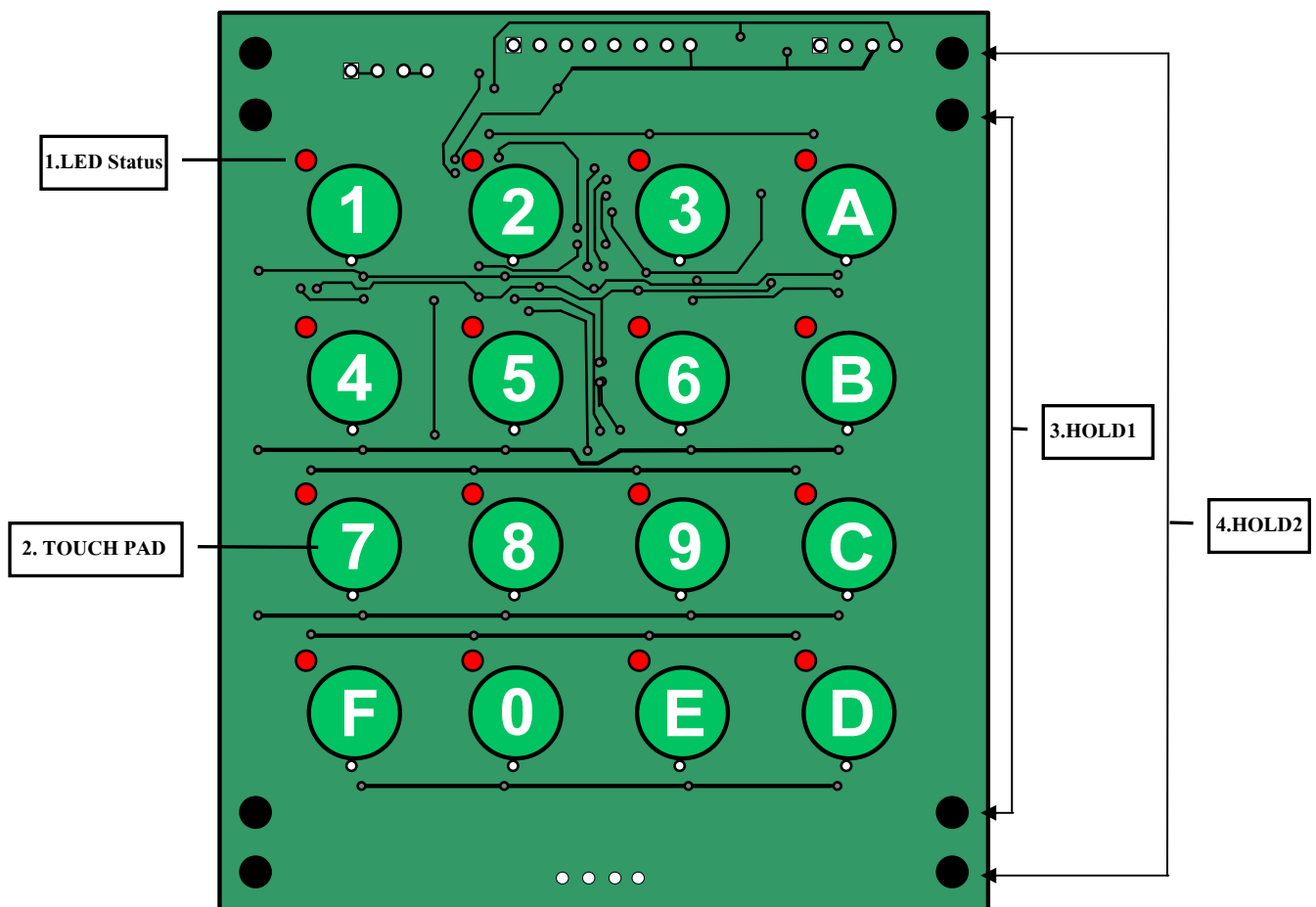


ET- TOUCH PAD 4x4

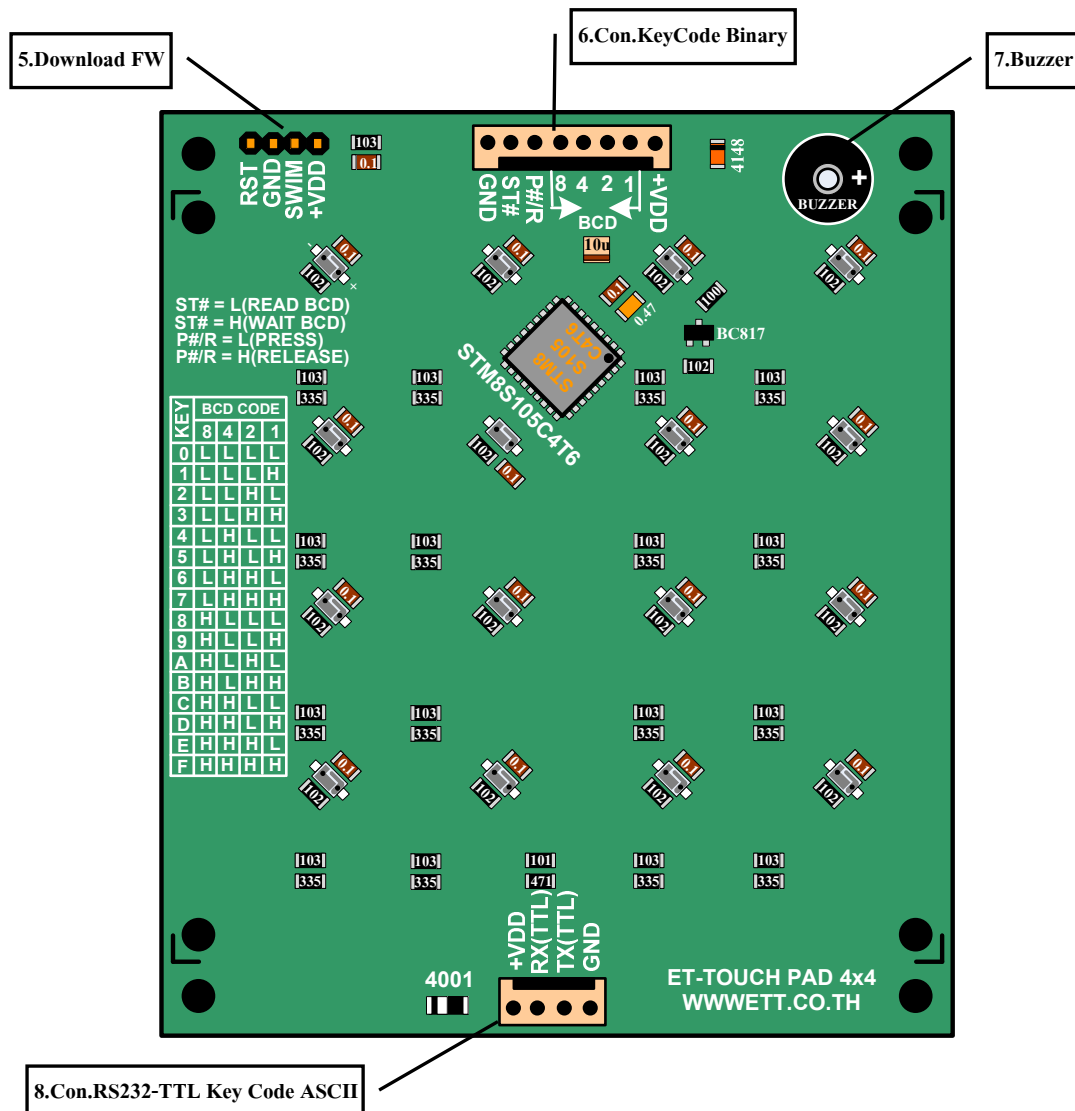
1. คุณสมบัติของบอร์ด ET-TOUCH PAD 4x4

- เป็น KEY Touch Sensing(สัมผัส) แบบ Capacitive sensing ขนาด 16 Key 4x4
- ใช้ไฟเลี้ยง +3.3 VDC หรือ +5 VDC
- แสดงสถานะการกด Key ของผู้ใช้งาน ด้วย เสียง และ LED ที่อยู่ในตำแหน่งของ Key นั้นๆ
- เวลาเริ่มต้นจ่ายไฟให้กับ ET-TOUCH PAD 4x4 จะมีเสียงและ LED ติดลักษณะไฟวิ่งเพื่อบอกสถานะเริ่มการทำงาน
- ส่งค่า Key Code ของ Key ที่กดออกมา 2 แบบคือ 1.) Binary Code(BCD8421) ผ่านทาง Connector 8 PIN โดยมีขา สัญญาณ ST# และ P#/R เพื่อบอกสถานะการกด หรือ ปลด Key 2.) ASCII Code ผ่านทางขั้วต่อ RS232-TTL(UART) Baud Rate 9600 ดาตัตัว โดยจะส่ง ASCII 'P' หรือ 'R' นำหน้าค่า Key Code ออกมาเพื่อให้รู้ว่าค่า Key Code ที่ส่งมา เกิดจากการกดหรือปลด Key
- แผ่นรอง Key Touch ถ้าเป็นพลาสติกใส หนาได้ประมาณ 1-2 มิลลิเมตร ถ้าเป็นวัสดุอื่น ความหนาจะขึ้นอยู่กับคุณสมบัติ ความไวทางไฟฟ้าของวัสดุนั้นๆ
- มี Key พิเศษ 1 Key สามารถใช้เป็น Key ปกติ หรือ ใช้เป็น Key FUN เพื่อกร่วมกับ Key อื่นๆที่เหลือได้(กด2keyพร้อมกัน)
- สามารถควบคุมการ On/OFF ของ Sound และ LED (เมื่อจะใช้งานเป็น Alarm)ได้ด้วย ASCII Command ผ่านทาง RS232

2. ลักษณะและโครงสร้างของบอร์ด ET-TOUCH PAD 4x4



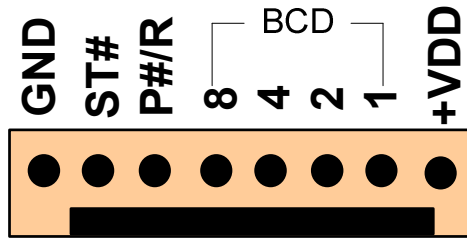
รูปที่2.1 แสดง PCB บอร์ดด้านบน



รูปที่ 2.2 แสดง PCB บอร์ดด้านหลัง

- 1. **LED Status** : เป็น LED แสดงสถานะการกด Key จะติดในลักษณะไฟวิ่งขณะจ่ายไฟให้กับบอร์ดในครั้งแรก และจะติดชั่วขณะเมื่อมีการกด Key เพื่อแสดงสถานะให้ผู้ใช้ทราบว่า Key ถูกกดแล้ว
- 2. **TOUCH PAD** : เป็นตำแหน่ง Key ที่ให้ผู้ใช้กด หรือสัมผัส โดยปกติจะมีแผ่นพลาสติกหนา 2 mm รองมาไว้แล้วไม่จำเป็นต้องสัมผัสกับลายปรินต์โดยตรงเพราะความไวจะสูง ส่วนตัวเลขที่แสดงบน PAD นั้นปกติจะไม่มี แต่ในรูปจะใส่กำกับไว้อ้างอิงเป็นชื่อคีย์เพื่อจะได้ระบุตำแหน่งของคีย์ให้ผู้ใช้เข้าใจเวลาอ้างถึงตำแหน่งคีย์ในคู่มือนี้ ซึ่งชื่อที่ตั้งนี้จะเป็นค่า Key Code ประจำ Key นั้นๆ ที่ส่งออกมาให้ผู้ใช้ โดยดูรายละเอียดค่า Key Code ของ Key ต่างได้จากตาราง KEY CODE
- 3. **HOLD1** : เป็นรูสำหรับยึดแผ่นพลาสติกรอง Touch Pad เข้ากับตัว PCB
- 4. **HOLD2** : เป็นรูสำหรับให้ผู้ใช้นำตัว ET-TOUCH PAD 4x4 ไปยึดติดกับกล่องหรืออื่นๆเพื่อใช้งาน
- 5. **Download FW** : ใช้สำหรับ Upgrade Firmware ให้กับ ET-TOUCH PAD 4x4 (ปกติจะต้องส่งมาให้ ETT ลงให้)
- 6. **Con.KeyCode Binary** : เป็น Connector ขนาด 8 PIN แสดงดังรูปที่ 2.3 จะทำหน้าที่ในการส่งค่า Key Code ของ Key ที่กดหรือปล่อย ในรูปแบบของ Binary BCD8421 และส่งค่าสถานะการกดและปล่อย Key ใน

รูป ของ Pulse และ Logic ออกมาให้ผู้ใช้นำไปใช้งาน พร้อมทั้งเป็นขั้วจ่ายไฟเลี้ยงให้กับ ET-TOUCH PAD 4x4 ด้วย



รูปที่ 2.3 แสดง Connector สำหรับอ่านค่า Key Code แบบ Binary BCD8421

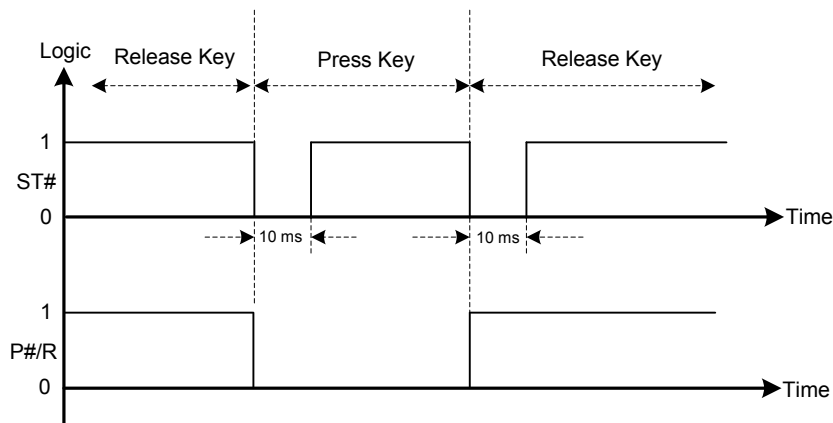
รายละเอียดของขาแต่ละ PIN

+VDD, GND = ไฟเลี้ยงบอร์ด VDC 3.3V (ใช้กับ MCU 3.3V) หรือ 5V (ใช้กับ MCU 5V)

BCD8421 = ขา ส่ง Key Code 4 bit โดย PIN หมายเลข 8 เป็นบิต MSB ค่า Key Code จะถูก ส่งออกมาทุกครั้งที่มีการกด Key และปล่อย Key โดยค่า Key Code ที่ส่งออกมา จะเป็นค่าของ Key ที่กดหรือ ปล่อยล่าสุด และค่าที่ส่งมานั้น ว่างกว่า จะมีการกด Key อื่นๆ ใหม่

P#/R = Press/Release บอกสถานะการกดหรือปล่อย Key โดยเมื่อกด 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

ST# = STROBE บอกสถานะการกดหรือปล่อย Key เช่นกัน แต่สัญญาณนี้จะส่งออกมาในลักษณะของ สัญญาณ Pulse คือในสถานะปกติสัญญาณนี้จะ เป็น Logic 1 ค้างไว้เสมอ เมื่อมีการกด Key สัญญาณ นี้ก็จะตกเป็น 0 ประมาณ 10 ms แล้วกลับขึ้นมาเป็น Logic 1 อัตโนมัติ จากนั้นเมื่อเราปล่อย Key สัญญาณ นี้ก็จะตกเป็น 0 ประมาณ 10 ms อีกครั้ง แล้วกลับขึ้นมาเป็น Logic 1 อัตโนมัติเช่นเดิม

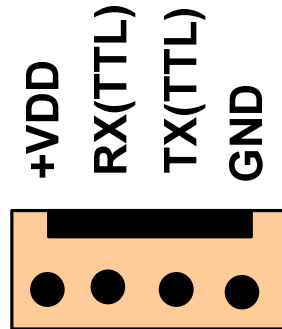


รูปที่ 2.4 แสดง Timing Diagram ของสัญญาณ ST# และ P#/R เมื่อ กด และ ปล่อย Key

สำหรับสัญญาณบอกสถานะการกดหรือปล่อย Key (P#/R และ ST#) นี้ เวลาใช้งานผู้ใช้อาจเลือกใช้สัญญาณใดสัญญาณหนึ่งหรือทั้ง 2 สัญญาณในการตรวจสอบสถานะการกดหรือปล่อย Key ก็ได้ แต่วัตถุประสงค์หลักที่ให้มีทั้ง 2 สัญญาณก็เพื่อเวลาใช้งาน ในลักษณะ Key ร่วมคือกด 2 Key พร้อมกันจะมีความจำเป็นต้องใช้ทั้ง 2 สัญญาณในการตรวจสอบสถานะการกดและปล่อย Key

- **7. Buzzer :** เป็น Buzzer จะส่งเสียงขณะจ่ายไฟให้กับบอร์ดในครั้งแรก และจะดัง “ปิ๊ป” เมื่อมีการกด Key ใดๆ

- 8. **Con.RS232-TTL Key Code ASCII** : : เป็น Connector RS232-TTL(UART) แสดงดังรูปที่ 2.5 จะทำหน้าที่ในการส่งค่าสถานะการ กด คือตัว “P” = 0x50 หรือสถานะการปล่อย คือตัว “R” = 0x52 ออกมาเป็น Byte แรกแล้วจึงตามด้วยค่า Key Code ของ Key ที่ถูกกดหรือปล่อย ออกมาเป็น Byte ที่ 2 ในรูปแบบของ ASCII CODE และปิดท้ายด้วย 0x0D เป็น Byte ที่ 3 เพื่อจบคำสั่ง สำหรับ Baud Rate ในการรับส่งจะถูกกำหนดไว้คงที่ ที่ 9600 bit/s



รูปที่ 2.5 แสดง Connector RS232-TTL(UART) สำหรับอ่านค่า Key Code แบบ ASCII CODE

รายละเอียดของขาแต่ละ PIN

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

RX(TTL) = ใช้รับ data จากผู้ใช้ ปกติไม่ได้ใช้งาน เพราะไม่มีการรับคำสั่งใดๆจากผู้ใช้

TX(TTL) = ใช้ส่งค่า สถานะการ กด/ปล่อย Key และค่า Key Code (ASCII Code) ให้กับผู้ใช้งาน โดยมีรูปแบบการส่ง data ตามตารางด้านล่าง

	ASCII CODE		
	Status Key (Byte 1)	Key Code (Byte 2)	End Byte (Byte 3)
PRESS	'P' (0x50)	'0-9', 'A-F'	0x0D
RELEASE	'R' (0x52)	'0-9', 'A-F'	0x0D

เช่น เมื่อกด Key 5 ค่าที่ส่งออกมาก็คือ P5 และ 0x0D หรือ 0x50, 0x35, 0x0D โดย 'P' คือค่าสถานะการกด ส่วน '5' คือค่า Key code และ 0x0D คือ Byte จบของคำสั่ง และเมื่อเราทำการปล่อย Key5 ก็จะมีการส่งค่า R5 และ 0x0D ออกมา โดย 'R' คือค่าสถานะการปล่อย ส่วน '5' คือ ค่า Key code ส่วน 0x0D คือ Byte จบของคำสั่ง เป็นต้น

ข้อควรระวัง การอ่าน Key Code ผ่านทางขั้วต่อ RS232-TTL นี้ย่ำลึ้มว่าขั้วต่อนี้ไม่มี Line driver Max232 ต่อไว้ ดังนั้นระดับสัญญาณ RS232 ที่จะมาต่ออ่านค่า Key Code ก็ต้องเป็นระดับ TTL เหมือนกัน ถ้า บอร์ด MCU ที่นำมาต่อมี Max232 อยู่ หรือต้องการอ่านค่า Key Code ไปแสดงที่ Hyper Terminal จาก ET-TOUCH PAD 4x4 โดยตรง ผู้ใช้จะต้องต่อ Line Driver Max232 กันระหว่างตัว ET-TOUCH PAD 4x4 กับตัวบอร์ด MCU หรือ RS232 ของ PC ไว้ด้วย

ในการใช้งานจริงนั้นผู้ใช้งานจะต้องเลือกอ่านค่า Key Code แบบใดแบบหนึ่งเท่านั้น คือ แบบ Binary BCD8421 หรือแบบ UART RS232-TTL เพื่อจะได้เลือกขั้ว Connector ต่อไปใช้งานได้อย่างถูกต้อง

ตาราง KEY CODE ของ ET-TOUCH PAD 4x4

KEY	FOR Binary MODE					FOR ASCII Mode (RS232 TTL)	
	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. การทำงานของ ET-TOUCH PAD 4x4

การทำงานโดยรวม เมื่อจ่ายไฟยังไม่มีกรกดคีย์ใดๆ ค่าสถานะของ PIN ต่างๆจะเป็นไปตามค่าเริ่มต้นที่กล่าวไว้ด้านล่าง ส่วนที่ขั้ว RS232 ก็จะไม่มีการส่ง data อะไรออกมา เมื่อมีการกดคีย์ใดคีย์หนึ่งค้างไว้คีย์อื่นๆที่เหลือจะถูกบล็อกไม่สามารถกดได้ (ยกเว้นคีย์ F) จนกว่าจะมีการปล่อยคีย์ที่กดเสียก่อนถึงจะกดคีย์อื่นๆได้ ในทางกลับกัน เมื่อกดคีย์ F จะยังสามารถกดคีย์อื่นๆที่เหลือได้ นั่นคือ นอกจากจะใช้งานคีย์ F เป็นคีย์ปกติเหมือนคีย์ อื่นๆได้แล้ว เรายังสามารถใช้คีย์ F เป็นคีย์ร่วมกับคีย์อื่นๆที่เหลือได้ด้วย ทุกครั้งที่มีการกดคีย์ใดๆก็ตามจะมีเสียง บี๊ด และ LED ประจำคีย์นั้นๆจะติดขึ้นมาเป็นเวลาชั่วขณะหนึ่งและหายไป เพื่อแสดงให้ผู้ใช้งานรู้ว่าคีย์นั้นถูกกด

เมื่อคีย์ใดถูกกดหรือปล่อยสถานะของ Key Code ทั้งของ Binary และ ASCII รวมทั้ง สัญญาณ ST# และ P#/R จะถูก Update ตามสถานะของ Key ที่ถูกกดและปล่อยล่าสุด นั่นคือ ไม่ว่าจะกดคีย์หรือปล่อยคีย์ค่า key Code จะถูกส่งออกมาเสมอ รวมทั้งสถานะของสัญญาณ ST# และ P#/R ก็จะมีการเปลี่ยนแปลงเสมอเช่นกัน ซึ่งดูสถานะการส่งสัญญาณ และการส่งค่า key code ได้ตามรูปแบบการทำงานด้านล่าง

1. การอ่าน Key Code แบบ Binary BCD8421

- ค่าเริ่มต้น(Default) = เมื่อจ่ายไฟเลี้ยงให้กับบอร์ด LED ของ key จะติด วังจาก key1 ไปจนเป็นรูปก้นหอย พร้อมกับส่งเสียงป๊อป ในขณะที่ LED วัง และที่ Connector 8PIN จะถูก Set ดังนี้

PIN ST# = 1 ; PIN P#/R = 1

PIN 8 = 0 ; PIN 4 = 0 ; PIN2 = 0 ; PIN1 = 0

- กด(Press) Key = เมื่อผู้ใช้กด Key ใดๆหนึ่ง Key ที่ Connector 8 PIN ก็จะมีการเปลี่ยนแปลงค่าสถานะต่างๆ โดยเรียงลำดับดังนี้

- 1) PIN ST# (เป็นสัญญาณ Strobe) จะเปลี่ยนสถานะ logic จาก 1 ไปเป็น 0 เป็นเวลา 10 ms จากนั้นจะกลับมาเป็น 1 อัตโนมัติ ตามรูปที่2.4
- 2) PIN P#/R (เป็นสัญญาณการ กด หรือ ปล่อย key) จะเปลี่ยนสถานะ logic จาก 1 ไปเป็น 0 ค้างไว้ตลอดที่คีย์ยัง ถูกกดอยู่ ตามรูปที่2.4
- 3) PIN BCD8,4,2,1 จะมีการเปลี่ยนสถานะตามค่า Key Code ของ key ที่กดล่าสุดออกมาค้างไว้

ในกรณีที่ใช้ Key F ทำหน้าที่เป็นคีย์ร่วมเพื่อกร่วมกับ key อื่นๆ เริ่มต้นเมื่อกด Key F สถานะการเปลี่ยนแปลงของ PIN ต่างๆ ที่เกิดขึ้น ก็จะเป็นไปตาม Step1-3 ที่กล่าวไว้ข้างต้น จากนั้นเมื่อกด Key ที่ 2 เป็น Key ใดๆก็ตามในขณะที่ Key F ยังคงกดค้างอยู่ ที่ PIN ST# ก็จะเป็นไปตาม Step1 เช่นเดิม ส่วน PIN P#/R ก็จะยังคงเป็น Logic 0 ค้างอยู่ไม่มีการเปลี่ยนแปลง ดังนั้นเวลาตรวจสอบการกด Key ที่ 2 ให้ตรวจสอบจากการเปลี่ยนแปลงของสัญญาณ ST# เมื่อ Key ที่ 2 ถูกกด ค่า Key Code ที่ PIN BCD8421 ก็จะเปลี่ยนไปตามค่าของ Key ที่มีการกดล่าสุด

- ปล่อย (Release)Key = เมื่อผู้ใช้ปล่อย Key ใดๆที่กดอยู่ ที่ Connector 8 PIN ก็จะมีการเปลี่ยนแปลงค่าสถานะต่างๆโดยเรียงลำดับดังนี้

- 1) PIN ST# (เป็นสัญญาณ Strobe) จะเปลี่ยนสถานะ logic จาก 1 ไปเป็น 0 เป็นเวลา 10 ms จากนั้นจะกลับมาเป็น 1 อัตโนมัติ ตามรูปที่2.4 ซึ่งเหมือนกับตอนกด Key
- 2) PIN P#/R (เป็นสัญญาณการ กด หรือ ปล่อย Key) จะเปลี่ยนสถานะ logic จาก 0 ไปเป็น 1 ค้างไว้ตลอดที่ยังไม่มีการกดคีย์ใหม่ ตามรูปที่2.4
- 3) PIN BCD8,4,2,1 จะมีการเปลี่ยนสถานะตามค่า Key Code ของ key ที่ปล่อยล่าสุดออกมาค้างไว้

ในกรณีที่ใช้ Key F ทำหน้าที่เป็นคีย์ร่วม เพื่อกร่วมกับ key อื่นๆ หลังจากที่มีการกด Key F และKey ร่วมอื่นๆไปแล้ว เมื่อปล่อยคีย์ใดก็ได้เป็นคีย์แรก สถานะการเปลี่ยนแปลงของ PIN ST# ก็จะเป็นไปตาม Step1 เช่นเดิม ส่วน PIN P#/R ก็จะยังคงมีสถานะเป็น Logic 0 ค้างอยู่ไม่มีการเปลี่ยนแปลงเนื่องจากยังมี Key ถูกกดอยู่อีก 1 Key ส่วน PIN BCD8421 ก็จะมีการเปลี่ยนแปลงค่า Key Code เป็นไปตาม Key ที่ปล่อยขณะนั้น ต่อมาเมื่อปล่อย Key ที่ 2 สถานะการเปลี่ยนแปลงของ PIN ต่างๆ ก็จะเป็นไปตาม Step1-3 ตามที่กล่าวไปข้างต้น ดังนั้นจะเห็นว่า ในการตรวจสอบสถานะการปล่อย Key เราจะอาศัยการตรวจสอบ สัญญาณ ST# และตรวจสอบว่า Key ใดถูกปล่อยโดยการอ่านค่า Key Code ส่วนสัญญาณ P#/R จะใช้ตรวจสอบว่าไม่มีการกด Key ใดๆแล้ว

2. การอ่าน Key Code แบบ ASCII

การอ่าน Key Code แบบ ASCII นี้ อุปกรณ์ที่นำมาต่อที่ Connector RS232 ของ ET-TOUCH PAD เพื่อทำการอ่านค่า จะต้องมียกระดับแรงดันเป็น TTL(5V) ถ้าไม่เช่นนั้นก็ต้องใช้ Max232 มาต่อเป็น Line Driver เพื่อให้สามารถใช้อ่านค่าผ่านทาง Serial Port ของ PC ได้ ส่วน Baud Rate ในการรับส่งข้อมูลจะถูกกำหนดตายตัวไว้ที่ 9600 bit/s

- ค่าเริ่มต้น(Default) = เมื่อจ่ายไฟเลี้ยงให้กับบอร์ด LED ของ key จะติด วังจาก key1 ไปจนเป็นรูปกันหอย พร้อมกับส่งเสียงป๊ด ในขณะที่ LED วัง และจะยังไม่มีการส่งข้อมูลใดๆออกมาที่ Port RS232

- กด(Press) Key = เมื่อผู้ใช้กด Key ใดๆหนึ่ง Key ตัว ET-TOUCH PAD ก็จะส่ง Data ออกมาที่ขั้วต่อ RS232 จำนวน 3

Byte คือ ASCII 'P' จะถูกส่งออกมาเป็น Byte แรก เพื่อบอกสถานะการกด Key และ ASCII Code ของ Key ที่กดจะถูกส่งออกมาเป็น Byte ที่ 2 และจบด้วย 0x0D เป็น Byte สุดท้าย ดังนั้น เมื่อผู้ใช้ทำการอ่านค่า Key Code ผู้ใช้ก็จะทราบได้ว่า Key Code ที่อ่านได้เป็น Key Code จากการกด และทราบได้ว่า key ที่กดเป็น Key อะไรด้วย

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

ตารางรูปแบบการส่ง data เมื่อมีการกด Key

ในกรณีที่ผู้ใช้ Key F ทำหน้าที่เป็นคีย์ร่วมเพื่อกดร่วมกับ key อื่นๆ เริ่มต้นเมื่อกด 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 จากนั้นเมื่อกด Key ที่ 2 เป็น Key ใดๆก็ตามในขณะที่ Key F ยังคงกดค้างอยู่ ET-TOUCH PAD ก็จะส่ง ASCII 'P'(0x50) ออกมาเป็น Byte แรกเช่นเดิม และ ASCII Key Code ของ Key ที่กดล่าสุดก็จะถูกส่งออกมาเป็น Byte ที่ 2 และจบด้วย 0x0D เป็น Byte สุดท้ายเช่นกัน นี่คือการทำงานขณะที่กด Key เมื่อใช้งาน Key F เป็น Key ร่วม ดังนั้นเวลาที่เขียนโปรแกรมอันดับแรกเราต้องตรวจสอบว่า Key ที่กดเป็น Key F ก่อนแล้วจึงตรวจสอบ Key ที่ 2 ที่กดร่วม ถ้า Key แรกที่กดไม่ใช่ Key F ก็ให้ถือว่า เป็นการกด Key เดี่ยว ไม่ใช่ Key ร่วม

- ปล่อย (Release)Key = เมื่อผู้ใช้ปล่อย Key ใดๆ Key หนึ่งที่กดอยู่ ตัว ET-TOUCH PAD ก็จะส่ง Data ออกมาที่ขั้วต่อ-

RS232 จำนวน 3 Byte คือ ASCII 'R' จะถูกส่งออกมาเป็น Byte แรก เพื่อบอกสถานะการปล่อย Key และ ASCII Code ของ Key ที่ปล่อยจะถูกส่งออกมาเป็น Byte ที่ 2 และจบด้วย 0x0D เป็น Byte สุดท้าย ดังนั้น เมื่อผู้ใช้ทำการอ่านค่า Key Code ผู้ใช้ก็จะทราบได้ว่า Key Code ที่อ่านได้เป็น Key Code จากการปล่อย Key และทราบได้ว่า key ที่ปล่อย เป็น Key อะไรด้วย

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

ตารางรูปแบบการส่ง data เมื่อมีการปล่อย Key

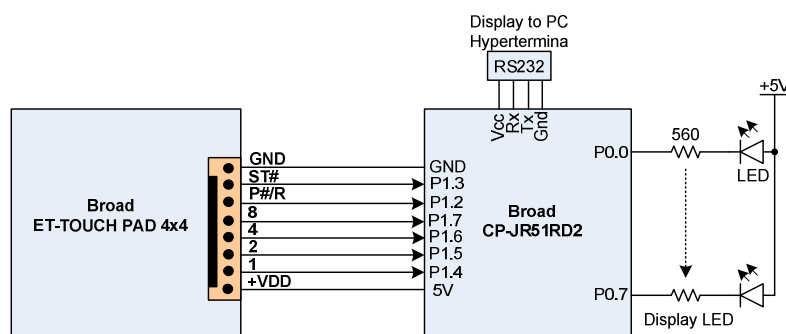
ในกรณีที่ผู้ใช้ Key F ทำหน้าที่เป็นคีย์ร่วมเพื่อกดร่วมกับ key อื่นๆ หลังจากที่มีการกด Key F และ Key ร่วมอื่นๆไปแล้ว เมื่อปล่อยคีย์ใดก็ได้เป็นคีย์แรก ตัว ET-TOUCH PAD ก็จะส่ง ASCII 'R'(0x52) ออกมาเป็น Byte แรก และ ASCII Key Code ของ Key ที่ปล่อยก็จะถูกส่งออกมาเป็น Byte ที่ 2 ตามด้วย 0x0D เป็น byte สุดท้าย จากนั้นเมื่อทำการปล่อยคีย์ที่ 2 ASCII 'R'(0x52) ก็จะถูกส่งออกมาเป็น Byte แรก เช่นเดิม และ ASCII Key Code ของ Key ที่ปล่อยล่าสุดก็จะถูกส่งออกมาเป็น Byte ที่ 2 และจบด้วย 0x0D เป็น Byte สุดท้าย เช่นเดิม นี่คือการทำงานขณะปล่อย Key เมื่อใช้งาน Key F เป็น Key ร่วม ดังนั้นเวลาที่เขียนโปรแกรมในส่วนของการปล่อย Key (มีการกด Key F+ Key อื่นๆอยู่) เราต้องตรวจสอบว่า Key ที่ปล่อยเป็น Key F หรือไม่ ถ้าใช่ก็ให้ออกจาก Loop ตรวจสอบ Key ร่วมได้เลย แต่ถ้าเป็น Key อื่นๆ ที่ไม่ใช่ Key F ก็ให้วนรออ่านค่า Key อื่นๆ ที่จะกดร่วมกับ Key F ต่อไป

4. ตัวอย่างโปรแกรมและการต่อวงจรอ่านค่า Key Code ด้วย MCU

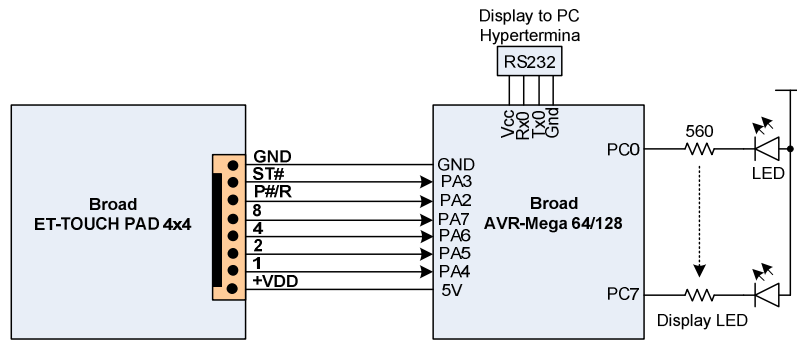
สำหรับตัวอย่างการอ่านค่า Key Code ทั้งแบบ Binary BCD8421 และ ASCII Code จะสามารถรองรับกับ MCU 3 ตระกูลคือ ARM7 LPC2138 โดยใช้ C-Keil Compiler , AVR MEGA 64/128 โดยใช้ C-WIN AVR และ MCS51-AT89C51ED2 โดยใช้ C-Keil Compiler โดยแต่ละ MCU จะมีตัวอย่างอยู่ 2 การอ่านคือ การอ่านค่า Key Code แบบ Binary BCD8421 และแบบ ASCII Code โดยมีรายละเอียดการทำงานของโปรแกรมดังนี้ (Source Code อยู่ใน CD)

ตัวอย่าง Read Key แบบ Binary BCD8421

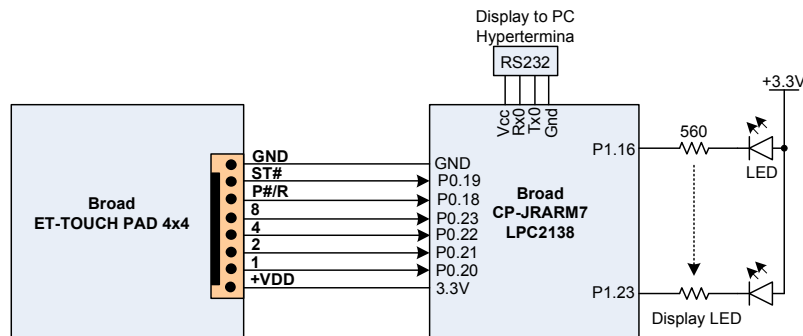
ตัวอย่าง *EX1_Read_1Key_BCD*: สำหรับตัวอย่างนี้ จะเป็นการอ่านค่า Key Code แบบ Binary โดยใช้งาน Key แต่ละ Key แบบ Key เดียว เมื่อผู้ใช้ทำการ Touch Key ตัว MCU ก็จะทำการตรวจสอบสัญญาณ ST# จากนั้นก็จะทำการอ่านค่า Key Code BCD8421 มาเก็บไว้ แล้วตรวจสอบสัญญาณ P#/R เพื่อตรวจสอบว่า Key ยังถูกกดอยู่จริง MCU ถึงจะทำการส่งค่า Key Code ออกทาง Port ที่ต่อ LED และพิมพ์ค่า Key Code ที่อ่านได้ออกแสดงที่ Hyper Terminal บน PC อีกทางหนึ่ง เช่น ถ้ากด Key2 ค่าที่ MCU ส่งออกมาแสดงก็คือ 0x02 นี่เป็นการทำงานโดยรวมในตัวอย่างที่ 1 ของแต่ละ MCU สำหรับตัวอย่างวงจรการต่อที่รองรับตัวอย่างที่ 1 นี้แสดงดังรูปด้านล่าง



รูปที่ 4.1 การต่อวงจรสำหรับตัวอย่างที่ 1 ของบอร์ด CP-JR51RD2 เข้ากับ ET-Touch Pad 4x4



รูปที่4.2 การต่อวงจรสำหรับตัวอย่างที่1 ของบอร์ด AVR MEGA64/128 เข้ากับ ET-Touch Pad 4x4



รูปที่4.3 การต่อวงจรสำหรับตัวอย่างที่1 ของบอร์ด CP-JR ARM7 LPC2138 เข้ากับ ET-Touch Pad 4x4

ตัวอย่าง EX2_Read_2Key_BCD: สำหรับตัวอย่างที่2 จะเป็น การอ่านค่า Key Code แบบ Binary โดยใช้งาน Key F เป็น Key ร่วมกับ Key อื่นๆด้วย เมื่อผู้ใช้ทำการ Touch Key ตัว MCU ก็จะทำการตรวจสอบสัญญาณ ST# หลังจากตรวจสอบพบสัญญาณ ST# แล้วจะต้องรอให้สัญญาณนี้กลับเป็น '1' ก่อนที่จะไปทำคำสั่งต่อไป ไม่เช่นนั้นอาจทำให้โปรแกรมทำงานไม่ตรงกับที่เราต้องการได้ เมื่อ ST# เป็น '1' แล้ว ก็จะทำการอ่านค่า Key Code BCD8421 มาเก็บไว้ จากนั้นก็ตรวจสอบสัญญาณ P#/R ว่ายังคงเป็น '0' อยู่หรือไม่เพื่อให้แน่ใจว่าสถานะ Key ยังคงถูกกดอยู่ จากนั้นก็นำ Key Code มาตรวจสอบว่า Key ที่กดใช่ Key F หรือไม่ ถ้าไม่ใช่ แสดงว่าเป็นการกด Key เดียว ก็ให้ไปทำโปรแกรมตอบสนองการกด Key เดียว ซึ่งก็คือให้ MCU ส่งค่า Key Code ออกทาง Port ที่ต่อ LED และพิมพ์ค่า Key Code ที่อ่านได้ออก แสดงที่ Hyper Terminal แต่ถ้า Key ที่กดเป็น Key F ก็จะต้องรอตรวจสอบค่า Key ร่วมที่จะกดต่อไป ด้วยการ ตรวจสอบสัญญาณ ST# เมื่อพบก็ให้ทำการอ่านค่า Key Code ของ Key ร่วมที่กด และตรวจสอบสัญญาณ P#/R เพื่อยืนยันว่า Key ร่วมถูกกดอยู่ จากนั้นก็ไปทำโปรแกรมตอบสนองการกด Key ร่วม ซึ่งก็คือให้ MCU ส่งค่า Key Code ออกทาง Port ที่ต่อ LED และพิมพ์ค่า Key Code ที่อ่านได้ออก แสดงที่ Hyper Terminal ต่อมาก็จะวนมาตรวจสอบว่า มีการปล่อย Key F หรือยัง ถ้ายังก็ให้กลับไปอ่าน Key ร่วมอื่นๆต่อไป

สำหรับในตัวอย่างที่ 2 นี้ วงจรการต่อใช้งานจะเหมือนกับตัวอย่างที่1

ตัวอย่าง EX3_Application_2Key_BCD: สำหรับในตัวอย่างที่ 3 นี้จะเป็นการประยุกต์ใช้งานโดยใช้ Key F เป็น Key ร่วมกับ Key อื่นๆด้วย ในขณะที่ยังไม่มีกรกด Key จะแสดงข้อความว่า “Not! Touch Key” ออกที่จอ LCD เมื่อมีการกด Key เดียวก็จะแสดงข้อความ “Touch Key = [ชื่อ Key ที่กด]” และถ้ากด Key F และ Key ร่วม อื่นๆ ก็จะแสดงข้อความ “Touch Key = FUN+[ชื่อ Key ที่กด]” เป็นต้น การทำงานของโปรแกรมนั้นจะมีรูปแบบเหมือนกับตัวอย่างที่ 2 จะต่างกันตรงส่วนของโปรแกรมตอบสนองการกด Key เท่านั้น

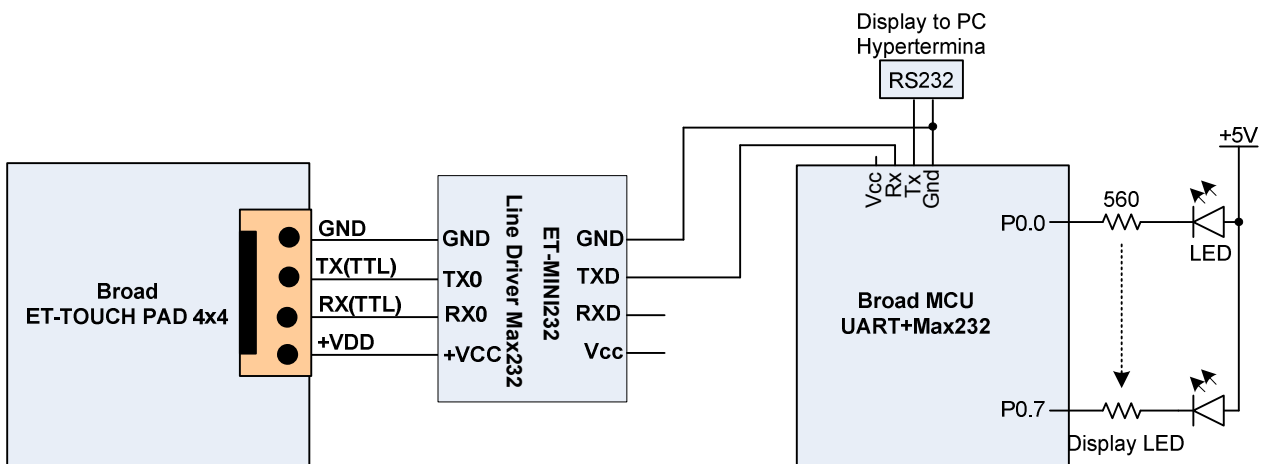
สำหรับวงจรการต่อใช้งานในส่วนของ ET-TOUCH PAD กับ MCU จะเหมือนกับตัวอย่างที่ 1 ส่วนการต่อจอ LCD กับตัว MCU ได้แสดงไว้ใน Source Code แล้ว

ตัวอย่าง Read Key แบบ ASCII CODE

การอ่านค่าแบบ ASCII Code ค่า Key Code ที่อ่านได้ในแต่ละ Key ที่กดจะเป็นไปตามตาราง KEY CODE ในช่อง ASCII MODE โดยสามารถอ่านค่าได้จากขั้วต่อ RS232(TTL) ของ ET-TOUCH PAD 4x4 ซึ่งจะใช้ BAUD RATE ในการรับส่ง อยู่ที่ 9600 bit/s เวลาต่อใช้งานสามารถต่อเข้ากับ TX,RX ของ MCU ได้โดยตรง แต่ถ้า MCU ที่นำมาใช้นั้นมี Line Driver Max232 ต่อไว้อยู่ ผู้ใช้จะต้องต่อ Line Driver Max232 ให้กับทาง ET-TOUCH PAD ด้วย เพื่อให้ระดับแรงดัน RS232 อยู่ในระดับเดียวกัน

ตัวอย่าง EX4_Read_1Key_ASCII: สำหรับตัวอย่างนี้ จะเป็นการอ่านค่า Key Code แบบ ASCII โดยใช้งาน Key แต่ละ Key แบบ Key เดียว และใช้การวน Loop ในการรับข้อมูลที่ส่งมาจาก ET-TOUCH PAD เมื่อผู้ใช้ทำการ Touch Key ตัว MCU ก็จะรับข้อมูล Byte แรกมาตรวจสอบว่าใช่ ASCII 'P' หรือไม่ ถ้าไม่ใช่ก็จะกลับไปวนรอรับข้อมูลอีก แต่ถ้าใช่ แสดงว่ามีการกด Key ก็จะมารอรับข้อมูลในส่วนที่เป็นค่า Key Code ของ Key ที่กด แล้วนำค่า Key Code ที่อ่านได้ไปแสดงที่ Hyper Terminal และที่ PORT LED ที่ต่อกับ MCU อยู่ เช่น ถ้ามีการกด Key 7 ค่าที่แสดงบน Hyper Terminal ก็จะแสดง ASCII เลข 7 ส่วนที่ LED ก็จะแสดงค่าเป็น 0x37 เป็นต้น

สำหรับตัวอย่างวงจรการต่อที่รองรับตัวอย่างที่4 นี้แสดงดังรูปด้านล่าง



รูปที่4.4 การต่อวงจรสำหรับตัวอย่างที่4 ของบอร์ด MCU เบอร์ต่างๆ เข้ากับ ET-Touch Pad 4x4

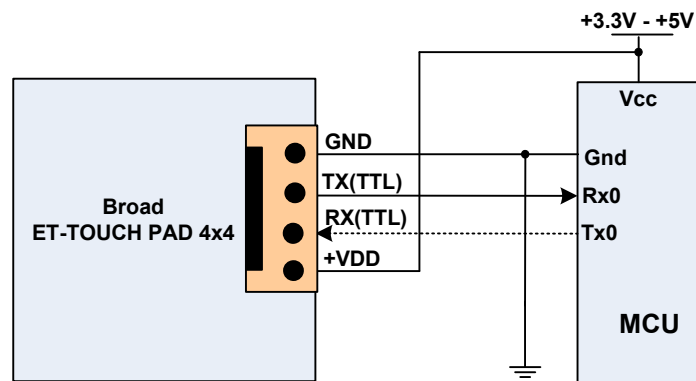
ในกรณีวงจรการต่อในรูปที่4.4 จะใช้สำหรับทดสอบกับตัวอย่างที่เขียนมาให้ เมื่อบอร์ด MCU มี UART เพียงช่องเดียวและ ต่อ MAX232 เป็น Line Driver ไว้แล้วดังนั้นในฝั่งของ ET-TOUCH PAD จึงต้องต่อ Max 232 อีกชุด คั่นไว้ด้วย

ตัวอย่าง EX5_Read_1Key_ASCII_INT: สำหรับในตัวอย่างนี้จะเหมือนกับตัวอย่างที่ 4 เพียงแต่ในการรับข้อมูลจะไม่ใช้การวน Loop ในการรับข้อมูล แต่จะใช้การตรวจสอบสัญญาณ Interrupt Rx คือเมื่อมีสัญญาณ Interrupt เกิดขึ้นโปรแกรมก็จะกระโดดไปรับข้อมูลมาตรวจสอบทันที ในส่วนของวงจรการต่อทดสอบ ก็จะเหมือนกับตัวอย่างที่ 4 โดยจะใช้ PIN ไหนต่อกับอะไรในแต่ละ MCU ได้ระบุไว้ใน Source Code แล้ว

ตัวอย่าง *EX6_Read_2Key_ASCII_INT*: ในตัวอย่างนี้ จะเป็นการอ่านค่า Key Code แบบ Binary โดยใช้งาน Key F เป็น Key ร่วมกับ Key อื่นด้วยและใช้สัญญาณ Interrupt เป็นตัวกำหนดจังหวะการรับข้อมูล เมื่อผู้ใช้ทำการ Touch Key โปรแกรมก็จะกระโดดไปรับข้อมูลมาเก็บไว้จนครบ 3 Byte จากนั้นก็จะตรวจสอบว่า Byte สุดท้ายใช่ 0x0D หรือไม่ ถ้าใช่แสดงว่ารับข้อมูลมาครบและถูกต้องแล้ว จากนั้นก็จะกลับมามตรวจสอบข้อมูลที่รับมา Byte แรก ว่าใช่ ASCII 'P' หรือไม่ถ้าใช่แสดงว่าเป็นการกด Key จากนั้นก็จะทำการตรวจสอบ data ที่รับมาใน Byte ที่2 ซึ่งก็คือค่า ASCII Key Code ของ Key ที่กด ว่ามีค่าเท่ากับ ASCII CODE 'F' หรือไม่ ถ้าไม่ใช่ก็แสดงว่าเป็นการกด Key เดียว ก็ให้ไปทำโปรแกรมตอบสนองการกด Key เดียว ซึ่งก็คือให้ MCU ส่งค่า Key Code นั้นออกทาง Port ที่ต่อ LED และพิมพ์ค่า Key Code ที่อ่านได้ออก แสดงที่ Hyper Terminal แต่ถ้า Key ที่กดเป็น Key F โปรแกรมก็จะต้องวนรอรับค่า คีย์ที่สองที่จะกดร่วมกับ Key F จนกว่าจะมีการปล่อย Key F ถึงจะมีการออกจาก Loop ได้

ส่วนวงจรการต่อเพื่อทดสอบโปรแกรมก็จะเหมือนกับรูปที่ 4.4 ของตัวอย่างที่4

สำหรับการต่อ ET-TOUCH PAD 4x4 เพื่ออ่านค่า ASCII Key Code ผ่านทาง RS232(TTL) เข้ากับ PORT UART ของ MCU โดยตรงนั้น สามารถต่อได้ดังรูปด้านล่าง โดยปกติตัว ET-TOUCH PAD จะส่ง Data ออกทาง RS232(TTL) เพียงอย่างเดียว ไม่ได้มีการรับข้อมูลจากภายนอกเข้ามาใช้งาน ดังนั้นสำหรับขา RX(TTL) นั้น อาจจะต่อหรือไม่ต่อก็ได้



รูปที่4.5 การต่อบอร์ด ET-Touch Pad 4x4 RS232(TTL) เข้ากับ PORT UART ของ MCU โดยตรง

Option เพิ่มเติม

สำหรับบอร์ดที่ลง Firmware V2 ไว้ ในส่วนของ RS232 นั้น นอกจากจะส่งค่า ASCII Key Code ได้เหมือนเดิมแล้ว เรายังเพิ่มความสามารถให้รองรับ ASCII Command จากผู้ใช้ เพื่อควบคุมการทำงานของ LED และ Buzzer ที่อยู่บนบอร์ดได้ด้วย วัตถุประสงค์ก็เพื่อให้ผู้ใช้สามารถใช้งานในส่วนของ LED และ Buzzer ทำหน้าที่เป็น Alarm ได้ในขณะที่ไม่มีการกด Key และเมื่อกด Key ตัว Key ก็จะกลับมาทำการส่งค่า Key Code ให้กับผู้ใช้ได้เหมือนเดิมโดยอัตโนมัติ

ดังนั้น จากรูปที่ 4.5 ถ้าผู้ใช้ต้องการใช้งานในส่วนของ Option เพิ่มเติมนี้ ก็จะต้องต่อขา RX(TTL) เข้ากับขา Tx0 ของบอร์ดที่จะนำมาใช้ควบคุมด้วย สำหรับคำสั่งการใช้งานสามารถดูได้จากหัวข้อด้านล่าง

COMMAND สำหรับ CONTROL LED&BUZZER

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 ก็คือค่า Enter จะไม่มีสัญลักษณ์ให้เรียกใช้งาน ดังนั้นจะแทนด้วย ASCII Code = 0x0D

COMMAND ‘S0’ (SOUND-OFF)

เป็นคำสั่ง Off เสียง ที่เกิดจากการส่งคำสั่ง ‘S1’ ซึ่งเสียงจะดังต่อเนื่อง ให้ใช้คำสั่งนี้หยุดเสียงนั้น มีรูปแบบคำสั่งดังนี้

Start	Command	Mark#1	Mark#2	END
Byte1	Byte2-3	Byte4	Byte5	Byte6
#	S0	=	I	Enter (0x0D)
Respond Command จาก Board				
*	OK			

Ex. ส่งคำสั่งหยุดเสียง

```

Main()
{
    char enter = 0x0D ;
    printf("#S0=I",enter) ;    //Sent Command Off-Sound
}

```

COMMAND ‘S1’ (SOUND-ON : Continuous)

เป็นคำสั่ง On เสียงให้ดังตลอดเวลา จะหยุดก็ต่อเมื่อ มีการส่งคำสั่ง “S0” มา Stop เสียง หรือ มีการ Touch Key มีรูปแบบคำสั่งดังนี้

Start	Command	Mark#1	Mark#2	END
Byte1	Byte2-3	Byte4	Byte5	Byte6
#	S1	=	I	Enter (0x0D)
Respond Command จาก Board				
*	OK			

Ex. ส่งคำสั่ง ให้เสียงดังยาวตลอดเวลา

```

Main()
{
    char enter = 0x0D ;
    printf("#S1=I",enter) ;    //Sent Command On-Sound Continuous
}

```

COMMAND ‘S2’ (SOUND-ON : One-Shot)

เป็นคำสั่ง On เสียงให้ดังตามช่วงระยะเวลาที่กำหนด โดยสามารถกำหนดช่วงเวลาที่ได้ตั้งแต่ 0-9 เมื่อครบระยะเวลาที่กำหนดเสียงก็จะ Off อัตโนมัติ การใช้คำสั่งนี้เสียงจะดัง สั้น-ยาว ต่างกันไป มีรูปแบบคำสั่งดังนี้

Start	Command	Mark#1	Sound On Time	END
Byte1	Byte2-3	Byte4	Byte5	Byte6
#	S2	=	‘0-9’	Enter (0x0D)
Respond Command จาก Board				
*	OK			

Sound On Time : กำหนดความสั้นยาวของเสียงได้ตั้งแต่ ‘0’ – ‘9’ ระดับ

Ex. ส่งคำสั่ง ให้เสียงดัง ความยาวที่ระดับ 5 แล้วหยุด

```

Main()
{
    char enter = 0x0D ;
    printf("#S2=5",enter) ;    //Sent Command On-Sound Level 5
}

```

COMMAND 'L0' (OFF-LED)

เป็นคำสั่ง OFF-LED ที่เกิดจากการ ON-LED ด้วยคำสั่ง 'L1' โดยสามารถเลือกรูปแบบการ OFF-LED ได้ 5 รูปแบบ มีรูปแบบคำสั่งดังนี้

Start	Command	Mark#1	Format OFF-LED	END
Byte1	Byte2-3	Byte4	Byte5	Byte6
#	L0	=	'0-4'	Enter (0x0D)
Respond Command จาก Board				
*	OK			

Format OFF-LED : กำหนดรูปแบบการ Off ของ LED โดย

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

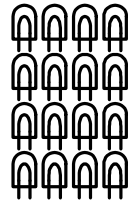
Ex. ส่งคำสั่ง ให้ LED ทุกดวง Off ทั้งหมด

```

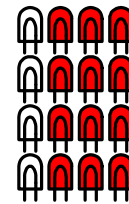
Main()
{
    char enter = 0x0D ;
    printf("#L0=0",enter); //Sent Command Off-LED ALL
}

```

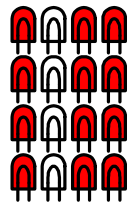
Format = '0'



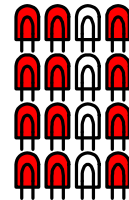
Format = '1'



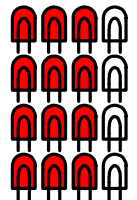
Format = '2'



Format = '3'



Format = '4'

**COMMAND 'L1' (ON-LED)**

เป็นคำสั่ง ON-LED โดยสามารถเลือกรูปแบบการ ON-LED ได้ 5 รูปแบบ มีรูปแบบคำสั่งดังนี้

Start	Command	Mark#1	Format ON-LED	END
Byte1	Byte2-3	Byte4	Byte5	Byte6
#	L1	=	'0-4'	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

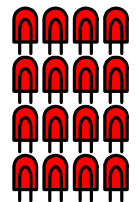
Ex. ส่งคำสั่ง ให้ LED ทุกดวง ON ทั้งหมด

```

Main()
{
    char enter = 0x0D ;
    printf("#L1=0",enter); //Sent Command On-LED ALL
}

```

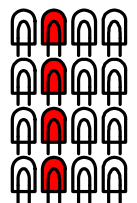
Format = '0'



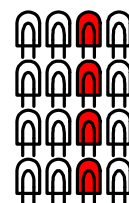
Format = '1'



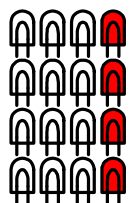
Format = '2'



Format = '3'

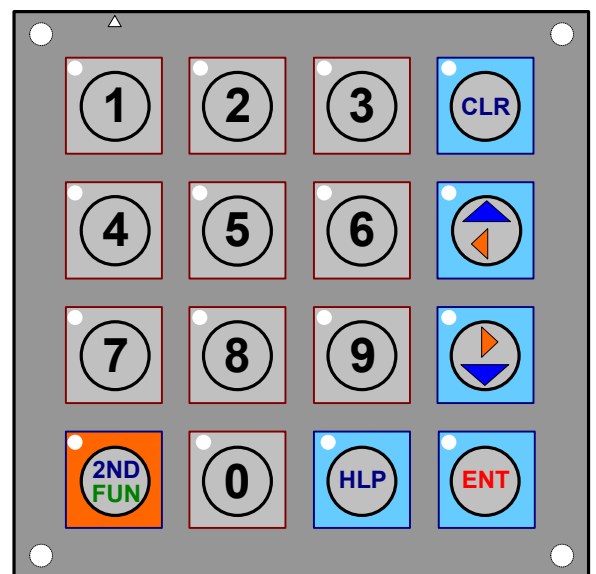
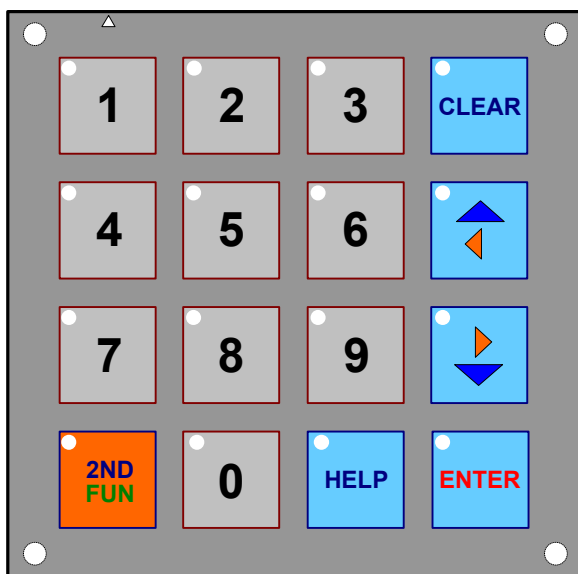
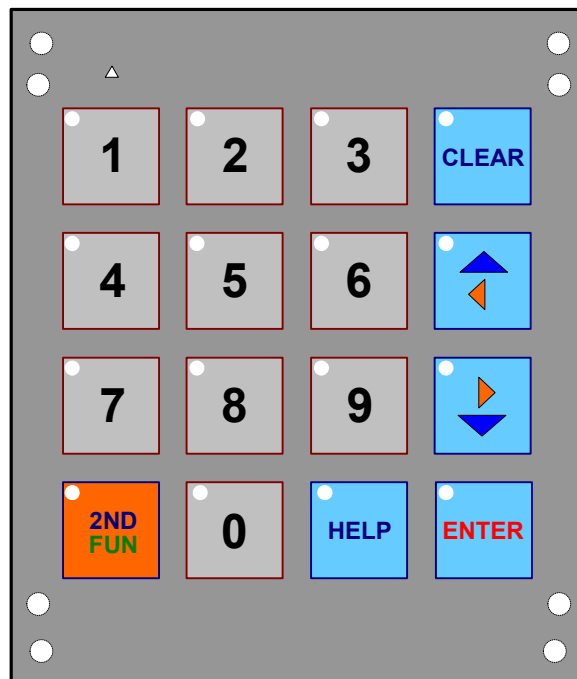


Format = '4'

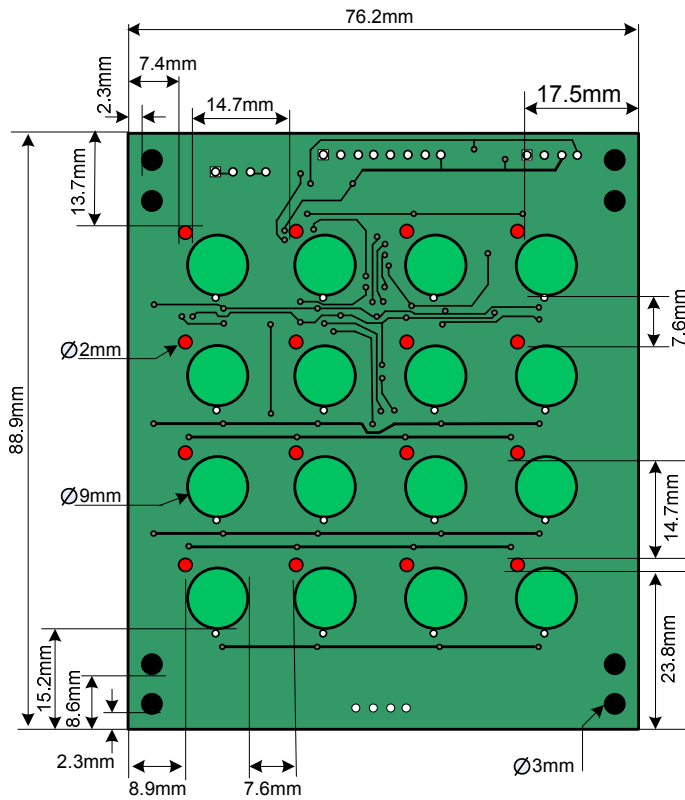


Note ในการส่งคำสั่งแบบต่อเนื่องระหว่างคำสั่งทุกคำสั่งควรมีการตรวจสอบ Respond Command ของคำสั่งที่ส่งออกไปเสมอ หรือจะใช้เวลา delay ด้วยค่าเวลาที่เหมาะสม ระหว่างคำสั่งแทนก็ได้ เพื่อป้องกันคำสั่งที่ส่งออกไปทับซ้อนกัน ซึ่งจะทำให้การส่งคำสั่งนั้นผิดพลาดได้

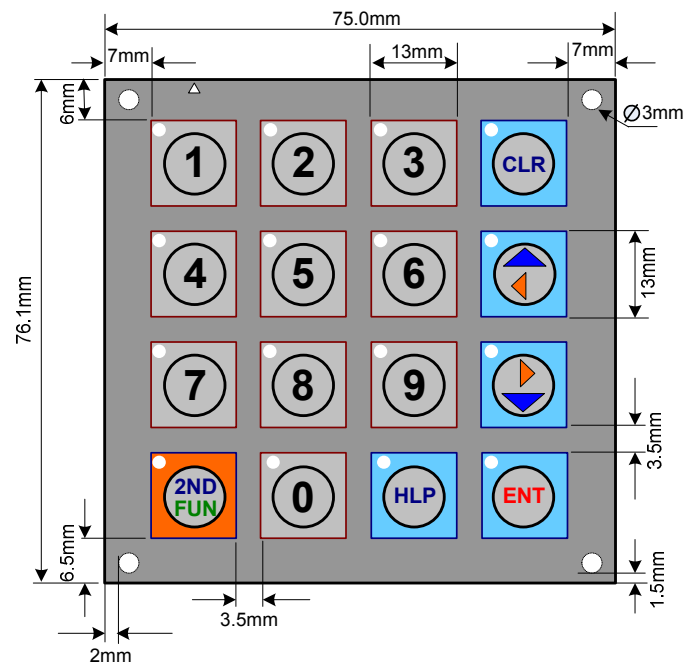
ตัวอย่าง KEY Label ขนาดเท่าของจริง



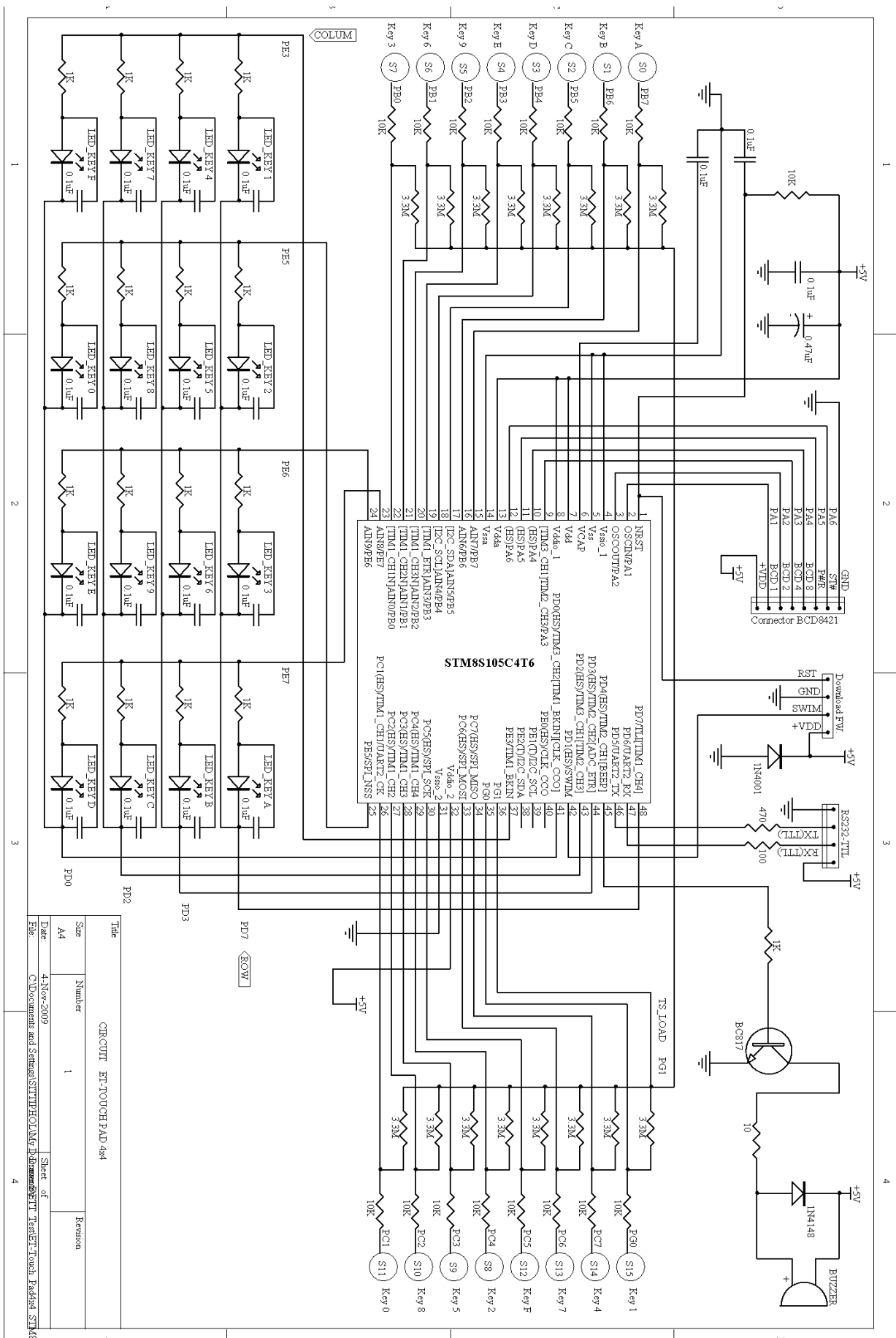
ขนาด PCB LAY OUT และ SAMPLE KEY LABEL



PCB LAY OUT(ด้านบน)



SAMPLE KEY LABEL



ET-TOUCH PAD 4x4