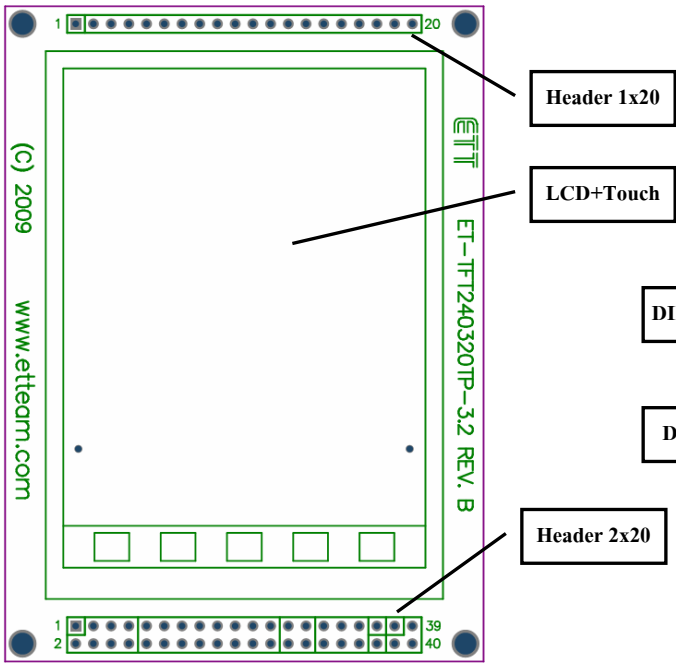
ET- TFT240320TP-3.2 REV.B

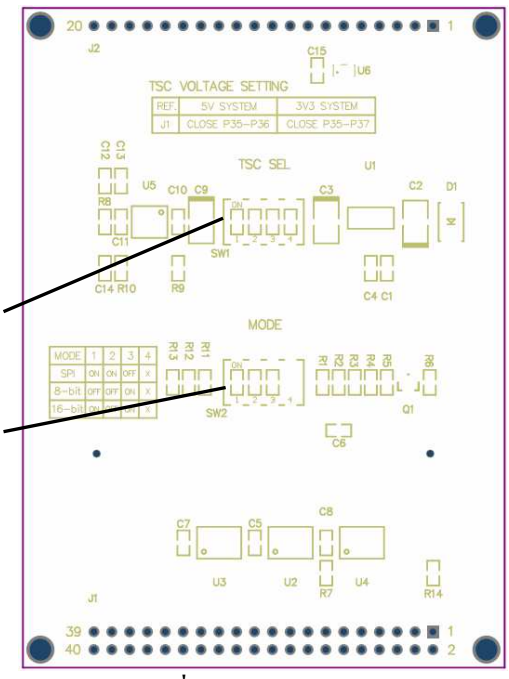
1. คุณสมบัติของบอร์ด ET-TFT240320TP-3.2 REV.B

- เป็น Display Module TFT LCD Color +Touch Screen ขนาด 240x320 Pixel
- ขนาดของหน้าจอ TFT 3.2 "
- ใช้ Single Chip Driver เบอร์ SPFD5408A
- ความละเอียดของสีแสดงได้ 65536 สี(RGB =R:5bit-G:6bit-B:5bit)
- เลือกการ Interface GLCD ผ่าน DIP SW2. MODE ได้ 3 Mode คือ 1) Parallel Mode 16-bit Interface
  2) Parallel Mode 8-bit Interface และ 3) Serial Mode SPI Interface
- ในส่วนของการควบคุม Touch Screen สามารถเลือก Interface ได้ 2 แบบด้วย DIP SW1.TSC SEL คือ Interface แบบ SPI โดย ผ่าน Chip Touch Screen Controller #ADS7846 (ADC มีความละเอียด 12 บิต) หรือ Interface โดยใช้ขา X-,X+,Y-Y+ ต่อเข้ากับขา ADC ของ MCU โดยตรงก็ได้ (การเขียนโปรแกรมควบคุมจะ ยุ่งยาก)
- ในการใช้งาน ถ้าไม่ต้องการใช้ Touch Screen สามารถ Control เฉพาะในส่วนของของตัว LCD อย่างเดียวก็ได้
- จำนวน I/O ของ MCU สำหรับใช้ ควบคุมในส่วนของ GLCD และ Touch Screen สรุปได้ดังนี้
  1) เมื่อใช้ใน Parallel Mode 16-bit Interface จะใช้ I/O ทั้งหมด 27 PIN
  2) เมื่อใช้ใน Parallel Mode 8-bit Interface จะใช้ I/O ทั้งหมด 19 PIN
  3) เมื่อใช้ใน Serial Mode SPI Interface จะใช้ I/O ทั้งหมด 10 PIN (ความถี่ SPI-Clock ขั้นต่ำ 10 MHz ถึงจะดี)
- สามารถต่อ Interface ได้ทั้ง MCU ที่ใช้ไฟเลี้ยง 5V และ 3.3V (ให้ดูรายละเอียดเพิ่มเติมในหัวข้อการต่อใช้งาน)
- Connector สำหรับต่อใช้งานใน Parallel Mode 8 bit และ16 bit จะใช้ Pin Header 2x20 ส่วนใน Serial Mode แบบ SPI จะใช้ Pin Header 1x20 ในการต่อใช้งาน
- ไฟเลี้ยงบอร์ด DC +5 V

2. ลักษณะและโครงสร้างของบอร์ด ET-TFT240320TP-3.2 REV.B



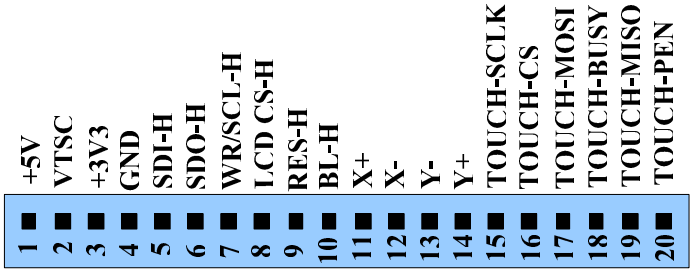
รูปที่2.1 (A) ด้านหน้าบอร์ด



รูปที่2.1 (B) ด้านหลังบอร์ด

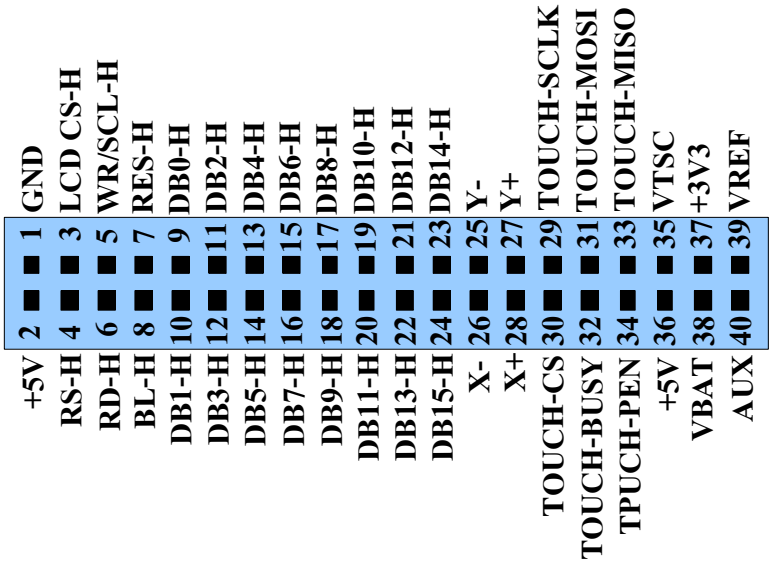
- *LCD+Touch* : จะเป็นเนื้อที่ของจอ LCD ขนาด 240x320 Pixel โดยด้านบนของจอจะถูกฉาบทับด้วยแผ่นของ Touch Screen แบบ Resistance

- *Header 1x20* : จะเป็น Connector ตัวผู้ ขนาด 1x20 Pin เพื่อให้ผู้ใช้ต่อสัญญาณในแบบ SPI –MODE จาก MCU เข้ามาควบคุมการทำงานของตัวจอ LCD และ Touch Screen ซึ่งหน้าที่ของ แต่ละขาจะแสดงดังตารางที่ 2.1 , 2.2



รูปที่2.2 ตำแหน่งขาสัญญาณที่ Header 1x20 (SPI-MODE) (มองจากด้านหน้าตามรูปที่2.1 (A))

- *Header 2x20* : จะเป็น Connector ตัวผู้ ขนาด 2x20 Pin เพื่อให้ผู้ใช้ต่อสัญญาณในแบบ Parallel-Mode (8,16 bit) จาก MCU เข้ามาควบคุมการทำงานของตัวจอ LCD และ Touch Screen ซึ่งหน้าที่ของแต่ละขาจะแสดงดังตารางที่ 2.1 , 2.2



รูปที่2.3 ตำแหน่งขาสัญญาณที่ Header 2x20 (Parallel-Mode:8,16 bit) (มองจากด้านหน้าตามรูปที่2.1 (A)

ตารางที่2.1 รายละเอียด PIN สำหรับใช้ Control GLCD

No. PIN		PIN-NAME	I/O	รายละเอียด
Header2x20 (Parallel Mode)	Header1x20 (SPI Mode)			
1	4	GND	Power LCD	ขา Ground
2	1	+5V	Power LCD	ขาไฟเลี้ยงบอร์ด +5 V
-	5	SDI-H	I	ขา Serial Data In จะใช้รับข้อมูล หรือ คำสั่งจาก MCU ซึ่งใช้ใน SPI Mode
-	6	SDO-H	O	ขา Serial Data Out จะใช้ส่งข้อมูลไปให้ MCU ซึ่งใช้ใน SPI Mode
3	8	LCD CS-H	I	ขา Chip-Select ใช้ Enable LCD Module (ใช้ทั้ง SPI และ Parallel Mode) โดย Low = LCD Module Enable และ High = LCD Module Disable
4	-	RS-H	I	ขา Register-Select : Low- เลือกการเข้าถึง Index(IR) หรือ Status (SR) Register High- เลือกการเข้าถึง Control Register(Address 00H-98H)
5	7	WR/SCL-H	I	ขา Write Strobe/Serial Clock จะทำหน้าที่ write data เมื่อได้รับสัญญาณ Low ใน Parallel Mode หรือ ให้เป็นขาสัญญาณ Clock ใน SPI Mode
6	-	RD-H	I	ขา Read Strobe จะทำหน้าที่ Read data ออกมาเมื่อได้รับสัญญาณ Low
7	9	RES-H	I	ขา Reset จะทำหน้าที่ Initial LCD Module เมื่อได้รับสัญญาณ Low
8	10	BL-H	I	ขา Black Light ไฟ Black Light ของ LCD จะติดเมื่อขานี้ได้รับสัญญาณเป็น High
9-24	-	DB0-DB15	I/O	ขา data bus Bi-directional 16 bit ใช้ส่งผ่านข้อมูล หรือ ชี้ตำแหน่งแอดเดรสของรีจีสเตอร์ จะใช้สำหรับ Parallel Mode

ตารางที่2.2 รายละเอียด PIN สำหรับใช้ Control Touch Screen

No. PIN		PIN-NAME	I/O	รายละเอียด
Header2x20 (Parallel Mode)	Header1x20 (SPI Mode)			
25*-28 *	11*-14*	**Y-,X-,Y+,X+**	I	ขา Y-,X-Y+,X+ เป็นขาใช้สำหรับอ่านตำแหน่ง Touch Screen โดยตรง ไม่ผ่าน Chip ADS7846 โดยจะต้องเลื่อน DIP SW1. TSC SCL มายังตำแหน่ง off ทั้งหมด
29	15	**TOUCH-SCLK**	I	เป็นขา DCLK ของ ADS7846ใช้เพื่อ Synchronizes serial data I/O
30	16	**TOUCH-CS**	I	เป็นขา CS ของ ADS7846 เมื่อได้รับสัญญาณ Low จะเป็นการ Enable Serial I/O Register ของตัว Chip ให้เริ่มทำงาน
31	17	**TOUCH-MOSI**	I	เป็นขา DIN ของ ADS7846 เมื่อขา CS เป็น Low ข้อมูลจะถูก Latch ที่ขอบขาขึ้นของสัญญาณ DCLK
32*	18*	**TOUCH-BUSY**	O	เป็นขา BUSY ของ ADS7846 จะเป็น High impedance เมื่อขา CS เป็น High
33	19	**TOUCH-MISO**	O	เป็นขา DOUT ของ ADS7846 เมื่อขา CS เป็น Low ข้อมูลจะถูก Shift ที่ขอบขาลงของ DCLK และ Output นี้จะเป็น high impedance เมื่อ CS เป็น High

ตารางที่2.2 (ต่อ)

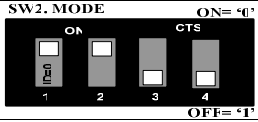
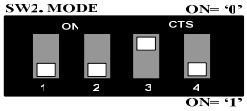
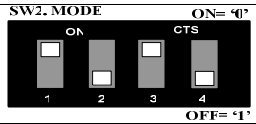
No. PIN		PIN-NAME	I/O	รายละเอียด
Header2x20 (Parallel Mode)	Header1x20 (SPI Mode)			
34	20	TOUCH-PEN	O	เป็นขา PENIRQ ของ ADS7846 เมื่อมีการสัมผัสจอ Touch Screen จะให้สัญญาณ Logic ออกมาเป็น Low (ได้ Pull-Up R10K ไว้ให้ในบอร์ดแล้ว)
35,36,37	1,2,3	VTSC,+5V, +3V3	Power Touch Screen	ใน 3 Pin นี้จะใช้เลือกไฟเลี้ยงให้กับตัว ADS7846 โดย ถ้าใช้กับ MCU 5V จะต้อง jump ขา VTSC เข้ากับขา +5V แต่ถ้าใช้กับ MCU 3.3V จะต้อง jump ขา VTSC 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 ADCของ ADS7846 ซึ่งไม่ได้นำมาใช้งานในส่วนนี้

(*) = ขาที่ไม่ได้ถูกนำมาต่อใช้งาน อ้างอิงกับตัวอย่างของอีทีทีที่ให้มา

- *DIP SW1. TSC SEL ด้านหลังบอร์ด :* เป็น Dip SW.มีด้วยกัน 4 ตัว ซึ่งเมื่อต้องการใช้งานในส่วนของ Touch Screen โดย Interface แบบ SPI ผ่าน Chip Touch Screen Controller #ADS7846 ก็ให้เลื่อน Dip SW. ทั้ง4ไปที่ตำแหน่ง 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 (Off)

- *DIP SW2.MODE ด้านหลังบอร์ด :* เป็น Dip SW. มีด้วยกัน 4 ตัว โดยจะใช้งานเพียง 3 ตัว คือ S1,S2 และ S3 โดย Dip SW2. นี้จะมีไว้สำหรับให้ผู้ใช้เลือก Mode การ Interface ระหว่างบอร์ด GLCD กับ MCU ที่นำมาต่อควบคุม โดยการเลือกโหมดจะเป็นไปตามตารางที่2.3

ตารางที่ 2.3 แสดงการเลือก Mode Interface LCD จาก DIP SW.2

DIP SW.2 MODE				รูปการ Set SW2.	MODE Interface
S1	S2	S3	S4		
ON/OFF*	ON	OFF	X		SPI-Mode
OFF	OFF	ON	X		Parallel 8-bit Mode
ON	OFF	ON	X		Parallel 16-bit Mode

X = เป็นอะไรก็ได้ ; ON/OFF* = ใน SPI Mode สวิตช์ S1 จะทำหน้าที่เลือก ID โดย ON= ให้ ID เป็น 0 และ OFF = ให้ ID เป็น 1 ในตัวอย่างโปรแกรมที่ให้มาจะใช้ ID เป็น 0 ดังนั้นต้อง Set สวิตช์ S1 ในตำแหน่ง ON

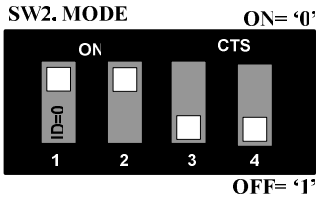
3. การนำบอร์ด ET-TFT240320TP-3.2 REV.B ไปต่อใช้งานร่วมกับ MCU

สำหรับการต่อใช้งานบอร์ด ET-TFT240320TP-3.2 REV.B ที่จะกล่าวถึงต่อจากนี้นั้นจะรองรับกับตัวอย่างที่ทางอีทีทีเขียนมาให้ โดยใช้ MCU AVR MEGA128, PIC 18F8722 (สำหรับ MCU แบบ 5 V) และใช้ MCU ARM7 LPC2138 (สำหรับ MCU แบบ 3.3V) ซึ่งวงจรการต่อ MCU ที่แสดงในรูปตัวอย่างด้านล่างนี้สามารถนำไปดัดแปลงเพื่อต่อกับ MCU เบอร์อื่น หรือ ตระกูลอื่นๆได้ แต่ ***จะต้องระวังในส่วนของขา 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

ในการต่อใช้งานนั้นผู้ใช้สามารถเลือกรูปแบบการต่อได้ 3 Mode Interface คือ Serial Interface SPI-Mode , Parallel Interface 8-bit Mode และ Parallel Interface 16-bit Mode โดยการเลือกโหมด Interface สามารถเลือกได้จาก DIP-SW2 ที่อยู่ด้านหลังบอร์ด สามารถสรุปตัวอย่างการต่อใน Mode ต่างๆได้ดังนี้

3.1) Serial Interface SPI-MODE ในโหมดนี้ SPI CLOCK ควรจะใช้ที่ความถี่อย่างน้อย 10 MHz ขึ้นไปเพื่อการตอบสนองในการแสดงผลบนหน้าจอ LCD ของ รูป หรือข้อความ ที่ถูกส่งมาจาก MCU ได้เร็วพอ

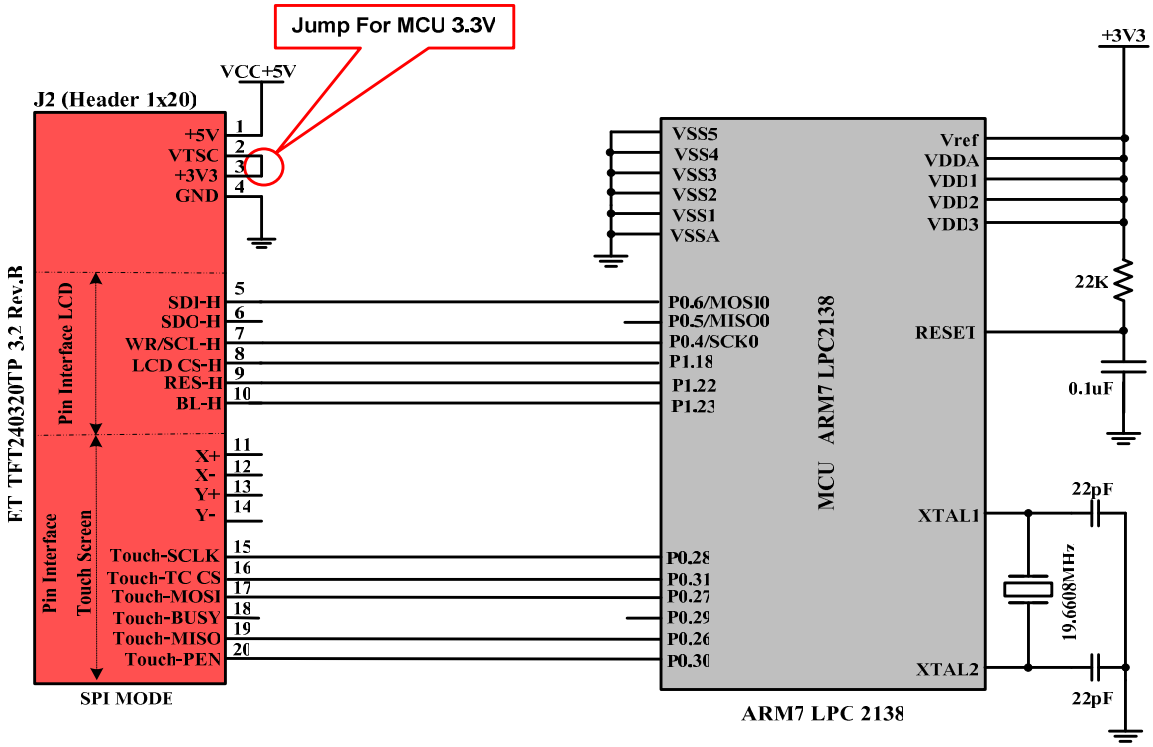
การ Set Mod Interface



Interface Mode : SPI (ID=0)
(Use PIN I/O = 10 PIN)

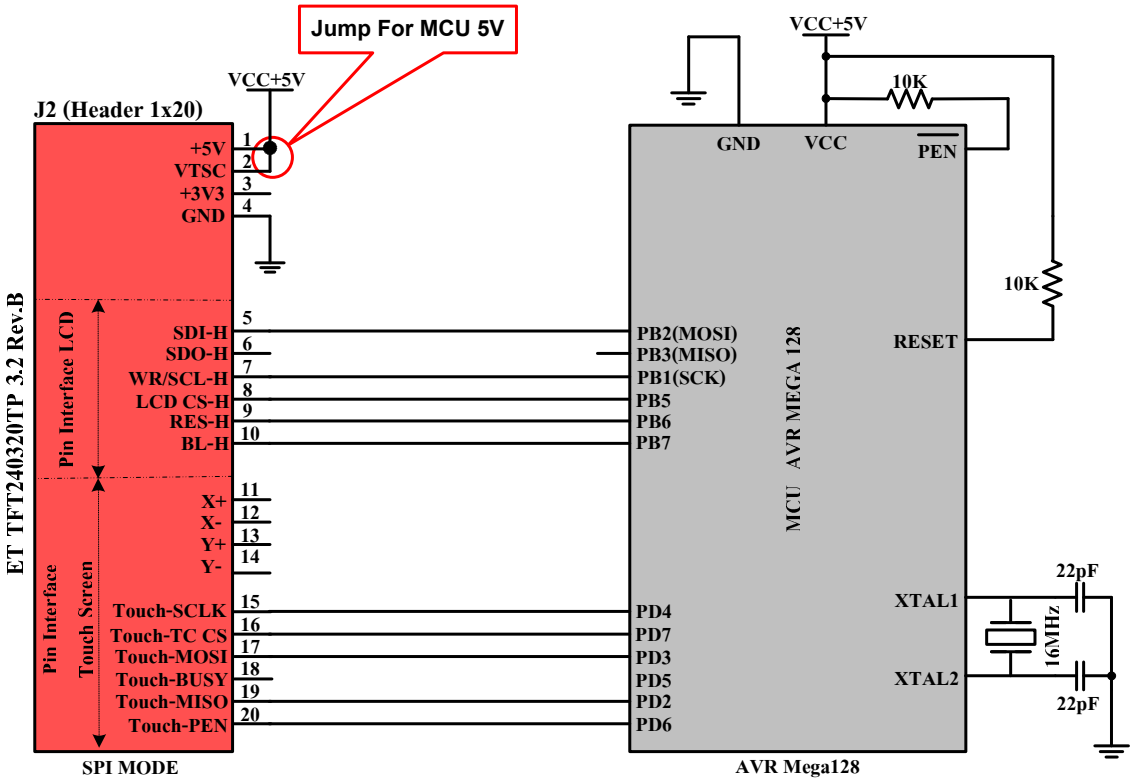
การต่อใช้งานกับ MCU 3.3V

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

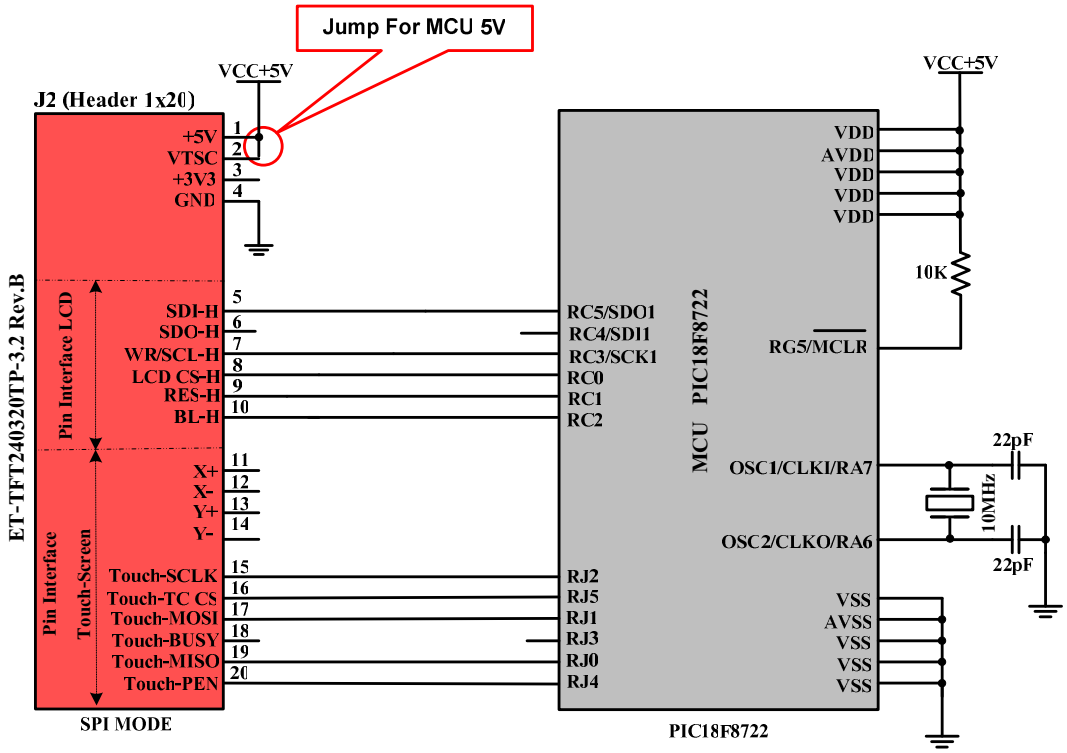


รูปที่3.1A ตัวอย่างการต่อบอร์ด ET-TFT240320TP-3.2 REV.B เข้ากับ MCU ARM7 #LPC 2138 (3.3V)

การต่อใช้งานกับ MCU 5 V

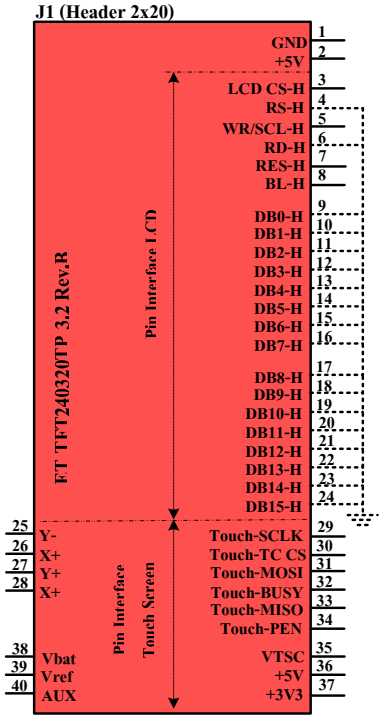


รูปที่3.1B ตัวอย่างการต่อบอร์ด ET-TFT240320TP-3.2 REV.B เข้ากับ MCU AVR #Mega 128 (5V)



รูปที่ 3.1C ตัวอย่างการต่อบอร์ด ET-TFT240320TP-3.2 REV.B เข้ากับ MCU PIC#18F8722 (5V)

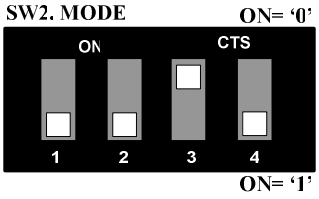
สำหรับ SPI Mode นี้ ในส่วนของ PIN Interface LCD ที่ไม่มีการต่อใช้งาน ที่ขั้วต่อ J1 (Header 2x20) ก็ควรจะต่อลงกราวด์ดังแสดงในรูปที่ 3.1D ตามเส้นประ เพื่อกำหนดสถานะทาง Logic ให้กับขาที่ไม่ได้ใช้งานของ LCD ที่แน่นอน (ในการทดลองใช้งานถ้าไม่ต่อลงกราวด์ปล่อยลอยไว้เฉยๆ ก็ยังไม่พบปัญหาในการใช้งานนะครับ)



รูปที่3.1D แสดงการต่อขาสัญญาณ PIN Interface LCD ที่ไม่มีการใช้งานลงกราวด์ตามเส้นประ

3.2) Parallel Interface 8-bit MODE

การ Set Mod Interface

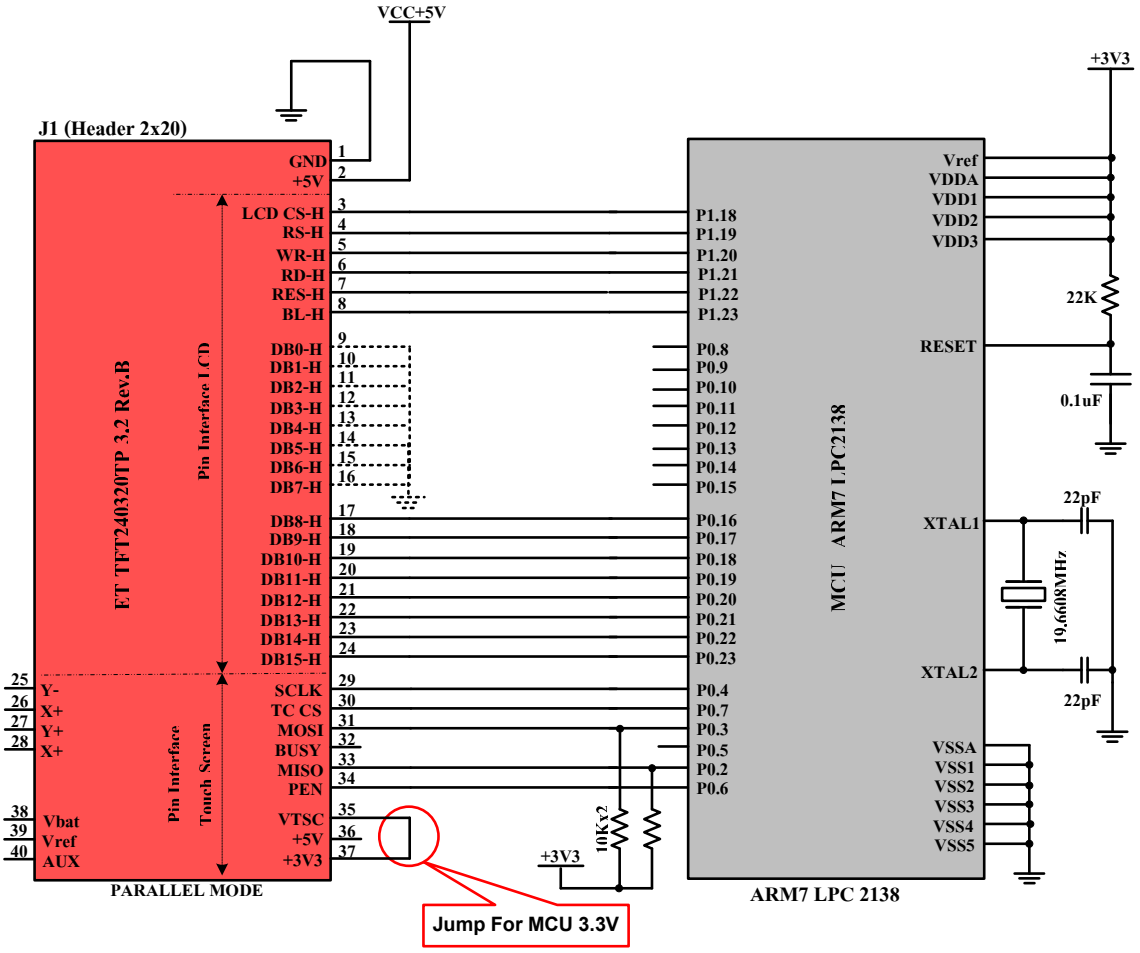


Interface Mode :Parallel 8 bit data
(Use PIN I/O = 19 PIN)

สำหรับ Parallel 8-bit Mode นี้ เราจะใช้ขา data ในการรับส่งข้อมูลเพียง 8 Bit คือ DB8-DB15 ในส่วนของ PIN Interface LCD ที่ไม่ได้ใช้งาน คือ DB0-DB7 ก็ควรจะต่อลงกราวด์ดังรูปตามเส้นประ เหตุผลก็เช่นเดียวกับใน SPI-Mode (ในการใช้งานจริงถ้าไม่ต่อก็ยังไม่พบปัญหาอะไรครับ)

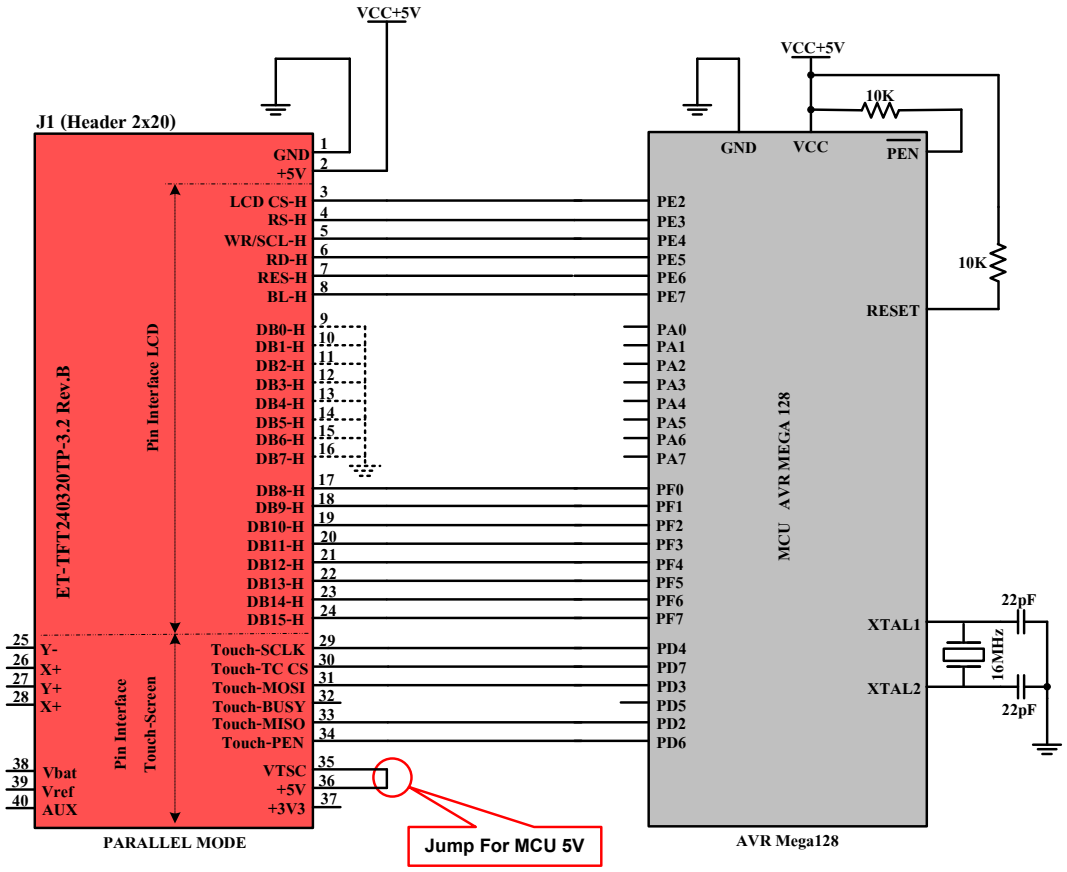
การต่อใช้งานกับ MCU 3.3V

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

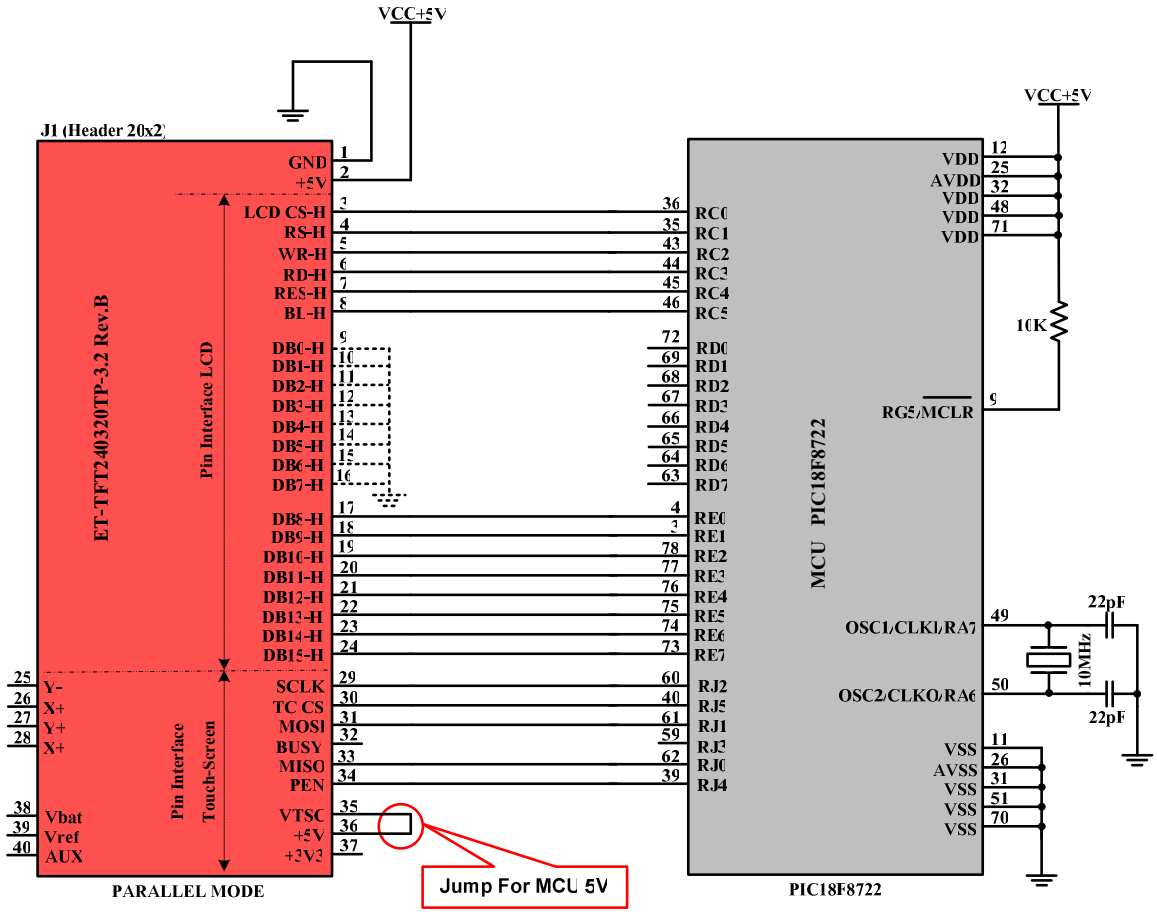


รูปที่3.2A ตัวอย่างการต่อบอร์ด ET-TFT240320TP-3.2 REV.B เข้ากับ MCU ARM7 #LPC 2138 (3.3V)

การต่อใช้งานกับ MCU 5 V



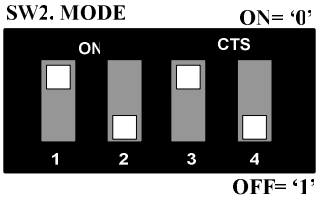
รูปที่3.2B ตัวอย่างการต่อบอร์ด ET-TFT240320TP-3.2 REV.B เข้ากับ MCU AVR #Mega 128 (5V)



รูปที่ 3.2C ตัวอย่างการต่อบอร์ด ET-TFT240320TP-3.2 REV.B เข้ากับ MCU PIC#18F8722 (5V)

3.3) Parallel Interface 16-bit MODE

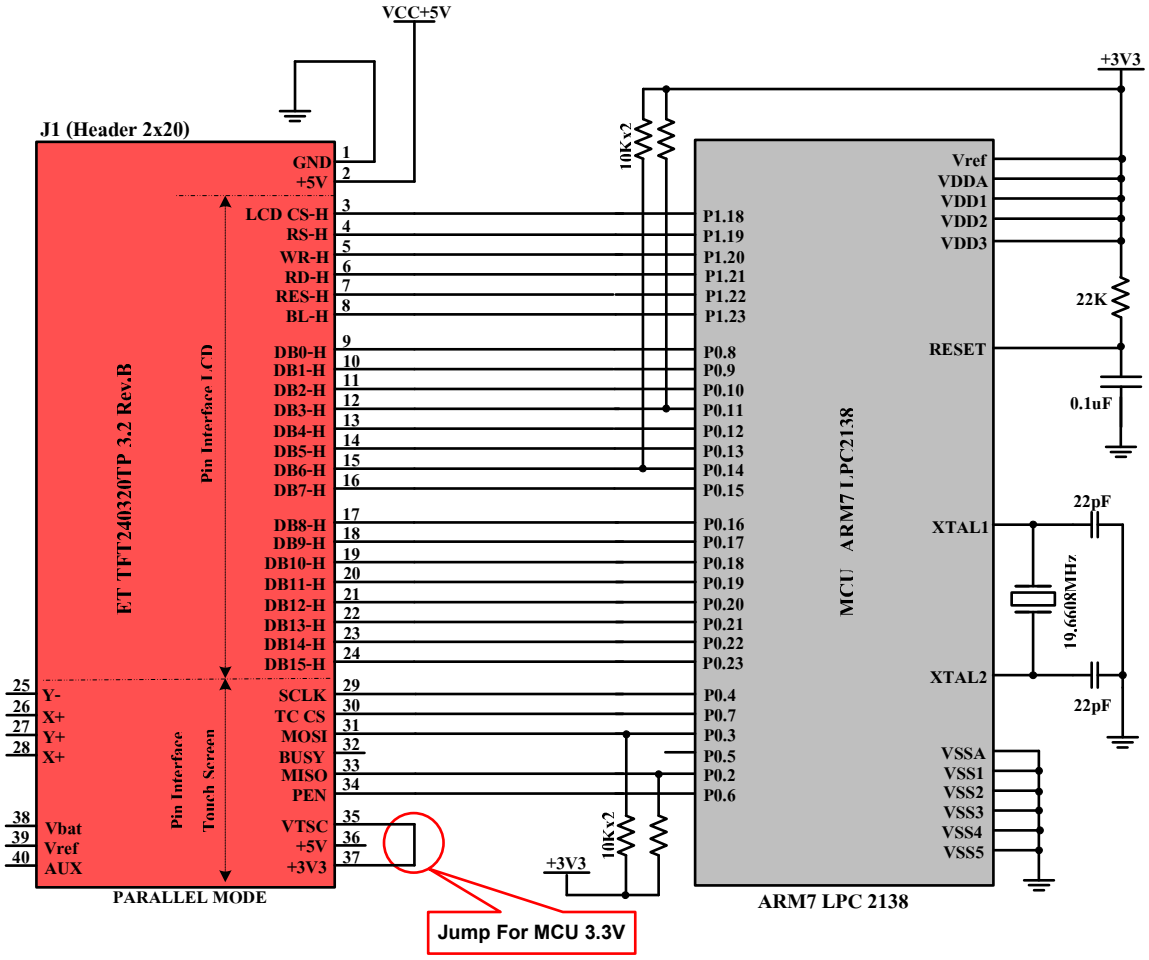
การ Set Mod Interface



Interface Mode : Parallel 16 bit data
(Use PIN I/O = 27 PIN)

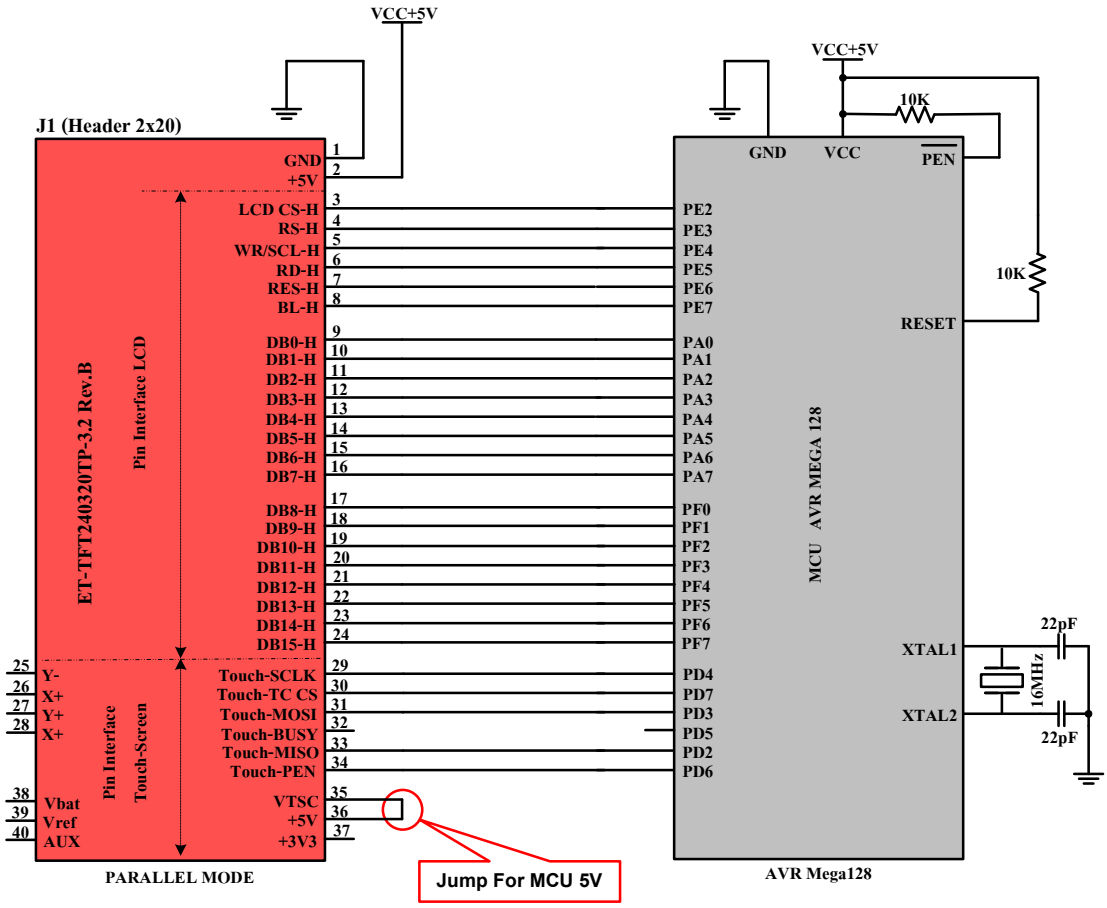
การต่อใช้งานกับ MCU 3.3V

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

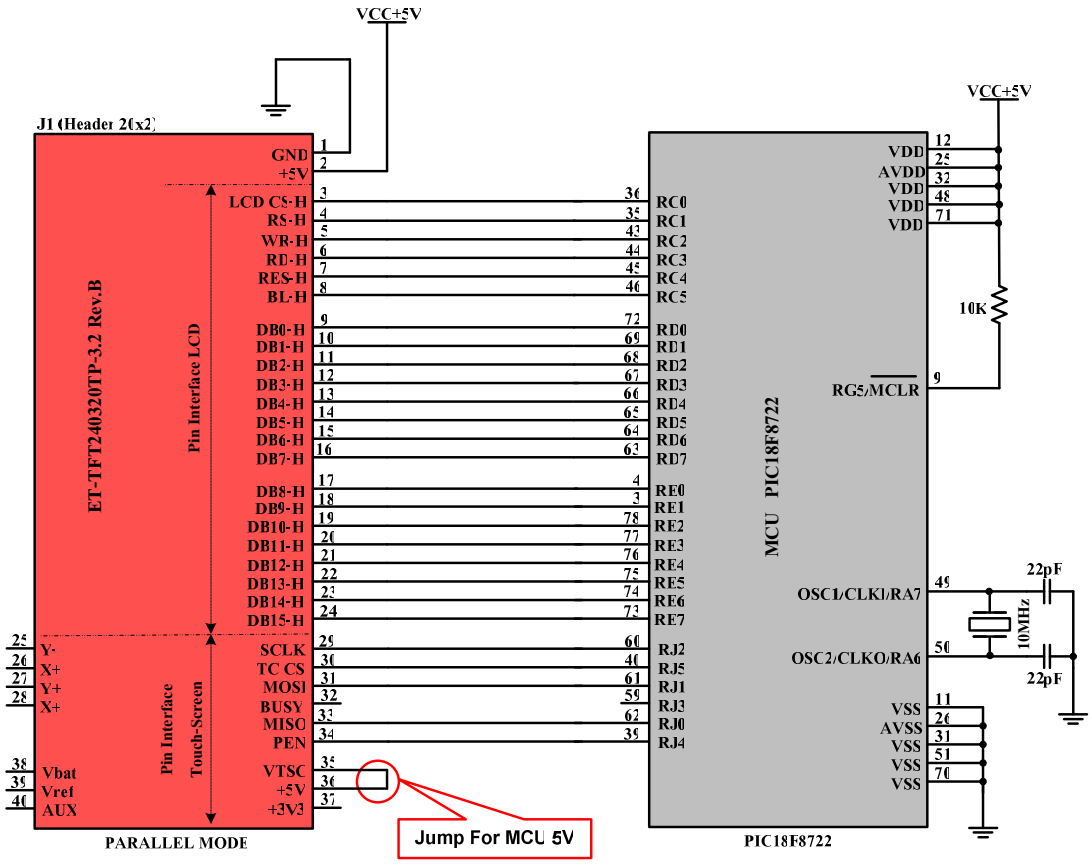


รูปที่3.3A ตัวอย่างการต่อบอร์ด ET-TFT240320TP-3.2 REV.B เข้ากับ MCU ARM7 #LPC 2138 (3.3V)

การต่อใช้งานกับ MCU 5 V



รูปที่3.3B ตัวอย่างการต่อบอร์ด ET-TFT240320TP-3.2 REV.B เข้ากับ MCU AVR #Mega 128 (5V)



รูปที่ 3.3C ตัวอย่างการต่อบอร์ด ET-TFT240320TP-3.2 REV.B เข้ากับ MCU PIC#18F8722 (5V)

4. หลักการ Control เบื้องต้นในการติดต่อกับ LCD และ Touch Screen

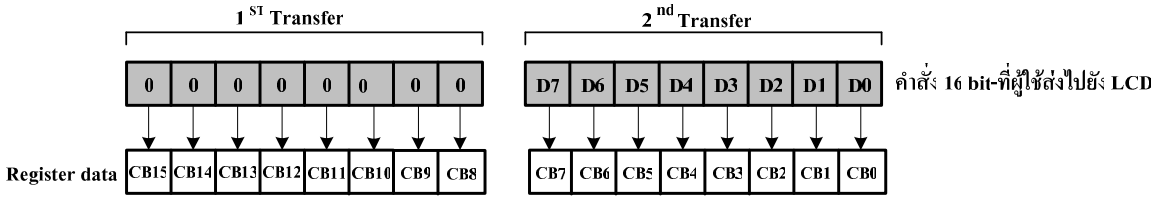
ในการเขียนโปรแกรมติดต่อกับบอร์ด ET-TFT240320TP-3.2 REV.B นั้น เพื่อให้ง่ายในการเขียนโปรแกรมให้ผู้ใช้มองแยกการควบคุมการทำงานออกเป็น 2 ส่วน คือ ส่วนที่เป็นจอ LCD ซึ่งในส่วนนี้จะเลือกการ Interface ได้ 3 แบบคือ Parallel 8-bit , 16-bit และแบบ SPI ส่วนที่เป็น Touch Screen จะใช้การ Interface แบบ SPI โดยหลักการเขียนโปรแกรมควบคุมการทำงาน ในที่นี้จะขอใช้ตัวอย่างที่ทางอีทีทีให้มาในแผ่น CD เป็นตัวอ้างอิง ซึ่งในแต่ละตัวอย่างที่ให้มาใน CD นั้นจะใช้หลักการเบื้องต้นในการติดต่อกับบอร์ดเหมือนๆกันหมด จะต่างกันเพียงรูปแบบการแสดงผลออกหน้าจอเท่านั้น ซึ่งสามารถสรุปหลักการทั้ง 2 ส่วนได้ดังนี้

4.1) การ Interface Control LCD ในส่วนของคำสั่งที่ใช้ควบคุมตัว LCD นั้น ให้ผู้ใช้ดูรายละเอียดการใช้งานได้จาก data Sheet “Driver_SPFD5408A.pdf ” ที่ให้มาใน CD ส่วนหลักการส่งคำสั่ง หรือ ส่ง Data ไปยัง LCD นั้นจะแยกออกเป็น 3 แบบ ตามการ Interface ซึ่งจะกล่าวถึงในหัวข้อย่อยต่อไป

ก่อนอื่นต้องทำความเข้าใจก่อนว่า ไม่ว่าจะ Interface ใน Mode ใดก็ตาม ในการส่ง Data ไปยัง LCD จะมีการส่ง Data ออกเป็น 2 ชุด คือ ในชุดแรกจะเป็นชุดของ Register ที่เราจะเข้าถึง ซึ่งก็คือค่าตำแหน่ง Address ของ Register Index ของคำสั่งนั้นๆ ซึ่งจะมีขนาด 8 bit เช่น คำสั่ง Write Data to GRAM(R22h) ค่าตำแหน่ง Address ของ Register Index ของคำสั่งนี้ก็คือ 0x22 แต่เวลาส่งเราจะต้องส่ง 16-bit ดังนั้นค่าที่ต้องส่งออกไปจริงคือ 0022H เป็นต้น ในชุดที่2 จะเป็นชุดของ Data ของคำสั่งนั้นๆ ซึ่งจะมีขนาด 16 bit เช่น เมื่อผู้ใช้ส่งคำสั่ง 0x22h ออกไปแล้วข้อมูลชุดที่ 2 ที่จะต้องส่งตามออกไปก็คือ Data สีหรือค่าอื่นๆของคำสั่งนั้นๆ สมมุติต้องการให้ปรากฏที่หน้าจอ LCD เป็นสีขาว 1 จุด ก็จะต้องส่ง data ออกไปคือ 0xFFFF เป็นต้น เมื่อพอเข้าใจหลักการส่งคำสั่งเบื้องต้นแล้ว ต่อไปเราจะมาพูดถึงการจัดเรียงข้อมูลภายในของ LCD ส่วนที่เป็นคำสั่ง และ Data ที่รับเข้ามาว่ามีการจัดเรียงอย่างไร รวมถึง Timing Diagram ในการ Read-Write Data จาก MCU ไปยัง LCD ของการ Interface ทั้ง 3 Mode ดังนี้

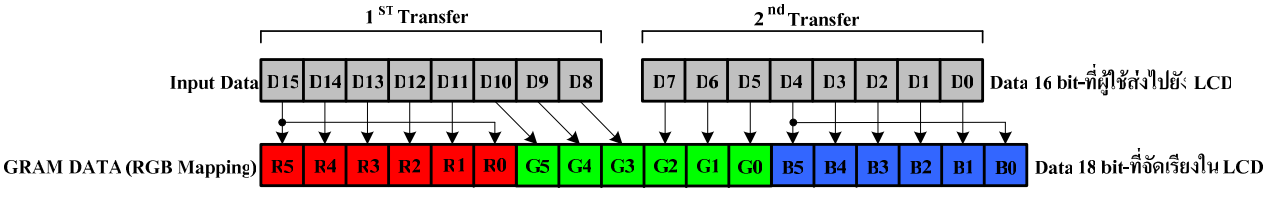
4.1.1) Interface SPI-MODE

สำหรับในโหมดนี้ LCD จะมีการจัดเรียงในส่วนของคำสั่ง และ Data ภายในใหม่ ตามที่ผู้ใช้ส่งเข้ามาดังรูป



รูปที่4.1.1A แสดงการจัดเรียงบิตของ คำสั่งที่รับเข้ามา

จากรูป 4.1.1A ใน 1st Transfer Byte High(D15…D8) เราจะส่งค่า 00H ออกไปเนื่องจาก Address Register ของคำสั่งมีขนาดเพียง Byte เดียว ซึ่งจะถูกกำหนดที่ตำแหน่ง Byte Low(D7…D0)



รูปที่4.1.1B แสดงการจัดเรียง Data บิตสีขนาด 16 bit 65K Colors ที่รับเข้ามา

จากรูป 4.1.1B เราจะเรียง Data บิตสี จากบิตสูงไปบิตต่ำเป็น RGB ซึ่งจะสอดคล้องกับตัวอย่างของอีทีที แต่ผู้ใช้สามารถจะเปลี่ยนให้เรียงบิตสีเป็น BGR ได้ โดยใช้คำสั่ง Entry Mode (R03H) เพื่อทำการ Set การเรียงบิตสีใหม่ ซึ่งดูรายละเอียดเพิ่มได้ใน Data Sheet ของ SPFD5408A

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

ขั้นตอนการ Write คำสั่ง และ Data จาก MCU ไปควบคุม LCD ในแบบ SPI-MODE

สำหรับคุณสมบัติของ SPI-Mode นี้จะทำงานที่ขอบขาขึ้นของสัญญาณ Clock การส่ง Data จะส่งแบบ Serial ครั้งละ 8 bit(1Byte) โดยจะส่งบิต MSB (บิตสูงสุด) ออกไปเป็นบิตแรก ในการส่งคำสั่ง หรือ Data จะมีรูปแบบการส่ง ครั้งละ 3 Byte ดังนี้

การ Write คำสั่ง(Index Register) – ต้องส่งก่อนการ Write data

1. Byte Start =70h(ID=0),74h(ID=1) -----> **2. คำสั่ง Byte High** = 00h -----> **3. คำสั่ง Byte Low** = Reg. NO (00h-A4h)

การ Write Data - ต้องส่งหลังจาก Write คำสั่งออกไปแล้ว

1. Byte Start =72h(ID=0),76h(ID=1) -----> **2. Data Byte High** = data (D15..D8) -----> **3. Data Byte Low** = data (D7..D0)

Byte ที่1 : จะเป็น Start Byte ต้องส่งเป็น Byte แรกเสมอ เพื่อเป็นตัวกำหนดรูปแบบการ อ่าน-เขียน และแยกข้อมูลที่ส่งไปให้ LCD ว่าเป็น คำสั่ง หรือ เป็น Data ซึ่ง Start Byte นี้จะมีขนาด 8 บิต โดยมีการจัดเรียงบิตดังนี้

BIT	7	6	5	4	3	2	1	0
Byte	Device ID Code						Control	
Start	0	1	1	1	0	ID	RS	R/W

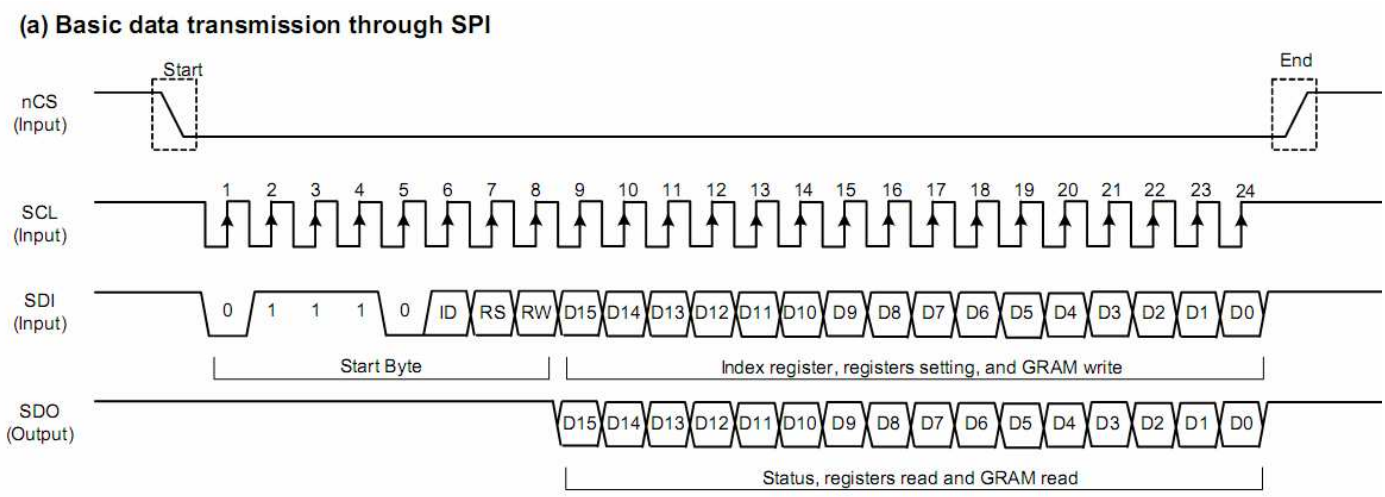
- Device ID Code : [Bit2..Bit7] โดย 6 บิตนี้ จะเป็นการกำหนด ID ของ LCD ซึ่งบิต3-บิต7 จะถูกกำหนดไว้ตายตัวแล้วเหลือไว้ให้ผู้ใช้กำหนดเองอีก 1 บิตคือ บิต2 ซึ่งสามารถกำหนดได้จาก DIP SW.2 ตัวที่ S1 ที่อยู่ด้านหลังบอร์ด เมื่อ S1 = ON จะได้ ID = 0 ถ้าอยู่ในตำแหน่ง OFF จะได้ ID=1
- Control : [Bit0..Bit1] 2 บิตนี้จะใช้กำหนดการอ่านเขียนข้อมูล และแยกข้อมูลที่ส่งไปยัง LCD ว่าเป็น คำสั่ง หรือ Data ซึ่งมีรูปแบบดังนี้

RS	R/W	Function
0	0	Write index register
0	1	Read a Status
1	0	Write Data
1	1	Read Data

Byte ที่2 : จะเป็น Byte High (8 bit High) ของคำสั่ง หรือ Data โดย ถ้าผู้ใช้กำหนดบิตใน Byteที่ 1 คือ RS=0 , R/W=0 ดังนั้น Byte ที่ 2 ที่จะส่งออกไปคือ Byte High ของคำสั่ง ซึ่งจะมีค่าเป็น 00H แต่ถ้าผู้ใช้กำหนดบิตใน Byteที่ 1 คือ RS=1 , R/W=0 ดังนั้น Byte ที่ 2 ที่จะส่งออกไปคือ Byte High ของ Data ซึ่งจะมีค่าเท่าไหร่ก็ขึ้นอยู่กับคำสั่งที่ผู้ใช้จะใช้งานว่ามีการกำหนด Data ไว้อย่างไร

Byte ที่3 : จะเป็น Byte Low (8 bit Low) ของคำสั่ง หรือ Data โดย ถ้าผู้ใช้กำหนดบิตใน Byteที่ 1 คือ RS=0 , R/W=0 ดังนั้น Byte ที่ 3 ที่จะส่งออกไปคือ Byte Low ของคำสั่ง ซึ่งจะเป็นค่าของ Register NO. (00h-A4h) ตาม data Sheet แต่ถ้าผู้ใช้กำหนดบิตใน Byteที่ 1 คือ RS=1 , R/W=0 ดังนั้น Byte ที่ 3 ที่จะส่งออกไปคือ Byte Low ของ Data ซึ่งจะมีค่าเท่าไหร่ก็ขึ้นอยู่กับ คำสั่งที่ผู้ใช้จะใช้งานว่ามีการกำหนด Data ไว้อย่างไร

Timing Diagram การ Read-Write คำสั่ง และ Data ใน SPI MODE



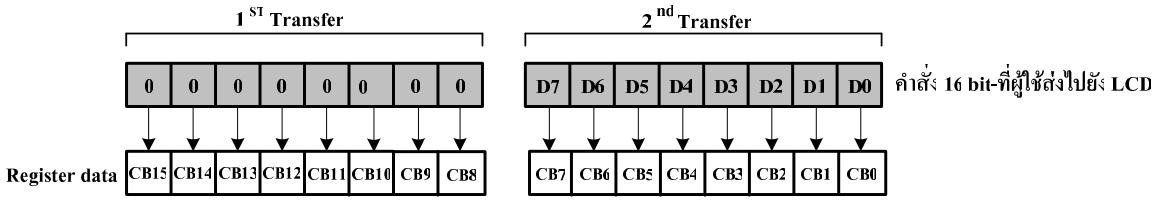
รูปที่4.1.1C แสดง Timing Diagram ในการ Read-Write คำสั่ง และ Data ให้ LCD ใน SPI-MODE

จาก Timing Diagram นี้จะแสดงการกำหนดจังหวะของสัญญาณการ อ่าน-เขียนข้อมูล ให้กับ LCD ใน SPI MODE โดยเริ่มจากในสภาวะปกติให้ CS เป็น High รอไว้ จากนั้นเริ่มสภาวะ Start โดยให้ CS เป็น Low จากนั้น เริ่ม Byte ที่1 ให้ MCU ส่ง Start Byte มายังขา SDI ของ LCD ตามจังหวะ Clock ขอบขาขึ้น ลูกที่1-8 ที่กำหนดโดย MCU ที่ขา SCL ต่อไปใน Byte ที่2 ก็ให้ MCU ส่งค่าคำสั่ง(Index register) หรือ Data ใน Byte High(D15-D8) ตามมา ในจังหวะ Clock ลูกที่9-16 และสุดท้าย Byte ที่ 3 ก็ให้ MCU ส่งค่าคำสั่ง(Index register) หรือ Data ใน Byte Low(D7-D0) ตามมาเป็น Byte สุดท้าย ในจังหวะ Clock ลูกที่ 17- 24 จากนั้นก็ Set ให้ขา CS เป็น High เป็นอันเสร็จสิ้นการ Write คำสั่ง หรือ Data ใน 3 Byte แรก เมื่อจะทำการส่ง Data อีกก็ให้กลับไปทำซ้ำตามขั้นตอนที่กล่าวไปข้างต้นใหม่

ให้สังเกตว่าในจังหวะ Clock ลูกที่9 ที่เราเริ่ม write data ใน Byte ที่ 2 (Bit-D15) ออกไปที่ขา SDI ตัว LCD ก็จะเริ่มส่ง Data (Bit-D15) ออกมาที่ขา SDO เช่นกัน ดังนั้นในจังหวะนี้ผู้ใช้สามารถเริ่มอ่านข้อมูล จาก LCD มาเก็บไว้ได้พร้อมๆกัน เพื่อนำไปใช้งานตามที่ผู้ใช้ต้องการ โดยข้อมูลที่อ่านได้นี้ ก็จะเป็นสถานะ Register ของ LCD หรือ Data แล้วแต่คำสั่งที่ผู้ใช้จะส่งไปอ่าน ว่าไปอ่าน Dataในตำแหน่ง Registerใดๆ

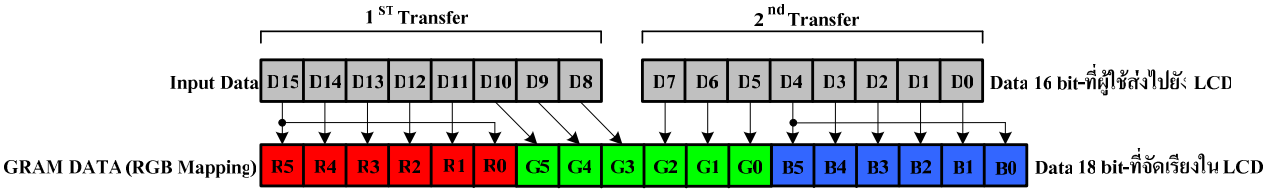
4.1.2) Interface Parallel 8-bit MODE

สำหรับในโหมดนี้ LCD จะมีการจัดเรียงในส่วนของคำสั่ง และ Data ภายในใหม่เหมือนกับโหมดอื่นๆ แต่การส่งคำสั่ง หรือ data จะส่งครั้งละ 8 บิต(1Byte) ต่อสัญญาณ Write(WR)เพียง1ลูก ตาม Timing Diagramรูปที่ 4.1.2C



รูปที่4.1.2A แสดงการจัดเรียงบิตของ คำสั่งที่รับเข้ามา

จากรูป 4.1.2A ใน 1 st Transfer Byte High(D15…D8) เราจะส่งค่า 00H ออกไปเนื่องจาก Address Register ของคำสั่งมีขนาดเพียง Byte เดียว ซึ่งจะถูกกำหนดที่ตำแหน่ง Byte Low(D7…D0)



รูปที่4.1.2B แสดงการจัดเรียง Data บิตสีขนาด 16 bit 65K Colors ที่รับเข้ามา

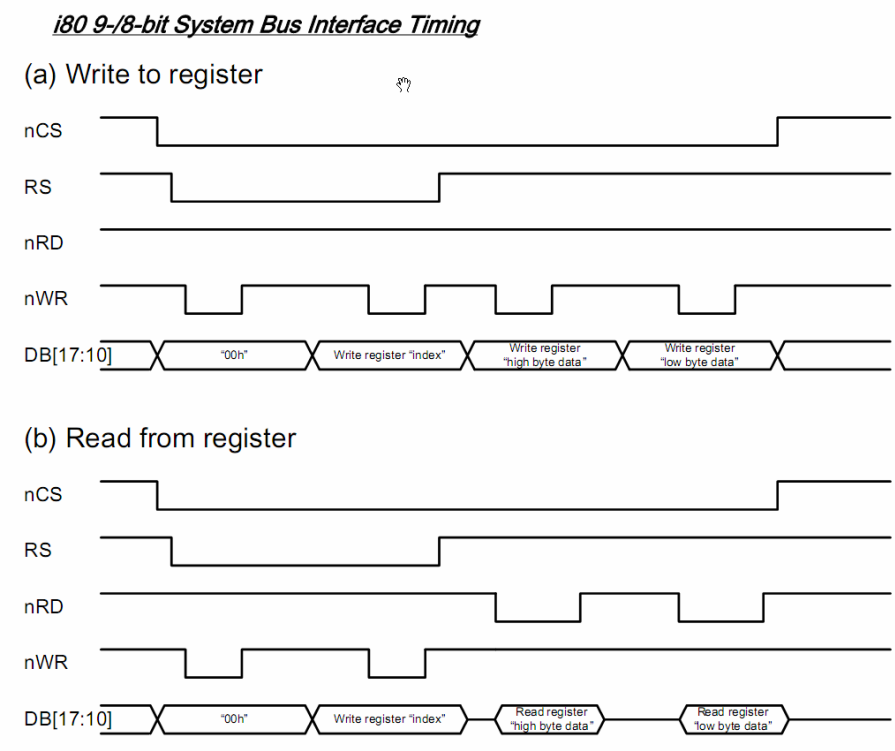
จากรูป 4.1.2B เราจะเรียง Data บิตสี จากบิตสูงไปบิตต่ำเป็น RGB ซึ่งจะสอดคล้องกับตัวอย่างของอีทีที แต่ผู้ใช้สามารถจะเปลี่ยนให้เรียงบิตสีเป็น BGR ได้ โดยใช้คำสั่ง Entry Mode (R03H) เพื่อทำการ Set การเรียงบิตสีใหม่ ซึ่งดูรายละเอียดเพิ่มได้ใน Data Sheet ของ SPFD5408A

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

ขั้นตอนการ Write คำสั่ง และ Data จาก MCU ไปควบคุม LCD ในแบบ Parallel 8-Bit MODE

สำหรับการส่ง คำสั่ง หรือ Data จาก MCU ไปยัง LCD ในโหมดนี้ เราจะต้องส่งบิตข้อมูลออกไปที่ขา DB8-DB15 ของบอร์ด LCD ของ ETT ตามวงจรข้างต้นเสมอ โดยจะส่งข้อมูลออกไปครั้งละ 1 Byte จำนวน 2 ครั้ง ต่อการส่งคำสั่ง หรือ Data 1 ครั้ง โดยแยกเป็น 2 Byte แรกที่เป็นคำสั่งต้องถูกส่งออกไปก่อน จากนั้นถึงจะส่ง อีก 2 Byte หลังที่เป็น data ของคำสั่งนั้นๆตามออกไป ซึ่งใน 2 Byte ที่เป็นคำสั่ง หรือ Data ที่ส่งออกไปนั้น Byte แรกที่ส่งจะต้องเป็น Byte High และ Byte ที่ 2 ถึงจะเป็น Byte Low โดยให้พิจารณาจังหวะการ อ่าน-เขียน คำสั่ง หรือ Data

ไปยัง LCD ได้จาก Timing Diagram ต่อไปนี้



รูปที่4.1.2C แสดง Timing Diagram ในการ Read-Write คำสั่ง และ Data ไปยัง LCD ใน Parallel 8-bit Mode

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

ก่อนอื่นเมื่อพิจารณาจาก Timing Diagram ในส่วนของการ Write จะเห็นว่าในการส่งข้อมูลออกไปยัง LCD ในแต่ละครั้งนั้นเราจะส่งข้อมูลออกไป 4 ชุดด้วยกันคือ ในชุดแรก(1Byte) จะเป็นชุดของ คำสั่ง Byte High ซึ่งก็คือค่า 00H ในชุดที่2 (1Byte) จะเป็นชุดของ คำสั่ง Byte Low ซึ่งก็คือค่าตำแหน่ง Address ของ Register Index (Register No.) ในชุดที่3(1Byte) จะเป็นชุดของ Data Byte High ส่วนในชุดที่4(1Byte) จะเป็นชุดของ Data Byte Low ของคำสั่งนั้นๆ จากนั้นเรามาเริ่มส่งคำสั่งตาม Timing Diagram ซึ่งสรุปเป็นขั้นตอนเพื่อนำไปใช้งานจริงได้ดังนี้

1) กำหนดขา RD,CS ให้เป็น 1 ไว้

2) กำหนดขา CS ให้เป็น 0 เพื่อ Enable LCD ให้รับข้อมูล

3) ส่งคำสั่ง Byte High เป็น Byte แรก ซึ่งจะมีค่าเป็น 00h เสมอ ออกมารอไว้ที่ขา DB8-DB15(อ้างอิงขาตามบอร์ดจริง)

4) กำหนดขา RS ให้เป็น 0 เพื่อกำหนดว่าข้อมูลที่ส่งไปคือ คำสั่ง

5) กำหนดขา WR ให้เป็น 0 เพื่อเริ่มต้นการ Write คำสั่ง Byte แรก

6) กำหนดขา WR ให้เป็น 1 คำสั่ง Byte แรกก็จะถูกส่งเรียบร้อย

7) ส่งคำสั่ง Byte Low เป็น Byte ที่สอง ซึ่งก็คือค่าของ Register Index ที่ผู้ใช้ต้องการ ออกมารอไว้ที่ขา DB8-DB15

8) กำหนดขา WR ให้เป็น 0 เพื่อเริ่มต้นการ Write คำสั่ง Byte ที่2

9) กำหนดขา WR ให้เป็น 1 คำสั่ง Byte ที่2ก็จะถูกส่งเรียบร้อย

10) กำหนดขา RS ให้เป็น 1 เพื่อจบการเขียนคำสั่ง และจะเป็นการกำหนดให้ข้อมูลที่ส่งต่อจากนี้คือ Data หลังจากส่งชุดคำสั่งไปแล้วต่อไปก็ตามด้วยการส่งชุด Data ของคำสั่งนั้นตามออกไปดังนี้

11) ให้ขา CS ยังคงเป็น 0 อยู่ ส่วนขา RS และ RD ก็ยังคงเป็น 1 ค้างไว้ เพื่อจะส่งในส่วนของ Data

12) ส่ง Data Byte High เป็น Byte ที่ 3 ออกมารอไว้ที่ขา DB8-DB15

13) กำหนดขา WR ให้เป็น 0 เพื่อเริ่มต้นการ Write Data Byte ที่ 3

14) กำหนดขา WR ให้เป็น 1 Data Byte ที่3 ก็จะถูกส่งเรียบร้อย

15) ส่ง Data Byte Low เป็น Byte ที่ 4 ออกมารอไว้ที่ขา DB8-DB15

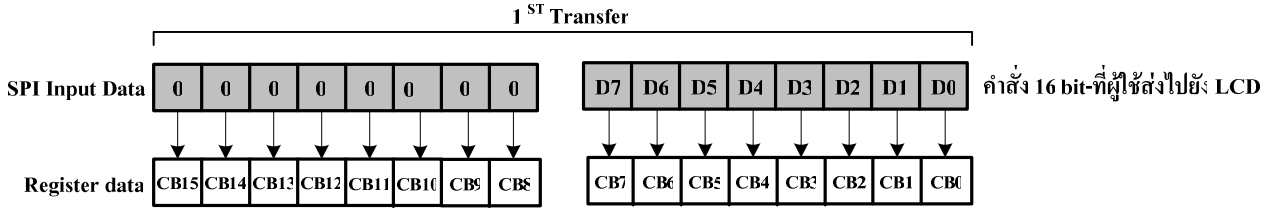
16) กำหนดขา WR ให้เป็น 0 เพื่อเริ่มต้นการ Write Data Byte ที่ 4

17) กำหนดขา WR ให้เป็น 1 Data Byte ที่4 ก็จะถูกส่งเรียบร้อย

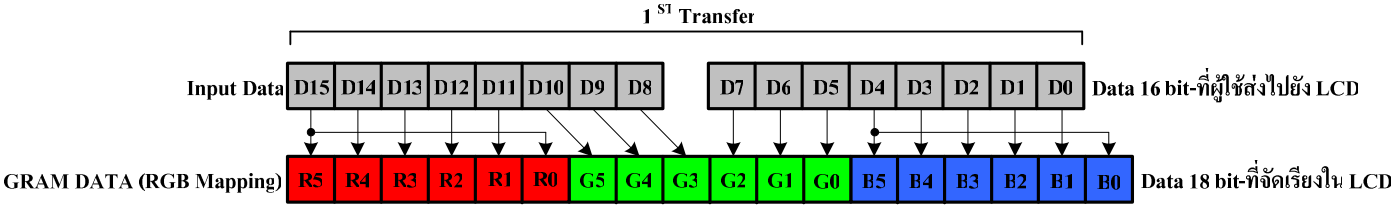
18) กำหนดขา CS ให้เป็น 1 เพื่อสิ้นสุดการส่ง คำสั่ง และ Data เมื่อจะส่งคำสั่งต่อไปก็ให้กลับไปเริ่มขั้นตอนที่1ใหม่

4.1.3 Interface Parallel 16-bit MODE

สำหรับในโหมดนี้ LCD จะมีการจัดเรียงในส่วนของคำสั่ง และ Data ภายในใหม่เหมือนกับ Mode อื่นๆ ที่กล่าวไปแล้วข้างต้น เพียงแต่จะต่างกันตรงที่ การส่งคำสั่ง หรือ data จะส่งครั้งละ 16 บิต(2Byte) ต่อสัญญาณ Write(WR) เพียง 1 ลูก ตาม Timing Diagram รูปที่ 4.1.3C



รูปที่4.1.3A แสดงการจัดเรียงบิตของ คำสั่งที่รับเข้ามา



รูปที่4.1.3B แสดงการจัดเรียง Data บิตสีขนาด 16 bit 65K Colors ที่รับเข้ามา

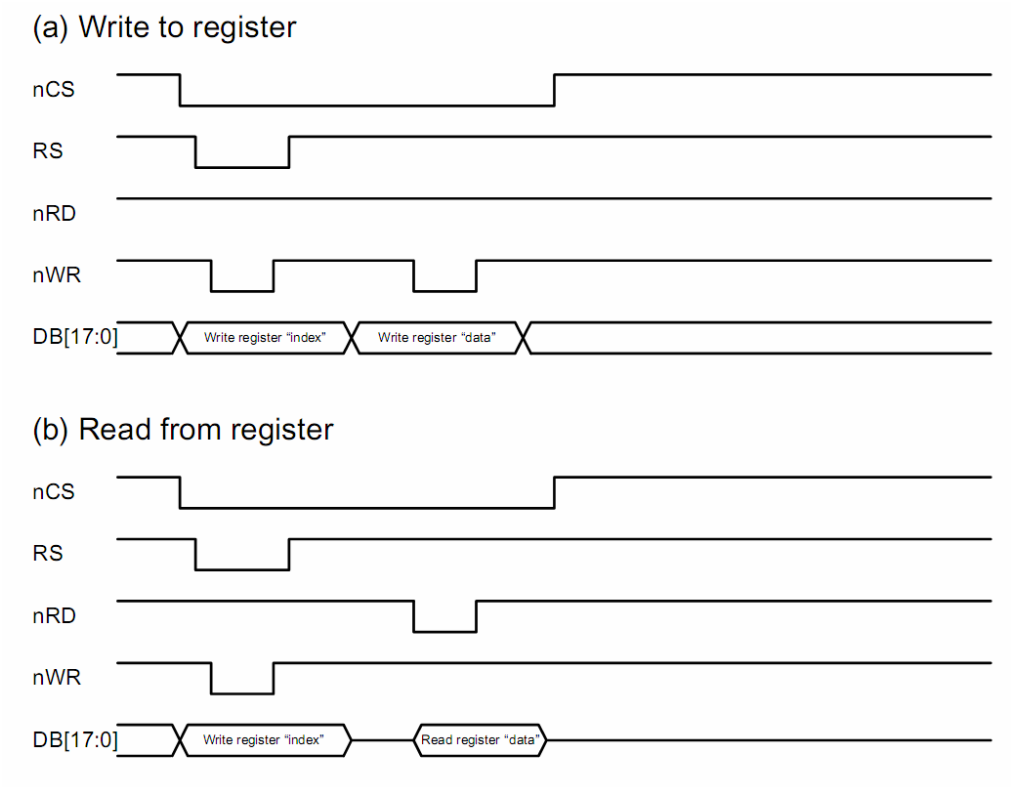
จากรูป 4.1.3B เราจะเรียง Data บิตสี จากบิตสูงไปบิตต่ำเป็น RGB ซึ่งจะสอดคล้องกับตัวอย่างของอีทีที แต่ผู้ใช้สามารถจะเปลี่ยนให้เรียงบิตสีเป็น BGR ได้ โดยใช้คำสั่ง Entry Mode (R03H) เพื่อทำการ Set การเรียงบิตสีใหม่ ซึ่งดูรายละเอียดเพิ่มได้ใน Data Sheet ของ SPFD5408A

เวลาที่ผู้ใช้จะส่งข้อมูล เพื่อแสดงจุดบนจอ LCD ผู้ใช้ก็จะต้องผสมสีเอาเองโดยอ้างอิงการเรียงบิตสีตามรูปด้านบนคือData D15-D11(5bit) จะเป็นส่วนของสีแดง , Data D10-D5(6bit) จะเป็นส่วนของสีเขียว , Data D4-D0(5bit)

จะเป็นส่วนของสีน้ำเงิน ซึ่งเมื่อผู้ใช้ส่ง Data เข้าไปยัง LCD แล้ว Data ก็จะถูกจัดเรียงข้อมูลใหม่เป็น 18 บิต อัตโนมัติตามรูป โดยความสว่างของสีจะไล่จาก มืด ไป สว่าง ซึ่งจะเรียงจากบิตต่ำไปหาบิตสูง เช่น ต้องการสีแดงที่มีความสว่างมากสุดก็จะได้ Data = 0xF800 หรือ ต้องการสีเขียวที่มีความสว่างน้อยสุด Data = 0x0020 เป็นต้น ถ้าต้องการสีอื่นนอกเหนือจาก 3 สีหลัก ผู้ใช้ก็จะต้องกำหนด บิต data ที่อยู่ในช่วงของแต่ละสีให้เหมาะสมก็จะได้สีออกมาตามที่ผู้ใช้ต้องการ เช่น ถ้าจะให้เป็นสีขาว ก็จะได้ Data = 0xFFFF หรือ สีดำ Data = 0x0000 เป็นต้น

ขั้นตอนการ Write คำสั่ง และ Data จาก MCU ไปควบคุม LCD ในแบบ Parallel 16-Bit MODE

สำหรับการส่ง คำสั่ง หรือ Data จาก MCU ไปยัง LCD ในโหมดนี้ เราจะต้องส่งบิตข้อมูลออกไปที่ขา DB0-DB15 ของบอร์ด LCD ของ ETT ตามวงจรข้างต้น โดยจะส่งข้อมูลออกไปครั้งละ 2 Byte จำนวน 1 ครั้ง ต่อการส่งคำสั่ง หรือ Data 1 ครั้ง โดยแยกเป็น 2 Byte แรกที่เป็นคำสั่งต้องถูกส่งออกไปก่อน จากนั้นถึงจะส่ง อีก 2 Byte หลังที่เป็น data ของคำสั่งนั้นๆตามออกไป โดยให้พิจารณาจังหวะการ อ่าน-เขียน คำสั่ง หรือ Data ไปยัง LCD ได้จาก Timing Diagram ต่อไปนี้



รูปที่4.1.3C แสดง Timing Diagram ในการ Read-Write คำสั่ง และ Data ไปยัง LCD ใน Parallel 16-bit Mode

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

ก่อนอื่นเมื่อพิจารณาจาก Timing Diagram ในส่วนของการ Write จะเห็นว่าในการส่งข้อมูลออกไปยัง LCD ในแต่ละครั้งนั้นเราจะส่งข้อมูลออกไป 2 ชุดด้วยกันคือ ในชุดแรก(2Byte) จะเป็นชุดของ คำสั่ง ซึ่งก็คือค่า 00H+ค่าตำแหน่ง Address ของ Register Index (Register No.) ในชุดที่2 (2Byte) จะเป็นชุดของ Data ของคำสั่งนั้นๆ ซึ่งจะมี

ขนาด 16 bit จากนั้นเรามาเริ่มส่งคำสั่งตาม Timing Diagram ซึ่งสรุปเป็นขั้นตอนเพื่อนำไปใช้งานจริงได้ดังนี้

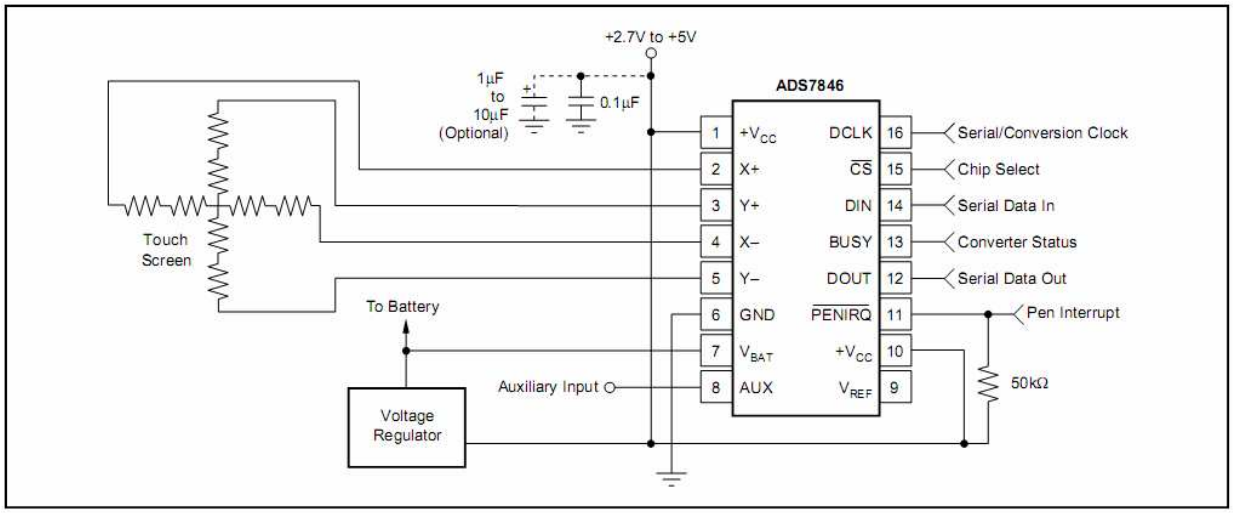
1) กำหนดขา RD,CS ให้เป็น 1 ไว้
2) กำหนดขา CS ให้เป็น 0 เพื่อ Enable LCD ให้รับข้อมูล
3) ส่งชุดคำสั่ง 16 bit ออกไปที่ Data Bus (DB0-DB15) โดย Data Bus 8 bit บน(DB8-DB15) กำหนดให้เป็นค่า 0x00
4) กำหนดขา RS ให้เป็น 0 เพื่อกำหนดว่าข้อมูลที่ส่งไปคือ คำสั่ง
5) กำหนดขา WR ให้เป็น 0 เพื่อเริ่มต้นการ Write คำสั่ง 2 Byte แรก
6) กำหนดขา WR ให้เป็น 1 คำสั่ง 2Byte แรกก็จะถูกส่งเรียบร้อย
7) กำหนดขา RS ให้เป็น 1 เพื่อจบการเขียนคำสั่ง และจะเป็นการกำหนดให้ข้อมูลที่ส่งต่อจากนี้คือ Data
หลังจากส่งชุดคำสั่งไปแล้วต่อไปก็ตามด้วยการส่งชุด 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(DB0-DB15)
10) กำหนดขา WR ให้เป็น 0 เพื่อเริ่มต้นการ Write Data 2 Byte หลัง
11) กำหนดขา WR ให้เป็น 1 Data 2Byte หลัง ก็จะถูกส่งเรียบร้อย
12) กำหนดขา CS ให้เป็น 1 เพื่อสิ้นสุดการส่ง คำสั่ง และ Data เมื่อจะส่งคำสั่งต่อไปก็ให้กลับไปเริ่มขั้นตอนที่1ใหม่

จากขั้นตอนที่กล่าวมานี้ เมื่อต้องการจะส่งคำสั่งอื่นๆต่ออีกก็ให้วนกลับไปเริ่มในขั้นตอนแรกใหม่ ซึ่งในการเขียนโปรแกรมนั้นผู้ใช้อาจเขียนฟังก์ชันให้รับค่าคำสั่งและค่าของ Data เข้ามาในฟังก์ชันพร้อมกันเลยแล้วทำการส่ง คำสั่ง กับ data ตาม Step ที่กล่าวข้างต้นก็ได้ หรือจะเขียนตามตัวอย่างของอีทีทีก็ได้ซึ่งจะแยกออกเป็น 2 ฟังก์ชันคือ ฟังก์สำหรับส่ง คำสั่ง และ ฟังก์ชันสำหรับส่ง Data เป็นต้น

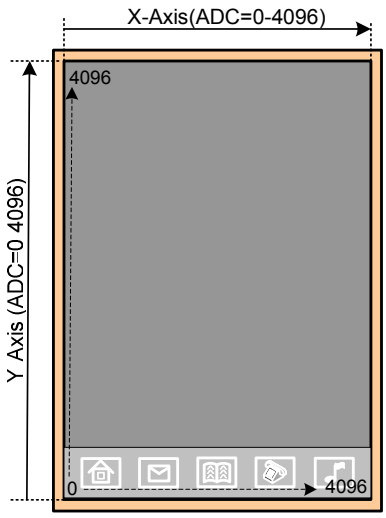
4.2) การ Interface Control Touch Screen ในส่วนของ Touch Screen นี้ จะแยกการ Control ออกมาจาก LCD ซึ่งในการ Control นั้นสามารถเลือกรูปแบบการ Interface ได้ 2 แบบ คือ Interface โดยใช้ขา Y-,Y+,X-,X+ ต่อเข้ากับขา ADC ของ MCU โดยตรงแล้วทำการเขียนโปรแกรมควบคุมการอ่านค่าเอาเอง ซึ่งจะทำให้เขียนโปรแกรมยาก ผู้ใช้จะต้องเข้าใจหลักการทำงานของตัว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

สำหรับการ Interface ที่จะแนะนำให้ใช้และจะสอดคล้องกับตัวอย่างที่ทางอีทีทีเขียนไว้ให้ด้วยซึ่งก็คือการ Interface ผ่าน Chip ADS7846 เมื่อผู้ใช้เลือกการ Interface แบบนี้ผู้ใช้จะต้องเลื่อน DIP SW1(S1-S4) ที่อยู่หลังบอร์ดไปที่ตำแหน่ง ON ทั้งหมดเพื่อเป็นการเชื่อมต่อขา 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 นี้มาใช้สำหรับคอยตรวจสอบว่ามีการสัมผัสจอ Touch Screen อยู่หรือไม่ เพื่อจะได้ไม่ต้องวนอ่านค่าตำแหน่งของ Touch Screen อยู่ตลอด เวลา จะอ่านเฉพาะเวลามีการสัมผัสจอเท่านั้นซึ่งจะทำให้โปรแกรมสมารถไปทำงานในส่วนอื่นๆได้



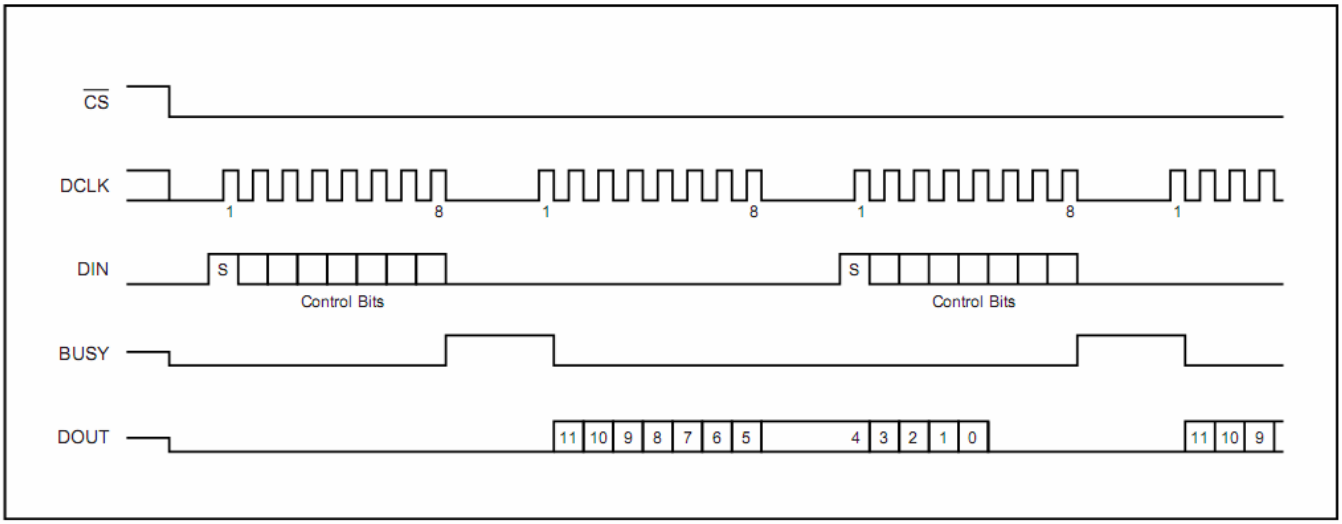
รูปที่4.2.1 แสดง วงจรการต่อในส่วนของ Touch Screen เข้ากับ ADS7846



รูปที่4.2.2 แสดงขอบเขตของ Touch Screen และค่า ADC ที่เกิดขึ้นตามแนวแกน X และ Y

ในรูปที่ 4.2.2 จะแสดงทิศทางแนวแกน X และ Y ของ Touch Screen และค่า ADC ที่จะอ่านได้ตามจุดต่างๆเมื่อมีการสัมผัสจอ ซึ่งโดยปกติค่า ADC ที่อ่านได้โดยผ่าน ADS7846 นั้น ค่าต่ำสุดจะอยู่ที่ประมาณ 600 (ไม่เป็น 0) ซึ่งเป็นค่า offset ของจอโดยจอแต่ละจอจะอ่านค่าเริ่มต้นนี้ออกมาไม่เท่ากัน ดังนั้นเวลาใช้งานจริงจึงจำเป็นที่จะต้องเขียนโปรแกรมไว้สำหรับ Calibrate จอก่อนเสมอ ซึ่งผู้ใช้สามารถ Copy ตัวอย่างของ ETT ไปใช้งานได้เลย โดยจะให้หลักการของเมตริกเข้ามาช่วยในการคำนวนหาค่าสัมประสิทธิ์ ที่ได้จากการ Calibrate ของผู้ใช้ โดยในตัวอย่างโปรแกรมจะให้ผู้ใช้ Touch จุด 3 จุด ถ้าผู้ใช้ Touch ได้ตรงกับตำแหน่ง Mark ก็จะทำให้เวลาใช้งานจริงหลังจาก Calibrate แล้วจะมีความแม่นยำสูง

- Timing Diagram การอ่านเขียนข้อมูลผ่าน ADS7846 หลังจากทราบหลักการทำงานเบื้องต้นในส่วนของ 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



รูปที่4.2.3 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 สร้างเอาเอง ไม่ใช้โมดูล SPI ภายในของ MCU เพื่อให้ผู้ใช้นำโปรแกรมไปแก้ไขได้ง่าย โดยในการส่งข้อมูลแบบ SPI นั้นมีหลักการอยู่ว่า เวลาเราส่งข้อมูลออกไปที่ขา MOSI(Dout) 1 bit แล้วตามด้วย Clock 1 ลูก ข้อมูลก็จะถูก Shift เข้าไปยัง Chip 1 บิต ในขณะเดียวกันตัว Chip ก็จะ Shift ข้อมูลออกมาที่ขา MISO(Din) 1 บิต เช่นกันซึ่งจะเป็นข้อมูลที่เราต้องอ่านเก็บไว้ ดังนั้นจะเห็นว่าในตัวอย่างที่ให้มา ฟังก์ชันก์ tcs_wr() จะทำหน้าที่เขียนและอ่านข้อมูล แบบ serial ขนาด 1 Byte(8bit) เมื่อเราสร้างฟังก์นี้ขึ้นมาได้แล้ว ต่อไปเราก็จะทำการส่ง Control byte และอ่านค่า ADC กลับมาเก็บไว้โดยผ่านฟังก์ชั้นนี้ โดยขั้นตอนในการอ่านค่า ADC จาก Touch Screen จะมีลำดับตามด้านล่าง

ในการอ่านค่า ADC จาก ADS7846 นั้นผู้ใช้จะต้องส่ง 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/$\overline{\text{DFR}}$	PD1	PD0

รูปที่4.2.4 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

<u>สรุปขั้นตอนการอ่านค่า ADC จาก ADS7846</u>

1) อ่านค่า Status จากขา PEN ของ ADS7846 ถ้าเป็น 0 (มีการสัมผัสจอ)ให้เริ่มกระบวนการอ่านค่าในขั้นตอนที่2 ต่อไป ถ้าอ่านได้ 1 (ยังไม่มีการสัมผัสจอ) ให้วนอ่านซ้ำ

2) กำหนดขา DCLK, CS, DOUT ให้เป็น 0

3) ส่ง Control Byte 0xD0 ไปยังขา DIN (MOSI) ของ ADS7846 เพื่อระบุการอ่านค่า ADC ในตำแหน่งแกน X ความละเอียด12 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 ออกมาบิตแรกจะเป็นบิตที่11 โดยจะเริ่มต้นที่ขอบขาลงของ DCLK ลูกที่2 เมื่อ DCLK ถูกส่งครบ 8 ลูก ก็จะอ่าน Data ในแกน X ได้ 0x0ddddddd (d=data bit11-bit5)

5) ส่ง Control Byte 0x90 ไปยังขา DIN (MOSI) ของ ADS7846 เพื่อระบุการอ่านค่า ADC ในตำแหน่งแกน Y ความละเอียด12 บิต ในขณะที่ส่ง Control Byte นี้ออกไป ค่า data ADC ในแกน X อีก 5 บิตสุดท้ายก็จะถูกส่งออกมา โดยเริ่มจาก bit4 ถึง bit0 โดยเรียง Data ดังนี้ 0xddddd000 (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 ออกมาบิตแรกจะเป็นบิตที่11 โดยจะเริ่มต้นที่ขอบขาลงของ DCLK ลูกที่2 เมื่อ DCLK ถูกส่งครบ 8 ลูก ก็จะอ่าน Data ในแกน Y ได้ 0x0ddddddd (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 0xddddd000 (d=data bit4-bit0)

8) เมื่ออ่านค่า ADC ทั้ง 2 แกนเรียบร้อยแล้วก็ให้ Set ขา CS เป็น 1 เพื่อจบการอ่านค่า จาก ADS7846

9) เมื่อจะอ่านค่าใหม่ก็ให้วนไปเริ่มต้นที่ขั้นตอนที่ 1 ใหม่

10) หลังจากได้ ค่าADC ในแต่ละแกนมาแล้วจะต้องนำค่าที่ได้ของแต่ละแกนมาเรียงใหม่โดยตัวแปรที่ใช้เก็บค่า ADC ที่อ่านควรจะเป็นตัวแปรแบบ 16 บิต ซึ่งจาก data ADC ที่อ่านได้ 7 bit แรกเมื่อเก็บในตัวแปร 16 บิตจะได้เป็น 0x000000000ddddddd และ 5 bit หลังเมื่อเก็บในตัวแปร16บิตจะได้เป็น 0x00000000ddddd000 จากนั้นให้ Shift data ที่อ่านได้ 7bit แรกไปทางซ้าย 5 bit และ shift data ที่อ่านได้ 5 bit หลัง ไปทางขวา 3 bit แล้วนำข้อมูลทั้ง 2 ชุดมา OR(|) กันก็จะได้ data ADC ของแกนที่อ่านได้ 12 บิต ดังนี้ 0x000ddddddddddddd นี่คือค่าที่เราจะนำไปใช้งานต่อไป

จากหลักการ Control LCD และ Touch Screen ทั้งหมดที่กล่าวมานี้จะเป็นกระบวนการทำงานในภาพรวมของการใช้งานบอร์ด ET-TFT240320TP-3.2 REV.B เมื่อผู้ใช้พอทราบหลักการ Control เบื้องต้นแล้ว เวลาจะเขียนโปรแกรมจริงๆนั้น ผู้ใช้สามารถคัดลอกในส่วนของฟังก์ชันในตัวอย่างที่ทางอีทีทีทำไว้ไปใช้ได้เลย โดยดูหลักการนำฟังก์ชันไปใช้งานได้ในหัวข้อ “อธิบายตัวอย่างโปรแกรม”

5. การใช้โปรแกรม Embedtools 3.31 Convert ไฟล์รูปภาพไปเป็น hex code

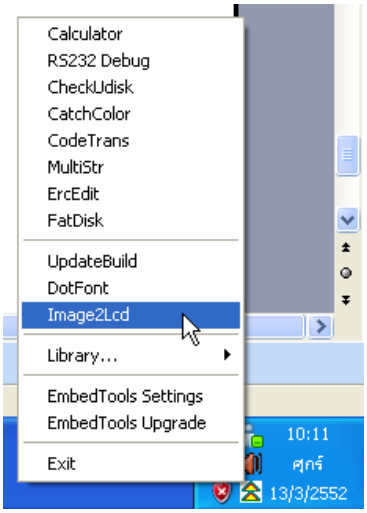
สำหรับในหัวข้อนี้จะกล่าวถึงวิธีการนำรูปภาพมาแปลงเป็น hex code เพื่อนำ hex code ที่ได้ส่งไปยัง LCD หน้าจอ LCD ก็จะแสดงรูปภาพออกมาให้เราตามต้องการ โดยเราจะใช้โปรแกรม “Embedtools 3.31” ซึ่งขั้นตอนการแปลงที่จะกล่าวดังต่อไปนี้ hex code ที่ได้ออกมาจากการแปลง จะรองรับกับตัวอย่างของอีทีที โดยจะใช้ฟังก์ชัน “plot_picture()”(อยู่ในตัวอย่าง Ex3_Touch_Button) เป็นตัวส่ง hex code จาก MCU ไปยังตัว LCD

ในกรณีที่ผู้ใช้ไม่ได้ initial lcd ตามตัวอย่างของอีทีที และไม่ได้ Set โปรแกรม Embedtools ในการ Convert ภาพตามขั้นตอนที่กล่าวข้างล่าง ผู้ใช้อาจจะไม่สามารถใช้ Function “plot_picture()” ได้ เพราะจะทำให้ทิศทางการส่งข้อมูลนั้นผิดพลาดไปได้ ผู้ใช้จะต้องเขียนโปรแกรมการ plot รูปเอาใหม่ให้สอดคล้องกับการ Set ค่า ต่างๆที่ผู้ใช้ได้กับหนดเอง

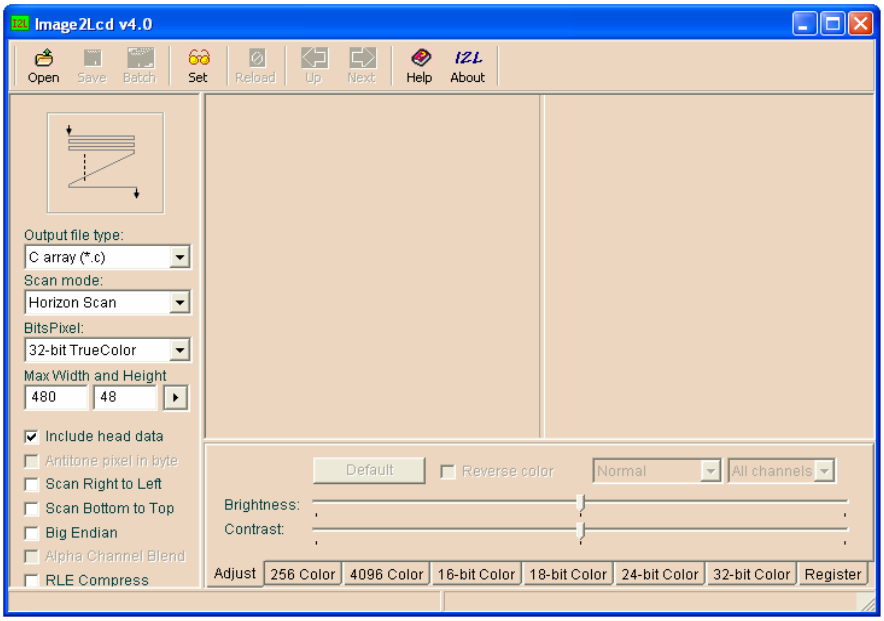
ขั้นตอนการใช้งานโปรแกรม Embedtools Convert File รูปเป็น Hex Code

1.) ทำการติดตั้งโปรแกรม Embedtools.exe ลงในเครื่อง (อยู่ใน Folder Embedtools3.31)

2.) หลังจากติดตั้งโปรแกรมเรียบร้อยแล้วจะมี icon ( ) ปรากฏอยู่ที่ Taskbar ดังรูปที่ 5.1จากนั้นให้คลิกขวาที่ icon จะมีแท็บหัวข้อขึ้นมา คลิกซ้ายเลือกหัวข้อ Image2LCD เพื่อ Run โปรแกรม Embedtools ขึ้นมา จะได้หน้าต่างดังรูปที่5.2

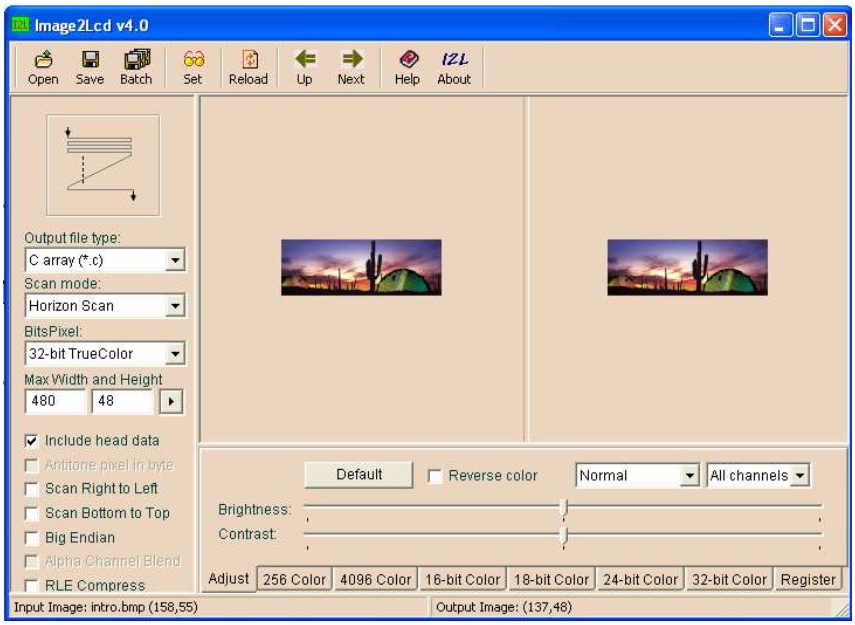


รูปที่5.1



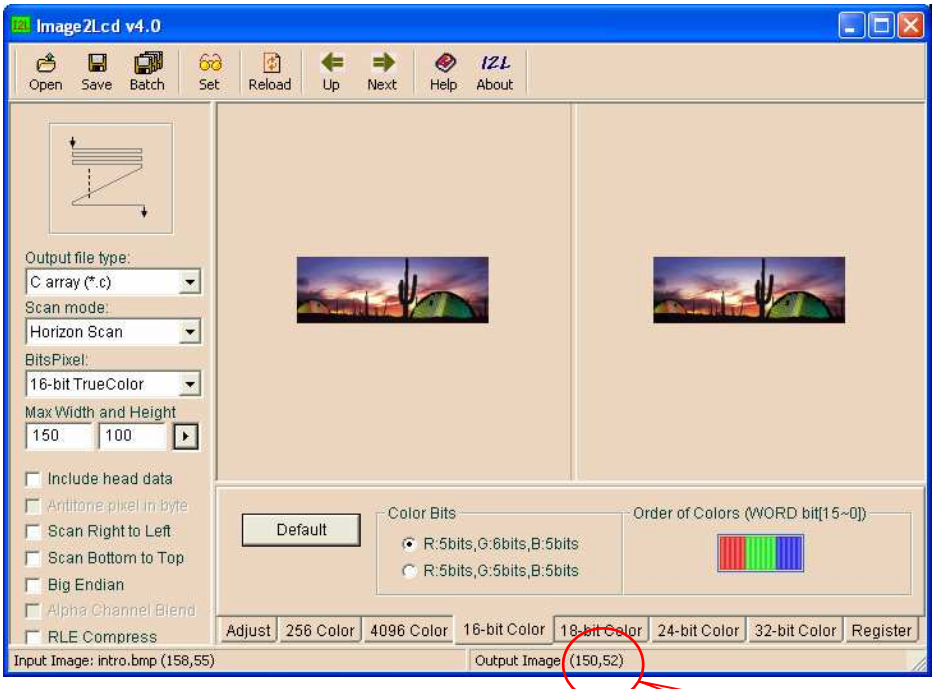
รูปที่5.2

3) คลิกที่ icon Open (Open) ด้านบนของโปรกรมเพื่อเลือกไฟล์รูปที่จะทำการ Convert โดยไฟล์รูปควรจะเป็นไฟล์นามสกุล jpg หรือ bmp เมื่อเปิดรูปขึ้นมาจะได้หน้าต่างดังรูปที่5.3



รูปที่5.3

4) หลังจากโหลดรูปเข้ามาในโปรแกรมแล้วให้ทำการ Set ค่าดังนี้ (ดูรูปที่5.4ประกอบ)



รูปที่5.4

- 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 เวลานำข้อมูลไปใช้จะต้องส่ง data ไปยัง LCD ตามทิศทางที่ สแกนด้วย
- Bits Pixel : *16-bit TrueColor* = กำหนดความละเอียดของบิตสี (ในโปรแกรมจะมีค่านี้เหมือนกัน 2 ค่า ให้เลือกค่าบน)
- Max Width and Height : ความกว้าง,ความสูง = กำหนดความกว้างและความสูงให้กับขนาดรูป เมื่อกำหนดแล้วให้กดที่ปุ่ม (▶) จะเห็นผลการเปลี่ยนแปลงของรูป ขนาดของรูปจะมีหน่วยเป็น Pixel ซึ่งจะไม่

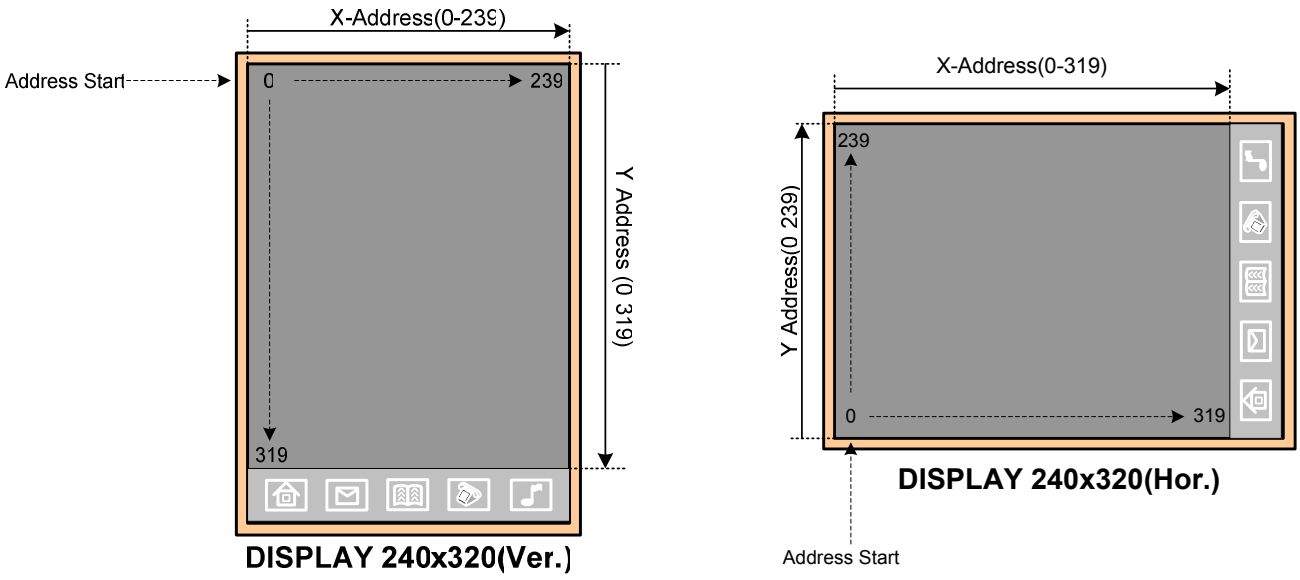
ใช่ขนาดที่แท้จริงของรูป ดังนั้นเวลาผู้ใช้เรียกใช้ฟังก์ชัน “plot_picture()” ตัวอย่างของอีทีที จะต้องใช้ค่าความกว้างและความสูงของรูปที่แท้จริงโดยให้ดูที่ช่อง Output Image ด้านล่าง ซึ่งจะเป็นขนาด Pixel ที่แท้จริงของรูป

- 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 (Save) เพื่อทำการ save file hex code เก็บไว้ เมื่อจะใช้งานก็ให้ใช้ note pad เปิด file ขึ้นมา จากนั้น ก็ Copy hex code ไปวางไว้ใน editor ที่ใช้เขียนโปรแกรม

6. การทำงานตัวอย่างโปรแกรม

สำหรับตัวอย่างของอีทีทีที่ให้มานั้นจะเขียนด้วยภาษาซี จะรองรับ MCU 3 ตระกูลได้แก่ AVR(Mega128) , PIC(18F8722),ARM7(LPC2138) ซึ่งในตัวอย่างจะมีทั้งการ Control LCD ในแนวตั้ง และแนวนอน ขึ้นอยู่กับผู้ใช้ว่าต้องการให้แสดงผลในแนวใด ก็ให้ดูตัวอย่างในแนวนั้นเป็นหลัก โดยตัวอย่างของ MCU แต่ละตระกูลนั้นจะเป็นตัวอย่างเหมือนๆกัน ซึ่งในตัวอย่างทั้งหมดนี้เราจะอ้างตำแหน่งแอดเดรสเริ่มต้นทางแกน X,Y ของจอตามทิศทางดังรูปด้านล่าง



รูปที่ 6.1 แสดงการอ้างแอดเดรสบนหน้าจอ LCD ในแนวตั้งและแนวนอน

จากรูปด้านบนเมื่อผู้ใช้นำฟังก์ชันก์ตัวอย่างที่ให้มาไปใช้งาน เวลากำหนดตำแหน่งแอดเดรส X,Y สำหรับเริ่ม Plot รูปหรือตัวอักษร ผู้ใช้ก็จะต้องมองตำแหน่งแอดเดรสอ้างอิงตามรูปด้านบน

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 หรือเขียนตัวหนังสือลงบนหน้าจอได้หน้าจอก็จะแสดงลายเส้นตามที่ผู้ใช้วาดหรือเขียน และสามารถ Touch ที่ Icon ตัวโน๊ต ( ) เพื่อเปลี่ยนสีเส้นที่จะวาด และ Touch ที่ Icon Home ( ) เพื่อ Clear Screen

- Ex3_Touch_Button ในตัวอย่างนี้ เมื่อ Run โปรแกรม ก็จะให้ผู้ใช้ Calibrate Touch Screen เช่นเดิม จากนั้นก็จะแสดงปุ่มสำหรับให้ผู้ใช้ Touch เมื่อผู้ใช้ Touch ที่ปุ่มใดๆบนหน้าจอ ในช่องหน้าต่างที่ว่างอยู่ก็จะแสดง รูปต่างๆให้เห็นตามปุ่มที่กด

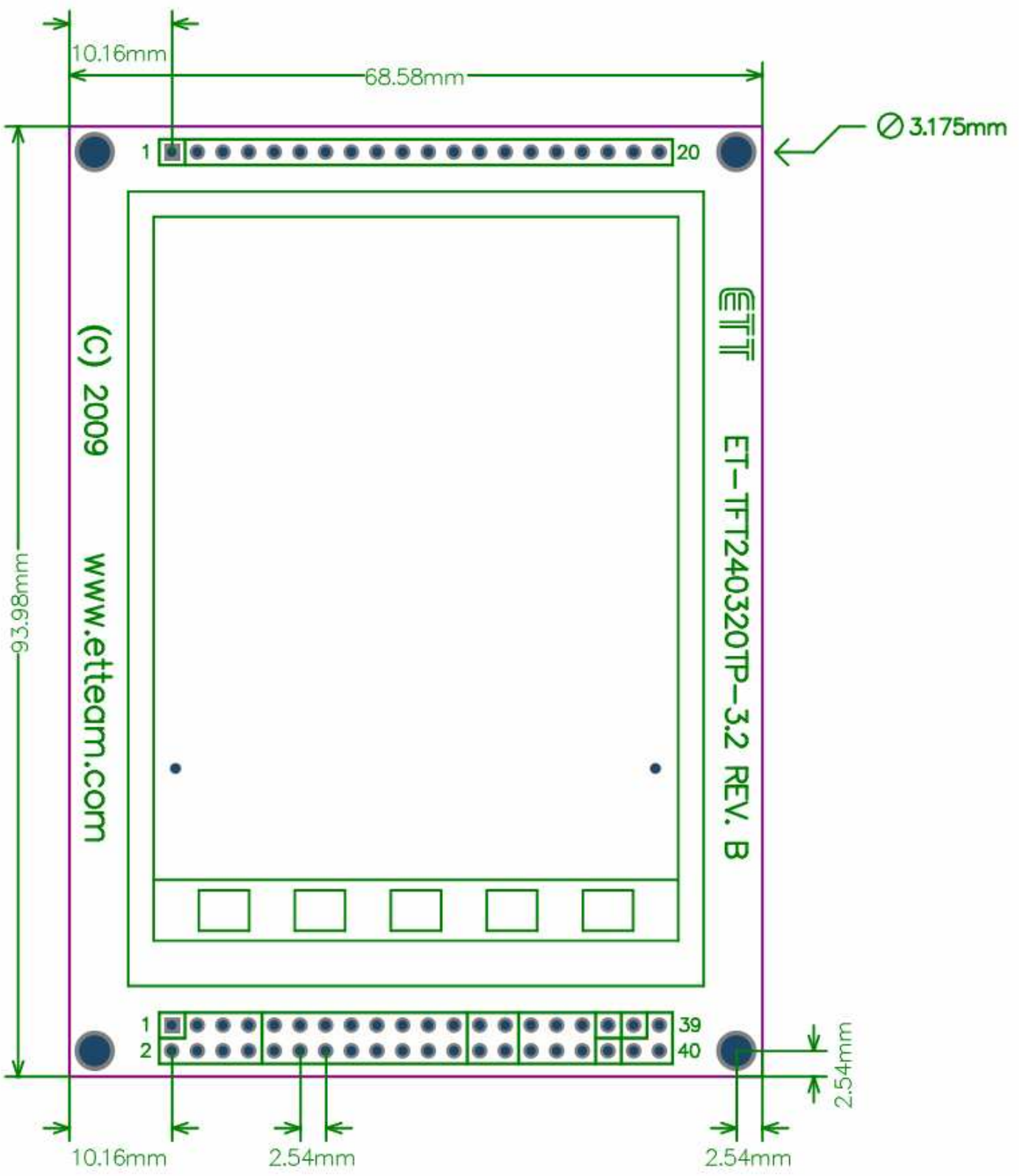
สำหรับในส่วนของการ Calibrate Touch Screen นั้น จากตัวอย่างจะเห็นว่าทุกครั้งที่เรา Reset MCU เราจะต้องทำการ Calibrate ใหม่เสมอ ซึ่งเวลาในการนำไปใช้งานจริงนั้นเราอาจจะต้องการ Calibrate เพียงครั้งเดียวเท่านั้น ซึ่งมีวิธีแก้โดย

<u>*วิธีที่1*</u> 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 E2Prompt ด้วยเพื่อใช้เป็น Flag Status สำหรับใช้ตรวจสอบว่ามีการ Calibrate ไปแล้วหรือยัง

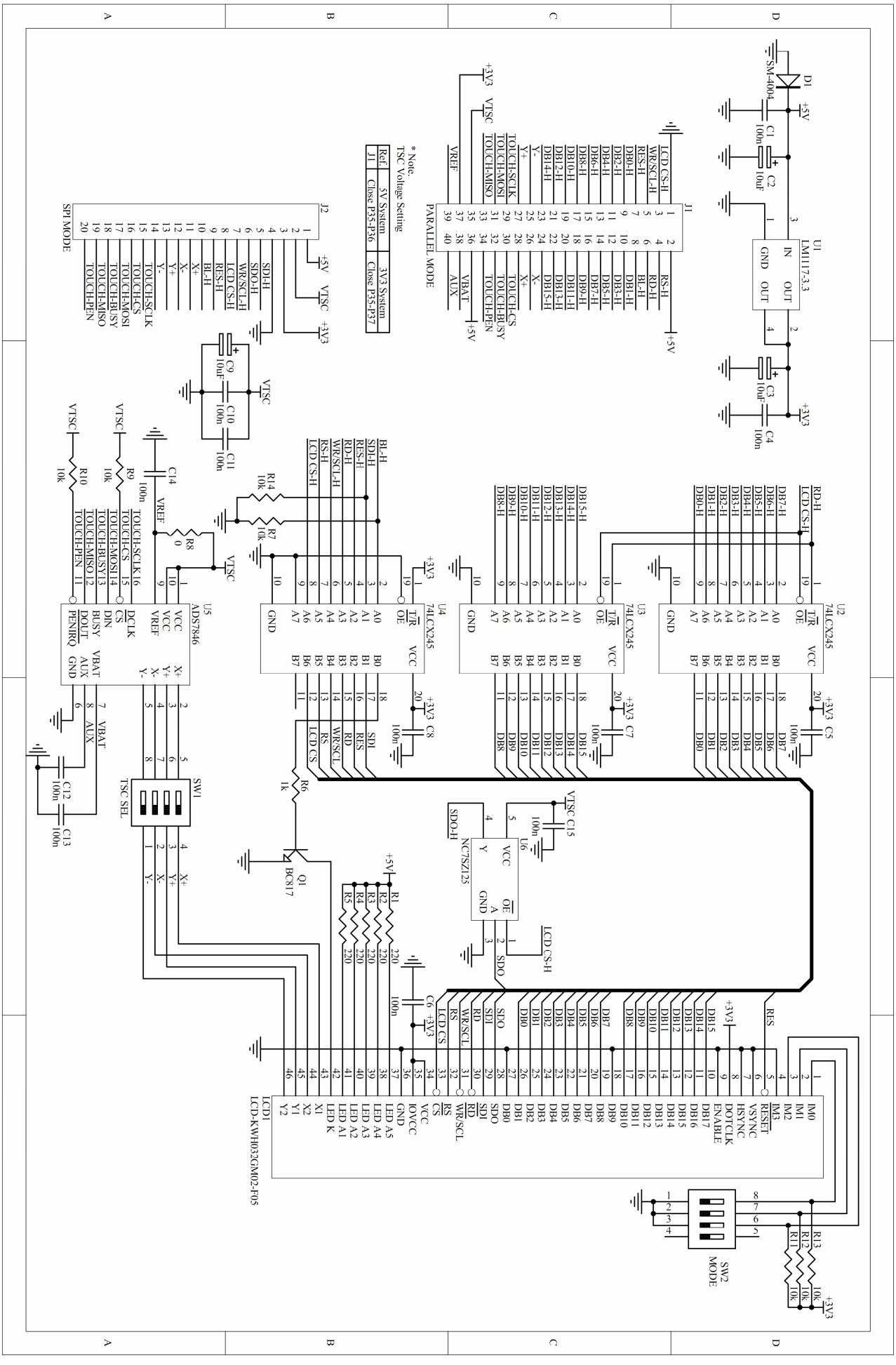
จากนั้นในส่วนของ main โปรแกรม ก่อนที่จะทำการเรียกใช้ฟังก์ชัน “touch_calibrate()” ให้ผู้ใช้อ่านค่า Status Flag จาก E2Prompt มาตรวจสอบก่อนว่าเป็นค่าเดียวกับค่าที่ผู้ใช้ได้ write เก็บไว้หรือไม่ ถ้าใช่ก็ไม่ต้องเรียกใช้งานฟังก์ชัน “touch_calibrate()” อีก ให้ไปอ่านค่าของตัวแปร divider,An,Bn,Cn,Dn,En,Fn ที่ได้เก็บไว้ใน E2Prompt ออกมา แล้วนำมาแทนค่าให้กับตัวแปร divider,An,Bn,Cn,Dn,En,Fn เหล่านี้เช่นเดิม แล้วก็ไปทำงานในส่วนอื่นของโปรแกรมต่อได้ เท่านี้ผู้ใช้ก็จะไม่ต้องเสียเวลาในการ Calibrate ทุกครั้งเมื่อจะใช้งาน LCD

<u>*วิธีที่2*</u> วิธีนี้ไม่ต้องใช้ E2Promt ทำได้โดย ในฟังก์ชันก์ “touch_calibrate()” ต่อจากบรรทัดการเรียกใช้ Function “set_matrix()” ให้ผู้ใช้เพิ่มคำสั่ง printf()เข้าไป(ในตัวอย่างเขียนไว้ให้แล้วแต่ปิดการใช้งานไว้) เพื่อ print ค่าของตัวแปร divider,An,Bn,Cn,Dn,En,Fn ออกมาแสดงผลทาง RS232 โดยใช้โปรแกม Hyper Terminal รับค่าของตัวแปรที่ print มาแสดงให้ผู้ใช้เห็น จากนั้นให้ผู้ใช้ทำการจดค่าของตัวแปรแต่ละตัวที่แสดงเก็บไว้ แล้วนำค่าที่ได้ไปกำหนดให้กับตัวแปร divider,An,Bn,Cn,Dn,En,Fn ที่ประกาศไว้เหนื่อ main() เวลาจะใช้งานผู้ใช้ก็สามารถตัดฟังก์ชัน “touch_calibrate()” ทิ้งได้ไม่ต้องเรียกใช้อีก ซึ่งวิธีนี้เวลาใช้งานจอ LCD หลายๆจอ ผู้ใช้อาจจะต้องเขียนโปรแกรมการ Calibrate เพื่อหาค่า divider,An,Bn,Cn,Dn,En,Fn ไว้ต่างหาก เนื่องจากถ้าผู้ใช้เปลี่ยนจอใหม่ถึงจะเป็นจอแบบเดียวกันผู้ใช้ก็จะต้อง Calibrate หาค่าใหม่เสมอ ไม่สามารถใช้ค่าที่อ่านได้จากจอเก่ามาใช้กับจอใหม่ได้เพราะจะทำให้ค่าแอดเดรสที่อ่านได้จากการ Touch ไม่ตรงกับตำแหน่ง address ของจอ LCD จริงๆ

<u>ข้อควรจำ:</u> ในการ Calibrate นั้นจะต้อง Touch ให้ตรงกับตำแหน่งที่มาร์คไว้ให้มากที่สุดเพื่อความแม่นยำในการนำไปใช้งาน



รูปที่ 6.2 รูปขนาดของบอร์ด ET-TFT240320TP-3.2 Rev.B



รูปที่ 6.3 วงจร ET-TFT240320TP-3.2 Rev.B