ET-TOUCH PAD 4x4

1. Features of Board ET-TOUCH PAD 4x4

- Be Capacitive Sensing Touch Key with 16 KEY 4x4
- Use +3.3VDC or +5VDC Power Supply
- Display status of pressing keys by voice and LED that is on that key
- When start supplying power into ET-TOUCH PAD 4x4, there is voice and LED is ON in the format of running light to notify user to know that the board is ready to operate.
- Have 2 types of Key Code of P#R that is sent out to tell the status of pressing key. Firstly, it is **Binary Code (BCD8421)** through Connector 8PIN; in this case, Pin ST# and Pin P#/R display the status of pressing key or releasing key. Secondly, it is **ASCII Code** through Connector RS232-TTL(UART) with the fixed Baud Rate 9600; in this case, it sends ASCII 'P' or 'R' before the value of Key Code to notify user to know that it is the Key Code that is sent out because of pressing key or releasing key.
- Has pad of Key Touch that is made from various materials. If it is transparent plastic, it should be 1-2 millimeter thin; on the other hand, if it is made of other materials, the thickness depends on the features and electric sensitivity of its.
- Has 1 special Key that is used as normal key or Key Fun that is used with other keys (press 2 keys simultaneously)

2. Features and Structure of Board ET-TOUCH PAD 4x4

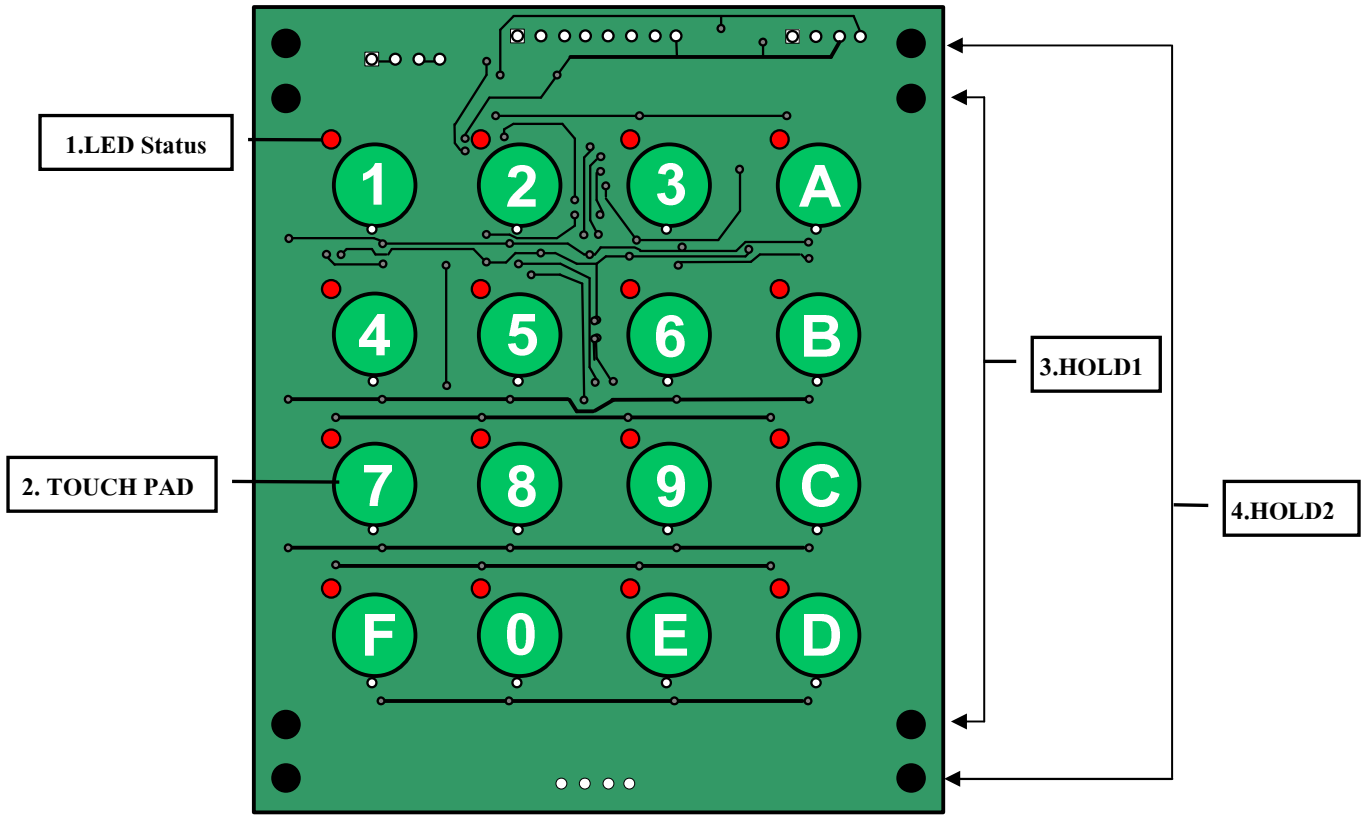


Figure 2.1 displays a bird's eye view of PCB Board.

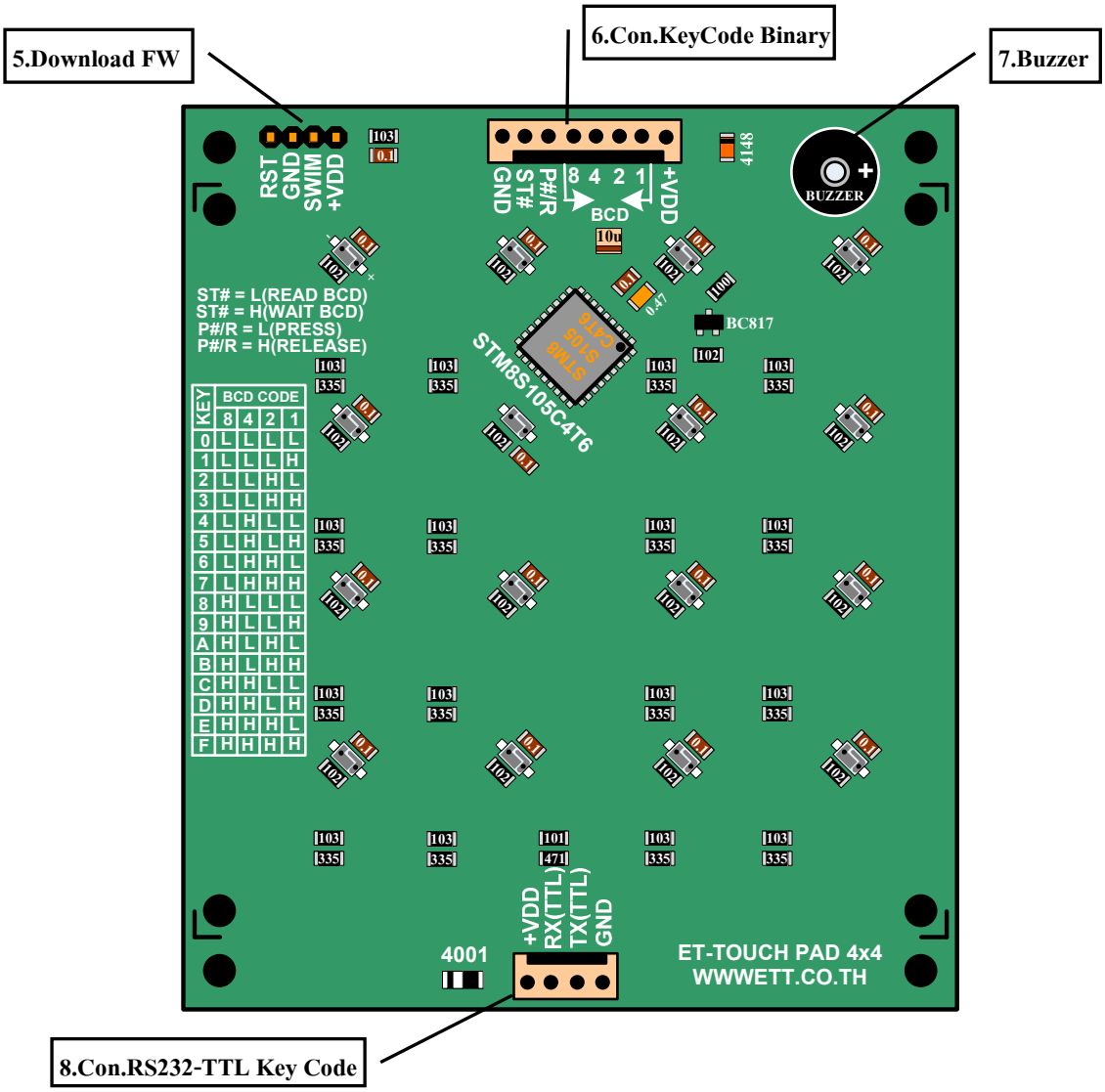


Figure 2.2 displays the back PCB of Board.

1. **LED Status:** It is LED to display status of pressing Key in the format of running light while user is supplying power into board in the first time. In this case, it is ON for awhile to notify user to know that the key has pressed.
2. **TOUCH PAD:** It is Key Position that user has pressed or touched key; normally, there is transparent plastic with 2mm. thin on it, so it is unnecessary to directly touch any printed design because it is high sensitivity. Moreover, there is no any number on the PAD; however, this example has already been written to be the reference or key's name and user will understand easily when specifying the key position in this manual. This key's name is the Key Code of that key and user can read more information about the Key Code of Key from Table of KEY CODE.
3. **HOLD1:** It is a hole to hold the plastic Touch PAD with PCB.
4. **HOLD2**: It is a hole to hold ET-TOUCH PAD 4x4 with box or other materials.

5. **Download FW:** It is used to upgrade Firmware to ET-TOUCH PAD 4x4 (normally, it has been done by ETT).

6. **Con.KeyCode Binary:** It is Connector 8PIN as shown in the picture 2.3 below. Its function is to send Key Code of Key that is pressed or released in the format of Binary BCD8421 and then send the status of pressing key and releasing key in the format of Pulse and Logic to user. Moreover, it is the Connector Power Supply for ET-TOUCH PAD 4x4. as well.

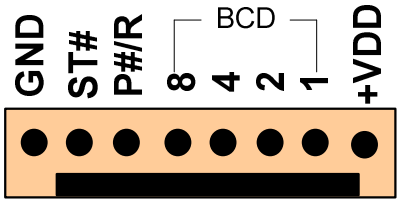


Figure 2.3 displays Connector to read the Key Code as Binary BCD8421.

Details of each PIN

+VDD,GND = It is 3.3V VDC Power Supply for Board (it is compatible with MCU 3.3V) or 5V (it is compatible with MCU 5V).

BCD8421 = It is PIN that sends Key Code 4Bit. This PIN No.8 is Bit MSB and Key Code always be sent out when key is pressed or released. This Key Code value is the value of the key that has been pressed or released recently and it is shown for awhile until the new key is pressed.

P#/R = It is Press/Release to tell the logic status of pressing or releasing key; when the key is pressed, its logic status is 0 and it is still 0 until the key is released. On the other hand, when the key is released or not pressed any key, the logic status of this pin is 1 and it is still 1 until the key is pressed and its logic status becomes 0.

ST# = It is STROBE to tell the logic status of pressing key or releasing key as well; however, this signal is sent out in the format of Signal Pulse. It means that when it is in normal status, this signal is always Logic 1 and it is still 1 until the key is pressed, this signal is 0 for 10ms and then returns to Logic 1 automatically. When the key is released, this signal is Logic 0 for 10ms again and then returns to Logic 1 automatically as well.

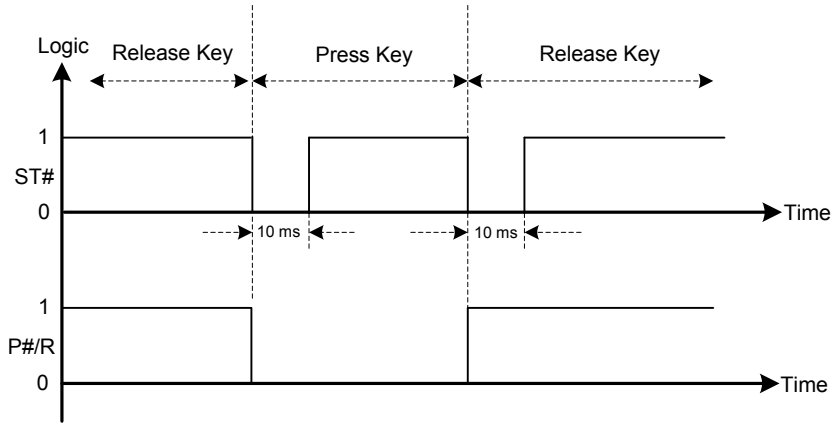


Figure 2.4 displays Timing Diagram of Signal ST# and P#/R when pressing key and releasing key.

The signal that tells the status of pressing key or releasing key (P#R and ST#), user can choose only one signal or both signals to check whether the status of key is pressed or released. The main objective that uses both

signals is to use when 2 keys are pressed simultaneously; so it is necessary to use both signals to check status of pressing key or releasing key.

7. **Buzzer:** It is Buzzer that makes voice while supplying power into board in the first time and its voice is "BEEP" while pressing any key.

8. **Con.RS232-TTL Key Code ASCII:** It is Connector RS232-TTL(UART) as shown in the picture 2.5. Its function is to send the status of pressing key that is "P"=0x50 or status of releasing key that is "R"=0x52 to be the first Byte; the second Byte is Key Code of the Key that is pressed or released in the format of ASCII CODE; and finally, the third Byte is 0x0D to end the command. The Baud Rate that is used to send/receive data is the fixed rate at 9600 bit/s.

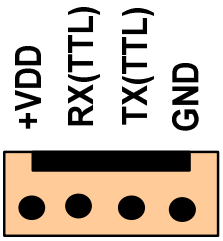


Figure 2.5 displays Connector RS232-TTL(UART) for reading Key Code in the format of ASCII CODE.

Details of each PIN

+VDD,GND = It is 3.3VDC Power Supply (be compatible with MCU 3.3V) or 5V (be compatible with MCU 5V).

RX(TTL) = It receives data from user; normally, it is unused because it does not receive any command from user.

TX(TTL) = It sends the status of pressing/releasing key and Key Code (ASCII Code) to user; the format of sending data is shown in Table below;

	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

For example, when Key 5 is pressed, the value that is sent out is P5 and 0x0D or0x50,0x35,0x0D. In this case, 'P' is the status value of pressing key; '5' is the Key Code; and 0x0D is the Byte to end the command. When the Key 5 is released, it sends the value of R5 and 0x0D; in this case,'R' is the status of releasing key; '5' is the Key Code; and 0x0D is the Byte to end of the command.

NOTE: Be careful of reading Key Code through Connector RS232-TTL because this connector is not connected with any Line Driver MAX232; so, the level of Signal RS232 that will be connected for reading Key Code needs to be the same level of signal TTL. If Board MCU that is connected has Max232 or user needs to read Key Code to display on Hyper Terminal from ET-TOUCH PAD 4x4 directly, user needs to connect Line Driver Max232 between ET-TOUCH PAD 4x4 and Board MCU or RS232 of PC.

For the actual application, user needs to choose only one method to read Key code; Binary BCD8421 or UART RS232-TTL because it chooses Connector correctly.

Table: KEY CODE of 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. Application of ET-TOUCH PAD 44x4

Normally, when supplying power into board without pressing any key, the status of pins is the default values as mentioned below and Connector RS232 does not send any data. If any key is pressed and remained pressed, the other rest of keys are locked and are unable to press any more (except Key F) until user releases the key that have pressed first. On the other hand, if Key F is pressed, user is able to press other rest of keys; it means that user is able to use Key F as normal key and user is able to use Key F with other rest of keys. Every time user presses any key, it makes voice as "beep" and LED of that key will be ON for a while and then OFF to notify user to know that that key is pressed.

When any key is pressed or released, the status of Key Code that is both Binary and ASCII, including Signal ST# and P#/R will be updated according to the status of key that has recently been pressed and release. It means that any key is pressed or released, it always sends out Key Code, including status of Signal ST# and P#/R that always changes as well. See more the status of sending signal and Key Code from the formats below;

3.1 Reading Key Code as Binary BCD8421

- **Default Value** = When supplied power into board, LED of KEY will be ON and it starts running from key1 to another keys as spiral and make voice as beep. Whiles LED is running, the value at Connector 8PIN is set as follows;

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

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

- **Press Key** = When user presses any key at Connector 8PIN, the operating status will be changed as follows;

1) **PIN ST#** (Signal Strobe)**:** It changes Logic status from 1 to 0 for 10ms and it becomes 1 automatically as shown in the picture 2.4.
2) **PIN P#/R** (Signal of pressing key or releasing key)**:** It changes Logic status from 1 to 0 and it is still this status as long as the key is pressed as shown in the picture 2.4.
3) **PIN BCD8,4,2,1:** It is changed according to the Key Code of the key that has lately pressed.

In case of using Key F with other keys and when user has pressed Key F, the status of Pins will be changed according to step 1-3 as mentioned above. Next, when user has pressed the second key that is any key while Key F is still pressed, status at PIN ST# is changed according to Step 1 and status of PIN P#/R is still Logic 0, not be changed. The method to check the second key that is pressed is to check the change of Signal ST# because when the second key is pressed, the value of Key Code at PIN BCD8421 will be changed according to the value of key that is lately pressed.

- **Release Key** = When user releases any key from pressing, the status at Connector 8PIN will be changed as follows;

1) **PIN ST#** (Signal Strobe)**:** It changes Logic status from 1 to 0 for 10ms and it becomes 1 automatically as shown in the picture 2.4. In this case, it is the same as the step of pressing key.
2) **PIN P#/R** (Signal of pressing key or releasing key)**:** It changes Logic status from 0 to 1 and it is still 1 as long as the key is still pressed as shown in the picture 2.4.
3) **PIN BCD8,4,2,1:** It changes the status according to Key Code of the Key that is lately released.

In case of using Key F with other keys; after user has pressed them simultaneously and then released any key first, the status of PIN ST# will be changed according to step 1; moreover, PIN P#/R is still Logic 1, not be changed because another one key is still pressed. PIN BCD8421 changes the Key Code according to the key that is released and when user has released the second key, the status of pins are changed according to step 1-3 as

mentioned above. The method to check status of releasing key is to check Signal ST# and read Key Code of the key that is released; for Signal P#/R is used to check and ensure that no key is pressed any more.

3.2 Reading Key Code as ASCII

If reading Key Code as ASCII, the device that is connected with Connector RS232 of ET-TOUCH PAD has to have voltage level as TTL(5V);otherwise it needs to connect MAX232 to be Line Driver and user can read value through Serial Port of PC. Baud Rate for sending/receiving data is fixed rate at 9600 bit/s.

- **Default Value** = When supplied power into Board, it makes LED of key ON and it starts running from key1 to another key as spiral and it makes voice as beep while LED is running; moreover, no data is sent out to Port RS232.

- Press Key = When user has pressed any Key, ET-TOUCH PAD will send 3 Byte Data to Connector RS232; the first Byte is ASCII 'P' to declare the status of pressing Key; the second Byte is ASCII Code of the Key that is pressed; and finally, the last Byte is 0x0D. When user has read the Key Code, user will know which key is pressed.

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

Table shows the format of sending data when pressing Key.

In case of using Key F with other key and user starts pressing Key F, ET-TOUCH PAD sends ASCII 'P'(0x50) to be the first Byte; follow by Key Code ASCII 'F'(0x46) to be the second Byte and finally, follow by 0x0D to be the last Byte. Next, when user presses any key to be the second key while Key F is still pressed, ET-TOUCH PAD sends data as follows; the first Byte is ASCII 'P'(0x50); the second Byte is ASCII Key Code of the key that is pressed lately; and finally, the last Byte is 0x0D. This is the operating feature while using Key F with other keys; when writing program, user needs to check the Key F that is pressed first and then check the second Key that is pressed. If the first Key that is pressed is not Key F, it means that it is the single key, not be used with other key.

- **Release Key** = When user has released any key from pressing, ET-TOUCH PAD sends 3 Byte Data to Connector RS232; the first Byte is ASCII 'R' to declare the status of releasing Key; the second Byte is ASCII Code of the Key that is released; and finally, the last Byte is 0x0D. When user has read the Key Code, user will know which key is released.

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

Table shows the format of sending data when releasing key.

In case of using Key F with other keys and after pressed Key F and other keys completely, if user released any key first, ET-TOUCH PAD sends data as follows; the first Byte is ASCII 'R'(0x52); the second Byte is ASCII Key Code of the Key that is released; and finally, the last Byte is 0x0D. Next, when user released the second key, it sends data as follows; the first Byte is ASCII 'R'(0x52); the second Byte is ASCII Key Code of Key that is released lately; and finally, the last Byte is 0x0D. This is the operating feature while releasing key when user uses Key F with other keys. When user writes program in the part of releasing key (still pressing Key F + other one key), user needs to check the Key that is released whether it is Key F or not. If yes, it needs to exit from Loop to check other one key; on the other hand, if it is not Key F, it needs to loop to read other key values for pressing with Key F.

4. Example Program and How to Interface Circuit to Read Key Code by MCU

There are examples for reading Key Code as Binary BCD8421 and ASCII Code to support 3 MCU families; ARM7 LPC2138 by using C-Keil Compiler; AVR MEGA 64/128 by using C-WIN AVR; and MCS51-AT89C51ED2 by using ASCII Code. There are 2 examples for each MCU; firstly, it reads Key Code as Binary BCD8421 and secondly, it reads Key Code as ASCII Code. See more information about the operation of this program as described below (Source Code is provided in CD);

Example of Reading Key Code as Binary BCD8421

***Example EX1_Read_1Key_BCD**:* It is the example 1 to read Key Code as Binary by using single key. When user touched Key, MCU checks signal ST#; reads and stores Key Code BCD8421; check signals P#/R to check whether the key has actually pressed. Next, MCU sends Key Code though Port that is connected with LED and then type the Key Code that is read to display on Hyper Terminal on PC. For example, if user presses Key2, MCU sends the value 0x02. This is the general operation of the example 1 for each MCU. The example circuit that supports example1 is displayed below;

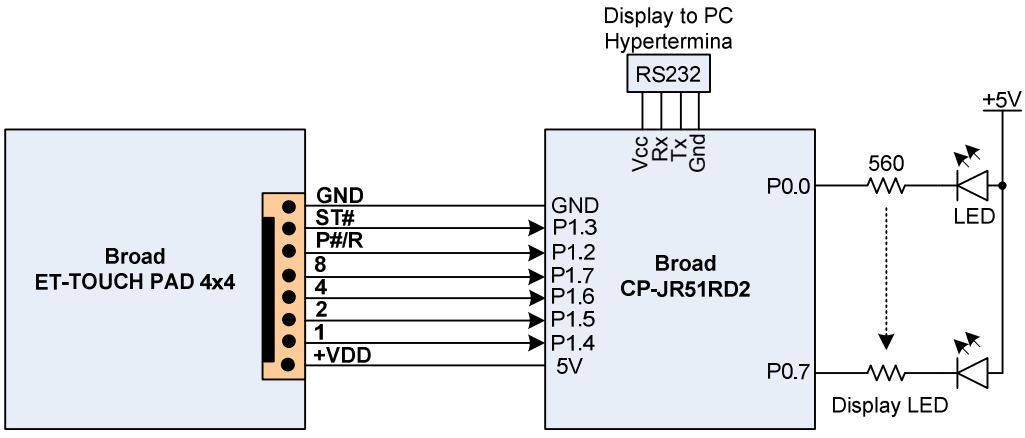


Figure 4.1 shows how to connect circuit of example 1 for Board CP-JR51RD2 and ET-TOUCH PAD 4x4

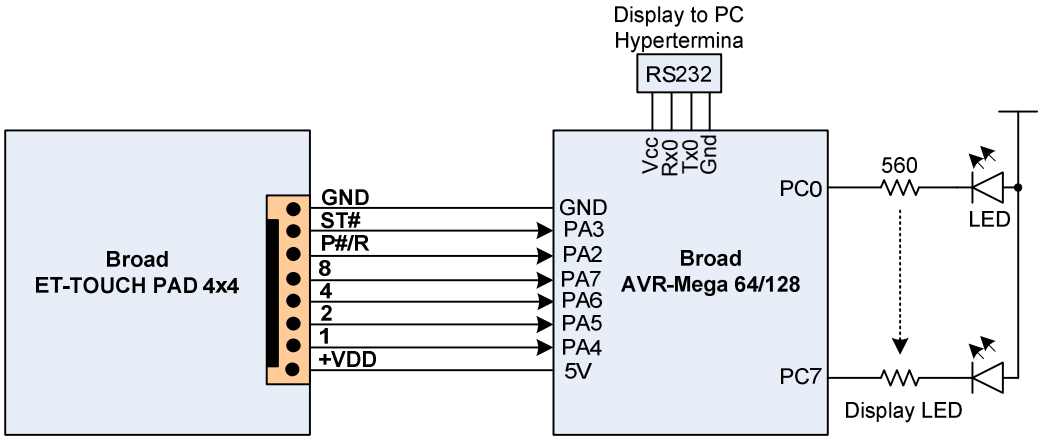


Figure 4.2 shows how to connect circuit of example 1 for Board AVR MEGA64/128 and ET-TOUCH PAD 4x4.

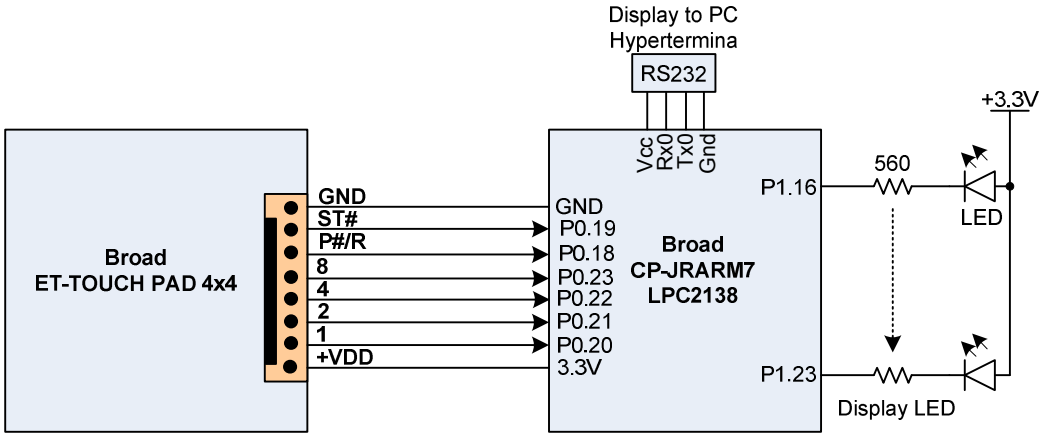


Figure 4.3 shows how to connect circuit of example 1 for Board CP-JR ARM LPC2138 and ET-TOUCH PAD 4x4.

Example EX2_Read_2Key_BCD: This is the example 2 that reads Key Code as Binary by using Key F with other Key. When user touched Key, MCU checks Signal ST#; after the signal ST# has been found, user needs to wait for the signal become '1' before going to the other steps, otherwise it makes program operate incorrectly and not correspond with the user's requirement. When the ST# became '1' successfully, it reads and stores Key Code BCD8421; next, it checks whether Signal P#/R is still '0' or not to ensure that the Key Status is still pressing. Next, it checks whether the Key Code that is pressed is the Key F or not; if no, it means that it is single key and program only responds to single key that is pressed. So, MCU sends the Key Code through Port that is connected with LED and types the Key Code that is read on Hyper Terminal. If the Key that is pressed is Key F, it needs to check the other one key that is pressed together by checking Signal ST#. After the Signal ST# has been found, it reads the Key Code of the Key that is pressed together and then checks Signal P#/R to ensure that the Key is still pressed together. Next, it makes program respond the key that is pressed together; in this case, MCU sends the Key Code through Port that is connected with LED and types the Key Code that is read to display on Hyper Terminal. Next, it loops to check whether Key F is released or not; if not, it loops to read other key.The example circuit that supports this example 2 is the same as the example 1 above.

***Example EX3_Application_2Key_BCD**:* It is the example 3 that is applied to use Key F with other key. If no key is pressed, it displays message **"Not! Touch Key"** on the LCD Display; on the other hand, if single key is pressed, it displays message **"Touch Key = [name of key that is pressed]**; or if Key F and other one key are pressed together, it displays message **"Touch Key = FUN+[name of Key that is pressed]**. The operating format of program is the same as example 2 but it is only different in the part of program that responds to pressing key.

The example circuit in the part of ET-TOUCH PAD and MCU is the same as the example 1; moreover, the example circuit in the part of LCD Display and MCU is provided in the Source Code.

Example of Reading Key Code as ASCII CODE

If it reads value as ASCII Code, the Key Code of each key that is pressed is corresponding with values in the Table of KEY CODE in the blank of ASCII MODE. User can read values from Connector RS232(TTL) of ET-TOUCH PAD 4x4 that uses BAUD RATE 9600 bit/s for sending/receiving data; moreover, it is able to interface with Pin TX,RX of MCU directly. However, if MCU has Line Driver Max232, it needs to interface Line Driver Max232 with ET-TOUCH PAD to adjust the voltage level of RS232 to be the same level.

***Example EX4_Read_1Key_ASCII**:* It is the example that reads Key Code as ASCII; in this case, it uses single key and loop to receive/send data from ET-TOUCH PAD. When user touched key, MCU receives the first Data Byte to check whether it is ASCII 'P' or not; if no, it loops to receive the new data. On other hand; if yes, it means that the Key is pressed, it waits for receiving Key Code of the Key that is pressed to display on Hyper Terminal and PORT LED that is connected with MCU. For example, if user pressed Key 7, the operating result that is displayed on Hyper Terminal is ASCII number 7 and LED will display the value as 0x37.

The example circuit that supports the example 4 is displayed below.

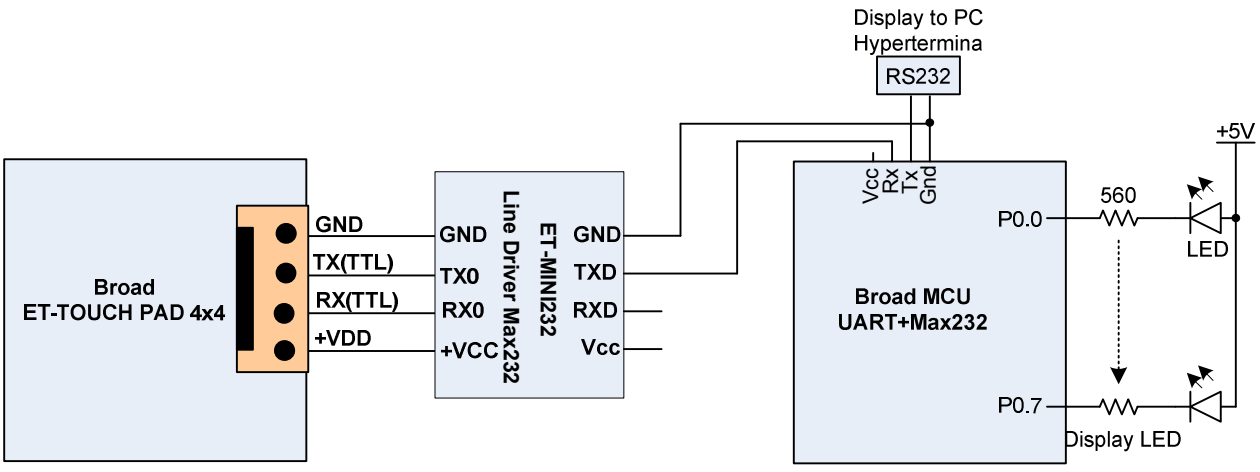


Figure 4.4 shows how to connect circuit of example 4 for Board MCU numbers and ET-TOUCH PAD 4x4.

The circuit in the picture 4.4 above, it is used to test the written example when the Board MCU has only one UART Channel and MAX232 is connected to be Line Driver; so, the side of ET-TOUCH PAD has to interface another one MAX232 between devices.

***Example EX5_Read_1Key_ASCII_INT**:* This example is similar to the example 4 but it receives data without looping to receive data any more; in this case, it checks Signal Interrupt Rx instead. When Signal Interrupt occurred, program jumps to receive data and check it instantly.

The example circuit is similar to the example 4; in this case, it provides Source Code that describes how to connect pins of each MCU completely.

***Example EX6_Read_2Key_ASCII_INT**:* It is the example that reads Key Code as Binary by using Key F with other Key; moreover, it uses Signal Interrupt to set the interval of receiving data. When user touched Key, Program jumps to receive and store all of 3Byte data, and then checks whether the last Byte is 0x0D or not; if yes, it means that it has received all data correctly and successfully. Next, it checks whether the first Byte is ASCII 'P' or not; if yes, it means that it pressed key. Next, it checks whether the second Byte is ASCII Key Code of the pressed Key that is equal to ASCII CODE 'F' or not; if no, it means that it pressed only one key and program responds to only one key pressing. MCU sends the Key Code through Port that is connected with LED and types the Key Code that is read to display on the Hyper Terminal. On the other hand, if the Key that is pressed is Key F, program has to loop to receive the second key that is pressed with Key F. It is able to exit from the Loop if the Key F is released.

The example circuit to test program is the same as picture 4.4 of the example 4 above.

The circuit below illustrates how to directly connect ET-TOUCH PAD 4x4 with PORT UART of MCU for reading ASCII Key Code through RS232(TTL). Normally, ET-TOUCH PAD only sends data through RS232(TTL), it does not receive any data from external; so, Pin RX(TTL) is connected or not be connected as required.

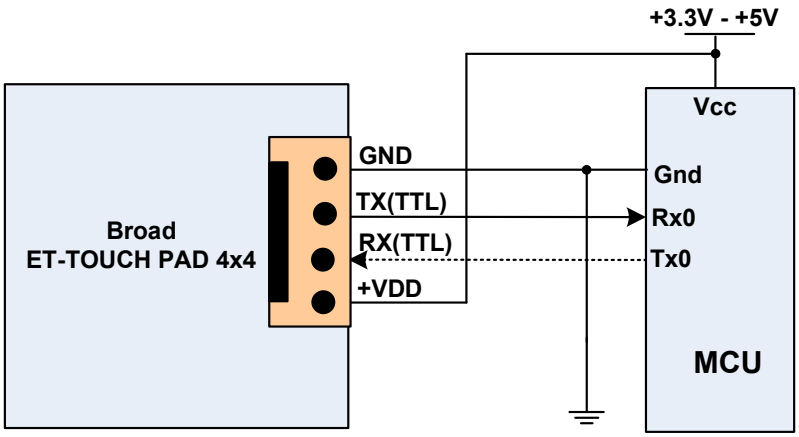
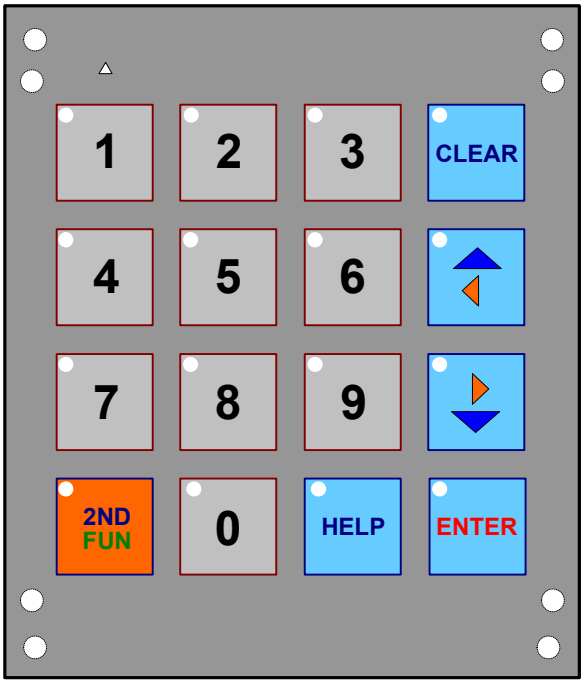
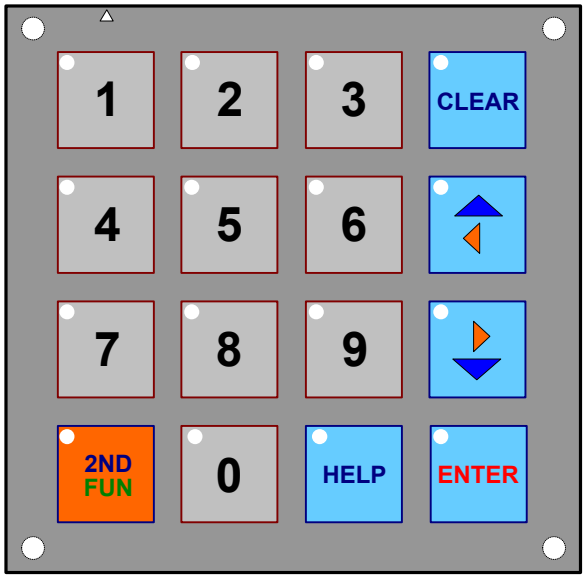


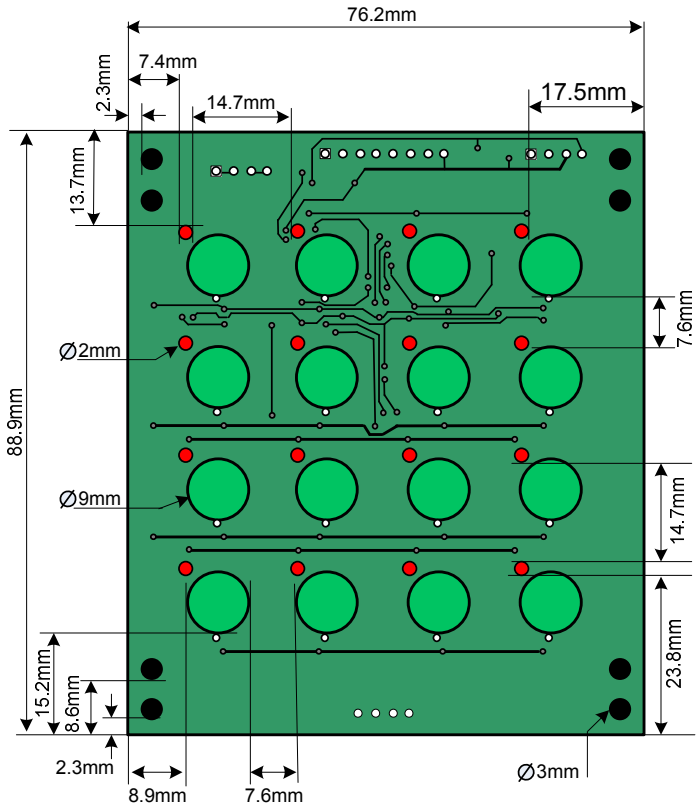
Figure 4.5 shows how to connect Board ET-TOUCH PAD 4x4 RS232(TTL) and PORT UART of MCU directly.

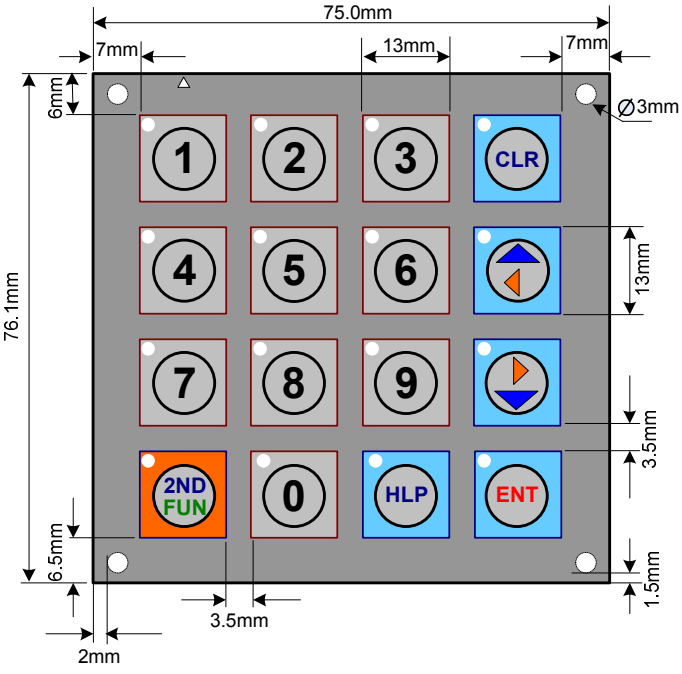
Actual size of KEY Label





Size of PCB LAY OUT and SAMPLE KEY LABEL





PCB LAY OUT(ด้านบน)

SAMPLE KEY LABEL

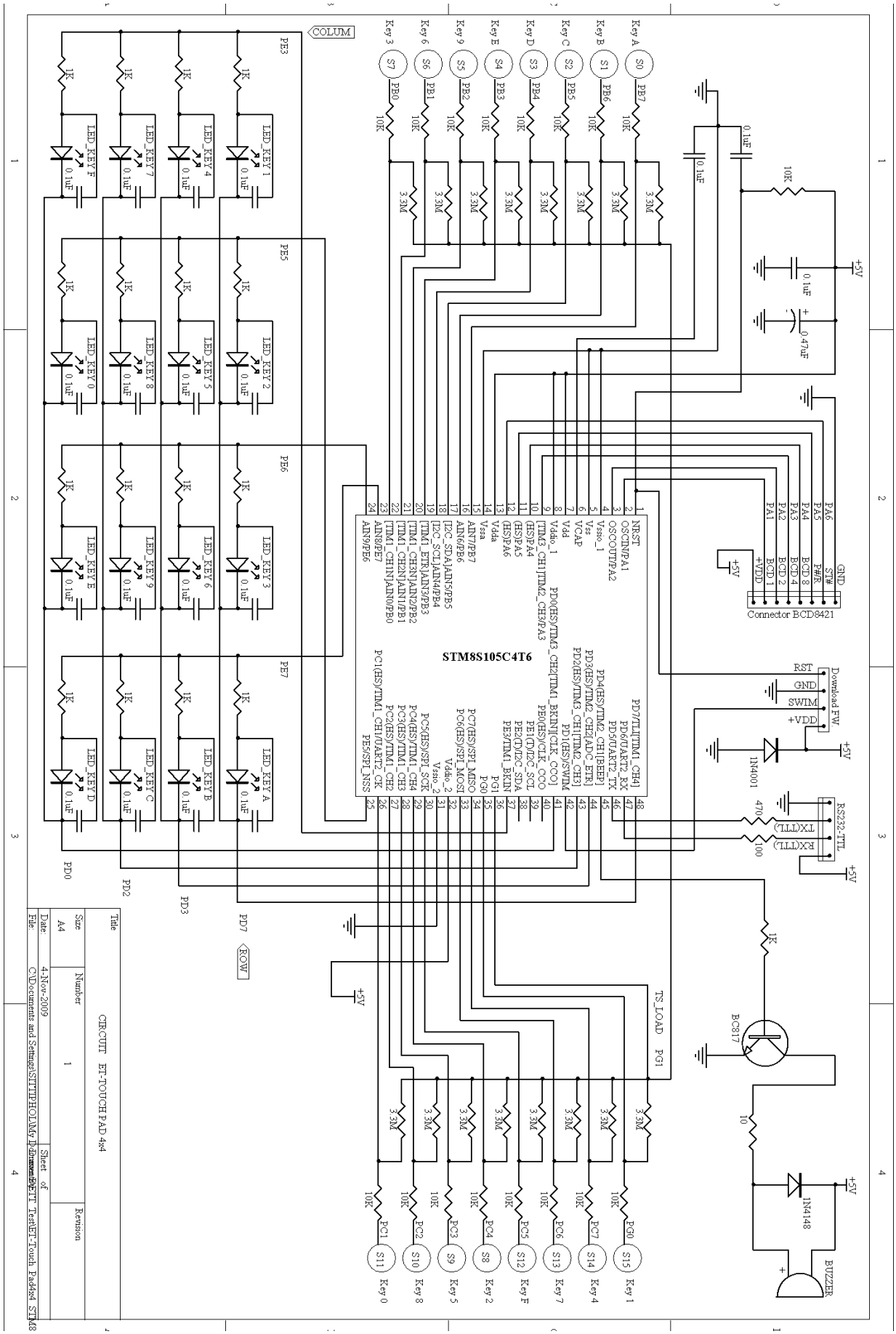


Figure displays circuit of ET-TOUCH PAD 4x4