ET- TFT240320TP-2.8

1. Specifications of Board ET-TFT240320TP-2.8

- Use Display Module TFT LCD Color + Touch Screen 240x320 Pixel
- Be 2.8" wide TFT screen
- Use Single Chip Driver No.HX8347-D
- Has high resolution of 65536 colors(RGB=R:5bit,G:6bit,B:5bit)
- Has 2 modes of Interface GLCD through DIP SW2.(MODE)
  1) Parallel Mode 16-bit Interface
  2) Parallel Mode 8-bit Interface
- Has 2 Interface Modes for controlling Touch Screen by using DIP SW1.TSC SEL. Firstly, it is SPI Interface that is interfaced through Chip Touch Screen Controller #ADS7846 (12BIT ADC). Secondly, it directly interfaces through Pin X-,X+,Y-,Y+ with Pin ADC of MCU (however, it is difficult to write program for controlling).
- When using this board but user does not require using Touch Screen, user can control in the particular part of LCD only.
- Can summarize amount of I/O of MCU for controlling the part of GLCD and Touch Screen as follows;
  1) When using Parallel Mode 16-Bit Interface, it uses 27-PIN I/O.
  2) When using Parallel Mode 8-Bit Interface, it uses 19-PIN I/O.
- Can interface with MCU that uses either 5V or 3.3V (read more information in the part of "How to use")
- Use Pin Header 2x20, 2.54mm Pitch to be Connector for using Parallel Mode 8Bit and 16Bit.
- Use +5VDC Power Supply for board

2. Feature and Structure of Board ET-TFT240320TP-2.8

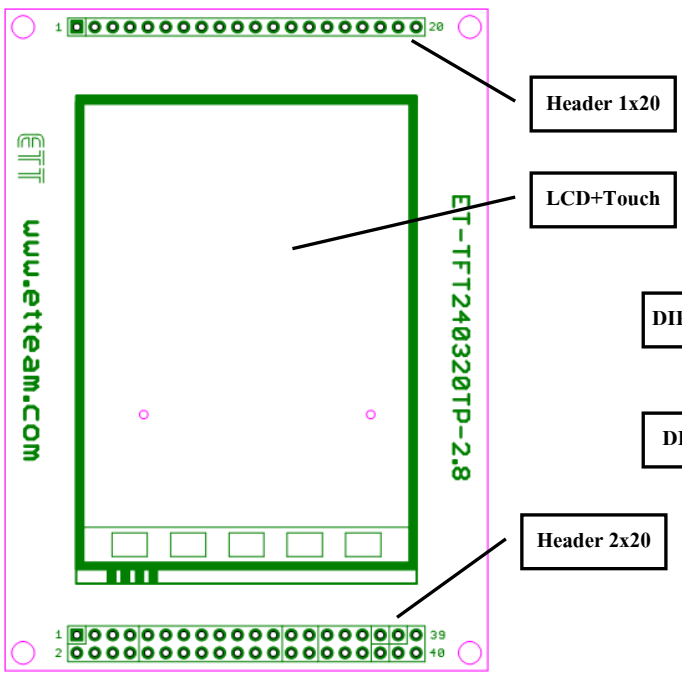


Figure 2.1 (A) Front board

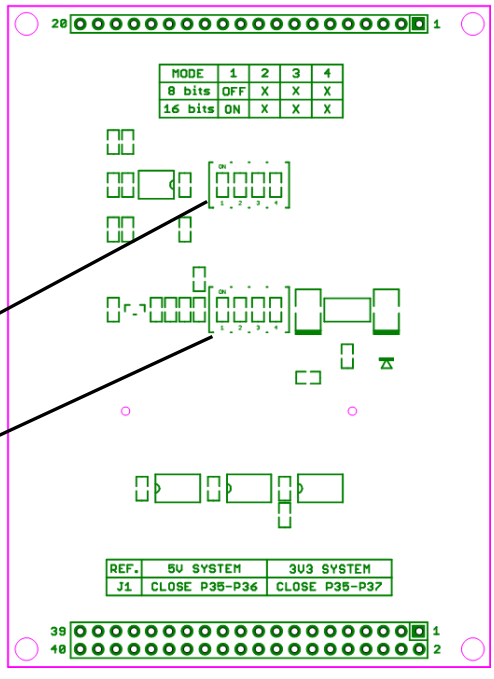


Figure 2.1 (B) Back board

- ***LCD+Touch***: It is area of LCD with 240x320 Pixel. The top of Screen is shielded by plate of Resistance Touch Screen.

- ***Header 1x20***: It is Connector 1x20 Pin MALE to interface with Pin Control LCD and Pin that is used to control IC Touch Screen is interfaced to other side. These pins are interfaced in parallel with pins on the side of Header 2x20.

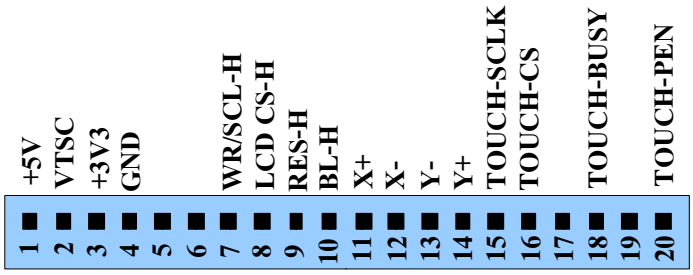


Figure2.2 pin position at Header 1X20 (referred to front board in the figure 2.1 (A)).

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

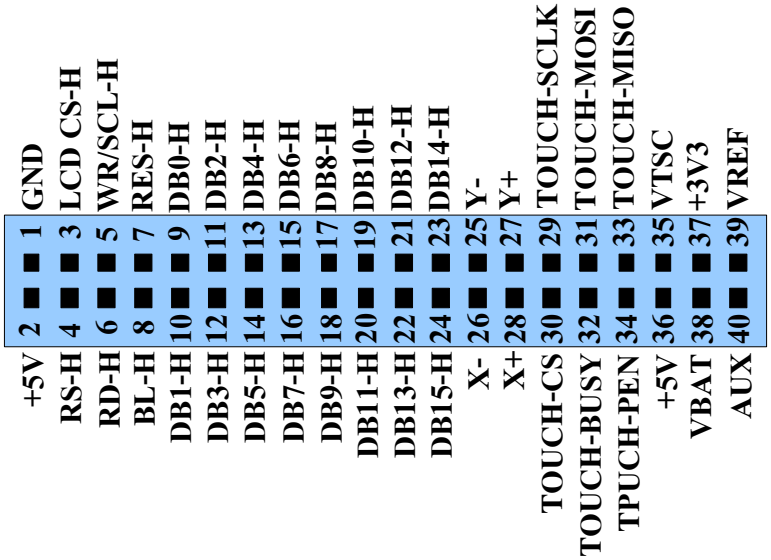


Figure 2.3 pin position at Header 2x20 (Parallel-Mode: 8Bit, 16Bit) (referred to front board in the figure 2.1 (A)).

Table 2.1: Details of PIN for controlling 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 for board
3	8	LCD CS-H	I	**Pin Chip-Select:** It is used to Enable LCD Display (it can be used with both SPI Mode and Parallel Mode). **Low** = LCD Module Enable **High** = LCD Module Disable
4	-	RS-H	I	**Pin Register-Select:** **Low** = It accesses Index(IR) or Status(SR) Register. **High** = It accesses Control Register(Address 00H-FFH).
5	7	WR-H	I	**Pin Write Strobe:** It is used to write Data when it has already received Signal Low in Parallel Mode.
6	8	RD-H	I	**Pin Read Strobe:** It is used to read Data when it has already received Signal Low.
7	9	RES-H	I	**Pin RESET:** It is used to initial LCD Module when it has already received Signal Low.
8	10	BL-H	I	Pin Black Light: The Black Light of LCD is lit up when this pin has already received Signal High.
9-24	-	DB0-DB15	I/O	**Pin Data Bus Bi-Directional 16 Bit:** It is used to pass/send data or point to the Address position of Register; it is used for Parallel Mode.

Table 2.2: Details of Pins for controlling 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+:** These pins are used to read position of Touch Screen directly, not through Chip ADS7846. It has to shift all DIP SW1.TSC SCL to the position "OFF".
29	15	**TOUCH-SCLK**	I	It is Pin DCLK of ADS7846 for 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 running.
31	17	**TOUCH-MOSI**	I	It is Pin DIN of ADS7846. When Pin CS is Low, data is 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 is shifted at the Falling Edge Pin of DCLK and this Output is High Impedance when CS is High.

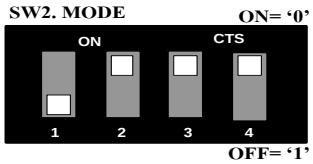
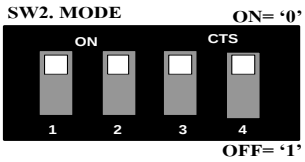
Table 2.2 (Continued)

No. PIN		PIN-NAME	I/O	Description
Header2x20 (Parallel Mode)	Header1x20 (SPI Mode)			
34	20	TOUCH-PEN	O	It is Pin PENIRQ of ADS7846. When Touch Screen is touched, it gives Signal Logic 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 it is used with MCU 5V, it has to jump Pin VTSC with Pin +5V; on the other hand, if it is used with MCU 3.3V, it has to jump Pin VTSC with Pin +3V3 (37).
38*	-	VBAT	I	It is Pin Vbat of ADS7846 but it is unused in the part of Touch Screen.
39*	-	VREF	I/O	It is Pin Vref of ADS7846 but it is unused in the part of Touch Screen.
40*	-	AUX	I	It is Pin AUX input to ADC of ADS7846 but it is unused in this part.

(*)= Unused pin, referred to the examples of ETT

- ***DIP SW1.TSC SEL on the back of board***: There are 4 DIP SW. If using Touch Screen in the format of SPI Interface through Chip Touch Screen Controller #ADS7846, it has to shift all 4 DIP SW. to the position ON (Default Value). Referred to the example program, it writes the program to support the communication of this operation mode as well. If using Interface Mode that interfaces Pin X-,X+,Y-,Y+ to with Pin ADC of MCU directly (ADS7846 is unused), it has to shift all 4 DIP SW. to position Default (OFF).
- ***DIP SW2.MODE on the back of Board***: There are 4 DIP SW. In this case, it uses only one switch that is S1. This DIP SW2 is used to choose the Interface Modes between Board GLCD and MCU that is interfaced for controlling the operation. In this case, it should choose the operation modes as shown in the Table 2.3

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

DIP SW.2 MODE				Format of setting SW2.	MODE Interface
S1	S2	S3	S4		
OFF	X	X	X		Parallel 8-bit Mode
ON	X	X	X		Parallel 16-bit Mode

X= whatever value

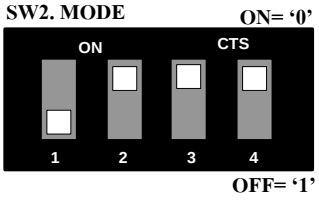
3. How to interface Board ET-TFT240320TP-2.8 with MCU

Now, we would like to describe about the method to interface Board ET-TFT240320TP-2.8 that supports the example program of ETT; in this case, it uses MCU AVR MEGA128, PIC18F8722 (for MCU 5V) and uses MCU ARM7 LPC2138 (for MCU 3.3V). Moreover, user can modify the circuit of interfacing MCU below to use with other MCU numbers or other series. ***Be careful with Pin VTSC, it has to jump Pin VTSC with Pin +5V or +3V3 correctly according to the actual connection, otherwise it makes MCU damaged.*** For example, if MCU runs by +5V, it has to jump Pin VTSC with Pin +5V correctly.

There are 2 Interface Modes; 8-Bit Parallel Interface Mode and 16-bit Parallel Interface Mode; in this case, user can shift DIP-SW2 on the back of board to choose Interfaces Mode. We can summarize example of interfacing circuits as follows;

3.1) 8-bit Parallel Interface MODE

How to set Interface Mode



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

This 8-bit Parallel Interface Mode uses Pin Data to only send-receive 8bit Data that is DB8-DB15. However, the part of PIN Interface LCD that is DB0-DB7 is unused; it should be interfaced with Ground as shown in the dotted line in the picture below (if it is floated, problem maybe happen).

How to interface with MCU 3.3V

Referred to this circuit, it interfaces Pin P0.2, P0.3, P0.11, and P0.14 of MCU with R Pull-Up because this Pin Port is Open Drain. If using MCU that has not any Pin Open Drain, it is unnecessary to interface with R Pull-Up.

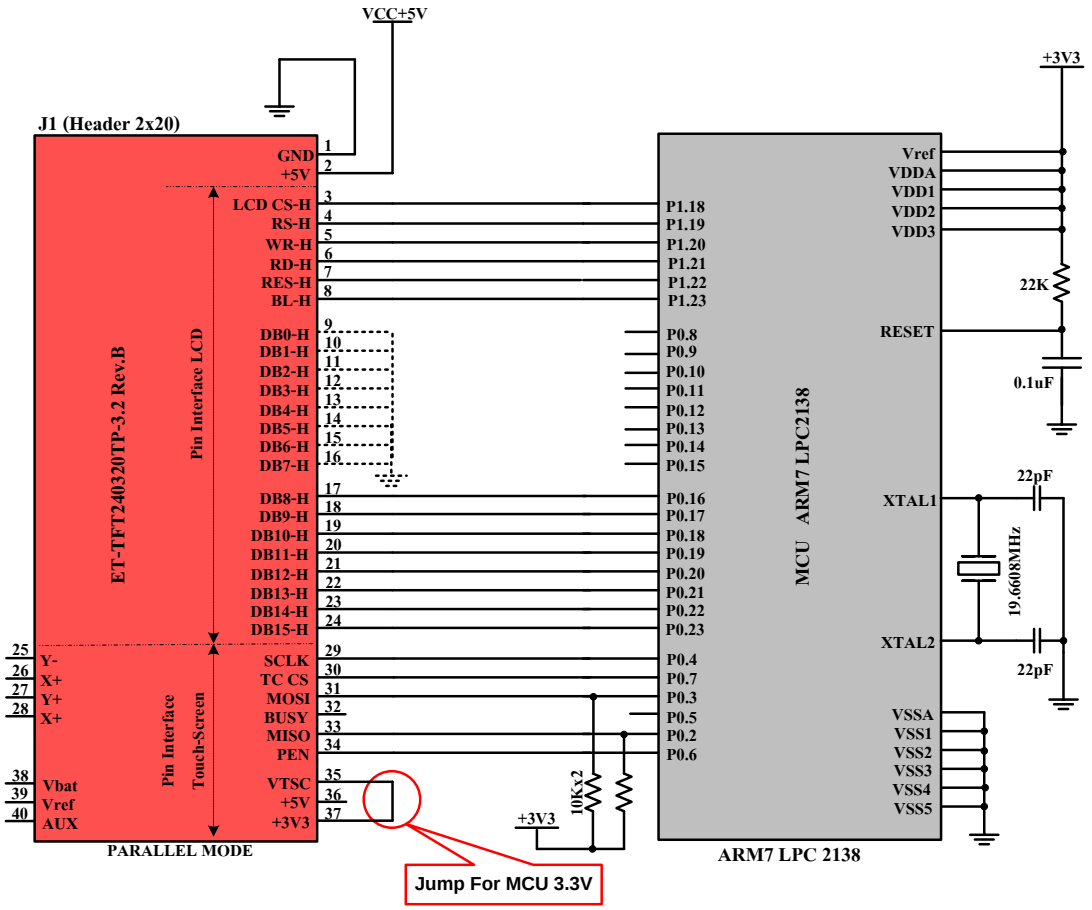


Figure 3.1A example of connecting Board ET-TFT240320TP-2.8 with MCU ARM7 #LPC2138(3.3V)

How to interface with MCU 5V

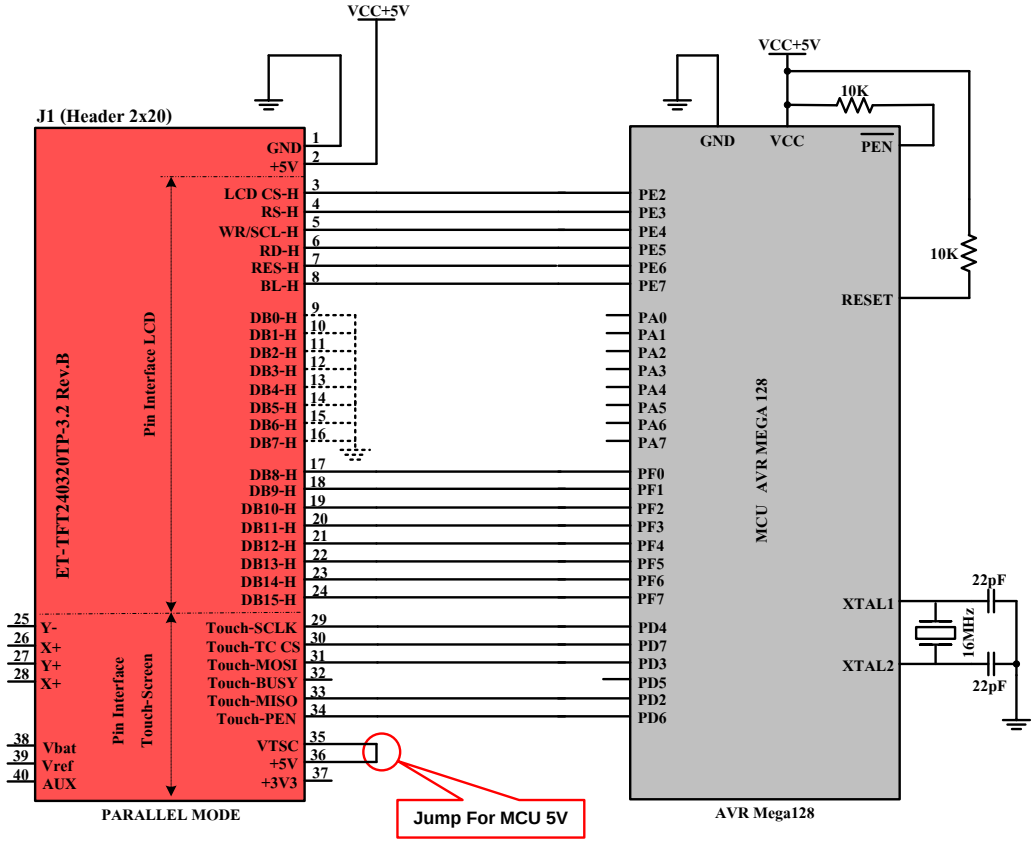


Figure 3.1B example of connecting Board ET-TFT240320TP-2.8 with MCU AVR #Mega128 (5V).

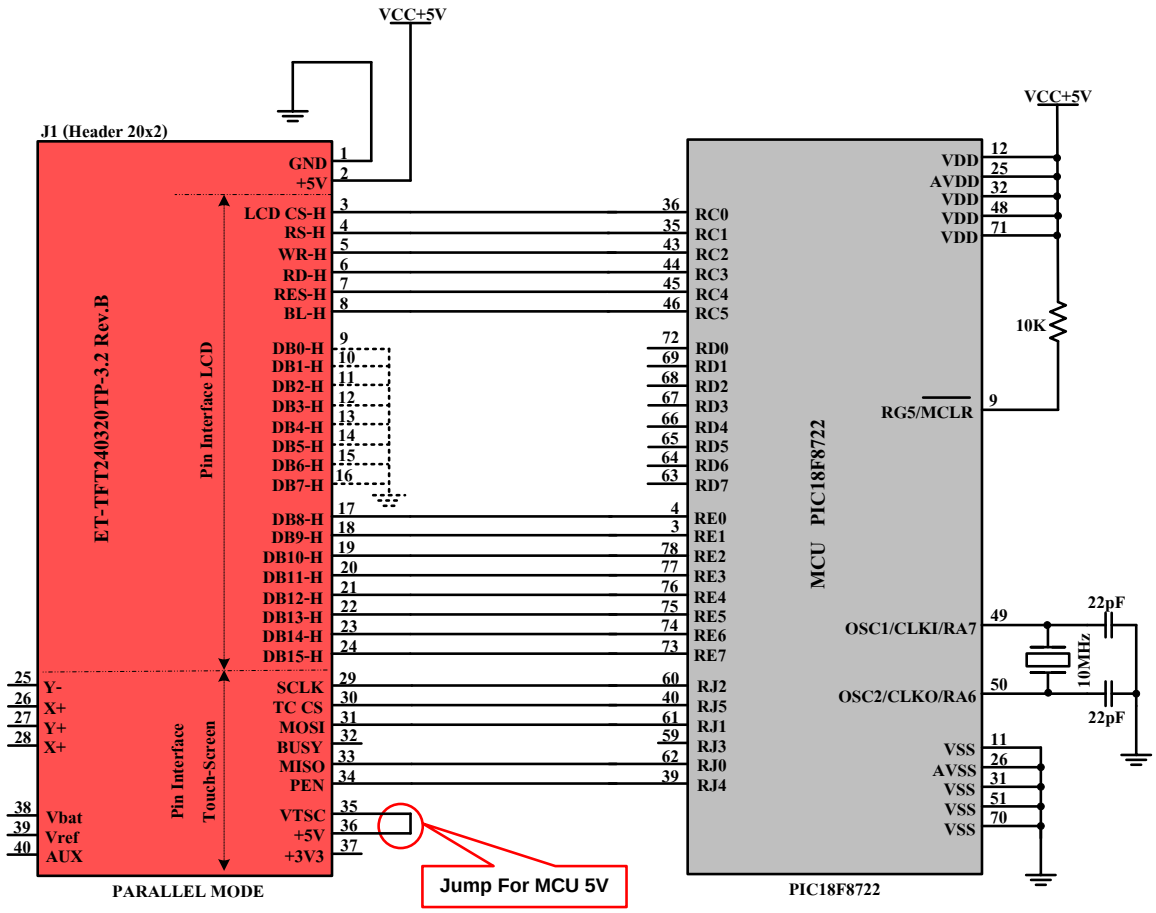
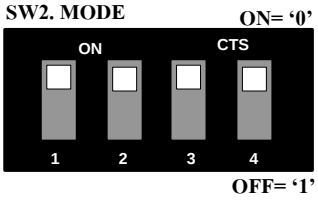


Figure 3.1C example of connecting Board ET-TFT240320TP-2.8 with MCU PIC #18F8722 (5V).

3.2) 16-bit Parallel Interface MODE

How to Set Interface Mode



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

How to interface with MCU 3.3V

Referred to this circuit, it interfaces Pin P0.2, P0.3, P0.11, and P0.14 of MCU with R Pull-Up because this Pin Port is Open Drain. If this MCU that is used has not any Pi Open Drain, it is unnecessary to interface R Pull-Up.

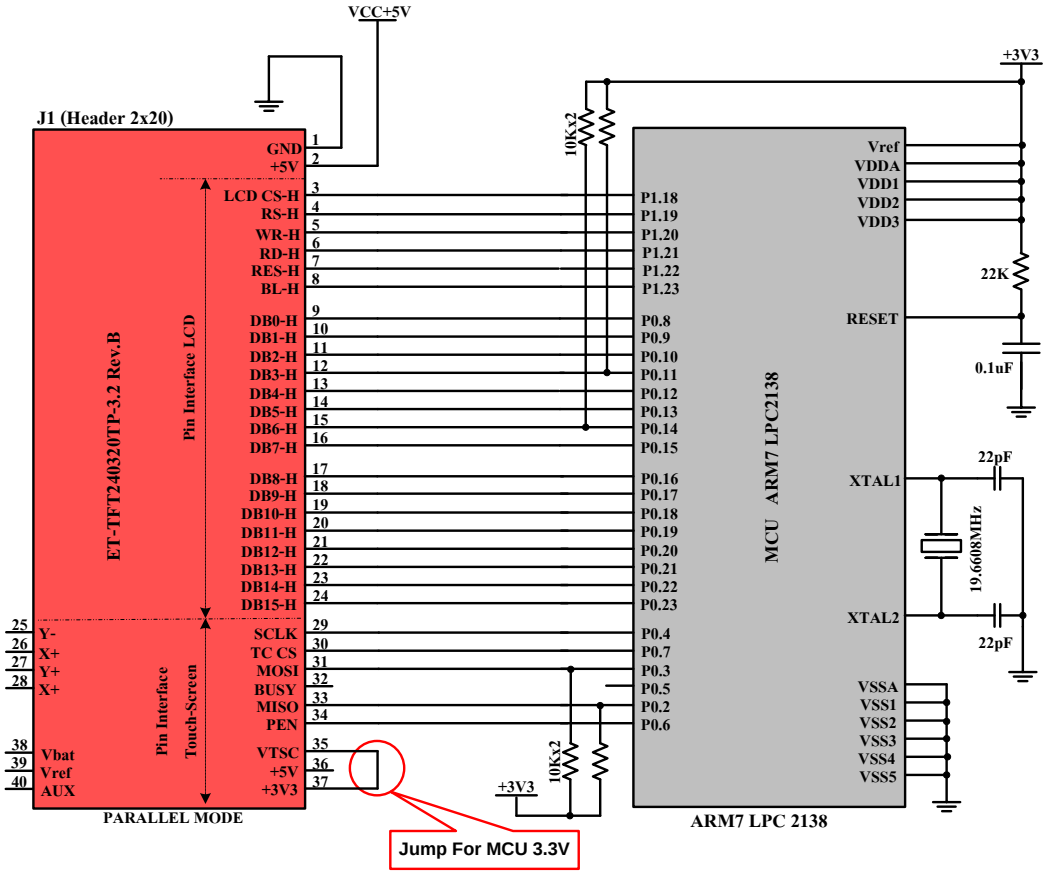


Figure 3.2A example of connecting Board ET-TFT240320TP-2.8 with MCU ARM7 #LPC2138(3.3V)

How to interface with MCU 5V

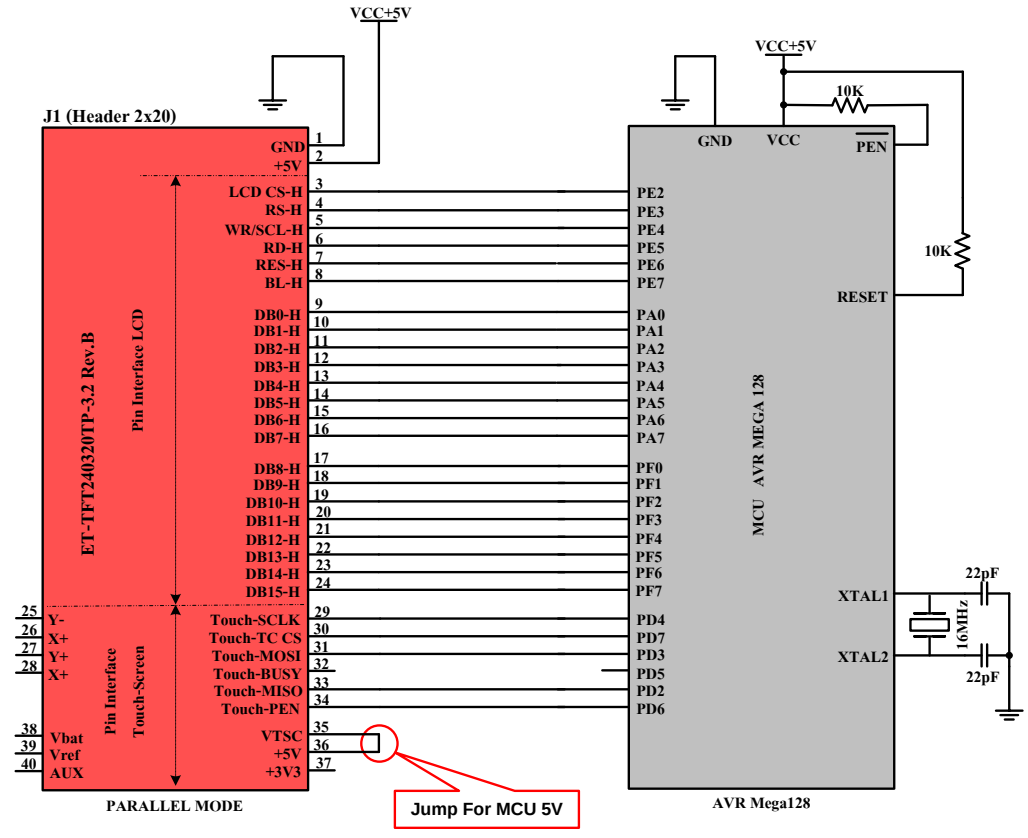


Figure 3.2B example of connecting Board ET-TFT240320TP-2.8 with MCU AVR #Mega128 (5V).

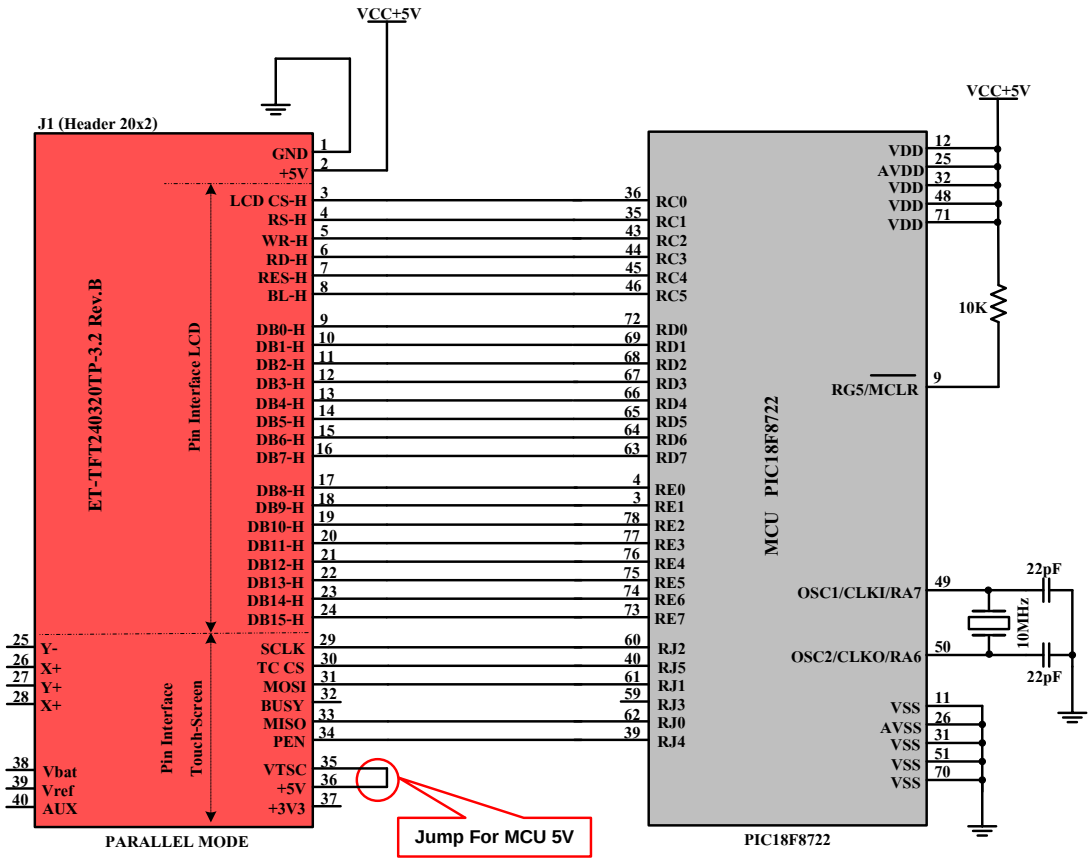


Figure 3.2C example of connecting Board ET-TFT240320TP-2.8 with MCU PIC #18F8722 (5V)

4. Basic Principles to Communicate with LCD and Touch Screen

When user writes program to communicate with Board ET-TFT240320TP-2.8, it is easier if user divides the operation into 2 parts that are LCD Display and Touch Screen. Firstly, there are 2 Interface Modes in the part of LCD Display; Parallel 8-bit and Parallel 16-bit. Secondly, it uses SPI Interface in the part of Touch Screen. Next, we will describe the basic principle of writing program to control the operation; in this case, we will refer to the example program of ETT that has been provided with CD-ROM. Each example program in CD-ROM uses the same basic principles to communicate with board, but it is only different in the part of results that will be shown through the Display. We can summarize the principle of both parts as follows;

4.1) Interface Control LCD: User can read more information about commands for controlling LCD from Data Sheet "HX8347-D.pdf" in CD-ROM. In this case, we divide the principle of sending command or Data to LCD into 3 formats according to Interface Mode and we will describe about it in the next minor section.

First of all, user has to understand that whatever format of Interface Mode always sends 2 sets of Data to LCD. The first set is Register that will be accessed; it is Address Position of Register Index of the 8-Bit Instruction. For example, the Instruction "*Write Data to GRAM(R22h)*" has the Address Position of Register Index of the instruction as "*0x22*"; so, it has to send out the value of "*22H*". The second set that will be sent out is 16-bit Data of the instruction. For example, when user has already sent out the Instruction of "*0x22h*", the

second data set that will be sent out is data of color or other values of the instruction. If user requires displaying a white dot on LCD Display, it has to send out the Data as "*0xFFFF*". When user has understood the basic principles of sending Instructions well; next, we will describe how to arrange data inside LCD, especially how to arrange the incoming Instruction and Data, and including the Timing Diagram of Read-Write Data from MCU to LCD of both Interface Modes as follows;

4.1.1) 8-bit Parallel Interface MODE

LCD in this operation mode arranges data in the part of internal Instruction and Data as shown in the figure 4.1.1A and 4.1.1B. In this case, it sends the 8-Bit(1Byte) Instruction or Data per 1 Signal Write (WR) in each time.

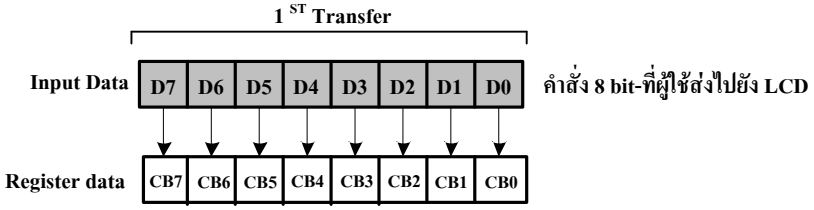


Figure 4.1.1A shows how to arrange bit of the incoming Instruction.

Referred to the figure 4.1.1A above, it only sends out 1-Byte Instruction because the Address Register of the Instruction is only 1-Byte and it is set at the position of Byte Low (D7…D0).

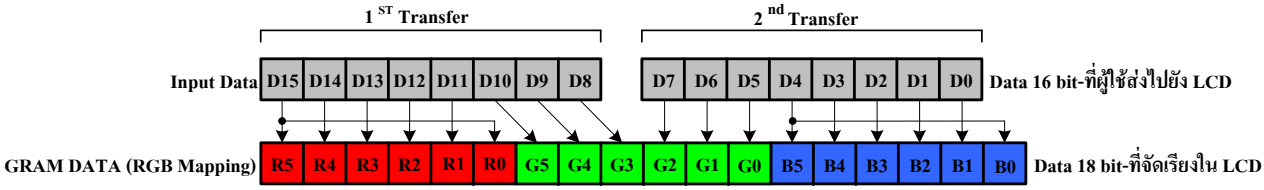


Figure 4.1.1B shows how to arrange the incoming Data Bit that is 16Bit 65K Colors.

Referred to the figure 4.1.1B, it arranges colored Data Bit from Upper Bit to Lower Bit in the format of "RGB", according to the example program of ETT. However, user can modify and change the arrangement of colored Data Bit to the format of "BGR" as required; in this case, it uses the Instruction "*Memory Access Control (R16H)*" to set the new arrangement of colored Data Bit. Read more information from Data Sheet of HX8347-D.

When user requires sending data to show any dot on the LCD Display, user can to mix colors by self by referring to the arrangement of colored Data Bit in the figure above. In this case, Data D15-D11(5Bit) is red; Data D10-D5(6Bit) is green; and Data D4-D0(5Bit) is blue. When user has sent Data into LCD completely, Data will be automatically arranged in the new format of 18Bit Data as shown in the figure. The contrast of colors is arranged from darker to lighter that is from lower bit to upper bit. For example, if user requires the lightest shade of red, it is "*Data = 0xF800*"; or if user requires the darkest shade of green, it is "*Data = 0x0020*". If user requires using other colors that are not these 3 main colors, user has to set the Data Bit in the range of each color

suitably; and finally, user will get the preferable color. For example, if user requires white, it should set the value as "*Data=0xFFFF*"; or if user requires black, it should set the value as "*Data=0x0000*".

Procedures for writing Instruction and Data from MCU to control LCD by 8-Bit Parallel MODE

When user requires sending Instruction or Data from MCU to LCD through this operation mode, it always sends Data Bit to Pin DB8-DB15 of Board LCD of ETT according to the circuit above. First of all, it has to send out 1Byte Instruction or Address Register; next, it has to send out 1Byte or 2Byte Data of the instruction, depending on Data of the instruction. In case of sending 2Byte Data, the first byte should be Byte High and the second Byte should be Byte Low. User should determine the following Timing Diagram for reading-writing Instruction or Data into LCD.

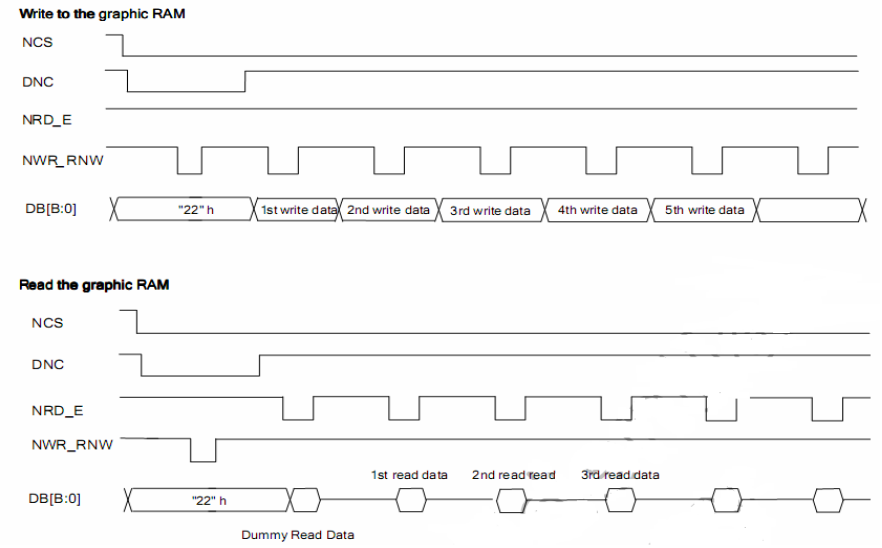


Figure 4.1.1C shows Timing Diagram for Read-Write Instruction and Data to LCD in the format of 8-bit Parallel Mode.

Referred to the Timing diagram above, we only describe the part of Write Data but we do not describe the part of Read Data because the example program does not Read Data from LCD. In this case, it uses Delay instead and it can control the LCD as well.

Referred to the Timing Diagram in the part of Write Data, the first Byte that will be sent out is Instruction or Register Address; in this case, it is "*22H*" that is the Instruction "Write SRAM". Next, it sends out 1Byte or 2Byte Data of the instruction, depending on amount of Data of each instruction. We can summarize the procedures of writing Instruction and Data as follows;

1) Set Pin RD, CS as "**1**".
2) Set Pin CS as "**0**" to enable LCD for receiving data.
3) Send Byte of Instruction or Address Register to Pin DB8-DB15 (referred to the actual pins on board).
4) Set Pin RS as "**0**" to set value for sending out; in this case, it is Instruction.
5) Set Pin PWR as "**0**" to start writing the first Byte Instruction.

6) Set Pin WR as "**1**"; the first Byte Instruction has been sent completely.
7) Set Pin RS as "**1**" to finish writing the Instruction and it sets the next value for sending out that is Data. After sent the Instruction completely, it has to send out the data of the Instruction as follows;
8) Still set Pin CS as "**0**" and Pin RS and RD as "**1**" to send Data.
9) Send out Data Byte High to Pin DB8-DB15 (in case of sending 2Byte Data).
10) Set Pin WR as "**0**" to start Write Data Byte High.
11) Set Pin WR as "**1**"; the Data Byte has been sent completely.
12) Send out Data Byte Low to Pin DB8-DB15.
13) Set Pin WR as "**0**" to start Write Data Byte Low.
14) Set Pin WR as "**1**"; the Data Byte Low has been sent completely.
15) Set Pin CS as "**1**" to finish sending the Instruction and Data. If user requires sending the next Instruction, please repeat the first procedure.

4.1.2 16-bit Parallel Interface MODE

This operation mode sets new arrangement of LCD, especially in the part of Instruction and Data as same as the operation Mode above. It is only different in the part of sending Instruction or Data because it sends 16Bit Instruction or Data (2Byte) per 1 Signal Write (WR) in each time according to Timing Diagram in the figure 4.1.2C.

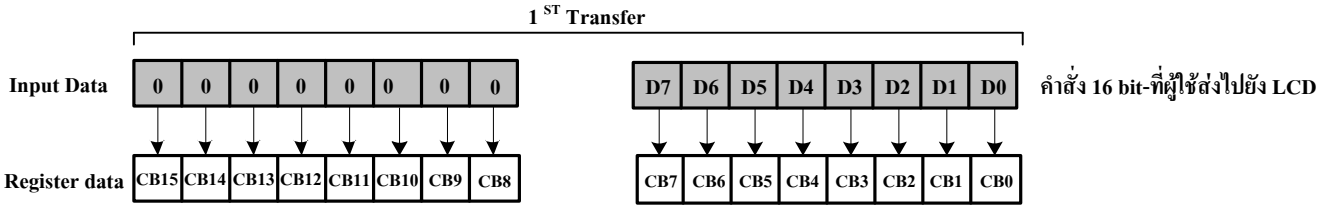


Figure 4.1.2A shows bit arrangement of the incoming Instruction.

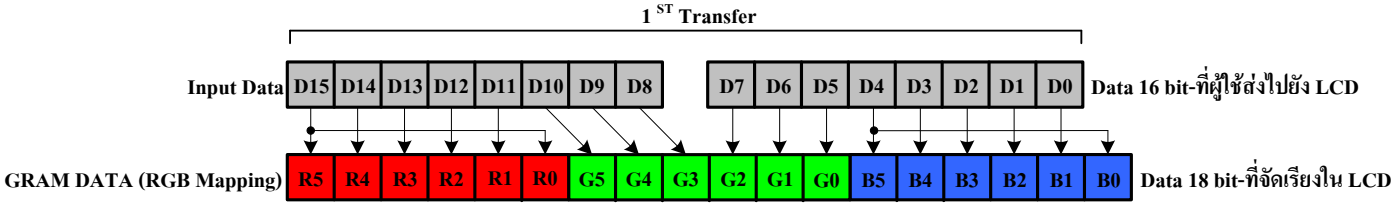


Figure 4.1.2A shows arrangement of 16Bit 65K Colored Data Bit that has been received.

Referred to the figure 4.1.2B, it arranges the colored Data Bit from upper bit to lower bit as "RGB" according to the examples of ETT. However, user can modify and change the arrangement of colored Data Bit as "BGR" as required; in this case, it uses the Instruction "*Memory Access Control (R16H)*" to set the new arrangement of colored Data Bit. Read more information in Data Sheet of SPFD5408A.

When user requires sending data to show any dot on the LCD Display, user can to mix colors by self by referring to the arrangement of colored Data Bit in the figure above. In this case,

Data D15-D11(5Bit) is red; Data D10-D5(6Bit) is green; and Data D4-D0(5Bit) is blue. When user has sent Data into LCD completely, Data will be automatically arranged in the new format of 18Bit Data as shown in the figure. The contrast of colors is arranged from darker to lighter that is from lower bit to upper bit. For example, if user requires the lightest shade of red, it is "*Data = 0xF800*"; or if user requires the darkest shade of green, it is "*Data = 0x0020*". If user requires using other colors that are not these 3 main colors, user has to set the Data Bit in the range of each color suitably; and finally, user will get the preferable color. For example, if user requires white, it should set the value as "*Data=0xFFFF*"; or if user requires black, it should set the value as "*Data=0x0000*".

Procedures for writing Instruction and Data from MCU to control LCD by 16-Bit Parallel MODE

When user requires sending Instruction or Data from MCU to LCD through this operation mode, it always sends Data Bit to Pin DB0-DB15 of Board LCD of ETT according to the circuit above. It sends out 2Byte Data per sending 1 Instruction or Data in each time; in this case, the first 2Byte is Instruction that must be sent out first and then it follows by the second 2Byte that is Data of the instruction. If there is only 1Byte Instruction or Data, in the part of Byte High has to send out "*0x00*" instead. User should determine the following Timing Diagram for reading-writing Instruction or Data into LCD.

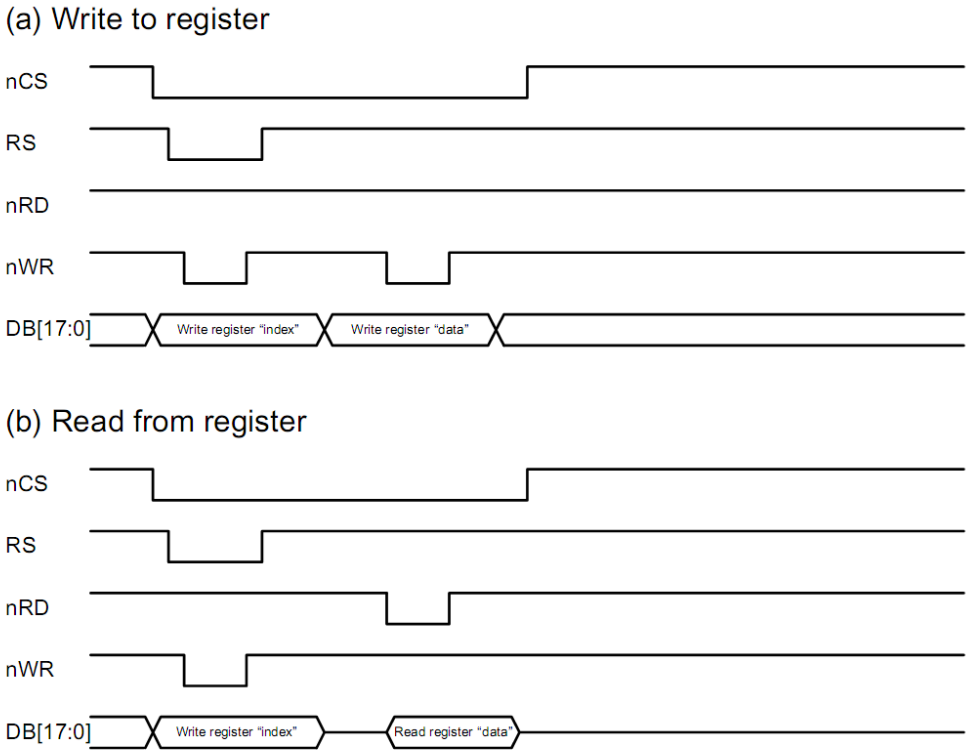


Figure 4.1.2C shows Timing Diagram for Read-Write Instruction and Data from MCU to LCD in the format of 16-Bit Parallel Mode,

Referred to the Timing diagram above, we only describe the part of Write Data but we do not describe the part of Read Data because the example program does not Read Data from LCD. In this case, it uses Delay instead and it can control the LCD as well.

Referred to the Timing Diagram in the part of Write Data, there are 2 sets of data that are sent out to LCD in each time. The first set (2Byte) is the Instruction that is 00H+Address position of Register Index (Register No.); and the second set (2Byte) is 16Bit Data of the instruction. Next, user can send out the instruction according to the Timing Diagram and we can summarize the procedures as follows;

1) Set Pin RD, CS as "**1**".
2) Set Pin CS as "**0**" to enable LCD for receiving data.
3) Send 16Bit Instruction Set to Data Bus (DB0-DB15); in this case, Data Bus 8Bit Upper (DB8-DB15) is set as "**0x00**".
4) Set Pin RS as "**0**" to set value for sending out; in this case, it is Instruction.
5) Set Pin WR as "**0**" to start writing the first 2Byte Instruction.
6) Set Pin WR as "**1**", the first 2Byte Instruction has been sent completely.
7) Set Pin RS as "**1**" to finish writing the Instruction and it sets the next value for sending out that is Data. After sent the Instruction completely, it has to send out the data of the Instruction as follows;
8) Still set Pin CS as "**0**" and Pin RS and RD as "**1**" to send Data.
9) Send out 16Bit Data of the Instruction to all 16Bit Data Bus (DB0-DB15). If it sends out 8Bit Data (1Byte), it has to set 8Bit Data Bus Upper (DB8-DB15) to be "**0x00**".
10) Set Pin WR as "**0**" to start writing the latter 2Byte Data.
11) Set Pin WR as "**1**"; the latter 2Byte Data has been sent completely
12) Set Pin CS as "**1**" to finish sending the Instruction and Data. If user requires sending the next Instruction, please repeat the first procedure.

Referred to the procedures above, user can repeat the first step if user requires sending other instructions. While writing program, user can writes function to receive Instruction and value of Data into the function simultaneously; and then user can send Instruction and Data according to steps above. Moreover, user can write function according to the examples of ETT; in this case, it separates the function into 2 parts; function for sending Instruction and function for sending Data.

4.2) Interface Control Touch Screen: In the part of this Touch Screen, it separates Control from LCD. There are 2 Interface Modes that can be chosen for this Control. Firstly, it interfaces Pin Y-,Y+,X-,X+ with Pin ADC of MCU directly and user has to write program to control the process of reading values by self. However, it is difficult to write program because user has to learn and understand the operating principles of Touch Screen well and user has to write the program correctly. In this case, we do not suggest user to choose this Interface Mode.

The Interface Mode that we will suggest user to choose and use is to interface signal through Chip ADS7846 according to the example of ETT. When user has chosen this Interface Mode, it has to shift all DIP SW(S1-S2) at the back of board to the position "ON" to interface Pin X+,Y+,X-,Y- of Touch Screen with Chip ADS7846 (normally, it has already been set to be default position). The Interface Mode that uses Chip ADS7846 is SPI Interface between MCU and Chip; user can read more information about communicating data with Chip for reading-writing position of Touch Screen correctly from Data Sheet "Touch_ADC7846N.pdf". First of all, we should look at the co-operation between Touch Screen with ADS7846.

- **Co-operation between Touch Screen and ADS7846**: It starts to read position of Touch Screen when user touched the Touch Screen, Chip ADS7846 converts the incoming Signal Analog through Pin X+,Y+,X-,Y-; and then it sends out Digital through Pin Serial Data Out. The ADC value that is read is 12Bit Resolution; so, the value that is read on both X-axis and Y-axis is in the range of 0-4095. While touching the Touch Screen, Pin PENNIRQ of Chip sends out Signal Interrupt Logic "0" for a while; user has to read status of this Signal Interrupt while writing program to check whether it is touching the Touch Screen. It saves much time because it does not loop to read the position of Touch Screen all the time, it only reads the value when it is touching the Touch Screen only; so, program can run in other parts independently.

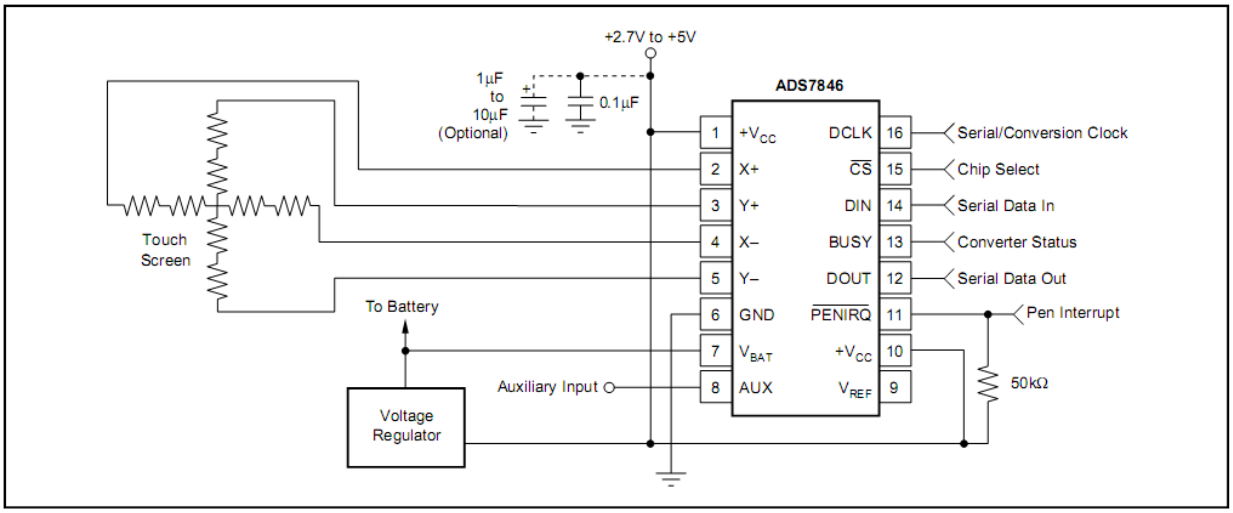


Figure 4.2.1 shows how to interface circuit in the part of Touch Screen and ADS7846.

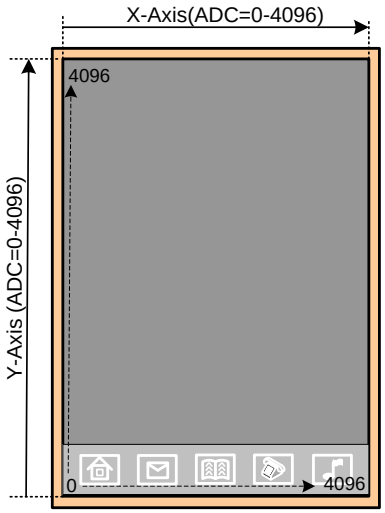


Figure 4.2.2 extent Touch Screen and ADC value that has happened on the X,Y axis

Referred to the figure 4.2.2 above, it shows the direction of X-axis and Y-axis of Touch Screen and the ADC value that is read on positions while touching Touch Screen. Normally, the ADC value that is read through ADS7846 has the minimum value at 600 (not 0); it is the Offset value of the Display. However, the Offset value of each Touch Screen is read unequally; so, user always writes program to calibrate the Touch screen. In this case, user can copy the example of ETT to calibrate the Touch Screen instantly. It uses principle of matrix to calculate the coefficient from calibrating the Touch Screen of user. The example program forces user to touch 3 positions on the Touch Screen; in this case, if user can touch the mark position accurately, it also makes the operation be greater accuracy.

- **Timing Diagram for Read-Write data through ADS7846:** When user understood the basic principles in the part of Touch Screen; next, we will describe how to read ADC Value from Touch Screen by using Chip ADS7846. First of all, user has to understand that ADC Value that is read from the Touch Screen is not the actual Address position; it cannot be used to refer to any position on LCD. User has to calculate this value to find out the actual position of LCD by self. When user has already got the actual Address position of LCD, user can use this value to replace the Instruction for controlling position of LCD Display. User can see and read all procedures from examples of ETT.

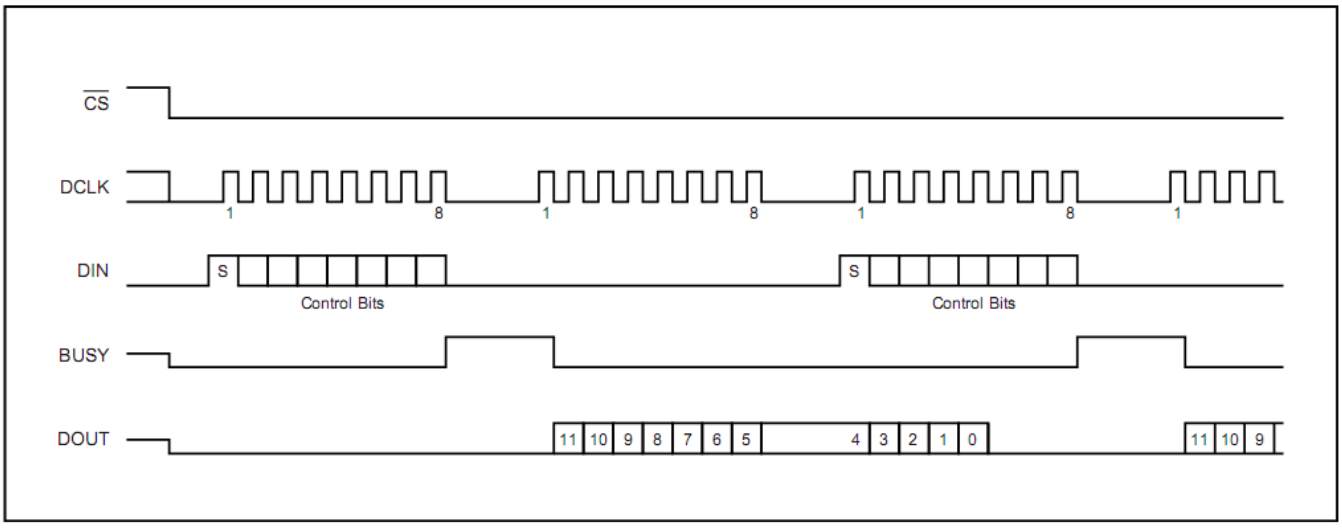


Figure 4.2.3 Conversion Timing Diagram, 16 Clock-per-Conversion, 8bit bus Interface.

This Timing Diagram shows process of reading the ADC value from Touch Screen through Chip ADS7846; it uses SPI Interface between MCU and ADS7846 according to the example of ETT. Referred to the SPI Interface in the example, it uses Pin I/O of MCU, it does not use Module SPI internal MCU; so, user can edit or modify program easily. The principle of sending data in format of SPI Interface is to send out 1-bit data to Pin MOSI(Dout) and then follows by 1 Clock, it shifts 1Bit data into the Chip. Meanwhile, the Chip also shifts 1-bit data out to Pin MISO(Din), it is data that user has to read and save. Referred to the example of ETT, function *tcs_wr()* is used to write and read 1Byte data (8bit) serially. When user has created this function completely, user can send Control Byte and read the ADC value to store through this function. The procedures of reading the ADC value from Touch Screen are listed below.

The method to read the ADC value from ADS7846 is to send Control Byte to ADS7846 first; it sets specifications for Chip before reading the value. The format 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 4.2.4 shows Control Byte of ADS7846.

User can read more details of Control Byte for each Bit from Data Sheet by self. Referred to the Control Byte in the example, it is "*0xD0*" for reading the ADC value on the X-axis of Touch Screen and "*0x90*" for reading the ADC value on the Y-axis of Touch Screen.

Summarize procedures to read ADC from ADS7846

1) Read Status from Pin PEN of ADS7846. If it is "**0**" (touching the Touch Screen), user can go to the next step to start the process of reading value; on the other hand, if it is "**1**" (has not touched the Touch Screen yet), it has to repeat reading.
2) Set Pin DCLK, CS, DOUT as "**0**"
3) Send Control Byte "0xD0" to Pin DIN (MOSI) of ADS7846, it specifies and reads the 12Bit ADC on X-axis.
4) Send Data "0x00" to Pin DIN (MOSI) of ADS7846. Meanwhile, it is sending data in each bit, ADS7846 also shifts the ADC value to Pin DOUT(MISO). The firsts Data Bit that is shifted out is the 11^{th} Bit and it begins at Falling Edge Pin of the 2^{nd} DCLK. When all of 8 DCLK has already been sent completely, the data that is read on X-axis is "*0x0ddddddd (d=data bit11-bit5)*".
5) Send Control Byte "0x90" to Pin DIN (MOSI) of ADS7846, it specifies and reads the 12Bit ADC on Y-axis. Meanwhile, it is sending out this Control Byte, the last 5Bit of Data ADC on X-axis will be sent out; in this case, it begins from Bit4 to Bit0. It arranges Data in the format of "*0xddddd000 (d=data bit4-bit0)*".
6) Send Data "0x00" to Pin DIN (MOSI) of ADS7846. Meanwhile, it is sending data in each bit, ADS7846 also shifts thee ADC value to Pin DOUT(MISO). The firsts bit that is shifted out is the 11^{th} Bit and it begins at the Falling Edge Pin of the 2^{nd} DCLK. When all of 8 DCLK has been sent completely, the Data that is read on Y-axis is "*0x0ddddddd (d=data bit11-bit5)*".
7) Send Data "0x00" to Pin DIN (MOSI) of ADS7846. Meanwhile, it is sending out this Data, the last 5bit of data ADC on Y-axis will be sent out; in this case, it begins from bit4 to bit0. It arranges data in the format of "*0xddddd000 (d=data bit4-bit0)*".
8) When it has read the ADC value on both axes completely, it has to set Pin CS as "**1**" to finish reading the value from DAS7846.
9) If user requires reading other value, please repeat the step 1.
10) When it has read the ADC value on each axis completely, user has to rearrange the value. It uses 16Bit Variable for storing the ADC value that is read; the first 7Bit Data that is stored in the 16Bit Variable is "*0x000000000ddddddd*", while the latter 5Bit Data that is stored in the 16Bit Variable is "*0x00000000ddddd000*". Next, it shifts the Data that is read; in this case, it shifts 5Bit of the first 7Bit to the left side for and it shifts 3Bit of the latter 5Bit data the right side. Next, it has to do OR(|) data of both sets and user will get the 12Bit ADC value of the axis for using; in this case, it is "*0x000ddddddddddddd*".

Referred to all principles of Control LCD and Touch Screen above, it is the general operation of using Board ET-TFT240320TP-2.8. When user understood the basic principles of Control well, user can copy functions in the examples of ETT to paste and use in the program's user instantly. Read more information about how to copy and use the functions from "Description of Example Program".

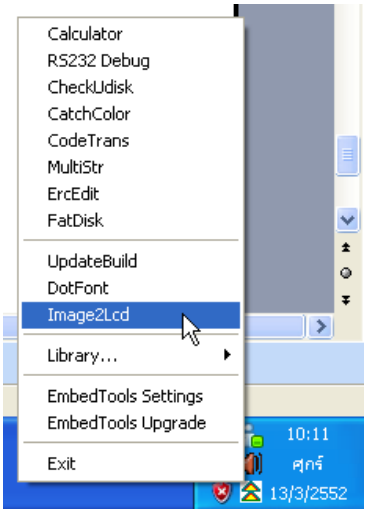
5. How to use Program Embedtools 3.31 to Convert image file into HEX Code

Now, we describe how to convert image file into HEX Code that will be send and display on the LCD Display as required; in this case, it uses Program "**Embedtools 3.31**". The process of converting this HEX Code supports the example from ETT; it uses function "*plot_picture()*" (it is in the example "*Ex3_Touch_Button*") to send the HEX Code from MCU to LCD Display.

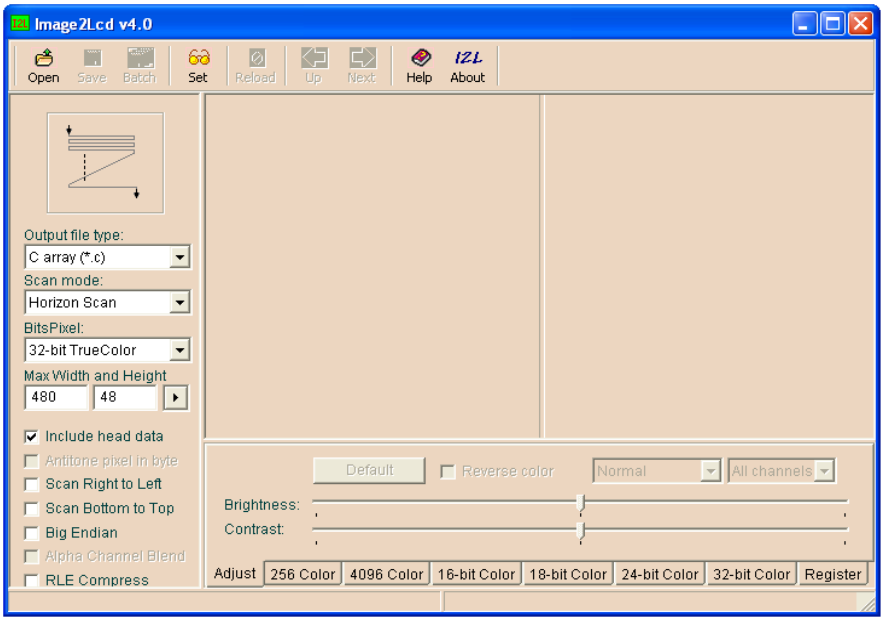
If user does not do *initial lcd* according to the example of ETT and does not set Program Embedtools to convert image file as mentioned below, user cannot use function "*plot_picture()*" because the direction of sending and receiving data incorrectly. In this case, user has to write program to plot image file by self according to the values that user has already set.

Procedures to use Program Embedtools to convert image file into HEX Code

1. Install Program *Embedtools.exe* into the computer PC (it is in the Folder Embedtools3.31).
2. After user has installed the program successfully, it appears the ICON( ) at the Taskbar as shown in the picture 5.1. Next, click right on the ICON, TAB appears; click left on *Image2LCD* to run Program Embedtools; and finally, it shows a window as shown in the picture 5.2.

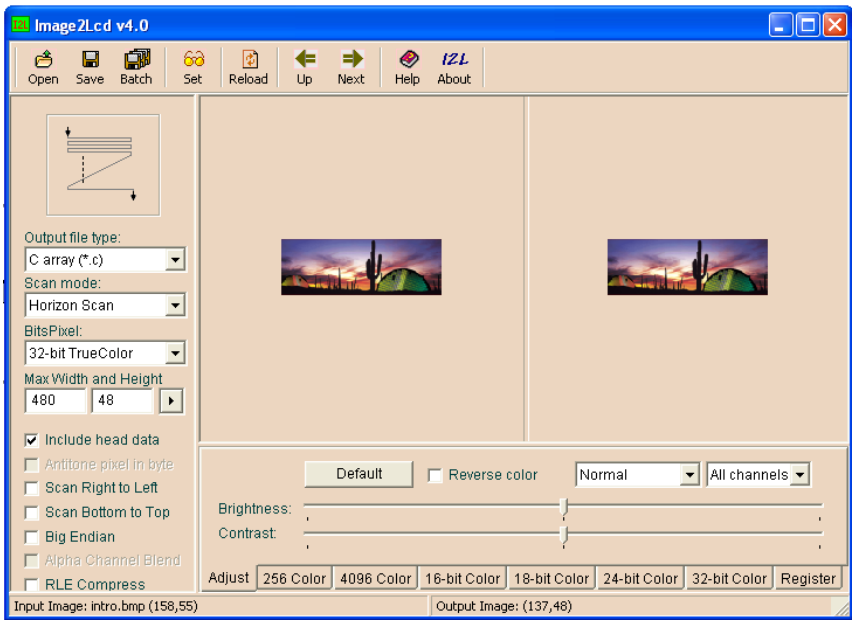


Picture 5.1



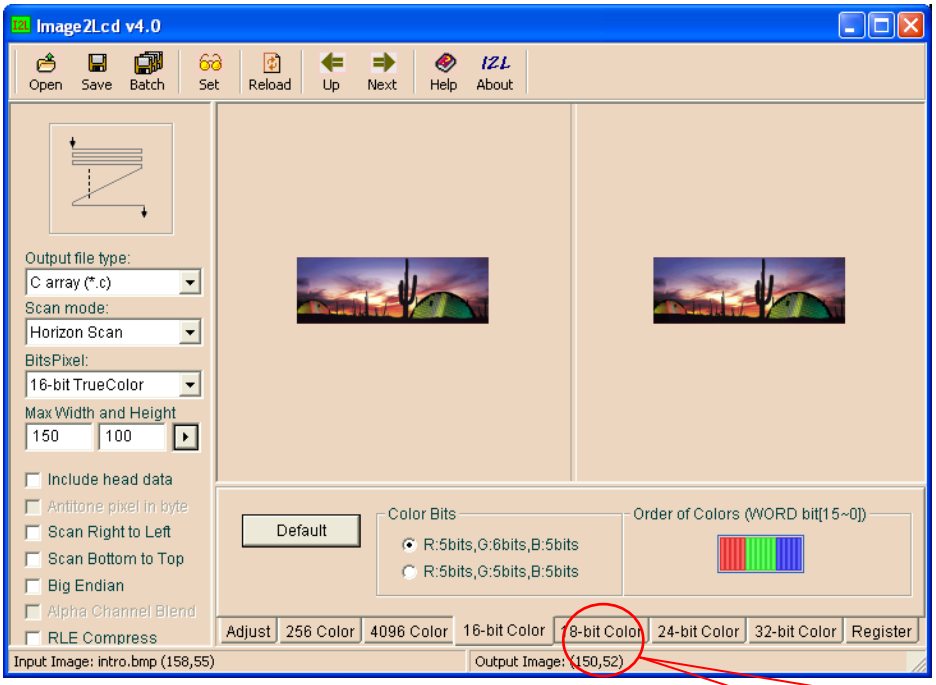
Picture 5.2

3. Click ICON OPEN(Open) on top of program to choose the preferable photo to convert; in this case, the image file should be the file surname as *.jpg* or *.bmp*. When user has opened the preferable photo, it shows window as shown in the picture 5.3.



Picture 5.3

4. After loaded the photo into the program successfully, it has to set following values (please look at picture 5.4 to be your guideline).



Picture 5.4

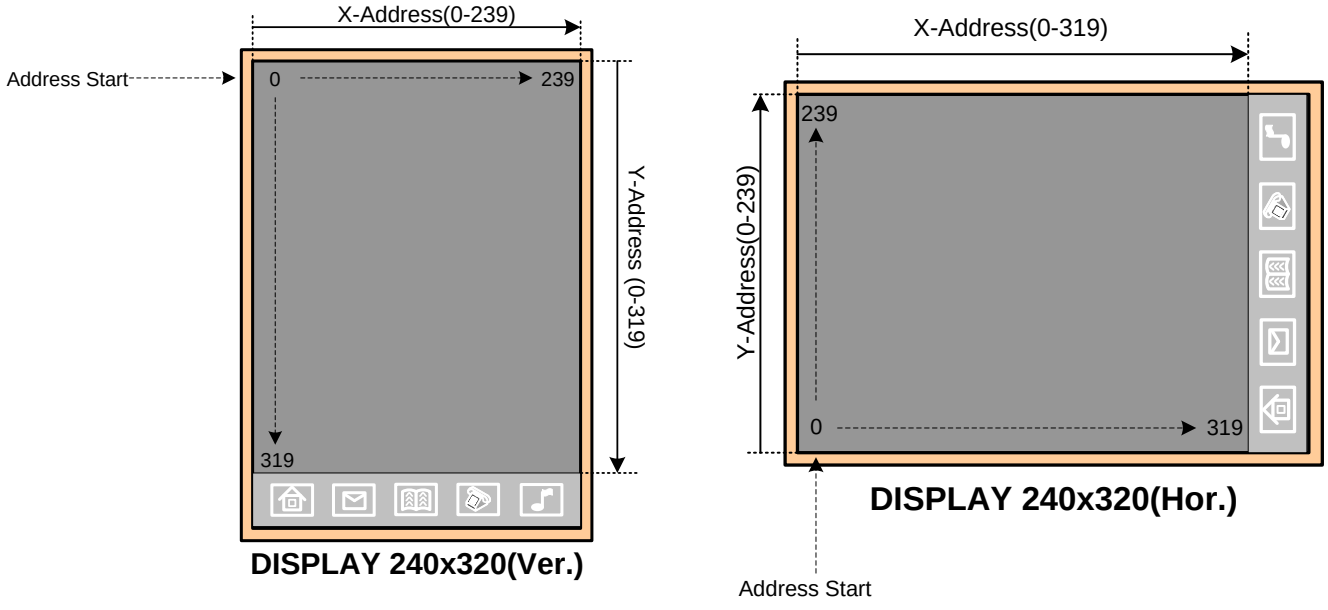
- **Output file type:** ***C array (*.C)*** = It sets data in the format of Array of C Language and then saves File Output as *.C*.
- **Scand mode:** ***Horizon Scan*** = It sets initial direction of scanning data. When using data, it has to send data to LCD according to the direction of scanning data.

- **Bits Pixel:** ***16-bit TrueColor*** = It sets resolution of colored Bit (there are 2 the same values in the program, please choose the upper value).
- **Max Width and Height: Width, Height** = Set width and height for the preferable picture. Click Button( ) after setting the values completely, user can see the change of picture, especially the picture size is in the unit of Pixel. However, it is not the actual picture size. If user calls and uses the function *"plot_picture()"* from example of ETT, it requires the actual width and height of the picture; in this case, user can look the actual Pixel of picture in the box "Output Image" below.
- **Include head data** = Remove the check sign from the box.
- **TAB 16-bit Color** = Click TAB 16-bit color in the box of "Color Bit", it should choose the value in the box of "R:5bit,G:6bit,B:5bit"; and in the blank of "Order of Colors (WORD bit[15~0]", it arranges colors in the format of "RGB". For other Tabs, user does not set any value.

5. When user has already set all values, click ICON Save ( ) to save file hex code. If user requires using this file, it uses Save te pad to open this file; next, user can copy the hex code and then paste in the editor for writing program instantly.

6. How to run Example Programs

Example Programs of ETT are written by C Language, it can support 3 MCU Series; AVR(Mega128), PIC(18F8722), and ARM7(LPC2138). There are 2 formats of Control LCD that can be found in this example; vertical line and horizontal line. It depends on user to choose the preferable format to display data and then user looks at the corresponding the example program mainly. Examples of each MCU series have the same format; all examples refer to the initial Address position on X,Y axis of the screen as shown in the picture below;



Picture 6.1 reference to the initial Address on LCD Display in vertical line and horizontal line.

Referred to the pictures above, if using function in the example program of ETT and setting any Address position of X,Y to plot picture or character, user should look at the reference address in the figure above.

There are 4 main example programs in both formats of vertical line and horizontal line as described below;

- ***Ex1_Touch_Position***: Referred to this example program, when it is run in the first time use, it forces user to touch 3 plus signs (+) accurately to calibrate the Touch Screen. After calibrated, MCU can read Address position correctly according to the actual Address position of LCD every time user touches the Screen as shown in the picture 16. After calibrated, it always shows the position of X,Y that has been touch at the bottom every time user touched any position on LCD Display.
- ***Ex2_Touch_Draw***: Referred to this example program, when it is run in the first time use, it forces user to calibrate Touch Screen as same as the first example program. Next, user can draw or write any character on the screen and it will display lines according to writing or drawing. Moreover, user can touch Icon Musical Note( ) to change color of line and user can touch Icon HOME ( ) to Clear Screen.
- ***Ex3_Touch_Button***: Referred to this example program, when it is run in the first time use, it forces user to calibrate Touch Screen as same as the first example program; next, it shows buttons. When user has touched any button on the screen, the blank window will show pictures according to the pressing button.

Referred to the example in the part of calibrate touch Screen, every time when user has RESET MCU, user has to calibrate the screen as well. However, user can calibrate the screen only one time in case of actual use. The method to edit this function is described below;

The 1st method: It has to interface more E2Prompt with MCU (if the MCU has no any E2PROMPT inside). Next, user should write program in Function "*touch_calibrate()*" after the line of calling Function "*set_matrix()*" by writing the value in the Variable *divider,An,Bn,Cn,Dn,En,Fn* to store in E2Prompt, and then user should write any more 1Byte Data to store in E2Prompt to be Flag Status. This Flag Status is used to check whether it has been calibrated completely.

Next, in the part of main program before calling Function "*touch_calibrate()*", user should read Status Flag from E2Prompt to check whether it is the same value as user has written and stored. If yes, it is unnecessary to call Function "*touch_calibrate()*" anymore; user can read the value of the Variable *divider,An,Bn,Cn,Dn,En,Fn* that has been stored in E2Prompt, and replace the value in the Variable *divider,An,Bn,Cn,Dn,En,Fn*; and finally, it can run in other parts of program. This method saves much time because user does not calibrate LCD Display every time using it.

The 2nd method: It does not use E2Prompt for his method. User should add more Instruction *printf()* into the Function "*touch_calibrate()*" after the line of calling Function "*set_matrix()*" (referred to the example, it has written this Instruction but it is disabled). It prints the value of Variable *divider,An,Bn,Cn,Dn,En,Fn* to show the result through RS232; in this case, it uses Program Hyper Terminal to receive the value of the

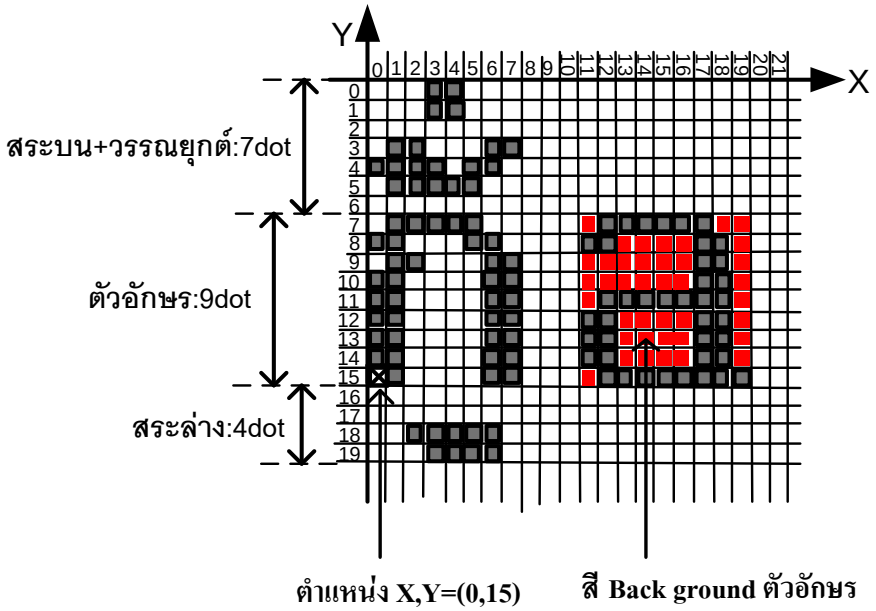
Variable that has been printed to display the result for user. Next, user should note down and remember the value of each variable because it will be used to set value for the Variable *divider,An,Bn,Cn,Dn,En,Fn* that has been declared above *main()*. While using, user can remove this Function "*touch_calibrate()*" and does not call it any more. In case of using this method with many LCD Displays, user has to write program to find out the value of *divider,An,Bn,Cn,Dn,En,Fn* to calibrate the display separately; because user has to find out the new value to calibrate the LCD Display every time user changes the new LCD Display although using the same LCD Display type. Remember, user cannot use the value of the old LCD Display to read the new LCD Display because the address position that is read does not accord with the actual address position of the new LCD Display.

NOTE: While calibrating, user has to touch the nearest marked position because it makes the operation more accurate.

- ***Ex4_Font_TH_EN***: Referred to this example program, when it is run in the first time use, it forces user to calibrate Touch Screen as same as the first example program; next, it shows text in both Thai and English. Then, it shows Button "Test Position" and Button "Example TH"; when user has touched any button on the screen, the blank box at the bottom will show text in both Thai and English.

It uses Function *printStr_TE(char *str,int cur_x,int cur_y,unsigned int fg_color,unsigned int bg_color)* to plot Font in both Thai and English. In this case,

**str* = Text in both Thai and English.
X = Position of pixel on X-axis that user requires displaying the character on the LCD Display.
Y = Position of pixel on Y-axis that user requires displaying the character on the LCD Display.
fg_color = Color of character or color of pixel that is plotted to be character.
bg_color = Color of background of character or color of pixel in the part that is not plotted (0x0001 = colorless)



Picture 6.2 shows how to place position of the character that is plotted on LCD Display.

Referred to the picture 6.2 above, the position of X, Y that has been replaced in the function is the initial position to display the preferable character on the Display. In this case, it refers to the bottom on the left corner of the character mainly; and the upper-lower vowel and a tone mark in Thai writing are added automatically.

It uses the general principle to type Thai character into the function above to display result on the Display. For example,

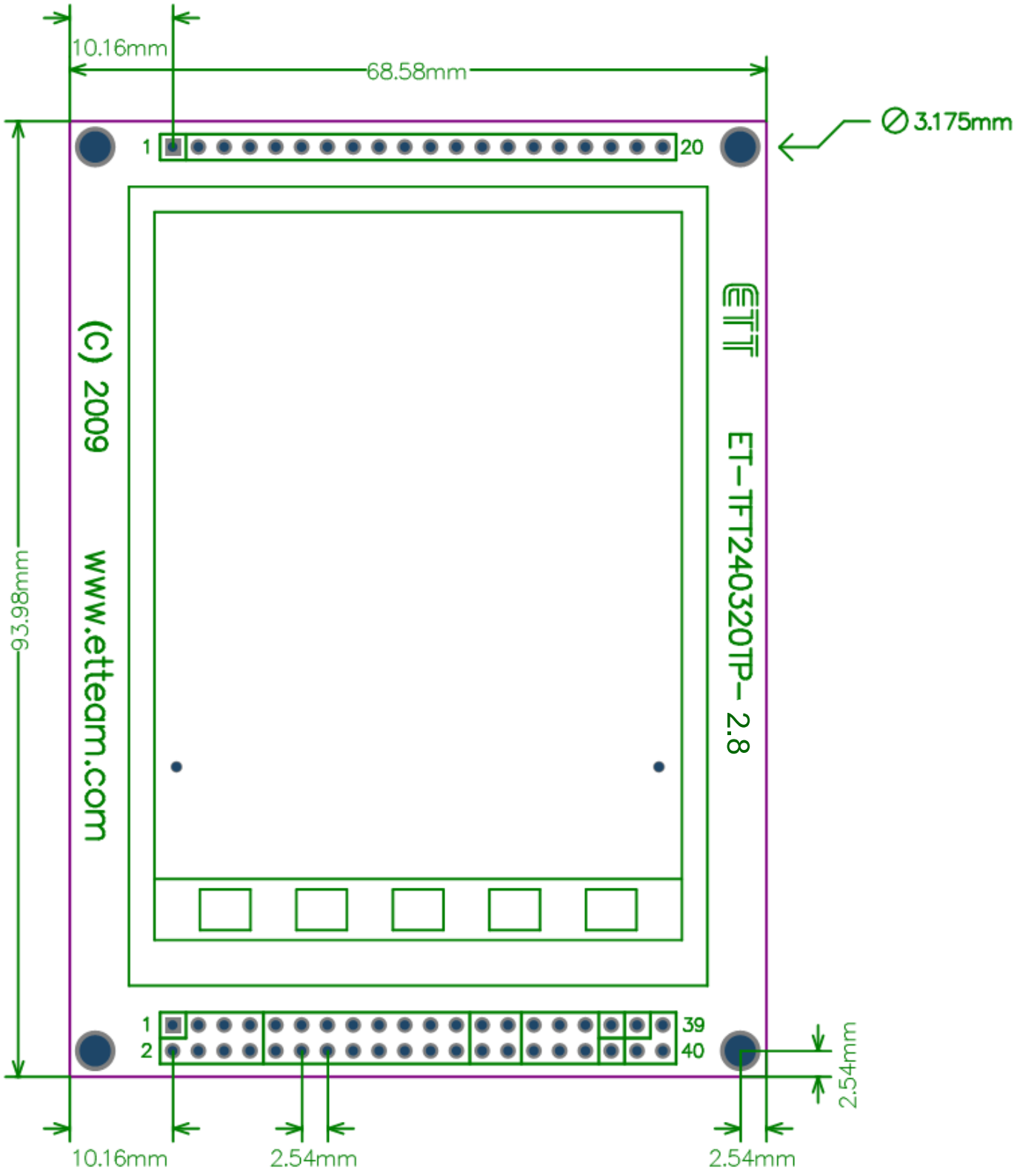
"น้ำ" = น- ◌ำ- ◌้ "เสื้อ" = เ-ส-◌ื- ◌้-อ "รั้ง" = ร- ◌ั- ◌้-ง
"สบู่" = ส-บ-◌ู- ◌่ , "ซื้อสินค้า" = ซ-◌ื- ◌้-อ-ส-◌ิ-น-ค- ◌้- า

In the part of ASCII CODE that is used to refer to Thai and English characters in the example program, user can refer to the Table of ASCI CODE below.

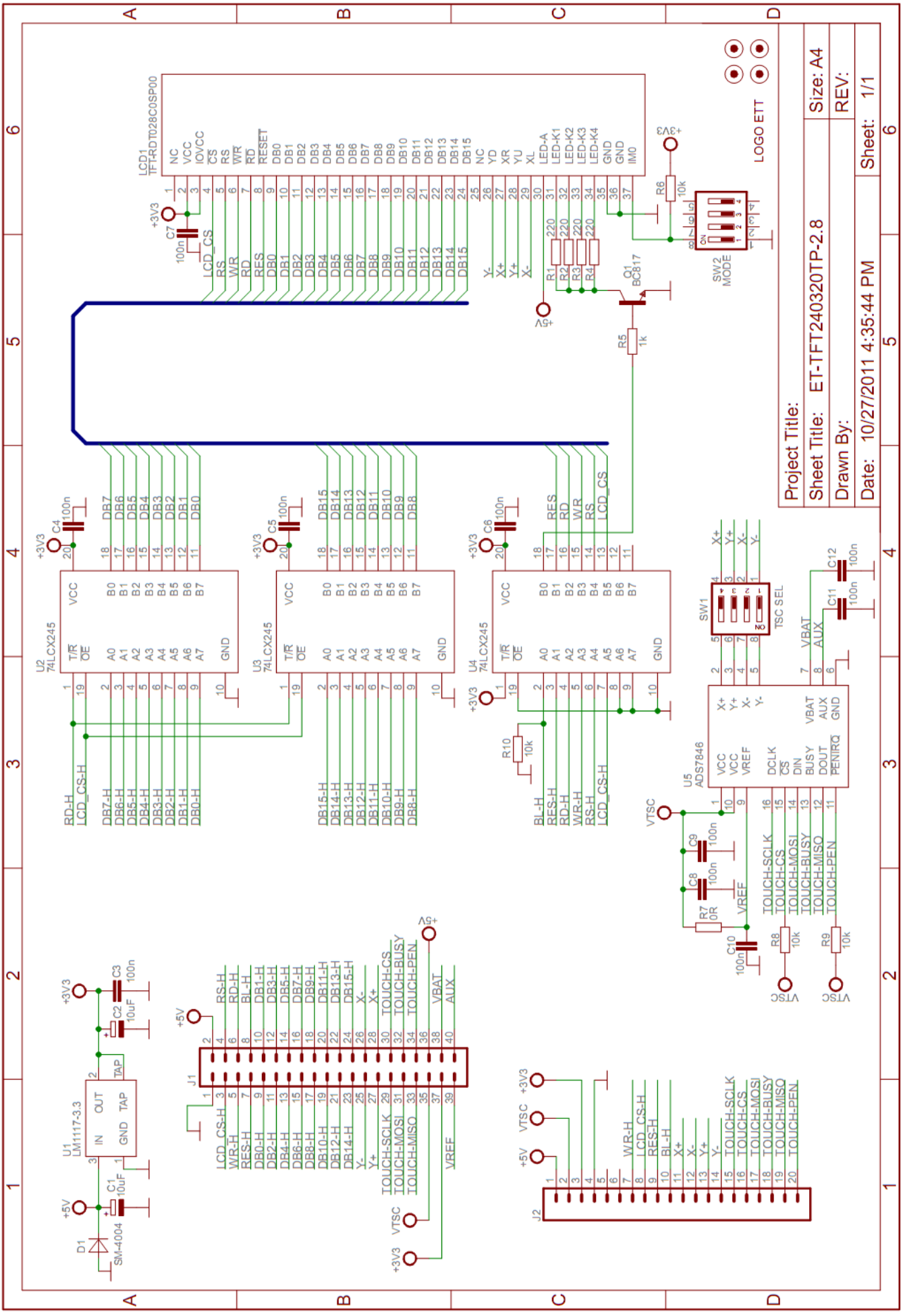
Table Font ASCII Code สมอ.

Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char
0	0	×	44	2C	,	88	58	X	132	84	×	176	B0	ฐ	220	DC	▄
1	1	×	45	2D	-	89	59	Y	133	85	...	177	B1	ฑ	221	DD	▼
2	2	×	46	2E	.	90	5A	Z	134	86	×	178	B2	ฒ	222	DE	▲
3	3	×	47	2F	/	91	5B	[	135	87	×	179	B3	ณ	223	DF	฿
4	4	×	48	30	0	92	5C	\	136	88	×	180	B4	ด	224	E0	เ
5	5	×	49	31	1	93	5D	]	137	89	×	181	B5	ต	225	E1	แ
6	6	×	50	32	2	94	5E	^	138	8A	×	182	B6	ถ	226	E2	โ
7	7	×	51	33	3	95	5F	_	139	8B	×	183	B7	ท	227	E3	ใ
8	8	×	52	34	4	96	60	`	140	8C	×	184	B8	ธ	228	E4	ไ
9	9	×	53	35	5	97	61	a	141	8D	×	185	B9	น	229	E5	×
10	0A	×	54	36	6	98	62	b	142	8E	×	186	BA	บ	230	E6	ๆ
11	0B	×	55	37	7	99	63	c	143	8F	×	187	BB	ป	231	E7	็
12	0C	×	56	38	8	100	64	d	144	90	×	188	BC	ผ	232	E8	่
13	0D	×	57	39	9	101	65	e	145	91	'	189	BD	ฝ	233	E9	้
14	0E	×	58	3A	:	102	66	f	146	92	'	190	BE	พ	234	EA	๊
15	0F	×	59	3B	;	103	67	g	147	93	"	191	BF	ฟ	235	EB	๋
16	10	×	60	3C	<	104	68	h	148	94	"	192	C0	ภ	236	EC	์
17	11	◀	61	3D	=	105	69	i	149	95	•	193	C1	ม	237	ED	๐
18	12	×	62	3E	>	106	6A	j	150	96	-	194	C2	ย	238	EE	×
19	13	×	63	3F	?	107	6B	k	151	97	—	195	C3	ร	239	EF	๏
20	14	×	64	40	@	108	6C	l	152	98	×	196	C4	ฤ	240	F0	๐
21	15	×	65	41	A	109	6D	m	153	99	×	197	C5	ล	241	F1	๑
22	16	×	66	42	B	110	6E	n	154	9A	×	198	C6	ฦ	242	F2	๒
23	17	×	67	43	C	111	6F	o	155	9B	×	199	C7	ว	243	F3	๓
24	18	×	68	44	D	112	70	p	156	9C	×	200	C8	ศ	244	F4	๔
25	19	×	69	45	E	113	71	q	157	9D	×	201	C9	ษ	245	F5	๕
26	1A	×	70	46	F	114	72	r	158	9E	×	202	CA	ส	246	F6	๖
27	1B	×	71	47	G	115	73	s	159	9F	×	203	CB	ห	247	F7	๗
28	1C	×	72	48	H	116	74	t	160	A0	×	204	CC	ฬ	248	F8	๘
29	1D	×	73	49	I	117	75	u	161	A1	ก	205	CD	อ	249	F9	๙
30	1E	×	74	4A	J	118	76	v	162	A2	ข	206	CE	ฮ	250	FA	๚
31	1F	×	75	4B	K	119	77	w	163	A3	ฃ	207	CF	ฯ	251	FB	๛
32	20		76	4C	L	120	78	x	164	A4	ค	208	D0	ะ	252	FC	×
33	21	!	77	4D	M	121	79	y	165	A5	ฅ	209	D1	ั	253	FD	×
34	22	"	78	4E	N	122	7A	z	166	A6	ฆ	210	D2	า	254	FE	×
35	23	#	79	4F	O	123	7B	{	167	A7	ง	211	D3	ำ	255	FF	×
36	24	$	80	50	P	124	7C	|	168	A8	จ	212	D4	ิ			
37	25	%	81	51	Q	125	7D	}	169	A9	ฉ	213	D5	ี			
38	26	&	82	52	R	126	7E	~	170	AA	ช	214	D6	ึ			
39	27	'	83	53	S	127	7F	▮	171	AB	ซ	215	D7	ื			
40	28	(	84	54	T	128	80	×	172	AC	ฌ	216	D8	ุ			
41	29	)	85	55	U	129	81	×	173	AD	ญ	217	D9	ู			
42	2A	*	86	56	V	130	82	×	174	AE	ฎ	218	DA	ฺ			
43	2B	+	87	57	W	131	83	×	175	AF	ฏ	219	DB	▶			

* × : ไม่มีตัวอักษร



Picture 6.3 shows the size of Board ET-TFT240320TP-2.8.



Picture 6.4 shows Circuit of ET-TFT240320TP-2.8.