Displaytech Ltd.®
a seacomp company

# ILI9341 - 8-bit MCU mode

4/26/2016

Scope: This document shows the read and write commands working on hardware. This includes getting the write command to work, which is a multi-parameter write,

Setup: Use DT022BTFT (or other Displaytech TFT with ILI9341) which has the ILI9341. Configure the IM pins for 8-bit MCU mode. Your setup might use a different micro-controller.

Conclusion: We could get the multi-parameter write commands to work with 8-bit MCU mode using the exact configuration as Snap-on. Please see scope traces and print statements below for additional details.

Suggestion: Verify that the commands are similar to the scope traces below. In addition, verify that the data is correct on all datalines (not just a few).

## Part 1: Read "READ_STATUS (0x09) & READ_MADCTL (0x0B)"

Yellow - CS
Turquoise - D/CX (RS)
Pink - RDX

Blue - WRX

This following code snippet which reads the status register then reads from the madctl register is captured in Figure 1.

```
DisplayEnable1();  // enable CS

uint16_t dat, p2, p3, p4, p5;
WriteCommand(READ_STATUS);   // 09h
dat = ReadData();
p2 = ReadData();
p3 = ReadData();
p4 = ReadData();
p5 = ReadData();

PRINT("dummy = 0x%04X\n", dat);
PRINT("p2 = 0x%04X\n", p2);
PRINT("p3 = 0x%04X\n", p3);
PRINT("p4 = 0x%04X\n", p4);
PRINT("p5 = 0x%04X\n", p5);

WriteCommand(READ_MADCTL);   // 0Bh
dat = ReadData();
p2 = ReadData();
p3 = ReadData();

PRINT("dummy = 0x%04X\n", dat);
PRINT("p2 = 0x%04X\n", p2);
PRINT("p3 = 0x%04X\n", p3);

DisplayDisable1();  // enable CS
```

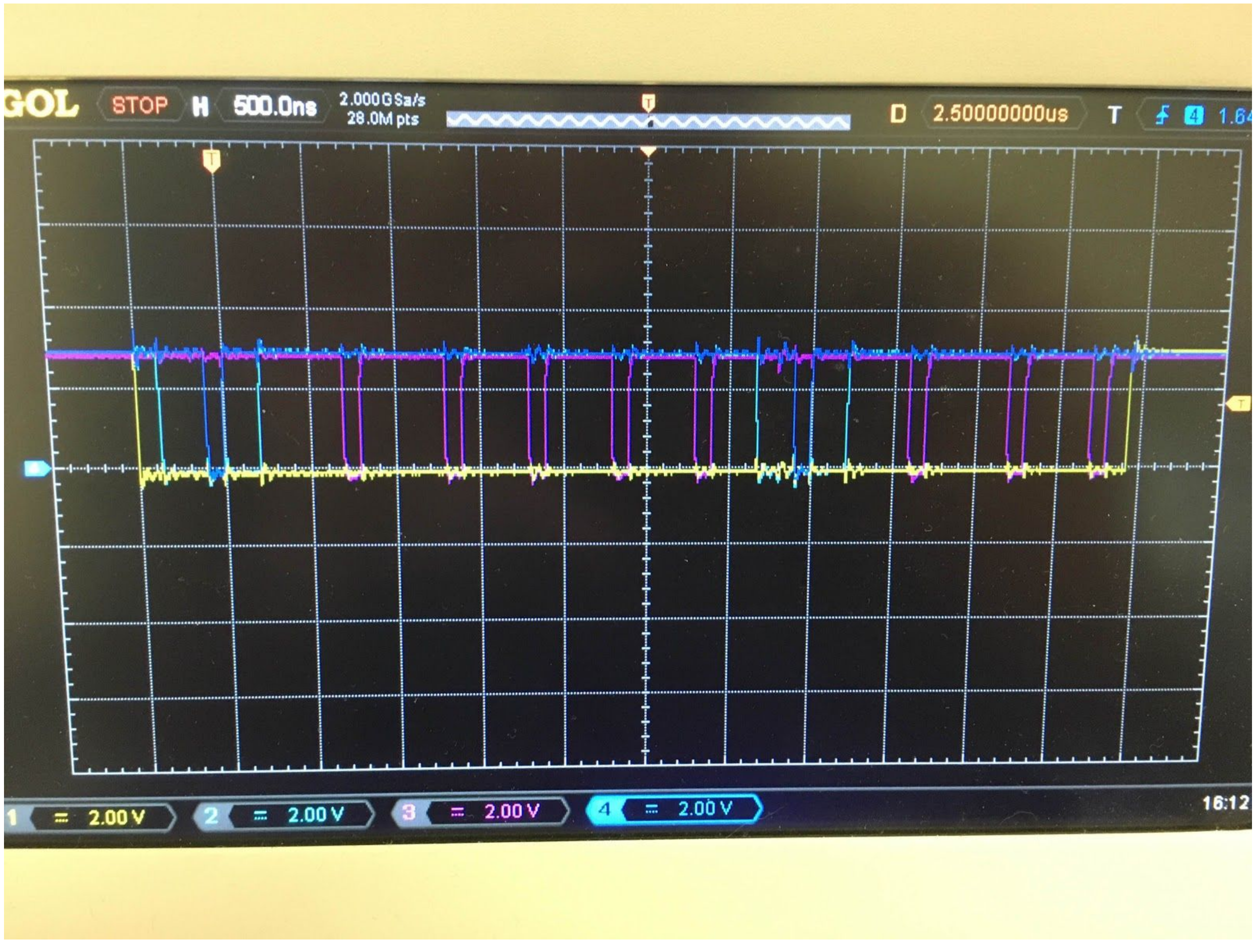
GOL
STOP
H
500.0ns
2.000GSa/s
28.0M pts
D
2.50000000us
T
1.64
1 = 2.00 V
2 = 2.00 V
3 = 2.00 V
4 = 2.00 V
16:12

Figure 1

## Part 2: Write “MEMORY_ACCESS_CTRL (0x36)”

The following code snippet which writes to the mactl register is captured in Figure 2.  Note: the CS happened to be asserted 9us prior to the D/CX line was set low.

```
DisplayEnable1();      // enable CS

 //Memory access
 WriteCommand(MEMORY_ACCESS_CTRL);    // 36h
 WriteData(0x00E8);

 DisplayDisable1();      // enable CS
```

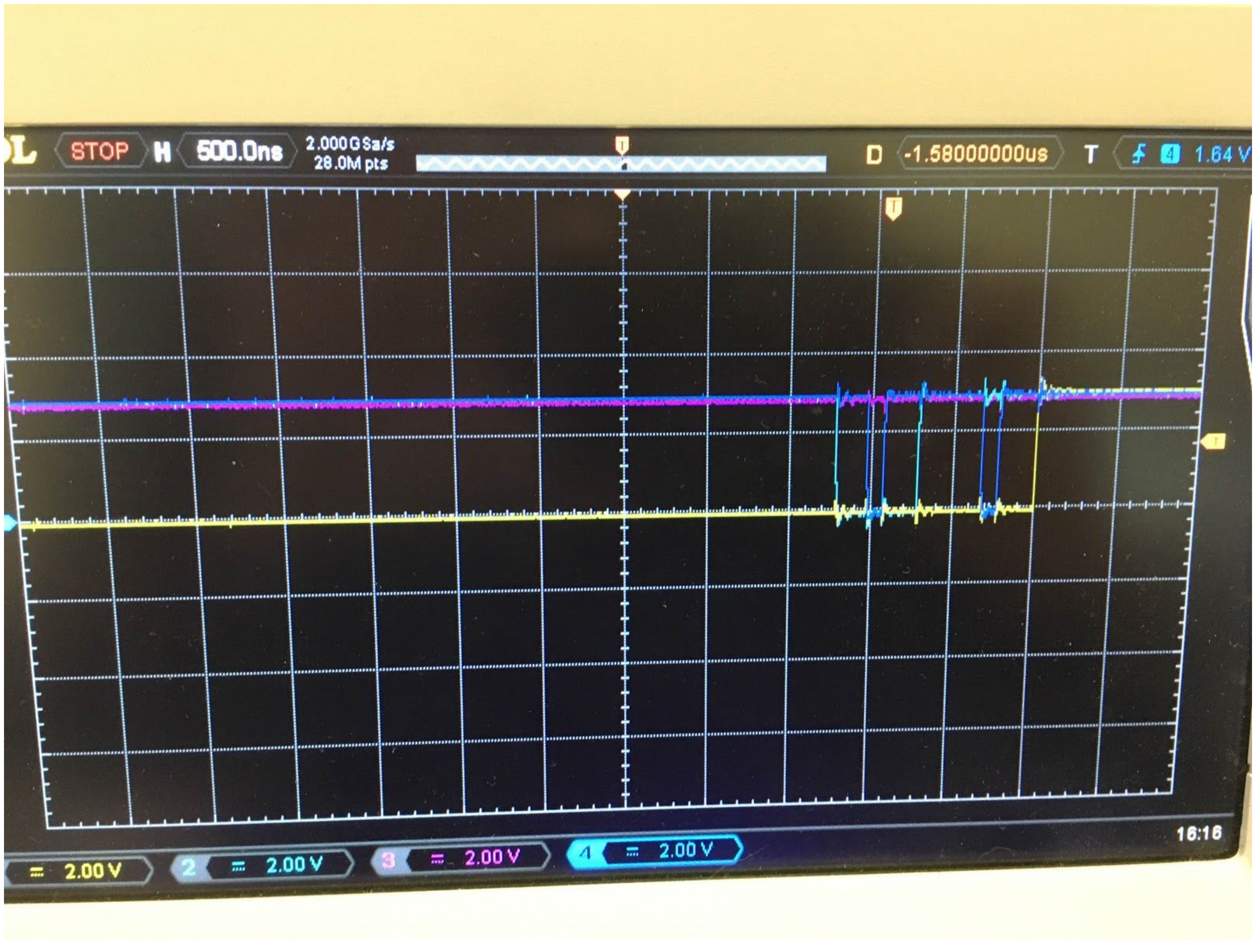

Figure 2

Console output:

Part 1:

```
// read default values from 0x09 and 0x0B
dummy = 0x0009
p2 = 0x0000
p3 = 0x0000
p4 = 0x0061
p5 = 0x0000
dummy = 0x000B
p2 = 0x0000
p3 = 0x0000
```

Part 2:

```
// write to 0x36 occurred here
```

Verify 1 & 2 Worked

Verify the that 0x36 command parameter was set correctly.
Notice that Part 1 value is p3 = 0x0000. Then Part 2 changed value. Then a read again shows that the value changed to p3 = 0x00E8.

```
// read new value in 0x0B
dummy = 0x000B
p2 = 0x0000
p3 = 0x00E8
```

## Appendix: Initialization Code

Ensure your reset sequence is correct. Here is our code:

```
    DisplayResetEnable();  // enable reset
    DelayMs(10);
    DisplayResetDisable();  // disable reset
    DelayMs(120);

    //Software Reset
    WriteCommand(SOFTWARE_RESET);
    //Supposed to wait at least 120ms, wait 250 to be safe
    DelayMs(250);
    WriteCommand(DISPLAY_OFF); // display off
```