

SOFTWARE MANUAL

LibAXDVK2(AXDVK2 Support Library)

Version 1.15

20180817



ON Semiconductor®

1. TABLE OF CONTENTS

1. TABLE OF CONTENTS	2
2. Introduction.....	4
3. Evaluation Board Peripherals	5
3.1. libaxlcd.h.....	5
3.1.1. void lcd_init(void)	5
3.1.2. void lcd_portinit(void).....	5
3.1.3. void lcd_setpos(uint8_t v).....	5
3.1.4. void lcd_writestr(const char *ch).....	5
3.1.5. void lcd_writehex16(uint16_t val, uint8_t nrdig, uint8_t flags) void lcd_writehex32(uint32_t val, uint8_t nrdig, uint8_t flags).....	5
3.1.6. void lcd_writenum16(uint16_t val, uint8_t nrdig, uint8_t flags) void lcd_writenum32(uint32_t val, uint8_t nrdig, uint8_t flags).....	5
3.1.7. void lcd_cleardisplay(void)	6
3.1.8. void lcd_clear(uint8_t pos, uint8_t len).....	6
3.1.9. void lcd_waitlong(void) void lcd_waitshort(void)	6
3.1.10. void lcd_writcmd(uint8_t cmd).....	6
3.1.11. void lcd_writedata(uint8_t d).....	6
3.2. libaxlcd2.h.....	6
3.2.1. void lcd2_init(void)	6
3.2.2. void lcd2_portinit(void).....	6
3.2.3. void lcd2_setpos(uint8_t v).....	6
3.2.4. void lcd2_writestr(const char *ch).....	7
3.2.5. void lcd2_writehex16(uint16_t val, uint8_t nrdig, uint8_t flags) void lcd2_writehex32(uint32_t val, uint8_t nrdig, uint8_t flags).....	7
3.2.6. void lcd2_writenum16(uint16_t val, uint8_t nrdig, uint8_t flags) void lcd2_writenum32(uint32_t val, uint8_t nrdig, uint8_t flags).....	7
3.2.7. void lcd2_cleardisplay(void)	7
3.2.8. void lcd2_clear(uint8_t pos, uint8_t len).....	7
3.2.9. void lcd2_tx(uint8_t v)	7
3.2.10. void lcd2_txcmdshort(uint8_t v).....	7
3.2.11. void lcd2_txcmdlong(uint8_t v).....	7
3.2.12. uint8_t lcd2_txbufferize(void).....	7
3.2.13. uint8_t lcd2_txfree(void).....	7
3.2.14. uint8_t lcd2_txidle(void).....	8
3.2.15. void lcd2_wait_txdone(void)	8
3.2.16. void lcd2_wait_txfree(uint8_t v).....	8

AXDVK2 Software Manual

3.2.17.	void lcd2_txadvance(uint8_t idx).....	8
3.2.18.	void lcd2_txpoke(uint8_t idx, uint8_t ch).....	8
3.2.19.	void lcd2_txpokehex(uint8_t idx, uint8_t ch).....	8
3.2.20.	uint8_t lcd2_poll(void).....	8
3.2.21.	LCD2_DEFINE_TXBUFFER(sz)	8
3.3.	libdvk2leds.h libminikitleds.h	9
3.3.1.	led0_state(x) led1_state(x) led2_state(x) led3_state(x)	9
3.3.2.	led0_on() led1_on() led2_on() led3_on()	9
3.3.3.	led0_off() led1_off() led2_off() led3_off()	9
3.3.4.	led0_toggle() led1_toggle() led2_toggle() led3_toggle()	9
4.	Contact Information	10

2. INTRODUCTION

LibAXDVK2 is a convenience library to ease the use of AX-DVK2 Evaluation Board for the AX8052 Microprocessor. It contains the following features:

- Interrupt-driven Liquid Crystal Display access (libaxlcd2.h)
- Legacy polled Liquid Crystal Display access (libaxlcd.h)

LibAXDVK2 is available in source and binary form for SDCC, Keil C51 and IAR ICC.

3. EVALUATION BOARD PERIPHERALS

3.1. LIBAXLCD.H

libmflcd.h contains routines for accessing the alphanumeric 2x16 character liquid crystal display (LCD) on the evaluation board. libmflcd.h routines are blocking and do not require an interrupt. The routines may take a very long time to complete due to the slowness of the LCD.

3.1.1. VOID LCD_INIT(VOID)

This function initializes the interface and resets the LC display and sets it up.

3.1.2. VOID LCD_PORTINIT(VOID)

This function only initializes the interface, but leaves the display alone. It can be used instead of lcd_init() when waking up from sleep or deepsleep while the display was kept powered, and it is undesirable if the display contents change.

3.1.3. VOID LCD_SETPOS(UINT8_T V)

This function positions the LC display write cursor. Top line characters are numbered from left to right from 0x00 to 0x0F, and bottom line characters are numbered from 0x40 to 0x4F. The cursor does not auto-wrap from the top to the bottom line. After calling lcd_setpos(), lcd_waitshort() must be called or an equivalent delay must be kept before accessing the LCD again.

3.1.4. VOID LCD_WRITESTR(CONST CHAR *CH)

This function writes the null terminated C string pointed to by ch to the display at the current write cursor location. \n causes the output to continue at the beginning of the second line.

3.1.5. VOID LCD_WRITEHEX16(UINT16_T VAL, UINT8_T NRDIG, UINT8_T FLAGS) VOID LCD_WRITEHEX32(UINT32_T VAL, UINT8_T NRDIG, UINT8_T FLAGS)

These functions write a hexadecimal number (val) to the display. The number of digits is given by nrdig. Leading characters are filled with zeros. flags is a bitwise or combination of the WRNUM constants documented in the LibMF manual.

3.1.6. VOID LCD_WRITENUM16(UINT16_T VAL, UINT8_T NRDIG, UINT8_T FLAGS) VOID LCD_WRITENUM32(UINT32_T VAL, UINT8_T NRDIG, UINT8_T FLAGS)

These functions write a decimal number (val) to the display. The number of digits is given by nrdig. flags is a bitwise or combination of the WRNUM constants documented in the LibMF manual.

3.1.7. `VOID LCD_CLEARDISPLAY(VOID)`

This function clears the complete LC Display. It must be followed by `lcd_waitlong`, or the equivalent delay, before accessing the LC Display again.

3.1.8. `VOID LCD_CLEAR(UINT8_T POS, UINT8_T LEN)`

This function clears `len` characters starting at position `pos`.

3.1.9. `VOID LCD_WAITLONG(VOID)` `VOID LCD_WAITSHORT(VOID)`

The LC display is very slow processing commands and data. These routines delay the microprocessor by a certain time by busy waiting. One of those routines, or the equivalent delay, should be called after each LC command or character. Which LC commands require the long delay is detailed in the LC display datasheet.

3.1.10. `VOID LCD_WRITECMD(UINT8_T CMD)`

This low level routine writes a command to the display.

3.1.11. `VOID LCD_WRITEDATA(UINT8_T D)`

This low level routine writes a character to the display.

3.2. LIBAXLCD2.H

`libmflcd2.h` contains routines for accessing the alphanumeric 2x16 character liquid crystal display (LCD) on the evaluation board. Unlike the routines from `libmflcd.h`, these routines are interrupt driven and (normally) nonblocking. They provide roughly the same interface as those from `libmflcd.h`, but `libmflcd.h` and `libmflcd2.h` routines must not be intermixed in the same project.

3.2.1. `VOID LCD2_INIT(VOID)`

This function initializes the interface and resets the LC display and sets it up.

3.2.2. `VOID LCD2_PORTINIT(VOID)`

This function only initializes the interface, but leaves the display alone. It can be used instead of `lcd2_init()` when waking up from sleep or deepsleep while the display was kept powered, and it is undesirable if the display contents change.

3.2.3. `VOID LCD2_SETPOS(UINT8_T V)`

This function positions the LC display write cursor. Top line characters are numbered from left to right from 0x00 to 0x0F, and bottom line characters are numbered from 0x40 to 0x4F. The cursor does not auto-wrap from the top to the bottom line.

3.2.4. `VOID LCD2_WRITESTR(CONST CHAR *CH)`

This function writes the null terminated C string pointed to by `ch` to the display at the current write cursor location. `\n` causes the output to continue at the beginning of the second line.

3.2.5. `VOID LCD2_WRITEHEX16(UINT16_T VAL, UINT8_T NRDIG, UINT8_T FLAGS)` `VOID LCD2_WRITEHEX32(UINT32_T VAL, UINT8_T NRDIG, UINT8_T FLAGS)`

These functions write a hexadecimal number (`val`) to the display. The number of digits is given by `nrdig`. Leading characters are filled with zeros. `flags` is a bitwise or combination of the `WRNUM` constants documented in the LibMF manual.

3.2.6. `VOID LCD2_WRITENUM16(UINT16_T VAL, UINT8_T NRDIG, UINT8_T FLAGS)` `VOID LCD2_WRITENUM32(UINT32_T VAL, UINT8_T NRDIG, UINT8_T FLAGS)`

These functions write a decimal number (`val`) to the display. The number of digits is given by `nrdig`. `flags` is a bitwise or combination of the `WRNUM` constants documented in the LibMF manual.

3.2.7. `VOID LCD2_CLEARDISPLAY(VOID)`

This function clears the complete LC Display.

3.2.8. `VOID LCD2_CLEAR(UINT8_T POS, UINT8_T LEN)`

This function clears `len` characters starting at position `pos`.

3.2.9. `VOID LCD2_TX(UINT8_T V)`

`lcd2_tx` writes one character to the LC display.

3.2.10. `VOID LCD2_TXCMDSHORT(UINT8_T V)`

`lcd2_txcmdshort` writes one command to the LC display. The command is expected to terminate within 50µs.

3.2.11. `VOID LCD2_TXCMDLONG(UINT8_T V)`

`lcd2_txcmdlong` writes one command to the LC display. The command is expected to terminate within 1.2ms.

3.2.12. `UINT8_T LCD2_TXBUFFERSIZE(VOID)`

This function returns the transmit buffer size. Buffer size is configurable.

3.2.13. `UINT8_T LCD2_TXFREE(VOID)`

`lcd2_txfree` returns the number of free space for characters in the FIFO.

3.2.14. UINT8_T LCD2_TXIDLE(VOID)

lcd2_txidle returns one if the LCD controller is idle, i.e. all transmit characters and commands have been sent to the LCD controller. This routine can be used to determine whether the microprocessor can enter sleep.

3.2.15. VOID LCD2_WAIT_TXDONE(VOID)

lcd2_wait_txdone blocks until the last character in the FIFO has been sent to the LCD controller.

3.2.16. VOID LCD2_WAIT_TXFREE(UINT8_T V)

lcd2_wait_txfree blocks until the number of free space for characters in the FIFO reaches or exceeds v.

3.2.17. VOID LCD2_TXADVANCE(UINT8_T IDX)

lcd2_txadvance adds the given number of characters at the end of the transmit FIFO. They must have been defined by lcd2_txpoke or lcd2_txpokehex before, otherwise the transmitted characters are undefined.

3.2.18. VOID LCD2_TXPOKE(UINT8_T IDX, UINT8_T CH)

lcd2_txpoke puts a character at the idx'th position after the end of the FIFO. It does not become part of the FIFO. In order to actually transmit the character, lcd2_txadvance must be called.

3.2.19. VOID LCD2_TXPOKEHEX(UINT8_T IDX, UINT8_T CH)

lcd2_txpoke puts a hexadecimal character at the idx'th position after the end of the FIFO. It does not become part of the FIFO. In order to actually transmit the character, lcd2_txadvance must be called.

3.2.20. UINT8_T LCD2_POLL(VOID)

Normally, data is transferred between the FIFO and the hardware using an interrupt handler. Sometimes though, an interrupt handler is undesirable; in this case, lcd2_poll can be called periodically to transfer data between the FIFO and the hardware if available. It returns a bit mask with bit 1 set if a byte was transferred from the FIFO to the hardware.

3.2.21. LCD2_DEFINE_TXBUFFER(SZ)

This macro defines the transmit buffer size. The argument needs to lie between 14 and 256 (inclusive). Note that the argument specifies the total buffer size; one command is unusable due to the design of the buffer pointers, therefore lcd2_txbuffer size will return one command less than the argument. The number of xdata bytes allocated is twice the argument.

3.3. LIBDVK2LEDS.H LIBMINIKITLEDS.H

These convenience headers provide macros to manipulate the LED's of the DVK2 and the Minikit boards, respectively.

3.3.1. LED0_STATE(X)
 LED1_STATE(X)
 LED2_STATE(X)
 LED3_STATE(X)

These macros set the state of the corresponding LED. Zero means off, one means on.

3.3.2. LED0_ON()
 LED1_ON()
 LED2_ON()
 LED3_ON()

These macros turn the corresponding LED on.

3.3.3. LED0_OFF()
 LED1_OFF()
 LED2_OFF()
 LED3_OFF()

These macros turn the corresponding LED off.

3.3.4. LED0_TOGGLE()
 LED1_TOGGLE()
 LED2_TOGGLE()
 LED3_TOGGLE()

These macros toggle the corresponding LED on or off.

4. CONTACT INFORMATION

ON Semiconductor
Oskar-Bider-Strasse 1
CH-8600 Dübendorf
SWITZERLAND

Phone +41 44 882 17 07
Fax +41 44 882 17 09
Email sales@onsemi.com
www.onsemi.com

For further product related or sales information please visit our website or contact your local representative.

ON Semiconductor and its trademarks are trademarks of Semiconductor Components Industries, LLC dba ON Semiconductor or its subsidiaries in the United States and/or other countries. ON Semiconductor owns the rights to a number of patents, trademarks, copyrights, trade secrets, and other intellectual property. A listing of ON Semiconductor's product/patent coverage may be accessed at www.onsemi.com/site/pdf/Patent-Marking.pdf. ON Semiconductor reserves the right to make changes without further notice to any products herein. ON Semiconductor makes no warranty, representation or guarantee regarding the suitability of its products for any particular purpose, nor does ON Semiconductor 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 special, consequential or incidental damages. Buyer is responsible for its products and applications using ON Semiconductor products, including compliance with all laws, regulations and safety requirements or standards, regardless of any support or applications information provided by ON Semiconductor. "Typical" parameters which may be provided in ON Semiconductor 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. ON Semiconductor does not convey any license under its patent rights nor the rights of others. ON Semiconductor products are not designed, intended, or authorized for use as a critical component in life support systems or any FDA Class 3 medical devices or medical devices with a same or similar classification in a foreign jurisdiction or any devices intended for implantation in the human body. Should Buyer purchase or use ON Semiconductor products for any such unintended or unauthorized application, Buyer shall indemnify and hold ON Semiconductor and its officers, employees, subsidiaries, affiliates, and distributors harmless against all claims, costs, damages, and expenses, and reasonable attorney fees arising out of, directly or indirectly, any claim of personal injury or death associated with such unintended or unauthorized use, even if such claim alleges that ON Semiconductor was negligent regarding the design or manufacture of the part. ON Semiconductor is an Equal Opportunity/Affirmative Action Employer. This literature is subject to all applicable copyright laws and is not for resale in any manner.