Document Information

<table>
<thead>
<tr>
<th>Abstract</th>
<th>This document introduces how to control MPU type LCD module by using EBI interface.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Apply to</td>
<td>Any NuMicro® Cortex®-M0 or Cortex®-M4 series which supports EBI function.</td>
</tr>
</tbody>
</table>

The information described in this document is the exclusive intellectual property of Nuvoton Technology Corporation and shall not be reproduced without permission from Nuvoton.

Nuvoton is providing this document only for reference purposes of NuMicro microcontroller based system design. Nuvoton assumes no responsibility for errors or omissions.

All data and specifications are subject to change without notice.

For additional information or questions, please contact: Nuvoton Technology Corporation.

www.nuvoton.com
# Table of Contents

## 1 OVERVIEW
- 1.1 Signal Connection ........................................................................................................... 3
- 1.2 EBI Register Setting ........................................................................................................ 5
- 1.3 Display Frame Rate Calculation ....................................................................................... 7

## 2 SAMPLE CODE
- 2.1 Overview .......................................................................................................................... 8
- 2.2 LCD Initialization by EBI Interface ................................................................................... 9
- 2.3 Main Function .................................................................................................................... 9
- 2.4 LCD Module Initialization ............................................................................................... 10
- 2.5 LCD Module Commands ............................................................................................... 13
- 2.6 Drawing Functions ........................................................................................................... 17
- 2.7 Software Definitions ........................................................................................................ 20

## 3 APPENDIX ......................................................................................................................... 22
1 Overview

The NuMicro® Cortex®-M0 or Cortex®-M4 series microcontroller (MCU) provides the External Bus Interface (EBI) function that can be used to control MPU type (e.g. Motorola 6800 or Intel 8080) LCD controller for most displaying applications.

Either 8 or 16-bit data width of EBI can be used to connect with external MPU type LCD module depending on how higher color depth of displaying in application will be used.

This application note provides a guide for user to know how to adjust signal timing and obtain the best display quality of LCD panel by configuring the EBI function in the NuMicro® Cortex®-M0 or Cortex®-M4 MCU series.

In Chapter 2, sample source code is provided for user reference.

The following configuration of LCD module is based on GFT024CA240320 which supports Intel 8080 interface provided by GI FAR Technology Co., Ltd.

1.1 Signal Connection

This section describes how to connect the NuMicro® series MCU with the GFT024CA240320 MPU type LCD module pin by pin.

User can refer to the Nano130 learning board in Figure 1.1-1 and test sample code on it.
The following table lists some important pins of the LCD module.

<table>
<thead>
<tr>
<th>PIN</th>
<th>Definition</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>CS</td>
<td>Chip select signal</td>
</tr>
<tr>
<td>7</td>
<td>RS</td>
<td>Register select signal</td>
</tr>
<tr>
<td>8</td>
<td>WR</td>
<td>Write strobe signal</td>
</tr>
<tr>
<td>9</td>
<td>RD</td>
<td>Read strobe signal</td>
</tr>
<tr>
<td>10-27</td>
<td>DB0~DB17</td>
<td>Data Bus 0 ~ 17</td>
</tr>
<tr>
<td>28</td>
<td>RESET</td>
<td>Reset pin</td>
</tr>
</tbody>
</table>

Table 1.1-1 Pin Definition of LCD Module

The following table shows the pins of EBI interface which will be used to connect with the LCD module based on the Nano130 series (8-bit data width is used as example).

<table>
<thead>
<tr>
<th>Definition</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>nCS</td>
<td>Chip select enable pin</td>
</tr>
<tr>
<td>nWR</td>
<td>Write enable pin</td>
</tr>
<tr>
<td>nRD</td>
<td>Read enable pin</td>
</tr>
<tr>
<td>AD0~AD7</td>
<td>Address/Data bus bit 0~7</td>
</tr>
</tbody>
</table>

Table 1.1-2 Pin Definition of Nano130 Series

Some I/O pins are also required to control the LCD module, as listed in the following table.

<table>
<thead>
<tr>
<th>Definition</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>GPIO-0(GPB6)</td>
<td>Active as register select signal (RS)</td>
</tr>
<tr>
<td>GPIO-1(GPD8)</td>
<td>Active as reset signal (RESET)</td>
</tr>
</tbody>
</table>

Table 1.1-3 Pin Definition of RS and RESET

According to the GFT024CA240320 spec, if 8-bit data width is used, data bus pin of module needs to be connected from pin-10 to pin-17.
The following figure shows the actual connection between EBI interface pins of the NuMicro® Cortex®-M0/M4 MCU and MPU type LCD module.

![Figure 1.1-2 Pin Connection between NuMicro® Cortex® M0/M4 MCU and LCD Module](image)

1.2 EBI Register Setting

Before configuring the register of EBI function, user needs to check the timing requirement of GFT024CA240320 spec. The following is a writing data example of Intel 8080 bus timing waveform.

![Figure 1.2-1 Timing Waveform of Intel 8080 Bus](image)

Some details about the signals in Figure 1.2-1 are described below:

1) **RS** signal is at low when register data is written into a LCD module; otherwise, only data will be written. Note that the RS signal will be pulled low before CS is active.
2) **CS/WR** signal is at low when any register or data is written.

According to the timing waveform in Figure 1.2-1, user needs to configure EBI register and control GPIO pins to fit the timing requirement of LCD module.

![Timing Waveform](image)

**Figure 1.2-2 Timing Waveform of NuMicro® Cortex® M0/M4 MCU EBI Bus**

Based on Figure 1.2-2, some EBI timing arguments such as tALE and tAHD can be adjusted by users. A better EBI register configurations on the Nano100 series platform is described below, where the system (HCLK) runs at 42 MHz and EBI clock is the same as HCLK (EBICON[10:8]=0).

1) **tALE**=1 (EBICON[18:16]=0), which is the default value. The address output will be ignored by the LCD module, so as to make the value as small as possible.

2) **tACC**=2 (EXTIME[4:0]=1). According to timing requirement of LCD module, the minimum time of RD/WR at low level is 45us. Thus, the value should be set to 2 * MCLK = 2 * 23us = 46us.

3) **tAHD**=1 (EXTIME[10:8]=0), which is the default value. The minimum requirement of write data hold time is 10ns. Thus, the default value is adequate.

4) Other arguments of EBI configurations (e.g. ExtlW2X, ExtlR2W, and ExtlR2R) are all default values.

**Note:** The LCD module receives 16-bit data and MCU needs to send 8-bit data (MSB first) twice in one RS active duration.
The actual output timing waveform from MCU will look like as following figure.

![Waveform Diagram](image)

**Figure 1.2-3 Output Timing of EBI interface to Simulate Intel 8080 Interface**

### 1.3 Display Frame Rate Calculation

This section describes calculation of display frame rate based on LCD module which has 240x320 screen resolution according to EBI settings.

According to the EBI setting, display command sequence of LCD module, sample code program sequence and optimized level of Keil project, the frame rate can be calculated as follows:

Draw one pixel time = time of “Command 0x20” + “Command 0x21” + “Command 0x22”. The command timing waveform (including commands 0x20, 0x21, and 0x22) will be shown as the following figure.

![Timing Waveform Diagram](image)

**Figure 1.3-1 Timing Waveform of Drawing One Pixel**

The delta time between tag “C” and “D” will be about 13us. Thus, the time for drawing the whole screen (in this module is 240x320 pixels) will be 13us*240*320 = 998400us = **998.4ms**.

In other word, it takes about **one second** to draw a frame (a whole screen).

To get a better frame rate, the following methods maybe are good to use.

- Draw a partial area of screen instead of the entire screen which needs to be drawn.
- Optimize draw related functions in sample code.
- Use “-O3” compiler optimization flag in project settings, and the time of drawing one pixel will be reduced to 9us with about 30% improvement.
2 Sample Code

2.1 Overview

To use or understand sample code in this section, a Nano130 learning board and a LCD module are needed. The LCD module is mounted on the learning board.

The flow of drawing on the LCD module is shown as follows:

![Diagram showing the flow of drawing on the LCD module]

Figure 2.1-1 Flow of Drawing on LCD Module
2.2 LCD Initialization by EBI Interface

The EBI and GPIO initialization use the EBI ALE pin for RS signal control and GPD8 for RESET pin. Please configure EBI-0 for 8-bit data transfer and set the fastest timing.

```c
void EBI_LCM_Init(void)
{
    // Enable EBI function with 8-bit data width
    EBI_Open(0, EBI_BUSWIDTH_8BIT, EBI_TIMING_FASTEST, 0, 0);

    // configure EBI ALE pin to RS for LCD driver control
    SYS->PB_L_MFP &= ~0x0F000000;
    PB->PMD |= 0x00001000; // output mode
    PB->DOUT |= (1<<6); // default HIGH

    // configure PD[8] pin to RESET for manual driver control
    SYS->PD_H_MFP &= ~0x0000000F;
    PD->PMD |= 0x00010000; // output mode
    PD->DOUT |= (1<<8); // default HIGH
}
```

2.3 Main Function

Main loop displays portrait or landscape text strings on the screen.

```c
int main (void)
{
    ...
    /* Initialize EBI and pins */
    EBI_LCM_Init();

    // read something from LCD driver
    OTM3225A_IDRead(ID_READ_ADDR, &regcontent);
    printf("ID = 0x%x \n", regcontent);

    while(1)
    {
        /* Display module initialization */
        OTM3225_init(RES_PORTRAIT_240X320);
        ClearScreen(0x0000);
        Show_String32x32(1, 1, "Nuvoton", 0xf800);
        Show_String32x32(1, 33, "Tech.", 0xf800);
        Show_String32x32(1, 65, "Company", 0xf800);
        Show_String32x32(1, 97, "Nano 100", 0xf800);
    }
}
```
Show_String32x32(1, 129, "Portrait", 0xf800);
ClearScreen(0x0000);

    /* Display module initialization */
    OTM3225_init(RES_LANDSCAPE_320X240);
    ClearScreen(0x0000);
    Show_String32x32(1, 1, "Nuvoton", 0xf800);
    Show_String32x32(1, 33, "Tech.", 0xf800);
    Show_String32x32(1, 65, "Company", 0xf800);
    Show_String32x32(1, 97, "Nano 100", 0xf800);
    Show_String32x32(1, 129, "Landscape", 0xf800);
    ClearScreen(0x0000);
}

2.4 LCD Module Initialization

Before configuring a register of LCD module by EBI interface, software must use RESET pin to do reset procedure first.

The following configurations are just for reference. User can modify some arguments of the LCD module to fit the request of specific application.

    /**
     * @brief  Initialize OTM3225 lcd module
     * @param res specific landscape or portrait display
     * @return none
     */
    void OTM3225_init(E_RES_SIZE res)
    {
        RST_HIGH;
        TIMER_Delay(TIMER0, 1000);     // Delay 1ms
        RST_LOW;
        TIMER_Delay(TIMER0, 10000);    // Delay 10ms. This delay time is necessary.
        RST_HIGH;
        TIMER_Delay(TIMER0, 50000);    // Delay 50 ms

        // LCD initialization
        OTM3225A_RegWrite(0x00E3, 0x3008);     // Set internal timing
        OTM3225A_RegWrite(0x00E7, 0x0012);     // Set internal timing
        OTM3225A_RegWrite(0x00EF, 0x1231);     // Set internal timing

        OTM3225A_RegWrite(0x0001, 0x0000);     // SM:0, SS:0
        OTM3225A_RegWrite(0x0002, 0x0700);     // set 1 line inversion
if(res == RES_LANDSCAPE_320X240)
    OTM3225A_RegWrite(0x0003, 0x1038);  // set GRAM write direction and BGR=1.
(resolution: 320x240)
else if (res == RES_PORTRAIT_240X320)
    OTM3225A_RegWrite(0x0003, 0x1030);  // set GRAM write direction and BGR=1.
(resolution: 240x320)

OTM3225A_RegWrite(0x0004, 0x0000);  // RCV:0, RCH:0, RSZ:0
OTM3225A_RegWrite(0x0008, 0x0008);  // FP[3:0]:0x8, BP[3:0]:0x8
OTM3225A_RegWrite(0x0009, 0x0000);  // set non-display area refresh cycle

ISC[3:0]
    OTM3225A_RegWrite(0x000A, 0x0000);  // FMARK_OE:0, FMI[2:0]:0
    OTM3225A_RegWrite(0x000B, 0x0000);  // ENC[2:0]:0, RM:0(system interface),
    DM[1:0]:0, RIM[1:0]=0x0
OTM3225A_RegWrite(0x0000D, 0x0000);  // FMP[8:0]:0
OTM3225A_RegWrite(0x000F, 0x0000);  // VSPL:0, HSPL:0, EPL:0, DPL:0

//*************Power On sequence ***************/
OTM3225A_RegWrite(0x0010, 0x0000);  // SAP, BT[3:0], AP, DSTB, SLP, STB
OTM3225A_RegWrite(0x0011, 0x0007);  // DC1[2:0], DC0[2:0], VC[2:0]
OTM3225A_RegWrite(0x0012, 0x0000);  // VCIRE:0, VRH[3:0]:0
OTM3225A_RegWrite(0x0013, 0x0000);  // VDV[4:0]:0
TIMER_Delay(TIMER0, 200000);

OTM3225A_RegWrite(0x0010, 0x1690);  //OLD:0169 G:0149 Other:17B0
OTM3225A_RegWrite(0x0011, 0x0147);  // Set DC1[2:0], DC0[2:0], VC[2:0]
TIMER_Delay(TIMER0, 50000);

OTM3225A_RegWrite(0x0012, 0x0091);  // External reference voltage= Vci
TIMER_Delay(TIMER0, 50000);

OTM3225A_RegWrite(0x0013, 0x1300);  // Set VDV[4:0] for VCOM amplitude
OTM3225A_RegWrite(0x0020, 0x0000);  // AD[7:0]:0
OTM3225A_RegWrite(0x0021, 0x0000);  // AD[16:8]:0

// ------------ Adjust the Gamma Curve ---------//
OTM3225A_RegWrite(0x0030, 0x0000);
OTM3225A_RegWrite(0x0031, 0x0404);
OTM3225A_RegWrite(0x0032, 0x0304);
OTM3225A_RegWrite(0x0035, 0x0005);
OTM3225A_RegWrite(0x0036, 0x1604);
OTM3225A_RegWrite(0x0037, 0x0304);
OTM3225A_RegWrite(0x0038, 0x0303);
OTM3225A_RegWrite(0x0039, 0x0707);
OTM3225A_RegWrite(0x003C, 0x0500);
OTM3225A_RegWrite(0x003D, 0x000F);

//----------------------- Set GRAM area -----------------------
OTM3225A_RegWrite(0x0050, 0x0000); // HSA[7:0]:0
OTM3225A_RegWrite(0x0051, 0x00EF); // HEA[7:0]:0xEF
OTM3225A_RegWrite(0x0052, 0x0000); // VSA[7:0]:0
OTM3225A_RegWrite(0x0053, 0x013F); // VEA[7:0]:0x13F

if(res == RES_LANDSCAPE_320X240)
    OTM3225A_RegWrite(0x0060, 0xA700); // Gate Scan Line
else if (res == RES_PORTRAIT_240X320)
    OTM3225A_RegWrite(0x0060, 0x2700); // GS:0. NL[5:0]:0x27, SCN[5:0]:0

OTM3225A_RegWrite(0x0061, 0x0001); // NDL, VLE, REV
OTM3225A_RegWrite(0x0064, 0x0000); // VL[8:0]:0

//---------------------- Partial Display Control ----------------------
OTM3225A_RegWrite(0x0080, 0x0000); // PTD[8:0]:0
OTM3225A_RegWrite(0x0081, 0x0000); // PTS[8:0]:0
OTM3225A_RegWrite(0x0082, 0x0000); // PTE[8:0]:0
OTM3225A_RegWrite(0x0083, 0x0000); // PTD[8:0]:0
OTM3225A_RegWrite(0x0084, 0x0000); // PTS[8:0]:0
OTM3225A_RegWrite(0x0085, 0x0000); // PTE[8:0]:0

//---------------------- Panel Control ----------------------
OTM3225A_RegWrite(0x0090, 0x0010); // DIVI[1:0]:0, RTNI[4:0]:0x10
OTM3225A_RegWrite(0x0092, 0x0600); // NOWI[2:0]:0x6

OTM3225A_RegWrite(0x0097, 0x0133); // 262K color and display ON

g_otm3225a_resolution = res;
### 2.5 LCD Module Commands

The LCD module read/write related functions via EBI interface.

```c
/**
 * @brief Specific which register of lcd module will be read or written
 * @param[in] addr is register address
 * @return none
 */
void OTM3225A_IndexWrite(uint16_t addr)
{
    RS_LOW;   // RS = LOW
    EBIAccessWrite(0x00);
    EBIAccessWrite(addr & 0xFF);
    RS_HIGH;  // RS = HIGH
}

/**
 * @brief Write data to OTM3225A
 * @param[in] data to be written
 * @return none
 */
void OTM3225A_DataWrite(uint16_t data)
{
    EBIAccessWrite((data & 0xFF00)>>8);
    EBIAccessWrite(data & 0xFF);
}

/**
 * @brief Read data from OTM3225A
 * @param[in] data to be read
 * @return none
 */
void OTM3225A_DataRead(uint16_t *data)
{
    *data = 0;
    *data = (EBIAccessRead())<<8;
    *data |= EBIAccessRead();
}
```
* @brief  Read a dummy data from EBI interface
* @param[in]  none
* @return none
*/
void OTM3225A_DummyRead(void)
{
    volatile uint8_t data;

    data = EBIAccessRead();
}

/**
* @brief  Write data to OTM3225A register
* @param[in]  addr register address
* @param[in]  data to be written
* @return none
*/
void OTM3225A_RegWrite(uint16_t addr, uint16_t data)
{
    OTM3225A_IndexWrite(addr);
    OTM3225A_DataWrite(data);
}

/**
* @brief  Read data from OTM3225A register
* @param[in]  addr register address
* @param[out]  data register data
* @return none
*/
void OTM3225A_RegRead(uint8_t addr, uint16_t *data)
{
    OTM3225A_IndexWrite(addr);
    OTM3225A_DataRead(data);
}

/**
* @brief  Read OTM3225A's ID
* @param[in]  addr ID address in EBI region (OTM3225 is 0x00)
* @param[out]  data ID
* @return none
*/
void OTM3225A_IDRead(uint16_t addr, uint16_t *data)
{
    OTM3225A_IndexWrite(addr);

    RS_LOW;                  // RS = LOW
    *data = 0;
    *data = (EBIAccessRead())<<8;
    *data |= EBIAccessRead();
    RS_HIGH;                 // RS = HIGH
}

/**
 * @brief  Enter sleep mode
 * @param none
 * @return none
 */
void OTM3225A_SleepIN(void)
{
    OTM3225A_RegWrite(0x0007, 0x0000);  // Display off
    TIMER_Delay(TIMER0, 10);
    OTM3225A_RegWrite(0x0010, 0x0002);  // Set to Sleep mode
    TIMER_Delay(TIMER0, 10);
}

/**
 * @brief  Exit sleep mode
 * @param none
 * @return none
 */
void OTM3225A_ExitSleep(void)
{
    OTM3225A_RegWrite(0x0010, 0x1690);
    TIMER_Delay(TIMER0, 10);
    OTM3225A_RegWrite(0x0007, 0x0133);  // Display on
    TIMER_Delay(TIMER0, 10);
}

/**
 * @brief  Enter standby mode
 * @param none
 * @return none
 */
void OTM3225A_EnterStandby(void)
{
    OTM3225A_RegWrite(0x0007, 0x0000);  //Display off
    TIMER_Delay(TIMER0, 10);
    OTM3225A_RegWrite(0x0010, 0x0001);  //Set to Deep Standby mode
    TIMER_Delay(TIMER0, 10);
}

/**
 * @brief  Exit standby mode
 * @param none
 * @return none
 */
void OTM3225A_ExitStandby(void)
{
    OTM3225A_RegWrite(0x0010, 0x1690);
    TIMER_Delay(TIMER0, 10);
    OTM3225A_RegWrite(0x0007, 0x0133);  //Display on
    TIMER_Delay(TIMER0, 10);
}

/**
 * @brief  Set cursor to x and y
 * @param startx x position
 * @param starty y position
 * @return none
 */
void OTM3225_window_start(uint16_t startx, uint16_t starty)
{
    uint16_t tmp;
    if(g_otm3225a_resolution==RES_LANDSCAPE_320X240)
    {
        tmp = starty;
        starty = startx;
        startx = tmp;
    }

    OTM3225A_RegWrite(0x0020, startx);
    OTM3225A_RegWrite(0x0021, starty);
2.6 Drawing Functions

Built-in drawing functions in sample code. User can modify these functions if different applications request.

```c
/**
 * @brief Display a 32x32 ascii character
 * @param x x position
 * @param y y position
 * @param ascii ascii value
 * @param color character color (16 bits, RGB565)
 * @return none
 */
void Show_Word32x32(uint16_t x, uint16_t y, uint8_t ascii, uint16_t color)
{
    int row=0, column=0, bit=0;
    int dot=0;

    for(column=0; column<32; column++)
    {
        OTM3225_window_start(x, y);
        for(row=0; row<4; row++)
        {
            for(bit=1; bit<=8; bit++)
            {
                if( (zk_table[ascii][dot] & 0x80) && (bit==1))
                    OTM3225A_DataWrite(color);
                else if( (zk_table[ascii][dot] & 0x40) && (bit==2))
                    OTM3225A_DataWrite(color);
                else if( (zk_table[ascii][dot] & 0x20) && (bit==3))
                    OTM3225A_DataWrite(color);
                else if( (zk_table[ascii][dot] & 0x10) && (bit==4))
                    OTM3225A_DataWrite(color);
                else if( (zk_table[ascii][dot] & 0x08) && (bit==5))
                    OTM3225A_DataWrite(color);
                else if( (zk_table[ascii][dot] & 0x04) && (bit==6))
                    OTM3225A_DataWrite(color);
                else if( (zk_table[ascii][dot] & 0x02) && (bit==7))
                    OTM3225A_DataWrite(color);
            }
        }
    }
}
```
else if( (zk_table[ascii][dot] & 0x01) && (bit==8))
    OTM3225A_DataWrite(color);
else
    OTM3225A_DataWrite(0x0000);//background color
OTM3225_window_start(x+(row*8)+bit, y);
}

dot++;
/**
 * @brief Clear screen to specific color
 * @param color fill with this color (16 bits, RGB565)
 * @return none
 */
void ClearScreen(uint16_t color)
{
    uint16_t xaxis, yaxis;

    OTM3225_window_start(0, 0);
    for(xaxis=0; xaxis<320; xaxis++)
    {
        for(yaxis=0; yaxis<240; yaxis++)
        {
            OTM3225A_DataWrite(color);
        }
    }
}

/**
 * @brief Draw at partial area of screen
 * @param x x position
 * @param y y position
 * @param width width of area
 * @param height height of area
 * @param color color of area (16 bits, RGB565)
 * @return none
 */
void DrawPartialScreen(uint16_t x, uint16_t y, uint16_t width, uint16_t height, uint16_t color)
{
    uint16_t xaxis, yaxis;

    for(yaxis=0; yaxis<height; yaxis++)
    {
        OTM3225_window_start(x, y++);
        for(xaxis=0; xaxis<width; xaxis++)
        {
            OTM3225A_DataWrite(color);
        }
    }
}
2.7 Software Definitions

Some definitions used in sample code.

<table>
<thead>
<tr>
<th>Definition</th>
<th>Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID_READ_ADDR</td>
<td>0x0000</td>
</tr>
<tr>
<td>DRIVER_OUT_CON_1_ADDR</td>
<td>0x0001</td>
</tr>
<tr>
<td>LCD_DRV_WAVEFORM_CON_ADDR</td>
<td>0x0002</td>
</tr>
<tr>
<td>ENTRY_MODE_ADDR</td>
<td>0x0003</td>
</tr>
<tr>
<td>SCALING_CON_ADDR</td>
<td>0x0004</td>
</tr>
<tr>
<td>DISPLAY_CON_1_ADDR</td>
<td>0x0007</td>
</tr>
<tr>
<td>DISPLAY_CON_2_ADDR</td>
<td>0x0008</td>
</tr>
<tr>
<td>DISPLAY_CON_3_ADDR</td>
<td>0x0009</td>
</tr>
<tr>
<td>FRAME_CYCLE_CON_ADDR</td>
<td>0x000A</td>
</tr>
<tr>
<td>EXT_DIS_INTERFACE_CON_1_ADDR</td>
<td>0x000C</td>
</tr>
<tr>
<td>FRAME_MARKER_POS_ADDR</td>
<td>0x000D</td>
</tr>
<tr>
<td>EXT_DIS_INTERFACE_CON_2_ADDR</td>
<td>0x000F</td>
</tr>
<tr>
<td>POWER_CON_1_ADDR</td>
<td>0x0010</td>
</tr>
<tr>
<td>POWER_CON_2_ADDR</td>
<td>0x0011</td>
</tr>
<tr>
<td>POWER_CON_3_ADDR</td>
<td>0x0012</td>
</tr>
<tr>
<td>POWER_CON_4_ADDR</td>
<td>0x0013</td>
</tr>
<tr>
<td>GRAM_ADDR_SET_HOR_ADDR</td>
<td>0x0020</td>
</tr>
<tr>
<td>GRAM_ADDR_SET_VER_ADDR</td>
<td>0x0021</td>
</tr>
<tr>
<td>GRAM_DATA_WR_ADDR</td>
<td>0x0022</td>
</tr>
<tr>
<td>POWER_CON_7_ADDR</td>
<td>0x0029</td>
</tr>
<tr>
<td>FRAMERATE_CON_ADDR</td>
<td>0x002B</td>
</tr>
<tr>
<td>GAMMA_CON_1_ADDR</td>
<td>0x0030</td>
</tr>
<tr>
<td>GAMMA_CON_2_ADDR</td>
<td>0x0031</td>
</tr>
<tr>
<td>GAMMA_CON_3_ADDR</td>
<td>0x0032</td>
</tr>
<tr>
<td>GAMMA_CON_4_ADDR</td>
<td>0x0035</td>
</tr>
<tr>
<td>GAMMA_CON_5_ADDR</td>
<td>0x0036</td>
</tr>
<tr>
<td>GAMMA_CON_6_ADDR</td>
<td>0x0037</td>
</tr>
<tr>
<td>GAMMA_CON_7_ADDR</td>
<td>0x0038</td>
</tr>
<tr>
<td>GAMMA_CON_8_ADDR</td>
<td>0x0039</td>
</tr>
<tr>
<td>GAMMA_CON_9_ADDR</td>
<td>0x003C</td>
</tr>
<tr>
<td>GAMMA_CON_10_ADDR</td>
<td>0x003D</td>
</tr>
<tr>
<td>WINDOW_HOR_RAM_START_ADDR</td>
<td>0x0050</td>
</tr>
<tr>
<td>WINDOW_HOR_RAM_END_ADDR</td>
<td>0x0051</td>
</tr>
<tr>
<td>WINDOW_VER_RAM_START_ADDR</td>
<td>0x0052</td>
</tr>
<tr>
<td>WINDOW_VER_RAM_END_ADDR</td>
<td>0x0053</td>
</tr>
<tr>
<td>GATE_DRIVER_SCAN_ADDR</td>
<td>0x0060</td>
</tr>
<tr>
<td>DRIVER_OUT_CON_2_ADDR</td>
<td>0x0061</td>
</tr>
</tbody>
</table>
#define VERTICAL_SCROLL_CON_ADDR (0x006A)
#define DIS_POS_PARTIAL_DIS_1_ADDR (0x0080)
#define RAM_ADDR_START_PARTIAL_1_ADDR (0x0081)
#define RAM_ADDR_END_PARTIAL_1_ADDR (0x0082)
#define DIS_POS_PARTIAL_DIS_2_ADDR (0x0083)
#define RAM_ADDR_START_PARTIAL_2_ADDR (0x0084)
#define RAM_ADDR_END_PARTIAL_2_ADDR (0x0085)
#define PANEL_INTERFACE_CON_1_ADDR (0x0090)
#define PANEL_INTERFACE_CON_2_ADDR (0x0092)
#define PANEL_INTERFACE_CON_3_ADDR (0x0095)
#define PANEL_INTERFACE_CON_4_ADDR (0x0097)

#define RS_HIGH PB->DOUT |= (1<<6); // RS = HIGH
#define RS_LOW PB->DOUT &= ~(1<<6); // RS = LOW

#define CS_HIGH PB->DOUT |= (1<<7); // CS = HIGH
#define CS_LOW PB->DOUT &= ~(1<<7); // CS = LOW

#define RST_HIGH PD->DOUT |= (1<<8); // RESET = HIGH
#define RST_LOW PD->DOUT &= ~(1<<8); // RESET = LOW

/* Select resolution type */
typedef enum {
    RES_PORTRAIT_240X320 = 0,
    RES_LANDSCAPE_320X240 = 1
} E_RES_SIZE;

/* EBI Write/Read address */
#define EBIAccessWrite(value) (*((volatile unsigned char *)(EBI_BASE_ADDR)) = value)
#define EBIAccessRead() (*((volatile unsigned char *)(EBI_BASE_ADDR)))
3 Appendix

The following figure is a schematic diagram showing the connection between a LCD module and the Nano130 learning board.
## Revision History

<table>
<thead>
<tr>
<th>Date</th>
<th>Revision</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>2015.06.03</td>
<td>1.00</td>
<td>Initially issued.</td>
</tr>
</tbody>
</table>
Important Notice

Nuvoton Products are neither intended nor warranted for usage in systems or equipment, any malfunction or failure of which may cause loss of human life, bodily injury or severe property damage. Such applications are deemed, “Insecure Usage”.

Insecure usage includes, but is not limited to: equipment for surgical implementation, atomic energy control instruments, airplane or spaceship instruments, the control or operation of dynamic, brake or safety systems designed for vehicular use, traffic signal instruments, all types of safety devices, and other applications intended to support or sustain life.

All Insecure Usage shall be made at customer’s risk, and in the event that third parties lay claims to Nuvoton as a result of customer’s Insecure Usage, customer shall indemnify the damages and liabilities thus incurred by Nuvoton.