LCM Test Tool Protocol Proposal

Application Note for 32-bit NuMicro® Family

Document Information

Abstract	The LCM Test Tool is the convenient tool to light on the LCM rapidly. It can export the dedicated text file to integrate with the same driver for different LCM, respectively. This document introduces the tool USB vendor command protocol and the exported text file contents in detail..
Apply to	N9H2x Series

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

www.nuvoton.com

Table of Contents

1 Introduction

The download tool protocol is based on USB Mass Storage Class command. It will follow the USB mass storage standard protocol.

1.1 Test Items for Startup

1.2 Mass Storage Flow

The Command/Data/Statue Flow (Figure 1) shows the flow for Command Transport, Data-In, Data-Out and Status Transport.

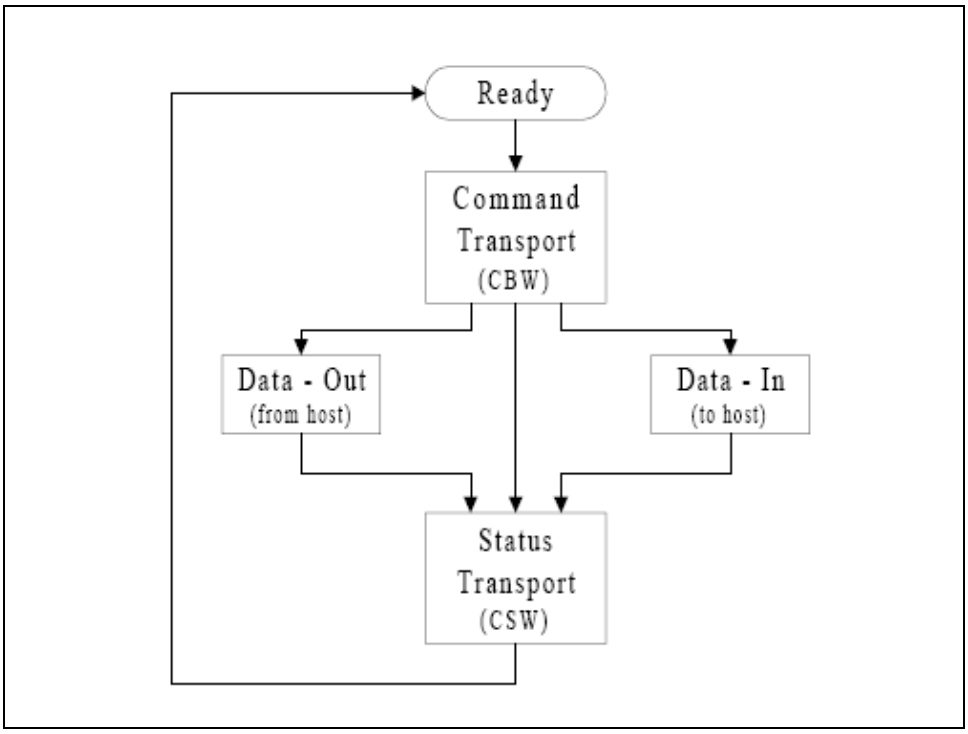


Figure 1 – Command/Data/Status Flow

1.3 Command Block Wrapper (CBW)

The CBW shall start on a packet boundary and shall end as a short packet with exactly 31 (1Fh) bytes transferred. The vendor command will follow the CBW definition.

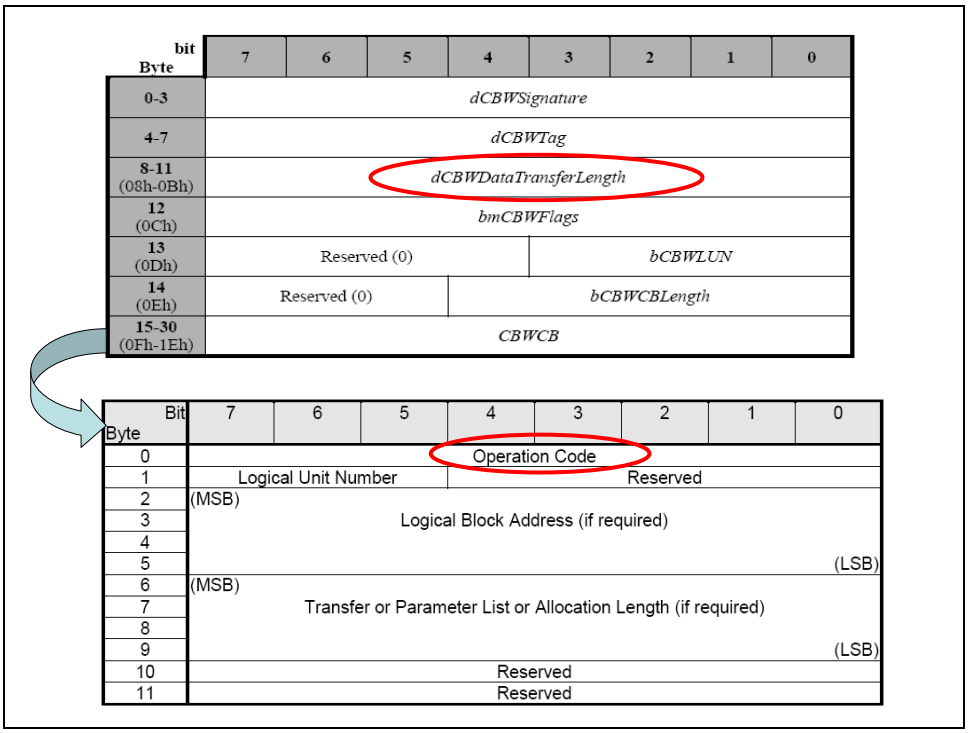
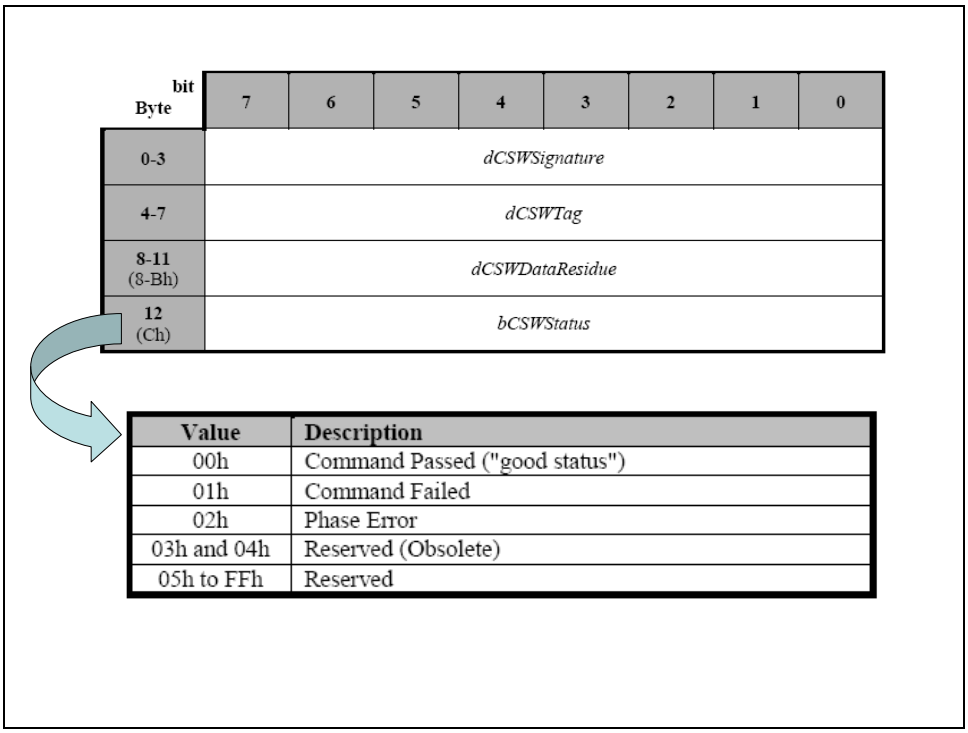


Figure 2 – Command Block Wrapper

1.4 Command Status Wrapper (CSW)

The CSW shall start on a packet boundary and shall end as a short packet with exactly 13 (0Dh) bytes transferred.



bit / Byte	7	6	5	4	3	2	1	0
0-3	*dCSWSignature*							
4-7	*dCSWTag*							
8-11 (8-Bh)	*dCSWDataResidue*							
12 (Ch)	*bCSWStatus*							

Value	Description
00h	Command Passed ("good status")
01h	Command Failed
02h	Phase Error
03h and 04h	Reserved (Obsolete)
05h to FFh	Reserved

Figure 3 – Command Status Wrapper

2 LCM Test Tool Vendor Command

Describe the LCM Test Tool vendor command in detail.

2.1 USB IBR Connect – 0x06 (0x51)

This command is used to connect with IBR USB device. It will disable the timer counter without timeout.

- CBW Data transfer length – 8
- Operation Code – 0x06
- *LUN* – 0x51
- *LBA (high byte)* – 0x01 (Open); 0x02 (Close)

IBR Connect Command

Byte \ Bit	7	6	5	4	3	2	1	0
0 – 3	CBW Signature (0x43425355)							
4 – 7	CBW Tag							
08h – 0Bh	CBW Data Transfer Length							
0Ch	--							
0Dh – 0Eh	--							
0Fh	Operation Code (0x06)							
10h	LUN (0x51)							
11h	0x01							
12h – 15h	--							
16h – 1Eh	--							

2.2 LCM Test Tool Write – 0x63

This command is used to select process functions.

- CBW Data transfer length – transfer length
- Operation Code – 0x63

TurboWriter Write Command

Byte \ Bit	7	6	5	4	3	2	1	0
0 – 3	CBW Signature (0x43425355)							
4 – 7	CBW Tag							
08h – 0Bh	CBW Data Transfer Length							
0Ch	--							
0Dh – 0Eh	--							
0Fh	Operation Code (0x63)							
10h – 14h	--							
15h	--							
16h	--							
17h – 1Eh	--							

2.3 LCM Test Tool Verify – 0x64

This command is used to select process functions.

- CBW Data transfer length – transfer length
- Operation Code – 0x64
-

TurboWriter Verify Command

Byte \ Bit	7	6	5	4	3	2	1	0
0 – 3	CBW Signature (0x43425355)							
4 – 7	CBW Tag							
08h – 0Bh	CBW Data Transfer Length							
0Ch	--							
0Dh – 0Eh	--							
0Fh	Operation Code (0x64)							
10h – 14h	--							
15h	--							
16h	--							
17h – 1Eh	--							

2.4 LCM Send Information – 0x71

This command is used to set LCM related information.

- CBW Data transfer length – structure length
- Operation Code – 0x71
- Parameters0 –
  - 0x00 – LCM Type
  - 0x01 – Resolution (for MPU & Sync LCM)
  - 0x02 – Pixel Clock Frequency (KHz, for Sync LCM)
  - 0x03 – Frame Buffer Data Type (for MPU & Sync LCM)
  - 0x04 – Horizontal Timing (for Sync LCM)
  - 0x05 – Vertical Timing (for Sync LCM)
  - 0x06 – Sync LCM Interface (for Sync LCM)
  - 0x07 – MPU LCM Interface (for MPU LCM)
  - 0x08 – Clock Polarity (for Sync LCM)
  - 0x09 – Sync LCM Configuration Interface Select (for Sync LCM)
  - 0x0A – Sync LCM SPI Configuration (for Sync LCM)
  - 0x0B – Sync LCM I2C Configuration (for Sync LCM)
  - 0x0C – LCM Internal Register Settings (for MPU & Sync LCM)
  - 0x0D – Backlight Control (for MPU & Sync LCM)
  - 0x0E – Reset Control (for MPU & Sync LCM)
  - 0x0F – Hsync Vsync DE Mode (for Sync LCM)
-

LCM Send Information Command

Byte \ Bit	7	6	5	4	3	2	1	0
0 – 3	CBW Signature (0x43425355)							
4 – 7	CBW Tag							
08h – 0Bh	CBW Data Transfer Length							
0Ch	--							
0Dh – 0Eh	--							
0Fh	Operation Code (0x71)							
10h – 14h	--							
15h	Parameter 0							
16h								
17h – 1Eh	--							

LCM Type

```
typedef enum {
        reserved = 0,
        eDRVVPOST_SYNC = 1,         // default value
        eDRVVPOST_MPU = 3
             } E_DRVVPOST_LCM_TYPE;

E_DRVVPOST_LCM_TYPE eLcmType:
```

- 0x01 – Sync Type LCM
- 0x03 – MPU Type LCM

Resolution

```
typedef struct {
        unsigned int  u32PixelPerLine;
        unsigned int  u32LinePerPanel;
             } S_DRVVPOST_LCM_WINDOW;

S_DRVVPOST_LCM_WINDOW sLcmWindow;
```

Pixel Clock Frequency

```
unsigned int    u32PixelClock;          (KHz)
```

Frame Buffer Data Type

```
typedef enum {
        eDRVVPOST_FRAME_RGB555 = 0,
        eDRVVPOST_FRAME_RGB565,
        eDRVVPOST_FRAME_RGBx888,
        eDRVVPOST_FRAME_RGB888x,
        eDRVVPOST_FRAME_CBYCRY,
        eDRVVPOST_FRAME_YCBYCR,
        eDRVVPOST_FRAME_CRYCBY,
        eDRVVPOST_FRAME_YCRYCB
             } E_DRVVPOST_FRAME_DATA_TYPE;

E_DRVVPOST_FRAME_DATA_TYPE eFrameDataType;
```

Horizontal Timing

```
typedef struct {
        unsigned int  u32PulseWidth;    // Horizontal sync pulse width
        unsigned int  u32BackPorch;          // Horizontal back porch
```

```
        unsigned int  u32FrontPorch;          // Horizontal front porch
            } S_DRVVPOST_SYNCLCM_HTIMING;

S_DRVVPOST_SYNCLCM_HTIMING sHTiming;
```

- The above settings are times the cycle of pixel clock.

Vertical Timing

```
typedef struct {
        unsigned char      u8PulseWidth;     // Vertical sync pulse width
        unsigned char      u8BackPorch;           // Vertical back porch
        unsigned char      u8FrontPorch;          // Vertical front porch
            } S_DRVVPOST_SYNCLCM_VTIMING;

S_DRVVPOST_SYNCLCM_VTIMING sVTiming;
```

- The above settings are times the cycle of pixel clock.

Sync LCM Interface

```
typedef enum {
        eDRVVPOST_RGB565 = 0,          // default value
        eDRVVPOST_RGB666,
        eDRVVPOST_RGB888,
        eDRVVPOST_CCIR601,
        eDRVVPOST_RGBDummy,
        eDRVVPOST_CCIR656,
        eDRVVPOST_RGBThrough
            } E_DRVVPOST_SYNCLCM_INTERFACE;

E_DRVVPOST_SYNCLCM_INTERFACE eSyncLcmInterface;
```

MPU LCM Interface

```
typedef enum {
        eDRVVPOST_MPU_8_8 = 0,
        eDRVVPOST_MPU_2_8_8,
        eDRVVPOST_MPU_6_6_6,
        eDRVVPOST_MPU_8_8_8,
        eDRVVPOST_MPU_9_9,
        eDRVVPOST_MPU_16,
        eDRVVPOST_MPU_16_2,
```

```
        eDRVVPOST_MPU_2_16,
        eDRVVPOST_MPU_16_8,
        eDRVVPOST_MPU_18,
        eDRVVPOST_MPU_18_6,
        eDRVVPOST_MPU_24
            } E_DRVVPOST_MPULCM_DATABUS;

E_DRVVPOST_MPULCM_DATABUS eMpuLcmInterface;
```

Clock Polarity

```
typedef struct {
        BOOL bHClock;                   // Hsync polarity
        BOOL bVClock;                   // Vsync polarity
        BOOL bDeClock;              // DE polarity
        BOOL bPixelClock;               // pixel clock polarity
            } S_DRVVPOST_CLK_POLARITY;

S_DRVVPOST_CLK_POLARITY sClockPolarity;
```

BOOL bHClock

- 0x00 – clock active low
- 0x01 – clock active high

BOOL bVClock

- 0x00 – clock active low
- 0x01 – clock active high

BOOL bDeClock

- 0x00 – clock active low
- 0x01 – clock active high

BOOL bPixelClock

- 0x00 – clock active low
- 0x01 – clock active high

Sync LCM Configuration Interface Select

```
typedef enum {
        eDRVVPOST_SPI = 0,
        eDRVVPOST_I2C_RESERVED,
        eDRVVPOST_NONE
            } E_DRVVPOST_CONFIG_INTERFACE;
```

E_DRVVPOST_CONFIG_INTERFACE eConfigInterface;

- If “eDRVVPOST_NONE” selected, it means that need not configure LCM internal registers.

Sync LCM SPI Configuration

```
typedef enum {
        eDRVVPOST_GPA = 0,
        eDRVVPOST_GPB,
        eDRVVPOST_GPC,
        eDRVVPOST_GPD,
        eDRVVPOST_GPE,
        eDRVVPOST_GPF,
        eDRVVPOST_GPG,
        eDRVVPOST_GPH
            } E_DRVVPOST_PORT_SELECT;

typedef enum {
    eDRVVPOST_PIN_0 = 0,
    eDRVVPOST_PIN_1,
    eDRVVPOST_PIN_2,
    eDRVVPOST_PIN_3,
    eDRVVPOST_PIN_4,
    eDRVVPOST_PIN_5,
    eDRVVPOST_PIN_6,
    eDRVVPOST_PIN_7,
    eDRVVPOST_PIN_8,
    eDRVVPOST_PIN_9,
    eDRVVPOST_PIN_10,
    eDRVVPOST_PIN_11,
    eDRVVPOST_PIN_12,
    eDRVVPOST_PIN_13,
    eDRVVPOST_PIN_14,
    eDRVVPOST_PIN_15
        } E_DRVVPOST_PIN_SELECT;

        typedef enum {
            eDRVVPOST_SPI_WR = 0,
            eDRVVPOST_SPI_RD
                } E_DRVVPOST_SPI_OPERATION;

        typedef struct {
            E_DRVVPOST_PORT_SELECT     eCSPort;              // CS port

            E_DRVVPOST_PIN_SELECT  eCSPin;          // CS pin
```

```
        E_DRVVPOST_PORT_SELECT     eClkPort;              // clock port

        E_DRVVPOST_PIN_SELECT   eClkPin;          // clock pin

        E_DRVVPOST_PORT_SELECT     eDataPort;             // data port

        E_DRVVPOST_PIN_SELECT   eDataPin;              // data pin
        unsigned char      u8SpiType;                  // reserved
setting
        unsigned char      u8RegIndexWidth;            // register index
width, 8/16/24
        unsigned char      u8RegDataWidth;             // register data width,
8/16/24
              } S_DRVVPOST_SPI_CONFIGURE;

S_DRVVPOST_SPI_CONFIGURE sSpiConfigure;
```

Sync LCM I2C Configuration

```
typedef struct {
        unsigned char      u8I2C_ID;                   // I2C ID
address
        E_DRVVPOST_PORT_SELECT     eClkPort;              // clock port

        E_DRVVPOST_PIN_SELECT   eClkPin;          // clock pin

        E_DRVVPOST_PORT_SELECT     eDataPort;             // data port

        E_DRVVPOST_PIN_SELECT   eDataPin;              // data pin
        unsigned char      u8RegIndexWidth;            // register index
width, 8/16/24
        unsigned char      u8RegDataWidth;             // register data width,
8/16/24
              } S_DRVVPOST_I2C_CONFIGURE;

S_DRVVPOST_I2C_CONFIGURE sI2cConfigure;
```

LCM Internal Register Settings

```
typedef struct {
        unsigned int  u32RegIndex;                    // register index
        unsigned int  u32RegData;                  // register data
              } S_DRVVPOST_LCM_REG;

S_DRVVPOST_LCM_REG lcm_reg[n];
```

Backlight Control

```
typedef struct {
        BOOL    bDoNeedEnaPort;       // check whether need to control
backlight ENA pin
                                      // 0: NO, 1: YES
        E_DRVVPOST_PORT_SELECT     eEnaPort;              // Ena port

        E_DRVVPOST_PIN_SELECT  eEnaPin;               // Ena pin
        E_DRVVPOST_PIN_STATE    eEnaStae;
        BOOL    bDoNeedPWMPort;    // check whether need to control
backlight PWM pin
                                      // 0: NO, 1: YES
        E_DRVVPOST_PORT_SELECT     ePWMPort;              // PWM port

        E_DRVVPOST_PIN_SELECT  ePWMPin;               // PWM pin
        E_DRVVPOST_PIN_STATE    ePWMState;
            } S_DRVVPOST_BACKLIGHT_CTRL;

S_DRVVPOST_BACKLIGHT_CTRL sBacklightCtrl;
```

Reset Control

```
typedef struct {
        BOOL    bDoNeedReset;         // check whether need to reset LCM
                                      // 0: NO, 1: YES
        E_DRVVPOST_PORT_SELECT     eResetPort;            // RESET
port
        E_DRVVPOST_PIN_SELECT  eResetPin;             // RESET pin
        E_DRVVPOST_PIN_STATE    eResetStae;
            } S_DRVVPOST_RESET_CTRL;

S_DRVVPOST_RESET_CTRL sResetCtrl;
```

Hsync Vsync De Mode

```
typedef enum {
        eDRVVPOST_HSYNC_VSYNC_DE = 0,// LCM needs Hsync, Vsync
and DE pins.
        eDRVVPOST_HSYNC_VSYNC_ONLY,  // LCM only needs Hsync and
Vsync pins
        eDRVVPOST_DE_ONLY                   // LCM only needs DE pin
            } E_DRVVPOST_HSYNC_VSYNC_DE_MODE;

E_DRVVPOST_HSYNC_VSYNC_DE_MODE eHsyncVsyncDeMode;
```

2.5 LCM Image Information Begin to Write – 0x72

This command is used to begin sending image information (image binary data, for example, RGB565, RGB888, …).

- CBW Data transfer length – transfer length
- Operation Code – 0x72

LCM Image Information Write Command

Byte \ Bit	7	6	5	4	3	2	1	0
0 – 3	CBW Signature (0x43425355)							
4 – 7	CBW Tag							
08h – 0Bh	CBW Data Transfer Length							
0Ch	--							
0Dh – 0Eh	--							
0Fh	Operation Code (0x72)							
10h – 14h	--							
15h	--							
16h	--							
17h – 1Eh	--							

2.6 LCM Image Information Continue to Write – 0x73

This command is used to continue sending image information (image binary data, for example, RGB565, RGB888, …).

- CBW Data transfer length – transfer length
- Operation Code – 0x73

LCM Image Information Conti. Write Command

Byte \ Bit	7	6	5	4	3	2	1	0
0 – 3	CBW Signature (0x43425355)							
4 – 7	CBW Tag							
08h – 0Bh	CBW Data Transfer Length							
0Ch	--							
0Dh – 0Eh	--							
0Fh	Operation Code (0x73)							
10h – 14h	--							
15h	--							
16h	--							
17h – 1Eh	--							

2.7 LCM Run – 0x74

This command is used to begin LCM display.

- CBW Data transfer length – transfer length = 0
- Operation Code – 0x74
-

LCM Run Command

Byte \ Bit	7	6	5	4	3	2	1	0
0 – 3	CBW Signature (0x43425355)							
4 – 7	CBW Tag							
08h – 0Bh	CBW Data Transfer Length (=0)							
0Ch	--							
0Dh – 0Eh	--							
0Fh	Operation Code (0x74)							
10h – 14h	--							
15h	--							
16h	--							
17h – 1Eh	--							

2.8 LCM Stop – 0x75

This command is used to stop LCM display.

- CBW Data transfer length – transfer length = 0
- Operation Code – 0x75
-

LCM Stop Command

Byte \ Bit	7	6	5	4	3	2	1	0
0 – 3	CBW Signature (0x43425355)							
4 – 7	CBW Tag							
08h – 0Bh	CBW Data Transfer Length (=0)							
0Ch	--							
0Dh – 0Eh	--							
0Fh	Operation Code (0x75)							
10h – 14h	--							
15h	--							
16h	--							
17h – 1Eh	--							

2.9 LCM Set Selected Register – 0x76

This command is used to set the selected LCM internal register.

- CBW Data transfer length – transfer length
- Operation Code – 0x76
-

LCM Set Selected Register Command

Byte \ Bit	7	6	5	4	3	2	1	0
0 – 3	CBW Signature (0x43425355)							
4 – 7	CBW Tag							
08h – 0Bh	CBW Data Transfer Length							
0Ch	--							
0Dh – 0Eh	--							
0Fh	Operation Code (0x76)							
10h – 14h	--							
15h								
16h								
17h – 1Eh	--							

Sent data after CBW

E_DRVVPOST_LCM_TYPE eLcmType (1-byte);
E_DRVVPOST_MPULCM_DATABUS eMpuLcmInterface (1-byte);
E_DRVVPOST_CONFIG_INTERFACE eConfigInterface (1-byte);
S_DRVVPOST_SPI_CONFIGURE sSpiConfigure (9-byte);
S_DRVVPOST_I2C_CONFIGURE sI2cConfigure (7-byte);
Reserved (1-byte);
S_DRVVPOST_LCM_REG lcm_reg (8-byte);

2.10 LCM Get Selected Register – 0x77

This command is used to get the selected LCM internal register.

- CBW Data transfer length – transfer length
- Operation Code – 0x77
- Parameters0 –
  - 0x00 – Send the register index and related information
  - 0x01 – Get the register contents.
-

LCM Get Selected Register Command

Byte \ Bit	7	6	5	4	3	2	1	0
0 – 3	CBW Signature (0x43425355)							
4 – 7	CBW Tag							
08h – 0Bh	CBW Data Transfer Length							
0Ch	--							
0Dh – 0Eh	--							
0Fh	Operation Code (0x77)							
10h – 14h	--							
15h	Parameter 0							
16h								
17h – 1Eh	--							

Send Register Index

E_DRVVPOST_LCM_TYPE eLcmType (1-byte);
E_DRVVPOST_MPULCM_DATABUS eMpuLcmInterface (1-byte);
E_DRVVPOST_CONFIG_INTERFACE eConfigInterface (1-byte);
S_DRVVPOST_SPI_CONFIGURE sSpiConfigure (9-byte);
S_DRVVPOST_I2C_CONFIGURE sI2cConfigure (7-byte);
Reserved (1-byte);
unsigned int u32RegIndex (4-byte);

Get Register Contents

unsigned int u32RegData (4-byte);

3 Export File

After using the above vendor command, the related LCM settings can be exported to become an initial file. The LCM driver can include this file to light on the LCM smoothly.

3.1 File Contents

The exported file contents will be arranged as follows.

LCM Type – word offset 0x00

This word is the selected LCM type.

- Word offset 0x00 – LCM type

Resolution – word offset 0x01–0x02

These words will record the LCM resolution.

- Word offset 0x00 – pixels per line
- Word offset 0x01 – lines per panel

Pixel Clock Frequency – word offset 0x03

It is the pixel clock frequency.

- Word offset 0x00 – pixel clock frequency

Frame Buffer Data Type – word offset 0x04

This word records the frame buffer data type. For example, RGB555, RGB565, YUV and so on.

- Word offset 0x00 – frame buffer data type

Horizontal Timing – word offset 0x05-0x07

These words will record the horizontal timing.

- Word offset 0x00 – horizontal sync pulse
- Word offset 0x01 – horizontal back porch
- Word offset 0x02 – horizontal front porch

Vertical Timing – word offset 0x08

This word will record the vertical timing.

- Byte offset 0x00 – vertical sync pulse

- Byte offset 0x01 – vertical back porch
- Byte offset 0x02 – vertical front porch

Sync LCM Interface – word offset 0x09

This word records the sync interface. For example, RGB interface (i.e., RGB565, RGB666 and RGB888), CCIR601, CCIR656 and so on.

- Word offset 0x00 – sync LCM interface

MPU LCM Interface – word offset 0x0A

This word records the MPU interface. For example, MPU_8_8, MPU_16 and so on.

- Word offset 0x00 – MPU LCM interface

Clock Polarity – word offset 0x0B

This word is the clock polarity setting.

- Byte offset 0x00 – horizontal clock polarity
- Byte offset 0x01 – vertical clock polarity
- Byte offset 0x02 – DE clock polarity
- Byte offset 0x03 – pixel clock polarity

Configure Interface Select – word offset 0x0C

This word is for the configure interface selection. For example, SPI or I2C or none.

- Word offset 0x00 – configure interface selection

SPI Configure – word offset 0x0D-0x0F

These words are for SPI configuration settings.

- Byte offset 0x00 – port selection for SPI CS pin. 0: GPA, 1: GPB and so on
- Byte offset 0x01 – pin selection for SPI CS pin. 0: pin-0, 1: pin-1 and so on
- Byte offset 0x02 – port selection for SPI CLK pin. 0: GPA, 1: GPB and so on
- Byte offset 0x03 – pin selection for SPI CLK pin. 0: pin-0, 1: pin-1 and so on
- Byte offset 0x04 – port selection for SPI DAT pin. 0: GPA, 1: GPB and so on

- Byte offset 0x05 – pin selection for SPI DAT pin. 0: pin-0, 1: pin-1 and so on
- Byte offset 0x06 – SPI type setting
- Byte offset 0x07 – LCM internal register index width
- Byte offset 0x08 – LCM internal register data width

I2C Configure – word offset 0x10-0x11

These words are for I2C configuration settings.

- Byte offset 0x00 – port selection for I2C CLK pin. 0: GPA, 1: GPB and so on
- Byte offset 0x01 – pin selection for I2C CLK pin. 0: pin-0, 1: pin-1 and so on
- Byte offset 0x02 – port selection for I2C DAT pin. 0: GPA, 1: GPB and so on
- Byte offset 0x03 – pin selection for I2C DAT pin. 0: pin-0, 1: pin-1 and so on
- Byte offset 0x04 – LCM internal register index width
- Byte offset 0x05 – LCM internal register data width

Backlight Control – word offset 0x12-0x13

These words are for backlight control settings.

- Byte offset 0x00 – indicates to control backlight ENA pin or not. 0: NO, 1: YES
- Byte offset 0x01 – port selection for ENA pin. 0: GPA, 1: GPB and so on
- Byte offset 0x02 – pin selection for ENA pin. 0: pin-0, 1: pin-1 and so on
- Byte offset 0x03 – if needs to control ENA pin, it indicates the pin state. 0: LOW, 1: HIGH
- Byte offset 0x04 – indicates to control backlight PWM pin or not. 0: NO, 1: YES
- Byte offset 0x05 – port selection for PWM pin. 0: GPA, 1: GPB and so on
- Byte offset 0x06 – pin selection for PWM pin. 0: pin-0, 1: pin-1 and so on
- Byte offset 0x07 – if needs to control PWM pin, it indicates the pin state. 0: LOW, 1: HIGH

Reset Control – word offset 0x14

These words are for reset control settings.

- Byte offset 0x00 – indicates to control LCM RESET pin or not. 0: NO, 1: YES
- Byte offset 0x01 – port selection for RESET pin. 0: GPA, 1: GPB and so on

- Byte offset 0x02 – pin selection for RESET pin. 0: pin-0, 1: pin-1 and so on
- Byte offset 0x03 – if needs to control RESET pin, it indicates the RESET active state. 0: LOW, 1: HIGH

Hsync Vsync De Mode Select – word offset 0x15

This word indicates that the LCM needs how many sync signals for display.

- Word offset 0x00 – Hsync, Vsync and DE mode selection

Register Number – word offset 0x34

This word indicates how many LCM registers need to be configured.

- Word offset 0x00 – indicates how many LCM registers

Register Contents – word offset 0x35 ~

These words record the configured register contents.

- Word offset 0x00 – LCM register index
- Word offset 0x01 – LCM register data
- Word offset 0x02 – LCM register index
- Word offset 0x03 – LCM register data
- …

The total register number is indicated in the above register number (*word offset 0x16*)

Revision History

Date	Revision	Description
May 10, 2019	1.00	Initially issued.

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.