ET-TFT240320TP-3.2 REV.B

1. Specifications of Board ET-TFT240320TP-3.2 REV.B

- Be Display Module TFT LCD Color + Touch Screen with 240x320 Pixel
- Be 3.2" TFT Display
- Use Single Chip Driver No.SPFD5408A
- Have 65536 fine colors (RGB = R:5bit-G:6bit-B:5bit)
- Choose Interface GLCD through DIP SW2. MODE; in this case, there are 3 Modes: 1) Parallel Mode 16-Bit Interface; 2) Parallel Mode 8-Bit Interface; and 3) Serial Mode SPI Interface
- Have 2 Types of Interfaces that can be chosen to control Touch Screen. Firstly, it is DIP SW1.TSC SEL, it is SPI Interface that is through Chip Touch Screen Controller #ADS7846 (12BIT ADC); and secondly, it is Interface by directly interfacing Pin X-,X+,Y-,Y+ with Pin ADC of MCU (it is quite difficult to write program for controlling).
- If do not use Touch Screen for actual application, be able to control only part of LCD.
- Summarize I/O amounts of MCU to control GLCD and Touch Screen as follows;
    1. When using Parallel Mode 16-Bit Interface, it uses 27 PIN of I/O in total.
    2. When using Parallel Mode 8-Bit Interface, it uses 19 PIN of I/O in total.
    3. When using Serial Mode SPI Interface, it uses 10PIN of I/O in total (it is better if using with the minimum of 10MHz SPI-Clock).
- Be able to interface with MCU that is 5V or 3.3V Power Supply (please read more information in the topic of connecting)
- Connector that is used in Parallel Mode 8-Bit and 16-Bit is Pin Header 2x20; and Connector that is used in SPI Serial Mode is Pin Header 1x20.
- DC +5V Power Supply for Board

2. Feature and Structure of Board ET-TFT240320TP-3.2 REV.B

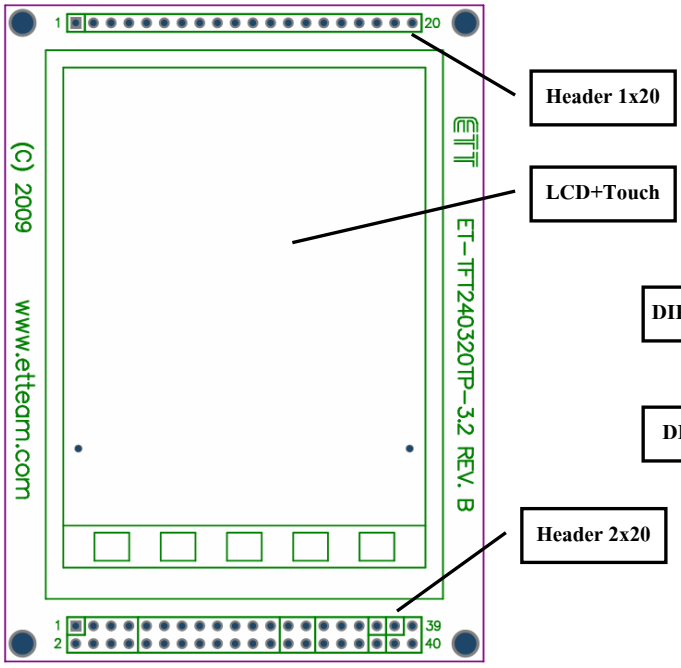


Figure 2.1(A) Front Board

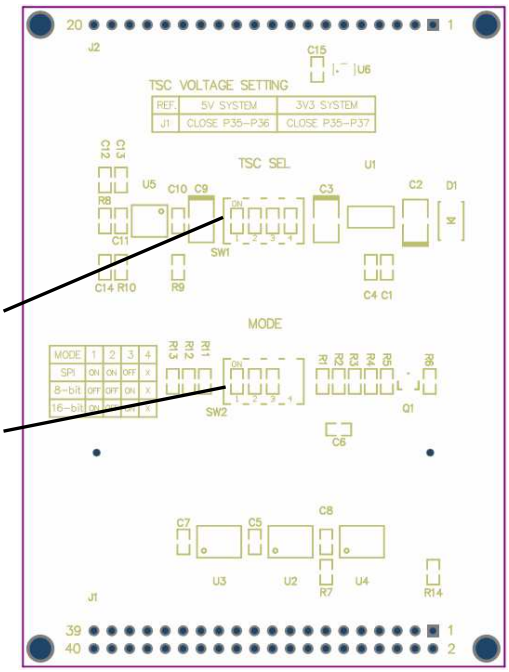


Figure 2.1(B) Back Board

- *LCD+Touch*: It is the area of 240x320 Pixel LCD Display and the surface of screen is coated by pad of Resistance Touch Screen.
- *Header 1x20*: It is Connector MALE 1x20 PIN; it is used to interface signal in the format of SPI-MODE from MCU to control the operation of LCD and Touch Screen. Function of each pin is displayed as shown in the Table 2.1, 2.2.

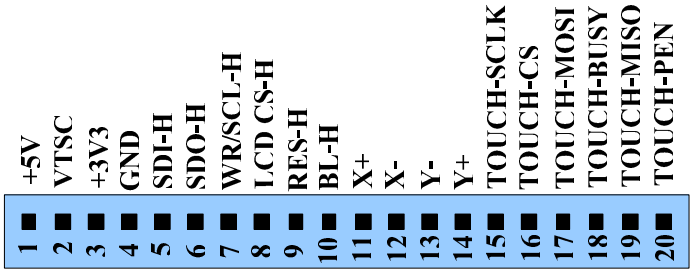


Figure 2.2 shows pin position of Header 1x20(SPI-MODE)(front view as shown in the figure 2.1(A)).

- *Header 2x20*: It is Connector MALE 2x20 PIN; it is used to interface signal in the format of Parallel-Mode (8-16 Bit) from MCU to control operation of LCD Display and Touch Screen. Function of each pin is shown in the Table 2.1, 2.2.

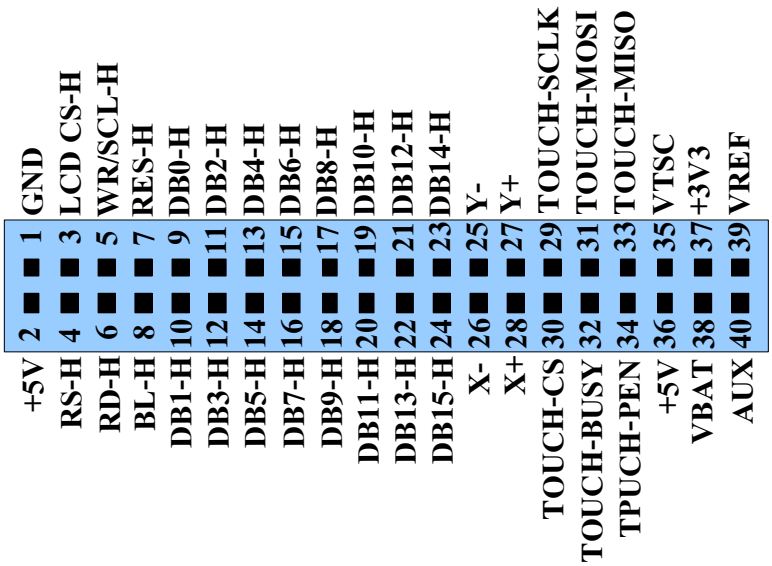


Figure 2.3 displays pin position of Header 2x20(Parallel-Mode:8,16 Bit)(front view as shown in the figure 2.1 (A)).

Table 2.1: Detailed PIN to Control GLCD

No. PIN		PIN-NAME	I/O	Description
Header2x20 (Parallel Mode)	Header1x20 (SPI Mode)			
1	4	GND	Power LCD	Pin Ground
2	1	+5V	Power LCD	Pin +5V Power Supply for Board
-	5	SDI-H	I	Pin Serial Data In: To receive data or command from MCU and it is used in SPI Mode.
-	6	SDO-H	O	Pin Serial Data Out: To transmit data to MCU and it is used in SPI Mode.
3	8	LCD CS-H	I	Pin Chip-Select: To Enable LCD Module (use with both SPI and Parallel Mode) Low = LCD Module Enable and High = LCD Module Disable
4	-	RS-H	I	Pin Register-Select: Low- Choose to access Index(IR) or Status (SR) Register High- Choose to access Control Register (Address 00H-98H)
5	7	WR/SCL-H	I	Pin Write-Strobe/Serial Clock: To write data when it has received Signal Low in Parallel Mode or use it to be Pin Clock in SPI Mode.
6	-	RD-H	I	Pin Read Strobe: To read data when it has received Signal Low.
7	9	RES-H	I	Pin Reset: To Initial LCD Module when it has received Signal Low.
8	10	BL-H	I	Pin Black Light: Black Light of LCD is lit when this pin has received Signal as High.
9-24	-	DB0-DB15	I/O	Pin Data Bus Bi-Directional 16-Bit: To transmit data or point address position of Register and it is used for Parallel mode.

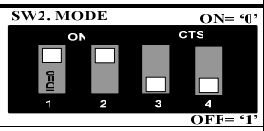
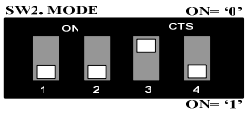
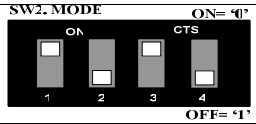
Table 2.2: Detailed PIN to Control Touch Screen

No. PIN		PIN-NAME	I/O	Description
Header2x20 (Parallel Mode)	Header1x20 (SPI Mode)			
25*-28 *	11*-14*	**Y-,X-,Y+,X+**	I	Pin Y-,X-,Y+,X+ is the pin to read position of Touch Screen directly without through Chip ADS7846. It needs to shift all DIP SW1.TSC SCL to position Off.
29	15	**TOUCH-SCLK**	I	It is Pin DCLK of ADS7846 to Synchronizes Serial Data I/O.
30	16	**TOUCH-CS**	I	It is Pin CS of ADS7846; when it has received Signal LOW, it Enables Serial I/O Register of Chip to start operating.
31	17	**TOUCH-MOSI**	I	It is Pin DIN of ADS7846; when Pin CS is Low, data will be latched at the rising edge pin of Signal DCLK.
32*	18*	**TOUCH-BUSY**	O	It is Pin BUSY of ADS7846; it is High Impedance when Pin CS is High.
33	19	**TOUCH-MISO**	O	It is Pin DOUT of ADS7846; when Pin CS is LOW, data will be shifted at the falling edge pin of DCLK and this Output is High Impedance when CS is High.
34	20	**TOUCH-PEN**	O	It is Pin PENIRQ of ADS7846; when Touch Screen is touched, it has given Signal Logic as LOW (it has already set Pull-Up R10K on board).
35,36,37	1,2,3	VTSC,+5V, +3V3	Power Touch Screen	These 3 Pins are used to choose Power Supply for ADS7846. If using with 5V MCU, it needs to jump Pin VTSC with Pin +5V; on the other hand, if using with 3.3V MCU, it needs to jump Pin VTSC with Pin +3V3(37).
38*	-	VBAT	I	It is Pin Vbat of ADS7846 that is not used in the part of Touch Screen.
39*	-	VREF	I/O	It is Pin Vref of ADS7846 that is not used in the part of Touch Screen.
40*	-	AUX	I	It is Pin AUX input to ADC of ADS7846 that is not used in this part.

(*) = Pin is not used and connected. Refer to the given examples of ETT.

- *DIP SW1.TSC SEL 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, it needs to shift all of 4 DIP SW. to position ON (Default). Moreover, the given example supports this communication mode as well. However, if user chooses the Interface type that directly interfaces Pin X-,X+,Y-,Y+ with Pin ADC of MCU (not use ADS7846), it needs to shift all of 4 DIP SW. to the opposite position of Default (Off).
- *DIP SW2.MODE on the back board:* There are 4 DIP SW. but it only uses 3 SW.; S1, S2 and S3. The DIP SW2. is used to choose Interface Mode between Board GLCD and MCU that is connected to control the operation. The method to choose the Interface Mode is displayed in Table 2.3.

Table 2.3: How to choose Mode Interface LCD from DIP SW.2

DIP SW.2 MODE				รูปการ Set SW2.	MODE Interface
S1	S2	S3	S4		
ON/OFF*	ON	OFF	X		SPI-Mode
OFF	OFF	ON	X		Parallel 8-bit Mode
ON	OFF	ON	X		Parallel 16-bit Mode

X = Anything; ON/OFF*= In SPI Mode, Switch S1 chooses ID; in this case, ON = ID is 0 and OFF = ID is 1.

The given example of program uses ID as 1; so, it needs to set Switch S1 to the position ON.

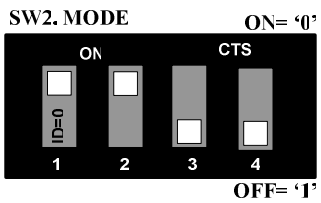
3. **How to connect Board ET-TFT240320TP-3.2 REV.B with MCU**

Now, it describes the method to connect Board ET-TFT240320TP-3.2 REV.B that supports the given ETT examples. It uses MCU AVR MEGA128, PIC 18F8722 (for 5V MCU) and MCU ARM7 LPC2138 (for 3.3V MCU). The circuit of MCU that is displayed below can be modified to connect with other MCU numbers or other families. ***However, be careful about Pin VTSC, it needs to jump with Pin that is either +5V or +3V3 Power Supply correctly and it is corresponding with Power Supply of MCU, otherwise, it makes the MCU damaged.*** For example, if using 5V MCU, it needs to jump Pin VTSC with Pin +5V.

There are 3 Interface Modes that can be chosen for connection and application; Serial Interface SPI-Mode, Parallel Interface 8-bit Mode and Parallel Interface 16-bit Mode. The method to choose any Interface Mode is to shift DIP SW2 on the back board and it can summarize example of connecting modes as follows;

3.1) Serial Interface SPI-MODE: The SPI CLOCK in this mode should use minimum of 10MHz frequency to respond to the display on LCD that is image or text from MCU in time.

How to Set Mode Interface



Interface Mode : SPI (ID=0)
(Use PIN I/O = 10 PIN)

How to connect with 3.3V MCU

Pin P0.2,P0.3,P0.11,P0.14 of MCU in this circuit is interfaced with R Pull-Up because this Pin Port is Open Drain; on the other hand, if the MCU that is interfaced has no any pin that is Open Drain, it need not to interface R Pull-Up.

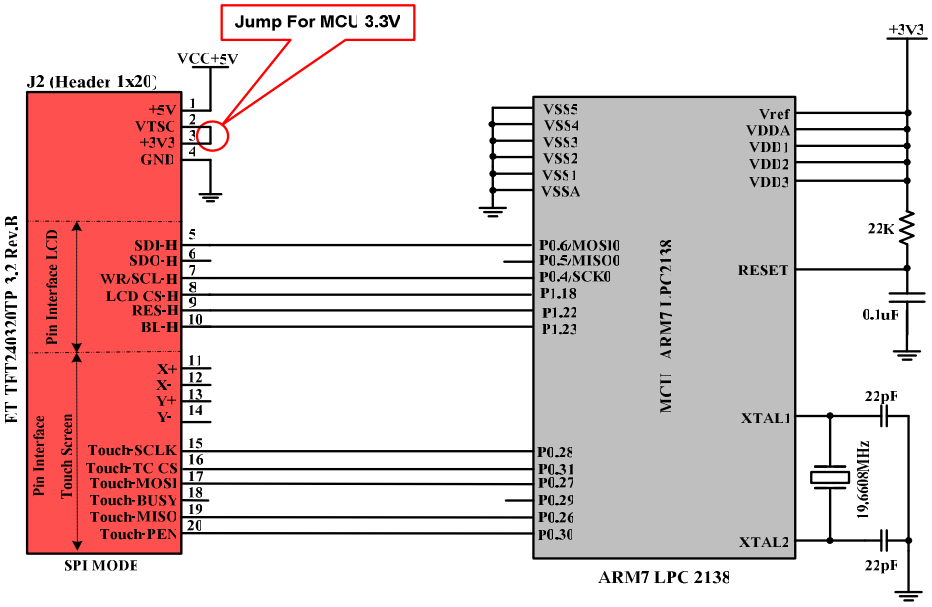


Figure 3.1A Display example of interfacing Board ET-TFT240320TP-3.2 REV.B with MCU ARM7 #LPC2138(3.3V).

How to connect with 5V MCU

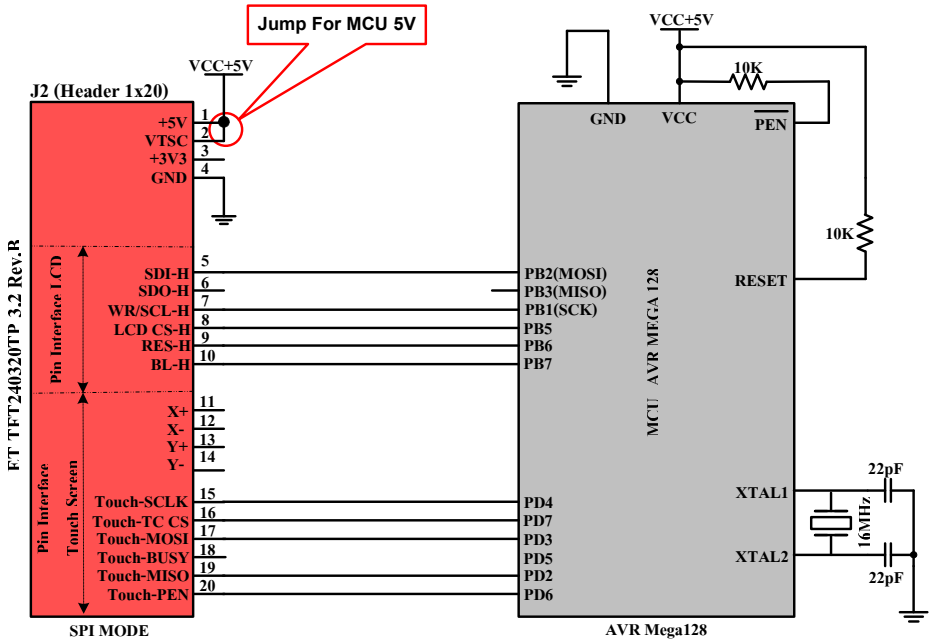
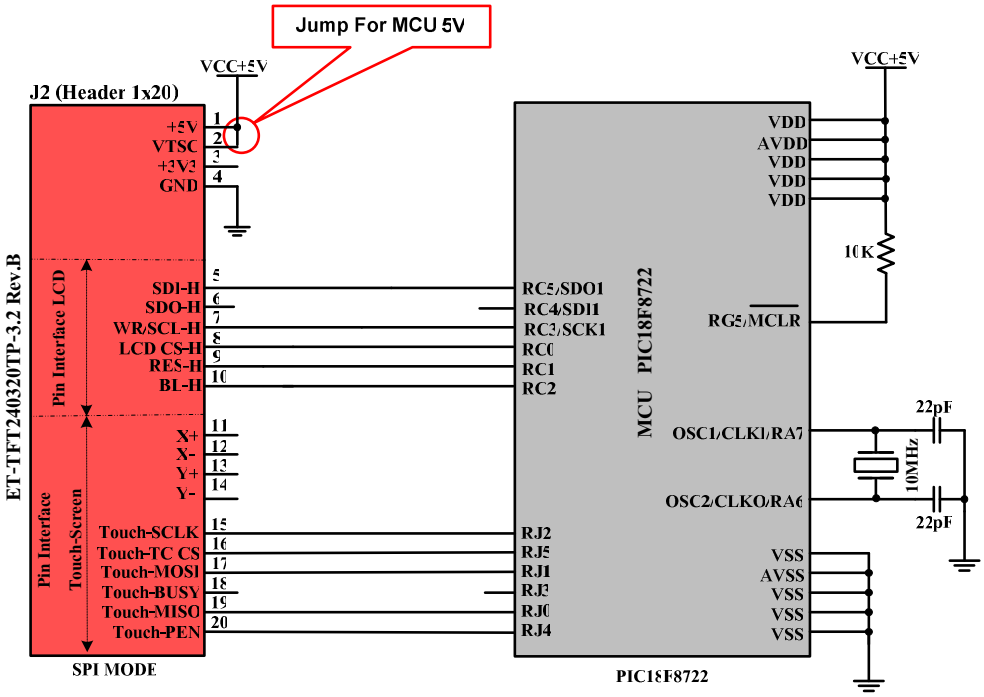


Figure 3.1B Displays example of interfacing Board ET-TFT240320TP-3.2 REV.B with MCU AVR #Mega128(5V).



Picture 3.1C Displays example of interfacing Board ET-TFT240320TP-3.2 REV.B with MCU PIC #18F8722(5V).

PIN Interface LCD of this SPI Mode is not used; so, Connector J1(Header 2x20) should connect into GROUND as shown with the dotted line in the picture 3.1D. In this case, it sets the Logic status for Pin of LCD that is not used (however, there is no any problem if this Pin is not connected into GROUND).

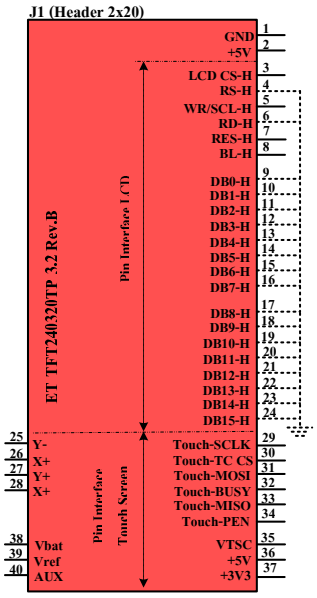
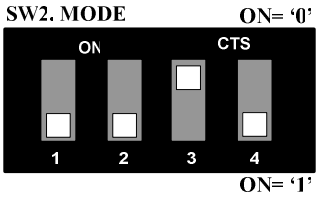


Figure 3.1D Displays how to connect Pin Interface LCD that is not used as identified with the dotted line above.

3.2) Parallel Interface 8-bit MODE

How to set Mode Interface



Interface Mode :Parallel 8 bit data
(Use PIN I/O = 19 PIN)

For this Parallel 8-Bit Mode, it uses Pin DATA to only receive-transmit 8-Bit data that is DB8-DB15. In the part of PIN Interface LCD that is DB0-DB7 is not used; so, it should be interfaced into GROUND as same as SPI MODE above (there is no any problem if it is not connected) as shown in the dotted line.

How to connect with 3.3V MCU

Pin P0.2,P0.3,P0.11,P0.14 of MCU in this circuit are connected with R Pull-Up because this Pin Port is Open Drain; so, if MCU that is connected has not any pin that is Open Drain, it need not to connect R Pull Up.

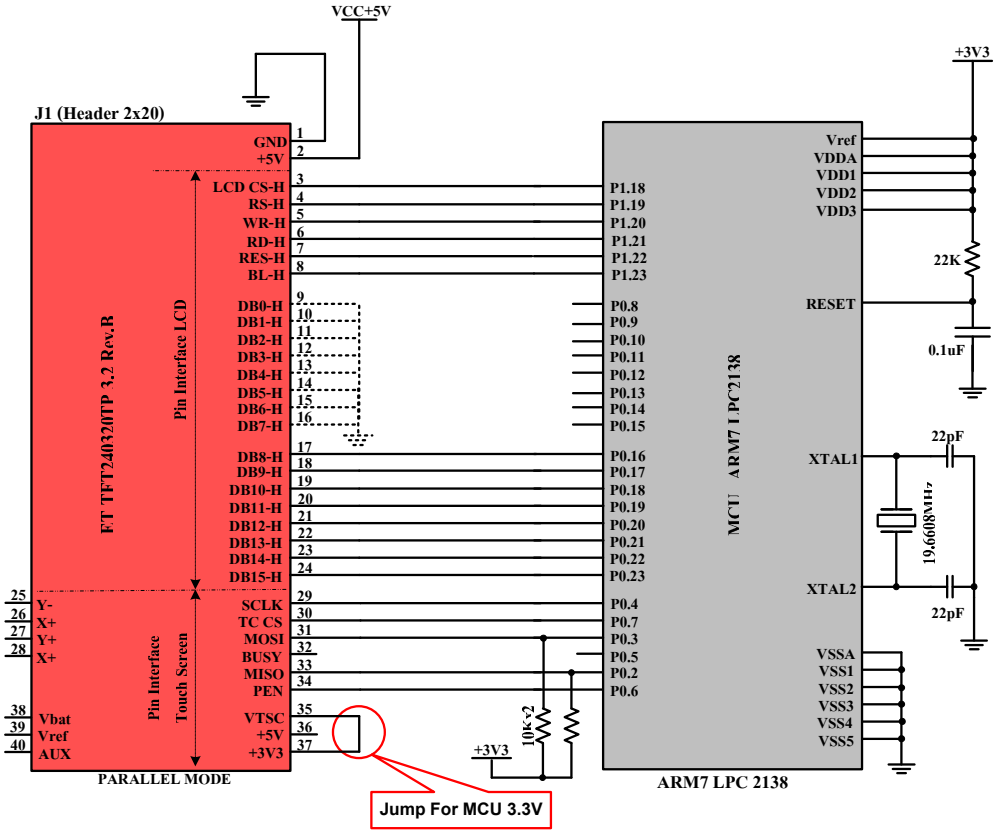


Figure 3.2A displays example of connecting Board ET-TFT240320TP-3.2 REV.B with MCU ARM7 #LPC2138(3.3V).

How to connect with 5V MCU

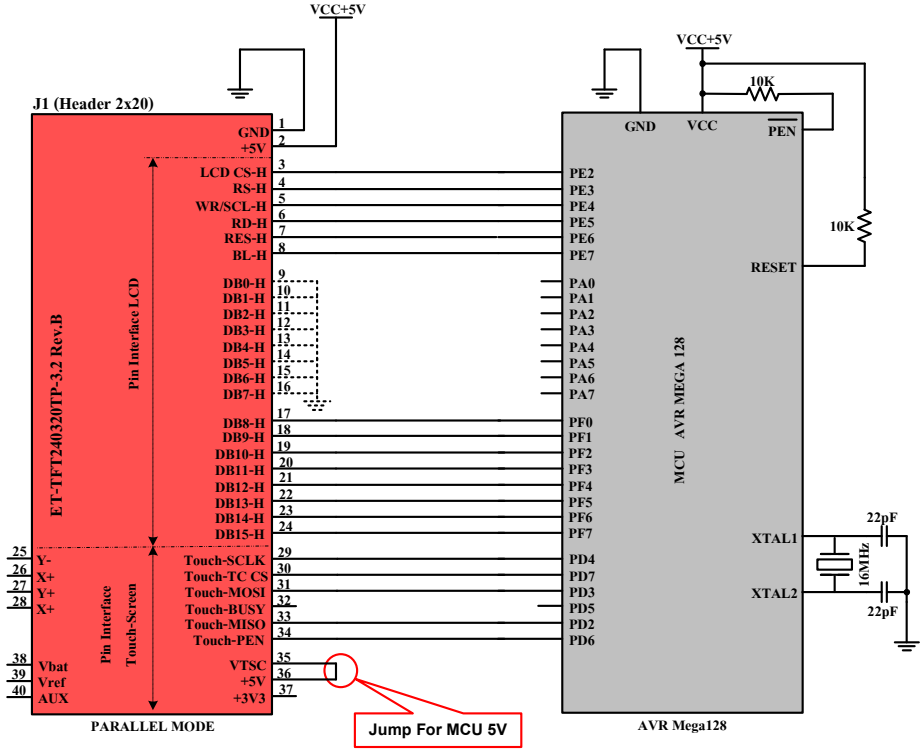


Figure 3.2B displays example of connecting Board ET-TFT240320TP-3.2 REV.B with MCU AVR #Mega128(5V).

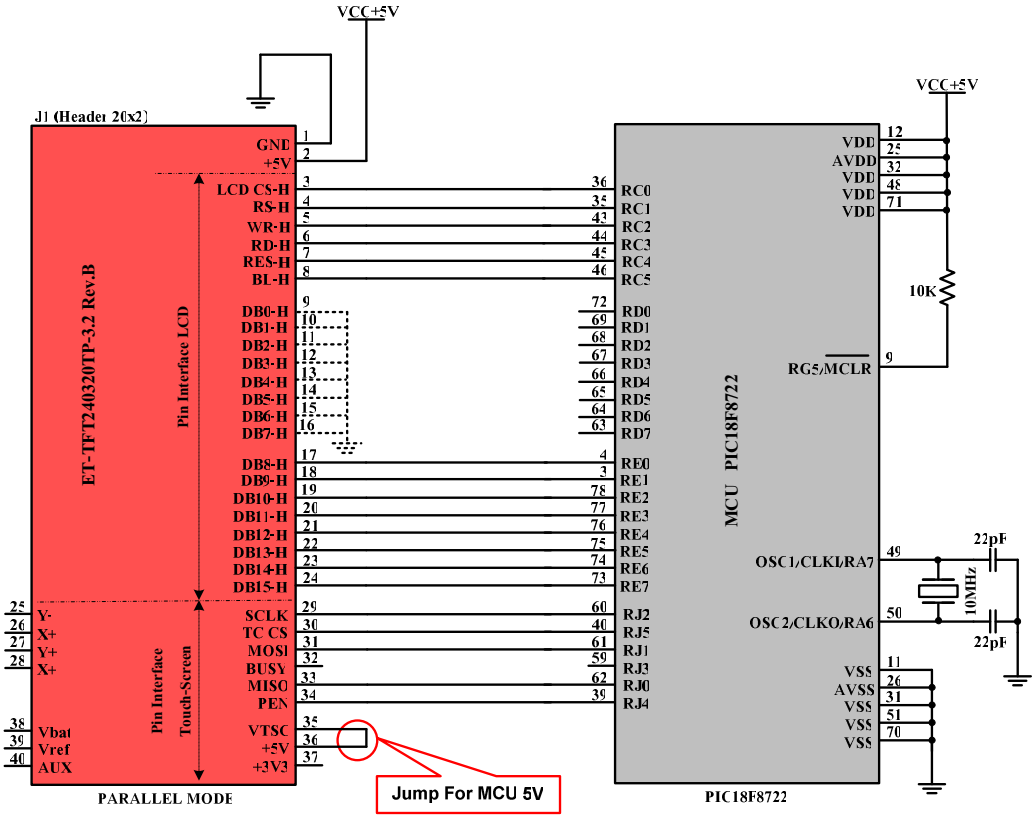
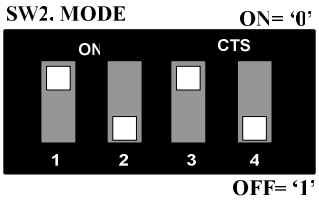


Figure 3.2C displays example of connecting Board ET-TFT240320TP-3.2 REV.B with MCU PIC #18F8722(5V).

3.3) Parallel Interface 16-bit MODE

How to set MODE Interface



Interface Mode : Parallel 16 bit data
(Use PIN I/O = 27 PIN)

How to connect with 3.3V MCU

Pin P0.2,P0.3,P0.11,P0.14 of MCU in this circuit is connected with R Pull-Up because this Pin Port is Open Drain; on the other hand, if MCU has not any Pin that is Open Drain, it need not to connect with R Pull-Up.

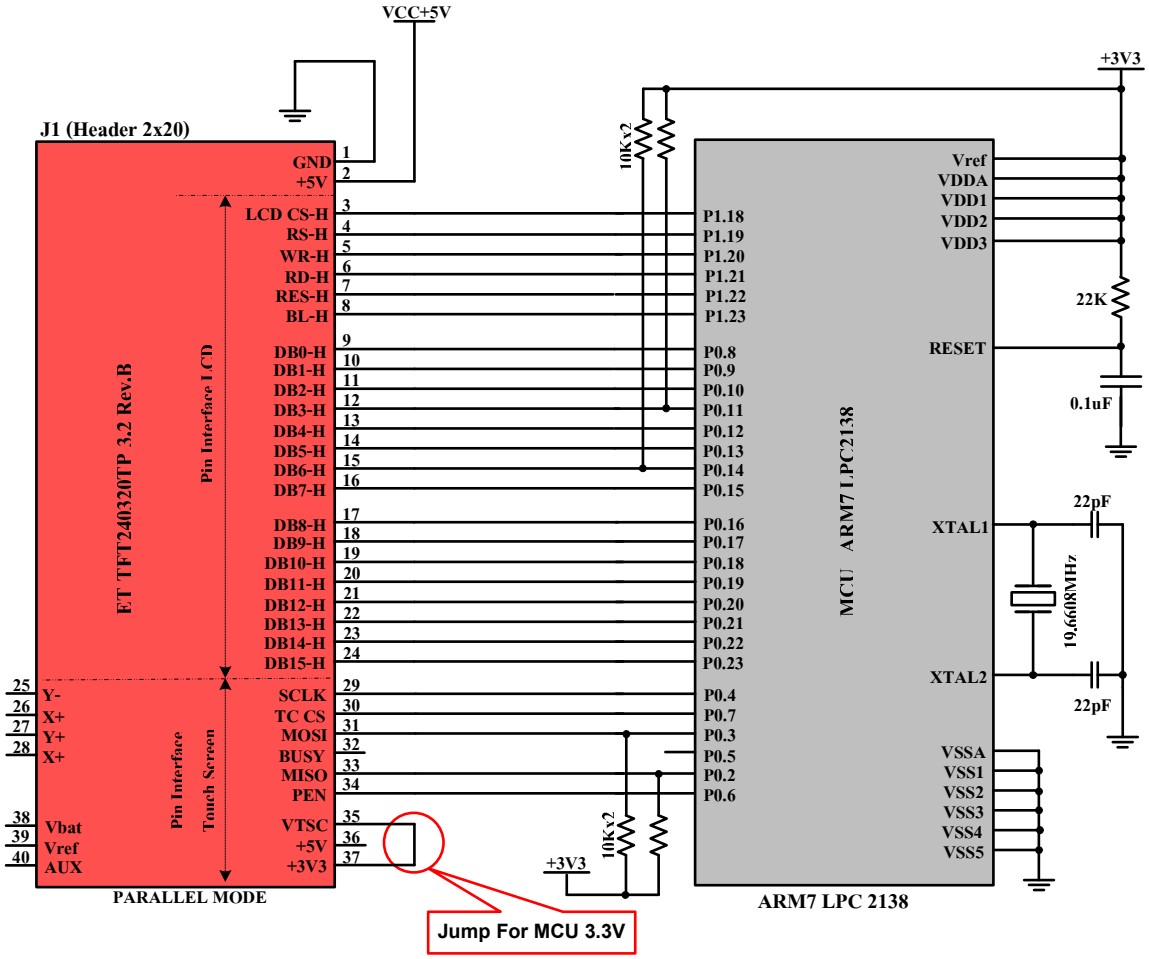


Figure 3.3A displays example of connecting Board ET-TFT240320TP-3.2 REV.B with MCU ARM7 #LPC2138(3.3V).

How to connect with 5V MCU

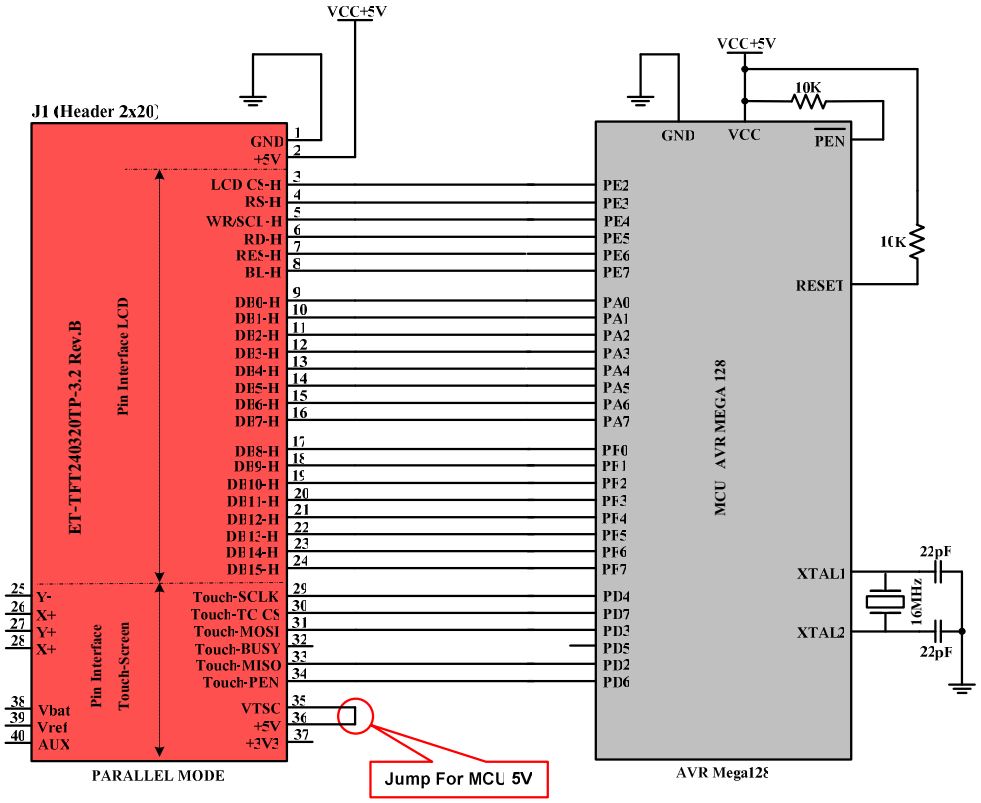


Figure 3.3B displays example of connecting Board ET-TFT240320TP-3.2 REV.B with MCU AVR #Mega128 (5V).

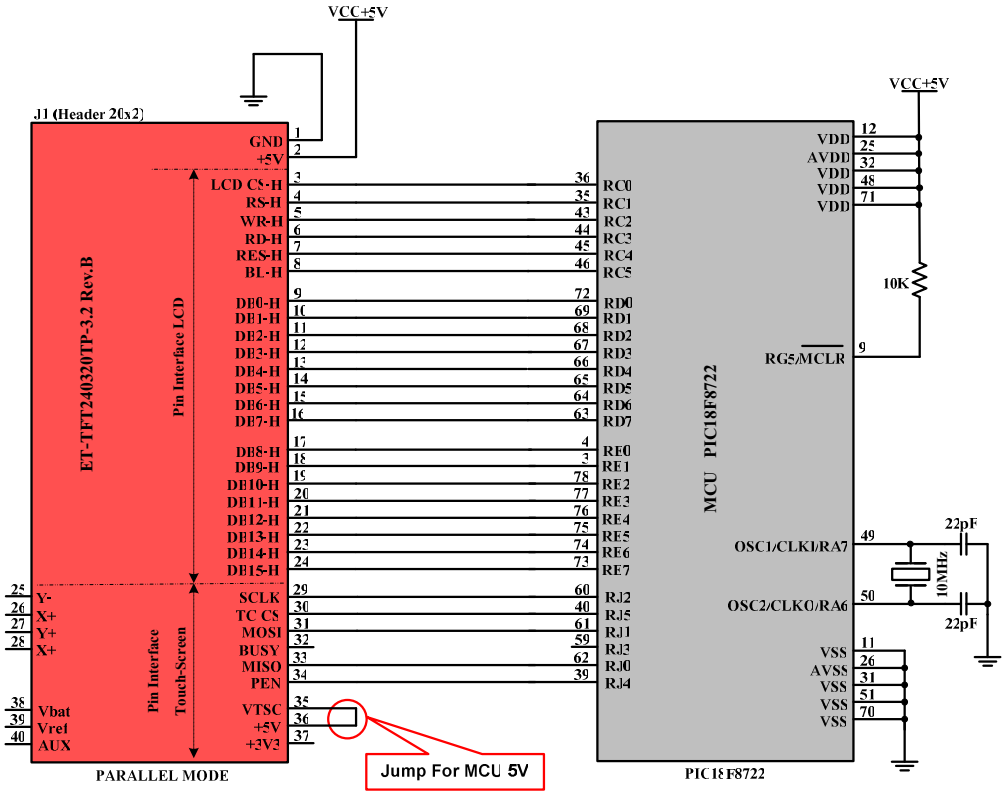


Figure 3.3C displays example of connecting Board ET-TFT240320TP-3.2 REV.B with MCU PIC #18F8722(5V).

4. Initial Principles of control over the connection of LCD and Touch Screen

For the method to write program to communicate with Board ET-TFT240320TP-3.2 REV.B is to be easier if user divides the operating control into 2 parts. Firstly, it is the part of LCD Display and there are 3 types of Interface Mode; Parallel 8-bit, 16-bit and SPI. Secondly, it is the part of Touch Screen that uses SPI Interface. In this case, it refers to ETT examples in CD-ROM to be principles to write program for controlling the operation; moreover, each example in CD-ROM uses the same initial principles to communicate with board but the format of displaying result on the LCD Display is only different. Both principles can be summarized as follows;

4.1) Interface Control LCD: User can read more detailed information of the command that is used to control LCD from Data Sheet **"Driver_SPFD5408A.pdf"** in CD-ROM. The principles to transmit command or Data to LCD are to divide into 3 types according the Interface and we will mention them in the next subtitle.

First of all, user needs to understand that there are 2 sets of data that is transmitted to LCD although it is interfaced in any Mode. Firstly, it is the set of Register that user will enter; in this case, it is 8-Bit 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; however, the actual value that will be transmitted is 16-Bit data, so, it is 0022H. Secondly, it is the set of 16-Bit Data of the command. When user has already transmitted the Command 0x22h, the second set of data that will be transmitted is colored Data or other values of the command. For example, if user sets LCD Display to display one white dot on the screen, it needs to transmit Data as 0xFFFF. After user understood well regarding the initial command; next, it describes how to arrange data internal LCD, especially, the incoming command and Data. Moreover, it includes Timing Diagram to Read-Write Data from MCU to LCD. In this case, there are 3 Mode of Interface as follows;

4.1.1) Interface SPI-MODE

LCD in this mode rearranges incoming command and Data that user has already transmitted as shown in the figure below;

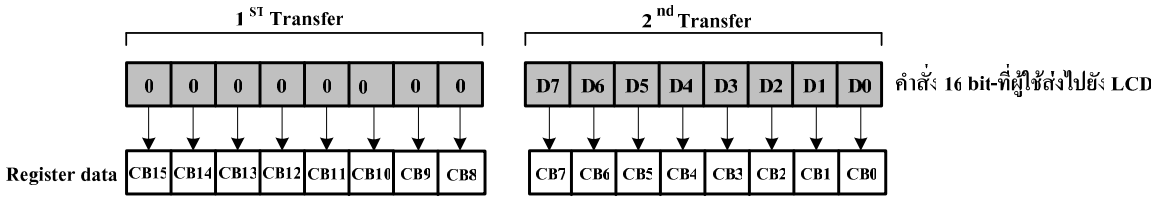


Figure 4.1.1A displays the Bit Arrangement of incoming command.

For the 1st Transfer Byte High(D15…D8) in the figure 4.1.1A above, it transmits the value 00H because the Address Register of command is only one byte; so, it is set the Address position at Byte Low(D7…D0).

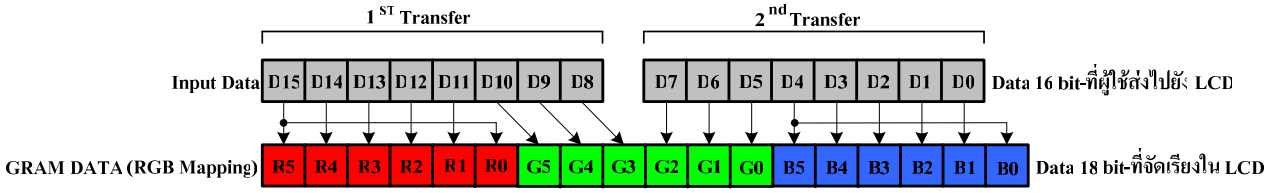


Figure 4.1.1B displays the arrangement of incoming colored Data Bit that is 16 bit 65K Colors.

From figure 4.1.1B above, it is arrangement of colored Bit Data from higher Bit to the lower Bit as RGB according to ETT examples. However, user can change and modify the arrangement of colored bit to be BGR by using Command Entry Mode (R03H); it rearranges colored bit and user can read more information from Data Sheet of SPFD5408A.

If user wants to transmit data to display dot on the LCD Display, it needs to mix colors by self by referring to the arrangement of colored bit as shown in the table above. Data D15-D11(5bit) is red color; Data D10-D5(6bit) is green color; and Data D4-D0(5bit) is blue color. When user has already transmitted Data into LCD, Data will be rearranged as 18-Bit data automatically as shown in the picture. Moreover, shade of colors will be arranged from darker to lighter or from the lower bit to the higher bit. For example, if it is the brightest red color, it is Data = 0xF800; or if it is the darkest green color, it is Data = 0x0020. If user wants colors other than these 3 main colors, user needs to set data bit to be in the proper range of each color and user will get the desired color. For example, if it is white color, it is Data = 0xFFFF; or if it is black color, it is Data = 0x0000.

Procedures to write Command and Data from MCU to control LCD in the format of SPI-MODE

The specifications of this SPI-MODE is to operate at the rising edge pin of Signal Clock and it transmits Data in the format of 8-Bit Serial (1 Byte) in each time; in this case, it transmits MSB Bit (the highest bit) first. The format of transmitting command or Data is 3 Byte in each time as described below;

Procedures to write Command (Index Register) - It needs to transmit before writing data.

1. Byte Start =70h(ID=0),74h(ID=1) -----> **2. Command Byte High** = 00h -----> **3. Command Byte Low** = Reg. NO (00h-A4h)

Procedure to write data – It needs to transmit after have written data successfully.

1. Byte Start =72h(ID=0),76h(ID=1) -----> **2. Data Byte High** = data (D15..D8) -----> **3. Data Byte Low** = data (D7..D0)

The 1st Byte: It is Start Byte and it always transmits the first Byte because it sets the format of reading-writing and dividing data that is transmitted into LCD to check whether it is Command or Data. This Start Byte is 8-Bit and the Bit Arrangement is shown as below;

BIT	7	6	5	4	3	2	1	0
Byte	Device ID Code						Control	
Start	0	1	1	1	0	ID	RS	R/W

- Device ID Code: [Bit2..Bit7] - This 6-Bit sets ID of LCD. In this case, Bit3-Bit7 is fixed value and other one Bit that is Bit2; user can set this bit by self from DIP SW.2 that is S1 on the back board. If S1 is in position ON, ID = 0; on the other hand, if S1 is in position OFF, ID = 1.

- Control: [Bit0..Bit1] – This 2-Bit sets format of writing-reading and dividing data that is transmitted to LCD to check whether it is Command or Data. Its format is shown as below;

RS	R/W	Function
0	0	Write index register
0	1	Read a Status
1	0	Write Data
1	1	Read Data

The 2nd Byte: It is Byte High (8-Bit High) of Command or Data. If user has set Bit in the 1st Byte as RS=0, R/W=0, the 2nd Byte will transmit Byte High of the command that is 00H. On the other hand, if user has set Bit in the 1st Byte as RS=1, R/W=0, the 2nd Byte will transmit Byte High of Data and its value depends on the command that is set for Data.

The 3rd Byte: It is Byte Low (8-Bit Low) of Command or Data. If user has set Bit in the 1st Byte as RS=0, R/W=0, the 3rd Byte will transmit Byte Low of command that is the value of Register NO.(00h-A4h) according to Data Sheet. On the other hand, if use has set Bit in the 1st Byte as RS=1, R/W=0, the 3rd Byte will transmit Byte Low of Data and its data depends on the command that is set for Data.

Timing Diagram to Read-Write Command and Data in SPI MODE

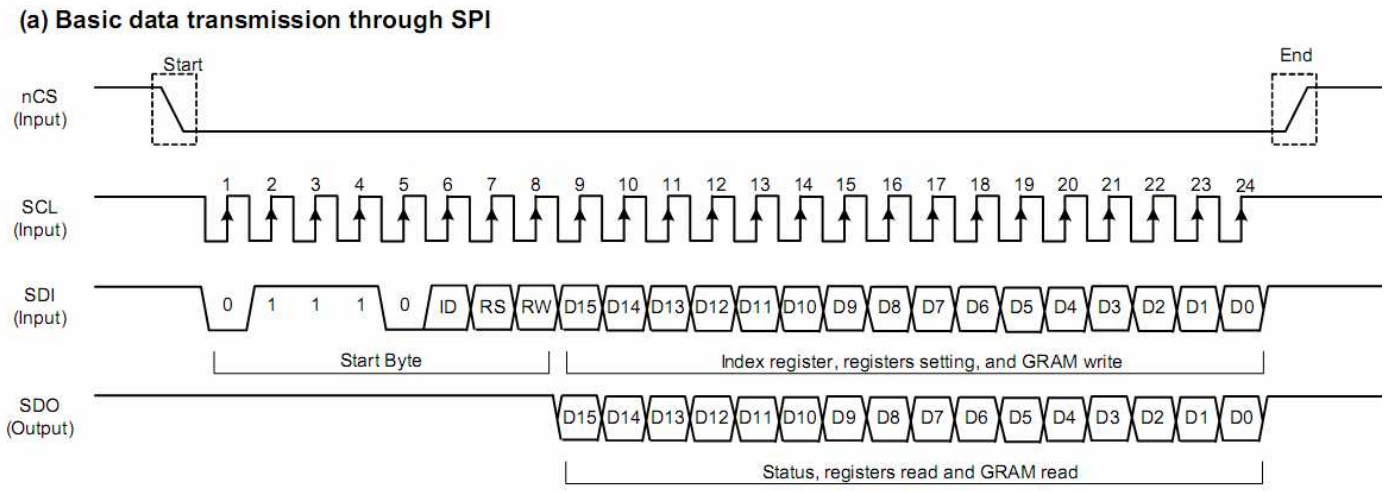


Figure 4.1.1C displays Timing Diagram to Read-Write Command and Data into LCD in SPI MODE.

From the Timing Diagram above, it displays how to set timing of signal for reading-writing data into LCD in SPI MODE. The normal status, it is set CS as High; the Start status, it is set CS as Low. Next, for the 1st Byte, MCU transmits Start Byte to Pin SDI of LCD according to the Clock timing of rising edge pin of the 1st-8th Wave that is set by MCU at Pin SCL. For the 2nd Byte, MCU transmits Command (Index Register) or Data in the Byte High(D15-D8) according to the Clock timing of the 9th-16th Wave. Lastly, for the 3rd Byte, MCU transmits Command (Index Register) or Data in Byte Low(D7-D0) according to the Clock timing of the 17th-24th Wave and then set Pin CS as High. It finishes the process of writing Command or Data for the first 3-Byte; if user needs to transmit new Data again, please start step No.1 repeatedly.

Note that the Clock timing of the 9th wave, user started writing data in the 2nd Byte(Bit-D15) to Pin SDI and LCD starts transmitting Data(Bit-D15) to Pin SDO as well. So, user can use this timing to read data from LCD and store them simultaneously to use in the future as preferred. The data that is read is Register Status of LCD or Data, depends on command that user has transmitted to read data at any Address Register as preferred.

4.1.2) Interface Parallel 8-bit MODE

LCD in this Mode rearranges Command and Data as same as other modes; however, it transmits 8-Bit Command or Data (1 Byte) per one Signal Wave (WR) in each time according to Timing Diagram as shown in the figure 4.1.2C.

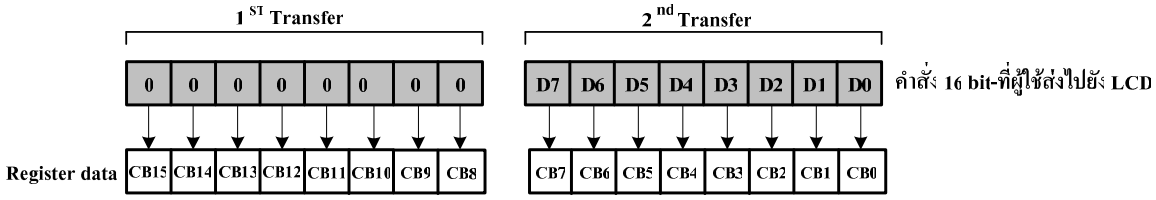


Figure 4.1.2.A displays Bit arrangement of incoming Command.

For the 1st Transfer Byte High(D15…D8) in the figure 4.1.2A above, it transmits value 00H because the Address Register of command is only 1 Byte and it is set at Byte Low(D7…D0).

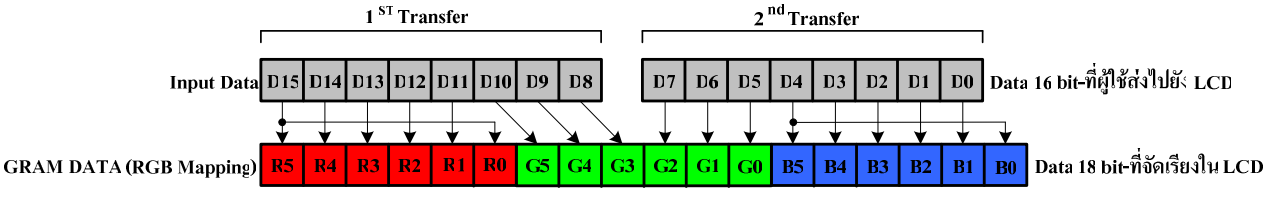


Figure 4.1.2B displays the arrangement of incoming colored Data bit that is 16 Bit 65K Colors.

From the figure 4.12B above, it arranges colored Data bit from the higher Bit to the lower Bit as RGB according to the ETT examples. However, user can change and modify the arrangement of colored bit into BGR by using Command Entry Mode(R03H) to set the new arrangement of colored bit. User can read more information from Data Sheet of SPFD5408A.

If user needs to transmit data to display dot on the LCD Display, it needs to mix colors by self by referring to the arrangement of colored bit as shown in the figure above. If it is Data D15-D11(5bit), it is red color; if it is Data D10-D5(6bit), it is green color; and if it is Data D4-D0(5bit), it is blue color. When user transmits Data into LCD, Data will be rearranged as 18-Bit automatically as shown in the figure. Shade of colors is arranged from darker to brighter or from the lower bit to the higher bit. For example, if user wants the brightest red color, it is Data = 0xF800; on the other hand, if user wants the darkest green color, it is Data = 0x0020. If user needs colors more than these 3 main colors, it needs to set Data Bit in the proper ranges of each color. For example, if it is Data = 0xFFFF, it is white color; or if it is Data = 0x0000, it is black color.

Procedures to write Command and Data from MCU to control LCD as Parallel 8-Bit MODE

The method to transmit Command or Data from MCU to LCD in this mode, it always needs to transmit Data Bit to Pin DB8-DB15 of Board LCD of ETT according to the circuit above. It transmits 1 Byte data twice per transmitting one Command or Data in each time. The first 2 Byte is the Command that always transmits first and then follow by the last 2 Byte that is Data of the Command. In this case, the 2 Byte that is Command or Data is transmitted, the 1st Byte needs to be Byte High and the 2nd Byte needs to be Byte Low. User can consider timing of reading-writing Command or Data into LCD as shown in the Timing Diagram below;

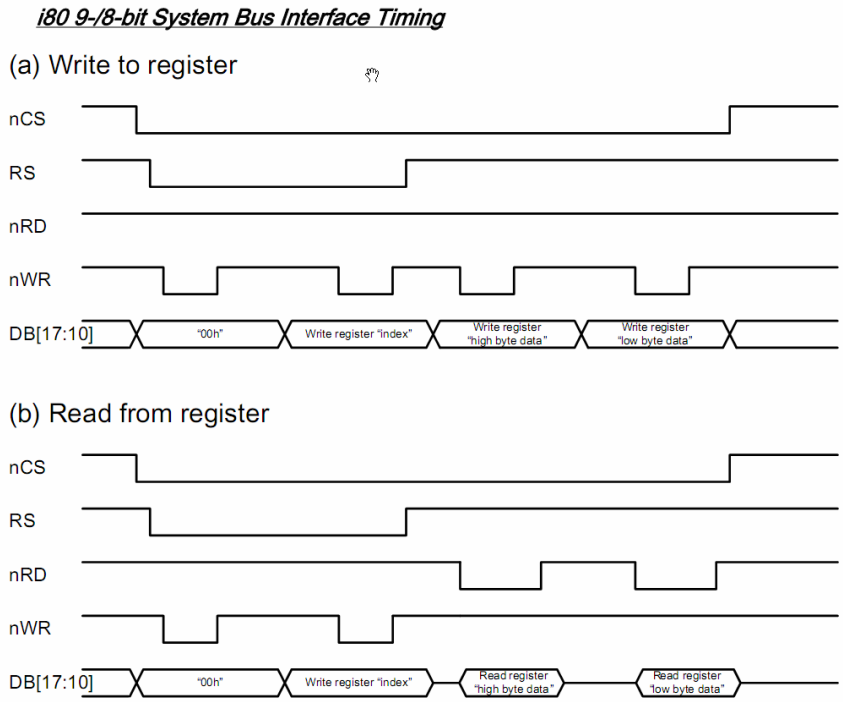


Figure 4.1.2C displays Timing Diagram to Read-Writ Command and Data into LCD as Parallel 8-Bit Mode.

From Timing Diagram above, it only describes how to write Data but it does not mention how to read Data because the example in CD-ROM does not read any data from LCD. In this case, it uses Delay instead; however, it can control LCD as well.

First of all, when user considers the Timing Diagram in the part of writing Data, it needs to transmit 4 sets of data to LCD in each time. The 1st set (1 Byte), it is the set of Command Byte High that is 00H; the 2nd set (1 Byte), it is the set of Command Byte Low that is the Address position of Register Index (Register No.); the 3rd set (1 Byte), it is the set of Data Byte High; and the 4th set (1 Byte), it is the set of Data Byte Low of the Command. Next, user can start transmitting Command according to Timing Diagram as summarized below;

1). Set Pin RD,CS to be 1.

2). Set Pin CS to be 0 to Enable LCD for receiving data.

3). The 1st Byte, it transmits Command Byte High that always is 00h to Pin DB8-DB15 (refer to the actual pin on board).

4). Set Pin RS to be 0; the value that is transmitted is Command.

5). Set Pin WR to be 0; it starts writing the 1st Command Byte.

6). Set Pin WR to be 1; the 1st Command Byte has already been transmitted.

7). The 2nd Byte, it transmits Command Byte Low that is the value of Register Index to Pin DB8-DB15.

8). Set Pin WR to be 0; it starts writing the 2nd Command Byte.

9). Set Pin WR to be 1; the 2nd Command Byte has already been transmitted.

10). Set Pin RS to be 1; it ends the process of writing Command and the next data that will be transmitted is Data. After transmitted the command set completely, it will follow by Data set of the Command as described below;

11). Still set Pin CS to be 0 and still set Pin RS and RD to be 1 for transmitting Data.

12). The 3rd Byte, it transmits Data Byte High to Pin DB8-DB15.

13). Set Pin WR to be 0 to start writing the 3rd Data Byte.

14). Set Pin WR to be 1 and the 3rd Data Byte has already been transmitted.

15). The 4th Byte, it transmits Data Byte Low to Pin DB8-DB15.

16). Set Pin WR to be 0; it starts writing the 4th Data Byte.

17). Set Pin WR to be 1 and the 4th Data Byte has already transmitted.

18). Set Pin CS to be 1; it ends the process of writing Command and Data. When user wants to transmit the next Command or Data, please start step No.1 again.

4.1.3 Interface Parallel 16-Bitr MODE

LCD in this mode rearranges Command and Data as same modes as mentioned above; however, the format of transmitting Command or Data is different because it transmits 16 Bit (2 Byte) Command or Data per one Signal Write(WR) in each time according to Timing Diagram as shown in the picture 4.1.3C.

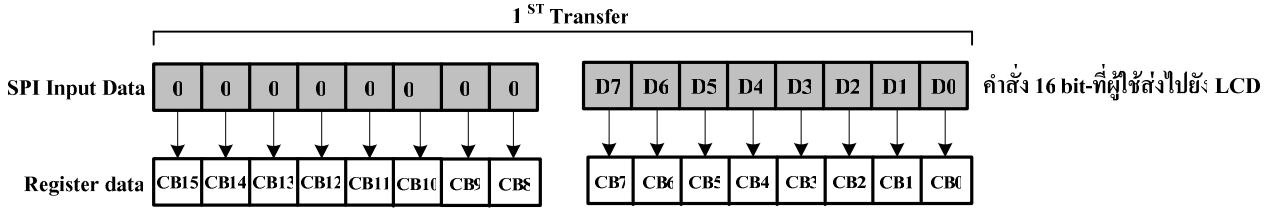


Figure 4.1.3A displays the Bit arrangement of the incoming Command.

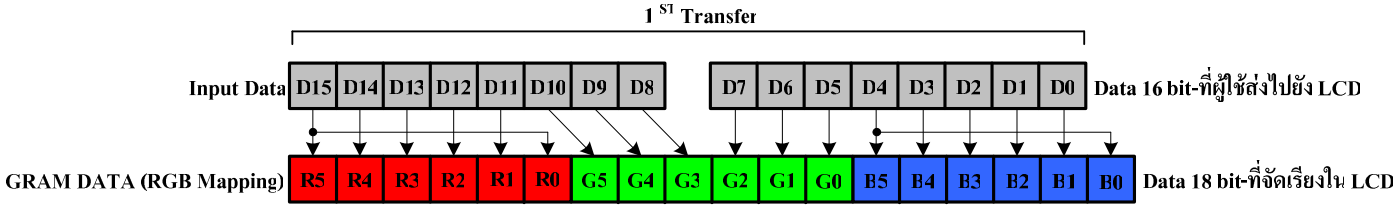


Figure 4.1.3B displays the arrangement of the incoming colored data bit that is 16 bit 65K Colors.

From the figure 4.13B above, it arranges colored data bit from higher bit to lower bit as RGB according to ETT examples. However, user can change and modify the arrangement of colored bit to be BGR by using Command Entry Mode(R03H) to reset the arrangement of colored bit as preferred. Please read more information from Data Sheet of SPFD5408A.

If user wants to transmit data to display dot on LCD Display, it needs to mix colors by self by referring to the arrangement of colored bit as shown in the figure above. If it Data D15-D11(5bit); it is red color; if it is Data D10-D5(6bit), it is green color; and if it is Data D4-D0(5bit), it is blue color. When user transmit Data to LCD, the Data will be rearranged as 18 Bit automatically as shown in the figure; in this case, the shade of colors arranges from darker to the brighter or from lower bit to the higher bit. For example, if user needs the brightest red color, it is Data = 0xF800; on the other hand, if user needs the darkest green color, it is Data = 0x0020. If user needs colors other than these 3 main colors, it needs to mix bit data to be in the proper range of each color and user will get the desired colors; for example, if it is white color, it is Data = 0xFFFF; or if it is black color, it is Data = 0x0000.

Procedures to Write Command and Data from MCU to control LCD as Parallel 16-Bit MODE

For the method to transmit Command or Data from MCU to LCD in this mode, it needs to transmit data to Pin DB0-DB15 of Board LCD of ETT according to the circuit above. In this case, it transmits 2 Byte data per one Command or Data in each time. It divides the value as follows; the first 2 Byte is the Command that needs to transmit first and then follows by the last 2 Byte that is Data of the Command. User can consider the timing of writing-reading Command or Data to LCD from Timing Diagram as follows;

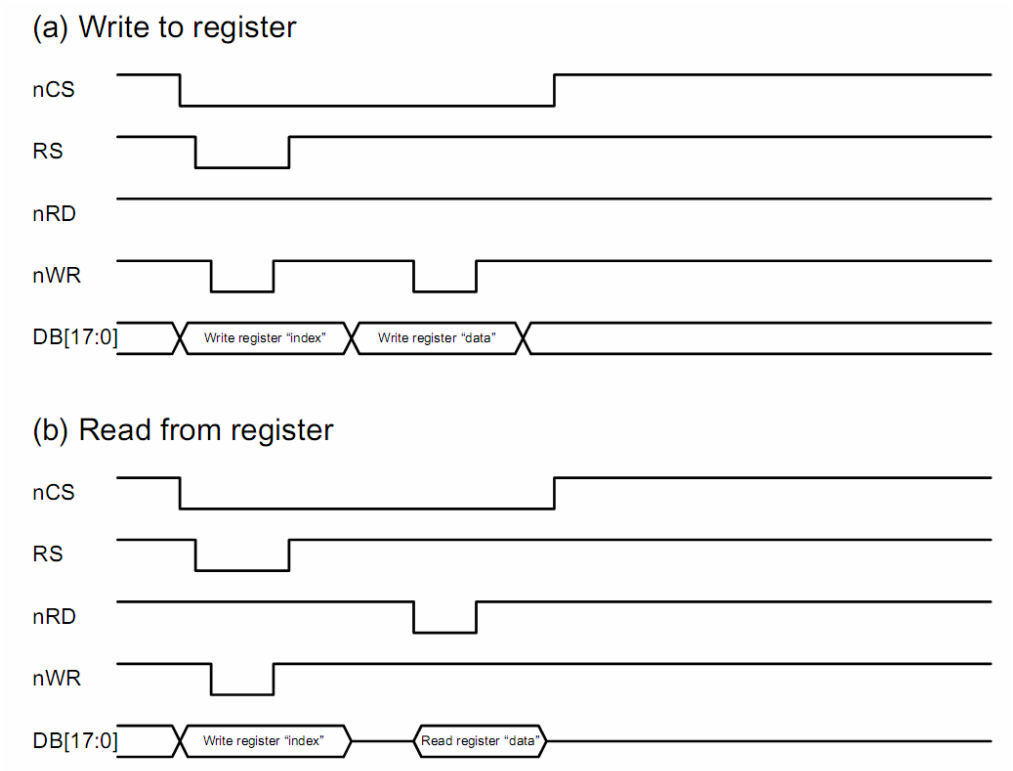


Figure 4.1.3C displays Timing Diagram to Read-Write Command and Data to LCD as Parallel 16-Bit Mode.

From Timing Diagram above, it only describes how to write Data but it does not describe how to read data because the ETT example does not read any data from LCD. However, it uses Delay instead and it can control LCD as well.

First of all, when user considers Timing Diagram in the part of writing data, there are 2 sets of data that are transmitted to LCD in each time. The 1st set(2 Byte), it is the Command set that is 00H+ Address position of Register Index (Register No.); and the 2nd set(2 Byte), it is the 16-Bit Data set of the Command. Next, user starts transmitting Command according to Timing Diagram as summarized below;

1). Set Pin RD,CS to be 1.

2). Set Pin CS to be 0 to Enable LCD for receiving Data.

3). Transmit 16-Bit Command to Data Bus(DB0-DB15); in this case, Data Bus 8 Bit Upper(DB8-DB15) is set to be 0x00.

4). Set Pin RS to be 0; data that is transmitted is Command.

5). Set Pin WR to be 0; it starts writing the first 2 Byte Command.

6). Set Pin WR to be 1; the first 2 Byte Command has already been transmitted.

7). Set Pin RS to be 1; it ends the process or writing Command; moreover, the next data that will be transmitted is Data.

After transmitted command completely, it follows by Data set of the Command as described below;

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

9). Transmit 16-Bit Data of the Command set to all of 16-Bit Data Bus(DB0-DB15).

10). Set Pin WR to be 0; it starts writing the last2Byte Data.

11). Set Pin WR to be 1; the last 2 Byte Data has already been transmitted.

12). Set Pin CS to be 1; it ends the process of transmitting Command and Data. If user wants to transmit the next data, please start step No.1 again.

From steps above, if user wants to transmit other data, please start the step No.1 again. The method to write program, user maybe write function to receive command and value of Data into function simultaneously and then transmit Command and Data as described above. Moreover, user maybe follow ETT example that divides data into 2 functions; one function for transmitting Command and other one function for transmitting Data.

4.2) Interface Control Touch screen: In the part of this Touch Screen, it divides Control from LCD; moreover, there are 2 types of Interface. Firstly, it directly connects Pin Y-,Y+,X-,X+ with Pin ADC of MCU and then write program to control the process of reading value by self. However, it is quite difficult to write program because user needs to know and understand principles regarding the operation of Touch Screen well. That the reason why we don't recommend user to use this Interface.

In this case, we will recommend the Interface through Chip ADS7846 according to ETT example. When user has chosen this Interface type, it needs to shift all DIP SW1(S1-S4) at the back board to the position ON to interface Pin X+,Y+,X-,Y- of Touch Screen with Chip ADS7846 (normally, it have already been set at position Default). If the Interface uses this Chip ADS7846, it uses SPI Interface between MCU and Chip. User can read more detailed communication of data and Chip to read-write data at the position of Touch Screen from Data Sheet "Touch_ADC7846N.pdf". First of all, user needs to know and understand the cooperation of Touch Screen and ADS7846.

- Application of Touch Screen and ADS7846: The method to read Address position of Touch Screen is to start operating after user touched Touch Screen; Chip ADS7846 converted Signal Analog that has received through PIN X+,Y+,X-,Y-; and the Digital value is transmitted through Pin Serial Data Out. The ADC value that has ready read is 12 Bit; so, 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 will transmit Signal Interrupt logic "0" for a while. While writing program, user needs to read status value of this Signal Interrupt to check whether Touch Screen is touching or not. Because it needs not to loop to read Address position of Touch Screen all the time, it only reads when the Touch Screen is touched; so, program can operate in other parts.

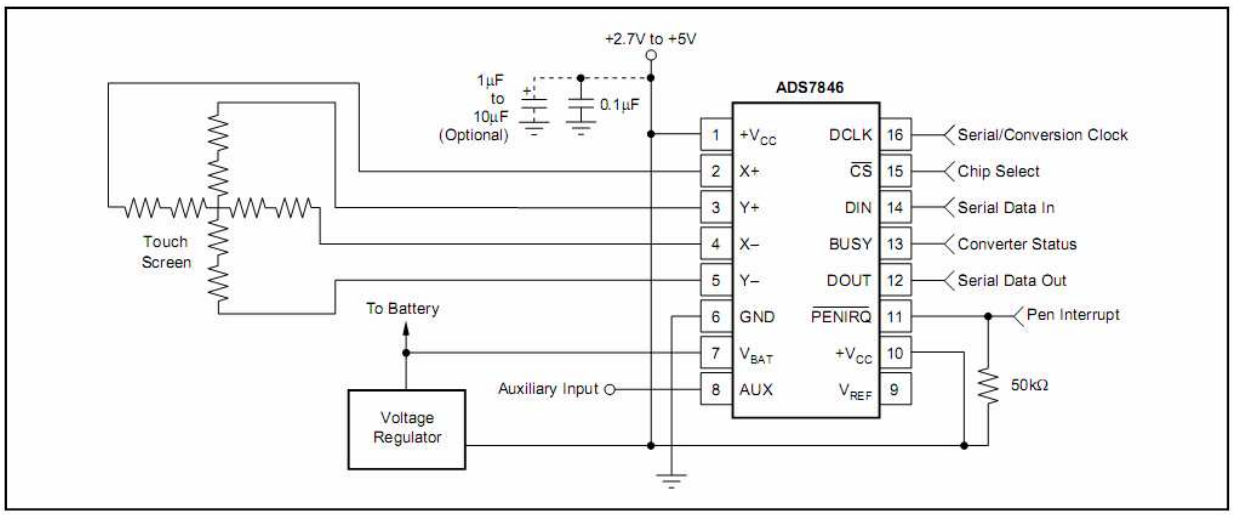


Figure 4.2.1 display how to connect circuit of Touch Screen and ADS7846.

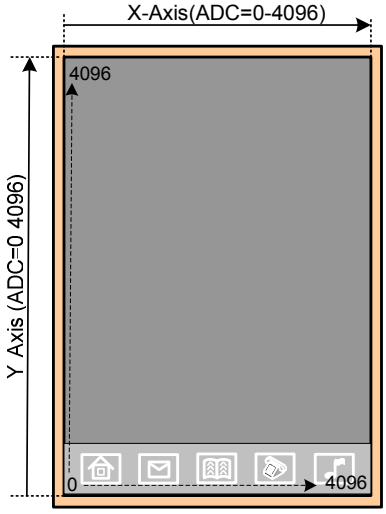


Figure 4.2.2 displays area of Touch Screen and the ADS value that has occurred on the line of X and Y axis.

From the figure 4.2.2 above, it displays direction of X and Y axis of Touch Screen and the ADC value that is read from positions while Touch Screen is touching. Normally, the minimum ADC value that is read through ADS7846 is 600 approximately (not 0) that is Offset value of screen; however, each screen can read this initial value unequally, it always needs to write program to Calibrate the screen first. User can copy and use ETT example instantly; in this case, it uses matrix principles to calculate coefficient value that is calibrated by user. There are 3 points in the example program and user needs to touch them accurately according to the marked position; after calibrated, it makes actual operation high accurate.

- Timing Diagram to read-write data through ADS7846: After user has known and understood initial principles in the part of Touch Screen well; next, it describes how to read ADC value from Touch Screen by using Chip ADS7846. First of all, user needs to understand that the ADC value that is read by touching Touch Screen is not the actual address position and it can not refer to any address position on the LCD Display. User needs to take this value to calculate to find

the actual address position on LCD Display again. When user got the actual address position, user can replace this value in command that controls position of LCD Display instantly. All these processes can be read from ETT examples.

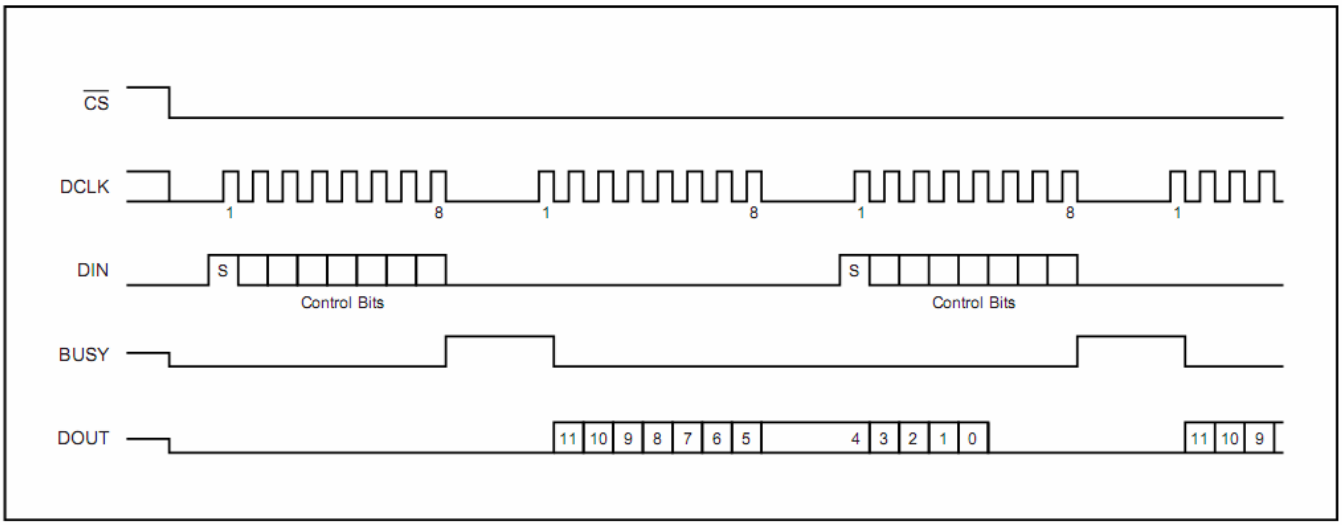


Figure 4.2.3 displays Conversion Timing Diagram, 16 Clock-per-Conversion, 8bit bus Interface.

This Timing Diagram is the process of reading ADC Value from Touch Screen's plate through Chip ADS7846 by using MCU Interface and SPI ADS7846 according to ETT Example. The SPI Communication in the example uses Pin I/O of MCU to build by self without using Module SPI internal MCU; so, user can change and modify program easily. The principle to transmit data as SPI is to transmit data to Pin MOSI(Dout) 1 Bit and then follow by 1 Clock successfully; Data will be shifted into Chip 1 Bit and Chip will shift out data to Pin MISO(Din) 1 Bit as well; in this case, this Data is the value that user needs to read and store this value. Function *tcs_wr()* in the example is to write and read data as 1 Byte(8 Bit) Serial. When user has built this function successfully, user can transmit Control Byte and then read and store the ADC through this function. The procedures to read the ADC value from Touch Screen are described below;

The method to read the ADC value from ADS7846, user always needs to transmit Control Byte to ADS7846 to set specifications into Chip before reading the value back. The format of Control Bit is shown as below;

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

Figure 4.2.4 displays Control Byte of ADS7846.

We do not mention detail of Control Byte of each Bit but user can read more detail from Data Sheet by self. This example uses Control Byte that is 0xD0 to read ADC value on the X axis of Touch Screen and 0x90 to read ADC value on the Y axis of Touch Screen.

Summarize procedures to read ADC Value from ADS7846

1). Read Status value from Pin PEN of ADS7846; if it is 0 (touching Touch Screen), it starts reading the next step NO.2; on the other hand, if it is 1 (Touch Screen has not touched yet), it needs to loop reading.

2). Set Pin DCLK,CS,DOUT to be 0

3). Transmit Control Byte 0xD0 to Pin DIN(MOSI) of ADS7846 to specify the 12Bit ADC value that is read on the X axis.

4). Transmit Data 0x00 to Pin DIN(MOSI) of ADS7846; while transmitting Data in each bit, ADS7846 will shift the ADC value to Pin DOUT(MISO). The 1st Data that is shifted is the 11th Bit and it starts at the falling edge pin of the 2nd DCLK. When all of 8 DCLK are transmitted completely, the Data result that can be read on the X axis is 0x0ddddddd (d=data bit11-bit5).

5). Transmit Control Byte 0x90 to Pin DIN(MOSI) of ADS7846 to specify the 12 Bit ADC value that is read on the Y axis. While transmitting this Control Byte, the last 5 Bit Data ADC on the X axis will be transmitted as well; in this case, it starts from bit4 to bit0 and the Data arrangement is 0xddddd000 (d=data bit4-bit0).

6). Transmit Data 0x00 to Pin DIN(MOSI) of ADS7846; while transmitting Data in each bit, ADS7846 will shift the ADC value to Pin DOUT(MISO). The 1st Data Bit that is shifted is the 11th Bit and it starts at the falling edge pin of the 2nd DCLK. When all of 8 DCLK are transmitted successfully, the Data result that can be read on the Y axis is 0x0ddddddd (d=data bit11-bit5).

7). Transmit Data 0x00 to Pin DIN(MOSI) of ADS7846; while transmitting this Data, the last 5 Bit data ADC on the Y axis will be transmitted. In this case, it starts from bit4 to bit 0 and the Data arrangement is 0xddddd000 (d=data bit4-bit0).

8). When the ADS values on both axes are read, it needs to set Pin CS as 1 to end the process of reading value from ADS7846.

9). When user wants to read new value, it starts the step No.1 again.

10). After user has got the ADC value on each axis successfully, user needs to rearrange the value on each axis. The variable that stores the ADC value should be 16 Bit variable; in this case, the first 7 Bit data ADC that is stored in 16 Bit variable is 0x000000000ddddddd and the last 5 Bit that is stored in 16 Bit variable will be 0x00000000ddddd000. Next, it shifts the first 7 Bit that is read to the left side 5 Bit and then shifts the last 5 Bit to the right side 3 Bit. Then, user takes the both sets of data to OR(|) and user will get 12 Bit data ADC of the axis that is read as 0x000ddddddddddddd; this is the value that user can apply.

From the principles of Control LCD and Touch Screen above, it is the general process of using Board ET-TFT240320TP-3.2 REV.B. When user has known and understood basic principles to control operation well; so, user can copy data in the part of function in the ETT examples to write actual program. User can read more principles to use function in the topic "Description of Example Program".

5. Application of Program Embedtools 3.31 Convert image files into hex code

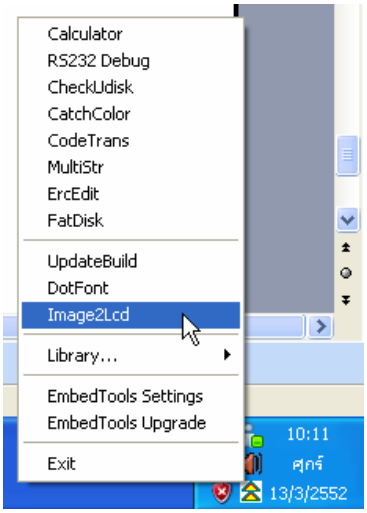
This subject describes how to convert image file into hex code and then transmit the hex code to LCD Display to display images as preferred; in this case, it uses Program "Embedtools 3.31". The data that is converted into hex code supports ETT example by using function "plot_picture()"(it is in example "Ex3_Touch_Button") to transmit the hex code from MCU to LCD.

If user does not use *initial lcd* according to ETT example and does not set Program Embedtools to convert any image as described above, user might not use *Function "plot_picture()"* because it makes direction of transmitting data incorrectly. User needs to write program to plot image again and it needs to correspond with the setting values by self.

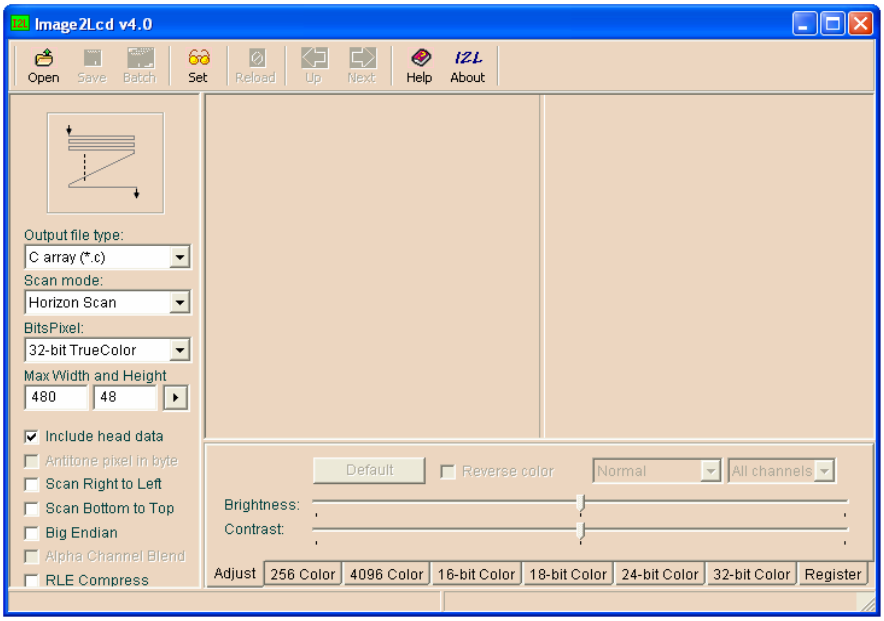
Procedures to use Program Embedtools Convert image file into Hex Code

1). Install Program Embedtolls.exe into computer (it is in Folder Embedtools 3.31).

2). After installed program successfully, there is Icon ( ) on the Task Bar as shown in the picture 5.1. Next, right-click at the Icon, it displays Tab; left-click to choose **Image2LCD** to run Program Embedtools and it will display window as shown in the picture 5.2.

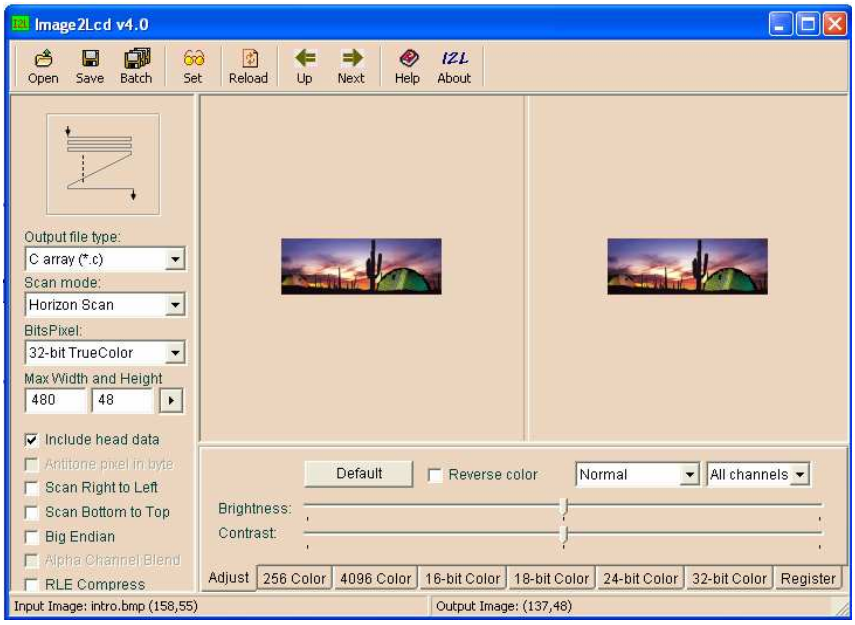


Picture 5.1



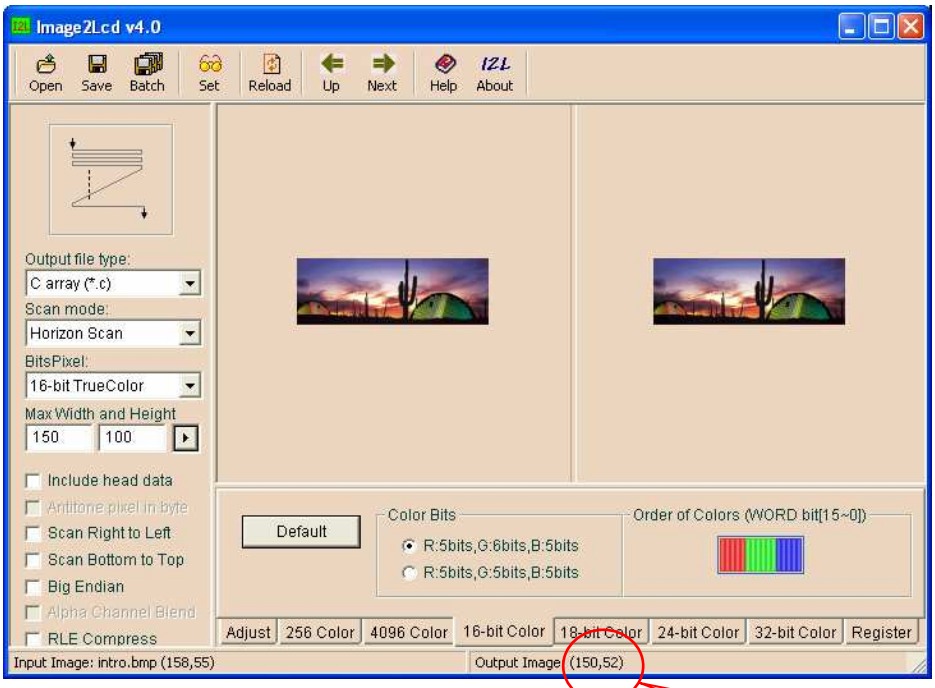
Picture 5.2

3). Click **Icon Open** ( ) on the top of program to choose image file to convert into hex code. In this case, the image file should be file surname .jpg or .bmp and finally, when it opens the image file, it will display window as shown in the picture 5.3.



Picture 5.3

4). After loaded image file into program successfully, set values as follows (see picture 5.4 as reference);



Picture 5.4

- Output file type: *C array (*.C)* = It sets Data to be the format of Array of C Language and then save Output file as .C
- Scand mode: *Horizon Scan* = It sets initial direction to Scan Data; if using data, it needs to transmit Data to LCD according to the direction that has been scanned.
- Bits Pixel: *16-bit TrueColor* = It sets the fine of colored bit (there are 2 same values in this program, please choose the upper value).

- Max Width and Height: Width, Height = It sets width and height for the image size. After set the values successfully, click button ( ), it displays change in the image size; in this case, its unit is Pixel. However, it is not the actual image size; if using Function *"plot_picture()"*, ETT example needs to use the actual width and height of image; in this case, user can see them from the gap of Output Image.

- Include head data = It removes the sign tick.

- TAB 16-bit Color = Click TAB 16-bit color in the gap of Color bit, choose R:5bit,G:6bit,B:5bit; in the gap of Order of Colors (WORD bit[15~0]), choose the color arrangement as RGB; and don't set any value for other TAB.

5) When set all values successfully, click Icon **Save** (Save) to Save file hex code. If user wants to use it, only use Notepad to open this file, copy the hex code and then paste it in the Editor for writing program.

6. Application of Example Program

The ETT example is written by C Language and supports 3 MCU families; AVR(Mega128), PIC(18F8722), and ARM7(LPC2138). This example includes the method to Control LCD in the format of horizontal and vertical lines, depends on user's requirements. If user chooses any format, user can use ETT example to be the main reference. Example of each family is the same because it refers to the initial address position on the X, Y axis of the display's direction as shown in the picture below;

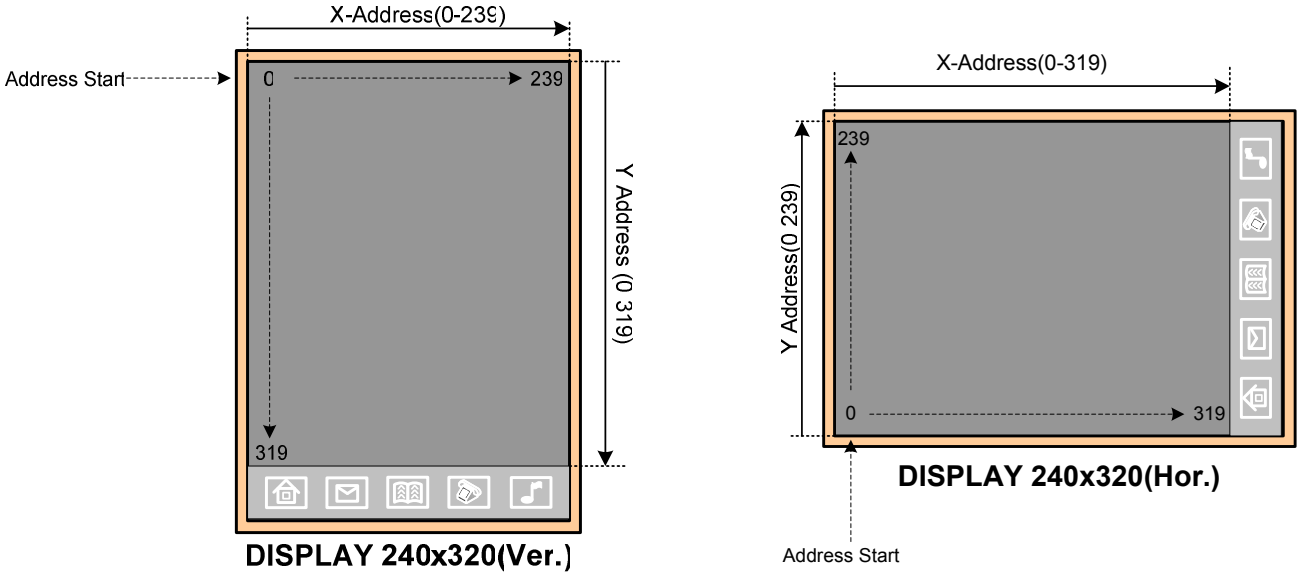


Figure 6.1 displays how to refer address position on LCD Display that is horizontal and vertical lines.

From the picture above; if using this example function, it needs to refer to the address position above to set address position of X, Y to start *Plot* image or character.

There are 3 main examples that are horizontal and vertical lines as described below;

- Ex1_Touch_Position: When run program by using this example; first of all, user needs to touch 3 points that are sign '+' to calibrate in the part of Touch Screen. After calibrated successfully and user touched the screen, it makes MCU can read

the address position high accuracy and correspond with the actual address position of LCD as shown in the picture 16. After calibrated successfully, when user touch any address position on LCD Display, it will display the value of address X,Y that is touched by user.

- *Ex2_Touch_Draw*: When run program by using this example, first of all, user needs to calibrate Touch Screen as same as the example 1 above. Next, user can draw any picture or write any character on screen and the screen will display lines as same as user did. Moreover, user can touch Icon Musical Note ( ) to change color of line and Icon Home ( ) to Clear Screen.

- *Ex3_Touch_Button*: When run program by using this example, first of all, user still needs to calibrate Touch Screen as described above. Next, it displays buttons to touch; when user touched any button on the screen, in the blank window will display pictures according to the button that is touched.

In the part of Calibrate Touch Screen in this example; every time when user resets MCU, user needs to calibrate repeatedly. However, user maybe calibrate only one time for actual application as described below;

The 1st Method: Interface E2Promt with MCU additionally (if MCU has not any internal E2PROMP). Next, internal Function *"touch_calibrate()"* that is next to the line of calling Function *"set_matrix()",* user needs to write program *write* value that is in the variable *divider,An,Bn,Cn,Dn,En,Fn* to store in E2Promt and then write any value for 1 Byte to store in E2Promt; this value is Flag Status to check whether it has already calibrated or not.

Next, in the part of main program before calling function *"touch_calibrate()",* user needs to read Status Flag from E2Prompt to check whether it is the same as value that has already been written and stored by user or not. If yes, it need not to call function *"touch_calibrate()"* repeatedly; user can read value of variable divider,An,Bn,Cn,Dn,En,Fn that has stored in E2Prompt to replace value of this variable divider,An,Bn,Cn,Dn,En,Fn instantly; and finally, program can operate in other parts without any error. This method helps user not waste much time to calibrate LCD.

The 2nd Method: This method need not use any E2Prompt but it adds Command *printf()* in function *"touch_calibrate()"* that is next to the line of calling Function *"set_matrix()"* (it has already been written in this example but it is disabled) to print value of variable *divider,An,Bn,Cn,Dn,En,Fn* and display the operating result through RS232. It uses Program Hyper Terminal to receive the value of variable that is printed to display results to user; next, user needs to write down and remember the value of each variable because it is used to set value for the variable *divider,An,Bn,Cn,Dn,En,Fn* that is declared above *main()*. If user wants to use this value, user can ignore and remove function *"touch_calibrate()"* because it is not called any more. If using this method with many screens, user maybe writ program to calibrate to find the value *divider,An,Bn,Cn,Dn,En,Fn* separately because user always needs to calibrate to find the new value repeatedly although it is the same type of screen. Remember, the value that is read from the old screen is unable to use for new screen because the address value that is read from touching screen does not correspond with address position of the actual LCD Display.

NOTE: The method to calibrate is to touch screen nearest the marked position to makes high accuracy.

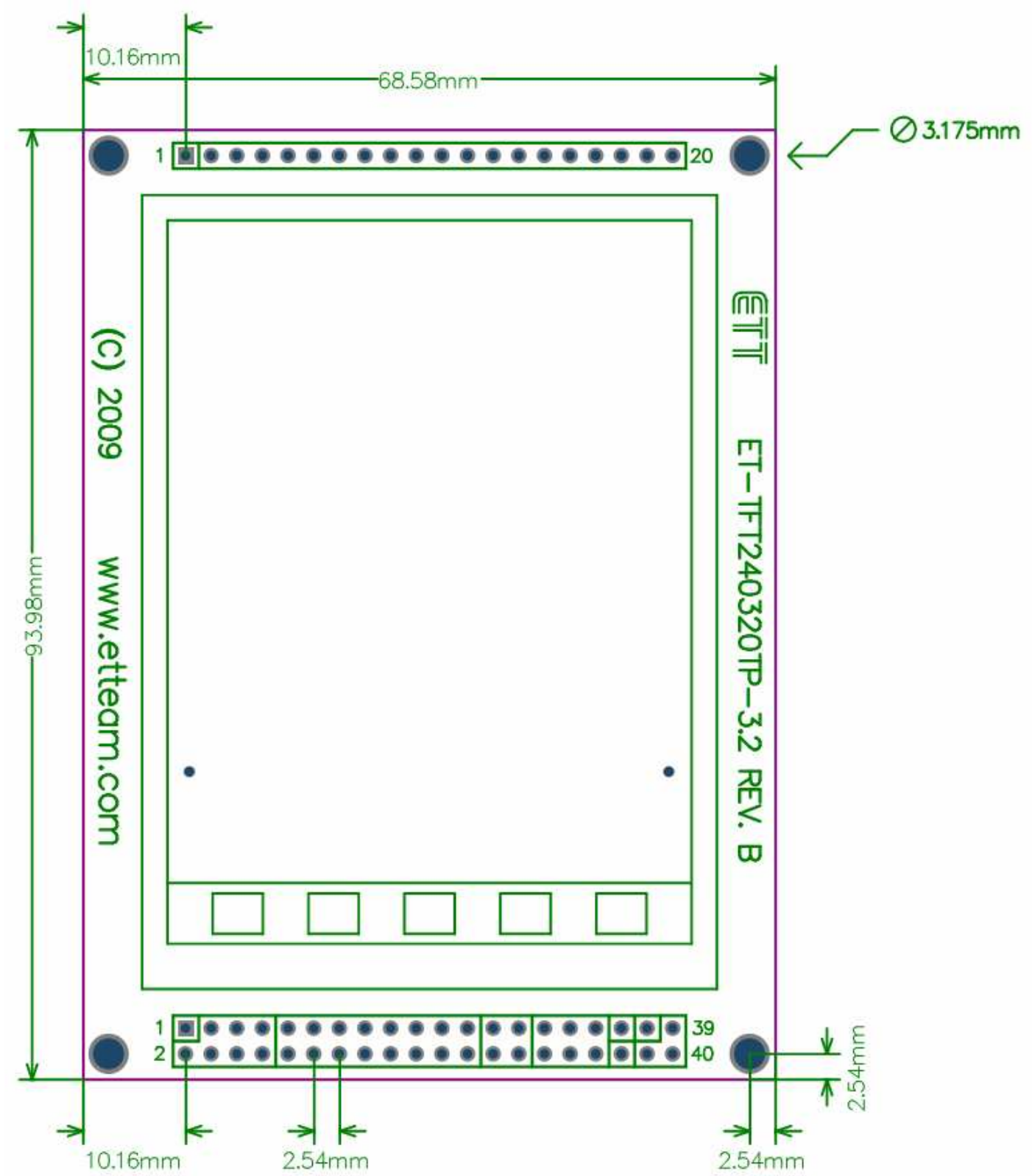


Figure 6.2 Displays size of Board ET-TFT240320TP-3.2 REV.B.

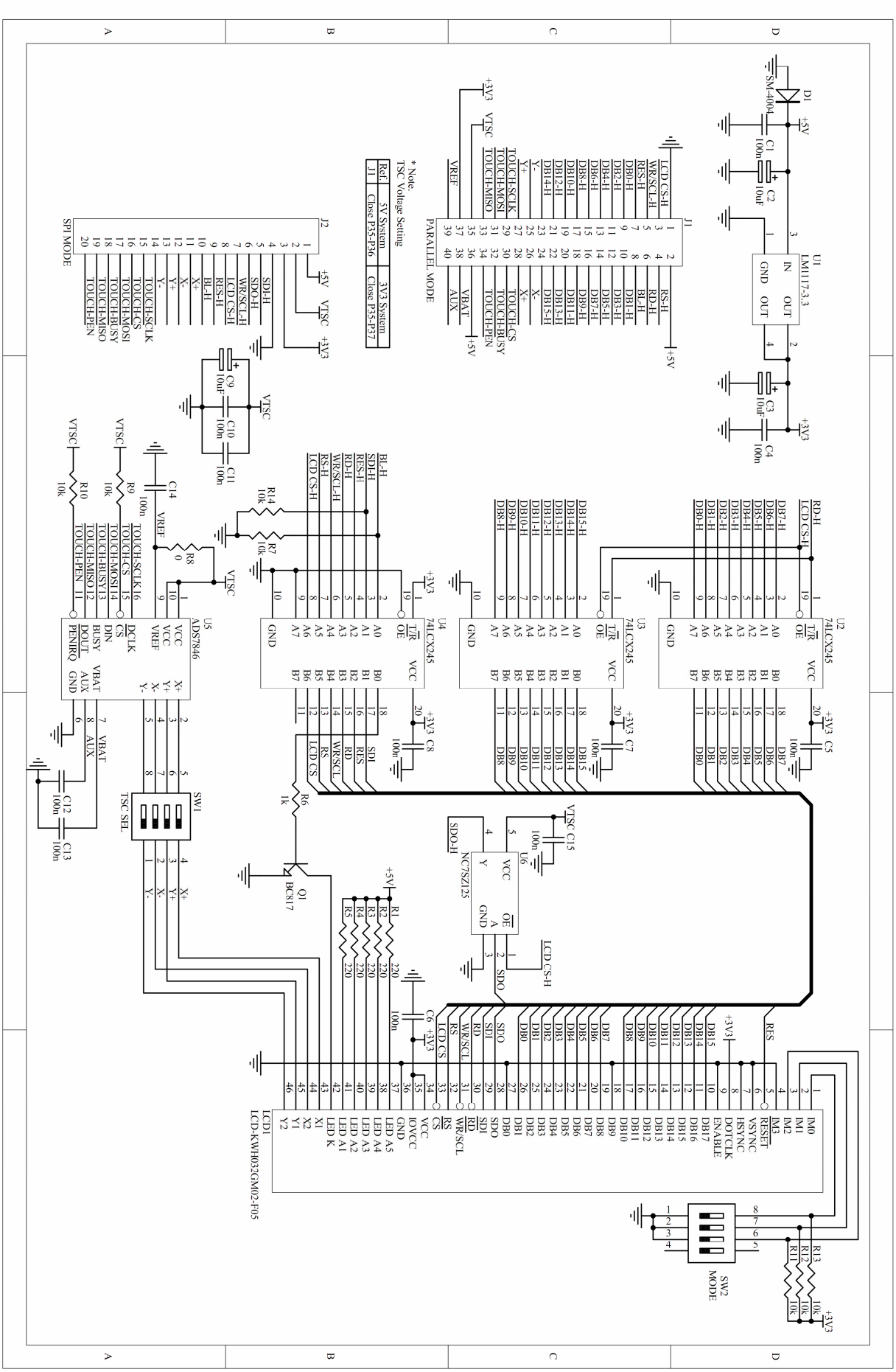


Figure 6.3 Displays circuit of ET-TFT240320TP-3.2 REV.B.