ET-TOUCH PAD 4x4 V2

1. SPECIFICATIONS of BOARD ET-TOUCH PAD 4x4 V2

- Be Capacitive Sensing 16-KEY Touch PAD (4x4)
- Use +3.3 VDC or +5VDC for Board
- Show status of pressing key of user by sound (Buzzer) and LED at particular Key's position
- Can enable/disable sound (ON/OFF Buzzer) and setup operating system of LED Status Key by Jumper independently
- ET-TOUCH PAD 4x4 V2 displays initial status (Power-On) by Beep sound and LED Status Key flashes on and off regularly
- Divide Board into 2 parts; Touch PAD Board and MCU Control Board. Both parts are connected together by strong Connector.
- Has 2 types of OUTPUT for Key Code of Key that is pressed by user as follows;

    1) **Binary Code (BCD8421)**: Send value through CONNECTOR 8 PIN;
    Pin T#/R and SHIF# shows status of pressing or releasing Key

    2) **ASCII Code**: Send value though Connector RS232 (TTL and Line Driver); setup fixed Baud Rate at 9600.
    ASCII 'P' or 'R' preceded Key Code to notify status of pressing or releasing Key

- If Key Touch Pad is made from translucent plastic, it is 1-3 millimeters thick (depending on humidity in the air). If it is made from other materials, the thickness depends on electric sensitivity of each material.
- Has 1 special Key that can be either used as Normal Key or Key SHIF that is mutually pressed with other remaining keys (both key must be pressed at the same time).
- Control the operating system of Buzzer and LED Status Key (when no Key is touched) by ASCII Command through RS232. It has to set Jumper to enable Buzzer and LED Status Key

2. FEATURE and STRUCTURE of BOARD ET-TOUCH PAD 4x4 V2

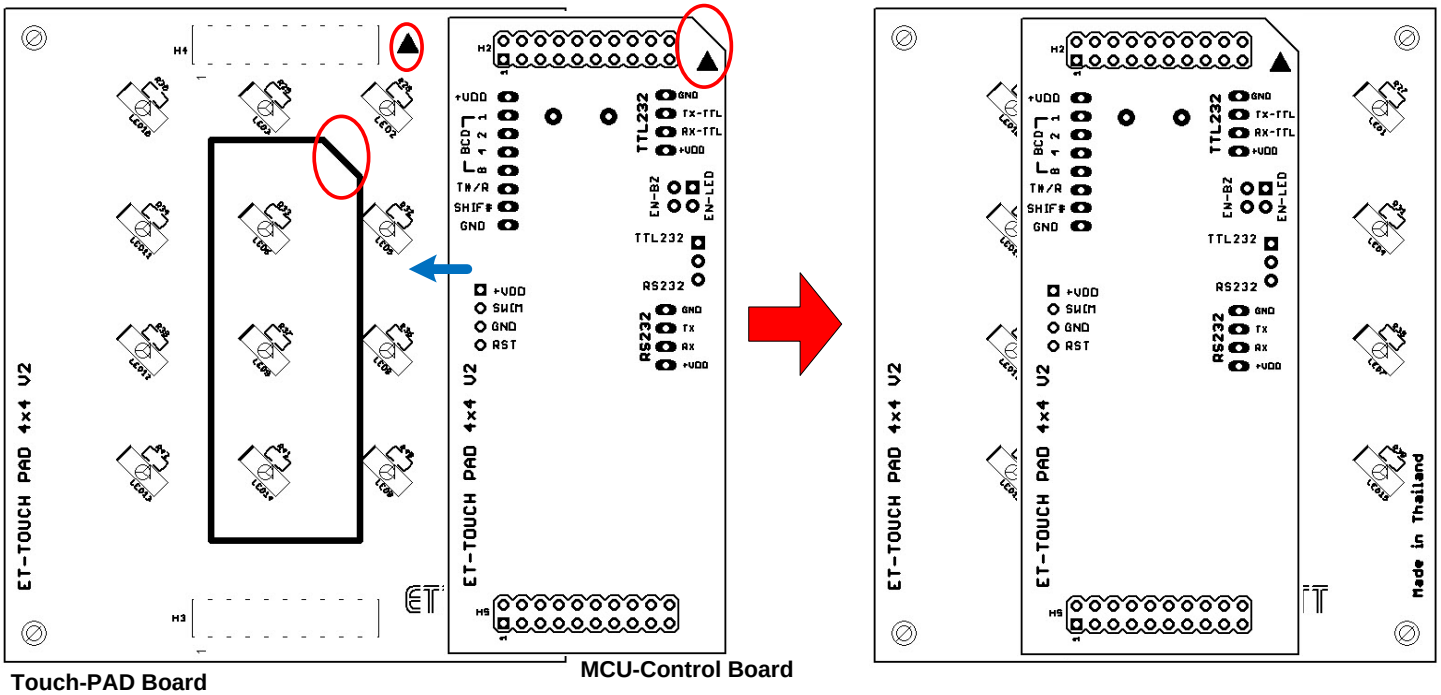


Figure 2.1 shows arrangement of Touch PAD Board and MCU Control Board.

From Figure 2.1 above, it shows how to assemble Board ET-TOUCH 4x4 V2 when separated this board into 2 parts and then re-assembled both parts together, please assemble board as shown in the figure above. While assembling both parts together, please notice direction of the arrowhead or cut-angle PCB of MCU-Control Board, it must be the same direction as shown at the back of Touch PAD Board; and finally, user can assembly both parts together.

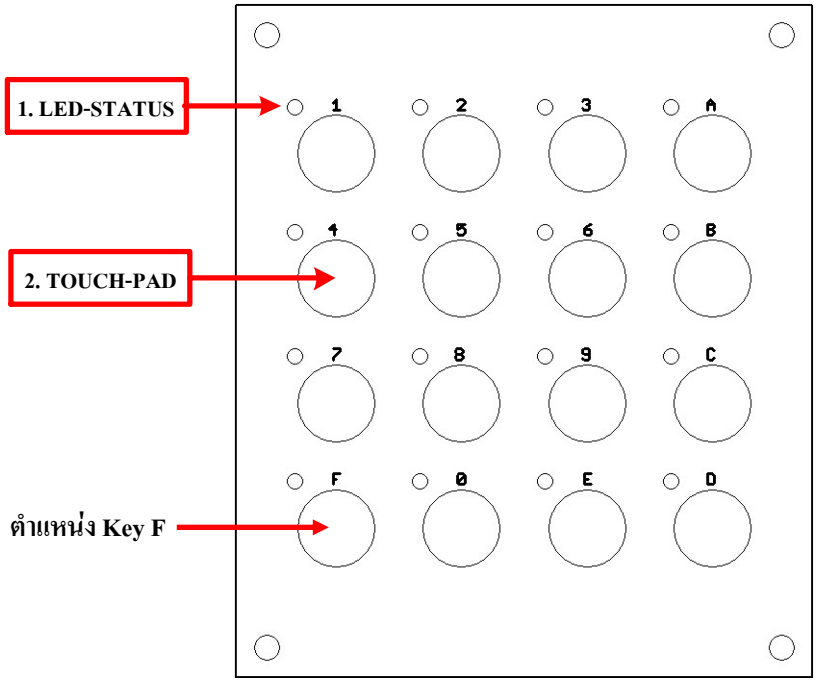


Figure 2.2 shows the front of Touch PAD Board.

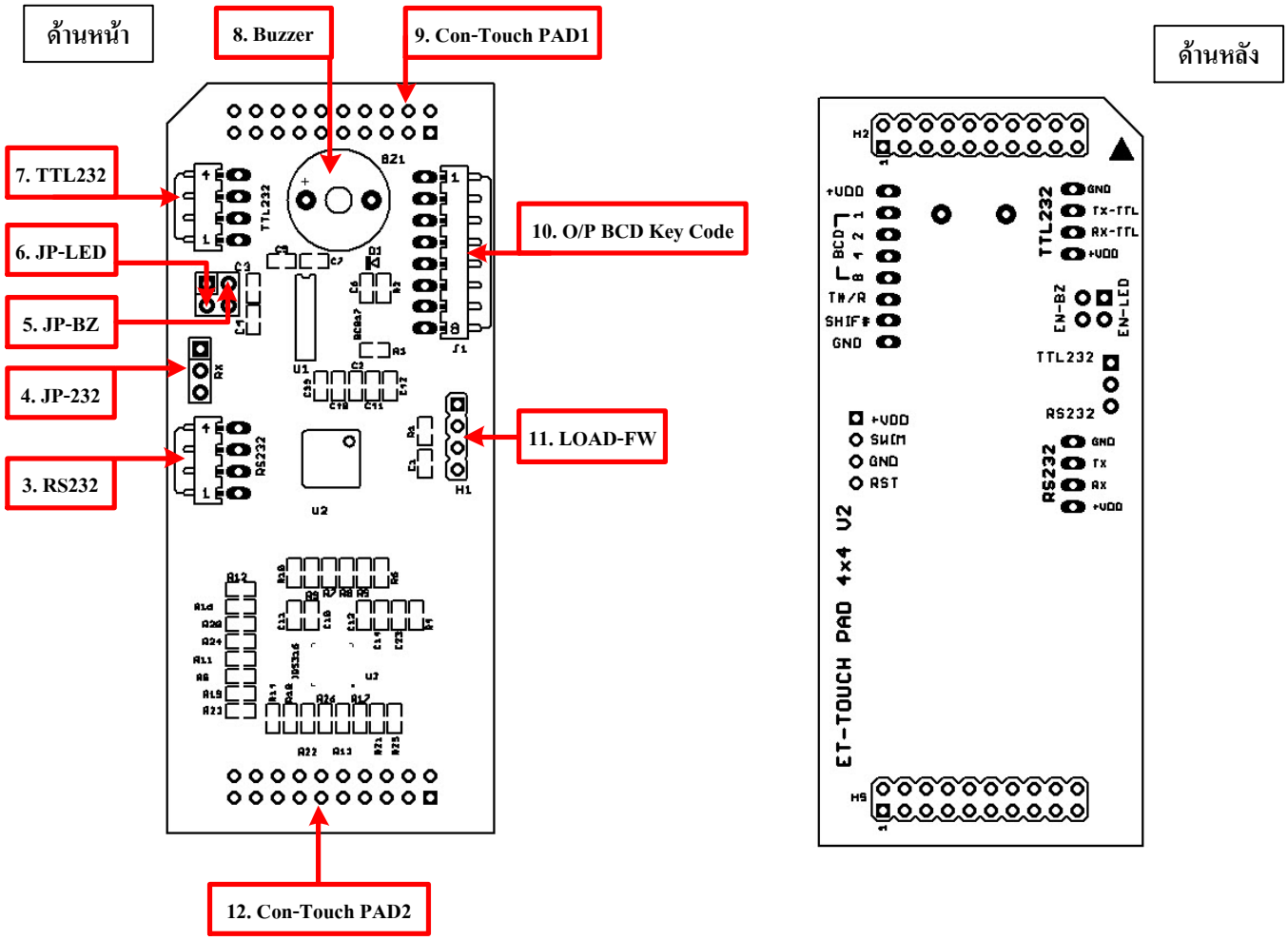


Figure 2.3 shows the front and the back of MCU Controller Board.

-1. LED-STATUS : This LED shows status of pressing Key; it regularly flashed on and off according to Key that is pressed/released by user. It regularly flashes on and off when Board is in Power-ON. This LED is controlled by ASCI Command through RS232. These16 LEDs are enabled by Jumper JP-LED(6).

-2. TOUCH-PAD : It is Key's position that user touches or presses. Numbers that are shown on PAD are unique Key Code of each Key; when any Key is pressed, it sends this Key Code through RS232 or Connector O/P BCD Key Code. For more information, please read Key Code of Keys from Table KEY CODE.

-3. RS232 : It is Connector RS232 Line Driver that is connected through IC Line Driver 3232 on board. So, it can be connected with PORT RS232 of PC; or, it is connected with Port RS232 of external MCU that is connected through IC Line Driver 3232 completely; in this case, it has to cross Cables between RX, TX. Function of this Connector is to send status of pressing/releasing key and send Key Code of each Key that is pressed/released to user in the format of ASCII Code; for more information, please read "How to read Key Code as ASCII". Moreover, it can receive user's Command to control operating system of Buzzer and LED STATUS while no KEY is pressed; for more information, please read "COMMAND for CONTROL LED&BUZZER".

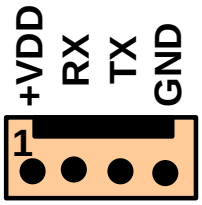


Figure 2.4 shows Connector RS232 Line Driver for sending Command and reading Key Code as ASCII Code.

Details of each PIN

+VDD,GND = Interface Power Supply for Board that is either VDC 3.3V (for MCU 3.3V) or 5V (for MCU 5V)

RX = Receive user's Command to control LED and Buzzer

TX = Send status of pressing/releasing Key and send Key Code (ASCII Code) to user

When using this Connector RS232, it has to set Jumper JP-232(4) to the position of RS232

-4. JP-232 : This Jumper is used to choose Connector 232. If setting Jumper to the position of TTL232, it chooses Connector TTL232 (No.7); or, if setting Jumper to the position of RS232, it chooses Connector RS232 (No.3).

Figure 2.5 shows how to set Jumper to choose Connector 232.

**When reading Key Code as ASCII Code, user has to choose only one Connector and it has to set Jumper JP-232 to the correct position. When user requires choosing any Connector, it depends on Board MCU that is connected with to read Key Code. If the Board has already been connected with IC Line Driver or it is connected to 232 of PIC, it chooses Connector RS232(3); or, if user requires connecting with Pin Uart of MCU directly, it chooses Connector TTL232(7).*

-5. JP-BZ **:** It is Jumper ON/OFF Buzzer. When looking from the back of Board MCU Control, the Jumper is below. If added this Jumper, it enables the Buzzer instantly (ON-Buzzer); or, if removed this Jumper, it disables the Buzzer instantly (OFF-Buzzer). Remember, when setting this Jumper, it has an effect on Command Control Buzzer that is used through RS232.

6. JP-LED : It is Jumper to setup operating system of LED Status Key. When looking from the back of Board MCU Control, the Jumper is above. If added this Jumper, it enables the operating system of LED Status Key to run as normal as setting in Firmware; or, if removed this Jumper, all LED is inactive. Remember, when setting this Jumper, it has an effect on Command Control LED Status Key that is used through RS232. ***NOTE: This Jumper has an effect on the operating system of LED as mentioned above when Board ET-TOUCH PAD 4x4 is Power-ON again after set Jumper completely***".

-7. TFT232 **:** It is Connector RS232 as Signal TTL that can be connected with Pin Uart (Rx,Tx) of MCU directly; in this case, it has to cross line between RX,TX while connecting. The operating system of this Connector is the same as Connector RS232 No.3 but the signal level only is different because it is lower, so it can be connected with Pin MCU directly.

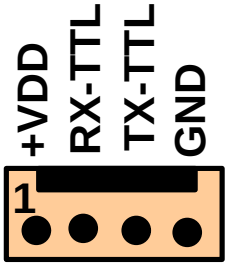


Figure 2.6 shows Connector TTL232 to send Command and read Key Code as ASCII CODE.

Details of each PIN

+VDD,GND = Interface Power Supply for Board that is either VDC 3.3V (for MCU 3.3V) or 5V (for MCU 5V)

RX-TTL = Receive Command Control LED and Buzzer from user

TX-TTL = Send status of pressing/removing Key and Key Code (ASCII Code) to user

When using this Connector TTL232, it has to set Jumper JP-232(4) to the position of TTL232.

-8. Buzzer **:** This Buzzer produces Beep sound when touching Key or produces musical sound when board is Power-On. It controls sound by ASCII Command through RS232. This Buzzer is enabled by Jumper JP-BZ(5).

-9. CON-Toch PAD1 **:** This Connector 20PIN is connected to Board TOUCH-PAD to control ON/OFF LED on Board.

-10. O/P BCD Key Code **:** This is Connector 8PIN as shown in the picture 2.7; it sends Key Code of any pressed/released Key in the format of Binary BCD8421. Moreover, it sends status of pressing/releasing Key to user and it is used as Connector Power Supply for ET-TOUCH PAD 4x4 V2.

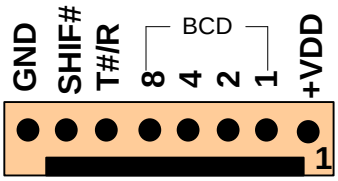


Figure 2.7 shows Connector to read Key Code as Binary BCD8421.

Details of each PIN

+VDD,GND = Interface Power Supply for Board that is either VDC 3.3V (for MCU 3.3V) or 5V (for MCU 5V)

BCD = This PIN sends Key Code 4BIT; PIN No.8 is Bit MSB. Key Code is sent out every time it pressed and released Key. The Key Code that is sent out is value of the latest Key that is pressed or released and the value still stands until any new Key is pressed.

T#/R = Touch/Release notifies status of pressing/release any Key. When pressing any Key, Logic becomes 0 and the Logic 0 still stands until the Key is released. On the other hand, when any Key is released or it does not press any Key, this Pin becomes Logic 1 and the Logic 1 still stands until it presses any new Key and this Pin becomes Logic 0 again.

SHIF# = This SHIFT notifies status of Key when pressing/releasing Key F together with another key (Key-F + another key) (it refers to the Key position in the figure 2.2 above. Key-F is setup as mutual Key). Normally, this PIN is Logic 1; when it presses the mutual Key that presses Key F together with another key (Key F + another key), this PIN becomes Logic 0 and the Logic 0 still remains until both Keys are released and this PIN becomes Logic 1 again (When pressing the mutual Key, it has to press and hold Key F for a while and then press another Key).

If using single Key, it can read status of pressing/releasing Key from PIN T#/R only. If using single Key and mutual Key together, it has to read status of pressing/releasing Key from PIN T#/R and SHIF#.

-11. LOAD-FW : It upgrades Firmware for ET-TOUCH PAD 4x4 V2 (normally, it has to return to ETT to upgrade Firmware).

-12. CON-Touch PAD2 : This Connector 20PIN is connected to Board TOUCH-PAD to receive signal touching from user.

***NOTE**: It has to choose only one type of reading Key Code that is either ASCII Code or Binary Code because it has to choose only one Connector type correctly. If reading Key Code as ASCII Code, it has to choose Connector RS232(3) or TTL232(7); or, if reading Key Code as Binary BCD8421, it has to choose Connector O/P BCD Key Code(10).*

In the part of Power Supply (+VDD,GND) for Board, user can choose only one Connector to interface signal, it is either Connector RS232(3) or TTL(7) or O/P BCD Key Code (10).

3. How to perform and read Key Code of ET-TOUCH PAD 4x4 V2

Generally, when supplied power into Board completely, LED Status Key flashes on and off regularly according to Beep sound. Status of PIN BCD is Logic 0 while Pin T#/R and SHIF# is Logic 1. For Connector 232, it does not send out any data. When pressing and still holding any Key that is not Key F, or pressing Key F together with another Key (Key F + another Key) and still holding, other remaining Keys are locked and user cannot press any key; in this case, it has to release all keys that are pressed and finally, user can press any key as required. Every time user presses any Key once, it produces Beep sound one time; LED of the pressed Key will be lit up and it is still ON until the pressed Key is released. After released keys, it turns off LED. Or, Key is pressed and held more than 20 seconds, it also turns off LED because it resumes the touching system as normal.

When any Key is pressed or released, Output Key Codes that are Binary Code and ASCII Code, including Signal T#/R will be updated according to status of the latest Key that is pressed/released. It means that whatever Key is pressed or released, it always sends out Key Code; moreover, it always changes status of Signal T #/R. For the status of Signal SHIF#, it is updated or changed when pressed or released Key F together with another Key (SHIF# changes its status when using 2 Keys at the same time).

For Buzzer and LED Status Key, it normally enables the operation by Enable Jumper JP-BZ and JP-LED. If user does not want to use any part, please remove Jumper of the unwanted device and then Power-On Board again.

Table: KEY CODE of 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 How to read Key Code as Binary BCD8421

The way to read Key Code as Binary BCD8421 is to read the value from Converter "O/P BCD Key Code". Signals of Convertor are changed according to status of operation as follows;

- *Initial Status (Default)* = When Power-ON, status at Connector O/P BCD Key Code is set as follows;

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

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

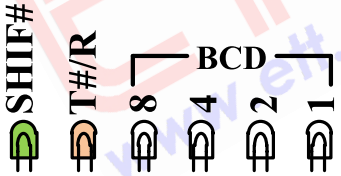


Figure 3.1 shows the Initial Status (Default) at Pins of Connector 8 Pin.

- ***Status of Press 1 Key*** = When pressed any Key, it produces Beep sound and LED Status of the pressed Key is lit up (ON) and status at Connector 8PIN is changed as follows;

1)PIN T#/R (it is signal of pressing or releasing 1 Key): It changes status of Logic from Logic 1 to Logic 0 and this status still stands throughout while the Key is still pressing. However, it does not change status of Pin SHIF#, it is still Logic 1.

2)It changes status of PIN BCD8,4,2,1 according to Key Code of the latest Key that is pressed, and the status still stands. For Key Code, please look at the Table on the upper left in the box of HEX.

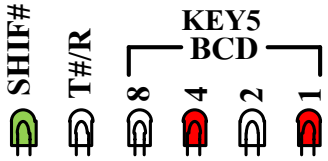


Figure 3.2 shows example of status at PINs of Connector 8PIN when pressed Key 5.

- ***Status of pressing 2 Keys simultaneously*** = When pressing 2 Keys simultaneously, it means that it presses Key F together with another Key (Key F + any Key), it produces Beep sound and LED Status Key of the Key that is pressed will be lit up in order of pressing Key. Moreover, it changes status of Connector 8 PIN as follows;

1) When pressed Key F, PIN T#/R changes status of Logic from Logic 1 to 0 and the status still stands throughout while pressing Key F.

2) PIN BCD8,4,2,1 sends out Key Code of Key F, it is 0x0F.

3) When pressing the 2nd Key that is the remaining Key (Key F is still pressed), PIN T#/R is still Logic 0 but PIN SHIF# changes the status from Logic 1 to 0 and the status still remains throughout while pressing Key F+ another Key.

4) PIN BCD8,4,2,1 sends out Key Code of the 2nd Key to user.

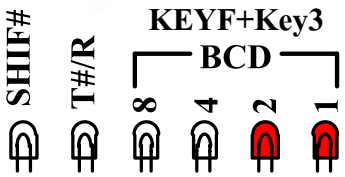


Figure 3.3 shows example of status at PINs of Connector 8PIN when pressing Key F together with another Key (Key F+Key3).

- ***Status of releasing 1 Key*** = When released any Key that is pressed, it changes status at Connector 8PIN as follows;

1) PIN T#/R (It is Signal of pressing or releasing Key) will be changed from Logic 0 to 1 and the status still stands throughout when it does not press any Key. PIN SHIF# does not change, it is still Logic 1.

2) PIN BCD8,4,2,1 changes status according to Key Code of the latest Key that is released and it is still.

- ***Status of releasing 2 Keys simultaneously*** = If pressing both keys at the same time and released either Key, signal at Connector 8PIN is not changed (the status is still the same as pressing both Keys simultaneously) but the status at Connector 8PIN is changed instantly when released both Keys (no key is pressed). It changes the status as follows;

1) PIN T#/R (it is signal of pressing or releasing Key) and PIN SHIF# changes status from Logic 0 to 1 and the status still remains throughout when it does not press any Key.

2) PIN BCD8,4,2,1 changes status according to Key Code of any Key that is pressed together with Key F and it is still.

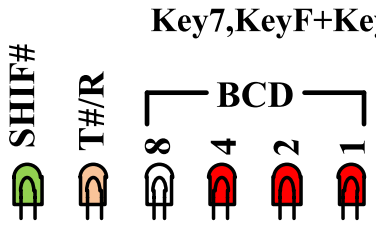


Figure 3.4 shows example of status at Pins of Connector 8PIN when releasing single Key(Key 7) or both Key (Key F+Key 7).

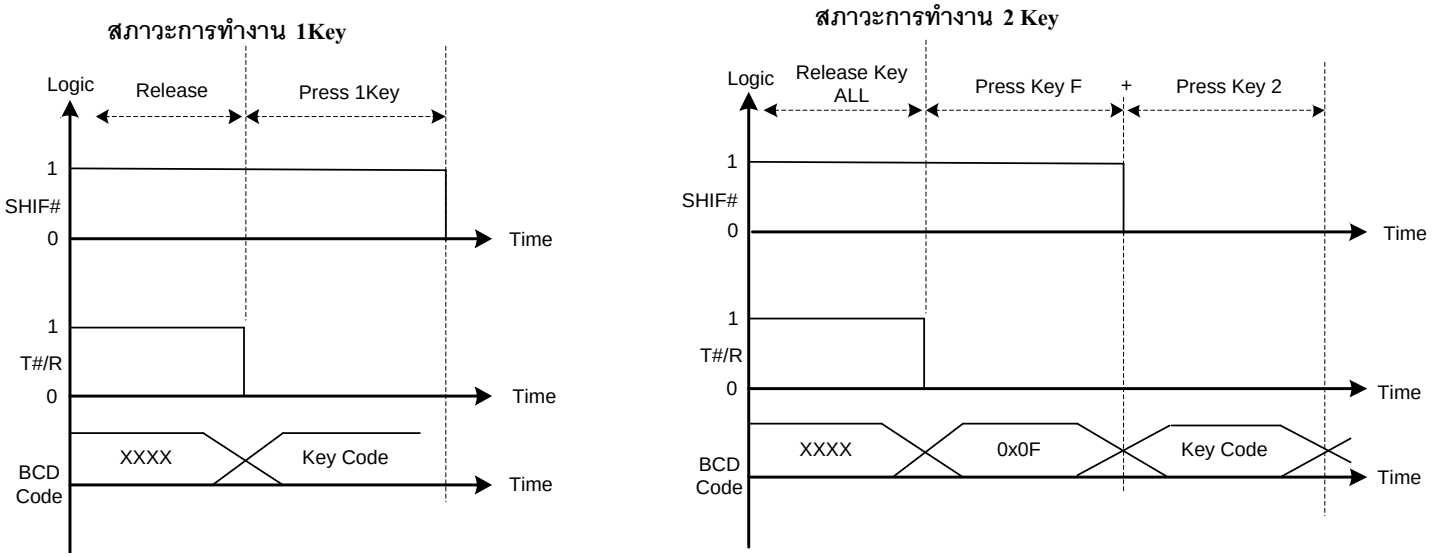


Figure 3.5 shows diagram of pressing Key as 1 Key and 2 Keys.

Summarize how to read Key Code as Binary BCD8421: As mentioned about state of pressing and releasing Key above, when writing program and it presses single Key, user can check status of pressing/releasing Key from Pin T#/R. If P#/R=0, it means that it pressed Key; or, if P#/R=1, it means that it does not press any Key. Next, it reads Key Code of the pressed Key from Pin BCD8421 and it will be used.

If pressing Key F together with another Key, user can check status of pressing Key by Pin T#/R and Pin SHIF#. If T#/R = 0 and SHIF# = 1, it means that it presses single Key; or, if T#/R = 0 and SHIF# = 0, it means that it presses Key F together with another Key. Next, it reads Key Code of the pressed Key from Pin BCD8421. When releasing another Key that is pressed together with Key F, it can check status of releasing Key from either Pin T#/R or Pin SHIF# or both pins because when releasing both Keys that are pressed together, the status of both Keys is also changed to Logic 1.

3.2 How to read Key Code as ASCII

When reading Key Code as ASCII, user can read the value from Connector “RS232 or TTL232”; it is Serial Port Interface and the Baud Rate is fixed as 9600 bit/s.

When choosing Connector type, it depends on Port 232 of Board that is connected with Touch PAD; it is either TTL or IC Driver 232. If it is TTL (Rx,Tx is connected from MCU directly), it has to connect signal at Connector TTL232 and set Jumper JP-232 to the position of TTL232; or, if Port232 of Board is connected through IC Driver 232 or it is connect to RS232 of PC, it has to connect signal at Connector RS232 of Touch PAD and set Jumper JP-232 to the position of RS232. Remember, whatever Connector user choose, user has to cross Cables from one side to another side correctly; in this case, it connects Pin Tx with Pin Rx of another side and then connects Pin Rx with Pin Tx of another side.

The Key Code that is sent out to user is ASCII Code; type of the value that is sent out depends on the state of pressing Key as follows;

-***Initial Status (Default)*** = When Power-ON, LED Status Key flashes on and off regularly according to Beep sound but it does not send out any data to Port 232. It means that Touch Pad does not send out any value through Port 232 if user does not press any key.

-***Status of pressing 1 Key*** = When pressing any one Key, ET-TOUCH PAD sends out 3 Byte Data to Port232. The first Byte it sends out is ASCII 'P' to notify that it is the status of pressing Key; the second Byte is ASCII Code the pressed Key; and finally, the third Byte is 0x0D to end. For Key Code, please look at the upper right of Table Key Code in the box of ASCII or 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

Table shows the format of sending Data when pressing one Key.

- ***Status of pressing 2 Keys simultaneously*** = It presses 2 Keys simultaneously, it means that it presses Key F together with another Key simultaneously (Key F + any Key).

1) When pressing Key F, the first Byte that ET-TOUCH PAD sends out is ASCII 'P' (0x50); the second Byte is Key Code ASCII 'F' (0x46); and finally, the last Byte is 0x0D to end.

2) The second Key that is pressed can be whatever Key as preferred. User has to press the second Key while Key F is still pressed. The first Byte that ET-TOUCH PAD sends out is ASCII 'F"(0x46); the second Byte is ASCII Key Code of the pressed Key; and finally, the last Byte is 0x0D to end.

	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)

Table shows the format of sending Data when pressing 2 Keys together.

NOTE: *When pressing 2 Keys together, there are 2 sets of Data that are sent out. The first set is Byte Data of Key F that is ASCII 'P', it is sent out to notify that it is status of pressing Key and its format is the same as pressing 1 Key as mentioned above. The second set is Byte Data of any Key that is pressed together with Key F; in this case, the first Byte that is sent out is ASCII 'F' to tell the status of pressing Key together with Key F and follow by Key Code of any Key that is pressed together with Key F.*

-***Status of releasing 1 Key*** = When released any Key that is pressed, ET-TOUCH PAD sends 3 Byte Data to Connector 232. The first Byte it sends out is ASCII 'R' to notify the status of releasing 1 Key; the second Byte is ASCII Code of the released Key; and the last Byte is 0x0D to end.

	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

Table shows the format of sending Data when releasing 1-Key and 2-Key.

-***Status of releasing 2 Keys simultaneously*** = When pressing both Keys that are pressing Key F together with another Key (Key F + any Key) at the same time and released either Key, ET-TOUCH PAD does not send any Data to Connector 232 until released 2 Keys completely. If it does not press any Key, ET-TOUCH PAD sends 3Byte Data to Connector 232 as mentioned in the Table above and the format is the same as releasing 1 Key.

Summarize how to read Key Code as ASCII: As mentioned about state of pressing and releasing Key above, when writing program and it presses single Key, user can check status of pressing/releasing Key from the capital letters 'P' and 'R'; 'P' = Press Key and 'R' = Release Key. This value is the first Byte that is sent out and user can read Key Code of the pressed Key in the second Byte.

If pressing Key F together with another Key, user can check status of pressing Key by the capital letters 'P' or 'F' that is rad in the first Bye. If it is the capital letter 'P', it means that it presses single Key; or, if it is the capital letter 'F', it means that it presses Key F together with another Key. Next, it reads Key Code of the pressed Key in the second Byte; user can know the format of the Key Code that is read is pressing single Key or pressing Key F together with another Key because user can set function of pressing Key correctly. When releasing another Key that is pressed together with Key F, user can check status of releasing Key by the capital letter 'R'; it is the first Byte that is sent out when released both Keys that are pressed together.

4. How to send Command Control Buzzer & LED Status Key

In the part of Port 232, it can send ASCII Key Code to user; moreover, it can support ASCII Command from user to control the operation of Buzzer and LED Status Key on Board. The main target is to use Buzzer and LED Status Key as Alarm Function when it does not press any Key; but when it presses any Key, Key returns to send Key Code to user automatically.

If user requires controlling the part of Buzzer and LED, it is the same as the way of reading Key Code as ASCII because it has to connect Port at Connector RS232 or TTL 232 to Port 232 of Board Controller. Moreover, it has to enable Jumper JP-BZ and JP-LED. The Baud Rate for sending Command is fixed at 9600 Bit/s.

The format of Command and Data that is used must be total capital letters and it must be in the format of ASCII Code that can be replaced by ASCII symbols such as '#'; or, it is replaced by ASCII Code that is equal to 0x23. There are 6 Byte Data for each Command; the last Byte is Enter that has no any symbol for calling; so, it must be replaced by ASCII Code = 0x0D. If user sends each Command correctly, user receives 3 Byte ASCII Command that is *OK to notify user to know that it is ready to receive the next Command. If user sends any Command incorrectly, there is no response from Board ET-TOUCH PAD 4x4 V2.

The format of Command is listed below;

1.) COMMAND 'BZ' (SOUND-ON/OFF)

This is Command ON/OFF Buzzer. If Data is set as '1', it enables Buzzer (ON) and it produces sound all the time. Or, if Data is set as '0', it disables Buzzer (OFF), there is no any sound. The format of Command is listed below;

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 Stop noise

'1': Buzzer ON Keep playing noise

Ex. Send Command ON-Buzzer for 1 second and then 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)

This Command produces Beep sound; in this case, it can set the length of Beep sound in the range of '0-9'. If setting Data = '1', the length of Beep sound is the shortest and it gradually increases the length of Beep sound according to the specific number. If setting Data = '0', the length of Beep sound is the longest. The format of Command is listed below;

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': Length of Beep sound

When setting '1', Beep sound is the shortest

'0', Beep sound is the longest

Ex. Send Command to produce Beep sound and set the length of Beep sound as level '5'

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

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

This Command ON-LED Status Key is to turn on all LED at once or turn on 4 LEDs in each time along of Colum or Row; it depends on the format that user chose. There are 9 formats of Command ON-LED as listed below;

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: Set format of ON LED as follows;

'0' = ON-LED ทั้งหมด 16 ดวง

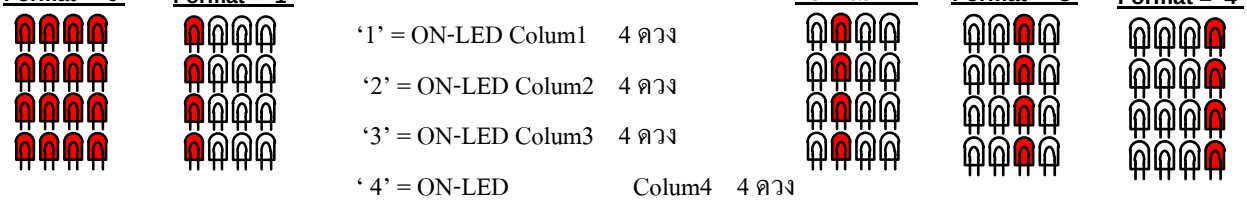
'1' = ON-LED Colum1 4 ดวง

'2' = ON-LED Colum2 4 ดวง

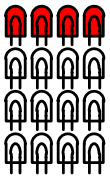
'3' = ON-LED Colum3 4 ดวง

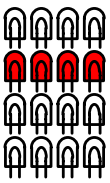
' 4' = ON-LED Colum4 4 ดวง

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



Format = '5' Format = '6'





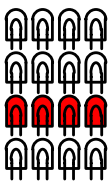
'5' = ON-LED Row1 4 ดวง

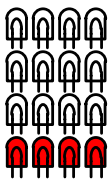
'6' = ON-LED Row2 4 ดวง

'7' = ON-LED Row3 4 ดวง

' 8' = ON-LED Row4 4 ดวง

Format = '7' Format = '8'





Ex. Send Command to turn on all LED (ON)

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

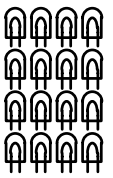
4.) COMMAND 'LD' (LED-DISABLE)

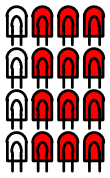
This Command OFF-LED Status key is to turn off all LED at once or turns off 4 LEDs in each time along of Colum or Row; it depends on the format that use chose. There are 9 formats of Command OFF-LED as follows;

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: Set format of OFF LED as follows;

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

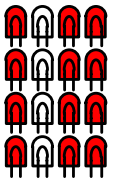
'5' = OFF-LED Row1 4 ดวง

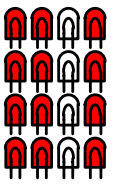
'6' = OFF-LED Row2 4 ดวง

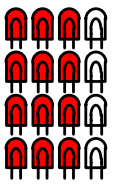
'7' = OFF-LED Row3 4 ดวง

' 8' = OFF-LED Row4 4 ดวง

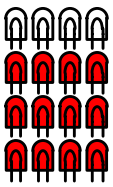
Format = '2' Format = '3' Format = '4'

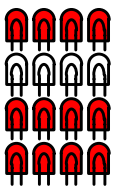




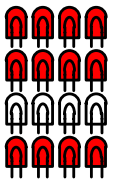


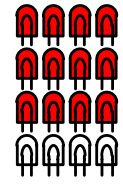
Format = '5' Format = '6'





Format = '7' Format = '8'





Ex. Send Command to turn off all LED at once

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

***NOTE**: When sending Commands consecutively, it always checks status of Respond Command between Commands that are sent out, or it may set the proper Delay between Commands because it protects commands that are consecutively sent out from overlaid and the transmission of data may be incorrect.*

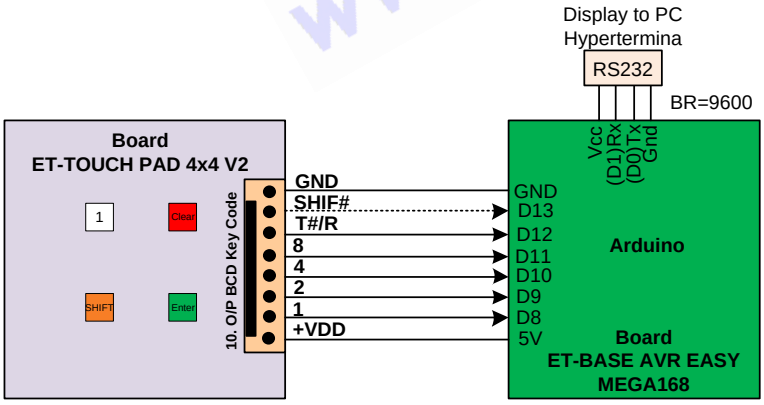
5. Example Programs and How to connect Circuit to read Key Code by MCU

This example illustrates how to read Key Code in the format of Binary BCD8421 and ASCII Code, it supports 4 series of MCU. If it is AVR EASY MEGA168, it uses Arduino Compiler; if it is AVR MEGA 61/128, it uses C-WIN AVR; if it is MCS51-AT89C51RE2, it uses C-Keil Compiler; and, if it is PIC8722, it uses CCS Compiler. There are 2 examples for each MCU; firstly, it reads Key Code in the format of Binary BCD8421; and secondly, it reads Key Code in the format of ASCII Code. It illustrates the operational detail of program as follows (Source Code is provided in CD).

Example of reading Key Code as Binary BCD8421

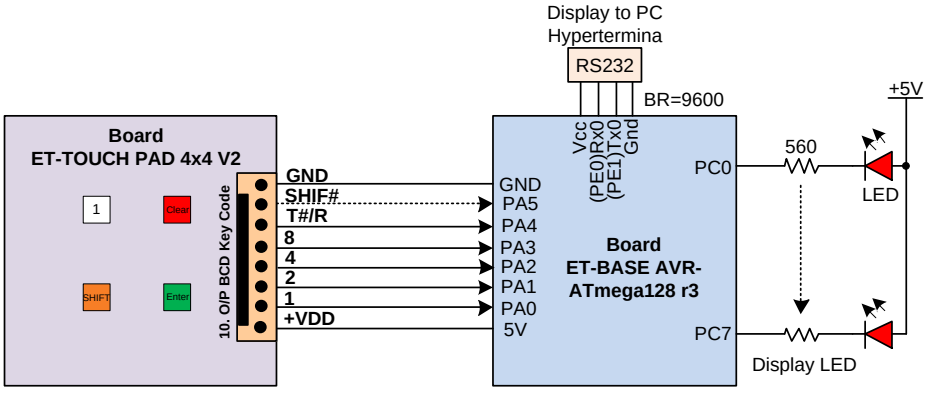
Ex1_Read_BCD_1Key: This example illustrates how to read Key Code as BCD8421. It uses single Key, so user can use Key F (SHIFT) as normal as other keys. For Pin SHIF#, it is not connected to MCU.

When user stars touching any Key, MCU will check Signal T#/R to check status of touching Key. If touched any Key (T#/R=0), it reads Key Code from Connector BCD8421 and sends out the Key Code through Port to display the result on the connective LED (except Arduino, it does not display any data in this part); and finally, it types the Key Code to display data at Hyper Terminal on PC. For example, if touched Key 5, LED shows the value as 0x05 and Hyper Terminal also shows message "Key_Code BCD = 0x05". Next, MCU will check Signal T#/R again to check the status of releasing Key (T#/R=1). When released Key completely, the program returns to check Signal T#/R to check status of touching the next Key. This is overall operation of the first example of each MCU. The example circuit that supports the first example is illustrated in the figure below;



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

Figure 5.1 shows how to connect circuit to read Key Code as BCD of Board ET-TOUCH PAD 4x4 V2 and ET-BASE AVR EASY MEGA168.



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

Figure 5.2 shows how to connect circuit to read Key Code as BCD of Board ET-TOUCH PAD 4x4 V2 and ET-BASE AVR ATmega128 R3.

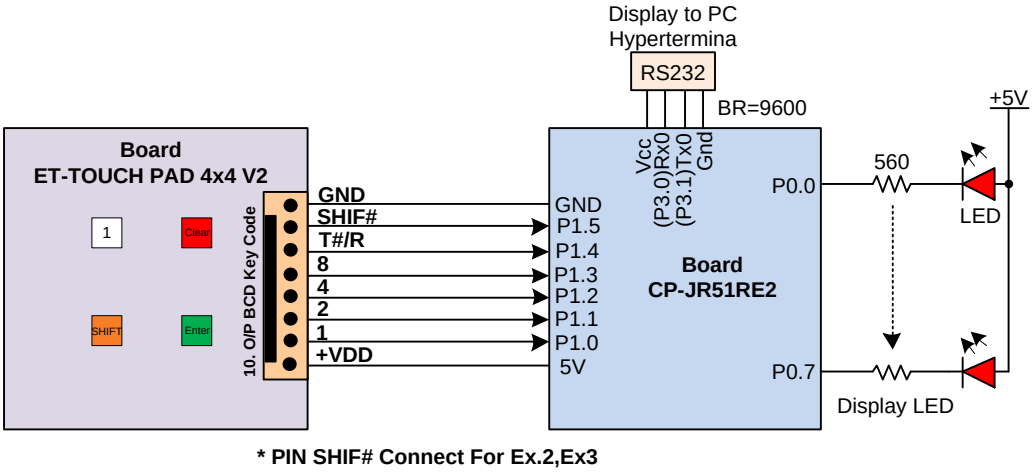


Figure 5.3 shows how to connect circuit to read Key Code as BCD of Board ET-TOUCH PAD 4x4 V2 and CP-JR51RE2.

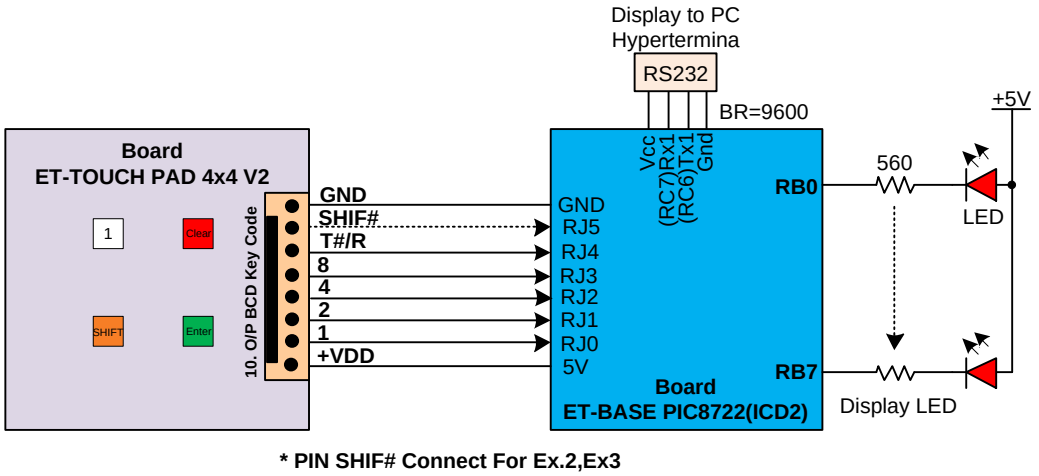


Figure 5.4 shows how to connect circuit to read Key Code as BCD of Board ET-TOUCH PAD 4x4 V2 and ET-BASE PIC8722 (ICD2).

Ex2_Read_BCD_ShifKey: This second example illustrates how to read Key Code as BCD8421, it is the same as the example 1 above but it presses Key F together with another Key. So, it still uses the same circuit as shown in the figure of the example 1 above but it has to interface Pin SHIF# to MCU.

When user starts touching any Key, MCU will check Signal T#/R to check status of touching Key. If touched any Key (T#/R=0), it reads Key Code from Connector BCD8421 and it checks if the value is KeyF (0x0F). If no, it sends out the Key Code through Port to display the result on the connective LCD (except Arduino, it does not display any data in this part); and finally, it types the Key Code to display at Hyper Terminal on PC as mentioned in the example 1 above. Next, MCU will check Signal T#/R again to check and wait for the status of releasing Key (T#/R=1). If the pressed Key is Key F, the program loops to read Key Code of another Key that is pressed together with KeyF. It checks Signal SHIF#; if it is 0, it means that another Key is pressed together with Key F; Program reads Key Code of the second Key from Connector BCD8421 and it sends out the Key Code through Port to display the data on the connective LCD(except Arduino, it does not display any data in this part); and finally, it types the Key Code to display data at Hyper Terminal on PC. For example, if touched KeyF+Key3, the value that is shown on LED is 0x03 (LED Bit7 is lit up to display the state of pressing Key Shift); and Hyper Terminal also displays the message “Key_Code BCD = Shift+0x03”. Next, MCU checks Signal SHIF# and T#/R to check and wait for the status of releasing both Keys (SHIF#,T#/R=1).

Ex3_Application_BCD_ShifKey: This example 3 illustrates how to apply the operation by pressing Key F together with another Key, the circuit's connection in the part of ET-TOUCH PAD and MCU is still the same as the example 1 above because it has to connect PIN SHIF# to MCU. In the part of display, it uses 6x2 LCD Display. For more information of the connection between LCD Display and MCU, please look in Source Code.

The operation in the part of reading Key Code is the same as the example 2 above, but it is different in the part of Program that responds to pressing Key only. In the part of LCD Display, it shows the message as shown in the figure 5.5 if it does not press any Key.

Figure 5.5

When pressing 1 Key that is not KeyF such as Key 8, LCD Display shows the message as shown in the figure 5.6.

Figure 5.6

If touching KeyF together with another key (Touching 2 Keys) such as Touch Key +Key5, LCD Display shows the message as shown in the figure 5.7. Remember, the example program allows touching Key F together with Key0-Key9 only.

Figure 5.7

Example of reading Key Code as ASCII CODE

When reading Key Code as ASCII CODE, please look at Table KEY CODE of ET-TOUCH PAD 4x4 V2 in the box of ASCII CODE in order to read value of Key Code of each Key that is pressed. User can read the value from Connector RS232(3) or TTL232(7) of ET-TOUCH PAD 4x4 V2 and it sets the fixed Baud Rate as 9600 bit/s. When connecting Cable, it has to cross Cables of RX,TX of MCU in order to connect with TX,RX of Board ET-Touch PAD, respectively. Please look at the example, it connects Cable at Connector RS232(3); so, it has to set Jumper 'JP-232(4)' of Board ET-TOUCH PAD to the position of RS232. The reason why it has to choose this Connector is because this Board MCU has already had Connector RS232 through Line Driver. Or, if user connects with Pin 232 (Uart) of MCU directly, it has to choose Connector TTL232(7) of ET-TOUCH PAD instead.

Ex1_Read_ASCII_1Key: This example illustrates how to read Key Code as ASCII. It presses single key, so user can use Key F (SHIFT) as normal as another keys; moreover, it uses Interrupt to setup interval of receiving data. In the part of Display, it shows ASCII Code in the format of HEX through Port of MCU that is connected with LED.

When user starts touching any Key, program skips to receive incoming data and store until all 3 Byte Data is completed. Next, it checks if the last Byte is 0x0D; if yes, it means that it received all 3 Byte Data completely and it is correct. Next, it returns to check if the first Byte that received is ASCII 'P"; if yes, it means that it touched Key; so, Program sends the second Byte Data that is ASCII CODE of the touched Key to display in the format of Hex through Port of the connective LED. For example, if touched Key 7, ASCII Code in the format of Hex is 0x37. Finally, Program returns to read Data to check state of releasing Key. If the first Byte Data is ASCII 'R', it means that it released Key completely; so, Program returns and waits for reading Data of the next touch (NOTE: For the example of Arduino, it can display the result of touching Key in the range of Key0-Key9 only). For the circuit connection that supports this example, it is shown in the figure below;

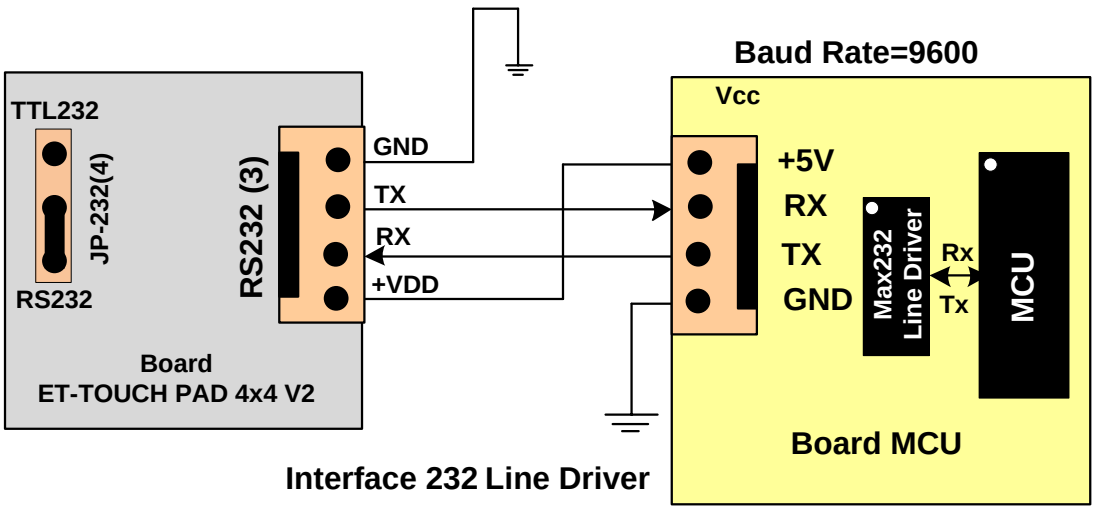


Figure 5.8 shows how to connect circuit to read Key Code as ASCII of Board MCU and ET-TOUCH PAD 4x4 V2 (through Line Driver).

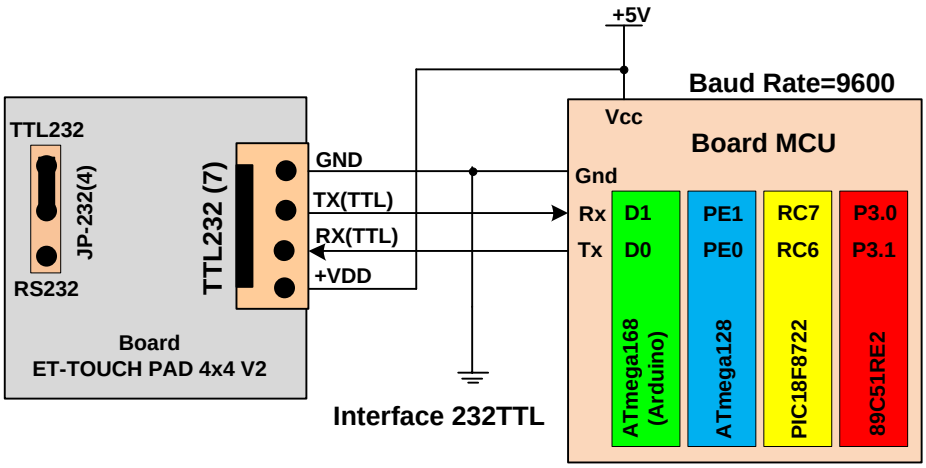
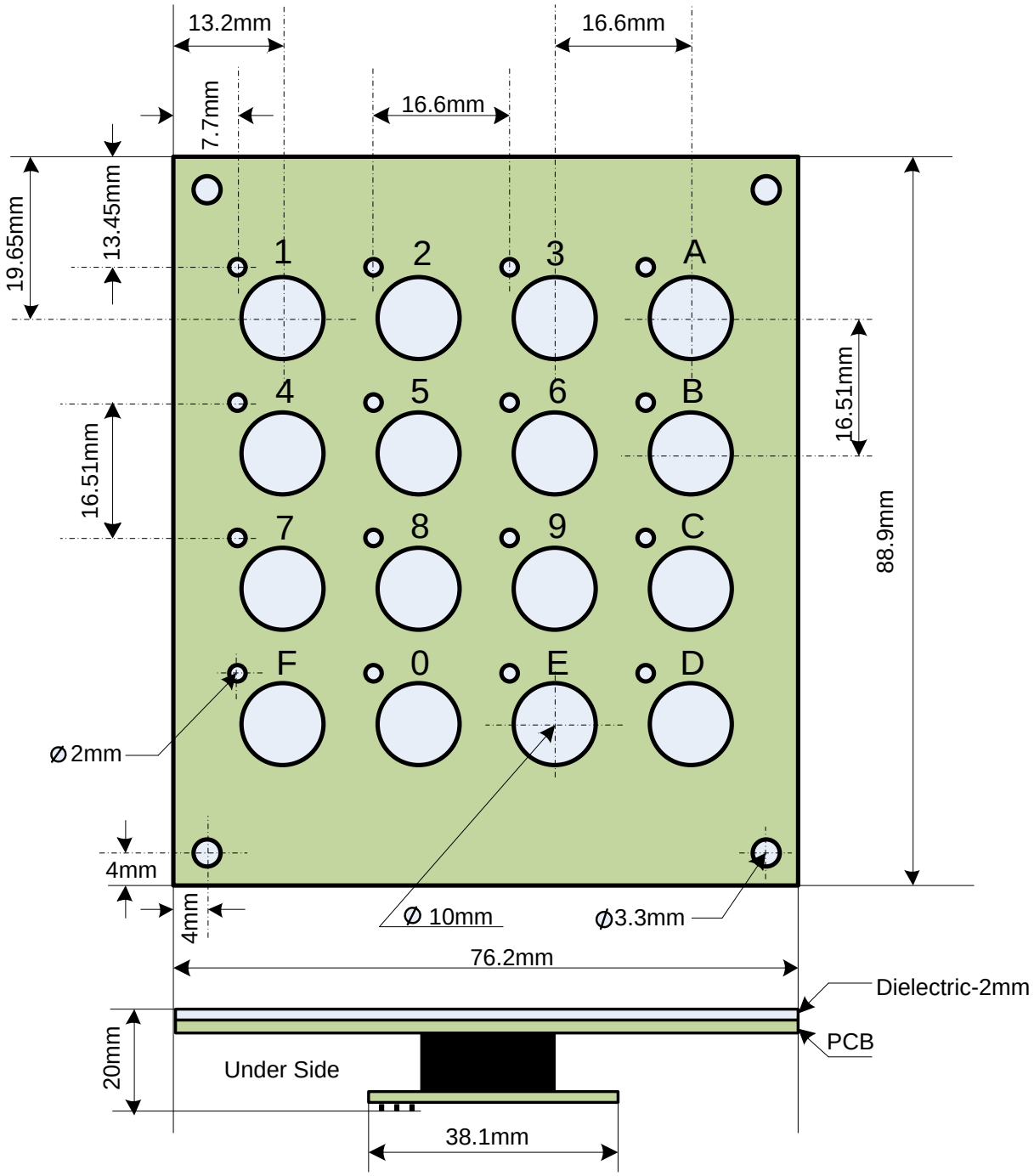


Figure 5.9 shows how to connect circuit to read Key Code as ASCII of Board MCU and ET-TOUCH PAD 4x4 V2 (TTL).

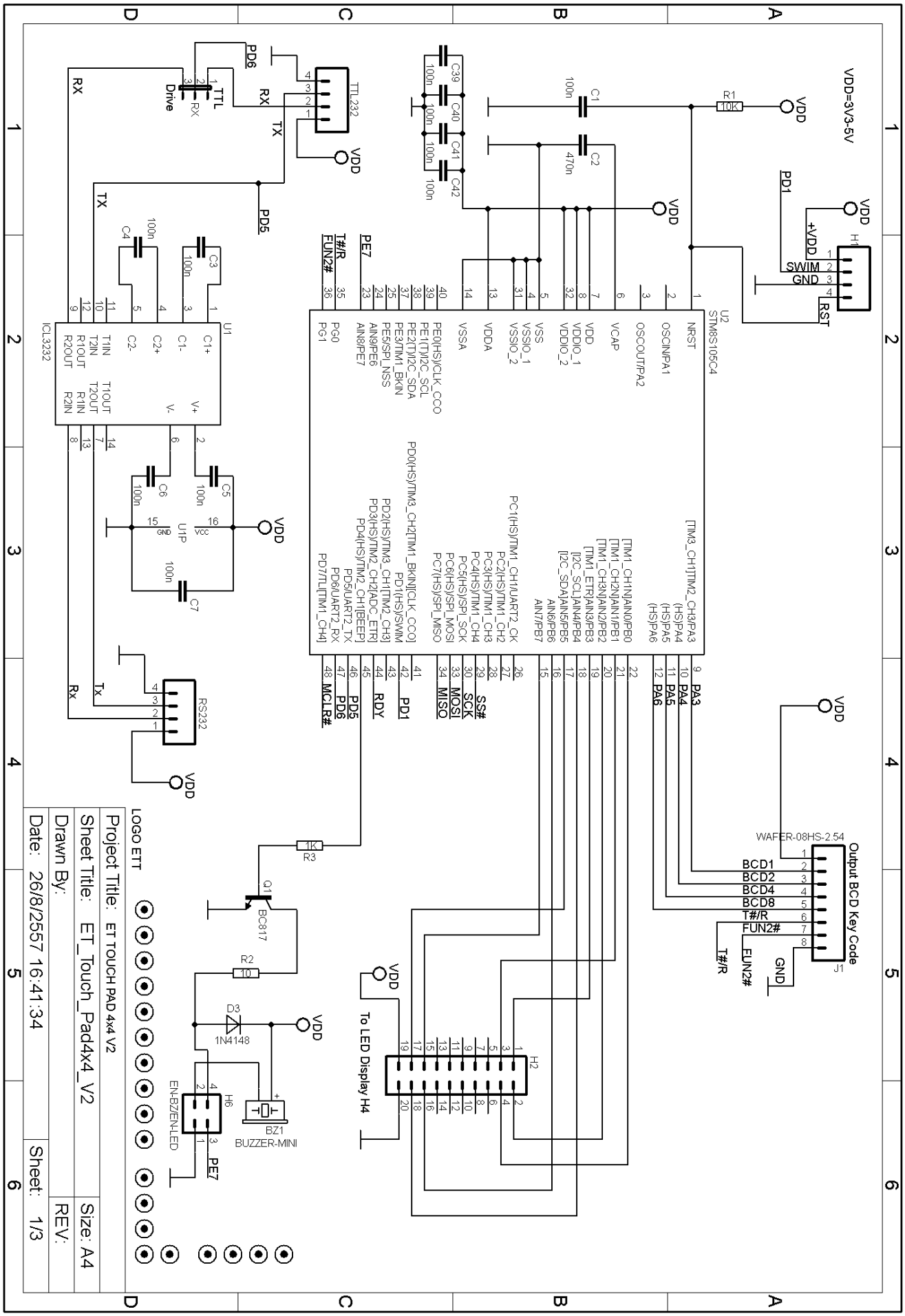
Ex2_Read_ASCII_ShifKey: This example illustrates how to read Key Code as ASCII. It presses Key F together with another Key and it uses Interrupt to setup interval of receiving data. In the part of Display, it shows ASCII Code in the format of HEX through Port of MCU that is connected with LED.

When user starts touching any Key, program skips to receive incoming data and store until all 3 Byte Data is completed. Next, it checks if the last Byte is 0x0D; if yes, it means that it received all 3 Byte Data completely and it is correct. Next, it returns to check if the first Byte that received is ASCII 'P"; if yes, it means that it touched Key. It checks the received Data in the second Byte that is ASCII Key Code of the touching Key; it checks if the value is equivalent to ASCII CODE 'F'; if no, it means that it presses single Key, Program sends the second ASCII CODE Byte to display in the format of Hex through Port of the connective LCD. For example, if touched Key 0, the ASCII Code in the format of Hex is 0x30 (NOTE: For the example of Arduino, it can display the result of touching Key in the range of Key0-Key9 only). If the Key that is touched is Key F, Program loops and waits for receiving the second Key that is pressed together with Key F. When touched the second Key, Program checks if the last Byte Data is equal to 0x0D; if yes, it returns to check if the first Byte is ASCII 'F'; and if yes, it means that it pressed the second Key together with Key F. So, Program sends the second ASCII CODE Byte of the touched Key to display in the format of Hex through Port of the connective LCD. For example, if touched KeyF+Key9, the ASCII Code in the format of Hex is 0x39 (NOTE: Other Keys that can be pressed together with Key F in this Program must be in the range of Key0-Key9 only). Finally, after touched completely, Program returns to read the Data again to check Data of releasing Key. If the first Byte Data is ASCII 'R', it means that it has already released Key; so, Program returns and waits for reading Data of the next touch. The circuit connection that is used to test this program is the same as the example 1 above.

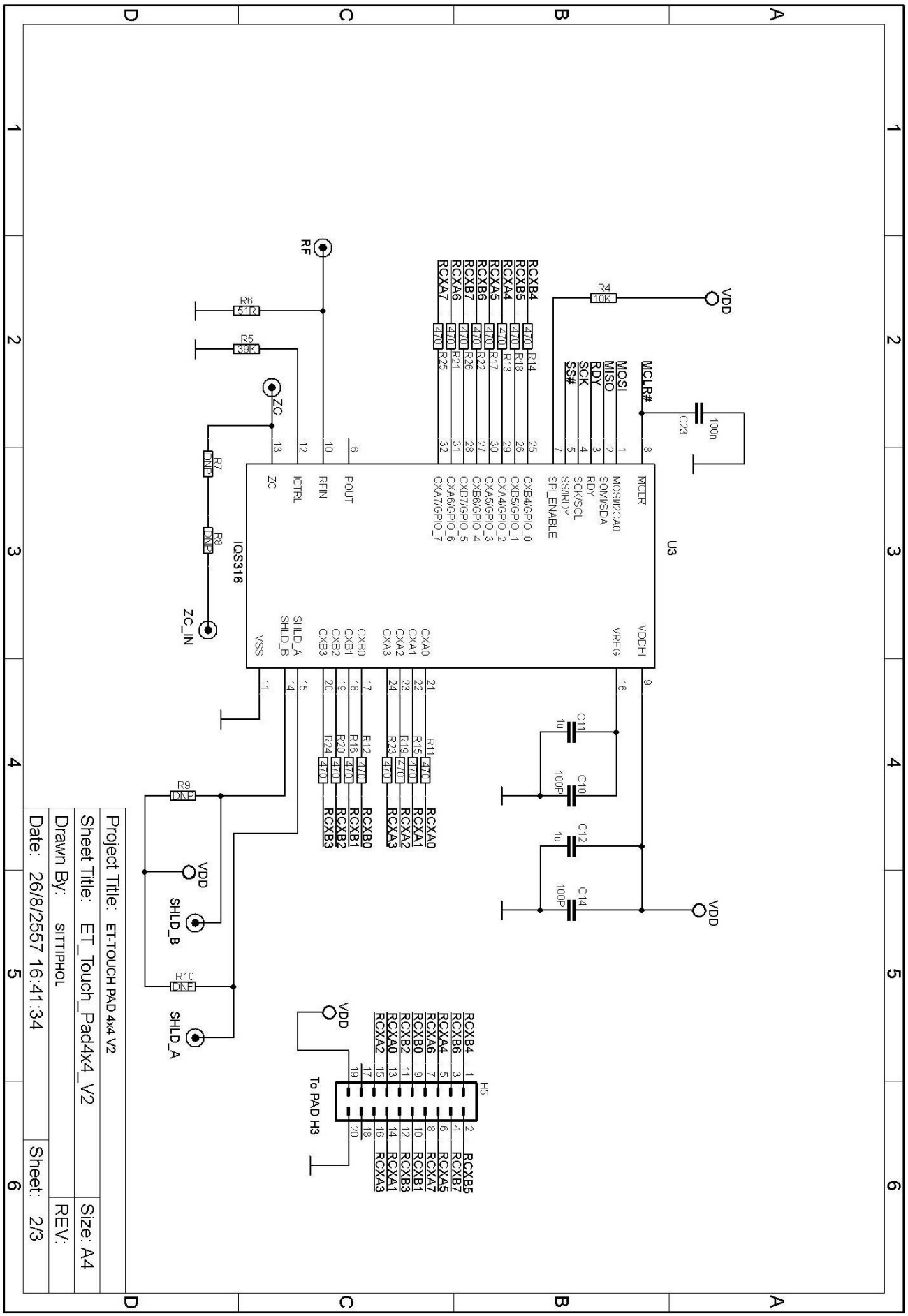
PCB LAYOUT



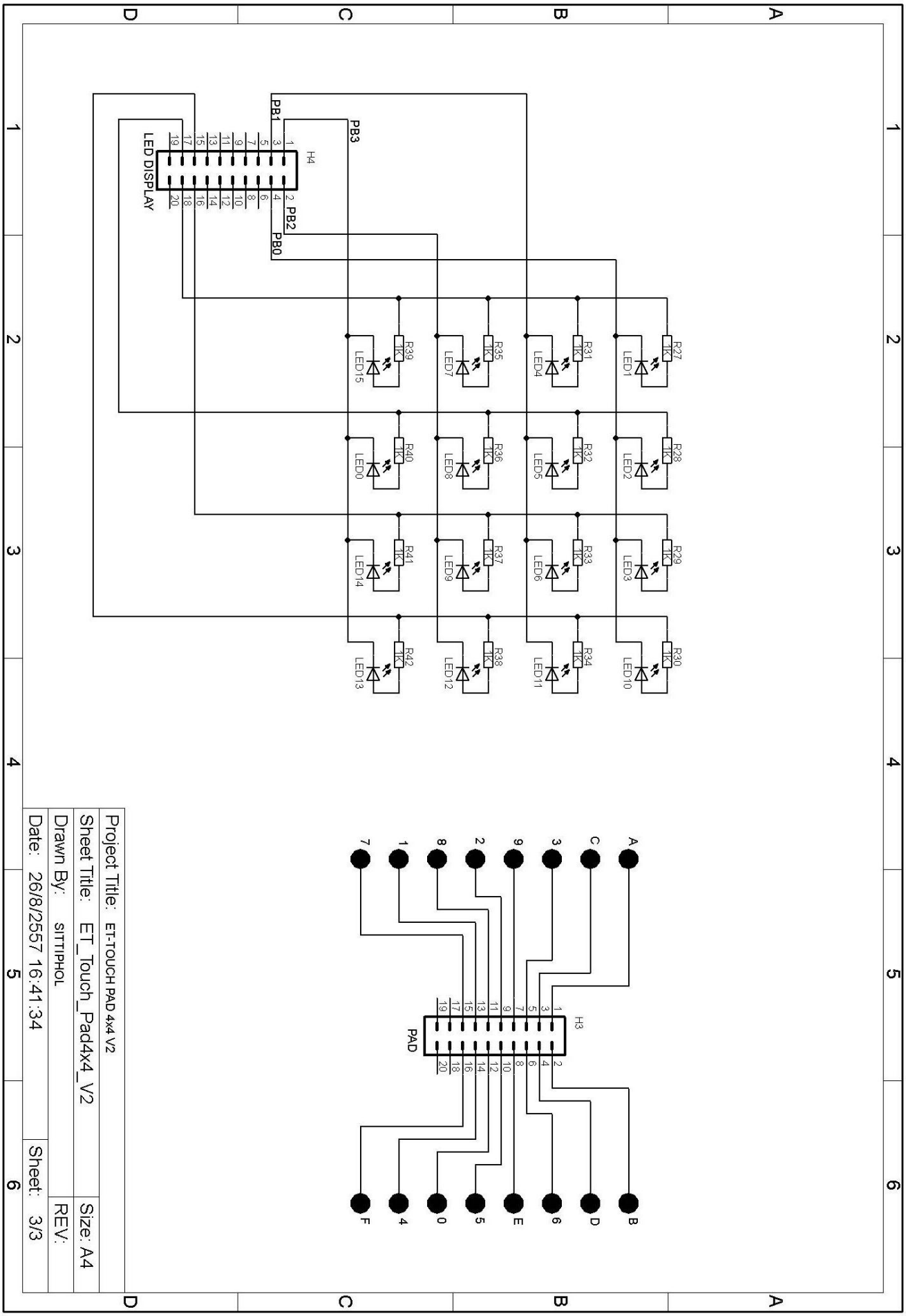
PCB LAY OUT



ET-TOUCH PAD 4x4 V2 Sheet1



ET-TOUCH PAD 4x4 V2 Sheet2



ET-TOUCH PAD 4x4 V2 Sheet3