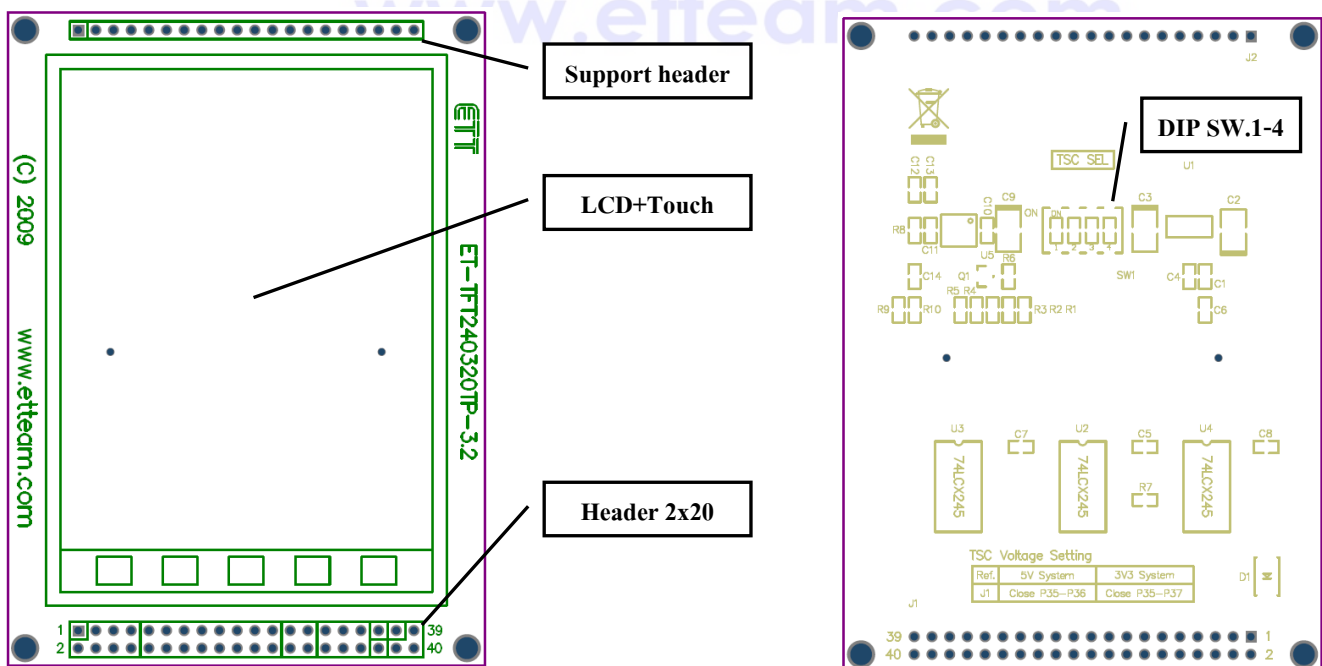


## ET- TFT240320TP-3.2

## 1. คุณสมบัติของบอร์ด ET-TFT240320TP-3.2

- เป็น Display Module TFT LCD Color +Touch Screen ขนาด 240x320 Pixel
- ขนาดของหน้าจอ TFT 3.2 "
- ใช้ Single Chip Driver เบอร์ ILI9320
- ความละเอียดของสีแสดงได้ 65536 สี(RGB =R:5bit-G:6bit-B:5bit)
- ในส่วนของการควบคุม LCD จะใช้การ Interface แบบ 16-bit data/address + ขา Control อีก 6 bit
- ในส่วนของการควบคุม Touch Screen สามารถเลือก Interface ได้ 2 แบบด้วย Dip SW. คือ Interface แบบ SPI โดยผ่าน Chip Touch Screen Controller #ADS7846 (ADC มีความละเอียด 12 บิต) หรือ Interface โดยใช้ขา X-,X+,Y-Y+ ต่อเข้ากับขา ADC ของ MCU โดยตรงก็ได้(การเขียนโปรแกรมควบคุมจะยุ่งยาก)
- ในการใช้งาน ถ้าไม่ต้องการใช้ Touch Screen สามารถ Control เฉพาะในส่วนของการแสดงผล LCD อย่างเดียวก็ได้
- สำหรับ MCU ที่จะนำมาต่อควบคุมการทำงานของบอร์ดแบบเต็มรูปแบบควรมี I/O-Port อย่างน้อย 28 PIN
- สามารถต่อ Interface ได้ทั้ง MCU ที่ใช้ไฟเลี้ยง 5V และ 3.3V (ให้ดูรายละเอียดเพิ่มเติมในหัวข้อการต่อใช้งาน)
- Connector สำหรับต่อใช้งานจะเป็นแบบ Pin Header 2x20 ระยะ pitch 2.54 mm
- ไฟเลี้ยงบอร์ด DC +5 V

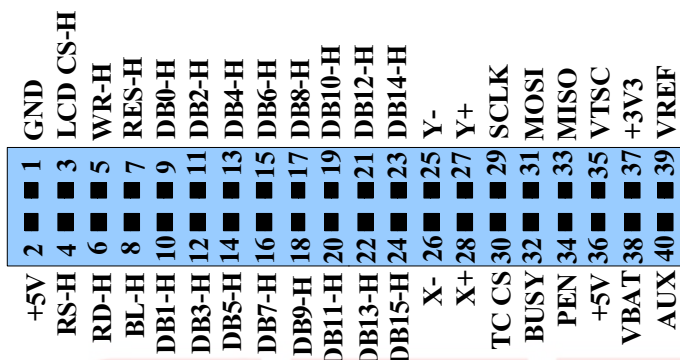
## 2. ลักษณะและโครงสร้างของบอร์ด ET-TFT240320TP-3.2



รูปที่ 1 (A) ด้านหน้าบอร์ด

รูปที่ 1 (B) ด้านหลังบอร์ด

- *Support Header* : จะเป็นรูช่องว่างมีไว้สำหรับต่อ Connector เพื่อใช้ประกอบตัวบอร์ดทางด้านบน
- *LCD+Touch* : จะเป็นเนื้อที่ของจอ LCD ขนาด 240x320 Pixel โดยด้านบนของจอจะถูกฉาบทับด้วยแผ่นของ Touch Screen แบบ Resistance
- *Header 2x20* : จะเป็น Connector ตัวผู้ ขนาด 2x20 Pin เพื่อให้ผู้ใช้ต่อสัญญาณจาก MCU เข้ามาควบคุมการทำงานของตัวจอ LCD และ Touch Screen โดยมีรายละเอียดของขา ดังนี้



รูปที่ 2 ตำแหน่งขาสัญญาณที่ Header 2x20 (มองจากด้านหน้าตามรูปที่ 1 (A))

### รายละเอียด PIN สำหรับใช้ Control LCD

No. PIN	PIN-NAME	I/O	รายละเอียด
1	GND	Power LCD	ขา Ground
2	+5V	Power LCD	ขาไฟเลี้ยงบอร์ด +5 V
3	LCD CS-H	I	ขา Chip-Select : Low - ILI9320 จะถูกเลือกและส่งข้อมูลเข้าไปได้ High - ไม่สามารถติดต่อกับ ILI9320 ได้
4	RS-H	I	ขา Register-Select : Low- เลือกการเข้าถึง Index(IR) Register หรือ Status Register(SR) High- เลือกการเข้าถึง Control Register(Address 00H-98H)
5	WR-H	I	ขา Write Strobe จะทำหน้าที่ write data เมื่อได้รับสัญญาณ Low
6	RD-H	I	ขา Read Strobe จะทำหน้าที่ Read data ออกมาเมื่อได้รับสัญญาณ Low
7	RES-H	I	ขา Reset จะทำหน้าที่ Initial ILI9320 เมื่อได้รับสัญญาณ Low
8	BL-H	I	ขา Black Light ไฟ, Black Light ของ LCD จะติดเมื่อกันนี้ได้รับสัญญาณเป็น High
9-24	DB0-DB15	I/O	ขา data bus Bi-directional 16 bit ใช้ส่งผ่านข้อมูล หรือ ใช้ตำแหน่งแอดเดรสของรีจิสเตอร์

## รายละเอียด PIN สำหรับใช้ Control Touch Screen

No. PIN	PIN-NAME	I/O	รายละเอียด
25*-28 *	Y-,X-,Y+,X+	I	ขา Y-,X-Y+,X+ Position Touch Screen เมื่อ Control Touch Screen โดยไม่ผ่าน Chip ADS7846 ให้เลื่อน DIP SW.1-4 หลังบอร์ดมายังตำแหน่ง off
29	SCLK	I	เป็นขา DCLK ของ ADS7846 ใช้เพื่อ Synchronizes serial data I/O
30	TC CS	I	เป็นขา CS ของ ADS7846 เมื่อได้รับสัญญาณ Low จะเป็นการ Enable Serial I/O Register ของตัว Chip ให้เริ่มทำงาน
31	MOSI	I	เป็นขา DIN ของ ADS7846 เมื่อขา CS เป็น Low ข้อมูลจะถูก Latch ที่ขอบขาขึ้นของสัญญาณ DCLK
32*	BUSY	O	เป็นขา BUSY ของ ADS7846 จะเป็น High impedance เมื่อขา CS เป็น High (ไม่ใช้ก็ได้)
33	MISO	O	เป็นขา DOUT ของ ADS7846 เมื่อขา CS เป็น Low ข้อมูลจะถูก Shift ที่ขอบขาลงของ DCLK และ Output นี้จะเป็น high impedance เมื่อ CS เป็น High
34	PEN	O	เป็นขา PENIRQ ของ ADS7846 เมื่อมีการสัมผัสจอ Touch Screen จะให้สัญญาณ Logic ออกมาเป็น Low (ได้ Pull-Up R10K ไว้ให้ในบอร์ดแล้ว)
35-37	VTSC,+5V, +3V3	Power Touch-Screen	ใน 3 Pin นี้จะใช้เลือกไฟเลี้ยงให้กับตัว ADS7846 โดย ถ้าใช้กับ MCU 5V จะต้อง jump ขา VTSC(35) เข้ากับขา +5V(36) แต่ถ้าใช้กับ MCU 3.3V จะต้อง jump ขา VTSC(35) เข้ากับขา +3V3 (37)
38*	VBAT	I	เป็นขา Vbat ของ ADS7846 ซึ่งจะไม่นำมาใช้งานในส่วนของ Touch Screen
39*	VREF	I/O	เป็นขา Vref ของ ADS7846 ซึ่งจะไม่นำมาใช้งานในส่วนของ Touch Screen
40*	AUX	I	เป็นขา AUX input to ADC ของ ADS7846 ซึ่งจะไม่นำมาใช้ในส่วนของ Touch Screen

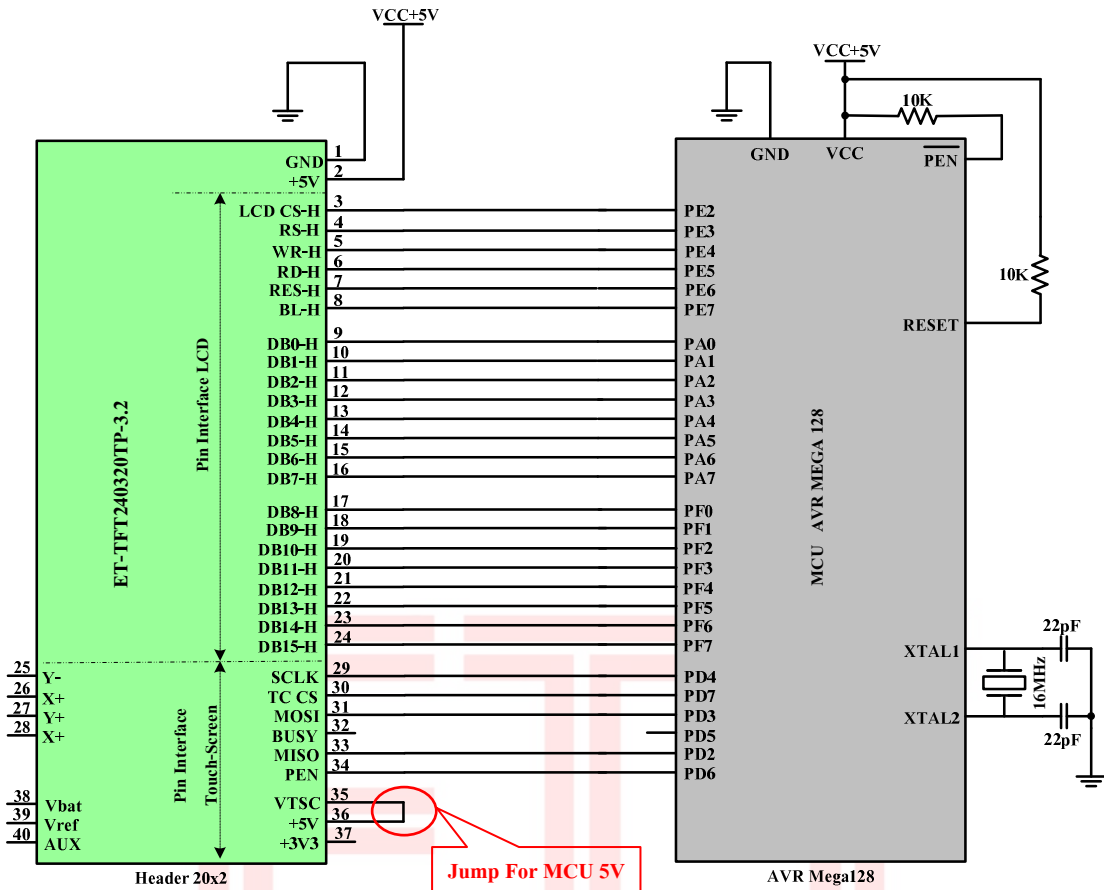
(\*) = ขาที่ไม่ได้ถูกนำมาต่อใช้งาน อ้างอิงกับตัวอย่างของอิตีทีทีให้มา

- DIP SW. 1-4 ด้านหลังบอร์ด : เป็น Dip SW. มีด้วยกัน 4 ตัว ซึ่งเมื่อต้องการใช้งานในส่วนของ Touch Screen โดย Interface แบบ SPI ผ่าน Chip Touch Screen Controller #ADS7846 ก็ให้เลื่อน Dip SW. ทั้ง 4 ไปที่ตำแหน่ง On (Default) ซึ่งในตัวอย่างโปรแกรมที่ให้มาก็จะเขียน Support การติดต่อในโหมดนี้มาให้เช่นกัน ในกรณีที่ผู้ใช้จะ Interface โดยใช้ขา X-,X+,Y-,Y+ ต่อเข้ากับขา ADC ของ MCU โดยตรง(ไม่ใช้งาน ADS7846) ก็ให้เลื่อน DIP SW. ทั้ง 4 ลงมายังตำแหน่งตรงข้ามกับตำแหน่ง Default

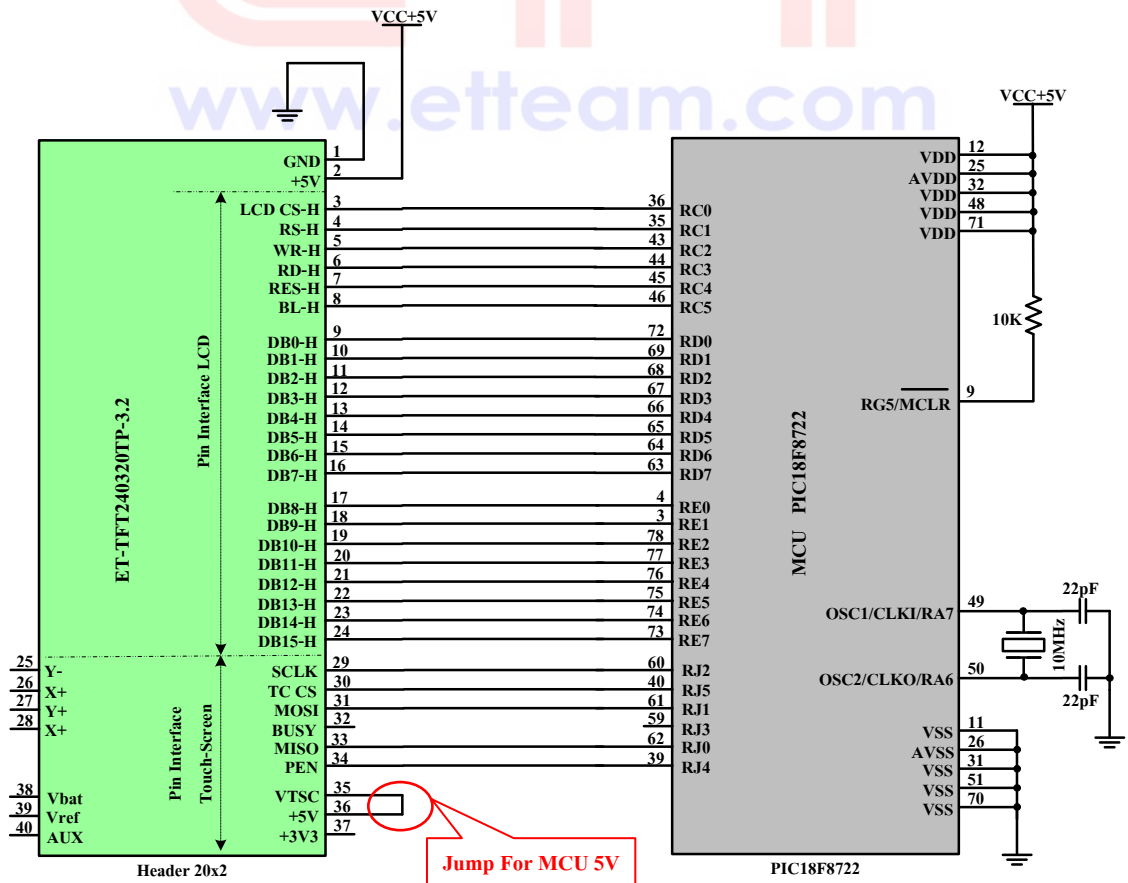
### 3. การนำบอร์ด ET-TFT240320TP-3.2 ไปต่อใช้งานกับ MCU

สำหรับการต่อใช้งานบอร์ด ET-TFT240320TP-3.2 ตามหัวข้อ 3.1) และ 3.2) นั้นจะรองรับกับตัวอย่างที่ทางอิตีทีทีเขียนมาให้ โดยใช้ MCU AVR MEGA128, PIC 18F8722 สำหรับการต่อกับ MCU แบบ 5 V และใช้ MCU ARM7 LPC2138 สำหรับการต่อกับ MCU แบบ 3.3V ซึ่งวงจรการต่อทั้ง 2 แบบนี้สามารถนำไปดัดแปลงเพื่อต่อกับ MCU เบอร์อื่น หรือ ตระกูลอื่นๆได้ แต่จะต้องระวังในส่วนของเขา VTSC จะต้องเลือก Jump กับขาไฟ +5V หรือ +3V3 ให้ถูกต้องตรงกับระดับไฟเลี้ยงของ MCU ที่นำมาต่อด้วย มิฉะนั้นอาจจะทำให้ MCU เสียหายได้ เช่น ถ้า MCU ทำงานที่ระดับไฟเลี้ยง 5 V ก็จะต้อง Jump ขา VTSC เข้ากับขา +5V เป็นต้น

3.1) การต่อใช้งานกับ MCU 5 V

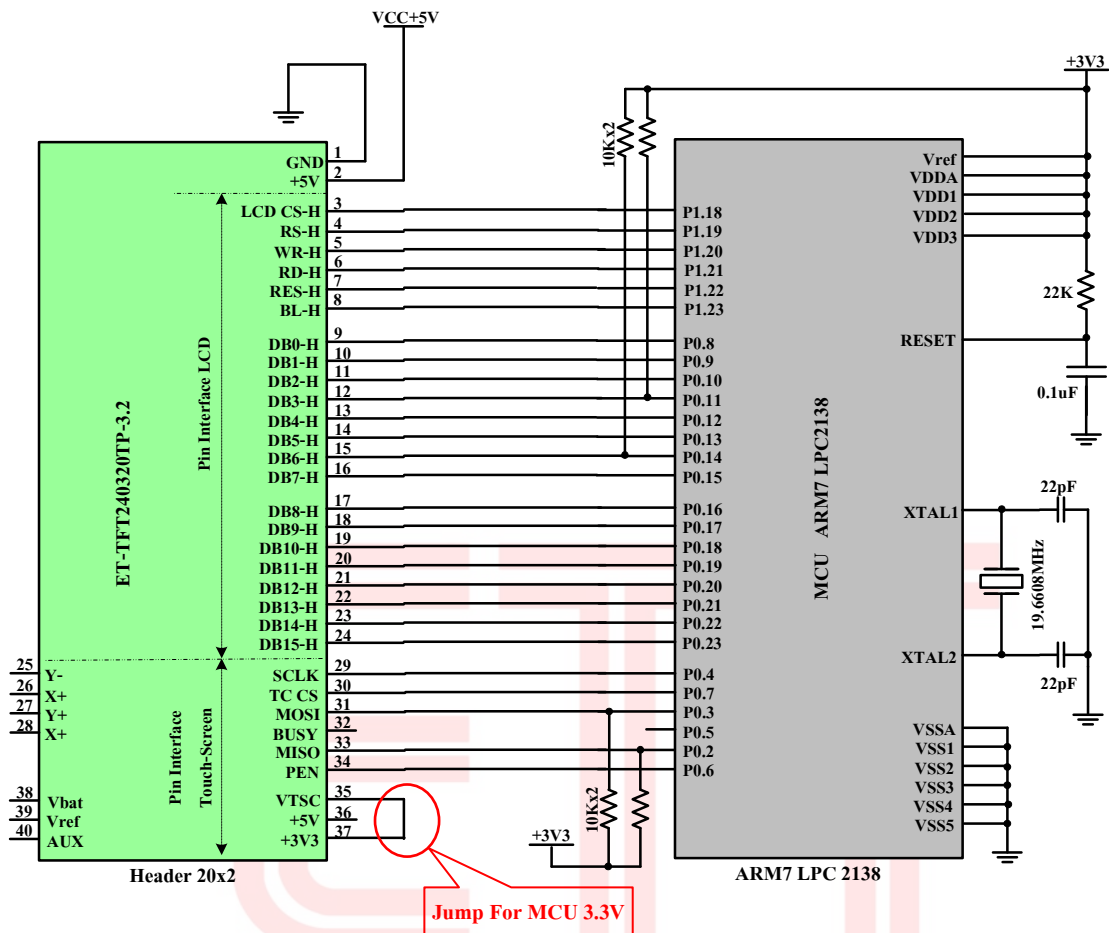


รูปที่3 ตัวอย่างการต่อบอร์ด ET-TFT240320TP-3.2 เข้ากับ MCU AVR #Mega 128 (5V)



รูปที่4 ตัวอย่างการต่อบอร์ด ET-TFT240320TP-3.2 เข้ากับ MCU PIC#18F8722 (5V)

3.2) การต่อใช้งานกับ MCU 3.3V สำหรับในวงจรนี้ สังเกตขา P0.2,P0.3,P0.11,P0.14 ของ MCU จะต่อ R Pull-Up เนื่องจากขา Port นี้เป็น Open Drain ซึ่งถ้า MCU ที่นำมาใช้ ไม่มีขาใดเป็น Open Drain ก็ไม่ต้องต่อ R Pull-up เข้าไป



รูปที่5 ตัวอย่างการต่อบอร์ด ET-TFT240320TP-3.2 เข้ากับ MCU ARM7 #LPC 2138

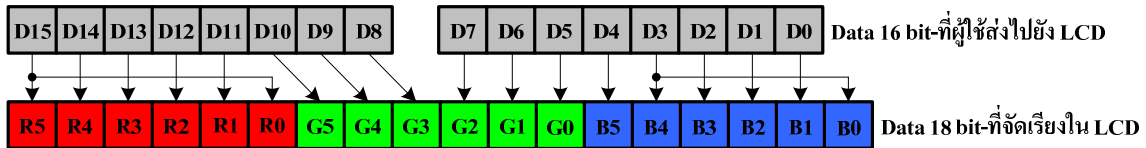
#### 4. หลักการ Control ติดต่อกับ LCD และ Touch Screen

ในการเขียนโปรแกรมติดต่อกับบอร์ด ET-TFT240320TP-3.2 นั้น เพื่อให้ง่ายในการเขียนโปรแกรมให้ผู้ใช้งานแยกการควบคุมการทำงานออกเป็น 2 ส่วน คือ ส่วนที่เป็นจอ LCD ซึ่งในส่วนนี้จะเป็นการ Interface แบบ Parallel และส่วนที่เป็น Touch Screen ซึ่งจะ Interface แบบ SPI โดยหลักการเขียนโปรแกรมควบคุมการทำงาน ในที่นี้จะขอใช้ตัวอย่างที่ทางอีทีทีให้มาในแผ่น CD เป็นตัวอ้างอิง ซึ่งในแต่ละตัวอย่างที่ให้มาใน CD นั้นจะใช้หลักการเบื้องต้นในการติดต่อกับบอร์ดเหมือนกันหมด จะต่างกันเพียงรูปแบบการแสดงผลออกหน้าจอเท่านั้น ซึ่งสามารถสรุปหลักการทั้ง 2 ส่วนได้ดังนี้

**4.1) การ Interface Control LCD** ในส่วนของคำสั่งที่ใช้ควบคุมตัว LCD นั้น ให้ผู้ใช้ดูรายละเอียดการใช้งานได้จาก data Sheet “GLCD\_ILI9320.pdf” ที่ให้มาใน CD ส่วนหลักการส่งคำสั่ง หรือ ส่ง Data ไปยัง LCD นั้นจะกล่าวถึงดังนี้

ก่อนอื่นให้ผู้ใช้เข้าใจก่อนว่า บอร์ด TFT240320TP-3.2 นี้ได้ถูกออกแบบไว้โดยตัว ให้ Interface ใน Mode 16 บิต (1 transfer/pixel) 65,536 color พุดง่ายก็คือ ผู้ใช้สามารถส่งข้อมูลขนาด 16 บิต 1 ครั้ง ไปยัง LCD ก็จะทำให้เกิดจุด 1 จุดหรือ 1 Pixel ปรากฏที่จอ LCD ดังนั้นก่อนจะทำการส่งข้อมูล ผู้ใช้จะต้องทำความเข้าใจโครงสร้างการเรียงบิตสีของข้อมูลที่จะทำการส่งออกไปก่อน เพื่อที่จะได้กำหนดสีที่จะใช้แสดงบนจอ LCD ได้ถูกต้อง ดังนี้

-โครงสร้างการจัดเรียงบิตสีของข้อมูล 16 บิต



รูปที่6 แสดงโครงสร้างการจัดเรียง data 16 bit Color (1 transfer/pixel)

จากรูปด้านบนเราจะเรียง Data บิตสี จากบิตสูงไปบิตต่ำเป็น RGB ซึ่งจะสอดคล้องกับตัวอย่างของอิตีที แต่ผู้ใช้สามารถจะเปลี่ยนให้เรียงบิตสีเป็น BGR ได้ โดยใช้คำสั่ง Entry Mode (R03H) เพื่อทำการ Set การเรียงบิตสีใหม่ซึ่งดูรายละเอียดเพิ่มเติมได้ใน Data Sheet

เวลาที่ผู้ใช้จะส่งข้อมูล เพื่อแสดงจุดบนจอ LCD ผู้ใช้จะต้องผสมสีเอาเอง โดยอ้างอิงการเรียงบิตสีตามรูปด้านบนคือ Data D15-D11(5bit) จะเป็นส่วนของสีแดง , Data D10-D5(6bit) จะเป็นส่วนของสีเขียว , Data D4-D0(5bit) จะเป็นส่วนของสีน้ำเงิน ซึ่งเมื่อผู้ใช้ส่ง Data เข้าไปยัง LCD แล้ว Data ก็จะถูกจัดเรียงข้อมูลใหม่เป็น 18 บิตอัตโนมัติตามรูป โดยความสว่างของสีจะไล่จาก มืด ไป สว่าง ซึ่งจะเรียงจากบิตต่ำไปหาบิตสูง เช่น ต้องการสีแดงที่มีความสว่างมากที่สุดก็จะได้ Data = 0xF800 หรือ ต้องการสีเขียวที่มีความสว่างน้อยสุด Data = 0x0020 เป็นต้น ถ้าต้องการสีอื่นนอกเหนือจาก 3 สีหลัก ผู้ใช้ก็จะต้องกำหนด บิต data ที่อยู่ในช่วงของแต่ละสีให้เหมาะสมก็จะได้สีออกมาตามที่ผู้ใช้ต้องการ เช่น ถ้าจะให้เป็นสีขาว ก็จะได้ Data = 0xFFFF หรือ สีดำ Data = 0x0000 เป็นต้น

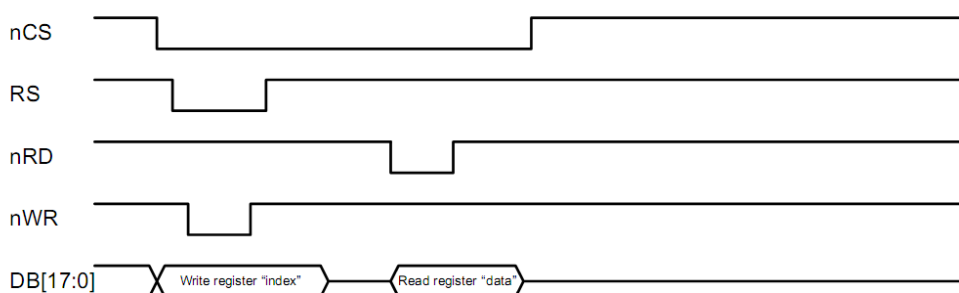
- Timing Diagram การส่งคำสั่ง และ Data

หลังจากที่ทราบการจัดเรียง Data บิตสีกันไปแล้ว ต่อไปเราจะมาดูขั้นตอนในการส่ง คำสั่ง และ Data ของคำสั่ง ออกไปยัง LCD ซึ่งสามารถสรุปได้ตาม Timing Diagram ในรูปด้านล่าง

(a) Write to register



(b) Read from register



รูปที่7 แสดง Timing Diagram ในการ Write/Read Data to LCD

จาก Timing Diagram จะขอกว่าเฉพาะในส่วนของการ Write Data ส่วนการ Read จะไม่ขอกว่าถึง เนื่องจากในตัวอย่างที่เราให้มาไม่ได้ใช้การ Read data กลับจาก LCD อาศัยการ Delay แทนซึ่งก็สามารถ Control LCD ได้เช่นกัน

ก่อนอื่นเมื่อพิจารณาจาก Timing Diagram ในส่วนของการ Write จะเห็นว่าในการส่งคำสั่งออกไปยัง LCD ในแต่ละคำสั่งนั้นเราจะส่งข้อมูลออกไป 2 ชุดด้วยกันคือ ในชุดแรกจะเป็นชุดของ คำสั่ง ซึ่งก็คือค่าตำแหน่ง Address ของ Register Index ของคำสั่งนั้นๆ ซึ่งจะมีขนาด 8 bit เช่น คำสั่ง Write Data to GRAM(R22h) ค่าตำแหน่ง Address ของ Register Index ของคำสั่งนี้ก็คือ 0x22 เป็นต้น ในชุดที่2 จะเป็นชุดของ Data ของคำสั่งนั้นๆ ซึ่งจะมีขนาด 16 bit เช่น เมื่อผู้ใช้ส่งคำสั่ง 0x22h ออกไปแล้วข้อมูลชุดที่ 2 ที่จะต้องส่งออกไปก็คือ Data สี สมมุติต้องการให้ปรากฏที่หน้าจอ LCD เป็นสีขาว 1 จุด ก็จะต้องส่ง data ออกไปคือ 0xFFFF เป็นต้น

ลักษณะของการส่งคำสั่ง Control LCD ในคำสั่งอื่นๆก็จะเป็นไปในลักษณะนี้ทั้งหมด เมื่อเรามองเห็นภาพกว้างๆในการส่งคำสั่งแล้ว ต่อไปเราจะมาดูรายละเอียดว่าเวลาส่งคำสั่ง ในส่วนของชุดคำสั่ง และชุด Data เราจะต้องกำหนดขา Control ต่างๆอย่างไรกันบ้างเพื่อให้ข้อมูลสามารถส่งเข้าไปยัง LCD ได้ถูกต้อง ซึ่งสามารถสรุปเป็นขั้นตอนตาม Timing Diagram ได้ดังนี้

#### สรุปขั้นตอนการส่งคำสั่งและส่ง Data Control LCD

- 1) กำหนดขา RD,CS ให้เป็น 1 ไว้
  - 2) กำหนดขา CS ให้เป็น 0
  - 3) ส่งชุดคำสั่ง 8 bit ออกไปที่ Data Bus 8 Bit ส่วน Data Bus 8 bit บน ให้ส่งค่า 0x00 ออกไป
  - 4) กำหนดขา RS ให้เป็น 0
  - 5) กำหนดขา WR ให้เป็น 0
  - 6) กำหนดขา WR ให้เป็น 1
  - 7) กำหนดขา RS ให้เป็น 1
- หลังจากส่งชุดคำสั่งไปแล้วต่อไปก็ทำตามด้วยการส่งชุด Data ของคำสั่งตามออกไปดังนี้
- 8) ให้ขา CS ยังคงเป็น 0 อยู่ ส่วนขา RS,RD ก็ยังคงเป็น 1 ค้างไว้
  - 9) ส่งชุด Data 16 bit ของชุดคำสั่งนั้นๆ ออกไปที่ Data Bus ทั้ง 16 บิต
  - 10) กำหนดขา WR ให้เป็น 0
  - 11) กำหนดขา WR ให้เป็น 1
  - 12) กำหนดขา CS ให้เป็น 1

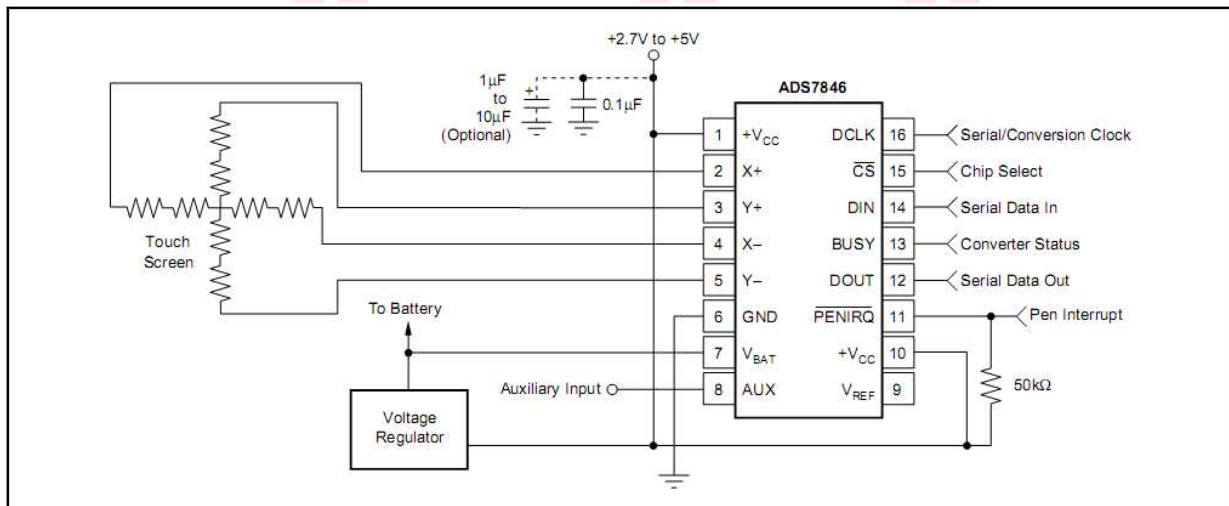
จากขั้นตอนที่กล่าวมานี้ เมื่อต้องการจะส่งคำสั่งอื่นๆต่อก็ให้วนกลับไปเริ่มในขั้นตอนแรกใหม่ ซึ่งในการเขียนโปรแกรมนั้นผู้ใช้อาจเขียนฟังก์ชันให้รับค่าคำสั่งและค่าของ Data เข้ามาในฟังก์ชันพร้อมกันเลยแล้วทำการส่ง คำสั่ง กับ data ยาวตาม Step ที่กล่าวข้างต้นก็ได้ หรือจะเขียนตามตัวอย่างของอิตีทีก็ได้ซึ่งจะแยกออกเป็น 2 ฟังก์ชันคือ ฟังก์ชันสำหรับส่งคำสั่ง และ ฟังก์ชันสำหรับส่ง Data

**4.2) การ Interface Control Touch Screen** ในส่วนของ Touch Screen นี้ จะแยกการ Control ออกมาจาก LCD ซึ่งในการ Control นั้นสามารถเลือกรูปแบบการ Interface ได้ 2 แบบ คือ Interface โดยใช้ขา Y-,Y+,X-,X+ ต่อเข้ากับขา ADC ของ MCU โดยตรงแล้วทำการเขียนโปรแกรมควบคุมการอ่านค่าเอาเอง ซึ่งจะทำให้เขียนโปรแกรมยาก ผู้ใช้จะต้องเข้าใจหลักการการทำงานของตัว Touch Screen จึงจะเขียนโปรแกรมได้ถูกต้อง ดังนั้นจะไม่แนะนำให้ใช้การ Interface แบบนี้

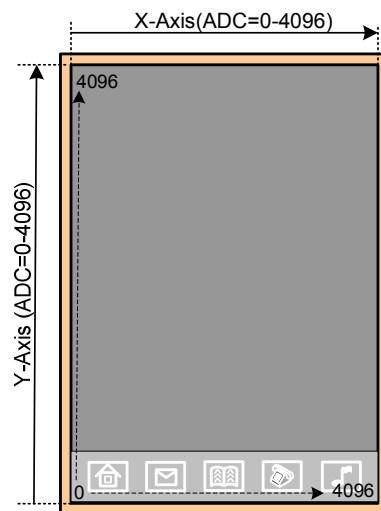


สำหรับการ Interface ที่จะแนะนำให้ใช้และจะสอดคล้องกับตัวอย่างที่ทางอีทีทีเขียนไว้ให้ด้วยซึ่งก็คือการ Interface ผ่าน Chip ADS7846 เมื่อผู้ใช้เลือกการ Interface แบบนี้ผู้ใช้จะต้องเลื่อน DIP SW.1-4 ที่อยู่หลังบอร์ดไปที่ตำแหน่ง ON เพื่อเป็นการเชื่อมต่อขา X+,Y+,X-,Y- ของ Touch Screen เข้ากับตัว Chip ADS7846 (ปกติจะถูก Set เป็นตำแหน่ง default ไว้แล้ว) ในการ Interface โดยใช้ Chip ADS7846 นี้จะใช้การ Interface แบบ SPI ระหว่าง MCU กับตัว Chip ซึ่งรายละเอียดในการติดต่อสื่อสารข้อมูลกับตัว Chip เพื่อทำการอ่านเขียนข้อมูลตำแหน่งของ Touch Screen มาใช้งาน สามารถดูเพิ่มเติมได้จาก Data Sheet “Touch\_ADC7846N.pdf” เพื่อความเข้าใจก่อนอื่นเราจะมาดูการทำงานร่วมกันของ Touch Screen กับ ADS7846

- **การทำงานของ Touch Screen ร่วมกับ ADS7846** ในการอ่านค่าตำแหน่งของ Touch Screen จะเริ่มจาก เมื่อผู้ใช้สัมผัสที่จอ Touch Screen ตัว Chip ADS7846 ก็จะทำการ Convert สัญญาณ Analog ที่รับเข้ามาทาง PIN X+,Y+,X-,Y- และส่งเป็นค่า Digital ออกมาทางขา Serial Data Out ค่า ADC ที่อ่านได้นี้จะมีความละเอียดที่ 12 บิต ดังนั้นค่าที่อ่านได้ทั้งทางแกน X และ Y จะอยู่ที่ 0-4095 ในขณะที่มีการสัมผัส Touch Screen นั้น ที่ขา PENIRQ ของ Chip ก็จะส่งสัญญาณ Interrupt logic “0” ออกมาชั่วขณะ เวลาที่เขียนโปรแกรมเราจะต้องอ่านค่าสถานะของสัญญาณ Interrupt นี้มาใช้สำหรับหลักคอยตรวจสอบ ว่ามีการสัมผัสจอ Touch Screen อยู่หรือไม่ เพื่อจะได้ไม่ต้องวนอ่านค่าตำแหน่งของ Touch Screen อยู่ตลอดเวลา จะอ่านเฉพาะเวลาที่มีการสัมผัสจอเท่านั้นซึ่งจะทำให้โปรแกรมสามารถไปทำงานในส่วนอื่นๆได้



รูปที่ 8 แสดง วงจรการต่อในส่วนของ Touch Screen เข้ากับ ADS7846

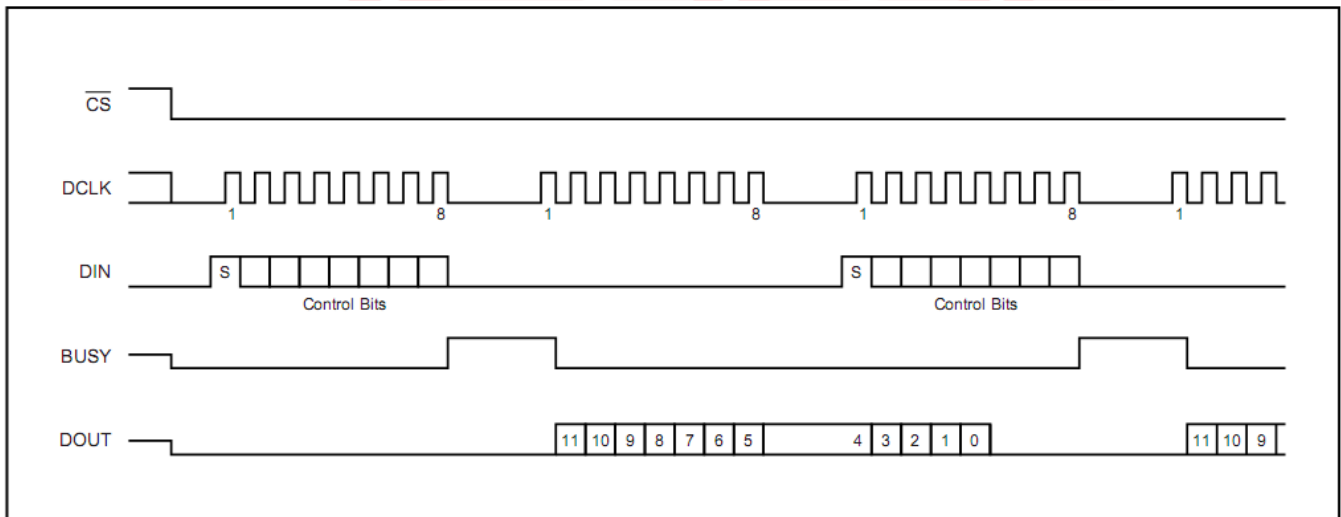


รูปที่ 9 แสดงขอบเขตของ Touch Screen และค่า ADC ที่เกิดขึ้นตามแนวแกน X และ Y



ในรูปที่ 9 จะแสดงทิศทางแนวแกน X และ Y ของ Touch Screen และค่า ADC ที่จะอ่านได้ตามจุดต่างๆเมื่อมีการสัมผัสจอ ซึ่งโดยปกติค่า ADC ที่อ่านได้โดยผ่าน ADS7846 นั้น ค่าต่ำสุดจะอยู่ที่ประมาณ 600 (ไม่เป็น 0) ซึ่งเป็นค่า offset ของจอโดยจอแต่ละจอจะอ่านค่าเริ่มต้นนี้ออกมาไม่เท่ากัน ดังนั้นเวลาใช้งานจริงจึงจำเป็นต้องเขียนโปรแกรมไว้สำหรับ Calibrate จอก่อนเสมอ ซึ่งผู้ใช้สามารถ Copy ตัวอย่างของ ETT ไปใช้งานได้เลย โดยจะให้หลักการของเมตริกเข้ามาช่วยในการคำนวณหาค่าสัมประสิทธิ์ ที่ได้จากการ Calibrate ของผู้ใช้ โดยในตัวอย่างโปรแกรมจะให้ผู้ใช้งาน Touch จุด 3 จุด ถ้าผู้ใช้ Touch ได้ตรงกับตำแหน่ง Mark ก็จะทำให้เวลาใช้งานจริงหลังจาก Calibrate แล้วจะมีความแม่นยำสูง

- **Timing Diagram การอ่านเขียนข้อมูลผ่าน ADS7846** หลังจากทราบหลักการการทำงานเบื้องต้นในส่วนของ Touch Screen ไปแล้ว ในหัวข้อนี้เราจะมาดูวิธี การอ่านค่า ADC จาก Touch Screen โดยใช้ Chip ADS7846 ต้องทำความเข้าใจก่อนว่าค่า ADC ที่อ่านมาได้จากการสัมผัสหน้าจอนี้ จะยังไม่ใช่ค่าตำแหน่งแอดเดรสจริงๆที่จะใช้อ้างอิงกับตำแหน่งบนจอ LCD ผู้ใช้จะต้องนำค่าที่ได้ไปเข้าสมาการ หาตำแหน่งจริงๆ ของจอ LCD อีกที เมื่อได้ตำแหน่งแอดเดรสที่แท้จริงแล้ว ถึงจะนำค่าตำแหน่งจริงนั้นไปใช้แทนลงในคำสั่งควบคุมตำแหน่งของจอ LCD อีกทีหนึ่ง ซึ่งกระบวนการทั้งหมดนี้สามารถดูได้จากตัวอย่างของอีทีที



รูปที่10 แสดง Conversion Timing Diagram,16 Clock-per-Conversion,8bit bus Interface

สำหรับ Timing Diagram นี้จะเป็นกระบวนการอ่านค่า ADC ที่ได้จากแผ่น Touch Screen ผ่านตัว Chip ADS7846 โดย จะใช้ MCU Interface กับ ADS7846 แบบ SPI ซึ่งจะสอดคล้องกับตัวอย่างของอีทีที โดยการสื่อสารแบบ SPI ตามตัวอย่างที่ให้มานั้นเราจะใช้ขา I/O ของ MCU สร้างเอง ไม่ใช่โมดูล SPI ภายในของ MCU เพื่อให้ผู้ใช้นำโปรแกรมไปแก้ไขได้ง่าย โดยในการส่งข้อมูลแบบ SPI นั้นมีหลักการอยู่ว่า เวลาเราส่งข้อมูลออกไปที่ขา MOSI(Dout) 1 bit แล้วตามด้วย Clock 1 ลูก ข้อมูลก็จะถูก Shift เข้าไปยัง Chip 1 บิต ในขณะเดียวกันตัว Chip ก็จะมี Shift ข้อมูลออกมาที่ขา MISO(Din) 1 บิต เช่นกันซึ่งจะเป็นข้อมูลที่เราต้องอ่านเก็บไว้ ดังนั้นจะเห็นว่าในตัวอย่างที่ให้มา ฟังก์ชันที่ tcs\_wr() จะทำหน้าที่เขียนและอ่านข้อมูล แบบ serial ขนาด 1 Byte(8bit) เมื่อเราสร้างฟังก์ชันขึ้นมาได้แล้ว ต่อไปเราก็จะทำการส่ง Control byte และอ่านค่า ADC กลับมาเก็บไว้โดยผ่านฟังก์ชันนี้ โดยขั้นตอนในการอ่านค่า ADC จาก Touch Screen จะมีลำดับตามด้านล่าง

ในการอ่านค่า ADC จาก ADS7846 นั้นผู้ใช้งานจะต้องส่ง Control Byte ไปยัง ADS7846 เพื่อกำหนดคุณสมบัติต่างๆ ให้กับตัว Chip ก่อนที่จะทำการอ่านค่ากลับมาทุกครั้ง โดยรูปแบบ Control Byte เป็นดังนี้

Bit7(MSB)	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
S	A2	A1	A0	MODE	SER/DFR	PD1	PD0

รูปที่11 Control Byte ของ ADS7846

รายละเอียดของ Control Byte ในแต่ละบิตสามารถดูได้ใน Data sheet จะไม่ขอกล่าวถึง สำหรับในตัวอย่างเราจะใช้ค่า Control Byte คือ 0xD0 สำหรับอ่านค่า ADC ในตำแหน่งแกน X ของ Touch Screen และค่า 0x90 สำหรับอ่านค่า ADC ในตำแหน่งแกน Y ของ Touch Screen

### สรุปขั้นตอนการอ่านค่า ADC จาก ADS7846

- 1) อ่านค่า Status จากขา PEN ของ ADS7846 ถ้าเป็น 0 (มีการสัมผัสจอ) ให้เริ่มกระบวนการอ่านค่าในขั้นตอนที่ 2 ต่อไป ถ้าอ่านได้ 1 (ยังไม่มีสัมผัสจอ) ให้วนอ่านซ้ำ
- 2) กำหนดขา DCLK, CS, DOUT ให้เป็น 0
- 3) ส่ง Control Byte 0xD0 ไปยังขา DIN (MOSI) ของ ADS7846 เพื่อระบุนการอ่านค่า ADC ในตำแหน่งแกน X ความละเอียด 12 บิต
- 4) ส่ง Data 0x00 ไปยังขา DIN (MOSI) ของ ADS7846 ซึ่งในขณะที่ส่ง data แต่ละบิตออกไป ตัว ADS7846 ก็จะ Shift ค่า ADC ออกมาที่ขา DOUT(MISO) ซึ่ง Data ที่ถูก Shift ออกมาบิตแรกจะเป็นบิตที่ 11 โดยจะเริ่มด้นที่ขอบขาลงของ DCLK ลูกที่ 2 เมื่อ DCLK ถูกส่งครบ 8 ลูก ก็จะอ่าน Data ในแกน X ได้ 0x0d d d d d d d (d=data bit11-bit5)
- 5) ส่ง Control Byte 0x90 ไปยังขา DIN (MOSI) ของ ADS7846 เพื่อระบุนการอ่านค่า ADC ในตำแหน่งแกน Y ความละเอียด 12 บิต ในขณะที่ส่ง Control Byte นี้ออกไป ค่า data ADC ในแกน X อีก 5 บิตสุดท้ายก็จะถูกส่งออกมา โดยเริ่มจาก bit4 ถึง bit0 โดยเรียง Data ดังนี้ 0x d d d d d d 000 (d=data bit4-bit0)
- 6) ส่ง Data 0x00 ไปยังขา DIN (MOSI) ของ ADS7846 ซึ่งในขณะที่ส่ง data แต่ละบิตออกไป ตัว ADS7846 ก็จะ Shift ค่า ADC ออกมาที่ขา DOUT(MISO) ซึ่ง Data ที่ถูก Shift ออกมาบิตแรกจะเป็นบิตที่ 11 โดยจะเริ่มด้นที่ขอบขาลงของ DCLK ลูกที่ 2 เมื่อ DCLK ถูกส่งครบ 8 ลูก ก็จะอ่าน Data ในแกน Y ได้ 0x0d d d d d d d (d=data bit11-bit5)
- 7) ส่ง Data 0x00 ไปยังขา DIN (MOSI) ของ ADS7846 ในขณะที่ส่ง Data ออกไป ค่า data ADC ในแกน Y อีก 5 บิตสุดท้ายก็จะถูกส่งออกมาโดยเริ่มจาก bit4 ถึง bit0 โดยเรียง Data ดังนี้ 0x d d d d d d 000 (d=data bit4-bit0)
- 8) เมื่ออ่านค่า ADC ทั้ง 2 แกนเรียบร้อยแล้วก็ให้ Set ขา CS เป็น 1 เพื่อจบการอ่านค่า จาก ADS7846
- 9) เมื่อจะอ่านค่าใหม่ก็ให้วนไปเริ่มต้นที่ขั้นตอนที่ 1 ใหม่
- 10) หลังจากได้ ค่า ADC ในแต่ละแกนมาแล้วจะต้องนำค่าที่ได้ของแต่ละแกนมาเรียงใหม่โดยตัวแปรที่ใช้เก็บค่า ADC ที่อ่านควรจะเป็นตัวแปรแบบ 16 บิต ซึ่งจาก data ADC ที่อ่านได้ 7 bit แรกเมื่อเก็บในตัวแปร 16 บิตจะได้เป็น 0x00000000 d d d d d d d d และ 5 bit หลังเมื่อเก็บในตัวแปร 16 บิตจะได้เป็น 0x00000000 d d d d d d 000 จากนั้นให้ Shift data ที่อ่านได้ 7 bit แรกไปทางซ้าย 5 bit และ shift data ที่อ่านได้ 5 bit หลัง ไปทางขวา 3 bit แล้วนำข้อมูลทั้ง 2 ชุดมา OR() กันก็จะได้ data ADC ของแกนที่อ่านได้ 12 บิต ดังนี้ 0x000 d d d d d d d d d d d d d d นี้คือค่าที่เราจะนำไปใช้งานต่อไป

จากหลักการ Control LCD และ Touch Screen ทั้งหมดที่กล่าวมานี้จะเป็นกระบวนการทำงานในภาพรวมของการใช้งานบอร์ด ET-TFT240320TP-3.2 เมื่อผู้ใช้พอทราบหลักการ Control เบื้องต้นแล้ว เวลาจะเขียนโปรแกรมจริงๆนั้น ผู้ใช้

สามารถคัดลอกในส่วนของฟังก์ชันในตัวอย่างที่ทางอิทีทีทำไว้ไปใช้ได้เลย โดยคุณหลักการนำฟังก์ชันไปใช้งานได้ในหัวข้อ “อธิบายตัวอย่าง โปรแกรม”

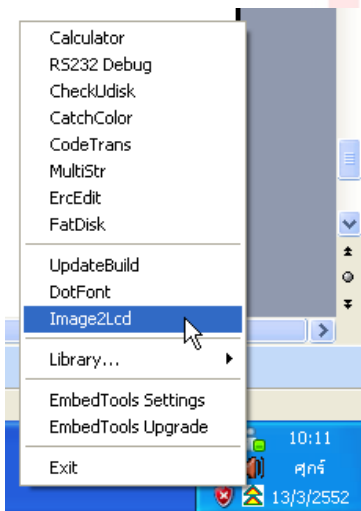
## 5. การใช้โปรแกรม Embedtools 3.31 Convert ไฟล์รูปภาพไปเป็น hex code

สำหรับในหัวข้อนี้จะกล่าวถึงวิธีการนำรูปภาพมาแปลงเป็น hex code เพื่อนำ hex code ที่ได้ส่งไปยัง LCD หน้าจอ LCD ก็จะแสดงรูปภาพออกมาให้เราตามต้องการ โดยเราจะใช้โปรแกรม “Embedtools 3.31” ซึ่งขั้นตอนการแปลงที่จะกล่าวดังต่อไปนี้ hex code ที่ได้ออกมาจากการแปลง จะรองรับกับตัวอย่างของอิทีที โดยจะใช้ฟังก์ชัน “plot\_picture()”(อยู่ในตัวอย่าง Ex3\_Touch\_Button) เป็นตัวส่ง hex code จาก MCU ไปยังตัว LCD

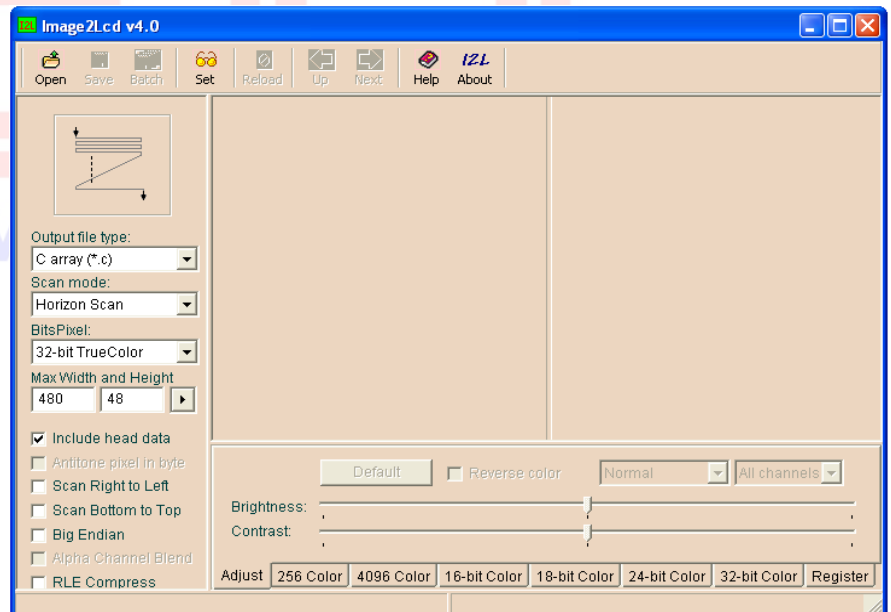
ในกรณีที่ผู้ใช้ไม่ได้ initial lcd ตามตัวอย่างของอิทีที และไม่ได้ Set โปรแกรม Embedtools ในการ Convert ภาพตามขั้นตอนที่กล่าวข้างล่าง ผู้ใช้จะไม่สามารถใช้ Function “plot\_picture()” ได้ เพราะจะทำให้ทิศทางการส่งข้อมูลนั้นผิดพลาดไปได้ ผู้ใช้จะต้องเขียนโปรแกรมการ plot รูปเอาใหม่ให้สอดคล้องกับการ Set ค่า ต่างๆที่ผู้ใช้ได้กำหนดเอง

### ขั้นตอนการใช้งาน โปรแกรม Embedtools Convert File รูปเป็น Hex Code

- 1.) ทำการติดตั้ง โปรแกรม Embedtools.exe ลงในเครื่อง (อยู่ใน Folder Embedtools3.31)
- 2.) หลังจากติดตั้งโปรแกรมเรียบร้อยแล้วจะมี icon (🖼️) ปรากฏอยู่ที่ Taskbar ดังรูปที่ 12 จากนั้นให้คลิกขวาที่ icon จะมีเด้บหัวข้อขึ้นมาให้คลิกซ้ายเลือกหัวข้อ Image2LCD เพื่อ Run โปรแกรม Embedtools ขึ้นมา จะได้หน้าต่างดังรูปที่13

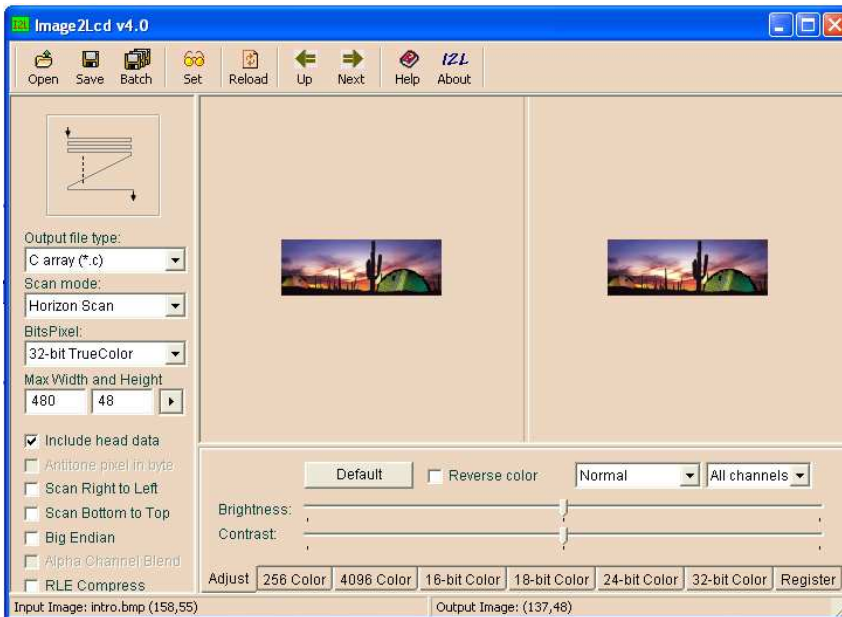


รูปที่12



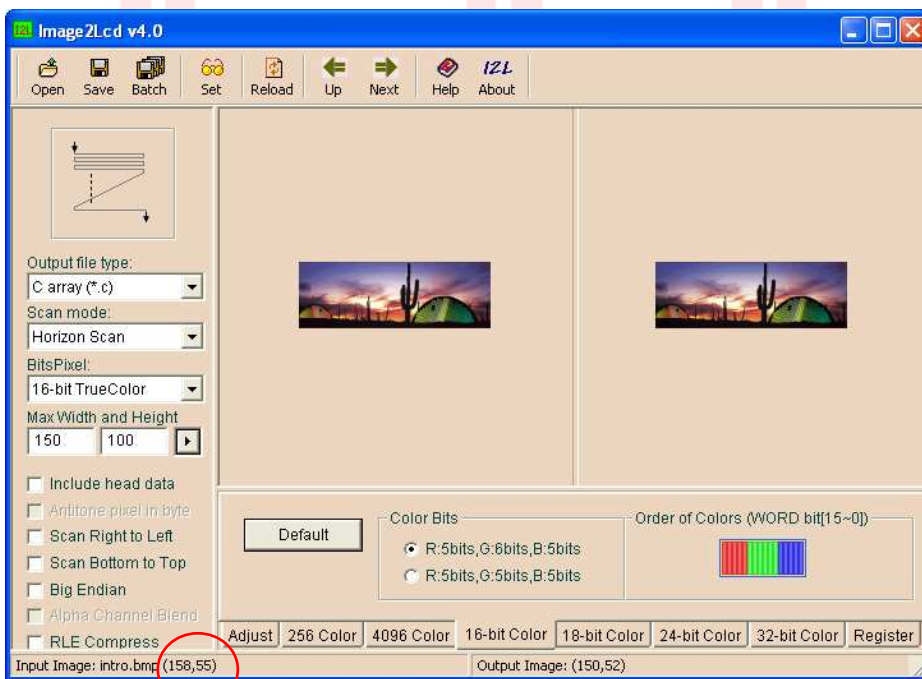
รูปที่13

- 3) คลิกที่ icon Open (📁) ด้านบนของโปรแกรมเพื่อเลือกไฟล์รูปที่จะทำการ Convert โดยไฟล์รูปควรจะเป็นไฟล์นามสกุล jpg หรือ bmp เมื่อเปิดรูปขึ้นมาจะได้หน้าต่างดังรูปที่14



รูปที่ 14

4) หลังจากโหลดรูปเข้ามาในโปรแกรมแล้วให้ทำการ Set ค่าดังนี้ (ดูรูปที่ 15 ประกอบ)




ขนาด Pixel จริงของรูป  
(ความกว้าง,ความสูง)

รูปที่ 15

- Output file type : *C array (\*.C)* = กำหนด data ให้อยู่ในรูป array ของภาษา C และ save ไฟล์ Output เป็นจุด C
- Scand mode : *Horizon Scan* = กำหนดทิศทางเริ่มต้น Scan data เวล่านำข้อมูลไปใช้จะต้องส่ง data ไปยัง LCD ตามทิศทางที่ สแกนด้วย
- Bits Pixel : *16-bit TrueColor* = กำหนดความละเอียดของบิตสี (ใน โปรแกรมจะมีค่านี้เหมือนกัน 2 ค่า ให้เลือกค่าบน)
- Max Width and Height : ความกว้าง,ความสูง = กำหนดความกว้างและความสูงให้กับขนาดรูป เมื่อกำหนดแล้วให้กด ที่ปุ่ม (▶) จะเห็นผลการเปลี่ยนแปลงของรูป ขนาดของรูปจะมีหน่วยเป็น Pixel ซึ่งจะไม่

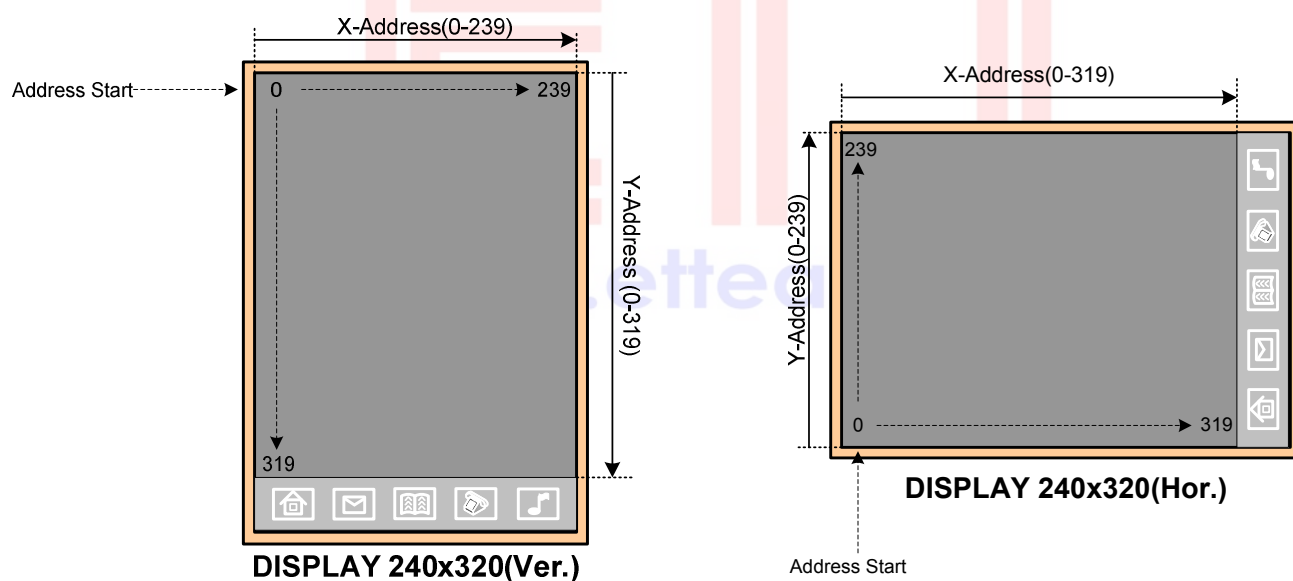
ใช้ขนาดที่แท้จริงของรูป ดังนั้นเวลาผู้ใช้เรียกใช้ฟังก์ชัน “plot\_picture()” ตัวอย่างของอิตีที จะต้องใช้ค่าความกว้างและความสูงของรูปที่แท้จริง โดยให้ดูที่ มุมซ้ายด้านล่างของหน้าต่าง ซึ่งจะ เป็นขนาด Pixel ที่แท้จริงของรูป

- Include head data = ให้ Tick เครื่องหมายถูกออก
- TAB 16-bit Color = คลิกที่ TAB 16-bit color ในช่อง Color bit ให้เลือกที่ช่อง R:5bit,G:6bit,B:5bit ส่วนในช่อง Order of Colors (WORD bit[15~0]) ให้เรียงสีเป็น RGB ส่วน TAB อื่นไม่ต้อง Set อะไร

5) เมื่อ Set ค่าต่างๆเรียบร้อยแล้ว ให้คลิกที่ Icon save (  ) เพื่อทำการ save file hex code เก็บไว้ เมื่อจะใช้งานก็ให้ใช้ note pad เปิด file ขึ้นมา จากนั้น ก็ Copy hex code ไปวางไว้ใน editor ที่ใช้เขียน โปรแกรม

## 6. การทำงานตัวอย่างโปรแกรม

สำหรับตัวอย่างของอิตีทีที่ให้มานั้นจะเขียนด้วยภาษาซี จะรองรับ MCU 3 ตระกูลได้แก่ AVR(Mega128) , PIC(18F8722),ARM7(LPC2138) ซึ่งในตัวอย่างจะมีทั้งการ Control LCD ในแนวตั้ง และแนวนอน ขึ้นอยู่กับผู้ใช้ว่าต้องการ ให้แสดงผลในแนวใด ก็ให้ดูตัวอย่างในแนวนั้นเป็นหลัก โดยตัวอย่างของ MCU แต่ละตระกูลนั้นจะเป็นตัวอย่างเหมือนกัน ซึ่งในตัวอย่างทั้งหมดนี้เราจะอ้างตำแหน่งแอดเดรสเริ่มต้นทางแกน X,Y ของจอตามทิศทางดังรูปด้านล่าง



รูปที่ 16 แสดงการอ้างแอดเดรสบนหน้าจอ LCD ในแนวตั้งและแนวนอน

จากรูปด้านบนเมื่อผู้นำฟังก์ชันตัวอย่างที่ให้มาไปใช้งาน เวลากำหนดตำแหน่งแอดเดรส X,Y สำหรับเริ่ม Plot รูปหรือตัวอักษร ผู้ใช้ก็ต้องมองตำแหน่งแอดเดรสอ้างอิงตามรูปด้านบน

สำหรับในตัวอย่างนั้นจะมีตัวอย่างหลักๆอยู่ด้วยกัน 3 ตัวอย่าง ไม่ว่าจะเป็นตัวอย่างในแนวนอน หรือแนวตั้งก็ตาม โดยมีรายละเอียดดังนี้



- **Ex1\_Touch\_Position** ในตัวอย่างนี้เมื่อ Run โปรแกรม เริ่มต้นจะให้ผู้ใช้ทำการ Touch ในตำแหน่งที่เป็นเครื่องหมาย + จำนวน 3 จุด เพื่อทำการ Calibrate ในส่วนของ Touch Screen เพื่อเวลาใช้งานหลังจาก Calibrate แล้ว เวลาผู้ใช้ Touch หน้าจอ จะทำ MCU สามารถอ่านตำแหน่งแอดเดรสได้ถูกต้องตรงกับตำแหน่งแอดเดรสจริงของ LCD ที่แสดงในรูปที่16 หลังจาก Calibrate เรียบร้อย เมื่อผู้ใช้ Touch ที่ตำแหน่งใดๆบนหน้าจอ LCD ที่หน้าจอด้านล่างก็จะแสดงค่าตำแหน่ง X,Y ในจุดที่ผู้ใช้ Touch ออกมาให้เห็น

- **Ex2\_Touch\_Draw** ในตัวอย่างนี้ เมื่อ Run โปรแกรมเริ่มต้น ก็จะต้อง Calibrate Touch Screen เหมือนกับตัวอย่างที่ 1 จากนั้นผู้ใช้สามารถทำการวาด หรือเขียนตัวหนังสือลงบนหน้าจอได้หน้าจอก็จะแสดงลายเส้นตามที่ใช้วาดหรือเขียน และสามารถ Touch ที่ Icon ตัวโน้ต (🎵) เพื่อเปลี่ยนสีเส้นที่จะวาด และ Touch ที่ Icon Home (🏠) เพื่อ Clear Screen

- **Ex3\_Touch\_Button** ในตัวอย่างนี้ เมื่อ Run โปรแกรม ก็จะให้ผู้ใช้ Calibrate Touch Screen เช่นเดิม จากนั้นก็จะแสดงปุ่ม สำหรับให้ผู้ใช้ Touch เมื่อผู้ใช้ Touch ที่ปุ่มใดๆบนหน้าจอ ในช่วงหน้าต่างที่วางอยู่ก็จะแสดง รูปต่างๆให้เห็นตามปุ่มที่กด

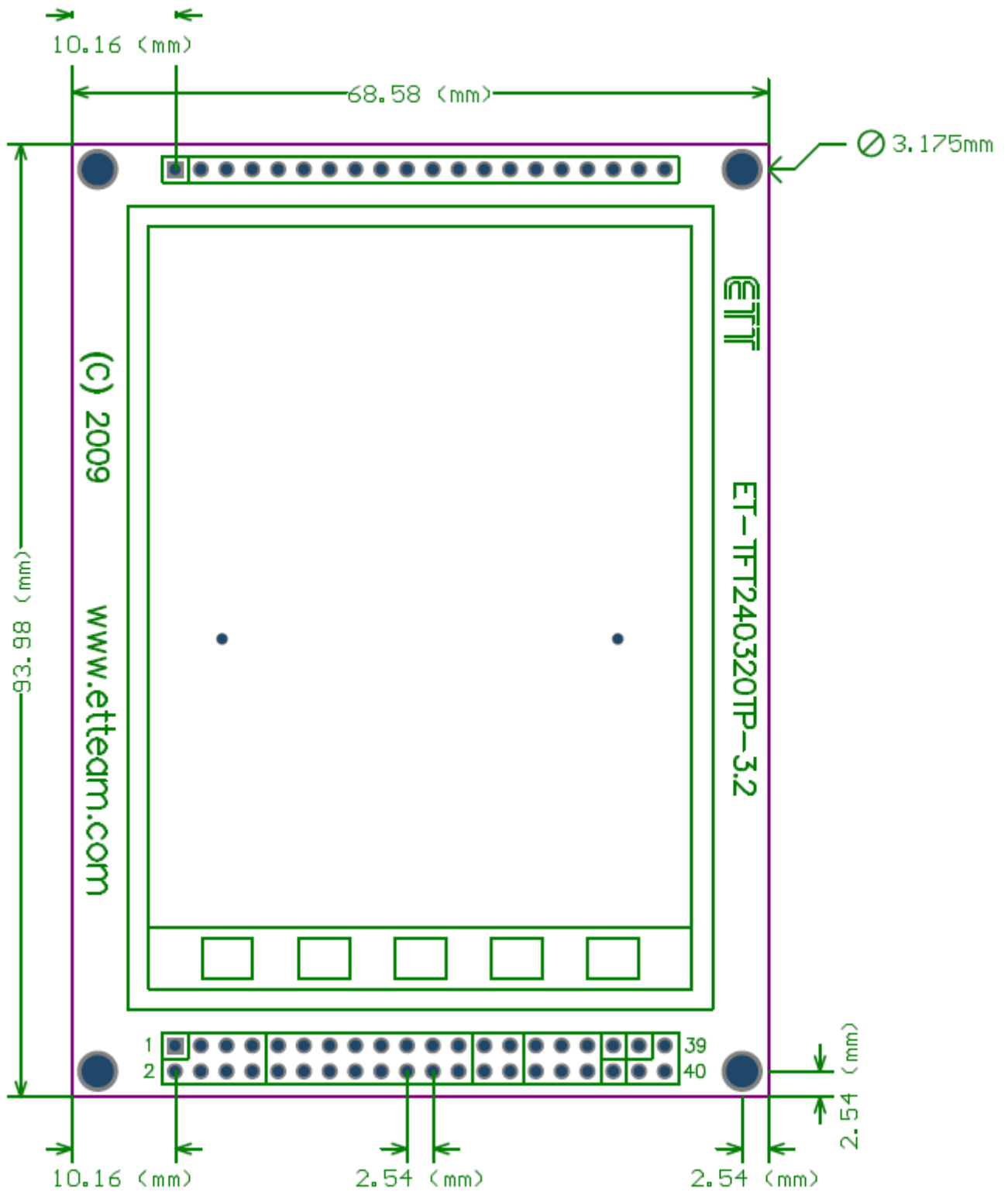
สำหรับในส่วนของการ Calibrate Touch Screen นั้น จากตัวอย่างจะเห็นว่าทุกครั้งที่เรา Reset MCU เราจะต้องทำการ Calibrate ใหม่เสมอ ซึ่งเวลาในการนำไปใช้งานจริงนั้นเราอาจจะต้องการ Calibrate เพียงครั้งเดียวเท่านั้น ซึ่งมีวิธีแก้โดย

**วิธีที่1** ให้ต่อ E2Promt เข้ากับ MCU เพิ่มเข้าไป (ถ้า MCU ที่ใช้ไม่มี E2PROMPT ภายใน) จากนั้นใน Function “touch\_calibrate()” ต่อจากบรรทัดการเรียกใช้ Function “set\_matrix()” ให้ผู้ใช้เขียนโปรแกรม write ค่าที่อยู่ในตัวแปร divider,An,Bn,Cn,Dn,En,Fn ไปเก็บไว้ใน E2Promt และ write ค่าอะไรก็ได้ อีก 1 Byte ไปเก็บไว้ใน E2Promt ด้วยเพื่อใช้เป็น Flag Status สำหรับใช้ตรวจสอบว่ามีการ Calibrate ไปแล้วหรือยัง

จากนั้นในส่วนของ main โปรแกรม ก่อนที่จะทำการเรียกใช้ฟังก์ชัน “touch\_calibrate()” ให้ผู้ใช้อ่านค่า Status Flag จาก E2Promt มาตรวจสอบก่อนว่าเป็นค่าเดียวกับค่าที่ผู้ใช้ได้ write เก็บไว้หรือไม่ ถ้าใช้ก็ไม่ต้องเรียกใช้งานฟังก์ชัน “touch\_calibrate()” อีก ให้ไปอ่านค่าของตัวแปร divider,An,Bn,Cn,Dn,En,Fn ที่ได้เก็บไว้ใน E2Promt ออกมา แล้วนำมาแทนค่าให้กับตัวแปร divider,An,Bn,Cn,Dn,En,Fn เหล่านี้เช่นเดิม แล้วก็ไปทำงานในส่วนอื่นของโปรแกรมต่อไป เท่านั้นผู้ใช้ก็จะไม่ต้องเสียเวลาในการ Calibrate ทุกครั้งเมื่อจะใช้งาน LCD

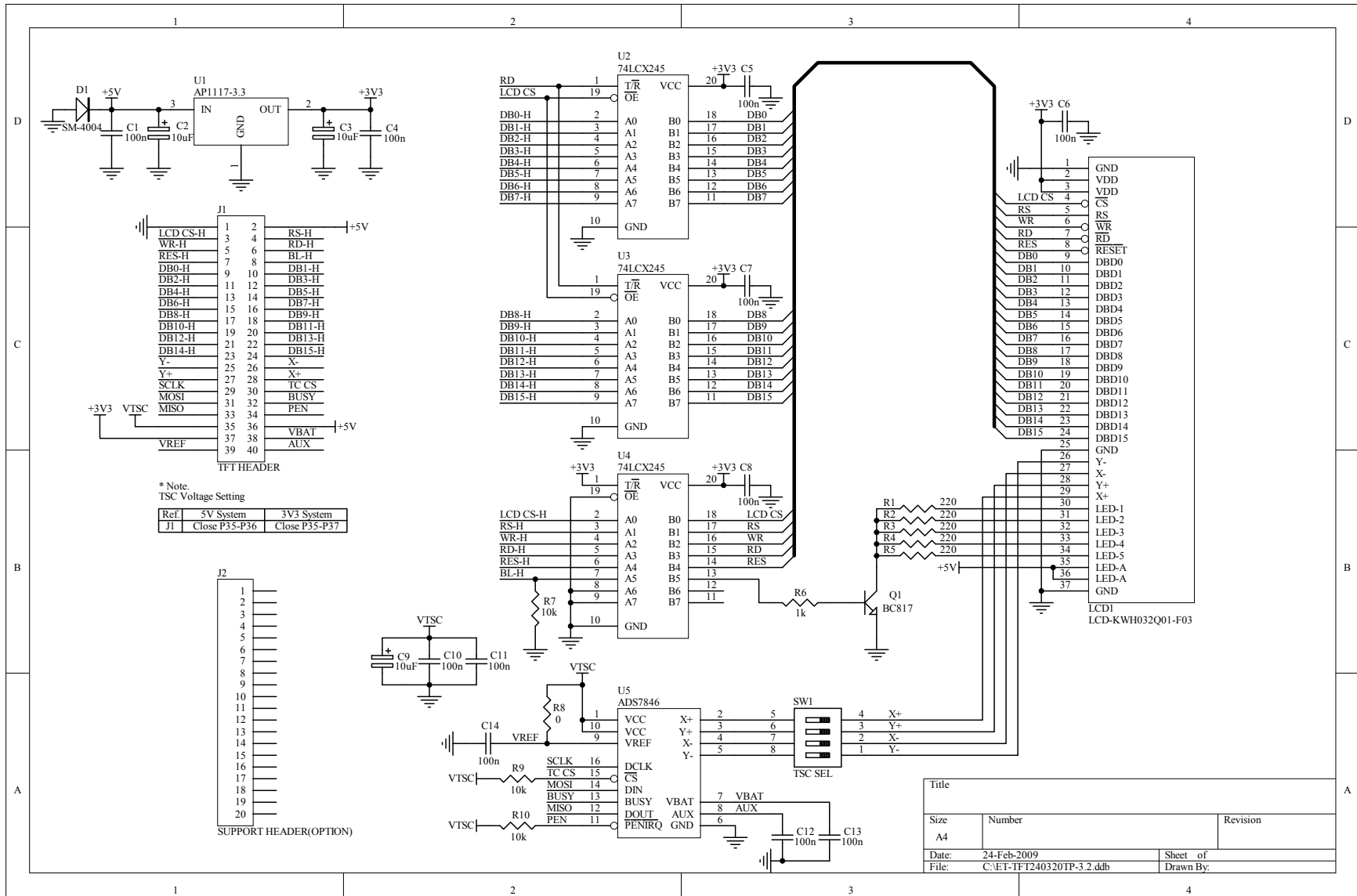
**วิธีที่2** วิธีนี้ไม่ต้องใช้ E2Promt ทำได้โดย ในฟังก์ชัน “touch\_calibrate()” ต่อจากบรรทัดการเรียกใช้ Function “set\_matrix()” ให้ผู้ใช้เพิ่มคำสั่ง printf()เข้าไป(ในตัวอย่างเขียนไว้ให้แล้วแต่ปิดการใช้งานไว้) เพื่อ print ค่าของตัวแปร divider,An,Bn,Cn,Dn,En,Fn ออกมาแสดงผลทาง RS232 โดยใช้โปรแกรม Hyper Terminal รับค่าของตัวแปรที่ print มาแสดง ให้ผู้ใช้เห็น จากนั้นให้ผู้ใช้ทำการจดค่าของตัวแปรแต่ละตัวที่แสดงเก็บไว้ แล้วนำค่าที่ได้ไปกำหนดให้กับตัวแปร divider,An,Bn,Cn,Dn,En,Fn ที่ประกาศไว้เหนือ main() เวลาจะใช้งานผู้ใช้ก็สามารถตัดฟังก์ชัน “touch\_calibrate()” ทิ้งได้ไม่ต้องเรียกใช้อีก ซึ่งวิธีนี้เวลาใช้งานจอ LCD หลายๆจอ ผู้ใช้อาจจะต้องเขียนโปรแกรมการ Calibrate เพื่อหาค่า divider,An,Bn,Cn,Dn,En,Fn ไว้ต่างหาก เนื่องจากถ้าผู้ใช้เปลี่ยนจอใหม่ถึงจะเป็นจอแบบเดียวกันผู้ใช้ก็ต้อง Calibrate หาค่าใหม่เสมอ ไม่สามารถใช้ค่าที่อ่านได้จากจอเก่ามาใช้กับจอใหม่ได้เพราะจะทำให้ค่าแอดเดรสที่อ่านได้จากการ Touch ไม่ตรงกับตำแหน่ง address ของจอ LCD จริงๆ

**ข้อควรจำ:** ในการ Calibrate นั้นจะต้อง Touch ให้ตรงกับตำแหน่งที่มาร์คไว้ให้มากที่สุดเพื่อความแม่นยำในการนำไปใช้งาน



รูปที่ 17 รูปขนาดของบอร์ด ET-TFT240320TP-3.2





\* Note.  
TSC Voltage Setting

Ref.	5V System	3V3 System
J1	Close P35-P36	Close P35-P37

Title		
Size A4	Number	Revision
Date: 24-Feb-2009	Sheet of	
File: C:\ET-IT1240320TP-3.2.ddb	Drawn By:	