

---

# Vector 2x Compass Notes

## Using the Vector 2x with the MIT Handy Board

Author: Doug Rinckes, doug.rinckes@iname.com

Date: January 2001

### Introduction

This is a guide to how to attach the Vector 2x compass module to the MIT Handy Board. This assumes that the SPI port is not available – possibly because you have already used it for a Polaroid range detector.

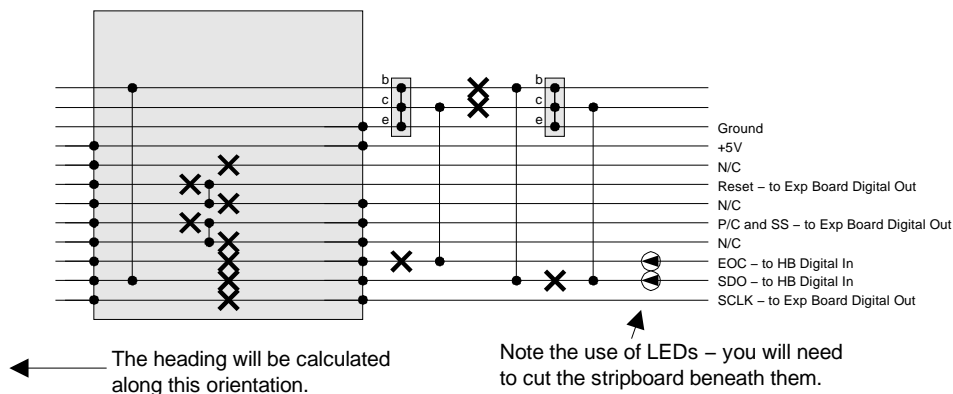
### Requirements

This requires an Expansion Board fitted – as we need the digital outputs.

### Circuit Board

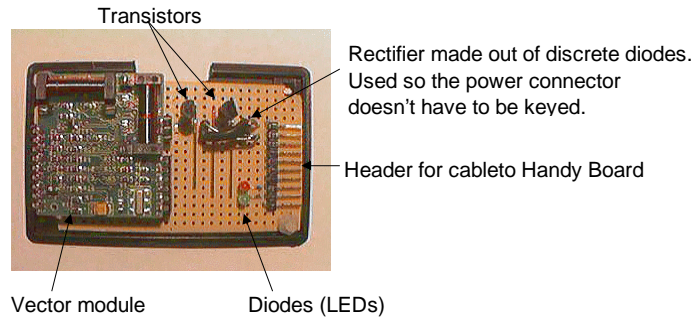
The following diagram is for a 0.1" stripboard layout. Each strip is indicated with a line, dots indicate connections, and crosses indicate where you should cut the strips.

If you want to make up a PCB for this, then you're probably smart enough to work it out.



The additional components are two BC548 NPN transistors (used to invert the voltage to the EOC and SDO pins) and two LEDs. These were used to add some resistance to the line and to aid debugging.

The compass module is used in Slave mode. The disadvantage of this is that the Handy Board has to request a reading, which takes around 0.26 seconds for the Vector to calculate. The advantage is that it only consumes Handy Board CPU while taking a reading, and also that it consumes a lot less power.



In the above image, there is a rectifier made out of discrete diodes. This allows the ground and +5V signals to be reversed, and was needed because the power connector used could be accidentally connected the wrong way around. This could be done using a single rectifier component – I just didn't have one lying around.

## Software

The following routine is based on code originally written by Randy Sargent. The pin connections are global variables set at the start of the file, and several `msleep()` calls have been removed. This appears to allow the retrieval of the final bit (which has to be retrieved within 20 usec of SCLK being raised the last time).

```

/*
    compass.c

    Routines for the vector compass module.

    Pinouts are specified as global variables. Must call compass_init
    before attempting to read the compass. It takes approx .26 seconds
    to take a reading, so if you're turning at the time ... :-{

    v1.0, Doug Rinckes, 2001
    doug.rinckes@iname.com
*/

/*
    digital outputs on the expansion board */
int sclk = 5;    /* green wire */
int pcss = 6;    /* white wire */
int reset = 7;   /* brown wire */

/*
    digital inputs on the handyboard */
int sdo = 14;    /* blue wire */
int eoc = 15;    /* purple wire */

void compass_init()
{
    set_digital_out(pcss);
    set_digital_out(sclk);
    set_digital_out(reset);
    clear_digital_out(reset);
    msleep(15L);
    set_digital_out(reset);
    msleep(5L);
    /*
       this long wait is to prevent attempting to read the compass
       until it is fully reset.
    */
    msleep(100L);
}

int compass_read()
{
    int bit;
    int heading = 0;
    int eoc_return;

    /* set sclk and pcss high, then drop pcss for a short time */
    set_digital_out(sclk);
    set_digital_out(pcss);
    msleep(5L);
    clear_digital_out(pcss);
    msleep(2L);
    set_digital_out(pcss);

    /* eoc should drop... */
}

```

```

eoc_return = digital(eoc);
if (eoc_return == 1) {
    /* hmm, eoc didn't drop. try poking it again */
    set_digital_out(pcss);
    msleep(5L);
    clear_digital_out(pcss);
    msleep(2L);
    set_digital_out(pcss);
    eoc_return = digital(eoc);
    /* if eoc still didn't drop, bomb out, otherwise continue */
    if (eoc_return == 1) {
        return -1;
    }
}

/* ok now wait until eoc goes high - indicates that the compass is ready */
while (digital(eoc) != 1) {
    ;
}

/* so eoc is now high, and the compass can be read */
/* drop pcss while data is being clocked out */

clear_digital_out(pcss);

/* clock 7 times to clock out seven bits which aren't used. */
for (bit = 1; bit <= 7; bit++) {
    clear_digital_out(sclk);
    set_digital_out(sclk);
}

/* get the next bit */
clear_digital_out(sclk);
set_digital_out(sclk);
if (digital(sdo) == 1) {
    heading=256;
}

/* now clock out the next 8 bits */
msleep(5L);
for (bit = 7; bit > 0; bit--) {
    clear_digital_out(sclk);
    set_digital_out(sclk);
    if (digital(sdo) != 0) {
        heading +=(int)((2.^(float)(bit))+1.);
    }
}
/*
clock out final bit (bit 0). causes floating point overflow if
done in previous loop.
*/
clear_digital_out(sclk);
set_digital_out(sclk);
if (digital(sdo) != 0) {
    heading +=1;
}

clear_digital_out(sclk);
set_digital_out(pcss);
set_digital_out(sclk);

return heading;
}

```