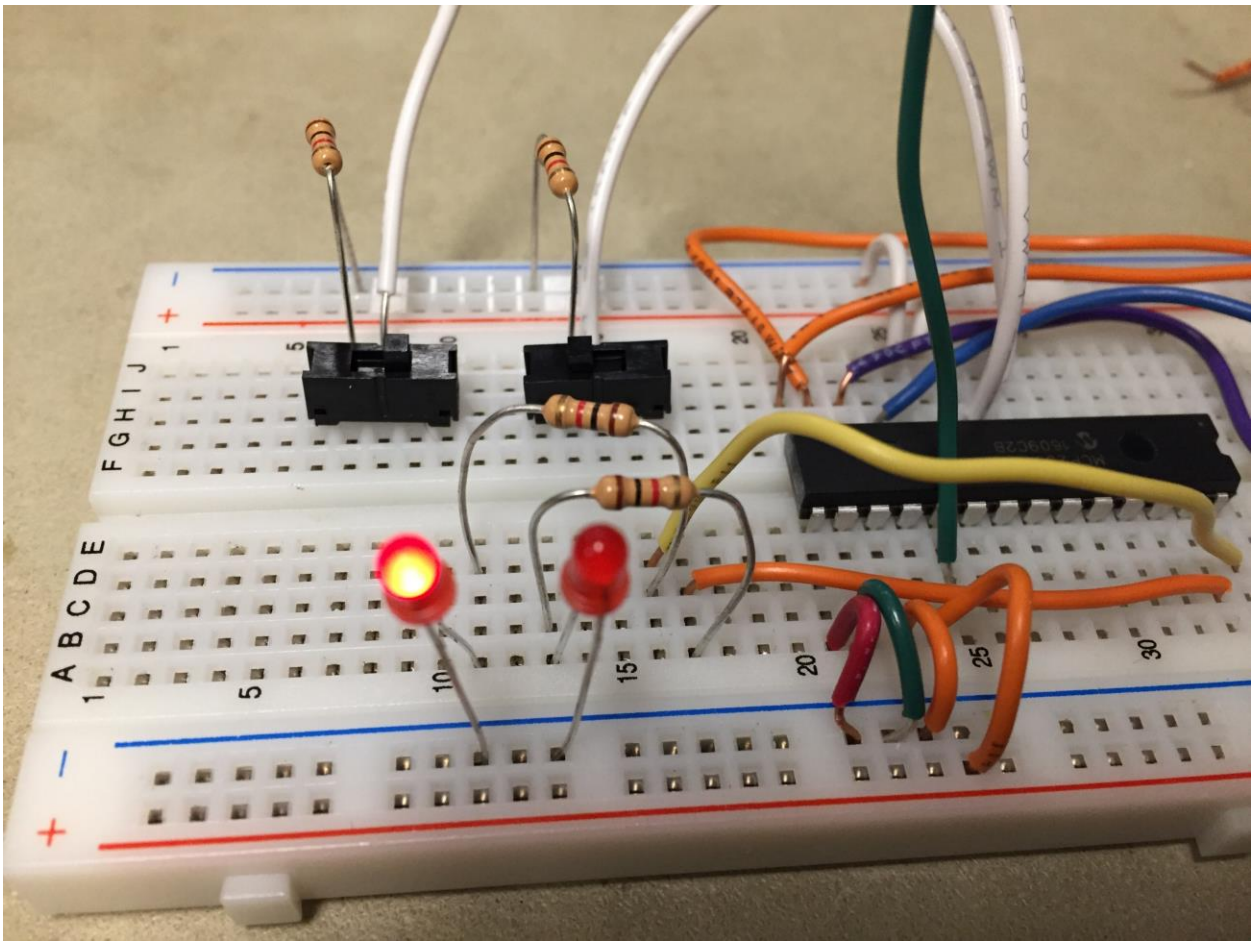
Purpose	Peripheral	Pin on PIC32	Connected to Pin on Port Expander
CS	RPB9	18	11
SCK	RPB15 (SCK2)	26	12
SDI	RPB2	6	14
SDO	RPB5	14	13
External Interrupt	RPA3	10	19 (INTZ)

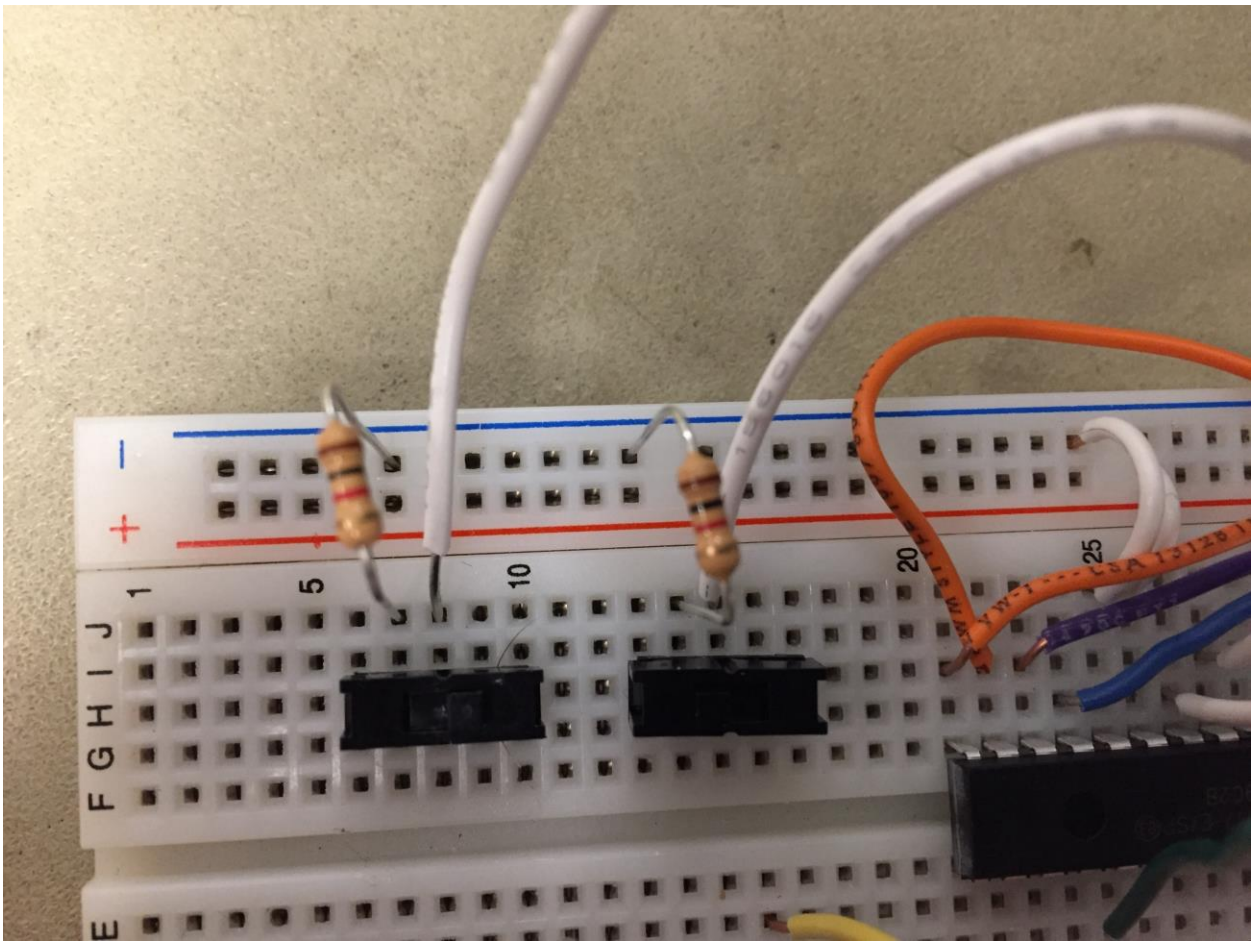
From the PIC32, I have power and ground connections going to the rails on the breadboard.

From the port expander, I have power and ground connections connected to the rails accordingly, and also two sets of an LED and 1kOhm resistor connected in series from the two output pins (PortY.6 and PortY.7 for our example). There are also two sets of a switch alternating between a 1kOhm resistor connected to ground and a broken wire (purpose: to make use of the internal pull-up resistors) connected to the two input pins (PortZ.6 and PortZ.7 for our example). I also have the three hardware address pins shorted to ground to make the SPI hardware address 0b000 for the port expander.

See pictures if needed (excuse my messy wires):


F G H I J
A B C D E
MICROSTICK II
GND VDD


A B C D E
F G H I J
1
5
10
15
20
25
30


F G H I J
1
5
10
20
25

Code overview:

I wrote a small library to interact with the port expander.

The initializer function initPE() is an example of how you might initialize the port expander on startup. It uses SPI channel 2, and the relevant peripheral settings as seen in the table above.

The various functions for enabling/disabling or setting/clearing various pins act as expected (read http://ww1.microchip.com/downloads/en/DeviceDoc/20001952C.pdf for more details, but recall the code will refer to Port Y and Z, and the original documentation will refer to Port A and B).

The write and read functions work similarly, as they work by sending the initial opcode with hardware address hard coded to 0b000, as well as setting the R/W for a read and clearing it for a write. On each transmission, I read from the SPI SDI register to clear out any junk information on most bytes sent, with the dummy data sent during a read transaction actually yielding relevant information from the SDI register.

The main body of the example code works by setting up the SPI channel and sending an initial control byte to the control register on the port expander. We also send control bytes to set PortY.6-7 as outputs and PortZ.6-7 as inputs. We also set those inputs to use the internal pull-up resistors on the port expander and enable interrupts if the value changes. On an interrupt, it will pull the INTZ pin low from a resting high state. This pairs with an external interrupt routine that reads port Z and writes the read data to port Y. We also have code to blink an LED and keep track of system time on the PIC32.