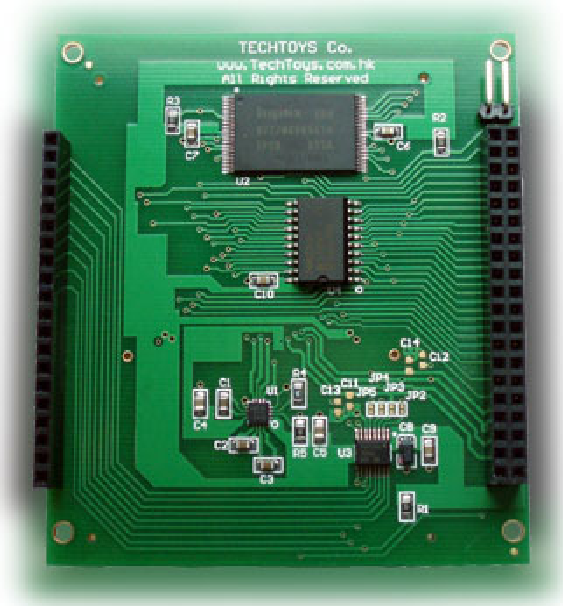
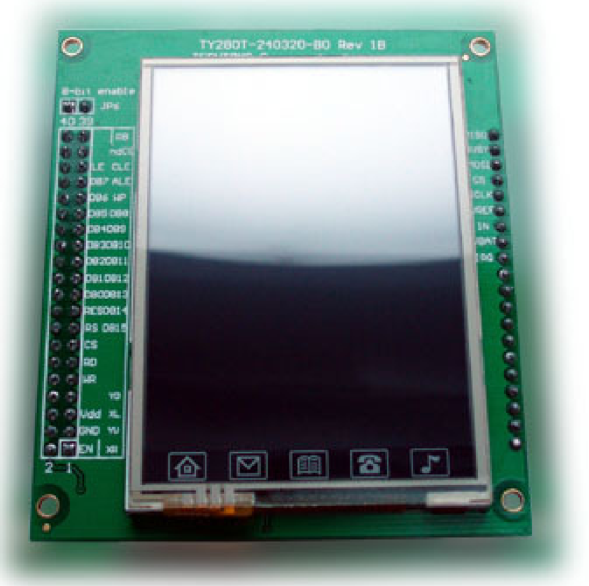
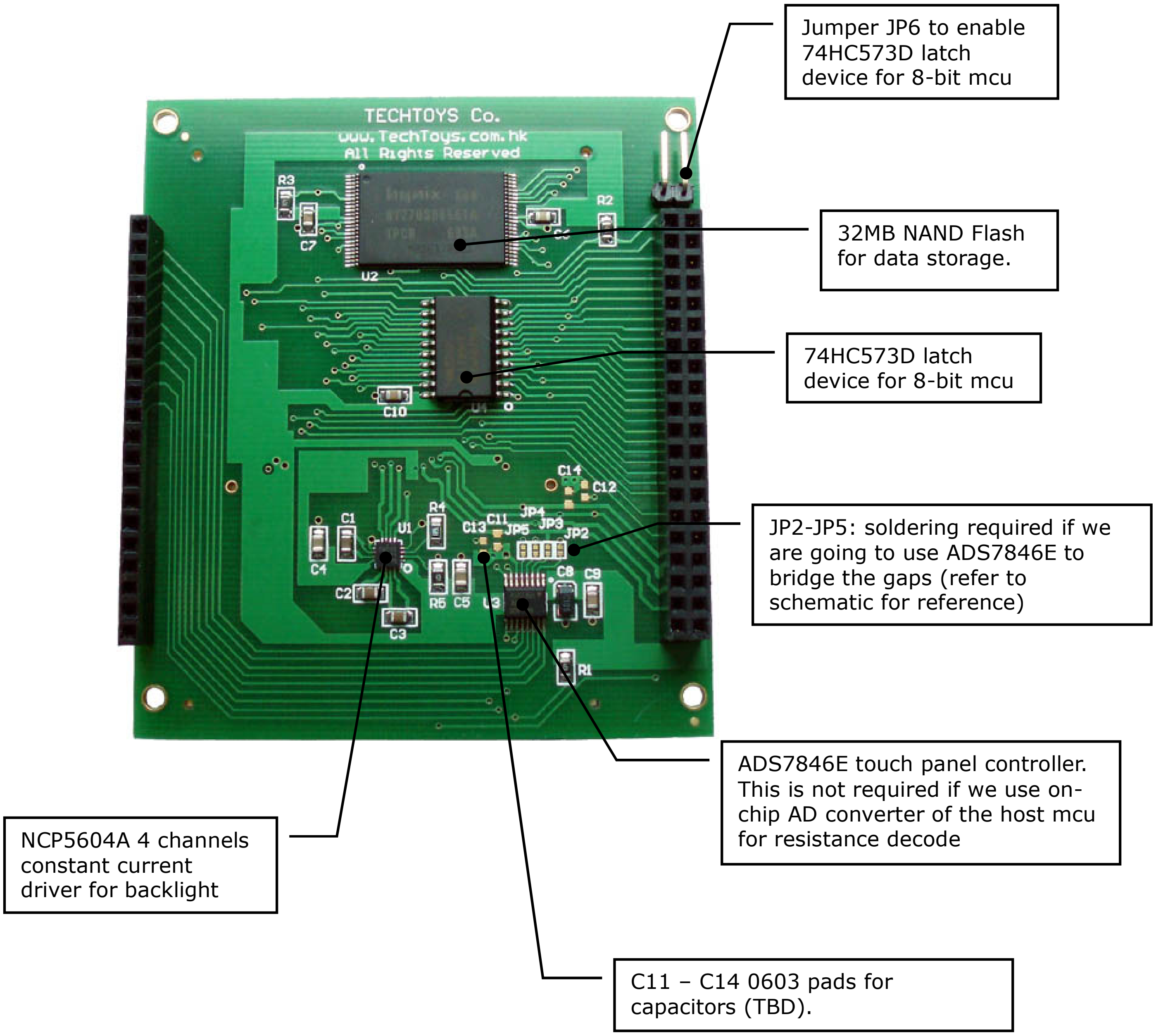
2.8” QVGA 262k TFT LCD module with Touch Panel integrated





INTRODUCTION

The part number TY280T_230320_BO (Board Rev 1B) is a development board for 2.8” QVGA TFT-LCD module completed with a 32MBx8bit NAND Flash, 74HC573D latch device, white LED backlight driver circuit, and a 4-wire touch screen controller all soldered onboard. Standardized 2.54mm PCB sockets have been included for sack of easy prototype procedure. Below please find an illustration of all components in details.



Jumper JP6 to enable 74HC573D latch device for 8-bit mcu

32MB NAND Flash for data storage.

74HC573D latch device for 8-bit mcu

JP2-JP5: soldering required if we are going to use ADS7846E to bridge the gaps (refer to schematic for reference)

ADS7846E touch panel controller. This is not required if we use on-chip AD converter of the host mcu for resistance decode

NCP5604A 4 channels constant current driver for backlight

C11 – C14 0603 pads for capacitors (TBD).

FEATURES OF THE LCD MODULE

ITEM	STANDARD VALUE	UNIT
LCD Type	2.8" QVGA TFT-LCD	-
Backlight	4 White LEDs in parallel	-
Module size	50.0(W) x 69.2(H) x 4.2(T) (with Touch Panel TP)	mm
TP viewing area	44.80 (W) x 63.10 (H)	mm
TP active area	44.20 (W) x 62.50 (H)	mm
LCD active area	43.20 (W) x 57.60 (H)	mm
Dot number	240 (RGB) x 320	-
Pixel pitch	0.18(h) x 0.18(v)	mm
Operation temperature	-10 ~ 70	°C
Storage temperature	-30 ~ 80	°C
Driver IC	ILI9325	-
Interface mode	8080 system 16 bit interface	-
Color mode	262k / 65k software control	-

ELECTRICAL CHARACTERISTICS*

ITEM		SYMBOL	CONDITION	MIN.	TYP.	MAX.	UNIT
Supply Voltage	Logic	V_{DD}	--	2.9	3.0	3.4	V
Input Voltage	H level	V_{IH}	--	$0.7V_{DD}$	--	V_{DD}	V
	L level	V_{IL}		0	--	$0.3V_{DD}$	
Current Consumption		I_{DD}	With internal voltage generation $V_{DD} = 3.0V$; $T_{amb} = 25℃$	--	--	20	mA

MAXIMUM RATINGS*

ITEM	SYMBOL	MIN.	MAX.	UNIT
Supply Voltage	V_{DD}	-0.4	4.0	V
Input Voltage	V_{IN}	-0.4	V_{DD} + 0.4	V
Operating temperature	T_{OPR}	-20	70	℃
Storage temperature	T_{STR}	-30	80	℃
Humidity	--	--	90	%RH

*Remarks: data for the bare LCD module only.

BACKLIGHT CHARACTERISTIC

ITEM	SYMBOL	MIN.	TYPICAL	MAX.	UNIT
LED module Forward voltage	V_{LED}	3.0	--	3.4	V
LED module current	I_{LED}	--	80	--	mA
L/G Surface Luminance	L_S	3200	--	4500	Cd/m^2
LCM Surface brightness uniformity	L_D	80	--	--	%

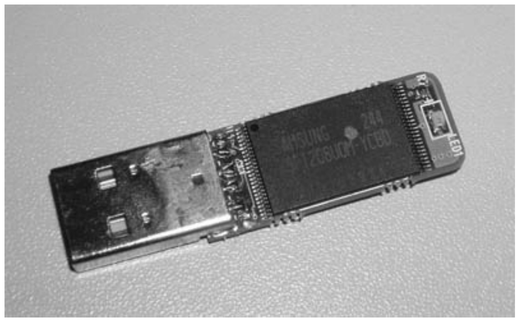
SCHEMATIC

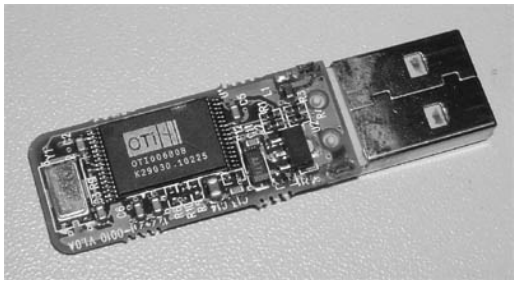
Please refer to our web site at

http://www.techtoys.com.hk/Displays/TY280T240320/TY280T_240320_BO.htm

for schematic of the breakout board. Besides the LCD module, there are several components of interest:

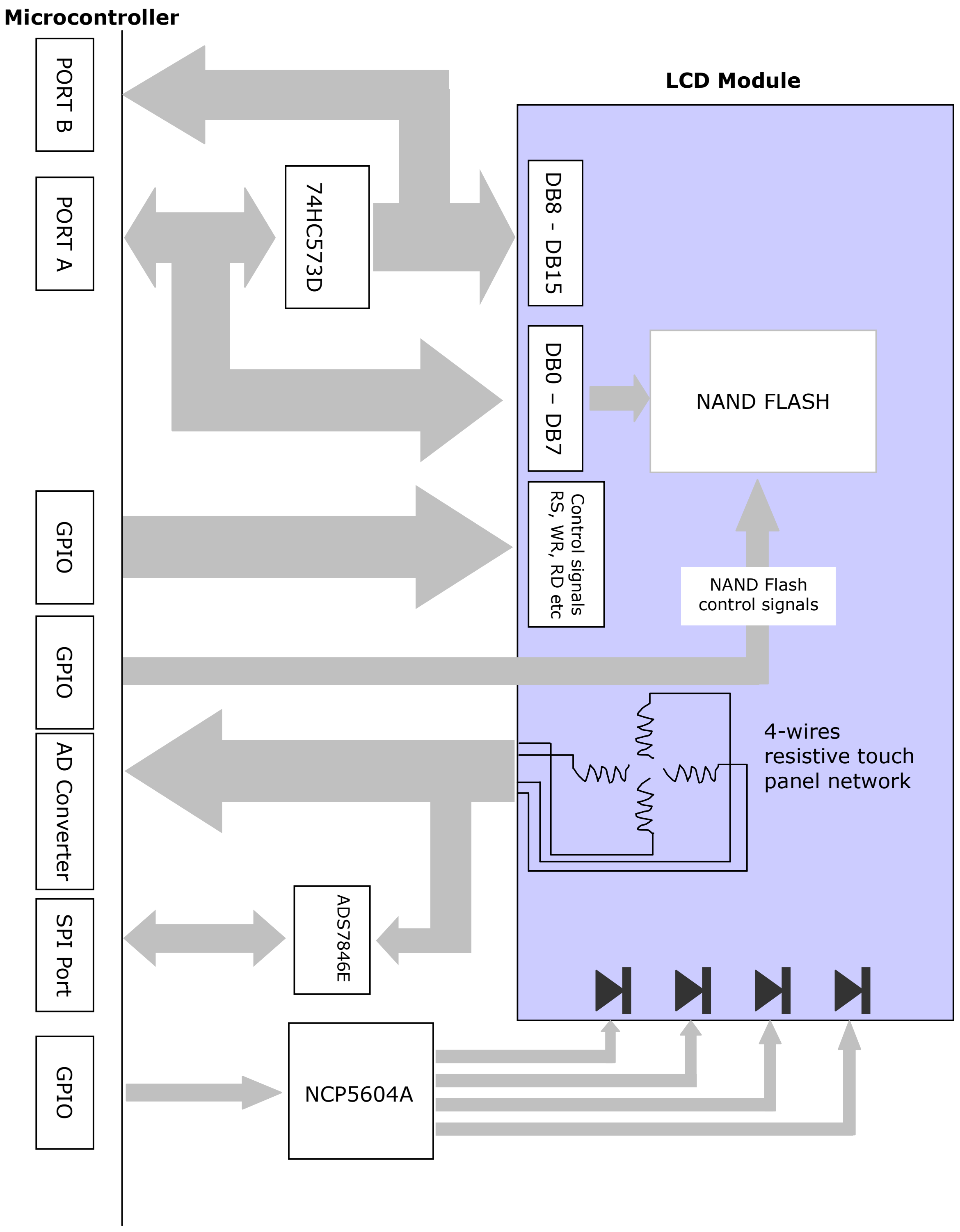
1. LED backlight circuit driven by On Semiconductor® NCP5604A
2. 32MB*8bit NAND Flash from Hynix Semiconductor®. This IC is useful for mass data storage such as jpeg or bitmap files. This is no stranger! Normally this chip is found in Thumb Drives today. Dedicated USB thumb drive controllers are used with firmware for low level USB connection, nandFlash driver as well as the USB Mass Storage Device firmware in a single controller chip (OTI as shown on the picture below).





3. Touch Panel controller ADS7846E from Texas Instrument®. This controller is optional because it is possible to interface directly the resistive wires to AD converters of a microcontroller. An example has been provided by Microchip GUI library. Demonstration program is available from our web site with port for Microchip Library using the Touch Panel. However, it is just in case that you do not want to use the Microchip library, ADS7846E is a nice device to play with. Please make sure JP2 – JP5 pads would be shunt to enable the controller. You need to apply solder a bit to these pads to complete the bridges joining XR, XL, YU, YD wires to ADS7846's inputs.
4. 3-State Octal latch 74HC573D for data IO. It is because the LCD module uses 16-bit addressing, we need a data latch to extend the bus width from 8-bit to 16-bit if we are using a low-end 8-bit microcontroller. A good example is PIC18LF4550 with embedded USB port. However, if there are enough data lines from the microcontroller, it is always advisable to use 16-bit addressing for higher efficiency. What we need is just to make sure JP6 jumper is not connected to disable the latch (high impedance output). In this case, we may use a complete 16-bit port to drive LCD module. Referring to block diagram on next page, it could be a microcontroller's PORT[0:15] directly interface DB[0:15] of the module.

BLOCK DIAGRAM



SOFTWARE : Microchip Graphics Display Solution

There are a number of GUI libraries known to the author. To name a few of those:

1. Easy GUI by IBIS Solutions ApS (www.easygui.com)
2. emWin supplied by Segger Microcontroller GmbH & Co. KG (www.segger.com)
3. Graphical User Interface (GUI) libraries by RAMTEX International (www.ramtex.dk)
4. µC/GUI by Micrium (www.micrium.com)
5. wxWidgets (http://www.wxwidgets.org/)

All of them focus on GUI libraries for embedded systems with charges. The last item being the wxWidget is an open source GUI; however, it seems not for small embedded systems as it requires C++ and powerful processor and OS like Windows CE.

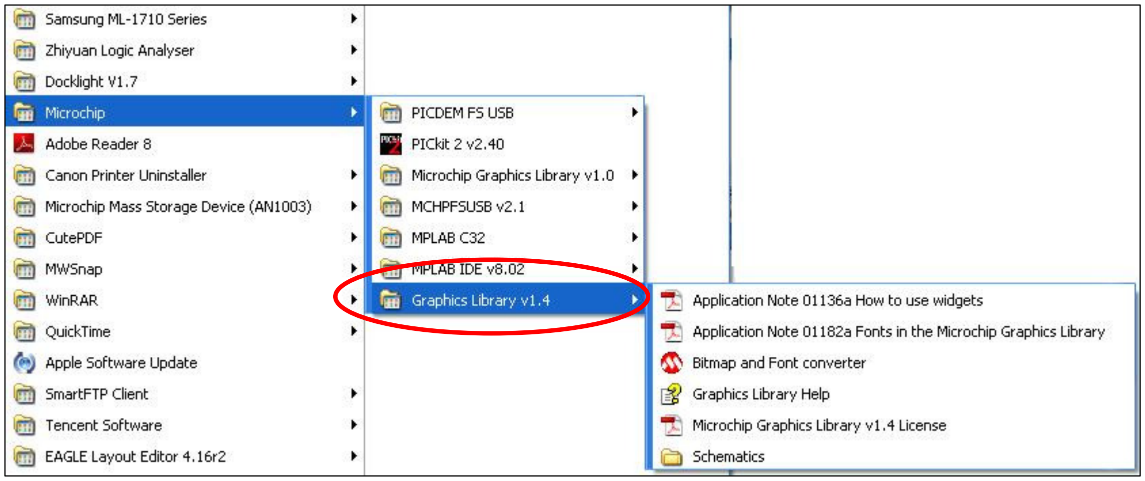
The author is deeply surprised by the strategy of Microchip that a full-blown Graphical User Interface (GUI) library is provided free with source code. One may find its relevant information from Microchip web site at

http://www.microchip.com/stellent/idcplg?IdcService=SS_GET_PAGE&nodeId=2608&param=en532067

Web seminars and application notes are available for download at no cost and a Free Licensed Microchip Graphic Library with schematics, drivers, documentation, and utilities also there as long as you are using this GUI on Microchip products. Normally such drivers and support would be sold at a certain cost, sometimes rather expensive too. Now we may use this library with only minor modification. This manual describes the procedure to port a new driver IC to the Microchip Graphic Library.

PROCEDURE

The first step is to download a copy of the Microchip Graphics Library. The current version at time of writing is v1.4. After download and installation, one would get a new folder under ***C:\Microchip Solutions*** if the default configurations have been accepted. Under Windows Explorer, browse to ***All Programs\Microchip\Microchip Graphics Library v1.4*** you will see all documentations in pdf and a help file. Full detail of the library has been provided under the ***Microchip Graphics Library Help (html help)*** and we hereby follow its instruction to port a new LCD driver for our 2.8" LCD module.


Samsung ML-1710 Series
Zhiyuan Logic Analyser
Docklight V1.7
Microchip
Adobe Reader 8
Canon Printer Uninstaller
Microchip Mass Storage Device (AN1003)
CutePDF
MWSnap
WinRAR
QuickTime
Apple Software Update
SmartFTP Client
Tencent Software
EAGLE Layout Editor 4.16r2
PICDEM FS USB
PICkit 2 v2.40
Microchip Graphics Library v1.0
MCHPFSUSB v2.1
MPLAB C32
MPLAB IDE v8.02
Graphics Library v1.4
Application Note 01136a How to use widgets
Application Note 01182a Fonts in the Microchip Graphics Library
Bitmap and Font converter
Graphics Library Help
Microchip Graphics Library v1.4 License
Schematics

LIBRARY STRUCTURE

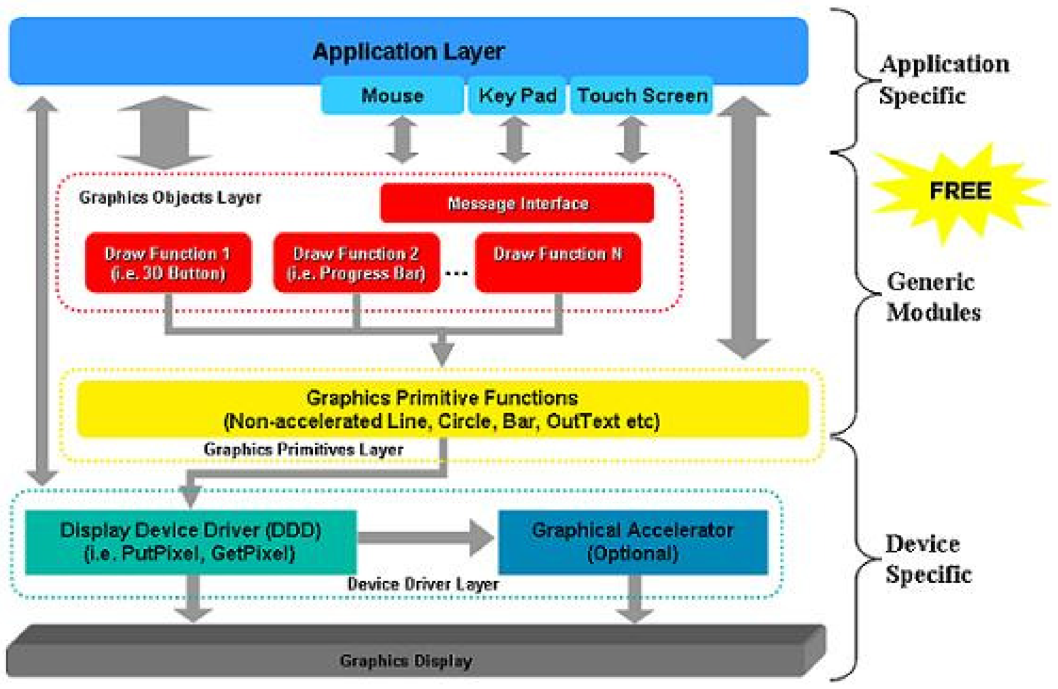
There are hundreds of LCD driver-ICs in the world. At time of writing only the following driver-ICs are supported by the original Graphics Library. They are located under

..\Microchip Solutions\Microchip\Graphics for *.c driver and
..\Microchip Solutions\Microchip\Include\Graphics for *.h header

Driver IC	Orientation
Densitron HIT1270L	Landscape
LG LGDP4531	Landscape / portrait
Renesas R61505	Landscape / portrait
Samsung S6D0129 / S6D0139	Landscape / portrait
Sino Wealth SH1101A OLED	Landscape
Orise SPFD5408	Landscape / portrait
Solomon Sys Tech SSD1339 OLED	128*128 matrix
Solomon Sys Tech SSD1906	Landscape / portrait
Solomon Sys Tech SSD1303	Landscape

If we are going to use the library with a third-party LCD module, we need to port the graphics library to our target driver IC. Thanks to the modular design, we only need to deal with the low-level I/O layer for ILI9325.

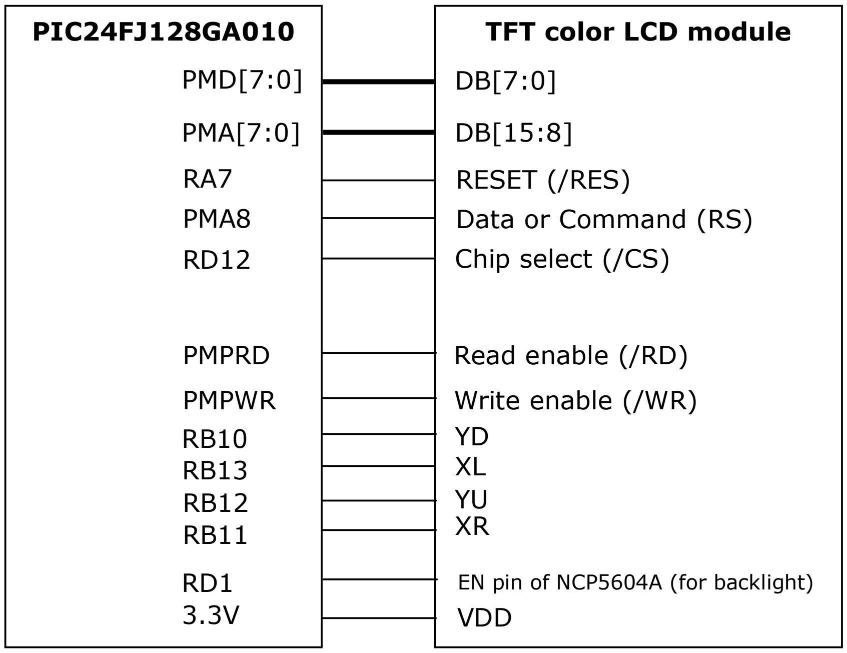
Navigate to ***Library Structure*** under ***Contents*** of the Help file we learn the library architecture. We only need to deal with the Device Specific layer and there are three files involved.


Application Layer
Mouse
Key Pad
Touch Screen
Graphics Objects Layer
Message Interface
Draw Function 1 (i.e. 3D Button)
Draw Function 2 (i.e. Progress Bar)
Draw Function N
Graphics Primitive Functions (Non-accelerated Line, Circle, Bar, OutText etc)
Graphics Primitives Layer
Display Device Driver (DDD) (i.e. PutPixel, GetPixel)
Graphical Accelerator (Optional)
Device Driver Layer
Graphics Display
Application Specific
FREE
Generic Modules
Device Specific

They are
..\Microchip\Graphics\device.c
..\Microchip\Include\Graphics\device.h
..\Microchip\Include\Graphics\Graphics.h

PRIMITIVE DEMO

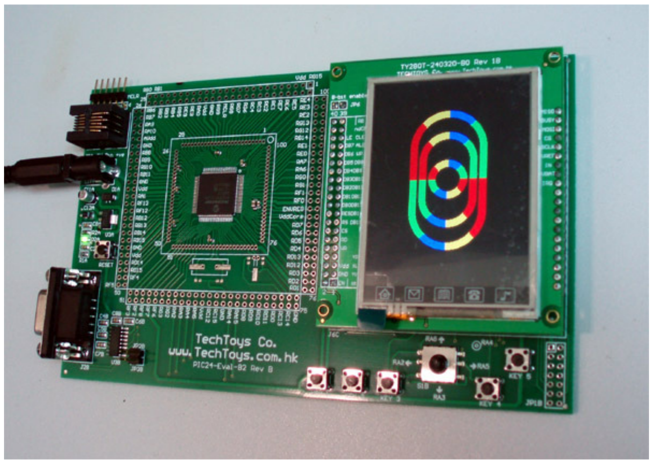
In our case, *device.c* and *device.h* become *ILI9325P_16BIT.c* and *ILI9325P_16.BIT.h* with indication of the driver IC, portrait orientation denoted as "P", and it is a 16 bit driver. There are several steps involved to start a new project with Microchip Graphics Library of course. One may refer to the help.html file again and navigate to ***Miscellaneous Topics*** → ***Starting a New Project*** to learn all relevant procedures to create a fresh project from scratch. Else, one may also consider downloading our fully-built example to get a kick-start with the Primitive Demo. This demonstration program is based on the hardware platform (part no: PIC24-Eval-B2 Rev B) for PIC24FJ128GA010 microcontroller. Let's take a look at the wiring for development board PIC24-Eval-1B.



Relevant hyperlinks:

www.techtoys.com.hk/Displays/TY280T240320/TY280T_240320_BO.htm under Doc 09

www.techtoys.com.hk/PIC_boards/PIC24-Eval-B2/PIC24-Eval-B2_RevB.htm



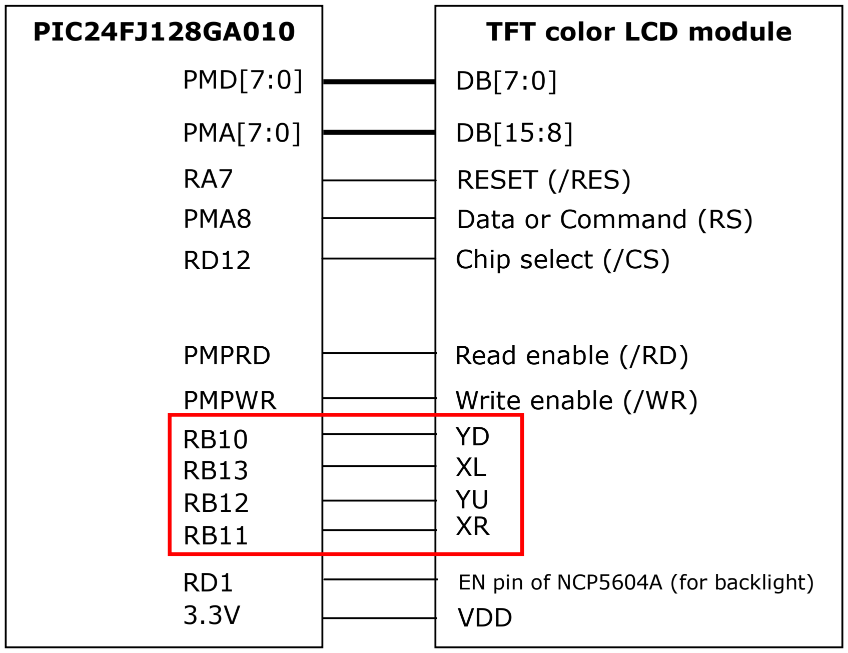
TOUCH PANEL DEMO (AN1136)

This example was modified from the source code for the application note AN1136 released by Microchip. This note discusses the know-how to use Widgets in Microchip Graphics Library. A port for our 2.8" TFT LCD has been finished with full source code available from the following hyperlink.

www.techtoys.com.hk/Displays/TY280T240320/TY280T_240320_BO.htm under Doc 10

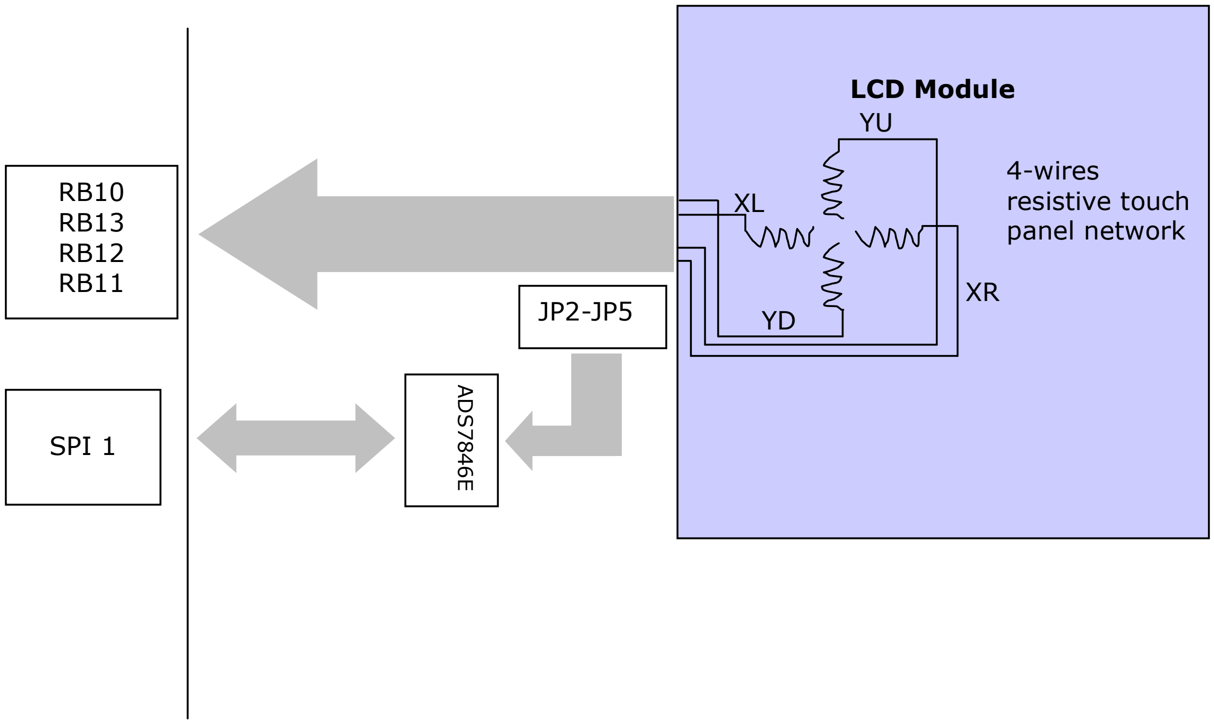
Location of AN1136 is available at
http://www.microchip.com/stellent/idcplg?IdcService=SS_GET_PAGE&nodeId=2608&param=en532067

In Touch Panel demo, we need to emphasize RB10, RB13, RB12, and RB11 with an interface to YD (Y Down), XL (X left), YU (Y Up), and XR (X Right) respectively.



A better illustration is repeated by the block diagram next page with focus on the Touch Panel portion.

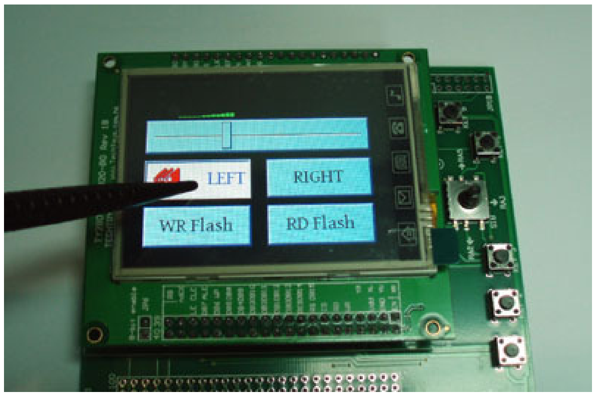
Microcontroller



We have two choices to decode the Touch Panel signal:

1. Use internal Analog-to-Digital converter (ADC) of the PIC24 microcontroller
2. Use the external ADS7846E Touch Panel driver onboard by manually shunt jumpers JP2 to JP5.

Since we are using the Microchip library for now, the first option has been used. Two files *TouchScreen.h* and *TouchScreen.c* take care of the decode task. A real-life demo by running the program is shown below. One may use finger or stylus for touch panel action.



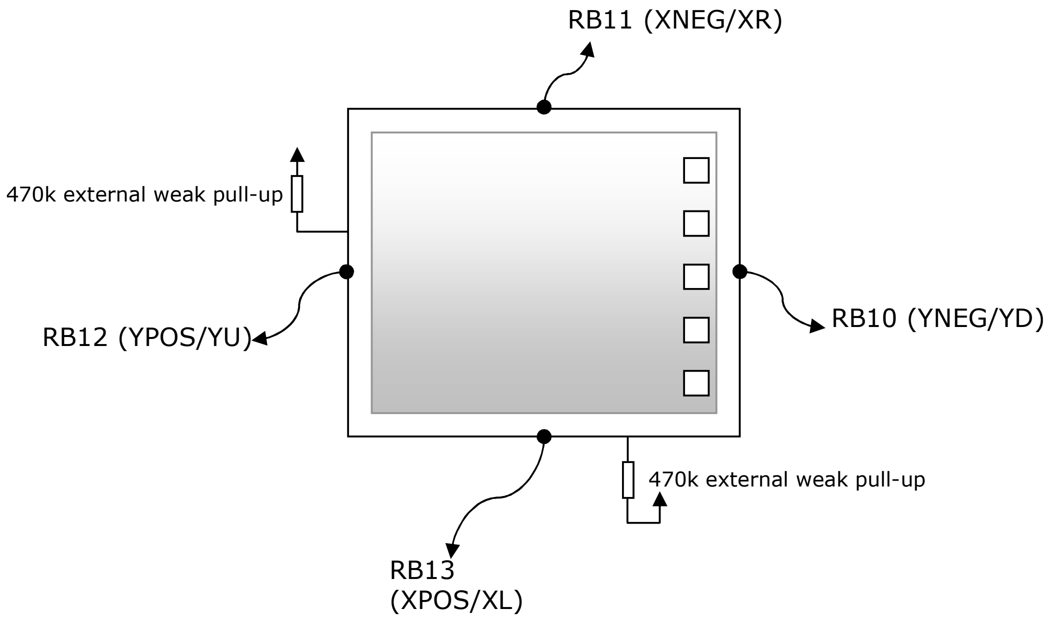
The file *TouchScreen.h* provides definitions for ADC channels plus a few necessary calibration constants such as *YMINCAL* and *YMAXCAL*. If you need to modify the source code for your particular hardware, you need to change the definitions

```
#define ADC_XPOS   13
#define ADC_YPOS   12
```

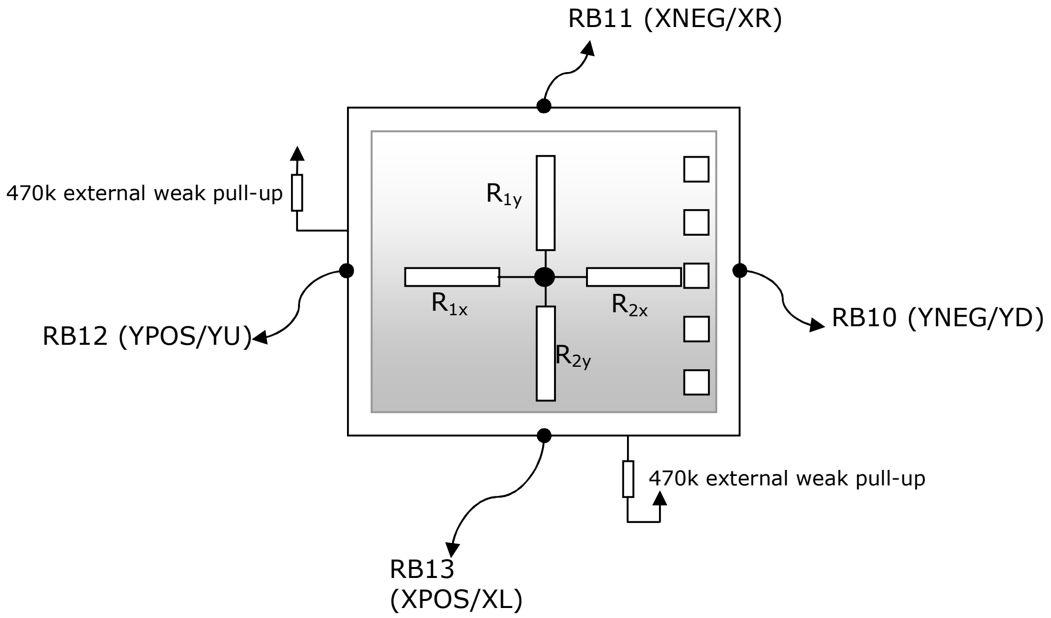
as well as X and Y port definitions on next page.

```
// Y port definitions
#define ADPCFG_XPOS        AD1PCFGbits.PCFG13
#define ADPCFG_XNEG        AD1PCFGbits.PCFG11
#define LAT_XPOS           LATBbits.LATB13
#define LAT_XNEG           LATBbits.LATB11
#define TRIS_XPOS          TRISBbits.TRISB13
#define TRIS_XNEG          TRISBbits.TRISB11

// X port definitions
#define ADPCFG_YPOS        AD1PCFGbits.PCFG12
#define ADPCFG_YNEG        AD1PCFGbits.PCFG10
#define LAT_YPOS           LATBbits.LATB12
#define LAT_YNEG           LATBbits.LATB10
#define TRIS_YPOS          TRISBbits.TRISB12
#define TRIS_YNEG          TRISBbits.TRISB10
```

We may illustrate the wiring diagram below if we view the LCD module in the landscape direction. Please note that this diagram is true only for the hardware platform PIC24-Eval-B2 Rev B evaluation board + TY280T-240320-BO Rev 1B LCD module.



A resistive touch panel is composed of several layers. The most important are two thin metallic electrically conductive and resistive layers separated by thin space. When the panel is touched at a certain position, the layers are connected at that point. The panel then acts similar to two voltage dividers with connected outputs.



The touch position may be calculated by the following equations.

Voltage in the X direction = $[R_{2x}/(R_{1x}+R_{2x})]$*Full voltage range
Voltage in the Y direction = $[R_{1y}/(R_{1y}+R_{2y})]$*Full voltage range

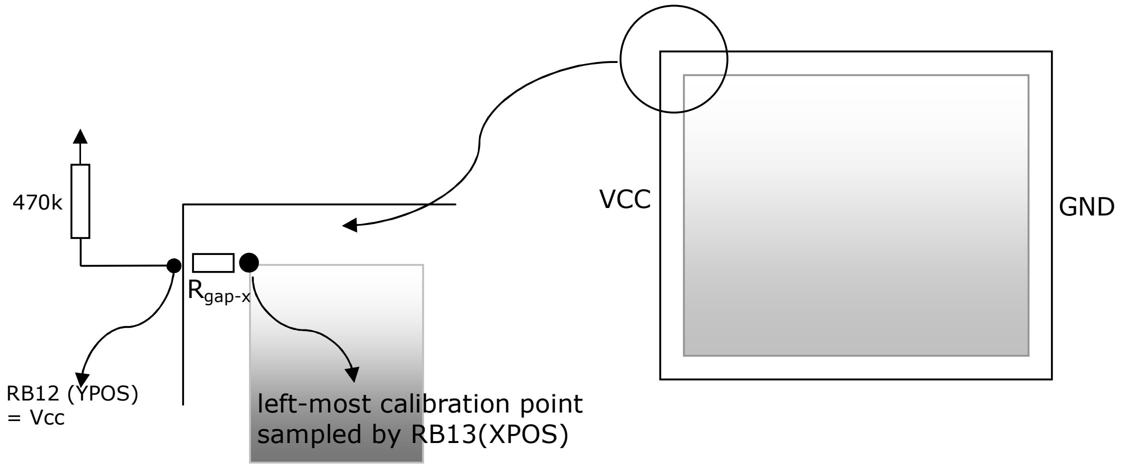
These two equations provide just a simplified model. The actual implementation is a lot more complicated than this. First, one needs to make sure the panel would be sampled frequent enough so that there is no valid touch action to miss. This is the purpose of timer interrupt *_T3Interrupt(void)* in *TouchScreen.c*. The period of Timer3 interrupt is defined under the function *TouchInit(void)* as *TOUCH_DELAY_PRESS*, which in turns has been defined as *1800/DEBOUNCE (1800/2=900)*. This provides a sampling period of approximately 450µsec. A state machine is defined in Timer3 interrupt function. First RB12 and RB10 would be "*energized*" to Vcc and ground to provide a voltage source for the potential divider in x-direction. Then, RB13 will be sampled for ADC input. If it is not a "no-touch" result and two conversions are equal, this is considered to be a valid touch action and the sequence will be swapped for a measurement in the y-direction *(one should refer to the source code under __attribute__ _ADC1Interrupt(void) for details)*.

Because every touch panel has its intrinsic resistance, the values for *XMINCAL*, *YMAXCAL*, *XMINCAL*, *XMAXCAL*, and *CAL_DELTA (namely default calibration values)* would be different. These constants are defined under the file *TouchScreen.h* and they are important for telling if the Touch panel has been touched or not. Therefore it is advised to calibrate for their variation prior to usage. One may read from the source code under *_ADC1Interrupt(void)* to note that the default calibration values do play an important role of how the program flow.

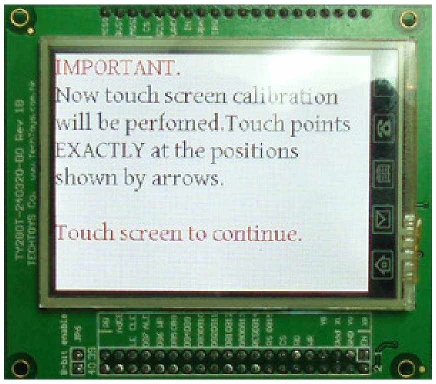
```
void __attribute__((interrupt, shadow, auto_psv)) _ADC1Interrupt(void)
{
static SHORT prevRes = -1;
static SHORT tempX;
SHORT res;

   res = ADC1BUF0;

   switch(state){
      case SET_X_CHANNEL:
         break;

      case MEASURE_X:
         // Check if screen is not touched
         if( (res<(XMINCAL-CAL_DELTA)) ||  (res>(XMAXCAL+CAL_DELTA)) )
         {
             PR3 = TOUCH_DELAY_PRESS;
             res  = -1;
         }else{
               PR3 = TOUCH_DELAY_MOVE;
         }
…..
```

Listing above shows an extract of the ADC routine. The testing condition for a no-touch is that, if the ADC result '*res'* is smaller than the constants *XMINCAL-CAL_DELTA* or it is larger than *XMAXCAL+CAL_DELTA*, the variable *'res'* will be assigned a value of -1 (0xFFFF) which implies no-touch.

Let's magnify the panel. If one looks closely to the edge of the Touch Panel, you will find a gap between the panel edge and the LCD pixel area underneath. That means the highest value measured from XPOS when no-touch will be higher than the left-most calibration point, say.

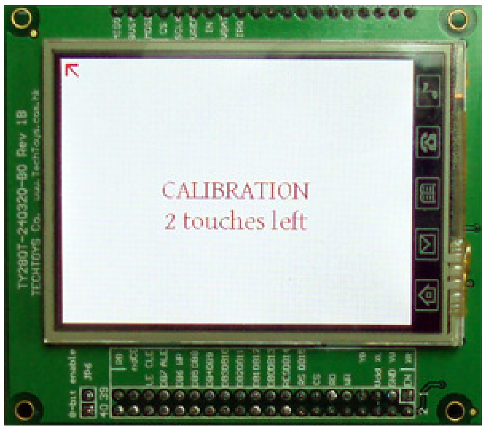


One may find out the maximum and minimum calibration values by **holding KEY1** on PIC24-Eval-B2 board and **press RESET**. The following screen will be shown.


IMPORTANT.
Now touch screen calibration will be perfomed.Touch points EXACTLY at the positions shown by arrows.

Touch screen to continue.

Touch screen to continue to bring up the next calibration screen. Try touching the top left corner to calibrate the panel. Calibration results will be stored in eeprom of PIC24 evaluation board after finishing all three calibration points at the top left, bottom left, and middle right. All goes smooth except when I changed the LCD module of a different specification. My experience is that, if *XMAXCAL* has been defined a value too low or *XMINCAL* too high; statements in the switch-case *MEASURE_X* will never measure a trigger at the top left, say.



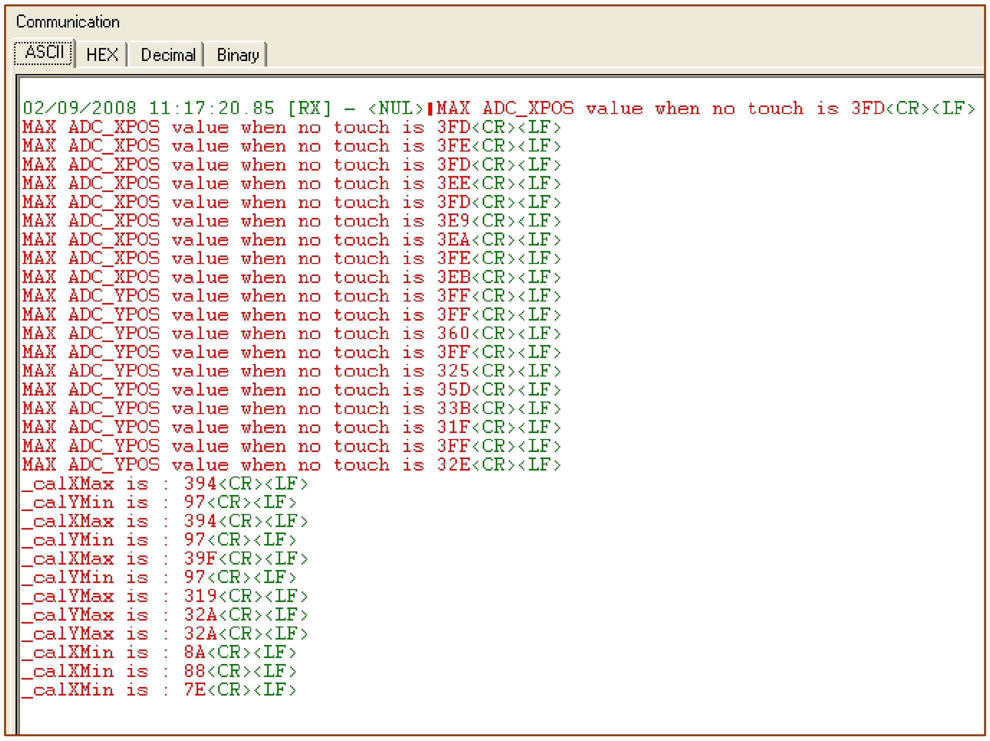
ADC result measured at RB13 at top left
= Vcc – (Vcc *R_{gap-x}/R_x)

with R_{gap-x} a finite value.

If we take noise into account, the ADC result is likely to fluctuate with numerous factors such as circuit design, PCB layout, power supply noise, etc. Suitable values of *XMAXCAL, XMINCAL, YMINCAL, and YMAXCAL* may be measured by simple printf statements append right after successful ADC measurements in the function *TouchCalibration(void).*

Connect a straight cable from J28 standard RS232 female connector to your PC's COM PORT. Start Docklight (www.docklight.de) or HyperTerminal Program and adjust the baud rate to 19200bps, 8-n-1 to record the calibration values. Restart the calibration procedure by holding KEY1 and press RESET. A screen shot similar to that on next page will be shown.

Screen shot of Docklight reflects several issues:



```
02/09/2008 11:17:20.85 [RX] - <NUL>MAX ADC_XPOS value when no touch is 3FD<CR><LF>
MAX ADC_XPOS value when no touch is 3FD<CR><LF>
MAX ADC_XPOS value when no touch is 3FE<CR><LF>
MAX ADC_XPOS value when no touch is 3FD<CR><LF>
MAX ADC_XPOS value when no touch is 3EE<CR><LF>
MAX ADC_XPOS value when no touch is 3FD<CR><LF>
MAX ADC_XPOS value when no touch is 3E9<CR><LF>
MAX ADC_XPOS value when no touch is 3EA<CR><LF>
MAX ADC_XPOS value when no touch is 3FE<CR><LF>
MAX ADC_XPOS value when no touch is 3EB<CR><LF>
MAX ADC_YPOS value when no touch is 3FF<CR><LF>
MAX ADC_YPOS value when no touch is 3FF<CR><LF>
MAX ADC_YPOS value when no touch is 360<CR><LF>
MAX ADC_YPOS value when no touch is 3FF<CR><LF>
MAX ADC_YPOS value when no touch is 325<CR><LF>
MAX ADC_YPOS value when no touch is 35D<CR><LF>
MAX ADC_YPOS value when no touch is 33B<CR><LF>
MAX ADC_YPOS value when no touch is 31F<CR><LF>
MAX ADC_YPOS value when no touch is 3FF<CR><LF>
MAX ADC_YPOS value when no touch is 32E<CR><LF>
_calXMax is : 394<CR><LF>
_calYMin is : 97<CR><LF>
_calXMax is : 394<CR><LF>
_calYMin is : 97<CR><LF>
_calXMax is : 39F<CR><LF>
_calYMin is : 97<CR><LF>
_calYMax is : 319<CR><LF>
_calYMax is : 32A<CR><LF>
_calYMax is : 32A<CR><LF>
_calXMin is : 8A<CR><LF>
_calXMin is : 88<CR><LF>
_calXMin is : 7E<CR><LF>
```

1. MAX ADC_XPOS value measured at XPOS when YPOS=Vcc, YNEG=GND, and no touch. This value fluctuated with a range from 0x3E9 to 0x3FD. The theoretical max 10-bit ADC result is 0x3FF so this is pretty close to it.

2. MAX ADC_YPOS value measured at YPOS when XPOS=Vcc, XNEG=GND, and no touch. This value fluctuated more than XPOS though the program flow is basically the same as case 1, except a simple swap between the measurement points. It ranges from a rather low value 0x31F to 0x3FF.

3. _calXMax when the Touch Panel was calibrated at the top left corner of the *active LCD area.* It recorded two instances of 0x394 and one instance of 0x39F. The max value (i.e. 0x39F) would be selected and saved in eeprom for position calculation later. This value is less than the least MAX ADC_XPOS value (0x3E9). Basically if we make a decision that, once the measured result '*res'* is larger than 0x39F, the touch point would be considered off-screen for XPOS. This translates to the following syntax.

   ```
   #define XMAXCAL     0x39F
   …
   if (res>XMAXCAL) res = -1;
   ….
   ```

   The original source code didn't include any comment about why a CAL_DELTA factor should be summed to XMAXCAL, nor why CAL_DELTA should be 0x60 in the original code. The author's feeling is that these are intrinsic properties of a Touch Panel. Every manufacturer may have different series of the same Touch Panel model number but the resistance may vary. In addition, a different circuit design and PCB layout is also playing important role since we are doing ADC on a digital circuit. By trail-an-error, a value of CAL_DELTA=0x30 instead of 0x60 has been used for 2.8" TFT LCD which seems fairly stable.

XMAXCAL+CAL_DELTA
= 0x39F+0x30
= 0x3CF

Once again, if '*XMAXCAL + CAL_DELTA*' has been assigned a value too low; ADC result '*res*' will always be higher than *XMAXCAL+CAL_DELTA* leading to no response from the top left corner during the calibration procedure. On the other hand, if this is set a value too high (or simply 0x3FF), the variable '*res*' will always be smaller even when there is no touch action leading to false trigger.

4. _calYMin when the Touch Panel was calibrated at the top left corner. It recorded all three instances of 0x97. This value is going to be YMINCAL.

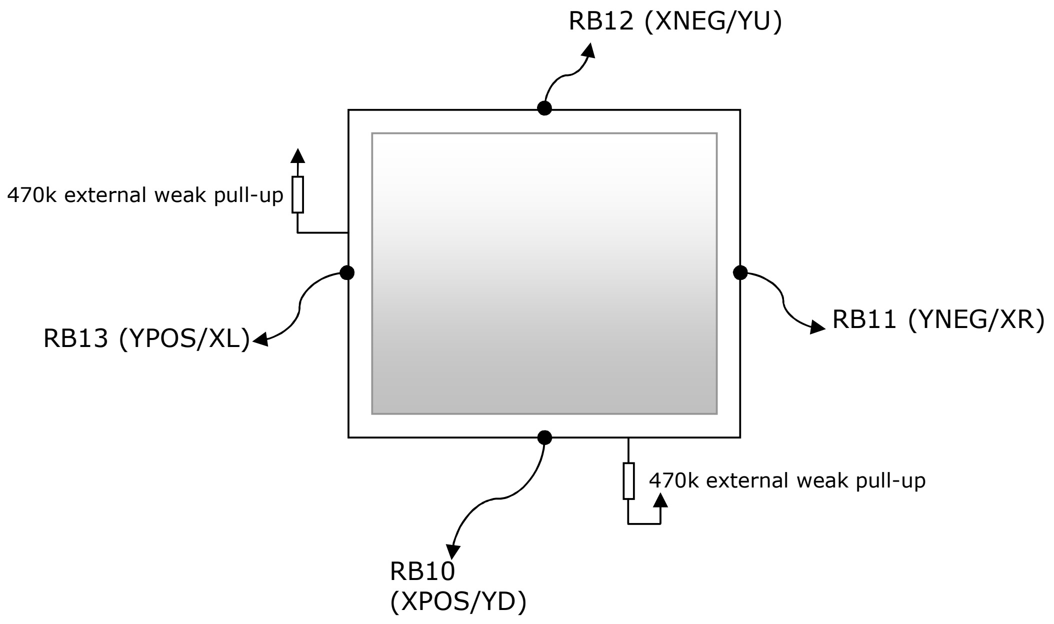
5. _calYMax when the Touch Panel was calibrated at the lower left corner. Similarly a maximum value of 0x32A has been recorded. However, ADC_YPOS was found to be much nosier than ADC_XPOS with a range from 0x31F to 0x3FF. With YMAXCAL = 0x32A and a CAL_DELTA of 0x30, there will be false trigger when the Touch Panel was not even touched at all. Fortunately, noise screening algorithm such than 2 consecutive measurements and a successful record for adcX values is required prior to adcY to avoid mistake.

6. _calXMin when the Touch Panel was calibrated at the right middle edge of the active LCD area. A value of 0x7E assigned to *XMINCAL*.

By finishing the calibration, one may continue and experience the modified program from us for button and slide-bar widgets, and a simple demo for nandFlash read/write.

TOUCH PANEL DEMO FOR A 3.5" PANEL

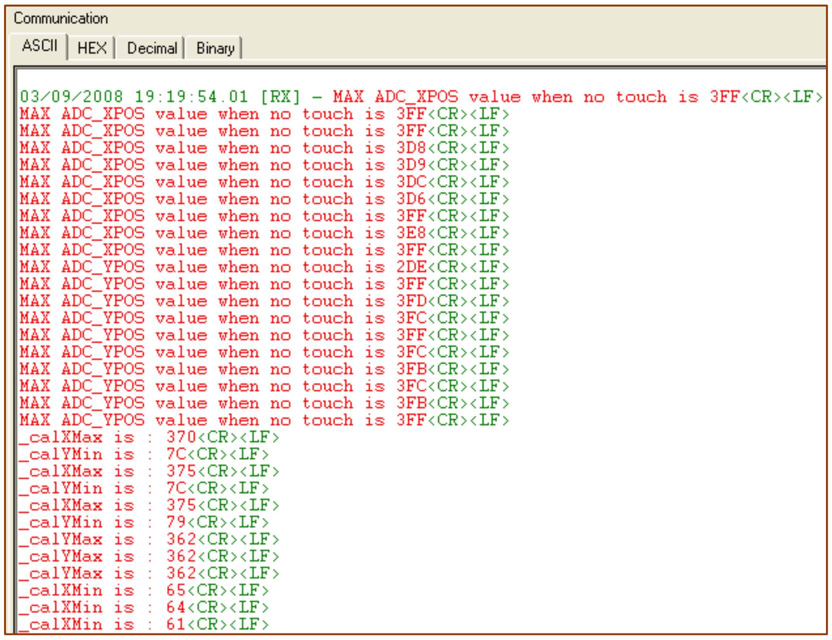
So, what to do if there is a new hardware? A new product is being developed with 3.5" LCD module with external image processor. Now, this new LCD module has Touch Panel built-in without funny icons like the 2.8" LCD. Its aspect ratio and intrinsic resistance are different as well. The schematic is also different. The connection with ADC channels when it is stacked on PIC24 Evaluation board is shown below.



In order to keep XPOS and YPOS to the same edges at bottom and left, the following definition has been adapted in the header *TouchScreen.h*.

```
…
#define ADC_XPOS        10
#define ADC_YPOS        13

…
// Y port definitions
#define ADPCFG_XPOS       AD1PCFGbits.PCFG10
#define ADPCFG_XNEG       AD1PCFGbits.PCFG12
#define LAT_XPOS          LATBbits.LATB10
#define LAT_XNEG          LATBbits.LATB12
#define TRIS_XPOS         TRISBbits.TRISB10
#define TRIS_XNEG         TRISBbits.TRISB12

// X port definitions
#define ADPCFG_YPOS       AD1PCFGbits.PCFG13
#define ADPCFG_YNEG       AD1PCFGbits.PCFG11
#define LAT_YPOS          LATBbits.LATB13
#define LAT_YNEG          LATBbits.LATB11
#define TRIS_YPOS         TRISBbits.TRISB13
#define TRIS_YNEG         TRISBbits.TRISB11
```

By the same calibration procedure described in last section, the following result has been found. It seems even though it is a larger screen with totally different intrinsic resistance, the calibration constants do not vary a lot. After finishing the calibration, the Touch Panel was tested showing fast and crisp responses.



```
03/09/2008 19:19:54.01 [RX] - MAX ADC_XPOS value when no touch is 3FF<CR><LF>
MAX ADC_XPOS value when no touch is 3FF<CR><LF>
MAX ADC_XPOS value when no touch is 3FF<CR><LF>
MAX ADC_XPOS value when no touch is 3D8<CR><LF>
MAX ADC_XPOS value when no touch is 3D9<CR><LF>
MAX ADC_XPOS value when no touch is 3DC<CR><LF>
MAX ADC_XPOS value when no touch is 3D6<CR><LF>
MAX ADC_XPOS value when no touch is 3FF<CR><LF>
MAX ADC_XPOS value when no touch is 3E8<CR><LF>
MAX ADC_XPOS value when no touch is 3FF<CR><LF>
MAX ADC_YPOS value when no touch is 2DE<CR><LF>
MAX ADC_YPOS value when no touch is 3FF<CR><LF>
MAX ADC_YPOS value when no touch is 3FD<CR><LF>
MAX ADC_YPOS value when no touch is 3FC<CR><LF>
MAX ADC_YPOS value when no touch is 3FF<CR><LF>
MAX ADC_YPOS value when no touch is 3FC<CR><LF>
MAX ADC_YPOS value when no touch is 3FB<CR><LF>
MAX ADC_YPOS value when no touch is 3FC<CR><LF>
MAX ADC_YPOS value when no touch is 3FB<CR><LF>
MAX ADC_YPOS value when no touch is 3FF<CR><LF>
_calXMax is : 370<CR><LF>
_calYMin is : 7C<CR><LF>
_calXMax is : 375<CR><LF>
_calYMin is : 7C<CR><LF>
_calXMax is : 375<CR><LF>
_calYMin is : 79<CR><LF>
_calYMax is : 362<CR><LF>
_calYMax is : 362<CR><LF>
_calYMax is : 362<CR><LF>
_calXMin is : 65<CR><LF>
_calXMin is : 64<CR><LF>
_calXMin is : 61<CR><LF>
```