ET- TFT240320TP-3.2

1. Specifications of Board ET-TFT240320TP-3.2

- Display Module TFT LCD Color + Touch Screen with 240x320 Pixel
- 3.2" TFT Screen size
- Use Single Chip Driver No.ILI9320
- 65536 Colors (RGB = R:5Bit-G:6Bit-B:5Bit)
- For LCD Controller, it uses 16-bit data/address Interface + 6BIT Control Pin.
- For Touch Screen Controller, there are 2 Interfaces types by using Dip SW. Firstly, it is SPI Interface through Chip Touch Screen Controller #ADS7846 (12BIT ADC); and secondly, it directly interfaces through Pin X-,X+,Y-,Y+ and Pin ADC of MCU (however, it is difficult to write program for controlling).
- If do not use Touch Screen, user can control the part of LCD only.
- Should interface with MCU that has I/O-Port at least 28PIN to control operation of board completely.
- Be compatible with MCU that uses both 5V or 3.3V Power Supply (see more information in "Application")
- Use Pin Header 2x20 Connector with 2.54mm. pitch
- Use DC +5V Power Supply

2. Features and Structure of Board ET-TFT240320TP-3.2

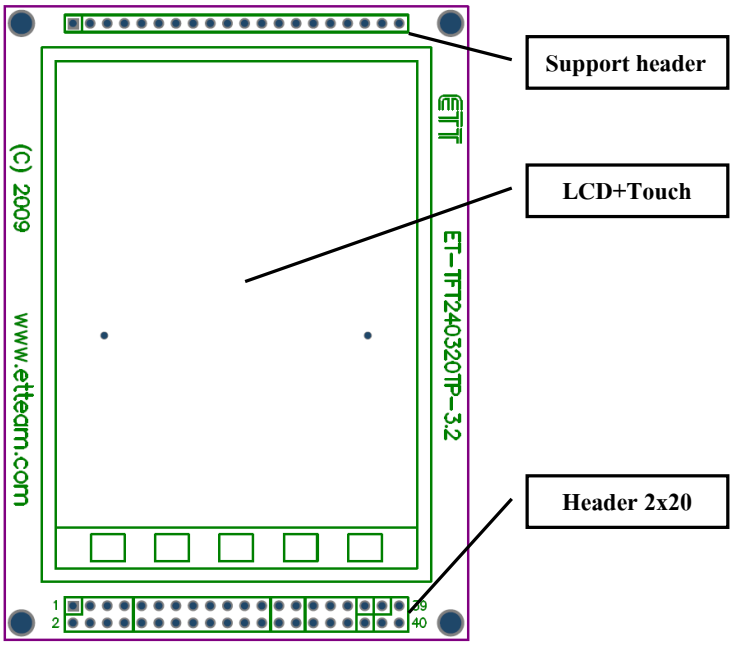


Figure 1(A) Front Board

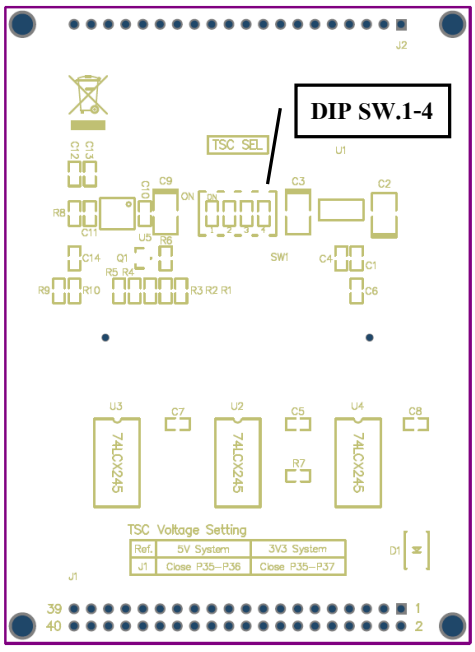


Figure 1 (B) Back Board

- *Support Header*: It is available hole to interface Connector and support the board above.

- *LCD+Touch:* It is area of 240x320 Pixel LCD Screen and on the screen is covered by Resistance Touch Screen.

- *Header 2x20*: It is 2x20 Pin Male Connector to interface signal from MCU and control operation of LCD and Touch Screen. The detail of theses pins are shown below.

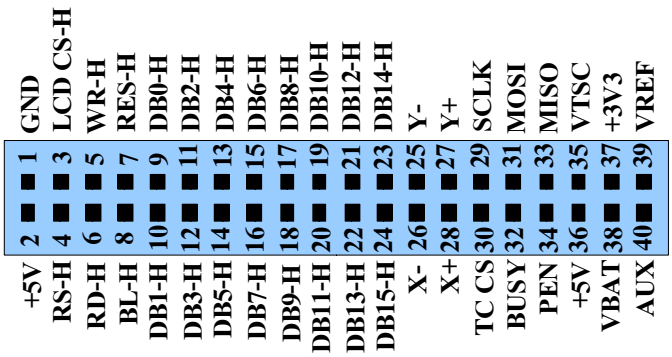


Figure 2 shows position of signal pin at Header 2x20(front view as shown in the figure 1(A)).

Details of Pin for Controlling LCD

No. PIN	PIN-NAME	I/O	Description
1	GND	Power LCD	Pin Ground
2	+5V	Power LCD	+5V Power Supply for board
3	LCD CS-H	I	Pin Chip-Select : Low - ILI9320 is chosen and allowed to send data High – Be unable to connect with ILI9320
4	RS-H	I	Pin Register-Select : Low- Access to Index(IR) Register or Status Register(SR) High- Access to Control Register(Address 00H-98H)
5	WR-H	I	Pin Write Strobe: To write data when received signal Low
6	RD-H	I	Pin Read Strobe: To Read data when received signal Low
7	RES-H	I	Pin Reset to Initial ILI9320 when received signal Low
8	BL-H	I	Pin Black Light: Black Light of LCD is ON, when received signal High
9-24	DB0-DB15	I/O	Pin data bus Bi-directional 16 bit sends data or points address position of Register

Details of Pin for controlling Touch Screen

No. PIN	PIN-NAME	I/O	Description
25*-28 *	Y-,X-,Y+,X+	I	Pin Y-,X-Y+,X+ Position Touch Screen: When it controls Touch Screen not through Chip ADS7846, must shift all DIP SW.1-4 on the back board to OFF position.
29	SCLK	I	It is Pin DCLK of ADS7846 to Synchronizes serial data I/O.
30	TC CS	I	It is Pin CS of ADS7846when received signal Low. It Enable Serial I/O Register of Chip to start running.
31	MOSI	I	It is Pin DIN of ADS7846 when Pin CS is Low and data is latches at the rising edge of signal DCLK.
32*	BUSY	O	It is Pin BUSY of ADS7846 and it is High impedance when Pin CS is High (option).
33	MISO	O	It is Pin DOUT of ADS7846. When Pin CS is Low, data is shifted at the falling edge of DCLK and this Output is high impedance when CS is High.
34	PEN	O	It is Pin PENIRQ of ADS7846when touched the Touch Screen and it gives signal Logic as Low (already Pull-Up R10K on board)
35-37	VTSC,+5V, +3V3	Power Touch-Screen	These 3 pins are used to choose Power Supply for ADS7846. If using with MCU 5V, it needs to interface Pin VTSC(35) with Pin +5V(36); on the other hand, if using with MCU 3.3V, it needs to interface Pin VTSC(35) with Pin +3V3 (37).
38*	VBAT	I	It is Pin Vbat of ADS7846 that is used in the part of Touch Screen
39*	VREF	I/O	It is Pin Vref of ADS7846 that is used in the part of Touch Screen.
40*	AUX	I	It is Pin AUX input to ADC of ADS7846 that is used in the part of Touch Screen.

(*) = Pin is not used, refer to the ETT examples.

-DIP SW.1-4 on the back board: There are 4 DIP SW. If using Touch Screen in the format of SPI Interface through Chip Touch Screen Controller #ADS7846, user must shift all 4 DIP SW. to ON position (Default) . The example program is also written to support this operation mode. However, if directly interfacing Pin X-, X+,Y- , Y+ with Pin ADC of MCU (not use ADS7846), it needs to shift all 4 DIP SW. to the opposite Default position.

3. How to interface Board ET-TFT240320TP-3.2 with MCU

The method to interface Board TFT240320T3.2 that is described in the topic 3.1) and 3.2) supports ETT example. It uses MCU AVR MEGA128, PIC 18F8722 to interface with MCU 5V and then uses MCU ARM7 LPC2138 to interface with MCU 3.3V. Both connecting types of these circuits can be adapted and applied to interface with other MCU numbers or families. ***However, be careful especially Pin VTSC, user has to interface it with either Pin +5V or +3V3 according to the exactly used Power Supply of MCU. Otherwise, it makes MCU damaged.*** For example, if MCU runs at 5V Power Supply, must interface Pin VTSC with Pin +5V.

3.1) How to interface with MCU 5V

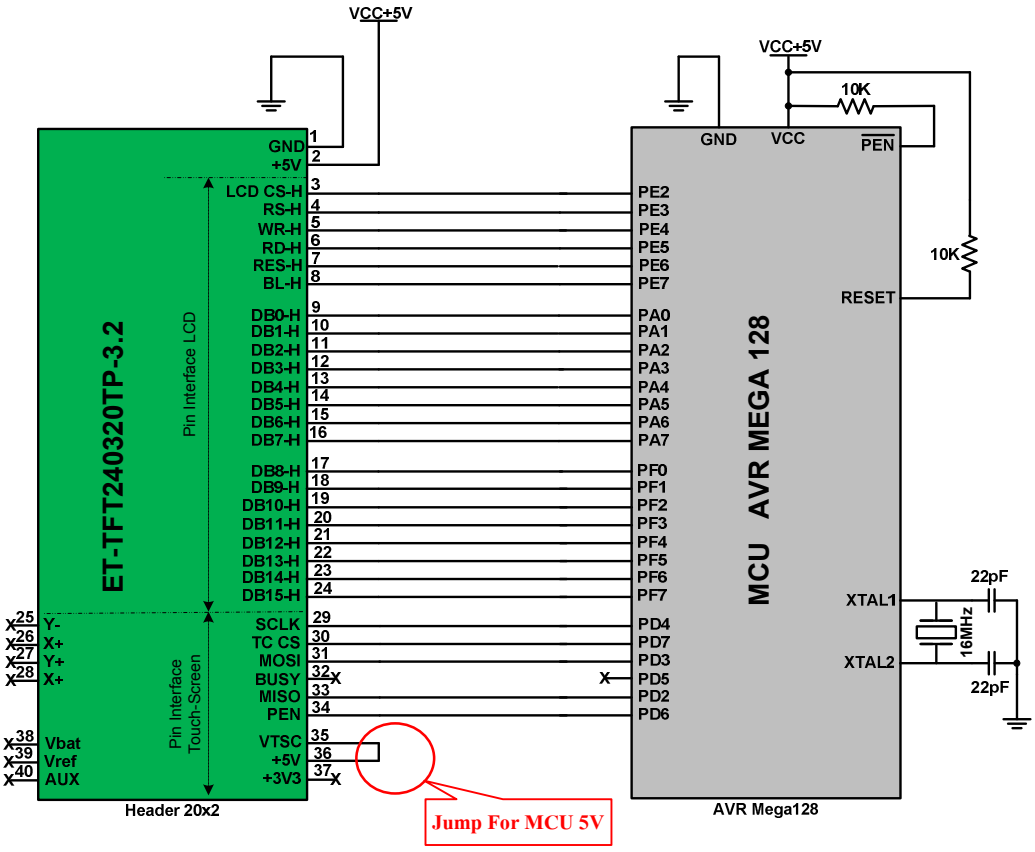


Figure 3 shows example of interfacing Board ET-TFT240320TP-3.2 with MCU AVR #Mega 128 (5V).

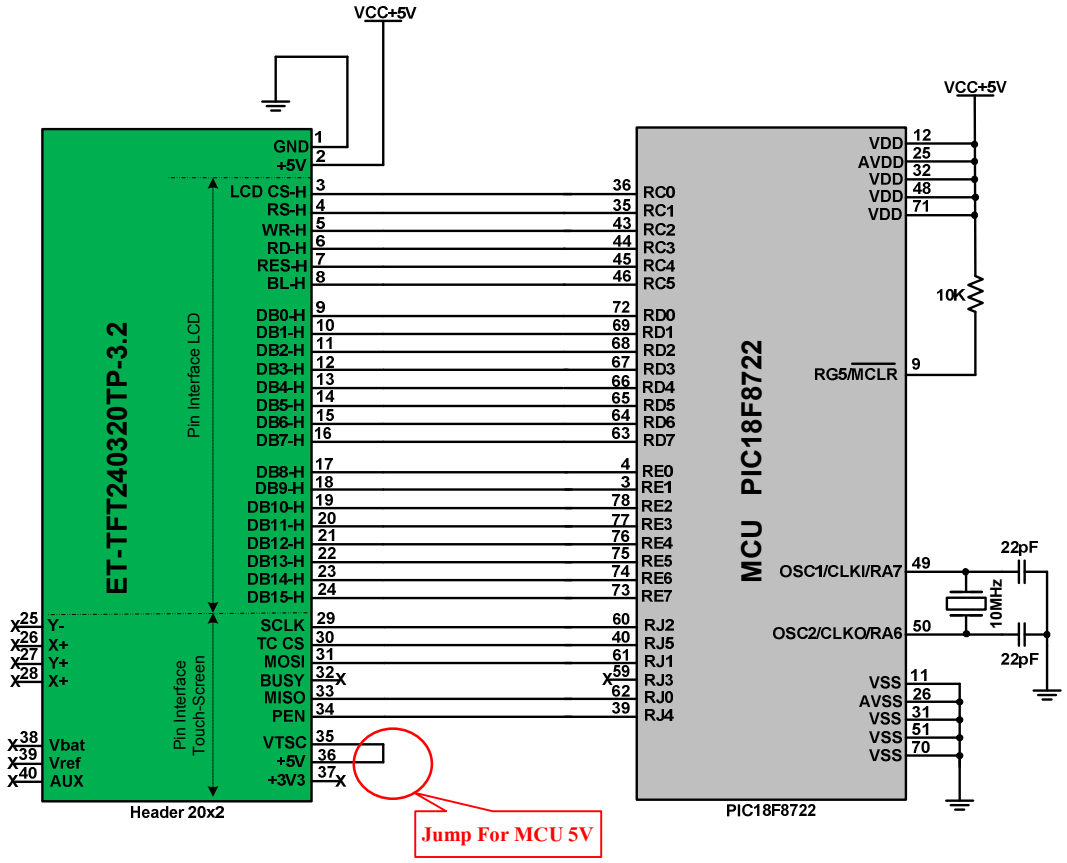


Figure 4 shows example of interfacing Board ET-TFT240320TP-3.2 with MCU PIC#18F8722 (5V).

3.2) If using with MCU 3.3V, please notice that Pin P0.2,P0.3,P0.11,P0.14 of MCU are interfaced with R Pull-Up because this Pin Port is Open Drain. However, if MCU that is used has not any Pin Open Drain, it is unnecessary to interface R Pull-up

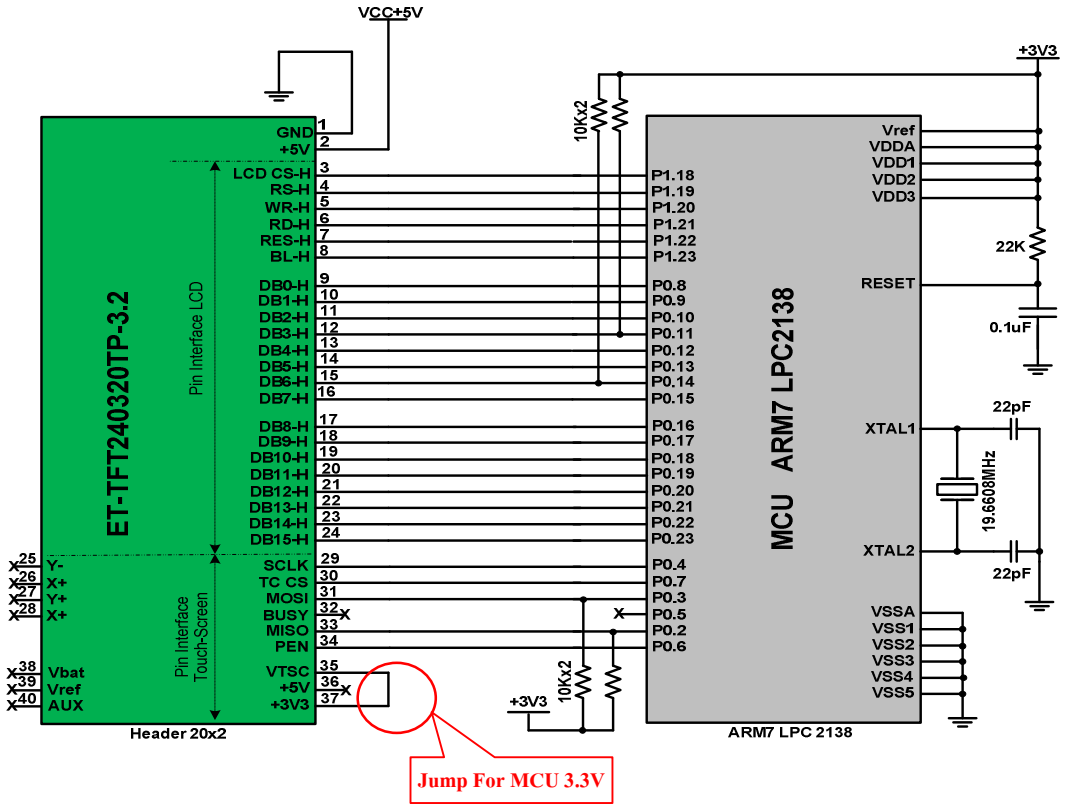


Figure 5 shows example of interfacing Board ET-TFT240320TP-3.2 with MCU ARM7 #LPC2138.

4. Principle of controlling LCD and Touch Screen

The method to write program to connect with Board ET-TFT240320TP-3.2 is to be easier if separating the control into 2 parts. Firstly, the part of LCD that is Parallel Interface; and secondly, the part of Touch Screen that is SPI Interface. In this case, we refer to the ETT examples that are provided in CD to describe the principles of writing program to control operation. Each example in CD uses the initial principle to connect with board similarly but it only differs in the format of results that are displayed on the screen. We can summarize the principle of both parts as follows;

4.1) Interface Control LCD: In the part of command that is used to control LCD, user should read more information from Data Sheet **"GLCD_ILI9320.pdf"** that is provided in CD. The principle of sending command or data to LCD is described below;

First of all, user needs to know that Board TFT240320TP-3.2 is fixedly designed in the format of Mode 16Bit Interface (1 transfer/pixel) 65,536 colors. It means that user can send 16Bit data once a time to LCD and it makes 1 dot or 1 pixel takes place on the LCD Display. So, user needs to understand the arrangement of colored bit data well before sending data because it makes user can set colors to display on LCD Display correctly as follows;

- Arrangement of 16 Bit colored bit data

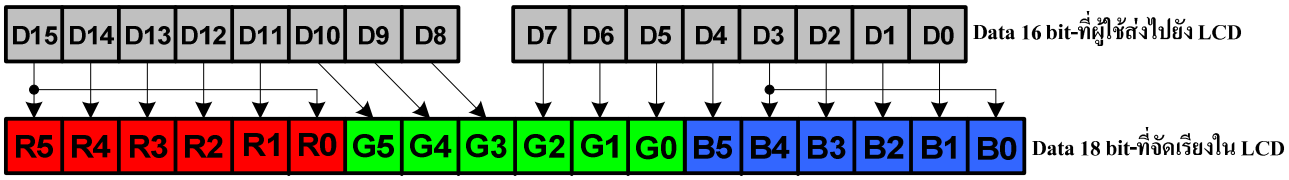


Figure 6 shows the arrangement of 16 bit colored data (1 transfer/pixel).

From figure above, it arranges the colored bit data from higher bit to the lower bit that is RGB according to the ETT example. However, user can change the new arrangement of this colored bit data to be BGR by using command **Entry Mode(R03H)**. Read more information in Data Sheet.

If user wants to send data to display dot on LCD Display, need to mix colors by self by referring to the colored bit arrangement in the figure above. In this case, **Data D15-D11(5bit)** is the part of red color; **Data D10-D5(6bit)** is the part of green color; and **Data D4-D0(5bit)** is the part of blue color. When user has already sent the data into LCD, the data will be arranged into the new format of 18Bit data automatically as shown in the figure above. The color is arranged from the dimmer to brighter; it means that it is arranged from the lower bit to higher bit. For example, if user wants the brightest red color, it is **Data = 0xF800**; or if user wants the dimmest green color, it is **Data = 0x0020**. Moreover, if user wants other colors more than these 3 main colors, user needs to set bit data that is in the range of each color properly by self and user will get the required colors. For example, if user wants white color, it is **Data = 0xFFFF**; or if user wants black color, it is **Data = 0x0000**.

- Timing Diagram of sending command and data

After user understood the arrangement of colored bit data well; next, we will describe how to send command and data of command to LCD. In this case, we can summarize the procedures as shown in the Timing Diagram below;

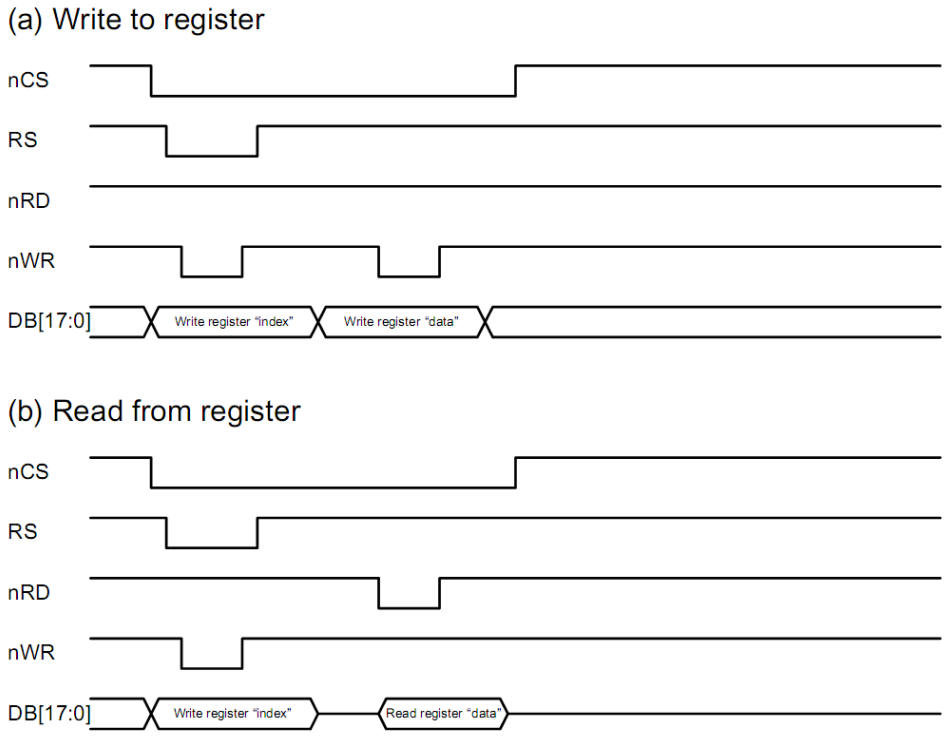


Figure 7 displays Timing Diagram to Write/Read Data to LCD.

From the Timing diagram above, we only mention about the part of Write Data. We do not mention about the part of Read Data because the ETT example does not use Read Data from LCD but it uses Delay instead; however, it also can control LCD.

First of all, notice that the Write Data in the Timing Diagram above, each command that is sent to LCD has 2 data sets. Firstly, it is 8Bit command set; in this case, it is the Address position of Register Index of the command. For example, if it is Command **"Write Data to GRAM(R22h)"**, the Address position of Register Index for this command is **0x22**. Secondly, it is 16Bit data set of the command. For example; if already sent Command **"0x22h"**, the second data set that will be sent is colored data. In this case, if user wants to display 1 white dot on LCD Display, need to send data as **"0xFFFF"**.

The feature of sending command "Control LCD" for other commands is similar. Next, we will mention how to set Pin Control for sending the command set and data set to LCD properly. In this case, we can summarize the procedures according to the Timing Diagram as follows;

Summarize the procedures for sending command and Data to Control LCD

1) Set Pin RD, CS to be 1

2) Set Pin CS to be 0

3) Send 8Bit Command Set to 8Bit Data Bus Lower and then send 0x00 to 8 Bit Data Bus Upper

4) Set Pin RS to be 0

5) Set Pin WR to be 0

6) Set Pin WR to be 1

7) Set Pin RS to be 1

After sent the command set successfully, user needs to send the data of the command as follows;

8) Still set Pin CS to be 0 and still set Pin RS, RD to be 1

9) Send 16Bit Data of the command set to all 16Bit Data Bus.

10) Set Pin WR to be 0

11) Set Pin WR to be 1

12) Set Pin CS to be 1

From the procedures above, if user wants to send other commands, needs to return to the first step again. For writing program, user can write function to receive command and data into the function simultaneously and then send command and data as described in steps above. Moreover, user can write program according to the ETT example that divides into 2 functions; function for sending command and function for sending data.

4.2) Interface Control Touch Screen: In the part of this Touch Screen, it separates the Control from LCD. There are 2 types of Interface for controlling. Firstly, it directly interfaces Pin Y-,Y+,X-,X+ with Pin ADC of MCU and then writes program to control reading by self. However, it is quite difficult to write program because user needs to understand the operating principles of Touch Screen well, otherwise it makes user can write program incorrectly. In this case, we don't recommend user to use the Interface type.

In this case, we will recommend the suitable Interface type that is corresponding with the ETT example; it is to interface through Chip ADS7846. If users chooses this Interface type, needs shift all DIP SW.1-4 on the back board to ON position to connect Pin X-,Y+,X-,Y- of Touch Screen with Chip ADS7846 (normally, it has already been set at the Default position). If interfacing by Chip ADS7846, it uses SPI Interface between MCU and this Chip. User can see more information regarding the communicating between data and this Chip to read-write the data position of Touch Screen from Data Sheet of "**Touch_ADC7846N.pdf".** First of all, we will mention about the cooperation between Touch Screen and ADS7846.

- The cooperation between Touch Screen and ADS7846 for reading position of Touch Screen begins when user touches the Touch Screen. It makes the Chip ADS7846 start converting signal Analog that received through Pin X+,Y+,X-,Y- and then send the Digital value through Pin Serial Data Out. The ADC value that is read is 12BIT; so, the the value that is read on the X and Y axis is in the range of 0-4095. While touching Touch Screen, Pin PENIRQ of Chip sends Signal Interrupt Logic "0" for a while; so, if user writes program, needs to read the status of this Signal Interrupt to wait for checking whether Touch Screen is being touching or not. It protects data from repeatedly reading the position data of

Touch Screen all time. It only reads when Touch Screen is touched, so it makes program can operate in other parts conveniently.

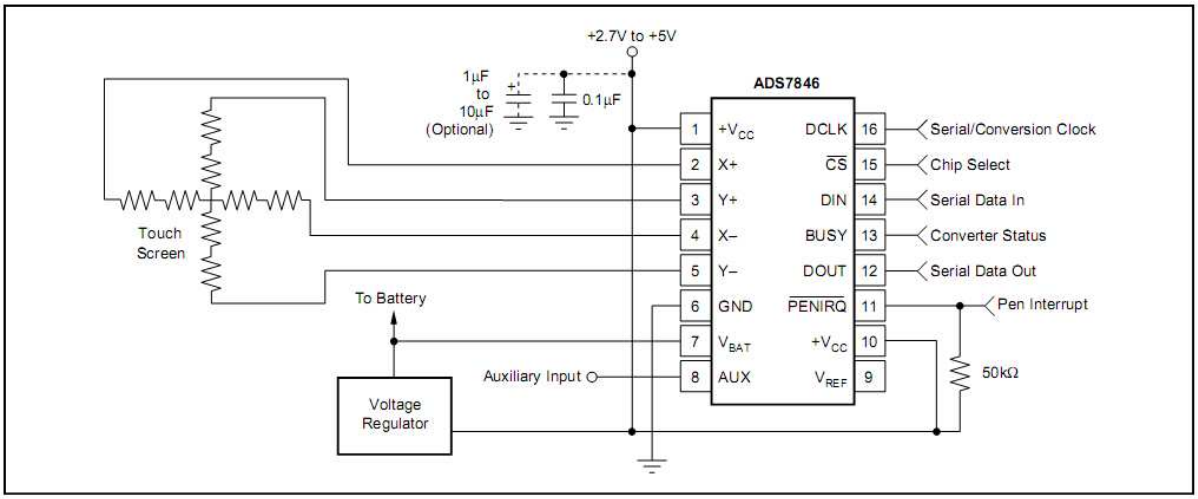


Figure 8 displays connecting circuit in the part of Touch Screen and ADS7846.

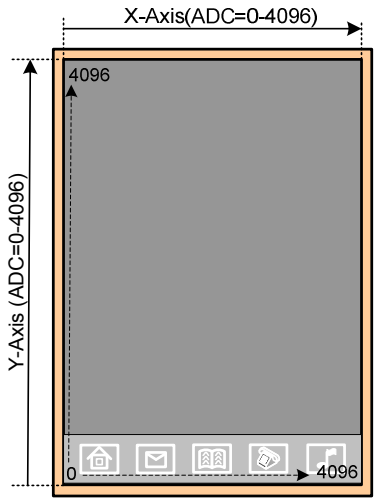


Figure 9 displays the extent of Touch Screen and ADC value that takes place along X and Y axis.

From the figure 9 above, it displays the directions along X and Y axis of Touch Screen, and the ADC value that will be read along the positions if touching screen. Normally, the lowest value of ADC value that is read through ADS7846 is 600 (not 0) approximately, it is the Offset value of the screen. Each screen reads the initial value unequally, so user always needs to write program to calibrate the screen first; in this case, user can copy ETT example to use instantly. It uses the matrix principle to calculate the coefficient that came from the calibrating of user. There are 3 points in the example program that user needs to touch. If user touches the point according the marked position, it makes the operation after calibrating high accuracy.

- Timing Diagram for reading data through ADS7846: After user knew and understood the initial principles in the part of Touch Screen well, next we will mention how to read the ADC value from Touch Screen by using Chip ADS7846. First of all, user needs to understand that the ADC value that is read from the Touch Screen is not the exact address position to refer to the position on LCD Display. User needs to use algebraic equation to find the exact position on the LCD Display. When got the exact address position successfully, user can replace the result in the command to control position on the LCD Display. User cal see all these procedures in the ETT example.

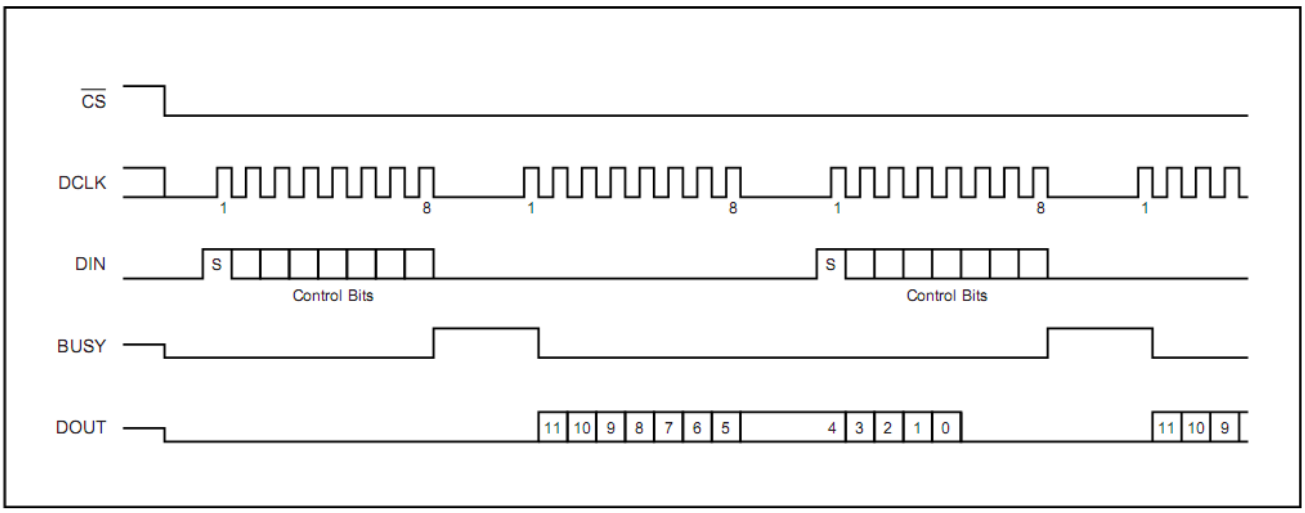


Figure 10 displays Conversion Timing Diagram, 16 Clock-per-Conversion, 8bit bus Interface.

This Timing Diagram is the process of reading the ADC value from CD Touch Screen through Chip ADS7846. It uses MCU Interface and SPI ADS7846 corresponding with the ETT example. The SPI Communication in the given example uses Pin I/O of MCU to build by self without using Module SPI internal MCU, so user can edit and modify program easily. The principle of sending SPI data is to send data to Pin MOSI(Dout) 1 bit and then follow by 1 Clock, the data will be shifted into Chip 1 Bit. Meanwhile, the Chip will also shift out data to Pin MISO (Din) 1 Bit, It is the data that will be stored and read. Refer to the given example, function **tcs_wr()** is used to write and read 1 Byte(8 Bit) serial data. When user has already built this function; next, we will send the Control Byte, read the ADC value and then store it through this function. The procedure of reading the ADC value from Touch Screen is described as below.

Before reading the ADC value from ADS7846, user always needs to send the Control Byte to ADS7846 to configure the specifications into Chip first. The feature of Control Byte is shown below;

Bit7(MSB)	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
S	A2	A1	A0	MODE	SER/$\overline{\text{DFR}}$	PD1	PD0

Figure 11 displays Control Byte of ADS7846.

We don't mention about description of the Control Byte in each bit, user can see more detailed information in Data Sheet. In the example, it uses **Control Byte 0x0D** to read ADC value on the X axis of Touch Screen and **Control Byte 0x90** to read ADC value on the Y axis of Touch Screen.

Summarize the procedures to read ADC value from ADS7846

1) Read status from Pin PEN of ADS7846; if it is 0 (touching the screen), start reading value in the next step 2; on the other hand, if it is 1 (not touching any screen yet), reads repeatedly.
2) Set Pin DCLK, CS, DOUT to be 0
3) Send **Control Byte 0x0D** to Pin DIN (MOSI) of ADS7846 to specify reading 12Bit ACD value on the X axis.
4) Send **Data 0x00** to Pin DIN (MOSI) of ADS7846. While sending out data in each bit, ADS7846 will also shift out the ADC value to Pin DOUT(MISO). The first bit data that is shifted out is the 11th bit at the falling edge of the second DCLK. When all 8 of DCLK are sent out completely, it reads data on the X axis as **0x0ddddddd (d=data bit11-bit5)**.
5) Send **Control Byte 0x90** to Pin DIN (MOSI) of ADS7846 to specify reading 12Bit ADC value on Y axis. While sending out this Control Byte, the last 5Bit data ADC on X axis will be also sent out. It begins with bit4 to bit0 and Data is arranged as **0xddddd000 (d=data bit4-bit0)**.
6) Send **Data 0x00** to Pin DIN (MOSI) of ADS7846. While sending out data in each bit, ADS7846 will also shift out ADC value to Pin DOUT(MISO). The first bit data that is shifted out is the 11th bit at the falling edge of the second DCLK. When all 8 of DCLK is sent completely, it reads on the Y axis as **0x0ddddddd (d=data bit11-bit5)**.
7) Send **Data 0x00** to Pin DIN (MOSI) of ADS7846. While sending out Data, the last 5Bit ADC data on the Y axis will be also sent out. It begins with bit4 to bit 0 and Data is arranged as **0xddddd000 (d=data bit4-bit0).**
8) When the ADC values on both axes are read completely, need to set Pin CS to be 1 to finish reading values from ADS7846.
9) If user wants to read the new value, repeatedly start the step 1.
10) After got the ADC value on each axis completely, user needs to arrange the values. The variable that is used to store the ADC value should be read as 16Bit. When the first 7Bit of data ADC is read, it is stored in the 16Bit variable as **0x000000000ddddddd ;** and the next 5Bit when it is stored in the 16bit variable is **0x000000000ddddddd**. Then shift out the first 7Bit data that is read to the left side 5Bit and then shift out the next 5Bit that is read to the right side 3Bit. Finally, must OR(|) both sets together, user got 12Bit data ADC of the axis that is read as **0x000ddddddddddddd**. This is the complete value and is ready to use.

From the principles of controlling LCD and Touch Screen above, it is overview of using of Board ET-TFT240320TP-3.2. When user understands the initial principles for controlling, it makes user can write program easier because user can copy the function in the ETT example instantly. See more detailed information regarding using functions in "Description of Example Program".

5. Application of Program Embedtools3.31 to Convert photo file into hex code

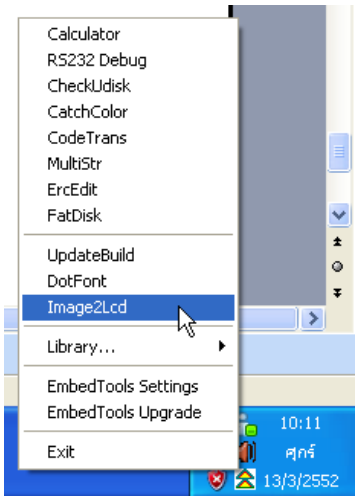
This section mentions how to convert photo file into hex code, send the hex code to LCD Display, and display photo file on the LCD Display as required. We use **Program "Embedtools3.31"** and we will describe procedures as below. The hex code that is converted by this program supports the ETT example by using function **"plot_picture()"** (in example **Ex3_Touch_Button**) to be transmitter of hex code from MCU to LCD.

If user does not initial LCD according to the ETT example and does not set Program Embedtools3.31 to convert the photo file as described below, it makes user unable to use **Function "plot_picture()"** because the direction of sending data maybe error. So, user needs to write program by self to plot the photo according to the setting values of user.

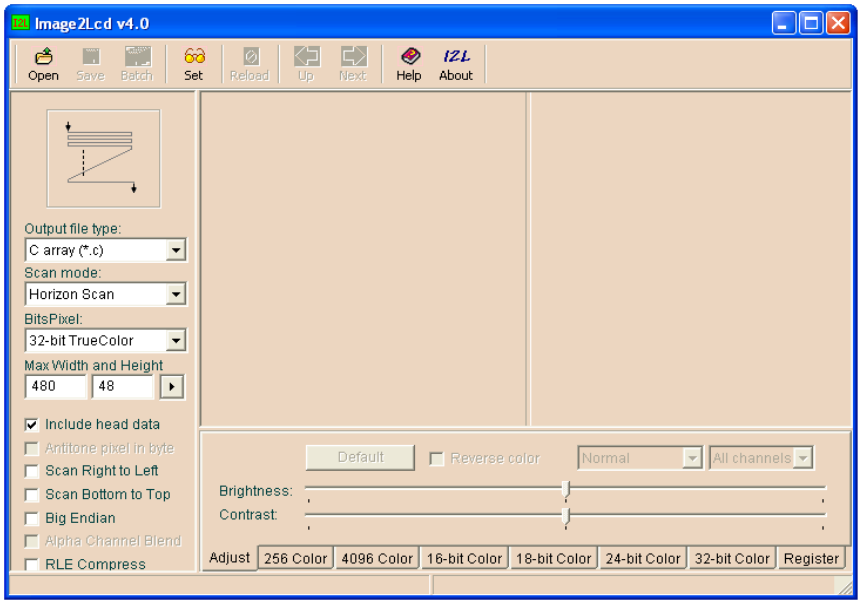
Procedures to use Program Embedtools 3.31 to Convert Photo File into Hex Code

1.) Install **Program Embedtools.exe** into computer (in **Folder Embedtools3.31**).

2.) After installed program successfully, it displays icon ( ) on the Task bar as shown in the picture 1.2. Click right on the icon, it makes the tab appear, click left to choose **Image2LCD** to run Program Embedtools and it will display window as shown in the picture 13.

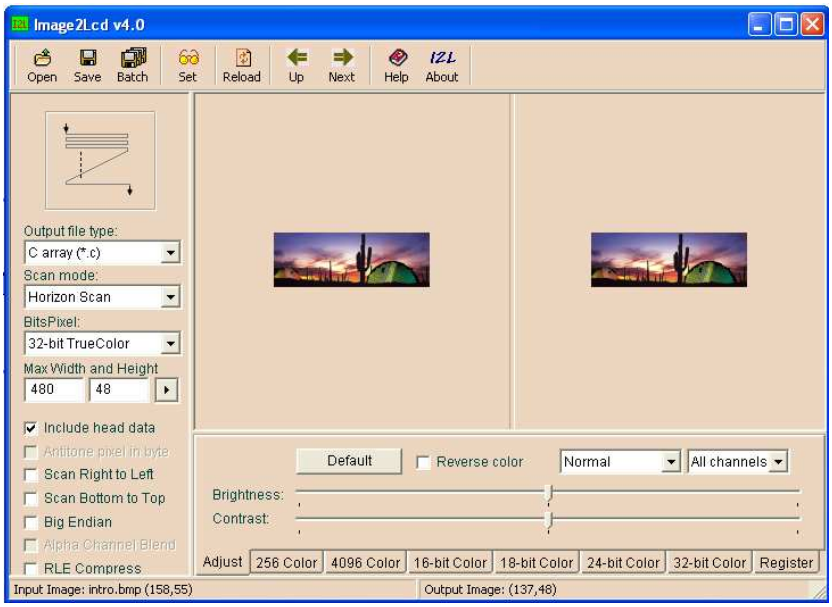


Picture 12



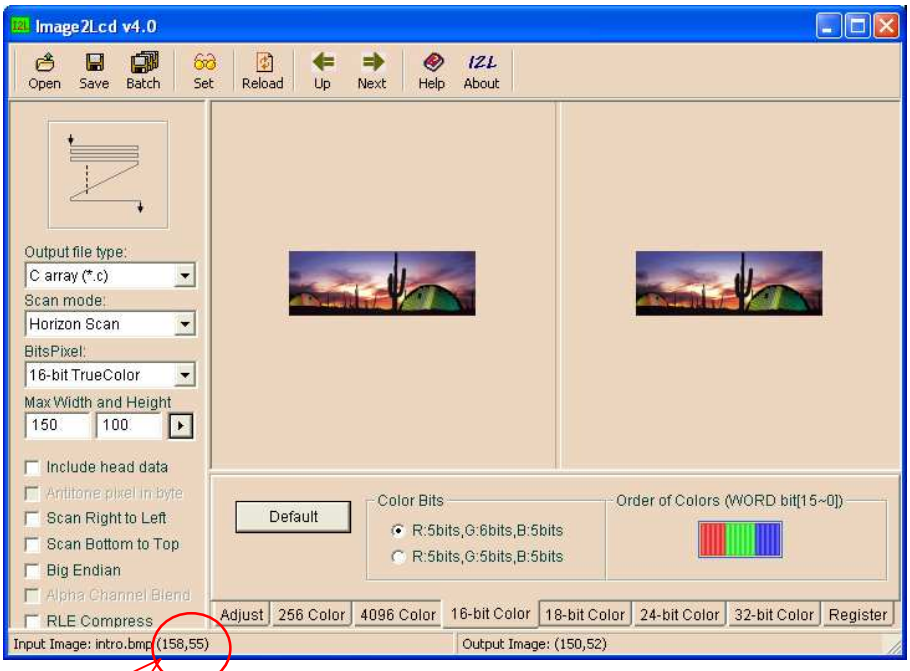
Picture 13

3.) Click icon Open( ) above the program to choose the required photo file to convert. The file surname should be .jpg or .bmp and when opened the photo successfully, it displays window as shown in the picture 14.



Picture 14

4.) After loaded photo into program successfully, set values as follows (refer to picture 15)



Size true Pixel of picture
(Width , Height)

Picture 15

- Output file type: *C array (*.C)* = Set data into the format of C Language Array and then save the Output file as dot C.
- Scan mode: *Horizon Scan* = Set the initial direction of Scan Data; when using this data, user needs to send the data to LCD according to the scanned direction.
- Bits Pixel: *16-bit TrueColor* = Set the color bit (there are 2 same values in the program, please choose the one above).
- Max Width and Height: The Width, the Height = Set the width and the height for photo completely and then click ( ). User can see the change of photo size in the unit of pixel; however, it is not the exact photo size. So, when user calls and uses **function "plot_picture()"** from ETT example, it needs to set the exact width and height of photo. In this case, user can see the true pixel of the photo size from the left bottom of the window.
- Include head data = Tick or remove the sign from the gap.
- TAB 16-bit Color = Click **TAB 16-bit color** in the gap of **Color bit** and then choose **R:5bit,G:6bit,B:5bit**. In the part of **Order of Colors (WORD bit[15~0])**, it needs to arrange the colors as RGB, and it does not set any value in other tab.

5.) When set all values successfully, please click Icon save (Save) to save file hex code. It uses Note Pad to open file, copy hex code and then paste it in Editor to write program.

6. Application of example program

The ETT example is written by C Language and supports 3 MCU families such as AVR(Mega128), PIC(18F8722), and ARM7(LPC2138). It controls LCD in the format of vertical and horizontal depend on user's requirement to display result in which format, so user can refer to the required format in the example. The examples of each MCU family are similar and all these examples refer to the initial address position on X,Y axis of screen according to the direction as shown below.

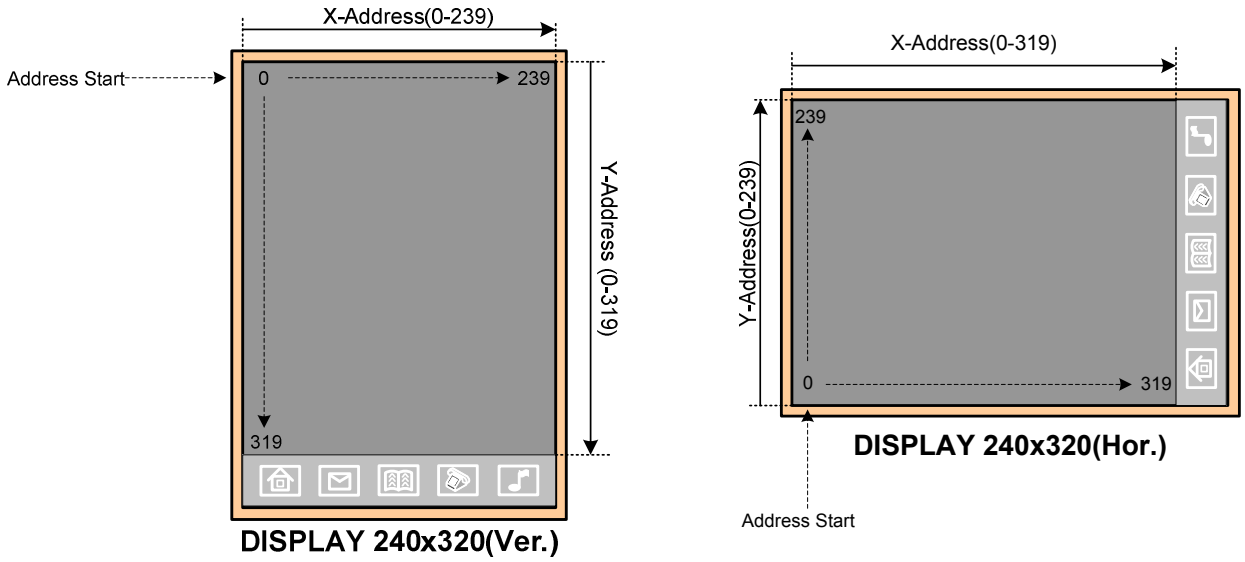


Figure 16 displays the reference to the address position on LCD Display in the vertical and horizontal direction.

If using the example function, user needs to see and refer to the address position according to the figure above to set address position of X,Y to start plotting photo or character.

There are 3 main examples that are either vertical or horizontal examples as described below;

- Ex1_Touch_Position: When initially running this program, first of all user needs to touch 3 points that are marked by + sign to calibrate value in the part of Touch Screen. After calibrated successfully; when user touched screen, it makes MCU be able to read the address position correctly according to the exact address position of LCD as displayed on the figure 16. After calibrated successfully; when user touches any position on the LCD Display, it displays the X,Y position that is touched at the bottom display.

- Ex2_Touch_Draw: When initially running program, user needs to calibrate Touch Screen as same as in the example1. Next, user can draw any picture or write any character on the screen as required, it displays the lines according to the user's writing or drawing. User can touch Icon Note ( ) to change the line for drawing and touch Icon Home( ) for Clear Screen.

- Ex3_Touch_Button: When running program; user also needs to calibrate Touch Screen and it displays buttons. When user touched any button on the screen, in the blank window displays pictures according to the pressing button.

In the part of Calibrate Touch Screen, user always needs to re-calibrate if resetting MCU as mentioned in the example. However, if user needs to calibrate only one time when it is exactly used, user can follow these instruction to solve this problem as follows;

Solution 1: Interface more E2Prompt with MCU (if using MCU has not any internal E2PROMPT); next, in the **Function "touch_calibrate()"** after the line of calling **Function "set_matrix()"**, user needs to write program to write value in the variable **divider,An,Bn,Cn,Dn,En,Fn** and then store it in E2Prompt. Finally, write any 1 byte value to store in E2Prompt, this value is used to be Flag Status to check whether it has already been calibrated or not.

In the part of main program; before calling **function "touch_calibrate()",** user needs to read the Status Flag from E2Prompt first to check whether it is the same value that is written and stored or not. If yes, it is unnecessary to call **function "touch_calibrate()"**, but user can read the value of the variable **divider,An,Bn,Cn,Dn,En,Fn** that is stored in E2Prompt instantly. Next, user can replace the value in the variable **divider,An,Bn,Cn,Dn,En,Fn;** so, user can run other parts of program and does not waste time to calibrate every time when using LCD.

Solution 2: This method does not use any E2Prompt. In the **function "touch_calibrate()"** after the line of calling **Function "set_matrix()"**, user needs to add **command printf()** (it has already been written in the example but it is disabled) to print value of the variable **divider,An,Bn,Cn,Dn,En,Fn** and display results through RS232. It uses Program Hyper Terminal to

receive value of the variable that is printed and then display it to user. Next, user needs to note down the value of each variable that is displayed because this value is used to configure the variable **divider,An,Bn,Cn,Dn,En,Fn** that is declared above **main()**. User can remove the **function "touch_calibrate()"** and does not call it any more when it is exactly used. If using this method with many LCD Displays, user needs to write program to calibrate and find the value **divider,An,Bn,Cn,Dn,En,Fn** separately. Although user changes the new screen, it always displays the same window as the old one to calibrate the new value again. User can not use the old value that is read from the old screen with the new screen because the address value that is read from touching is not corresponding with the address position of the exact LCD Display.

NOTE: ***For the calibrating, user needs to touch screen that is the most corresponding with the marked position because it makes the exact application high accuracy.***

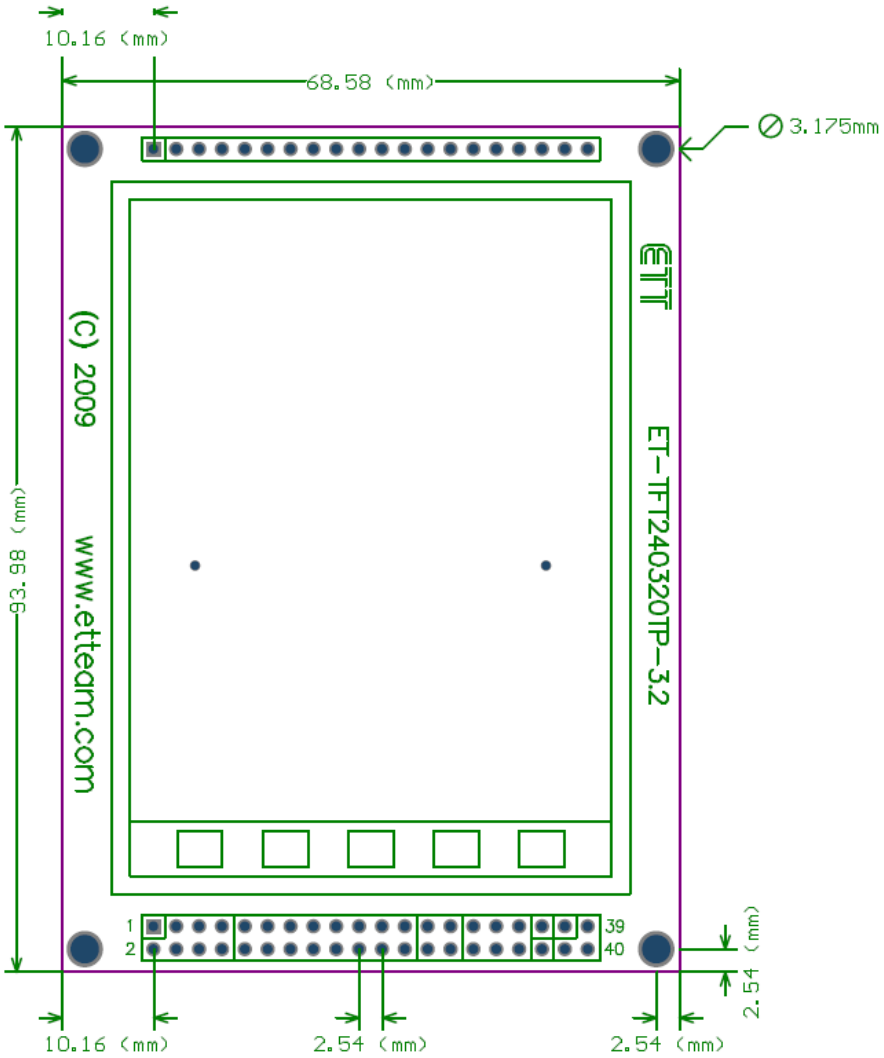
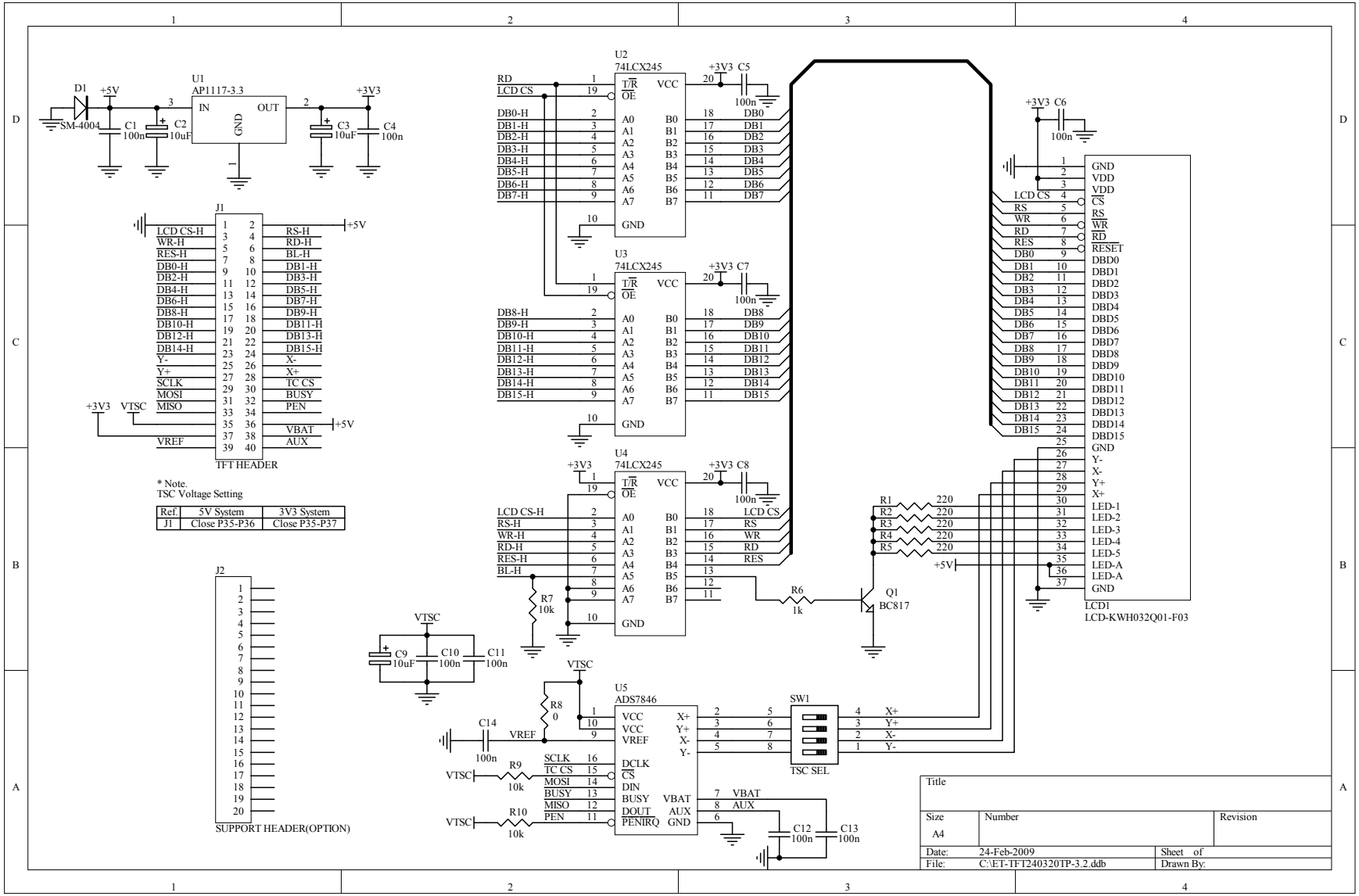


Figure 17 displays the size of Board ET-TFT240320TP-3.2.


U1
AP1117-3.3
IN
OUT
GND
D1
SM-4004
+5V
+3V3
C1 100n
C2 10uF
C3 10uF
C4 100n
J1
TFT HEADER
LCD CS-H
WR-H
RES-H
DB0-H
DB2-H
DB4-H
DB6-H
DB8-H
DB10-H
DB12-H
DB14-H
Y-
Y+
SCLK
MOSI
MISO
VREF
RS-H
RD-H
BL-H
DB1-H
DB3-H
DB5-H
DB7-H
DB9-H
DB11-H
DB13-H
DB15-H
X-
X+
TC CS
BUSY
PEN
VBAT
AUX
VTSC
* Note.
TSC Voltage Setting
Ref. | 5V System | 3V3 System
J1 | Close P35-P36 | Close P35-P37
J2
SUPPORT HEADER(OPTION)
U2
74LCX245
U3
74LCX245
U4
74LCX245
U5
ADS7846
VCC
GND
T/R
OE
A0
A1
A2
A3
A4
A5
A6
A7
B0
B1
B2
B3
B4
B5
B6
B7
RD
LCD CS
DB0-H
DB1-H
DB2-H
DB3-H
DB4-H
DB5-H
DB6-H
DB7-H
DB8-H
DB9-H
DB10-H
DB11-H
DB12-H
DB13-H
DB14-H
DB15-H
DB0
DB1
DB2
DB3
DB4
DB5
DB6
DB7
DB8
DB9
DB10
DB11
DB12
DB13
DB14
DB15
LCD CS-H
RS-H
WR-H
RD-H
RES-H
BL-H
LCD CS
RS
WR
RD
RES
R1 220
R2 220
R3 220
R4 220
R5 220
R6 1k
R7 10k
R8 0
R9 10k
R10 10k
Q1
BC817
C5 100n
C6 100n
C7 100n
C8 100n
C9 10uF
C10 100n
C11 100n
C12 100n
C13 100n
C14 100n
LCD1
LCD-KWH032Q01-F03
GND
VDD
CS
RS
WR
RD
RESET
DBD0
DBD1
DBD2
DBD3
DBD4
DBD5
DBD6
DBD7
DBD8
DBD9
DBD10
DBD11
DBD12
DBD13
DBD14
DBD15
Y-
X-
Y+
X+
LED-1
LED-2
LED-3
LED-4
LED-5
LED-A
SW1
TSC SEL
SCLK
TC CS
MOSI
BUSY
MISO
PEN
DCLK
DIN
DOUT
PENIRQ
Title
Size
A4
Number
Revision
Date: 24-Feb-2009
Sheet of
File: C:\ET-TFT240320TP-3.2.ddb
Drawn By: