AN12579

Using SLCD controller on K32L2B3 MCU

Rev. 0 — 12/2019 Application Note

Contents

1 Introduction

We can find customized Segment Liquid Crystal Displays (SLCD) technologies everywhere. For example:

- in products that measure the PH level of swimming pools.
- monitors used to measure specific gases in a mine.
- in thermometers used to see if a child is running a fever.

SLCD is one of the oldest display technologies. It is still one of the most popular technologies, due to the lowest price and power consumption.

Segment LCD displays, also called static displays or glass-only displays, consist of two pieces of Indium Tin Oxide (ITO) glass with a twisted nematic fluid sandwiched in between. A static display is a segment display with one pin for each segment. A segment is any line, dot, or symbol that can be turned on and off independently.

NXP K32L2B3 MCU integrates an SLCD controller module with up to eight backplanes and 47 frontplanes, such as, 8 × 47 or 4 × 51. This document describes the usage of on-chip SLCD controller by enabling an SLCD device, S401M16KR, which is a four-digit 0.17" seven segment custom LCD panel.

2 Hardware

2.1 S401M16KR SLCD device

S401M16KR SLCD device contains four digits displayed on the panel. Each digit is shown with seven segments and one dot or colon, as shown in Figure 1.

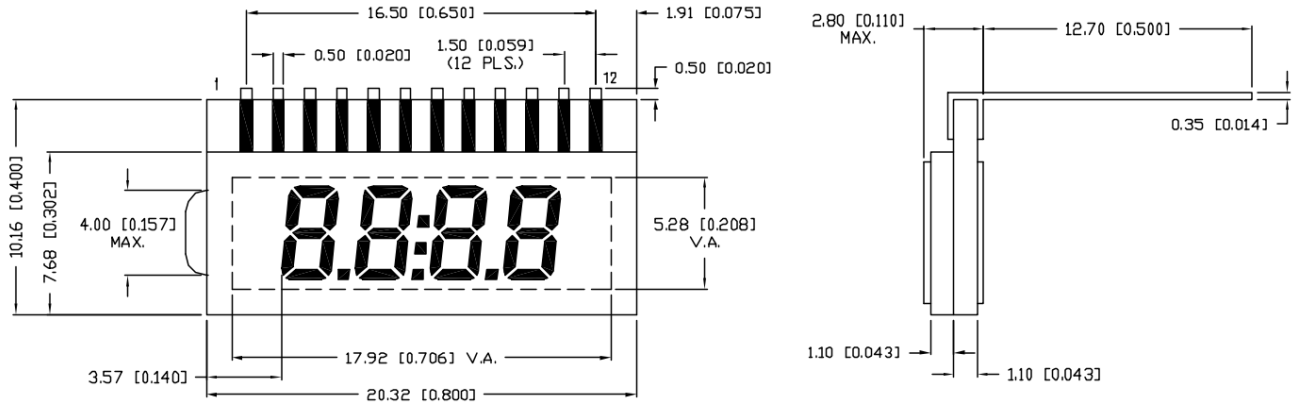

Figure 1. S401M16KR SLCD device

S401M16KR SLCD device contains four COM pins and eight DATA pins as the control signals. COM pins and data pins control a matrix indicating which segments is ON and others are OFF at a specific time, as shown in Figure 2.

PIN	1	2	3	4	5	6	7	8	9	10	11	12
COM0	COM0	/	/	/	1D	DP1	2D	DP2	3D	DP3	4D	COL
COM1	/	COM1	/	/	1E	1C	2E	2C	3E	3C	4E	4C
COM2	/	/	COM2	/	1G	1B	2G	2B	3G	3B	4G	4B
COM3	/	/	/	COM3	1F	1A	2F	2A	3F	3A	4F	4A

Figure 2. S401M16KR SLCD device

COM pins are enabled one by one for each step. In each step, activated by their own COM pin, the eight data pins are outputting the control level signals to turn on and off the segments. The segments for each COM are ON and OFF line by line. Once the cycle with the four steps runs quickly, some segments are seen ON together, as a whole displaying view (even they are not in the same line in the matrix).

Considering the controlling signals as an activating matrix, see Table 1.

Table 1. Activating metrix for controlling signals

nCS	D0	D1	D2	D3	D4	D5	D6	D7
COM0	1D	1DP	2D	2DP	3D	3DP	4D	4DP
COM1	1E	1C	2E	2C	3E	3C	4E	4C
COM2	1G	1B	2G	2B	3G	3B	4G	4B
COM3	1F	1A	2F	2A	3F	3A	4F	4A

For each digit position, different numbers are assembled by various segments. Figure 3 shows 0-9 numbers in the direct segment way.

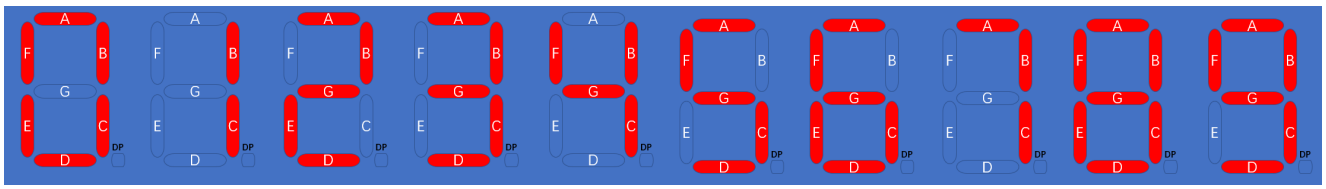

Figure 3. SLCD digits

Table 2 described the related information.

Table 2. SLCD digit information

Number	Segment	COM0 (.D)	COM1 (CE)	COM2 (BG)	COM3 (AF)
0	ABCDEF	*D (0 × 1)	CE (0 × 3)	B* (0 × 2)	AF (0 × 3)
1	BC	*D (0 × 1)	CE (0 × 3)	B* (0 × 2)	AF (0 × 3)
2	ABDEG	*D (0 × 1)	CE (0 × 3)	B* (0 × 2)	AF (0 × 3)
3	ABCDG	*D (0 × 1)	CE (0 × 3)	B* (0 × 2)	AF (0 × 3)
4	BCFG	*D (0 × 1)	CE (0 × 3)	B* (0 × 2)	AF (0 × 3)

Table continues on the next page...

Table 2. SLCD digit information (continued)

Number	Segment	COM0 (.D)	COM1 (CE)	COM2 (BG)	COM3 (AF)
5	ACDFG	*D (0 × 1)	CE (0 × 3)	B* (0 × 2)	AF (0 × 3)
6	ACDEFG	*D (0 × 1)	CE (0 × 3)	B* (0 × 2)	AF (0 × 3)
7	ABC	*D (0 × 1)	CE (0 × 3)	B* (0 × 2)	AF (0 × 3)
8	ABCDEFG	*D (0 × 1)	CE (0 × 3)	B* (0 × 2)	AF (0 × 3)
9	ABCDFG	*D (0 × 1)	CE (0 × 3)	B* (0 × 2)	AF (0 × 3)
None	—	*D (0 × 1)	CE (0 × 3)	B* (0 × 2)	AF (0 × 3)
Dot	DP	*D (0 × 1)	CE (0 × 3)	B* (0 × 2)	AF (0 × 3)

As shown in Table 2, we can get the code to show different numbers for each period activated by indicated COMx pin.

We have the code array in source file:

```
#define SLCD_ON_SHOW_COUNT 11u

const uint8_t SLCD_NUMBER_TABLE[][SLCD_COMx_COUNT] =
{
    /* COM0, COM1, COM2, COM3 */
    { 0x1, 0x3, 0x2, 0x3 }, /* SLCD_ON_SHOW_NUMBER_0 */
    { 0x0, 0x2, 0x2, 0x0 }, /* SLCD_ON_SHOW_NUMBER_1 */
    { 0x1, 0x1, 0x3, 0x2 }, /* SLCD_ON_SHOW_NUMBER_2 */
    { 0x1, 0x2, 0x3, 0x2 }, /* SLCD_ON_SHOW_NUMBER_3 */
    { 0x0, 0x2, 0x3, 0x1 }, /* SLCD_ON_SHOW_NUMBER_4 */
    { 0x1, 0x2, 0x1, 0x3 }, /* SLCD_ON_SHOW_NUMBER_5 */
    { 0x1, 0x3, 0x1, 0x3 }, /* SLCD_ON_SHOW_NUMBER_6 */
    { 0x0, 0x2, 0x2, 0x2 }, /* SLCD_ON_SHOW_NUMBER_7 */
    { 0x1, 0x3, 0x3, 0x3 }, /* SLCD_ON_SHOW_NUMBER_8 */
    { 0x1, 0x2, 0x3, 0x3 }, /* SLCD_ON_SHOW_NUMBER_9 */
    { 0x0, 0x0, 0x0, 0x0 }, /* SLCD_ON_SHOW_NONE */
    { 0x2, 0x0, 0x0, 0x0 }, /* SLCD_ON_SHOW_DP */
};
```

NOTE

The array only contains two pins for each digit. A four-digit parallel can be extended duplicately with eight pins. The following section describes the usage of four digits.

2.2 FRDM-K32L2B3 board

On the FRDM-K32L2B3 board, the SLCD device is connected to the MCU with the pins. The schematic is as shown in Figure 4.

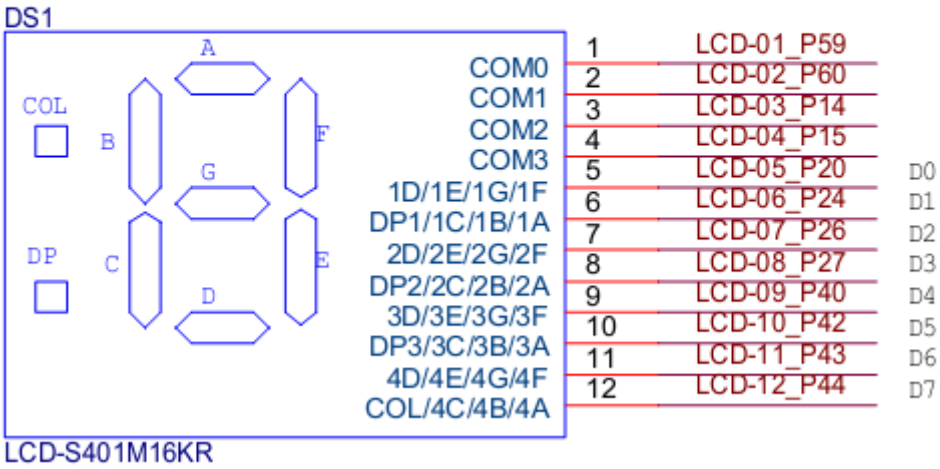

Figure 4. SLCD connections

Table 3 shows the functional information about pin settings.

Table 3. Functional information about pin settings

Functional ID	SLCD pin	MCU pin	ALT	Comments
LCD-01	P59	PTE20	ALT0	COM0
LCD-02	P60	PTE21	ALT0	COM1
LCD-03	P14	PTB18	ALT0	COM2
LCD-04	P15	PTB19	ALT0	COM3
LCD-05	P20	PTC0	ALT0	D0
LCD-06	P24	PTC4	ALT0	D1
LCD-07	P26	PTC5	ALT0	D2
LCD-08	P27	PTC6	ALT0	D3
LCD-09	P40	PTD0	ALT0	D4
LCD-10	P42	PTD2	ALT0	D5
LCD-11	P43	PTD3	ALT0	D6
LCD-12	P44	PTD4	ALT0	D7

Considered as a bus, the data signals are assembled by eight separated pins, named from **D0** to **D7**. The coding is performed to control these signals and the signals from each pin are seen as a whole data through the bus.

To operate all the control signals like a bus, the signal indexes are arranged into two arrays in the source code:

```
/* Define the sync bus and the data bus. */
#define SLCD_COMx_COUNT 4u
#define SLCD_DATA_BUS_WIDTH 8u
  /* Define the pins for sync bus and data bus. */
```

```
const uint8_t SLCD_PIN_COMx[SLCD_COMx_COUNT] =
{
    59, /* COM0. */
    60, /* COM1. */
    14, /* COM2. */
    15 /* COM3. */
};
const uint8_t SLCD_PIN_DATA[SLCD_DATA_BUS_WIDTH] =
{
    20, /* D0. */
    24, /* D1. */
    26, /* D2. */
    27, /* D3. */
    40, /* D4. */
    42, /* D5. */
    43, /* D6. */
    44 /* D7. */
    };
```

3 Basic usgae

The SLCD controller on K32L2B is easy to use. After enabling clock and setting up pin mux functions, for the basic usage without the blink and fault detection feature, only one control register, LCD General Control Register (LCD_GCR), is necessary for initializing the controller. The following provides a group of typical settings:

```
      /* Setup slcd controller. */
LCD->GCR = LCD_GCR_DUTY(3)      /* Selects the duty cycle of the LCD controller driver. 3: 4 COMx
lines. */
         | LCD_GCR_LCLK(2)      /* Clock divider for clock source. 0-7 */
         | LCD_GCR_SOURCE(0)    /* LCD clock source. 1:use MCGIRCLK. 0:OSC32K */
         | LCD_GCR_LCDEN(0)     /* Disable the controller during setting. */
         | LCD_GCR_LCDSTP(0)    /* Keep LCD module alive in STOP modes. */
         | LCD_GCR_LCDDOZE(1)   /* Keep LCD module alive in DOZE mode. */
         | LCD_GCR_FFR(0)       /* Select the frame rate mode. 0:standard frame rate. */
         | LCD_GCR_ALTSOURCE(0) /* Select the alternate clock source. no available when using default
clock source.*/
         | LCD_GCR_ALTDIV(0)    /* Clock divider for alternate clock source. no available when using
default clock.*/
         | LCD_GCR_FDCIEN(0)    /* Enables an LCD interrupt event when fault detection is completed.
*/
         | LCD_GCR_PADSAFE(0)   /* Force safe state on LCD pad control, locking all LCD control bits.
*/
         | LCD_GCR_VSUPPLY(0)   /* Select the power voltage supply. 0: from internal Vdd. */
         | LCD_GCR_LADJ(1)      /* Configures SLCD to handle different LCD glass capacitance.*/
         | LCD_GCR_CPSEL(1)     /* Selects the LCD controller charge pump or a resistor network to
supply the LCD voltages V_LLx. */
         | LCD_GCR_RVTRIM(8)    /* Regulated Voltage Trim. no available when disabled.*/
         | LCD_GCR_RVEN(0)      /* Regulated Voltage Enable. disabled. */
        ;
```

The pins on the MCU are required to be mapped to the SLCD control bus for COMx signals and Dx signals.

- The mappings of the COMx signals are configured as the back panel pins.
- The mappings of the Dx signals are configured as the front panel pins.
- The used pins are initialized with the LCD Pin Enable registers (LCD_PEN0, LCD_PEN1), LCD Back Plane Enable registers (LCD_BPEN0, LCD_BPEN1), while LCD_PENx enables all the pins in use and LCD_BPENx selects them as front panel or back panel.

- LCD_WF8Bx registers are for signal timing sequence of each pin.

The following describes the usage of LCD_WF8Bx registers:

- Each register in the LCD_WF8Bx array is for one LCD signal pin. The index of the array is also for the functional pin of the SLCD module. For example, LCD_WF8B[59] is for the signal pin of the SLCD module, LCD_P59.
- Each bit in the LCD_WF8Bx register is for Step 1, while the index of bit is also for Step 1. For example, the bit **2** in LCD_WF8B[59] responds to the believer of LCD_P59 when in Step 2 of the whole cycle (including four steps or eight steps).

In the software, controlling the data signals is a little different from the hardware. The software searches the data for the parallel pins first, and then arrange the data timing sequence. However, the hardware searches the data timing for each pin first, and then assemble the parallel pins into bus with 8-bit width. Therefore, a converting function is designed in the source code project.

```
/**
* @brief Set the data on SLCD control bus
* @param com_idx The index of step (COMx), 0-3.
* @param show_dat The display code to the bus for current step.
*/
void slcd_set_bus_data(uint8_t com_idx, uint8_t show_dat)
{
    uint8_t bit_mask = (1u << com_idx);
    for (uint8_t i = 0u; i < SLCD_DATA_BUS_WIDTH; i++)
    {
        if (show_dat & 0x1)
       {
            LCD->WF8B[SLCD_PIN_DATA[i]] |= bit_mask;
        }
        else
        {
            LCD->WF8B[SLCD_PIN_DATA[i]] = ~bit_mask;
        }
         show_dat >>= 1u;
     }
 }
```

An API function is created to assemble the segment codes into the displaying matrix for the four digits. With this API, the complex matrix conversion is not required. You just need to tell the MCU which number you want to show and in which position you want it to be, as the software handles all the conversion automatically.

```
/* keep the unchanged displaying code in the matrix. */
static uint8_t slcd_on_show_numbers[SLCD_COMx_COUNT];

/**
* @brief Set the displaying number in the digital position of SLCD device.
* @param index The index of digital position, 0-3.
* @param number The value of showing number, 0-10, while 10 is "none".
* @param en_dp Enable showing the dop in current digital positon, true or false.
*/
void slcd_set_number(uint8_t index, uint8_t number, bool en_dp)
{
    uint8_t tmp8 = 0u;
    for (uint8_t i = 0u; i < SLCD_COMx_COUNT; i++)
    {
        tmp8 = slcd_on_show_numbers[i] & (~(0x3 << (2 * index))); /* clear old setting code.*/
        tmp8 |= (SLCD_NUMBER_TABLE[number][i] << (2 * index)); /* add new setting code. */
        if (en_dp)
        {
            tmp8 |= SLCD_NUMBER_TABLE[SLCD_ON_SHOW_NUMBER_DP][i] << (2 * index); /* add new setting
```

```
for dot point. */
          }
          slcd_on_show_numbers[i] = tmp8;
          slcd_set_bus_data(i, slcd_on_show_numbers[i]);
      }
}
```

In the application `main()` function, the source code to show the changing numbers on the target SLCD is as below:

```
int main(void)
{
    bool en_dp;

    /* init board hardware. */
    BOARD_InitPins();
    BOARD_BootClockRUN();
    BOARD_InitDebugConsole();

    PRINTF("slcd basic example.\r\n");
    /* init the clock and pins for slcd, setup the controller for slcd. */
    slcd_init();

    en_dp = false;
    while (1)
    {
         for (uint8_t i = 0u; i < SLCD_ON_SHOW_COUNT; i++)
         {
             GETCHAR();

             slcd_stop(); /* stop the slcd controller before updating displaying. */
             slcd_set_number(0, i , en_dp);
             slcd_set_number(1, (i+1)%SLCD_ON_SHOW_COUNT, en_dp);
             slcd_set_number(2, (i+2)%SLCD_ON_SHOW_COUNT, en_dp);
             slcd_set_number(3, (i+3)%SLCD_ON_SHOW_COUNT, en_dp);
             slcd_start();
      }
      en_dp = !en_dp;
    }
}
```

Download the project and run it on the FRDM-K32L2B board. The numbers are displayed on the SLCD, as shown in Figure 5.

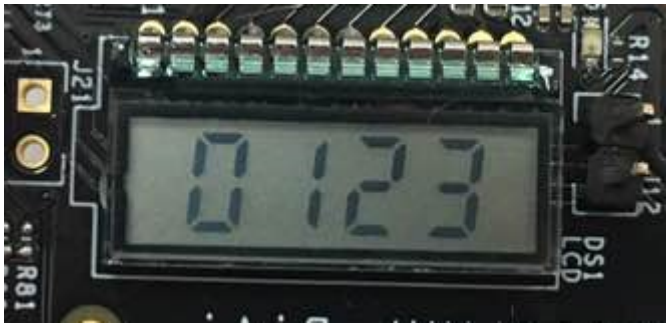

Figure 5. SLCD connections

For the whole runnable source code project, `slcd_basic`, see to the attached code package

4 Usage in low power mode

On the MCU with SLCD controller, the SLCD controller is specially supported by some extra low power STOP modes. In these modes, almost all the hardware are powered off to same energy. The SLCD controller controls its pins to refresh the SLCD device, to keep it displaying on the panel.

The SLCD can work in almost all the power mode except the **VLLS0** mode. For details, refer to the **Power Management** chapter in User Manual.

Table 4. SLCD in low power mode

Modules	VLPR	VLPW	Stop	VLPS	LLS	VLLSx
Segment LCD	FF Async operation in CPO	FF	Async operation FF in PSTOP2	Async operation	Async operation	Async operation, OFF in VLLS0

The critical settings to keep the SLCD working in low power modes are:

- **Clock source**: Make sure that the clock source for SLCD controller is still alive in the target low power mode. For example, if the clock source of the SLCD controller is 32 K OSC, this clock source is enabled in every mode, even low in the VLLSx mode.
- **Pin mux**: Make sure that the used pins for SLCD controller are configured as analog function (ALT0) and the pins of the other (digital) functions are locked (the voltage level can not be changed) in VLLSx mode. Only when the pins of the SLCD controller are active, the SLCD controller outpus the waves and the SLCD panel keeps displaying the digits.
- **The low power mode of the SLCD controller**: Enable the lower power support of the SLCD controller by setting `LCD_GCR[LCDSTP]` bit and `LCD_GCR[LCDDOZE]` bit as 0s, to keep the SLCD controller still working in STOP and WAIT modes.

After completing the parameters above, set the SLCD controller for displaying and then enter the lower power STOP mode. In the lower power STOP modes, the displaying on the SLCD panel is ON, as the SLCD controller is still outputting the refresh waves.

In the example project shown in Figure 6, in the VLLS3 mode, with RTC and SLCD controller running, the measuring current on FRDM-K32L2B board is as low as about 7 uA.

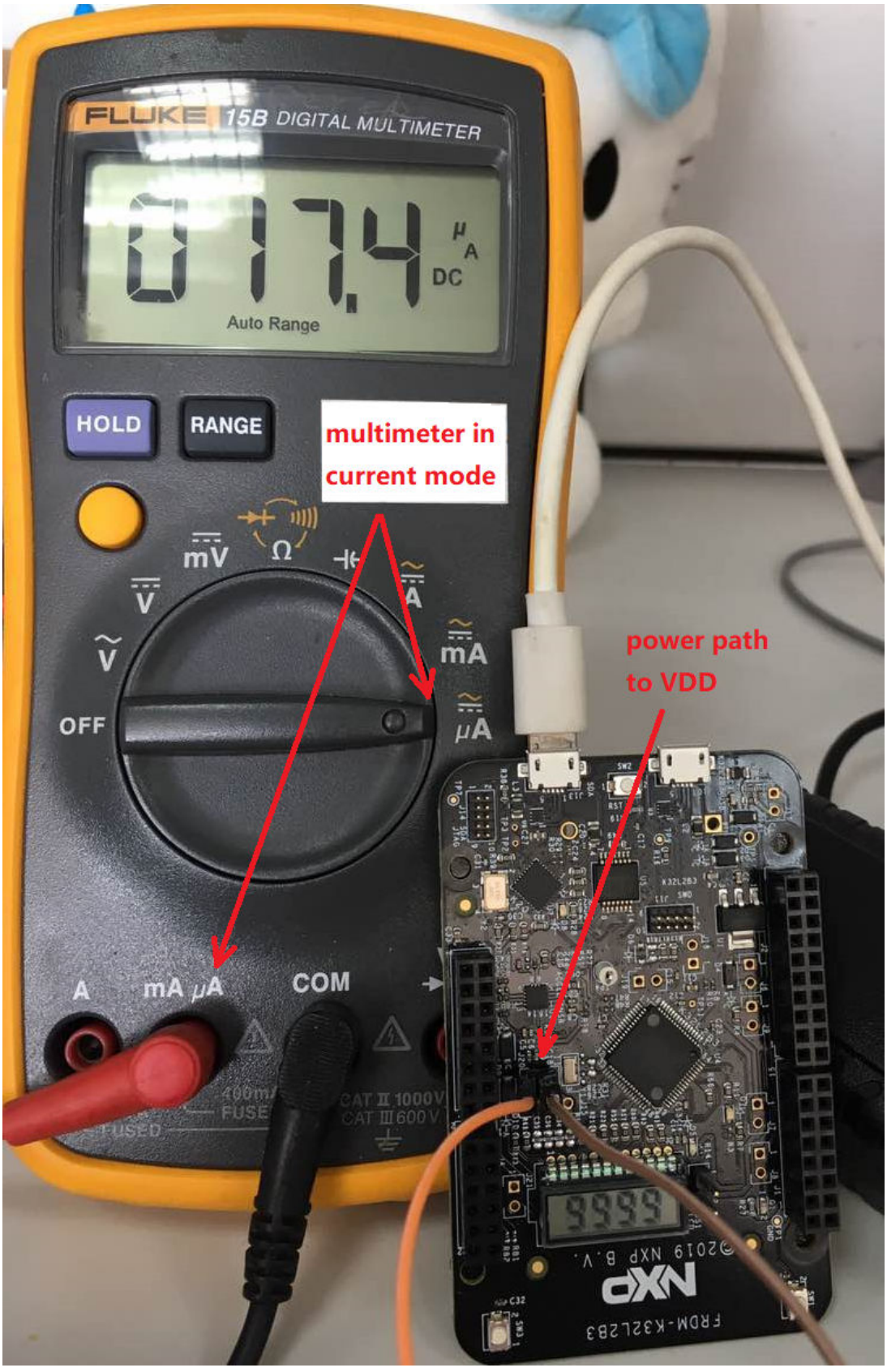

Figure 6. SLCD low power display

For the whole runnable source code project, `slcd_low_power`, see the attached code package.

5 Conclusion

This document descibes basic usages of on-chip SLCD controller on K32L2B MCU, with the example projects based on the FRDM-K32L2B board. The SLCD controller can control the SLCD device to display the contents on its panel automatically with suitable configurations on the hardware. Even in the low power modes, the SLCD controller can still work with very low energy. It indicates that the K32L2B with on-chip SLCD controller can be used in the energy-sensitive application field.

6 References

- https://focuslcds.com/segment-lcd/

How To Reach Us

Home Page:

nxp.com

Web Support:

nxp.com/support

Information in this document is provided solely to enable system and software implementers to use NXP products. There are no express or implied copyright licenses granted hereunder to design or fabricate any integrated circuits based on the information in this document. NXP reserves the right to make changes without further notice to any products herein.

NXP makes no warranty, representation, or guarantee regarding the suitability of its products for any particular purpose, nor does NXP assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. “Typical” parameters that may be provided in NXP data sheets and/or specifications can and do vary in different applications, and actual performance may vary over time. All operating parameters, including “typicals,” must be validated for each customer application by customer's technical experts. NXP does not convey any license under its patent rights nor the rights of others. NXP sells products pursuant to standard terms and conditions of sale, which can be found at the following address: nxp.com/SalesTermsandConditions.

While NXP has implemented advanced security features, all products may be subject to unidentified vulnerabilities. Customers are responsible for the design and operation of their applications and products to reduce the effect of these vulnerabilities on customer’s applications and products, and NXP accepts no liability for any vulnerability that is discovered. Customers should implement appropriate design and operating safeguards to minimize the risks associated with their applications and products.

For more information, please visit: http://www.nxp.com
For sales office addresses, please send an email to: salesaddresses@nxp.com

Date of release: 12/2019
Document identifier: AN12579