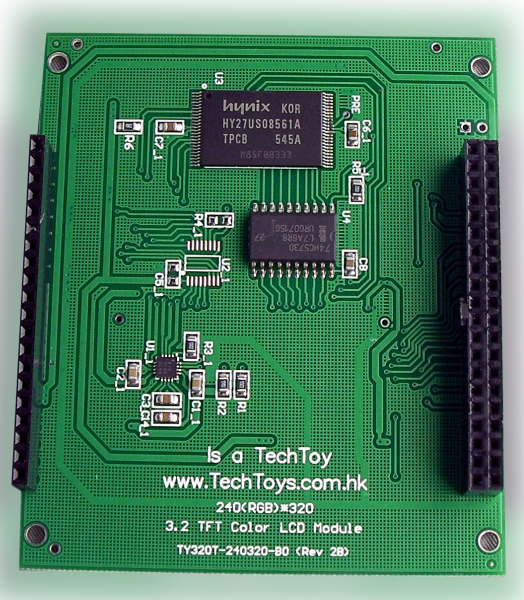
3.2” QVGA 262k TFT LCD module with Microchip Graphics Library



INTRODUCTION

The part number TY320T_230320_BO (Board Rev 2B) is a development board for 3.2" QVGA TFT-LCD module completed with a 32MBx8bit NAND Flash, white LED backlight circuit, and a 4-wire touch screen controller all soldered onboard. Standardized 2.54mm PCB sockets have been included for sack of easy prototyping.

FEATURES OF THE LCD MODULE

ITEM	STANDARD VALUE	UNIT
LCD Type	3.2" QVGA TFT-LCD	-
Backlight	5 White LEDs in parallel	-
Module size	57.54 x 79.20 x 4.4 (with touch panel TP)	mm
TP viewing area	51.20 x 71.20	mm
TP active area	50.20 x 70.20	mm
LCD active area	48.60 x 64.80	mm
Dot number	240 (RGB) x 320	-
Pixel pitch	0.2025(h) x 0.2025(v)	mm
Operation temperature	-10 ~ 70	°C
Storage temperature	-30 ~ 80	°C
Driver IC	ILI9320	-
Interface mode	8080 system 16 bit interface	-
Color mode	262k / 65k configurable via software	-

SCHEMATIC

Please refer to our web site at

http://www.techtoys.com.hk/Displays/TY320T240320/TY320T240320.htm

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

1. LED backlight circuit by Texas Instrument's TPS60230
2. 32MB*8bit NAND Flash from Hynix Semiconductor
3. Touch screen controller ADS7846E from Texas Instrument again. However, this controller is optional because it is also possible to interface directly the 4-wire resistive wires to AD converters of a microcontroller for touch panel signal. An example is provided by Microchip GUI library
4. 3-State Octal latch 74HC573 for data IO. This device provides an alternative to 16-bit addressing to the LCD module

SOFTWARE : Microchip Graphics Display Solution

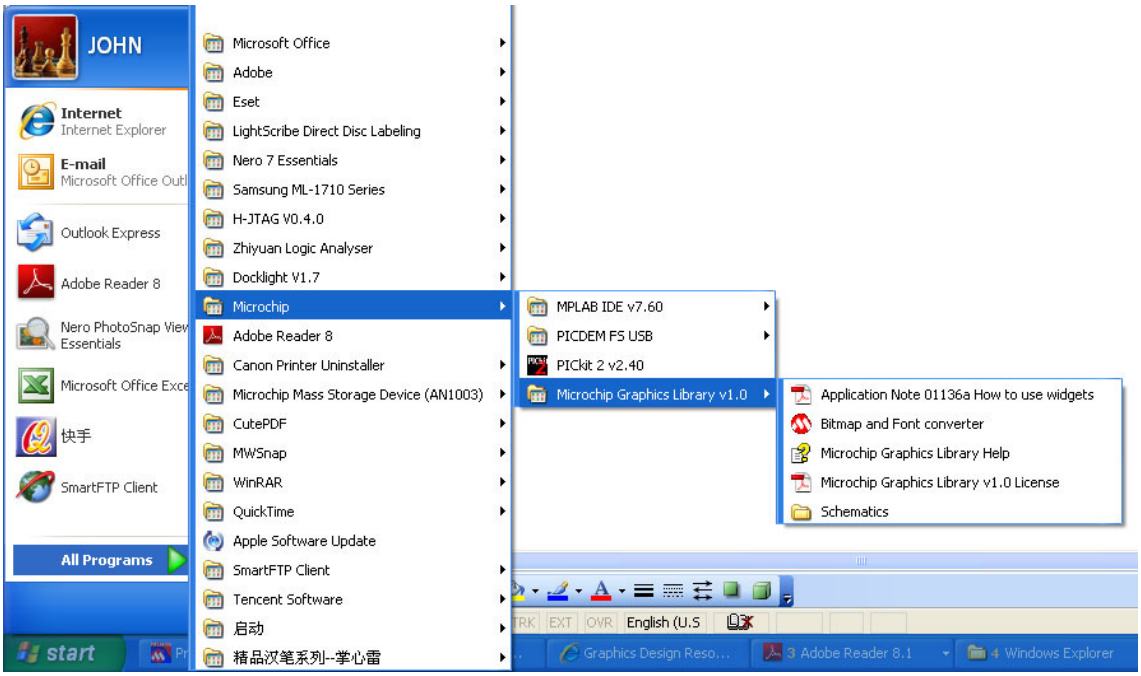
The author was deeply surprised by the Microchip move that a full-blown Graphical User Interface (GUI) library is provided free with source code. One may find its relevant information from Microchip web site at

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. 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 is to describe 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 v1.0. After download and install, 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 v1.0*** 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 3.2" LCD module.



LIBRARY STRUCTURE

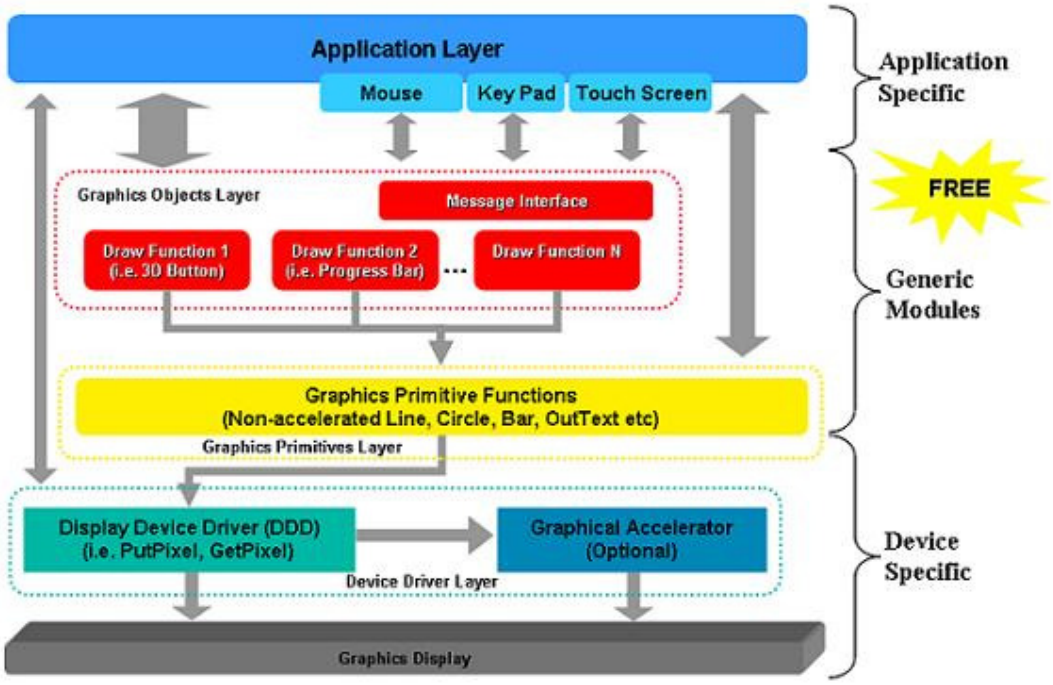
There are hundreds of LCD driver-ICs in the world. At time of writing only the following driver-ICs are supported by the original Graphics Library. They are located under

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

Driver IC	Orientation
Densitron HIT1270L	Landscape
LG LGDP4531L	Landscape
Renesas R61505U	Landscape
Samsung R61505U	Portrait
Samsung S6D0129L	Landscape
Samsung S6D0129P	Portrait
Solomon Sys Tech SSD1339	Landscape / portrait

If we are going to use the library with a third-party LCD module, we need to do a bit scripting. Thanks to the modular design, we only need to deal with the low-level I/O layer for ILI9320 driver IC.

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.



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

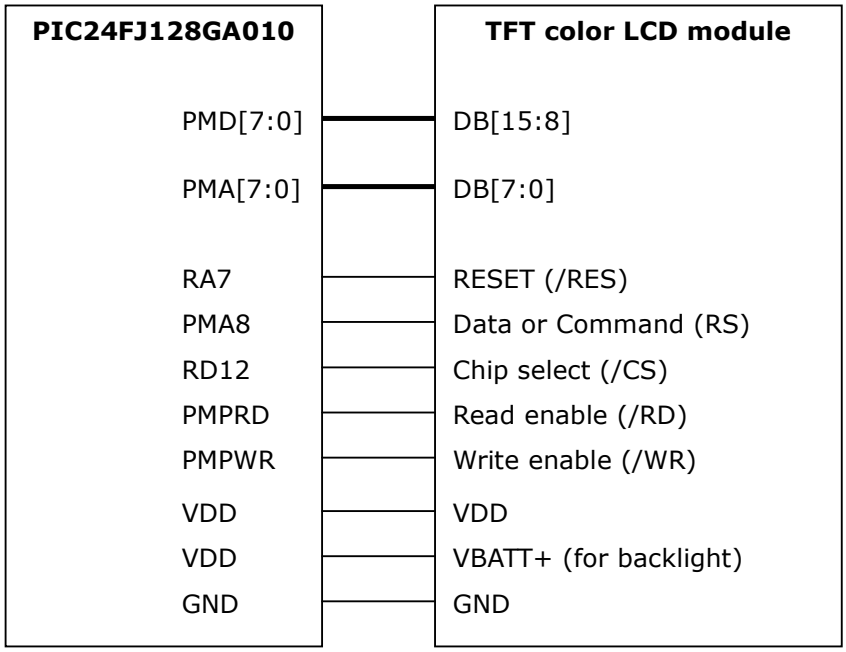
In our case, *device.c* and *device.h* become *ILI9320P_16BIT.c* and *ILI9302P_16.BIT.h* with indication of the driver IC, portrait orientation, and it is a 16 bit driver. There are other steps involved to start a new project with Microchip Graphics Library of course. One may refer to the help.html file 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 kickstart with the Primitive Demo.

The location of our example is located under Doc 03 at http://www.techtoys.com.hk/Displays/TY320T240320/TY320T240320.htm.

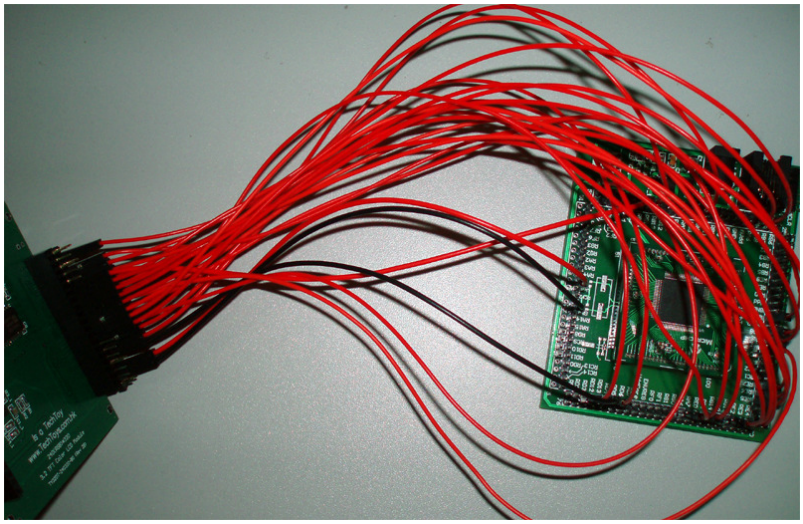
The Parallel Master Port (PMP) module is a new feature of Microchip PIC24 series. It is a parallel 8-bit I/O module specifically designed to communicate with parallel devices, such as communication peripherals, LCDs, and external memory devices, etc. One may refer to advanced information (DS39713A) from Microchip for further details. The PMP module has been used for ILI9320 driver. Because it is a 8-bit data I/O module whereas our LCD is a 16-bit device, there are two ways to solve this issue:

1. Use 74HC573 device to latch lower byte thus convert an 8-bit I/O to 16-bit. The down side of this method is that, it is not possible to read data back from the LCD because 74HC573 is a one-way device. The good thing is; we save 8 data lines for the microcontroller.
2. Use all 16 pins from the microcontroller for all data lines DB[15:0] of the LCD module. Because there are only 8-bit I/O for data being the PMD[7:0], we need to figure out where come the other 8-bit I/O lines which must be relevant to PMP module. In my case, the address bus PMA[7:0] has been employed to solve this issue.

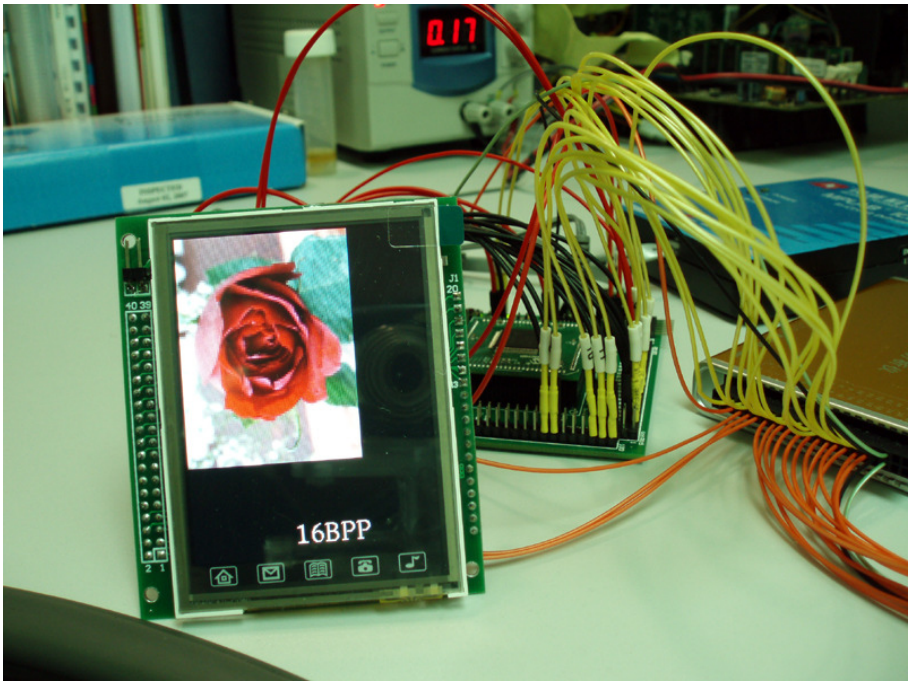
First of all, let's take a look at the wiring.



The PMP module is a highly configurable I/O module which means it can be more than a single way to connect the LCD. This hardware design is NOT the sole configuration to make thing work but it serves a good start. The hardware employed for this experiment is a simple breakout board for PIC24FJ128GA010 (Part number PIC24-Eval-B1 Rev 1a) plus "few" pieces 0.100" that fly from the breakout board to LCD. It looks messy but it just takes a little patience to complete the wiring. No soldering required!



Basically once we finished the wiring with the microcontroller programmed, we would be able to view the demo in action as follows. A QuickTime movie is available under Doc 04 at http://www.techtoys.com.hk/Displays/TY320T240320/TY320T240320.htm.



THE PMP MODULE

To understand how things work we need to study the timing diagram of the LCD plus the syntax required for the PMP module to generate the corresponding I/O signals. The driver IC of the LCD is ILITEK's ILI9320 from Taiwan. Its data sheet is available for download from a simple keyword search on Google/Yahoo. This 115-page data sheet looks daunting at first but what really matters are the interface mode and instruction sets.

First, we need to accept the fact that TY320T_230320_BO has been configured internally for i80-system 16-bit interface. This was done by wiring the IM3:IM0 pins to be 0:0:1:0 internally. The 262k or 65k color can be displayed through the 16-bit microcontroller interface. When the 262k color is selected, two transfers (1st transfer: 2 bits, 2nd transfer: 16 bits or 1st transfer: 16 bits, 2nd transfer: 2 bits) are necessary for the 16-bit microcontroller interface. On the other hand, if 65k is selected, only a single data transfer in 16-bits would be required. The color format would be configured as 5-bit RED, 6-bit GREEN, and 5-bit BLUE in such case. One may refer to page 28 of ILI9320 data sheet for more details.

Whether the display is configured to display 262k or 65k is selected by TRI bit in the Entry Mode registry. Instruction registers control all display settings of the LCD glass such as the frame frequency, contrast, color mode, display ON/OFF, etc. The Entry Mode register belongs to one of those. Normally, initialization of a LCD would be done prior to all display jobs. What initialization means is just a sequence of writing different parameters to individual instruction register. For example, if the Entry Mode register has been written a value of 0x1030, the corresponding bit position of this 16-bit register would be set to 0b0001 0000 0011 0000.

	D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
Entry Mode (R03h)	TRI	DFM	0	BGR	0	DACKE	HWM	0	ORG	0	I/D1	I/D0	AM	0	0	0
0x1030	0	0	0	1	0	0	0	0	0	0	1	1	0	0	0	0

The resultant effect is :

AM = 0	⇒ GRAM address is updated in horizontal writing direction
I/D[1:0] = 1:1	⇒ address counter automatically increment for both horizontal and vertical
ORG = 0	⇒ Origin address not moved
BGR = 1	⇒ Swap RGB data to BGR in writing into GRAM
TRI = 0	⇒ 65,536 colors. With 16-bit interface, 1 pixel updated by a single data transfer

One may refer to the data sheet for how such parameters would change the display settings. Nevertheless, after a proper LCD initialization the correct (or appropriate) parameters would be established suiting individual LCD glass. In our case, such initialization routine is performed by the C function **ResetDevice (void)** in **ILI9320P_16BIT.c**. Because the Microchip Graphics Library has been designed in a modular manner, we just need to change this **ResetDevice** function plus a few low-level functions to use a new display.

```
void ResetDevice(void){

   CS_TRIS_BIT = 0;                                          (1a)
   CS_LAT_BIT = 1;                                           (1b)

   RST_TRIS_BIT = 0;                                         (2a)
   RST_LAT_BIT = 0;                                          (2b)

   // PMP setup
   PMMODEbits.MODE  = 0b10;          // Master 2             (3a)
   PMMODEbits.WAITB = 0b00;                                  (3b)
   PMMODEbits.WAITM = 0b0001;                                (3c)
   PMMODEbits.WAITE = 0b00;                                  (3d)
   PMAEN = 0x01FF;                                           (3e)
   PMCONbits.PTRDEN = 1;                                     (3f)
   PMCONbits.PTWREN = 1;                                     (3g)
   PMCONbits.PMPEN  = 1;             //enable PMP module     (3h)

   // Reset controller
   RST_LAT_BIT = 0;                                          (4a)
   DelayMs(2);
   RST_LAT_BIT = 1;                                          (4b)
   DelayMs(2);


       .....
    SetReg(0xe5,0x8000);             //start osc             (5a)
    SetReg(0x00,0x0001);                                     (5b)
       .....
```

Listing 1 Extract of ResetDevice(void)

Listing 1 shows an extract of the ResetDevice(void) function. For complete source code, please refer to file ILI9320P_16BIT.c available at Doc 03 http://www.techtoys.com.hk/Displays/TY320T240320/TY320T240320.htm.

Every hardware design will be different from one another so the PMP setup will also change. In our case, the lower byte of 16 bits address (PMA[7:0]) has been used for data DB[7:0] of the LCD, therefore we need to perform code line 3e as follow.

```
PMAEN = 0x01FF  //to enable addressing for PMA[7:0] plus PMA[8] for RS
```

Honestly, the author didn't successfully configure everything in one shot. It took a couple of hours for trail-and-error before a seemingly working I/O pattern could be obtained. What matters for a new driver to work is to make sure the functions *SetReg(BYTE index, WORD value)* which in turns *SetIndex(index)* and *WriteData(byte1,byte0)* match the timing requirement.

Listing 2 shows *SetReg(BYTE index, WORD value).*

```
void  SetReg(BYTE index, WORD value){
   SetIndex(index);
   WriteData(((WORD_VAL)value).v[1],((WORD_VAL)value).v[0]);
}
```

Listing 2 SetReg function

The function *SetIndex(index)* selects which index register to write.
Function *WriteData(byte1,byte0)* writes 16-bit word content to an index register.

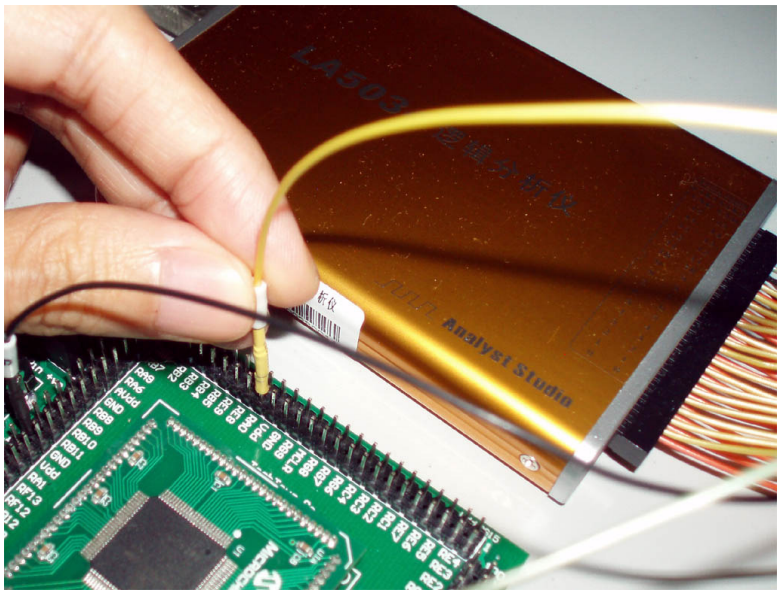
```
#define SetIndex(index)
PMADDR=(0x00FF&index);PMPDelay();PMDIN1=0x00;PMPDelay();
```

Listing 3 SetIndex(index)

```
#define WriteData(byte1,byte0)
PMADDR=(0x0100|byte0);PMPDelay();PMDIN1=byte1;PMPDelay();
```

Listing 4 WriteData(byte1,byte0)

Digital signals displayed in timing diagrams help a lot in this case. We always don't know what is going on with those 16-bit I/O plus all those control signals during program debug. A Logic Analyzer wired up to monitor the waveforms. Breakpoints at appropriate position revealed the key information on signals as shown next page.



Here shows the beauty of the PMP module. There is no discrete I/O operation for WR, RD, CS, and DC(RS) required. Once the PMP module has been configured correctly, these signals will be generated automatically for you when PMDIN1 is modified.

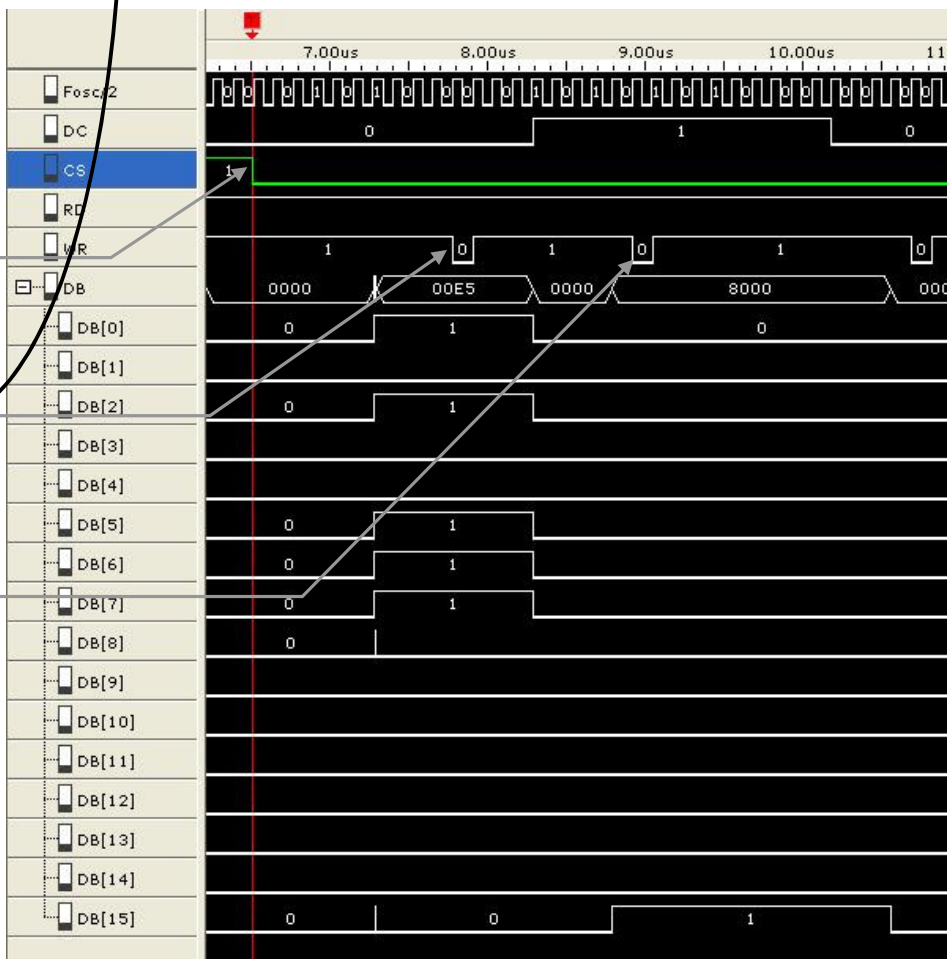
```
Nop();
Nop();

// Enable LCD
CS_LAT_BIT = 0;

/*** Setup display for ILI9320 driver IC ***/
SetReg(0xe5,0x8000);         //start osc
SetReg(0x00,0x0001);
DelayMs(10);
SetReg(0xa4,0x0001);         //calb

SetReg(0x07,0x0000);         //display control(1)
DelayMs(10);

/*** Display Setting ***/
SetReg(0x01, 0x0100);   //Driver output control (1)
```



LA triggered at CS from High to Low. This is the breakpoint

WR strobe from High to Low when data bus shows 0xe5

The second WR strobe when data bus 0x8000 established

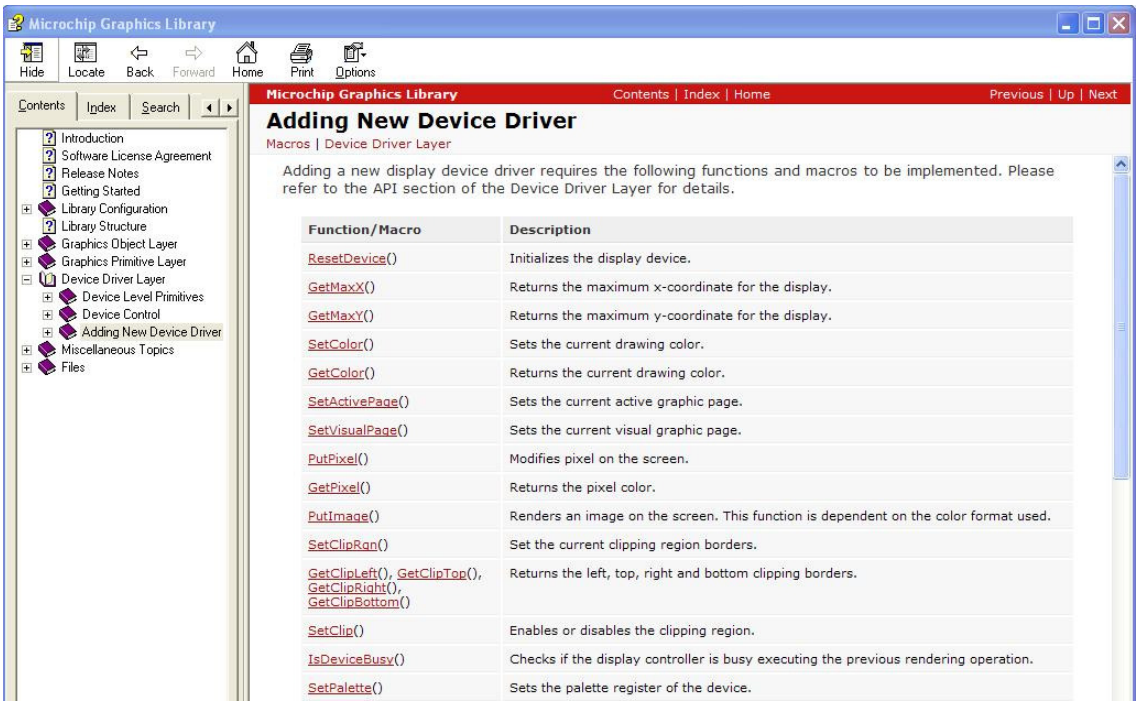
From the timing diagram, it is possible to optimize the delay between successive data writes to speed up display. For present settings, time delay between two WR strobes recorded 1.14us with a WR strobe width of 130ns. We have plenty of space before the limit of 100ns for bus cycle time and 50ns for write low-level pulse width would be reached.

Next, we need to define the correct index registers for SetAddress function under ILI9320P_16BIT.h. This is the function to locate a particular pixel on the LCD panel.

```
#define SetAddress(addr2,addr1,addr0)  \
SetIndex(0x20);WriteData(0,addr0);\
SetIndex(0x21);WriteData(addr2,addr1);\
SetIndex(0x22);
```

Listing 5 SetAddress(addr2,addr1,addr0)

Then we need to define the SCREEN_HOR_SIZE and SCREEN_VER_SIZE under ILI9302P_16BIT.h. For more systematic way of adding a new device driver, please refer to help.html again under which there is a complete section of how a new LCD driver could be created.



Last but not the least, the last step is to include the following statement under Graphics.h header.

#include "ILI9320P_16BIT.h”

Again, a complete project is available under Doc 03 at http://www.techtoys.com.hk/Displays/TY320T240320/TY320T240320.htm.

The story does not finish here. There remains the NAND Flash driver, Touch Screen driver, and 8-bit operation that we need to work on further. Please refer to separate application notes for reference.