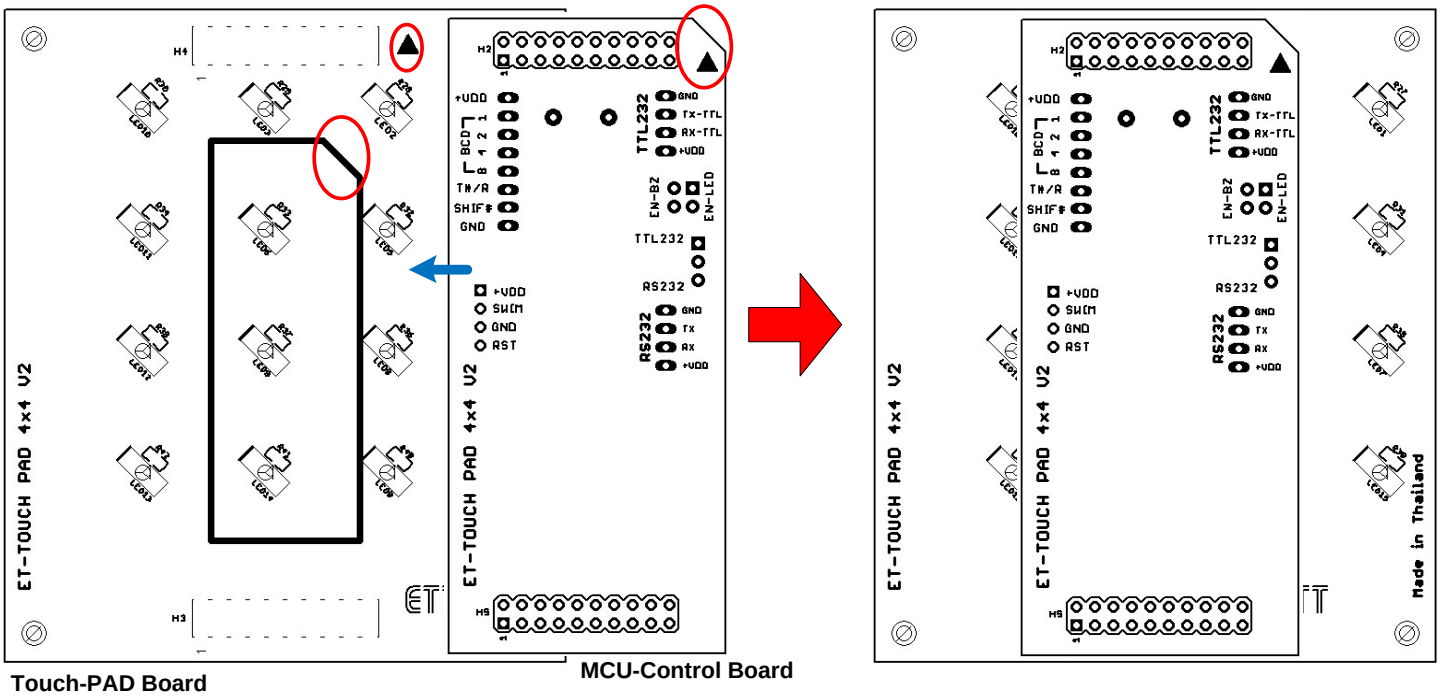
ET- TOUCH PAD 4x4 V2

1. คุณสมบัติของบอร์ด ET-TOUCH PAD 4x4 V2

- เป็น KEY Touch (สัมผัส) แบบ Capacitive sensing ขนาด 16 Key (4x4)
- ไฟเลี้ยงบอร์ด +3.3 VDC หรือ +5 VDC
- แสดงสถานะ การกด Key ของผู้ใช้ด้วย เสียง(Buzzer) และ LED ที่อยู่ในตำแหน่งของ Key นั้นๆ
- สามารถ ON/OFF เสียง (Buzzer) และกำหนดการทำงานของ LED Status Key ได้ด้วย Jumper โดยแยกอิสระกัน
- ET-TOUCH PAD 4x4 V2 จะแสดงสถานะเริ่มต้นทำงาน(Power-On) ด้วยเสียง Beep และ LED Status Key กระพริบเป็นจังหวะ
- ตัวบอร์ดสามารถแยกได้เป็น 2 ส่วนคือ ส่วน Touch PAD บอร์ด และส่วน MCU Control บอร์ด ซึ่งจะเชื่อมกันด้วย Connector ที่แข็งแรง
- OUTPUT สำหรับค่า Key Code ของ Key ที่ผู้ใช้กด จะให้ค่าออกมา 2 แบบคือ
  1) แบบ Binary Code (BCD8421) จะส่งค่าผ่านทาง Connector 8 PIN โดยมีขา สัญญาณ T#/R และ SHIF# เพื่อบอกสถานะ การกด หรือ ปล่อย Key
  2) แบบ ASCII Code จะส่งค่าผ่านทางขั้วต่อ RS232(TTLและLine Drive) ซึ่งจะ Fix ค่า Baud Rate ในการรับส่งไว้ที่ 9600 โดยจะส่ง ASCII ‘P’ หรือ ‘R’ นำหน้าค่า Key Code ออกมา เพื่อบอกสถานะ การ กด/ปล่อย Key
- แผ่นรอง Key Touch ถ้าเป็นพลาสติกใส หนาได้ประมาณ 1-3 มิลลิเมตร(ขึ้นอยู่กับค่าความชื้นในอากาศด้วย) ถ้าเป็นวัสดุอื่น ความหนาจะขึ้นอยู่กับคุณสมบัติความไวทางไฟฟ้าของวัสดุนั้นๆ
- มี Key พิเศษ 1 Key สามารถใช้เป็น Key ปกติ หรือ ใช้เป็น Key SHIF เพื่อกดร่วมกับ Key อื่นๆที่เหลือได้(กด2keyพร้อมกัน)
- สามารถควบคุมการทำงานของ Buzzer และ LED Status Key (เมื่อไม่มีการ Touch Key) ได้ด้วย ASCII Command ผ่านทาง RS232 โดยจะต้อง Set Jumper สำหรับ Enable Buzzer และ LED Status Key ไว้ด้วย

2. ลักษณะและโครงสร้างของบอร์ด ET-TOUCH PAD 4x4 V2

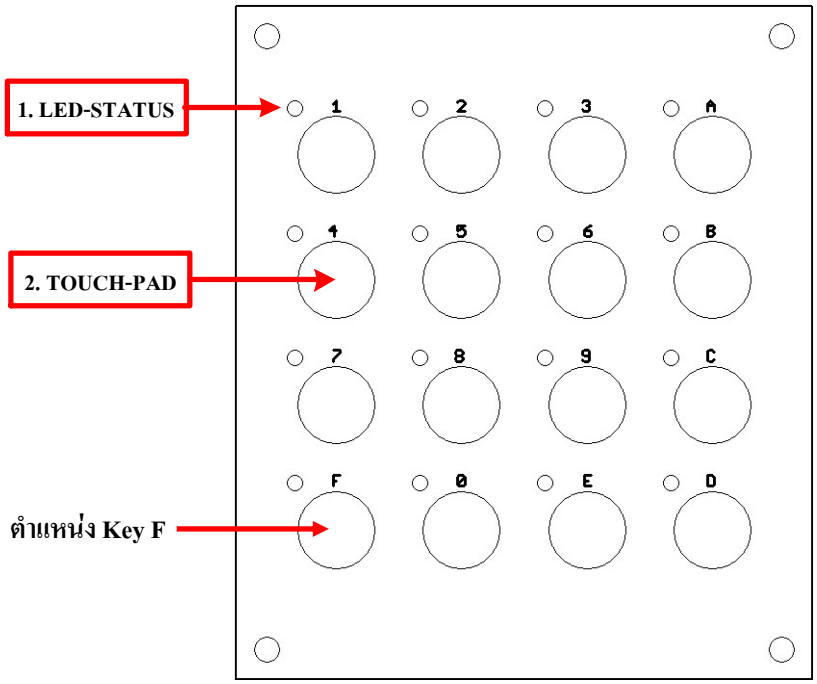


Touch-PAD Board

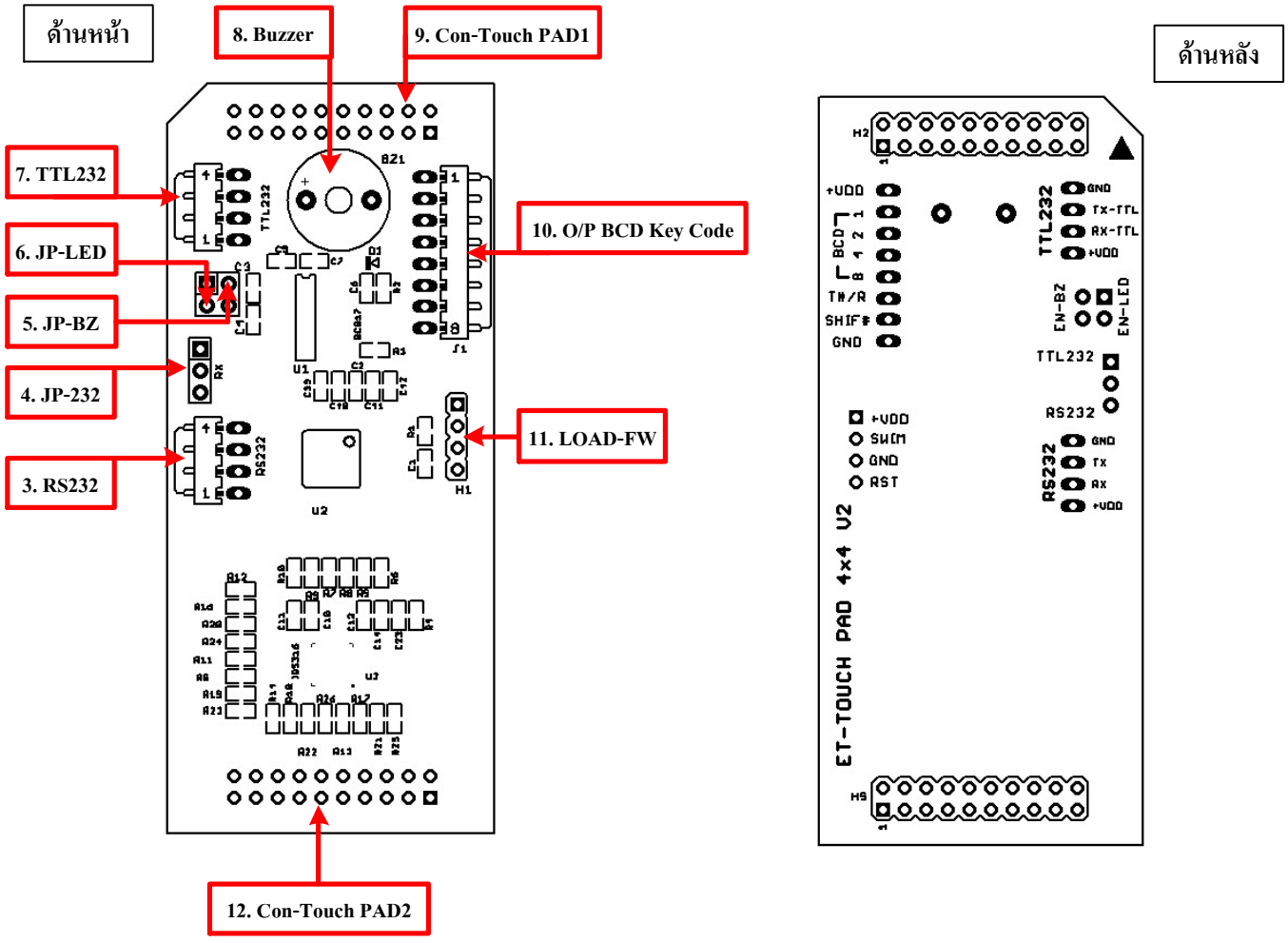
MCU-Control Board

รูปที่2.1 แสดงการวางบอร์ด Touch PAD และ MCU Control สำหรับใช้งาน

จากรูปที่2.1 แสดงการประกอบบอร์ด ET-TOUCH PAD 4x4 V2 สำหรับใช้งาน เมื่อผู้ใช้ถอดบอร์ดทั้งสองออกจากกันแล้วจะประกอบเข้าด้วยกันใหม่ ให้ประกอบดังรูปข้างต้น โดยในการประกอบให้หันลูกศร หรือPCB ด้านหัวตัด ของ MCU-Control Board ไปทางเดียวกับรูปที่ Screen อยู่ด้านหลังของ Touch-PAD Board ดังรูป แล้วจึงประกบบอร์ดทั้งสองเข้าด้วยกัน



รูปที่2.2 แสดง Touch PAD บอร์ดด้านหน้า

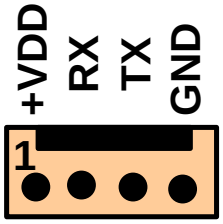


รูปที่2.3 แสดง MCU Control บอร์ด ด้านหน้าและด้านหลัง

- 1. LED-STATUS : เป็น LED แสดงสถานะ การกด Key ซึ่งจะ ติด/ดับ ตามจังหวะ การ กด/ปล่อย Key ของผู้ใช้ และจะติดกระพริบ เป็นจังหวะเมื่อบอร์ด Power-ON ผู้ใช้สามารถ Control LED นี้ได้ด้วย ASCII Command ผ่าน RS232 โดย LED นี้ (16ดวง) จะถูก Enable การทำงานด้วย Jumper JP-LED (6)

- 2. TOUCH-PAD : เป็นตำแหน่ง Key สำหรับให้ผู้ใช้กด หรือสัมผัส ส่วนตัวเลขที่แสดงบน PAD จะเป็นค่า Key Code ประจำ Key นั้นๆเมื่อถูกกดจะส่งค่า Key ออกมาทาง RS232 หรือ ทาง Connector O/P BCD Key Code โดยดูรายละเอียดค่า Key Code ของ Key ต่างได้จากตาราง KEY CODE

- 3. RS232 : เป็นขั้วต่อ RS232 Line Driver ซึ่งได้ต่อผ่าน IC Line Drive 3232 ที่อยู่บนบอร์ดออกมาแล้ว ดังนั้นขั้วต่อนี้ผู้ใช้สามารถนำไปต่อเข้ากับ PORT RS232 ของ PC ได้ หรือต่อเข้ากับ Port RS232 ของ MCU ภายนอกที่มีการต่อผ่าน IC Line Drive 3232 ออกมาเช่นเดียวกัน ซึ่งการต่อก็ให้ ไขว้สายระหว่าง RX,TX ด้วย สำหรับหน้าที่ของขั้วต่อนี้จะใช้ส่งสถานะ การกด/ปล่อย Key และส่งค่า Key Code ของ Key ที่ กด/ปล่อย ออกมาในรูปแบบของ ASCII CODE ให้กับผู้ใช้ ซึ่งรายละเอียดสามารถดูได้ในหัวข้อ “ การอ่าน Key Code แบบ ASCII ” นอกจากนี้ก็ยังใช้รับคำสั่งจากผู้ใช้เพื่อควบคุมการทำงานของ Buzzer และ LED STAUS ในขณะที่ Key ไม่ถูกกดด้วย โดยรายละเอียดการส่งคำสั่งสามารถดูได้ในหัวข้อ “COMMAND สำหรับ CONTROL LED&BUZZER”



รูปที่2.4 แสดง ขั้วต่อ RS232 Line Driver สำหรับต่อส่งคำสั่งและอ่านค่า Key Code แบบ ASCII CODE

รายละเอียดของขาแต่ละ PIN

+VDD,GND = ใช้สำหรับต่อไฟเลี้ยงบอร์ด VDC 3.3V(ใช้กับ MCU3.3V) หรือ 5V(ใช้กับ MCU 5V)

RX = ใช้รับคำสั่ง Control LED และ Buzzer จากผู้ใช้

TX = ใช้ส่งค่า สถานะ การ กด/ปล่อย Key และค่า Key Code (ASCII Code) ให้กับผู้ใช้

** เมื่อจะใช้งานขั้วต่อ RS232 นี้ จะต้อง Set Jumper JP-232(4) มาทางด้าน RS232 ด้วย**

- 4. JP-232 : เป็น Jumper ใช้สำหรับเลือกขั้วต่อใช้งาน 232 โดย ถ้า Set Jumper ไปทางด้าน TTL232 จะเป็นการเลือกใช้งานขั้วต่อ TTL232(หมายเลข7) แต่ถ้า Set Jumper ไปทางด้าน RS232 จะเป็นการเลือกใช้งานขั้วต่อ RS232(หมายเลข3)

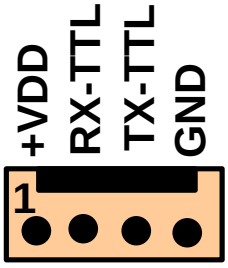
รูปที่2.5 แสดงการ Set Jumper เลือกใช้งานขั้วต่อ 232

**ในการอ่านค่า Key Code แบบ ASCII Code ผู้ใช้ต้องเลือกขั้วต่อ 232 ขั้วต่อใดขั้วต่อหนึ่งเท่านั้น และ Set Jumper JP-232 ให้ถูกด้าน ซึ่งการจะเลือกใช้งานขั้วต่อใดก็ขึ้นอยู่กับบอร์ด MCU ที่ผู้ใช้นำมาต่ออ่านค่า Key Code โดยถ้าบอร์ดที่นำมาต่อมีการต่อ IC Line Drive 3232 ไว้แล้ว หรือต่อไปยัง 232 ของ PC ก็ให้เลือกใช้ขั้วต่อ RS232(3) แต่ถ้าต้องการนำไปต่อกับขา Uart ของ MCU โดยตรงก็ให้เลือกใช้ขั้วต่อ TTL232(7) เป็นต้น*

- 5. JP-BZ : เป็น Jumper สำหรับ ON/OFF Buzzer เมื่อมองจากด้านหลังของบอร์ด MCU Control ก็คือ Jumper ด้านล่าง โดยถ้าใส่ Jumper จะเป็นการ ON- Buzzer ทันที และถ้าถอด Jumper ออกจะเป็นการ OFF- Buzzer ทันทีเช่นกัน ซึ่งการกำหนด Jumper นี้จะมีผลต่อการใช้คำสั่ง Control Buzzer ทาง RS232 ด้วย

- 6. JP-LED : เป็น Jumper สำหรับ กำหนดการทำงานของ LED Status Key เมื่อมองจากด้านหลังของบอร์ด MCU Control ก็คือ Jumper ด้านบน โดยถ้าใส่ Jumper จะเป็นการกำหนดให้ LED Status Key ทำงานตามปกติที่กำหนดไว้ใน Firmware แต่ถ้าถอด Jumper ออก LED ทั้งหมดจะไม่ทำงาน ซึ่งการกำหนด Jumper นี้จะมีผลต่อการใช้คำสั่ง Control LED Status Key ทาง RS232 ด้วย ***" การ Set Jumper นี้จะมีผลต่อการทำงานของ LED ตามที่กล่าวไว้ข้างต้น ก็ต่อเมื่อตัวบอร์ด ET-Touch PAD 4x4 มีการ Power-ON ใหม่หลังจาก Set Jumper "***

- 7. TTL232 : เป็นขั้วต่อ RS232 ในระดับสัญญาณแบบTTL ซึ่งขั้วต่อนี้ผู้ใช้สามารถนำไปต่อเข้ากับ Pin Uart (Rx,Tx) ของ MCU ได้โดยตรง ในการต่อให้ไขว้สายระหว่าง RX,TX ด้วย การทำงานของขั้วต่อนี้จะเหมือนกับขั้วต่อ RS232 หมายเลข 3 ทุกประการเพียงแต่จะมีระดับของสัญญาณทำงานที่ต่ำกว่าทำให้สามารถนำไปต่อกับขา MCU ได้โดยตรง



รูปที่2.6 แสดง ขั้วต่อ TTL232 สำหรับต่อส่งคำสั่งและอ่านค่า Key Code แบบ ASCII CODE

รายละเอียดของขาแต่ละ PIN

+VDD,GND = ใช้สำหรับต่อไฟเลี้ยงบอร์ด VDC 3.3V(ใช้กับ MCU3.3V) หรือ 5V(ใช้กับ MCU 5V)

RX-TTL = ใช้รับคำสั่ง Control LED และ Buzzer จากผู้ใช้

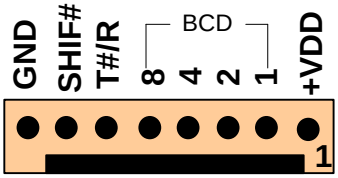
TX-TTL = ใช้ส่งค่า สถานะ การ กด/ปล่อย Key และค่า Key Code (ASCII Code) ให้กับผู้ใช้

** เมื่อจะใช้งานขั้วต่อ TTL232 นี้ จะต้อง Set Jumper JP-232(4) มาทางด้าน TTL232 ด้วย**

- 8. Buzzer : Buzzer เป็นตัวกำเนิดเสียง Beep เมื่อมีการ Touch Key หรือกำเนิดเสียงดนตรีเมื่อบอร์ด Power-On โดยสามารถ Control เสียงได้ด้วย Ascii Command ทาง RS232 ซึ่ง Buzzer นี้จะถูก Enable การทำงานด้วย Jumper JP-BZ (5)

- 9. CON-Touch PAD1 : เป็นขั้วต่อ 20 Pin ใช้สำหรับต่อไปยังบอร์ด TOUCH-PAD เพื่อ Control การ On/Off ของ LED บนบอร์ด

- 10. O/P BCD Key Code : เป็น Connector ขนาด 8 PIN แสดงดังรูปที่ 2.7 ทำหน้าที่ในการส่งค่า Key Code ของ Key ที่กดหรือปล่อย ในรูปแบบของ Binary BCD8421 และยังทำหน้าที่ส่งค่าสถานะ การ กด/ปล่อย Key ออกมาให้ผู้ใช้นำไปใช้งาน พร้อมทั้งเป็นขั้วจ่ายไฟเลี้ยงให้กับ ET-TOUCH PAD 4x4 V2 ด้วย



รูปที่2.7 แสดง Connector สำหรับอ่านค่า Key Code แบบ Binary BCD8421

<u>รายละเอียดของขาแต่ละ PIN</u>

+VDD,GND = ใช้สำหรับต่อไฟเลี้ยงบอร์ด VDC 3.3V(ใช้กับ MCU3.3V) หรือ 5V(ใช้กับ MCU 5V)

BCD = เป็นขา ส่ง Key Code 4 bit โดย PIN หมายเลข 8 เป็นบิต MSB ค่า Key Code จะถูก ส่งออกมาทุกครั้งที่มี การกด Key และปล่อย Key โดยค่า Key Code ที่ส่งออกมา จะเป็นค่าของ Key ที่กดหรือปล่อยล่าสุด และค้างค่านั้นไว้จนกว่า จะมีการกด Key อื่นๆใหม่

T#/R = Touch/Release บอกสถานะ การ กด/ปล่อย Key ใดๆโดยเมื่อกด Key จะให้ Logic เป็น 0 ค้างไว้จนกว่า Key จะถูกปล่อย ในทางกลับกัน เมื่อ Key ถูกปล่อยหรือไม่มีการ กด Key ขานี้ก็จะให้ Logic เป็น 1 ค้างไว้เช่นกัน จนกว่าจะมีการกด Key ถึงจะให้ Logic เป็น 0 อีก

SHIF# = SHIFT บอกสถานะ การ กด/ปล่อย Key ร่วม+Key ใดๆ (อ้างอิงตำแหน่ง Key ตามรูปที่ 2.2 โดย Key-F จะถูกกำหนดให้เป็น Key ร่วม) โดยปกติ Pin นี้จะให้ Logic เป็น 1 แต่เมื่อมีการกด Key ร่วม คือกด Key F + Key ใดๆ Pin นี้จะให้ Logic เป็น 0 จนกว่า Key ทั้ง 2 จะถูกปล่อย Pin นี้ก็จะกลับมาเป็น Logic 1 เหมือนเดิม (การกด Key ร่วมนั้นผู้ใช้ จะต้องกด Key F ค้างไว้ก่อนแล้วจึงตามด้วย Key ใดๆ)

ในกรณีที่ผู้ใช้ใช้งาน Key แบบเดี่ยว ให้ผู้ใช้อ่านค่าสะถานะ การ กด/ปล่อย Key จาก Pin T#/R เท่านั้น แต่ถ้าผู้ใช้มีการใช้งาน Key แบบเดี่ยว และแบบ Key ร่วมด้วย ผู้ใช้จะต้องอ่านค่าสะถานะ การ กด/ปล่อย Key จาก Pin T#/R และ SHIF#

- 11. LOAD-FW : ใช้ Upgrade Firmware ให้กับ ET-TOUCH PAD 4x4 V2 (ปกติจะต้องส่งมาให้ทาง ETT Upgrade)

- 12. CON-Touch PAD2 : เป็นขั้วต่อ 20 Pin ใช้สำหรับต่อไปยังบอร์ด TOUCH-PAD เพื่อ รับสัญญาณการ Touch จากผู้ใช้

<u>หมายเหตุ</u> ในการอ่านค่า Key Code นั้นผู้ใช้ควรเลือกอ่านค่าแบบใดแบบหนึ่งเท่านั้น ระหว่าง ASCII Code หรือ Binary Code เพื่อเป็นการเลือกต่อใช้งาน Connecter ชุดใดชุดหนึ่งเท่านั้น โดยถ้าต้องการอ่านค่า Key Codec แบบ ASCII Code ก็ให้เลือกต่อ ใช้งาน Connecter RS232(3) หรือ TTL232(7) แต่ถ้าต้องการอ่านค่า Key Code แบบ Binary BCD8421 ก็ให้เลือกต่อใช้งาน Connecter O/P BCD Key Code(10)

ในส่วนของไฟเลี้ยงบอร์ดนั้น(+VDD,GND) ผู้ใช้สามารถเลือกต่อเข้าที่ Connecter ตัวใดตัวหนึ่งเท่านั้นระหว่าง Connecter RS232(3) หรือ TTL232(7) หรือ O/P BCD Key Code(10)

3. การทำงานและการอ่านค่า Key Code ของ ET-TOUCH PAD 4x4 V2

การทำงานโดยรวม เมื่อจ่ายไฟให้กับบอร์ด LED Status Key จะกระพริบเป็นจังหวะตามเสียง Beep ค่าสถานะของ PIN BCD จะเป็น Logic0 ส่วน Pin T#/R และ SHIF# จะเป็น Logic1 ส่วนที่ขั้ว 232 จะไม่มีการส่ง data อะไรออกมา เมื่อมีการกดคีย์ใดคีย์หนึ่งที่ไม่ใช่ Key F หรือกด Key F ร่วมกับ Key อื่นๆค้างอยู่ Key ที่เหลือจะถูกล็อคไม่สามารถกดได้จนกว่าจะมีการปล่อย Key ที่กดทั้งหมดเสียก่อนถึงจะกด Key ใดๆได้อีก ทุกครั้งที่มีการกดคีย์ใดๆก็ตามจะมีเสียง Beep 1ครั้ง และ LED ประจำคีย์นั้นๆก็จะติดค้างไปจนกว่าจะมีการปล่อย Key , LED ถึงจะดับ หรือ กด Key ค้างไว้นานเกิน 20 วินาที LED ก็จะดับเช่นกัน เนื่องจาก ระบบการ Touch ถูกปรับสภาวะการทำงานใหม่

เมื่อคีย์ใดถูกกดหรือปล่อย Output Key Code ทั้งแบบ Binary และ ASCII รวมทั้ง สัญญาณ T#/R จะถูก Update ตามสถานะของ Key ที่ถูกกดและปล่อยล่าสุด นั่นคือ ไม่ว่าจะกดคีย์หรือปล่อยคีย์ ค่า Key Code จะถูกส่งออกมาเสมอ รวมทั้งสถานะของสัญญาณ T#/R ก็จะมีการเปลี่ยนแปลงเสมอเช่นกัน ส่วนสถานะของสัญญาณ SHIF# จะถูก Update หรือมีการเปลี่ยนแปลงเมื่อ มีการกดหรือปล่อย Key F กับ Key ร่วมใดๆเสียก่อน (SHIF# จะมีการเปลี่ยนแปลงสถานะเมื่อมีการใช้งาน 2 Keyร่วมกัน)

สำหรับเสียง Buzzer และ LED Status Key ปกติจะถูก Set ให้ทำงาน โดยการ Enable Jumper JP-BZ และ JP-LED ไว้ ถ้าผู้ใช้ไม่ต้องการใช้งานส่วนใดก็ให้ถอด Jumper ของตัวอุปกรณ์ที่ไม่ต้องการใช้งานออกแล้วทำการ Power-On บอร์ดใหม่

ตาราง KEY CODE ของ ET-TOUCH PAD 4x4 V2

KEY	FOR Binary MODE						FOR ASCII Mode (RS232 TTL-Line Drive)	
	BCD 8421 KEY CODE						ASCII KEY CODE	
	8	4	2	1	HEX		ASCII	HEX
1	0	0	0	1	0x01		'1'	0x31
2	0	0	1	0	0x02		'2'	0x32
3	0	0	1	1	0x03		'3'	0x33
4	0	1	0	0	0x04		'4'	0x34
5	0	1	0	1	0x05		'5'	0x35
6	0	1	1	0	0x06		'6'	0x36
7	0	1	1	1	0x07		'7'	0x37
8	1	0	0	0	0x08		'8'	0x38
9	1	0	0	1	0x09		'9'	0x39
0	0	0	0	0	0x00		'0'	0x30
A	1	0	1	0	0x0A		'A'	0x41
B	1	0	1	1	0x0B		'B'	0x42
C	1	1	0	0	0x0C		'C'	0x43
D	1	1	0	1	0x0D		'D'	0x44
E	1	1	1	0	0x0E		'E'	0x45
F	1	1	1	1	0x0F		'F'	0x46

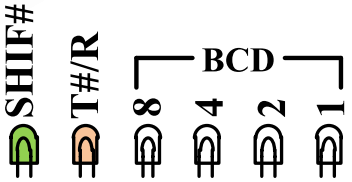
3.1 การอ่าน Key Code แบบ Binary BCD8421

การอ่านค่า Key Code แบบ Binary BCD8421 จะอ่านค่าจาก Connecter "O/P BCD Key Code" โดยสัญญาณต่างๆของ Connecter จะมีการเปลี่ยนแปลงตามสภาวะการทำงานดังนี้

- สภาวะ เริ่มต้น(Default) = เมื่อ Power-ON ที่ Connector O/P BCD Key Code จะถูก Set ดังนี้

PIN SHIF# = 1 ; PIN T#/R = 1

PIN8 = 0 ; PIN4 = 0 ; PIN2 = 0 ; PIN1 = 0

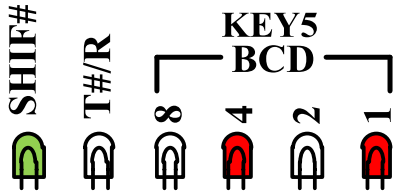


รูปที่ 3.1 แสดงสถานะDefault ที่ Pin ต่างๆ ของ Connecter 8 Pin

- ***สภาวะ กด(Press) Key 1 Key*** = เมื่อผู้ใช้กด Key ใดๆหนึ่ง Key เสียง Beep จะดังขึ้น LED Status ของ Key ที่กดจะติด และที่ Connector 8 PIN ก็จะมีการเปลี่ยนแปลงค่าสถานะต่างๆดังนี้

1) PIN T#/R (เป็นสัญญาณการ กด หรือ ปล่อย key 1 key) จะเปลี่ยนสถานะ logic จาก 1 ไปเป็น 0 ค้างไว้ตลอดที่มีการกด Key อยู่ ส่วน Pin SHIF# จะไม่มีการเปลี่ยนแปลงยังคงเป็น Logic 1

2) PIN BCD8,4,2,1 จะมีการเปลี่ยนสถานะตามค่า Key Code ของ key ที่กดล่าสุดออกมาค้างไว้ โดยค่า Key Code จะเป็นไปตามตารางด้านบนซีกซ้ายมือในช่อง HEX



รูปที่ 3.2 แสดงตัวอย่างสถานะที่ Pin ต่างๆ ของ Connecter 8 Pin เมื่อมีการกด Key 5

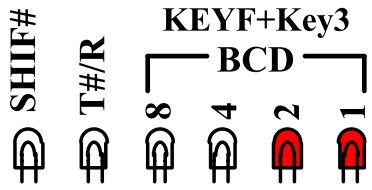
- ***สภาวะ กด(Press) Key ร่วม 2 Key*** = สำหรับการกด Key ร่วม 2 Key นั้น จะเป็นการกด Key F+Key ใดๆ ร่วมกัน 2 Key เสียง Beep จะดังและ LED Status Key ที่กด จะติดตามลำดับ Key ที่กด และ Connector 8 PIN ก็จะมีการเปลี่ยนแปลงค่าสถานะดังนี้

1) เมื่อกด Key F: PIN T#/R จะเปลี่ยนสถานะ logic จาก 1 ไปเป็น 0 ค้างไว้ตลอดที่มีการกด Key F

2) PIN BCD8,4,2,1 จะส่งค่า Key Code ของ key F ออกมาซึ่งก็คือค่า 0x0F

3) เมื่อกด Key ที่2 เป็น Key ใดๆที่เหลืออยู่ (Key F ยังกดอยู่) : PIN T#/R จะยังคงสถานะ Logic 0 ค้าง อยู่ PIN SHIF# จะเปลี่ยนสถานะ logic จาก 1 ไปเป็น 0 ค้างไว้ตลอดที่มีการกด Key F + Key ใดๆ

4) PIN BCD8,4,2,1 จะส่งค่า Key Code ของ key ที่2 ที่ถูกกดออกมาให้ผู้ใช้ค้างไว้



รูปที่ 3.3 แสดงตัวอย่างสถานะที่ Pin ต่างๆ ของ Connecter 8 Pin เมื่อมีการกด Key ร่วม (Key F+Key3)

- ***สภาวะ ปล่อย (Release) Key 1Key*** = เมื่อผู้ใช้ปล่อย Key ใดๆที่กดอยู่ ที่ Connector 8 PIN ก็จะมีการเปลี่ยนแปลงค่าสถานะดังนี้

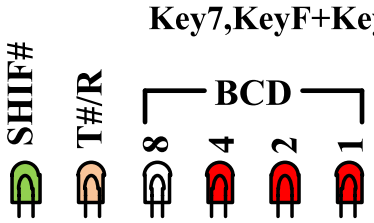
1) PIN T#/R (เป็นสัญญาณการ กด หรือ ปล่อย Key) จะเปลี่ยนสถานะ logic จาก 0 ไปเป็น 1 ค้างไว้ตลอดที่ยังไม่มีการกดคีย์ใดๆ ส่วน Pin SHIF# จะไม่มีการเปลี่ยนแปลงยังคงเป็น Logic 1

2) PIN BCD8,4,2,1 จะมีการเปลี่ยนสถานะตามค่า Key Code ของ Key ที่ปล่อยล่าสุดออกมาค้างไว้

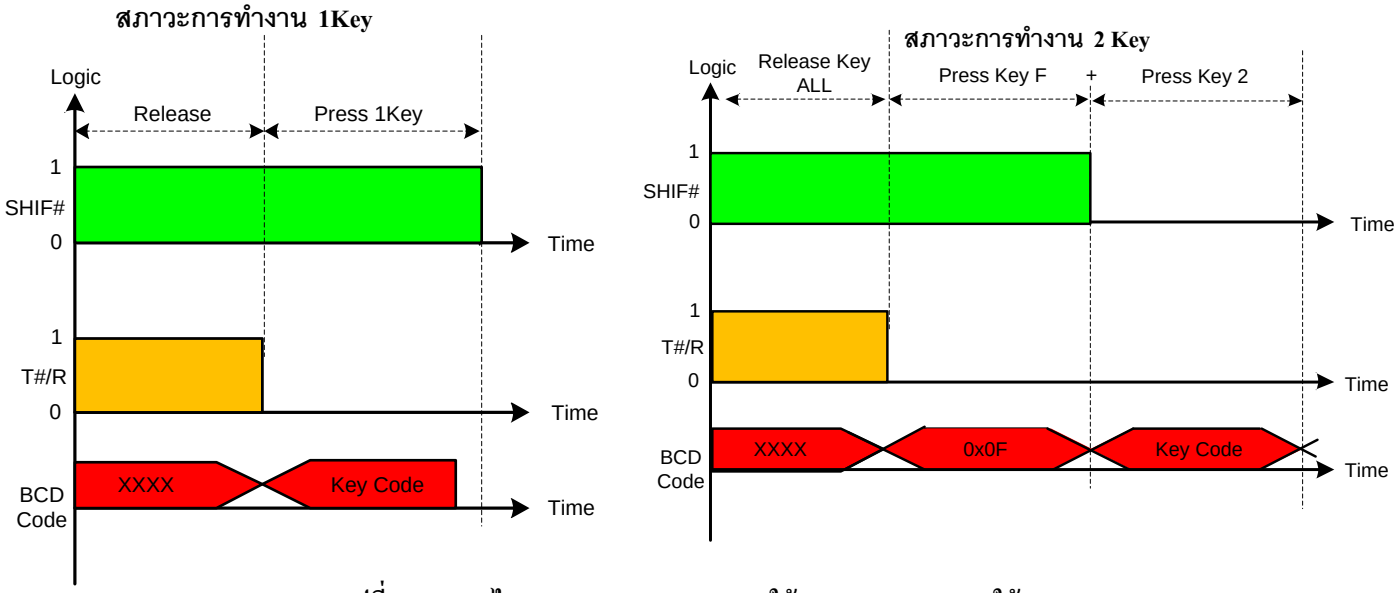
- ***สภาวะ ปล่อย (Release) Key ร่วม 2 Key*** = ในกรณีที่มีการใช้งาน Key ร่วม เมื่อผู้ใช้ปล่อย Key ใดเพียง Key เดียว สัญญาณที่ Connecter 8 Pin จะยังไม่มีการเปลี่ยนแปลง(สภาวะจะยังเหมือนตอนที่กด Key ร่วมทั้ง 2 อยู่) แต่เมื่อมีการปล่อย Key ร่วมทั้ง 2 Key ที่กดอยู่ (ไม่มีการกด Key ใดๆ) ที่ Connector 8 PIN ก็จะมีการเปลี่ยนแปลงค่าสถานะดังนี้

1) PIN T#/R (เป็นสัญญาณการ กด หรือ ปล่อย Key) และ Pin SHIF# จะเปลี่ยนสถานะ logic จาก 0 ไปเป็น 1 ค้างไว้ตลอดที่ยังไม่มีการกดคีย์ใดๆใหม่

2) PIN BCD8,4,2,1 จะมีการเปลี่ยนสถานะตามค่า Key Code ของ Key ใดๆ ที่กดร่วมกับ Key F ออกมาค้างไว้



รูปที่ 3.4 แสดงตัวอย่างสถานะที่ Pin ต่างๆ ของ Connecter 8 Pin เมื่อมีการปล่อย Key เดียว(Key7) หรือ Key ร่วม (Key F+Key7)



รูปที่3.5 แสดง ไดอะแกรมการกด Key แบบใช้งาน 1 Key และแบบใช้งาน 2 Key

สรุปการอ่านค่า Key Code แบบ Binary BCD8421: จากสภาวะการกดและปล่อย Key ที่กล่าวไปข้างต้นในการเขียนโปรแกรมนั้นถ้าไม่มีการใช้งาน Key ร่วม ผู้ใช้สามารถตรวจสอบสถานะ การกด/ปล่อย Key ได้จาก Pin T#/R โดยถ้า T#/R=0 Key ถูกกด ถ้า T#/R=1 ไม่มีการกด Key จากนั้นก็อ่านค่า Key Code ของ Key ที่กดได้จาก Pin BCD8421 ไปใช้งาน

และในกรณีที่มีการใช้งาน Key ร่วม ผู้ใช้สามารถตรวจสอบการกด Key ได้จาก Pin T#/R และ Pin SHIFT # โดยถ้า T#/R=0 และ SHIF# =1 แสดงว่าเป็นการกด Key เดียว แต่ถ้า T#/R=0 และSHIF# = 0 แสดงว่าเป็นการกด Key ร่วม จากนั้นก็ทำการอ่านค่า Key Code ของ Key ที่ถูกกดได้จาก Pin BCD8421 เช่นเดิม ส่วนการปล่อย Key ร่วม ผู้ใช้สามารถตรวจสอบสถานะ การปล่อย Key ได้จาก Pin T#/R หรือ Pin SHIF# Pin ใด Pin หนึ่ง หรือทั้ง 2 Pin พร้อมกันก็ได้ เนื่องจากเมื่อมีการปล่อย Key ร่วมทั้ง 2 Key แล้วค่าสถานะของ Pin ทั้ง 2 นี้จะเปลี่ยนเป็น Logic 1 เหมือนกัน

3.2 การอ่าน Key Code แบบ ASCII

การอ่าน Key Code แบบ ASCII ผู้ใช้สามารถอ่านค่าได้จาก Connecter “RS232 หรือ TTL232” ซึ่งจะเป็นการเชื่อมต่อแบบ Serial Port โดย Baud Rate ในการสื่อสารจะถูกกำหนดตายตัวไว้ที่ 9600 bit/s

การเลือกต่อ Connecter ใช้งานก็จะขึ้นอยู่กับผู้ใช้ว่า บอร์ดที่นำมาต่อร่วมกับ Touch Pad นั้นในส่วนที่เป็น Port 232 เป็นแบบ TTL หรือเป็นแบบผ่าน IC Drive 232 ถ้าเป็นแบบ TTL(Rx,Tx มาจากขา MCU โดย ตรง) ก็ให้เลือกต่อเข้าที่ Connecter TTL232 และ Set Jumper JP-232 มาทางด้าน TTL232 แต่ถ้า Port232 ของบอร์ดที่นำมาต่อร่วมมีการต่อผ่าน IC Drive 232 ออกมา หรือเป็นการต่อไปยัง RS232 ของ PC ก็ให้ผู้ใช้เลือกต่อเข้าที่ Connecter RS232 ของ Touch Pad และให้ Set Jumper JP-232 มาทางด้าน RS232 โดยไม่ว่าจะเลือกต่อใช้งาน Connecter ใดก็ตามเวลาต่อผู้ใช้จะต้องต่อ ขา TX เข้ากับขา Rx ของอีกฝั่ง ,และต่อขา RX เข้ากับขา Tx ของอีกฝั่ง ซึ่งจะเป็นลักษณะการต่อแบบไขว้สายกัน

สำหรับค่า Key Code ที่ส่งออกมาให้ผู้ใช้นั้นจะเป็นค่า ASCII Code โดยรูปแบบของค่าที่ส่งออกมานั้นจะเป็นไปตามสภาวะการกด Key ดังนี้

- ***สภาวะเริ่มต้น(Default)*** = เมื่อ Power-ON LED Status Key จะกระพริบเป็นจังหวะตามเสียง Beep และจะยังไม่มีการส่งข้อมูลใดๆ ออกมาที่ Port 232 นั่นคือถ้าไม่มีการกด Key ตัว Touch Pad จะไม่มีการส่งค่าใดๆออกมาทาง Port 232

- ***สภาวะกด(Press) Key 1Key*** = เมื่อมีการกด Key ใดๆหนึ่ง Key ตัว ET-TOUCH PAD ก็จะส่ง Data ออกมาที่ Port232 จำนวน 3 Byte คือ ASCII 'P' จะถูกส่งออกมาเป็น Byte แรก เพื่อบอกสถานะ การกด Key และ ASCII Code ของ Key ที่กดจะถูกส่งออกมาเป็น Byte ที่2 และจบด้วยการส่ง 0x0D เป็น Byte สุดท้าย โดยค่า Key Code จะเป็นไปตามตาราง Key Code ด้านบนซีกขวามือในช่อง ASCII หรือ HEX

	ASCII CODE		Hex
	Status Key (Byte 1)	Key Code (Byte 2)	End Byte (Byte 3)
PRESS KEY	'P' (0x50)	'0-9', 'A-F'	0x0D

Ex. กด Key 5 ค่าที่ได้ก็จะเป็น ***P5 และ 0x0D***

ตารางรูปแบบการส่ง data เมื่อมีการกด Key 1Key

- ***สภาวะ กด(Press) Key ร่วม 2 Key*** = สำหรับการกด Key ร่วม 2 Key นั้น จะเป็นการกด KeyF + Key ใดๆ ร่วมกัน 2 Key เริ่มต้น

1) กด Key F ตัว ET-TOUCH PAD ก็จะส่ง ASCII 'P'(0x50) ออกมาเป็น Byte แรก และ Key Code ASCII 'F'(0x46) ออกมาเป็น Byte ที่ 2 และจบด้วย 0x0D เป็น Byte สุดท้าย
2) กด Key ที่ 2 เป็น Key ใดก็ได้ ในขณะที่ Key F ยังคงกดค้างอยู่ ET-TOUCH PAD ก็จะส่ง ASCII 'F'(0x46) ออกมาเป็น Byte แรก และตามด้วย ASCII Key Code ของ Key ที่กดออกมาเป็น Byte ที่ 2 และจบด้วย 0x0D เป็น Byte สุดท้าย

	ASCII CODE		Hex
	Status Key (Byte 1)	Key Code (Byte 2)	End Byte (Byte 3)
1.PRESS Key F	'P' (0x50)	'F'	0x0D
2.PRESS Key Other	'F' (0x46)	'0-9', 'A-D'	0x0D

Ex. กด Key F+Key2 ค่าที่ได้ก็จะเป็น
PF และ 0x0D
F2 และ 0x0D (นำค่านี้ไปใช้)

ตารางรูปแบบการส่ง data เมื่อมีการกด Key ร่วม

สังเกตว่าการใช้งาน Key ร่วมจะมีการส่ง Data ออกมาให้ 2 ชุด ชุดแรกจะเป็น Data ของ Key F โดย ASCII 'P' จะถูกส่งออกมาเป็น Byte แรก เพื่อบอกสถานะ การกด Key ซึ่งจะเป็นรูปแบบเหมือนกับสภาวะการกด Key 1 Key ตามที่กล่าวไปข้างต้น ชุดที่2จะเป็น Data ของ Key ใดๆที่เรากดร่วม โดย ASCII 'F' จะถูกส่งออกมาเป็น Byte แรก เพื่อบอกสถานะการกด Key ร่วม แล้วจึงตามด้วยค่า Key Code ของ Key ใดๆ ที่ถูกกดร่วม

- ***สภาวะ ปล่อย (Release)Key 1Key*** = เมื่อผู้ใช้ปล่อย Key ใดๆ ที่กดอยู่ ตัว ET-TOUCH PAD ก็จะส่ง Data ออกมาที่ขั้วต่อ 232 จำนวน 3 Byte คือ ASCII 'R' จะถูกส่งออกมาเป็น Byte แรก เพื่อบอกสถานการณ์ปล่อย Key และ ASCII Code ของ Key ที่ปล่อยจะถูกส่งออกมาเป็น Byte ที่2 และจบด้วย 0x0D เป็น Byte สุดท้าย

	ASCII CODE		Hex
	Status Key (Byte 1)	Key Code (Byte 2)	End Byte (Byte 3)
RELEASE KEY	'R' (0x52)	'0-9', 'A-F'	0x0D

Ex. ปล่อย Key 7 ค่าที่ได้ก็จะเป็น
R7 และ 0x0D

ตารางรูปแบบการส่ง data เมื่อมีการปล่อย Key *แบบ 1 Key และ 2 Key*

- สภาวะ ปล่อย (Release)Key ร่วม 2Key = เมื่อมีการใช้งาน Key ร่วม คือกด KeyF + Key ใดๆ ร่วมกัน 2 Key อยู่ ถ้าผู้ใช้ทำการปล่อย Key ใด Key หนึ่ง ตัว ET-TOUCH PAD จะยังไม่ส่ง Data ออกมาที่ขั้วต่อ 232 จนกว่าผู้ใช้จะมีการปล่อย Key ร่วมทั้ง 2 Key ที่กดอยู่ (ไม่มีการกด Key ใดๆ) ตัว ET-TOUCH PAD ถึงจะส่ง Data ออกมาที่ขั้วต่อ 232 จำนวน 3 Byte ดังตารางด้านบน ซึ่งจะมีรูปแบบเหมือนกับการปล่อย Key 1 Key

สรุปการอ่านค่า Key Code แบบ ASCII : จากสภาวะการกดและปล่อย Key ที่กล่าวไปข้างต้นในการเขียนโปรแกรมนั้นถ้าไม่มีการ ใช้งาน Key ร่วม ผู้ใช้สามารถตรวจสอบสถานะ การกด/ปล่อย Key ได้จากตัวอักษร ‘P’=กด Key และ ‘R’= ไม่มีการกด Key ซึ่งจะถูกส่งออกมาเป็น Byte แรก และสามารถอ่านค่า Key Code ของ Key ที่ถูกกดหรือปล่อยได้ใน Byte ที่ 2

และในกรณีที่มีการใช้งาน Key ร่วม ผู้ใช้สามารถตรวจสอบการกด Key ได้จากตัวอักษร ‘P’ หรือ ‘F’ ที่อ่านได้ใน Byte แรก โดยถ้าเป็น อักษร ‘P’ แสดงว่าเป็นการกด Key เดียว แต่ถ้าเป็นอักษร ‘F’ แสดงว่าเป็นการกด Key ร่วม จากนั้นทำการอ่านค่า Key Code ของ Key ที่กดได้ใน Byte ที่2 ผู้ใช้ก็จะทราบได้ว่า Key Code ที่อ่านได้ เกิดจากการกด Key เดียว หรือ Key ร่วม เพื่อจะได้ กำหนดฟังก์ชันการทำงานของการกด Key ได้ถูกต้อง ส่วนการปล่อย Key ร่วม ผู้ใช้สามารถตรวจสอบสถานะได้จากตัวอักษร ‘R’ ซึ่ง จะถูกส่งออกมาเป็น Byte แรกเมื่อ Key ร่วมทั้ง 2 Key ถูกปล่อย

4. การส่ง Command Control Buzzer & LED Status Key

สำหรับในส่วนของ Port 232 นั้น นอกจากจะใช้ส่งค่า ASCII Key Code ให้กับผู้ใช้อ่านไปใช้งานแล้วยังสามารถใช้รับคำสั่ง ASCII Command จากผู้ใช้ เพื่อควบคุมการทำงานของ Buzzer และ LED Status Key ที่อยู่บนบอร์ดได้ด้วย วัตถุประสงค์ก็เพื่อให้ผู้ใช้สามารถใช้งานในส่วนของ Buzzer และ LED Status Key ทำหน้าที่เป็น Alarm ได้ในขณะที่ไม่มีการกด Key และเมื่อกด Key ตัว Key ก็จะกลับมาทำการส่งค่า Key Code ให้กับผู้ใช้ได้เหมือนเดิมโดยอัตโนมัติ

ดังนั้นถ้าผู้ใช้ต้องการจะ Control Buzzer และ LED ในส่วนนี้ ก็จะต้องต่อ Port ใช้งานที่ขั้วต่อ RS232 หรือ TTL 232 เหมือนกับการอ่านค่า Key Code แบบ ASCII ไปยัง Port 232 ของบอร์ดที่จะนำมาใช้ควบคุม และจะต้อง Enable Jumper JP-BZ และ JP-LED ไว้ด้วย โดย Baud Rate สำหรับการส่ง Command จะถูกกำหนดไว้ตายตัวที่ 9600 bit/s

รูปแบบของ Command และ Data ที่ใช้นั้นจะใช้ตัว พิมพ์ใหญ่ทั้งหมด และอยู่ในรูปของ ASCII Code ซึ่งสามารถแทนด้วยสัญลักษณ์ ASCII เช่น ‘#’ หรือ แทนด้วย ASCII Code ซึ่งจะเท่ากับ 0x23 เป็นต้น โดยในแต่ละ Command จะมี Data อยู่ 6 Byte สำหรับ Byte สุดท้าย คือ ค่า Enter จะไม่มีสัญลักษณ์ให้เรียกใช้งาน ดังนั้นจะแทนด้วย ASCII Code = 0x0D และในแต่ละ Command ที่ส่งไปถ้าถูกต้อง ผู้ใช้ก็จะได้รับ ASCII Command ตอบกลับมา 3 Byte คือ *OK เพื่อแจ้งให้ผู้ใช้ทราบว่าพร้อมรับคำสั่งต่อไป ถ้าผู้ใช้มีการส่ง Command ผิดรูปแบบ จะไม่มีการตอบกลับใดๆจากบอร์ด ET-TOUCH PAD 4x4 V2 สำหรับรูปแบบ Command ที่ใช้งานมีดังนี้

1.) COMMAND ‘BZ’ (SOUND-ON/OFF)

เป็นคำสั่ง ON/OFF เสียง Buzzer โดยถ้ากำหนดค่า Data เป็น ‘1’ = Buzzer ON เสียงจะดังต่อเนื่องตลอดเวลา แต่ถ้าค่า Data เป็น ‘0’ = Buzzer OFF ไม่มีเสียง ซึ่งมีรูปแบบคำสั่งดังนี้

Start	Command	Mark#1	Data	END
Byte1	Byte2-3	Byte4	Byte5	Byte6
#	BZ	=	‘0-1’	Enter (0x0D)
Respond Command จาก Board				
*	OK			

Data = ‘0’ : Buzzer OFF หยุดเสียง
‘1’ : Buzzzer ON เสียงดังต่อเนื่อง

Ex. ส่งคำสั่ง ON-Buzzer เป็นเวลา 1 วินาที แล้ว OFF-Buzzer

```
Main()
  {
    char  enter = 0x0D     ;
    printf("#BZ=1",enter) ;    //Sent Command On-Buzzer
    delay_ms(1000)         ;
    printf("#BZ=0",enter) ;    //Sent Command Off-Buzzer
  }
```

2.) COMMAND 'BP' (SOUND BEEP)

เป็นคำสั่งกำเนิดเสียง Beep โดยสามารถกำหนดความยาวของเสียง Beep ได้ตั้งแต่ '0-9' โดยถ้า Data = '1' เสียง Beep จะมีความยาวเสียงสั้นสุด และความยาวของเสียง Beep จะเพิ่มขึ้นตามลำดับหมายเลขที่กำหนด ซึ่งถ้าค่า Data = '0' จะให้ความยาวของเสียง Beep ดังยาวสุด รูปแบบคำสั่งมีดังนี้

Start	Command	Mark#1	Data	END
Byte1	Byte2-3	Byte4	Byte5	Byte6
#	BP	=	'0-9'	Enter (0x0D)
Respond Command จาก Board				
*	OK			

Data = '0-9' : ความยาวเสียง Beep เมื่อ '1' ให้ความยาวเสียง Beep สั้นสุด
'0' ให้ความยาวเสียง Beep ยาวสุด

Ex. ส่งคำสั่งกำเนิดเสียง Beep กำหนดความยาวเสียง Beep ระดับ 5

```
Main()
  {
    char  enter = 0x0D     ;
    printf("#BP=5",enter) ;    //Sent Command Beep
  }
```

3.) COMMAND 'LE' (LED-ENABLE)

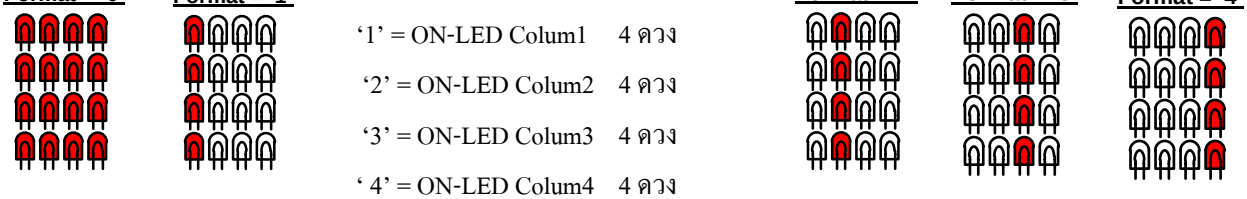
เป็นคำสั่ง ON-LED Status Key ทั้งหมด หรือ ให้ ON ครั้งละ 4 ดวง ตามแนว Colum หรือ Row ซึ่งขึ้นอยู่กับการเลือกรูปแบบของผู้ใช้ โดยสามารถเลือกรูปแบบการ ON-LEDได้ 9 รูปแบบ โดยมีรูปแบบคำสั่งดังนี้

Start	Command	Mark#1	Format ON-LED	END
Byte1	Byte2-3	Byte4	Byte5	Byte6
#	LE	=	'0-8'	Enter (0x0D)
Respond Command จาก Board				
*	OK			

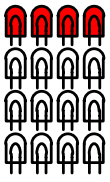
Format ON-LED : กำหนดรูปแบบการ ON ของ LED โดย

'0' = ON-LED ทั้งหมด 16 ดวง
'1' = ON-LED Colum1 4 ดวง
'2' = ON-LED Colum2 4 ดวง
'3' = ON-LED Colum3 4 ดวง
' 4' = ON-LED Colum4 4 ดวง

Format = '0' Format = '1' Format = '2' Format = '3' Format = '4'



Format = '5' Format = '6'



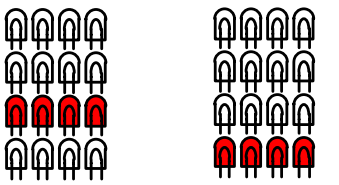
'5' = ON-LED Row1 4 ดวง

'6' = ON-LED Row2 4 ดวง

'7' = ON-LED Row3 4 ดวง

' 8' = ON-LED Row4 4 ดวง

Format = '7' Format = '8'



Ex. ส่งคำสั่ง ให้ LED ทุกดวง ON ทั้งหมด

```
Main()
  {  char  enter = 0x0D   ;
     printf("#LE=0",enter) ;    //Sent Command On-LED ALL
  }
```

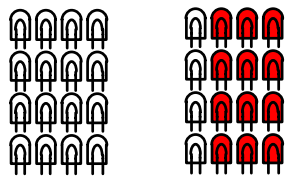
4.) COMMAND 'LD' (LED-DISABLE)

เป็นคำสั่ง OFF-LED Status Key ทั้งหมด หรือ ให้ OFF ครั้งละ 4 ดวง ตามแนว Colum หรือ Row ซึ่งขึ้นอยู่กับการเลือกรูปแบบของผู้ใช้ โดยสามารถเลือกรูปแบบการ OFF-LEDได้ 9 รูปแบบ โดยมีรูปแบบคำสั่งดังนี้

Start	Command	Mark#1	Format OFF-LED	END
Byte1	Byte2-3	Byte4	Byte5	Byte6
#	LD	=	'0-8'	Enter (0x0D)
Respond Command จาก Board				
*	OK			

Format OFF-LED : กำหนดรูปแบบการ OFF ของ LED โดย

Format = '0' Format = '1'



'0' = OFF-LED ทั้งหมด 16 ดวง

'1' = OFF-LED Colum1 4 ดวง

'2' = OFF-LED Colum2 4 ดวง

'3' = OFF-LED Colum3 4 ดวง

' 4' = OFF-LED Colum4 4 ดวง

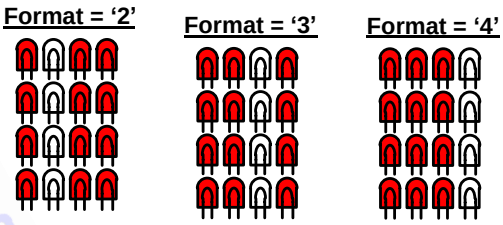
'5' = OFF-LED Row1 4 ดวง

'6' = OFF-LED Row2 4 ดวง

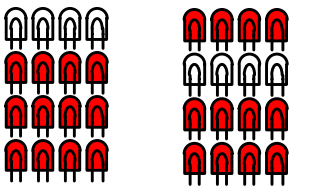
'7' = OFF-LED Row3 4 ดวง

' 8' = OFF-LED Row4 4 ดวง

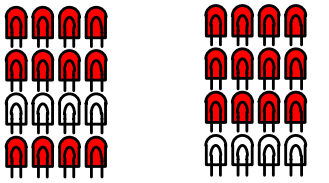
Format = '2' Format = '3' Format = '4'



Format = '5' Format = '6'



Format = '7' Format = '8'



Ex. ส่งคำสั่ง ให้ LED ทุกดวง OFF ทั้งหมด

```
Main()
  {  char  enter = 0x0D   ;
     printf("#LD=0",enter) ;    //Sent Command OFF-LED ALL
  }
```

Note *ในการส่งคำสั่งแบบต่อเนื่องระหว่างคำสั่งทุกคำสั่งควรจะมีการตรวจสอบ Respond Command ของคำสั่งที่ส่งออกไปเสมอ หรือจะใช้การ delay ด้วยค่าเวลาที่เหมาะสม ระหว่างคำสั่งแทนก็ได้ เพื่อป้องกันคำสั่งที่ส่งออกไปทับซ้อนกัน ซึ่งจะทำให้การส่งคำสั่งนั้นผิดพลาดได้*

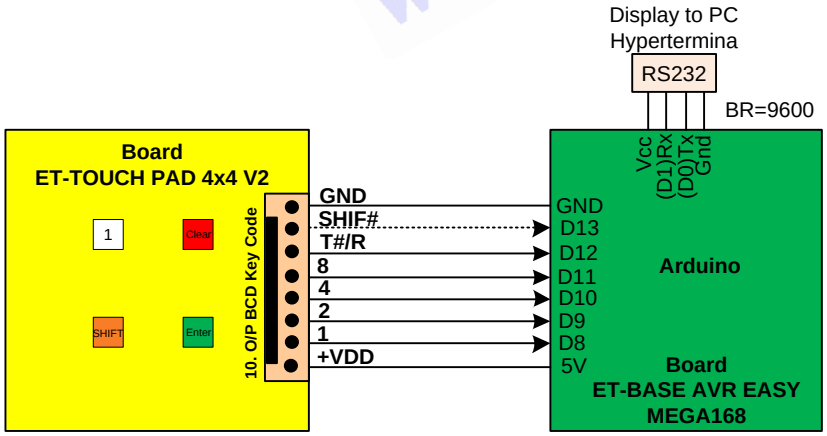
5. ตัวอย่างโปรแกรมและการต่อวงจรอ่านค่า Key Code ด้วย MCU

สำหรับตัวอย่างการอ่านค่า Key Code ทั้งแบบ Binary BCD8421 และ ASCII Code จะทำมารองรับกับ MCU 4 ตระกูลคือ AVR EASY MEGA168 โดยใช้ Arduino Compiler , AVR MEGA 64/128 โดยใช้ C-WIN AVR และ MCS51-AT89C51RE2 โดยใช้ C-Keil Compiler และ PIC8722 โดยใช้ CCS Compiler ซึ่งแต่ละ MCU จะมีตัวอย่างอยู่ 2 รูปแบบคือ การอ่านค่า Key Code แบบ Binary BCD8421 และแบบ ASCII Code โดยมีรายละเอียดการทำงานของโปรแกรมดังนี้ (Source Code อยู่ใน CD)

ตัวอย่าง Read Key แบบ Binary BCD8421

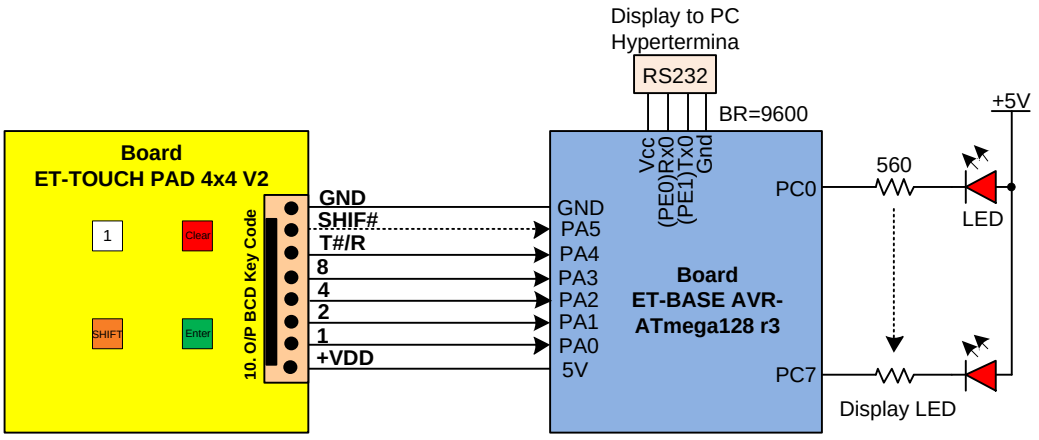
ตัวอย่าง Ex1_Read_BCD_1Key : สำหรับตัวอย่างนี้ จะเป็นการอ่านค่า Key Code แบบ BCD8421 โดยใช้งาน Key แต่ละ Key แบบ Key เดียว ดังนั้นเราสามารถใช้งาน Key F (SHIFT) ได้เหมือน Key อื่นๆ ส่วน Pin SHIF# จะไม่ถูกต่อใช้งานไปยัง MCU

เริ่มต้นเมื่อผู้ใช้ทำการ Touch Key ตัว MCU ก็จะทำการตรวจสอบสัญญาณ T#/R เพื่อตรวจสอบสถานะ การ Touch Key ถ้ามีการ Touch (T#/R=0) ก็จะทำการอ่านค่า Key Code จากขั้วต่อ BCD8421 แล้วส่งค่า Key Code ที่อ่านได้ออกแสดงผลทาง Port ที่ต่อ LED ไว้ (ยกเว้น Arduino ไม่แสดงผลในส่วนนี้) และพิมพ์ค่า Key Code ออกไปแสดงที่ Hyper terminal บน PC ด้วย ตัวอย่างเช่น ถ้า Touch Key5 ที่ LED ก็จะแสดงค่าเป็น 0x05 และที่ Hyper terminal ก็จะแสดงข้อความ “Key_Code BCD = 0x05 ” เป็นต้น จากนั้น MCUก็จะทำการตรวจสอบสัญญาณ T#/R อีกครั้ง เพื่อตรวจสอบสถานะ การปล่อย Key (T#/R=1) เมื่อมีการปล่อย Key โปรแกรมก็จะวนกลับมาตรวจสอบสัญญาณ T#/R อีกเพื่อตรวจสอบการ Touch Key ต่อไป นี่เป็นการทำงานโดยรวมในตัวอย่างที่ 1 ของแต่ละ MCU สำหรับตัวอย่างวงจรการต่อที่รองรับตัวอย่างที่1จะแสดงดังรูปด้านล่าง



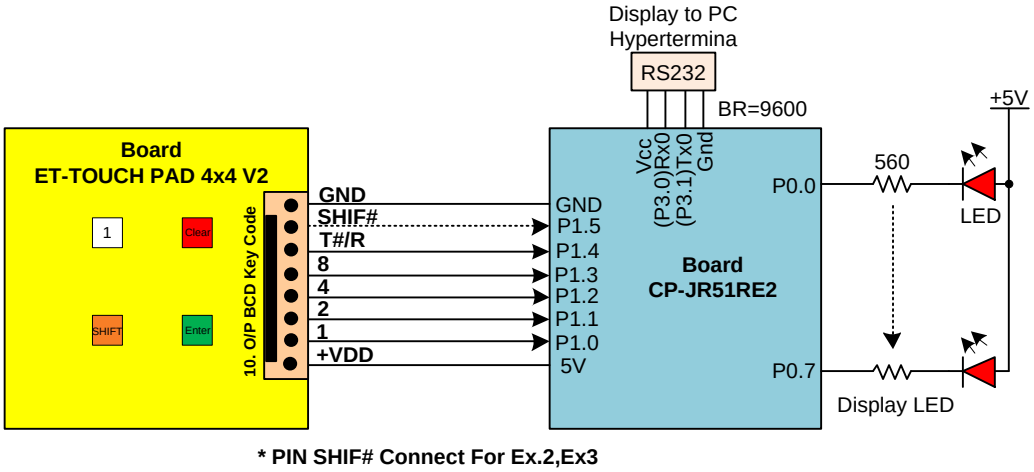
*** PIN SHIF# Connect For Ex.2,Ex3**

รูปที่5.1 การต่อวงจรอ่านค่า Key Code แบบ BCD ของบอร์ด ET-Touch Pad 4x4 V2 กับ ET-BASE AVR EASY MEGA168



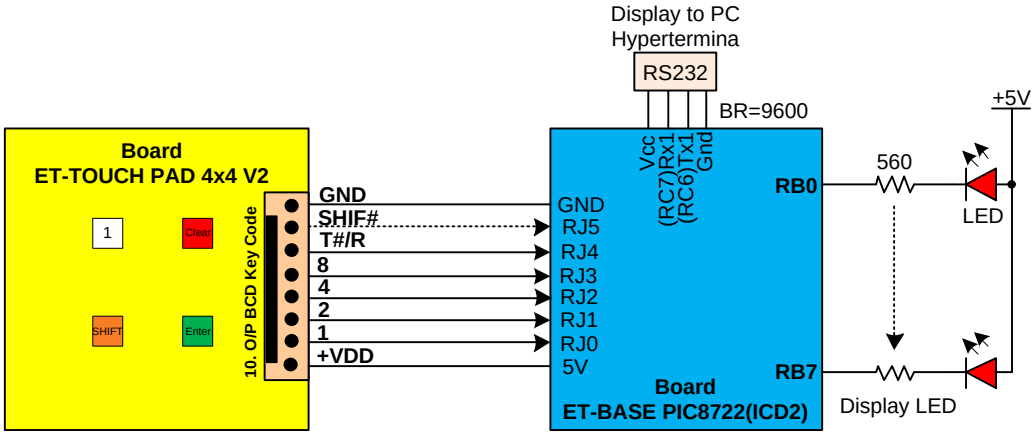
*** PIN SHIF# Connect For Ex.2,Ex3**

รูปที่5.2 การต่อวงจรอ่านค่า Key Code แบบ BCD ของบอร์ด ET-Touch Pad 4x4 V2 กับ ET-BASE AVR ATmega128 r3



*** PIN SHIF# Connect For Ex.2,Ex3**

รูปที่5.3 การต่อวงจรอ่านค่า Key Code แบบ BCD ของบอร์ด ET-Touch Pad 4x4 V2 กับ CP-JR51RE2



*** PIN SHIF# Connect For Ex.2,Ex3**

รูปที่5.4 การต่อวงจรอ่านค่า Key Code แบบ BCD ของบอร์ด ET-Touch Pad 4x4 V2 กับ ET-BASE PIC8722 (ICD2)

ตัวอย่าง Ex2_Read_BCD_ShifKey : สำหรับตัวอย่างที่2 จะเป็น การอ่านค่า Key Code แบบ BCD8421 เหมือนกับตัวอย่างที่1 แต่จะใช้งาน Key F เป็น Key ร่วมกับ Key อื่นๆด้วย ดังนั้นวงจรการต่อใช้งานตามรูปในตัวอย่างที่1จะต้องต่อ PIN SHIF# ไปยัง MCU ด้วย

เริ่มต้นเมื่อผู้ใช้ทำการ Touch Key ตัว MCU ก็จะทำการตรวจสอบสัญญาณ T#/R เพื่อตรวจสอบสถานะ การ Touch Key ถ้ามีการ Touch (T#/R=0) ก็จะทำการอ่านค่า Key Code จากขั้วต่อ BCD8421 แล้วตรวจสอบค่าที่ได้ว่าใช่ KeyF (0x0F)หรือไม่ ถ้าไม่ใช่ก็จะส่งค่า Key Code ออกแสดงผลทาง Port ที่ต่อ LED ไว้ (ยกเว้น Arduino ไม่แสดงผลในส่วนนี้) และพิมพ์ค่า Key Code ออกไปแสดงที่ Hyper terminal บน PC เหมือนกับตัวอย่างที่1 จากนั้น MCUก็จะทำการตรวจสอบสัญญาณ T#/R อีกครั้ง เพื่อตรวจสอบสถานะรอการปล่อย Key (T#/R=1) แต่ถ้า Key ที่กดเป็นKey F โปรแกรมก็จะวนรออ่านค่า Key Code ของ Key ที่จะกดร่วมกับ KeyF โดยคอยตรวจสอบสัญญาณ SHIF# ถ้าเป็น 0 แสดงว่ามีการกด Key ที่ 2 ร่วมกับ KeyF อยู่ โปรแกรมก็จะทำการอ่านค่า Key Code ของ Key ที่ 2 จากขั้วต่อ BCD8421 แล้วทำการ ส่งค่า Key Code ออกแสดงผลทาง Port ที่ต่อ LED ไว้ (ยกเว้น Arduino ไม่แสดงผลในส่วนนี้) และพิมพ์ค่า Key Code ออกไปแสดงที่ Hyper terminal บน PC ตัวอย่างเช่น ถ้า Touch KeyF+Key3 ที่ LED ก็จะแสดงค่าเป็น 0x03(LED Bit7 จะติดแสดงสถานะ การกด Key Shift) และที่ Hyper terminal ก็จะแสดงข้อความ “Key_Code BCD = Shift+0x03 ” เป็นต้น จากนั้น MCUก็จะทำการตรวจสอบสัญญาณ SHIF# และ T#/R เพื่อตรวจสอบสถานะรอการปล่อย Keyทั้ง 2 Key ที่ถูก Touch (SHIF#,T#/R=1)

ตัวอย่าง Ex3_Application_BCD_ShifKey : สำหรับในตัวอย่างที่ 3 นี้จะเป็นการประยุกต์ใช้งานโดยใช้ Key F เป็น Key ร่วมกับ Key อื่นๆ เช่นเดิม ดังนั้นวงจรการต่อในส่วนของ ET-TOUCH PAD กับ MCU จะเหมือนตัวอย่างที่1ซึ่งจะต้องต่อ PIN SHIF# ไปยัง MCU ด้วย ในส่วนของการแสดงผลจะใช้การแสดงผลบนจอ LCD ขนาด 16x2 โดยวงจรการต่อจอ LCD กับ MCU ให้ดูได้ใน Source Code

การทำงานในส่วนของการอ่านค่า Key Code จะเหมือนกับตัวอย่างที่2 ทุกประการ จะต่างกันตรงส่วนของโปรแกรมตอบสนองการกด Key เท่านั้น โดยในส่วนของการแสดงผลขณะยังไม่มีการกด Key ที่ LCD จะแสดงข้อความดังรูปที่5.5

รูปที่5.5

เมื่อมีการกด 1 Key ที่ไม่ใช่ KeyF เช่น Touch Key8 บน LCD ก็จะแสดงข้อความ ดังรูปที่5.6

รูปที่5.6

และถ้า Touch KeyF ร่วมกับ Key อื่นๆ (Touch 2 Key) เช่น Touch KeyF+Key5 บนจอ LCD ก็จะแสดงข้อความดังรูปที่5.7 โดยในตัวอย่างโปรแกรมจะกำหนดให้ Touch KeyF ร่วมกับ Key0-Key9 ได้เท่านั้น

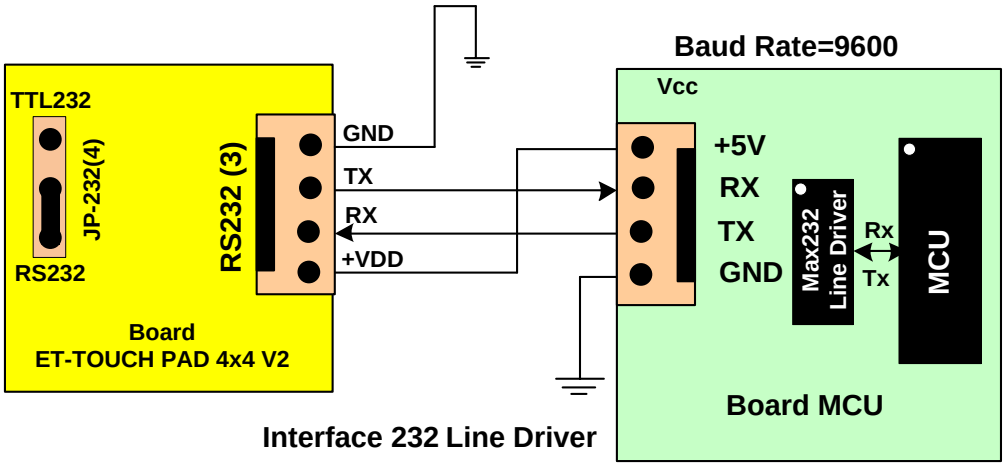
รูปที่5.7

ตัวอย่าง Read Key แบบ ASCII CODE

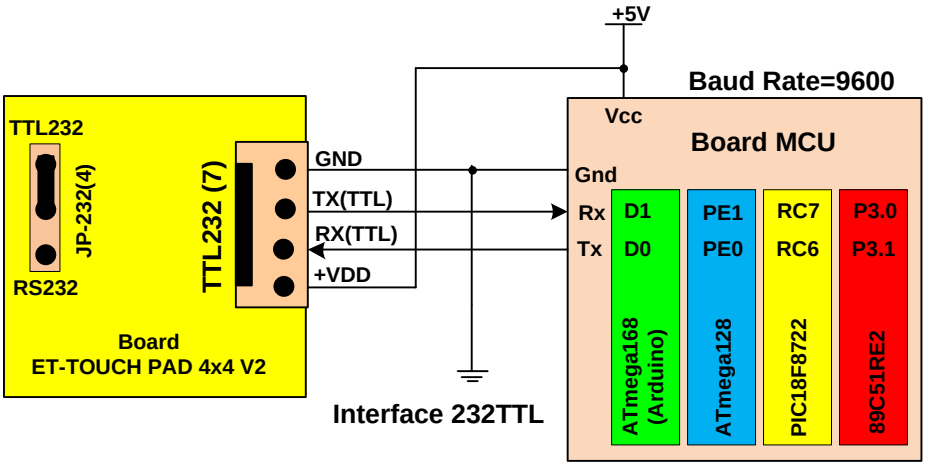
การอ่านค่าแบบ ASCII Code ค่า Key Code ที่อ่านได้ในแต่ละ Key ที่กดจะเป็นไปตามตาราง KEY CODE ของ ET-TOUCH PAD 4x4 V2 ในช่อง ASCII MODE โดยผู้ใช้สามารถอ่านค่าได้จากขั้วต่อ RS232(3) หรือ TTL232(7) ของ ET-TOUCH PAD 4x4 V2 ซึ่งจะใช้ BAUD RATE ในการรับส่ง อยู่ที่ 9600 bit/s เวลาต่อสายใช้งานก็ให้ต่อสายแบบไขว้ คือ RX,TX ของ MCU ต่อเข้ากับ TX,RX ของบอร์ด ET-Touch Pad ตามลำดับ ซึ่งในตัวอย่างเราจะต่อสายใช้งานที่ขั้วต่อ RS232(3) ดังนั้นผู้ใช้จะต้อง Set Jumper 'JP-232 (4)' ของบอร์ด ET- Touch Pad มาทางด้าน RS232ด้วย ที่เลือกใช้ขั้วต่อนี้เนื่องจากบอร์ด MCU ที่นำมาทดลองมีขั้วต่อ RS232 แบบผ่าน Line Driver ไว้ให้แล้ว แต่ถ้าผู้ใช้จะนำไปต่อเข้ากับขา 232 (Uart)ของ MCU โดยตรงก็ให้เลือกใช้งานขั้วต่อ TTL232(7) ของ ET-Touch Pad แทน

ตัวอย่าง Ex1_Read_ASCII_1Key : สำหรับตัวอย่างนี้ จะเป็นการอ่านค่า Key Code แบบ ASCII โดยใช้งาน Key แต่ละ Key แบบ Key เดียว ดังนั้นเราสามารถใช้งาน Key F (SHIFT) ได้เหมือน Key อื่นๆ และในตัวอย่างจะใช้ Interrupt เป็นตัวกำหนดจังหวะการรับข้อมูล โดยในส่วนของการแสดงผลจะแสดงค่า ASCII Code ในรูปของ Hex ออกทาง Port ของ MCU ที่ต่อ LED ไว้

เริ่มต้นเมื่อผู้ใช้ทำการ Touch Key โปรแกรมก็จะกระโดดไปรับข้อมูลมาเก็บไว้ จนครบ 3 Byte จากนั้นก็จะตรวจสอบว่า Byte สุดท้ายใช่ 0x0D หรือไม่ ถ้าใช่แสดงว่ารับข้อมูลมาครบและถูกต้องแล้ว จากนั้นก็จะกลับมาตรวจสอบข้อมูลที่รับมา Byte แรก ว่าใช่ ASCII 'P' หรือไม่ถ้าใช่แสดงว่าเป็นการ Touch Key โปรแกรมก็จะทำการ ส่งค่า data Byte ที่2 ซึ่งจะเป็นค่า ASCII Code ของ Key ที่ Touch ออกไปแสดงยัง Port ที่ต่อ LEDไว้ในรูปของ Hex เช่น ถ้ามีการ Touch Key 7 ค่า ASCII Code ในรูป Hex ก็คือ 0x37 เป็นต้น สุดท้ายโปรแกรมก็จะวนรออ่าน Data อีกครั้ง เพื่อตรวจสอบ Data ในการปล่อย Key ถ้า Data Byte แรกเป็น ASCII 'R' แสดงว่ามีการปล่อย Key แล้ว โปรแกรมก็จะกลับไปวนรออ่าน data สำหรับ Touch ในครั้งต่อไป (ในตัวอย่างของ Arduino เรากำหนดให้แสดงผลการ Touch ได้ตั้งแต่ Key0-Key9 เท่านั้น) สำหรับ ตัวอย่างวงจรการต่อที่รองรับตัวอย่างแสดงดังรูปด้านล่าง



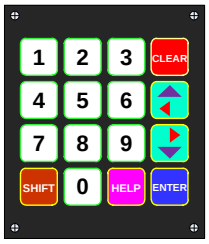
รูปที่5.8 การต่อวงจรอ่านค่า Key Code แบบ ASCII ของบอร์ด MCU กับ ET-Touch Pad 4x4 V2 (ผ่าน Line Driver)

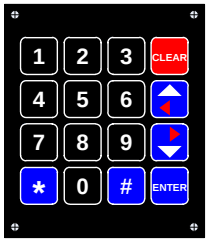


รูปที่5.9 การต่อวงจรอ่านค่า Key Code แบบ ASCII ของบอร์ด MCU กับ ET-Touch Pad 4x4 V2 (แบบ TTL)

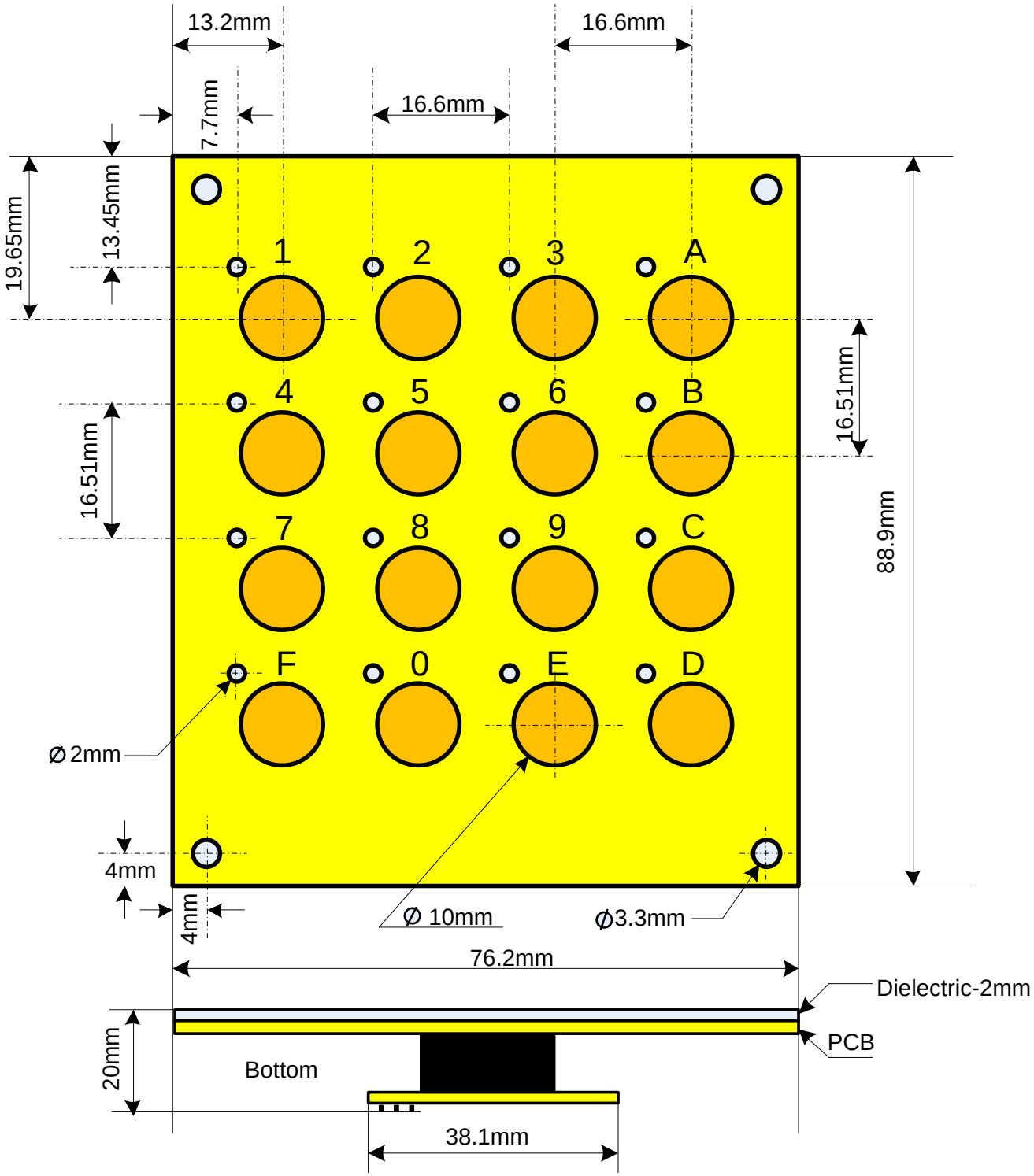
ตัวอย่าง Ex2_Read_ASCII_ShifKey : ในตัวอย่างนี้ จะเป็นการอ่านค่า Key Code แบบ ASCII โดยใช้งาน Key F เป็น Key ร่วมกับ Key อื่นๆ และใช้ Interrupt เป็นตัวกำหนดจังหวะการรับข้อมูล ในส่วนของการแสดงผลจะแสดงค่า ASCII Code ในรูปของ Hex ออกทาง Port ของ MCU ที่ต่อ LED ไว้

เริ่มต้นเมื่อผู้ใช้ทำการ Touch Key โปรแกรมก็จะกระโดดไปรับข้อมูลมาเก็บไว้ จนครบ 3 Byte จากนั้นก็จะตรวจสอบว่า Byte สุดท้ายใช่ 0x0D หรือไม่ ถ้าใช่แสดงว่ารับข้อมูลมาครบและถูกต้องแล้ว จากนั้นก็จะกลับมาตรวจสอบข้อมูลที่รับมา Byte แรก ว่าใช่ ASCII 'P' หรือไม่ถ้าใช่แสดงว่าเป็นการ Touch Key จากนั้นก็จะทำการตรวจสอบ Data ที่รับมาใน Byte ที่2 ซึ่งก็คือค่า ASCII Key Code ของ Key ที่ Touch ว่ามีค่าเท่ากับ ASCII CODE 'F' หรือไม่ถ้าไม่ใช่ก็แสดงว่าเป็นการ Touch Key เดียว โปรแกรมก็จะส่งค่า ASCII Code Byte ที่ 2 ออกไปแสดงยัง Port ที่ต่อ LEDไว้ในรูปของ Hex เช่น ถ้ามีการ Touch Key 0 ค่า ASCII Code ในรูป Hex ก็คือ 0x30 เป็นต้น (ในตัวอย่างของ Arduino เรากำหนดให้แสดงผลการ Touch ได้ตั้งแต่ Key0-Key9 เท่านั้น) แต่ถ้า Key ที่ Touch เป็น Key F โปรแกรมก็จะวนรอรับค่า คีย์ที่สองที่จะกดร่วมกับ Key F ซึ่งเมื่อมีการ Touch Key ที่2 โปรแกรมก็จะตรวจสอบ Data Byte สุดท้ายก่อนเช่นเดิมว่าเท่ากับ 0x0D หรือไม่ถ้าใช่ก็จะกลับมาตรวจสอบ Byte แรกว่า ใช่ ASCII 'F' หรือไม่ ถ้าใช่แสดงว่าเป็นการ Touch Keyร่วมที่2 โปรแกรมก็จะทำการส่งค่า ASCII Code Byte ที่ 2 ของ Key ที่ Touch ออกไปแสดงยัง Port ที่ต่อ LEDไว้ในรูปของ Hex เช่น ถ้ามีการ Touch KeyF+Key9 ค่า ASCII Code ในรูป Hex ก็คือ 0x39 เป็นต้น (ในโปรแกรมจะกำหนดให้ Touch KeyFร่วมกับ Key0 ถึง Key9 ได้เท่านั้น) สุดท้ายหลังมีการ Touch เกิดขึ้นแล้วโปรแกรมก็จะวนรออ่าน Data อีกครั้ง เพื่อตรวจสอบ Data ในการปล่อย Key ถ้า Data Byte แรกเป็น ASCII 'R' แสดงว่ามีการปล่อย Key แล้ว โปรแกรมก็จะกลับไปวนรออ่าน Data สำหรับ Touch ในครั้งต่อไป ส่วนวงจรการต่อเพื่อทดสอบโปรแกรมก็จะเหมือนกับตัวอย่างที่1 ข้างต้น...

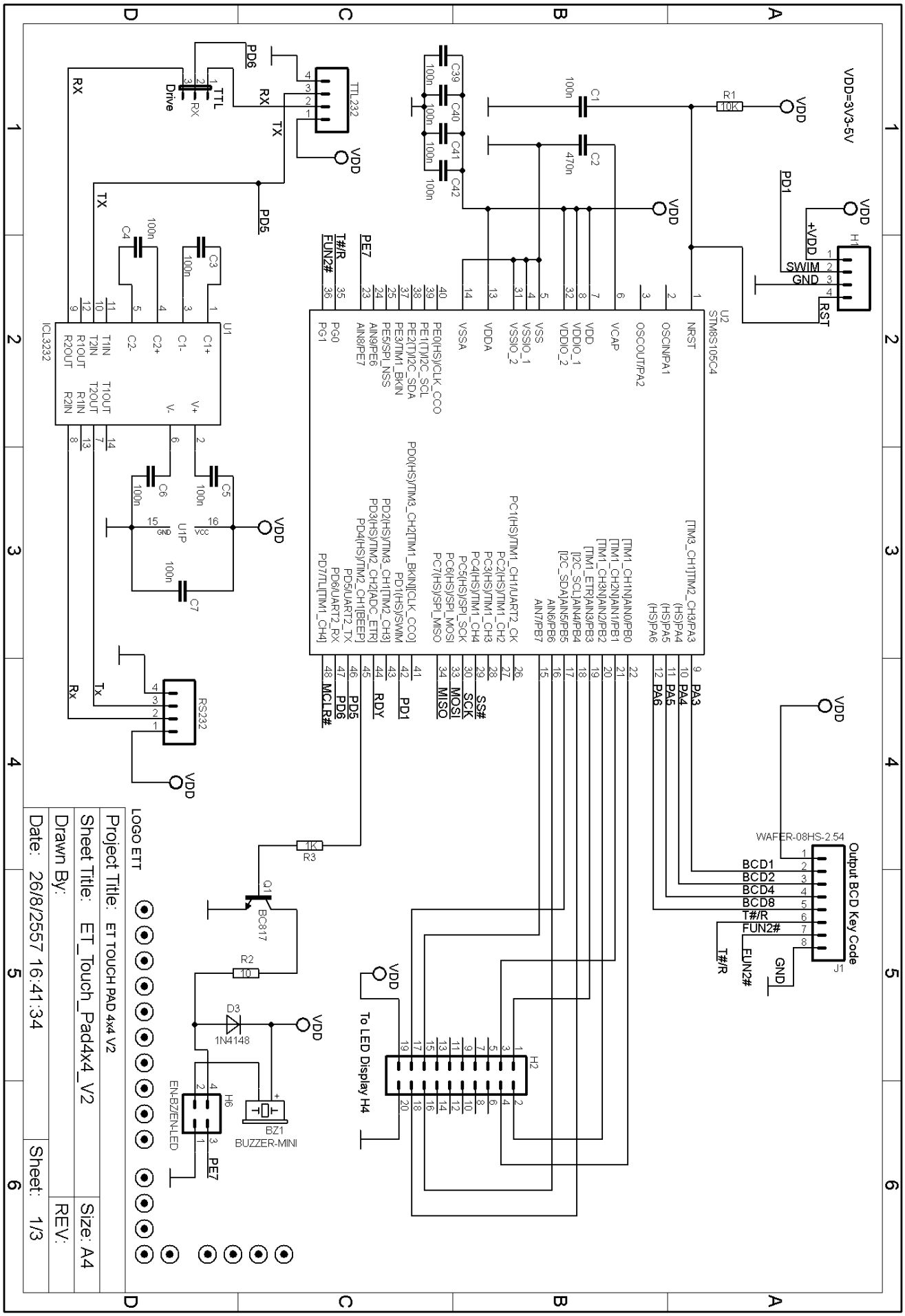




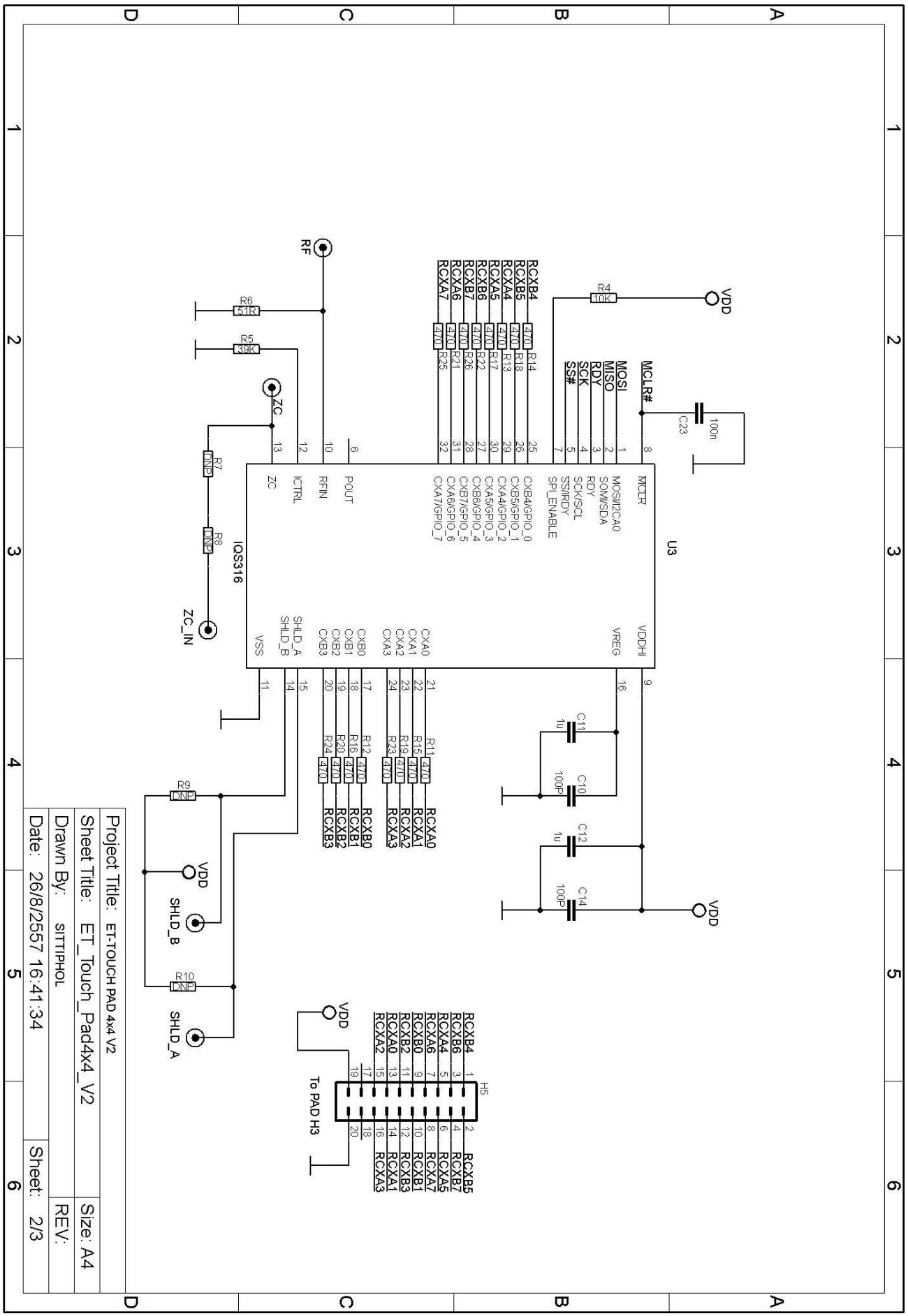
ขนาด PCB LAY OUT



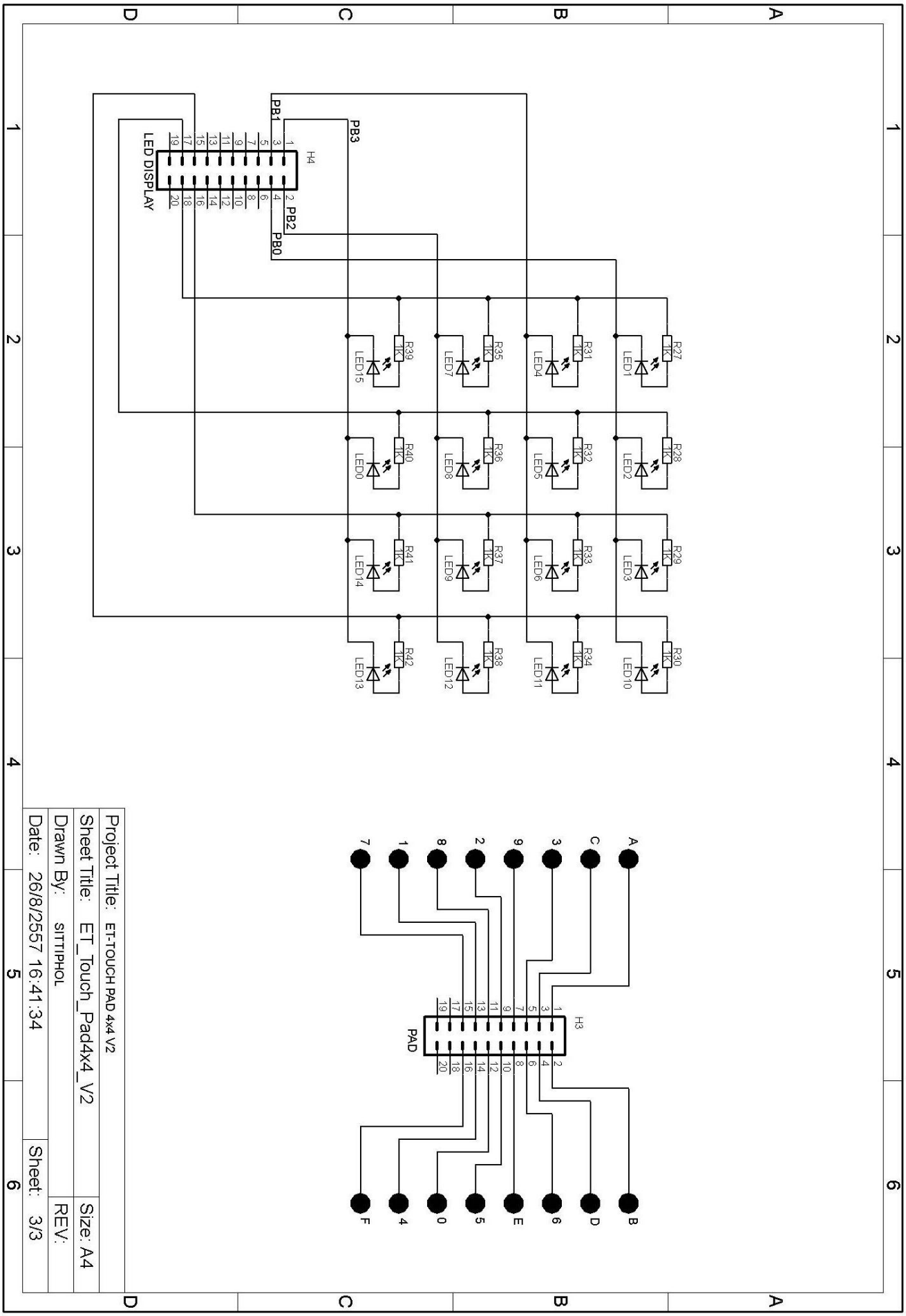
PCB LAY OUT



รูปวงจร ET-TOUCH PAD 4x4 V2 Sheet1



รูปวงจร ET-TOUCH PAD 4x4 V2 Sheet2



รูปวงจร ET-TOUCH PAD 4x4 V2 Sheet3