1.3inch Memory LCD User Manual

OVERVIEW

This is a black/white bicolor LCD display module with embedded memory, 1.3inch diagonal, 144x168 resolution, communicating via SPI interface.

It features lower power consumption compared to normal LCDs, and higher refresh rate compared to e-Papers without "ghosting" issue.

FEATURES

- 144x168 resolution, high contrast rate, clear displaying
- SPI interface, requires minimum GPIO for controlling
- Low power consumption, wide viewing angle, still viewable even under sunlight
- Comes with development resources and manual (examples for Raspberry Pi/Arduino/STM32)

SPECIFICATIONS

LCD type : Memory LCD

Interface : SPI

Display color : black, white

Resolution : 144x168

Display size : 20.88mm × 24.36mm

Operating voltage : 3.3V / 5V

Power : 1mW (typ.)

LCD type : Memory LCD

INTERFACES

PIN NO.	SYMBOL	DESCRIPTION
1	VCC	Power (3.3V / 5V input)
2	GND	Ground
3	MISO	Module SPI data output
4	MOSI	Module SPI data input
5	SCLK	Module SPI clock input
6	LCD_CS	Memory LCD chip selection
7	RAM_CS	SPI RAM chip selection
8	DISP	Display on/off signal
9	EIN	External Vcom signal

CONTENT

WORKING PROTOCOL

Different with common monochrome LCD and SPI TFT LCD, 1.3inch Memory LCD has features:

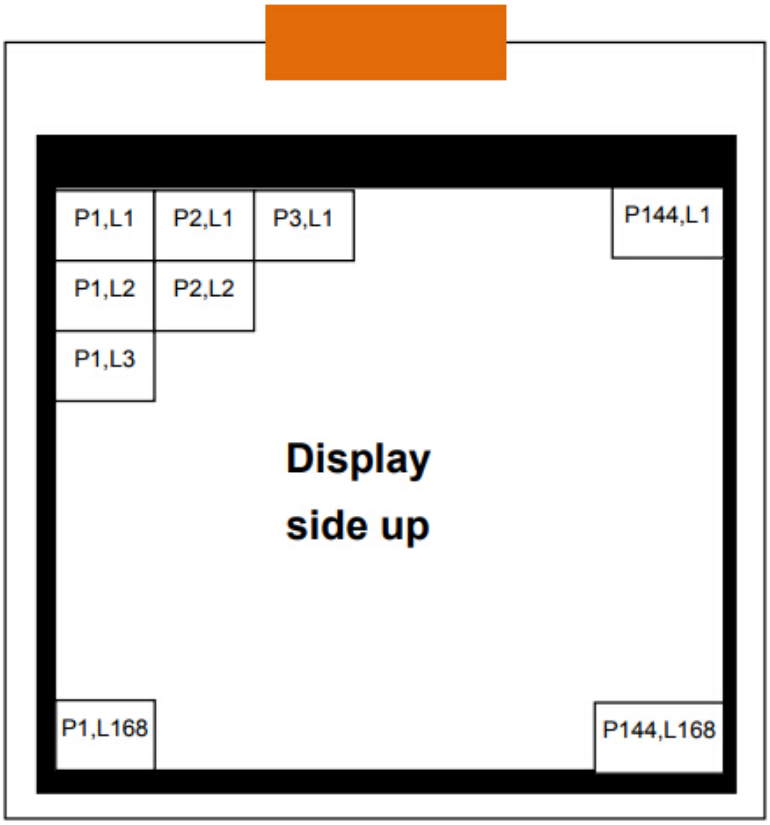
- Driver circuit is etched to ITO of LCD instead of external driver IC
- There is not internal RAM, works like a write only. So, image buffer is released by master
    - It requires 144x168/9 = 3024 bytes for a frame data. However, Arduino UNO R3 has only 2k RAM, we integrate a SRAM (SPI interface) on PCB as data buffer.
- Data updates line by line. You need to re-save and update whole line of data even just convert one pixel.

COMMUNICATION TIMING

The SPI timings of Memory LCD are similar

- Set CS to High, and delay for certain time ad the begin of communicating
- The first byte is display mode of Memory LCD
- The second byte: the line address where data updated. Note that the line address is begin from 1 instead of 0, LSB format:

Data position in display〔H,V〕



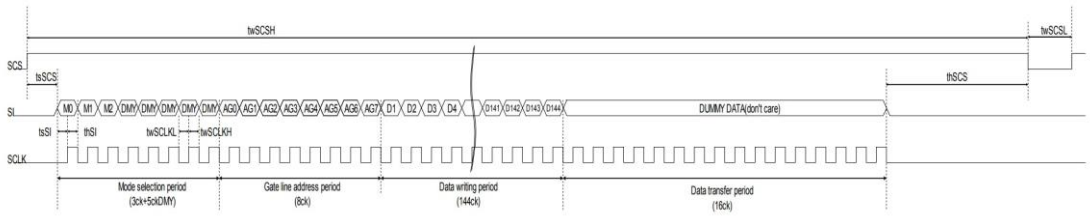
Gate line address setting

GL	AG0	AG1	AG2	AG3	AG4	AG5	AG6	AG7
1	1	0	0	0	0	0	0	0
2	0	1	0	0	0	0	0	0
3	1	1	0	0	0	0	0	0
4	0	0	1	0	0	0	0	0
5	1	0	1	0	0	0	0	0
6	0	1	1	0	0	0	0	0
7	1	1	1	0	0	0	0	0
8	0	0	0	1	0	0	0	0
:	:	:	:	:	:	:	:	:
161	1	0	0	0	0	1	0	1
162	0	1	0	0	0	1	0	1
163	1	1	0	0	0	1	0	1
164	0	0	1	0	0	1	0	1
165	1	0	1	0	0	1	0	1
166	0	1	1	0	0	1	0	1
167	1	1	1	0	0	1	0	1
168	0	0	0	1	0	1	0	1

- Following bytes are image data+16bit null data + delay time
- Set CS pin to Low to finish communicating

DATA UPDATING (ONE LINE)

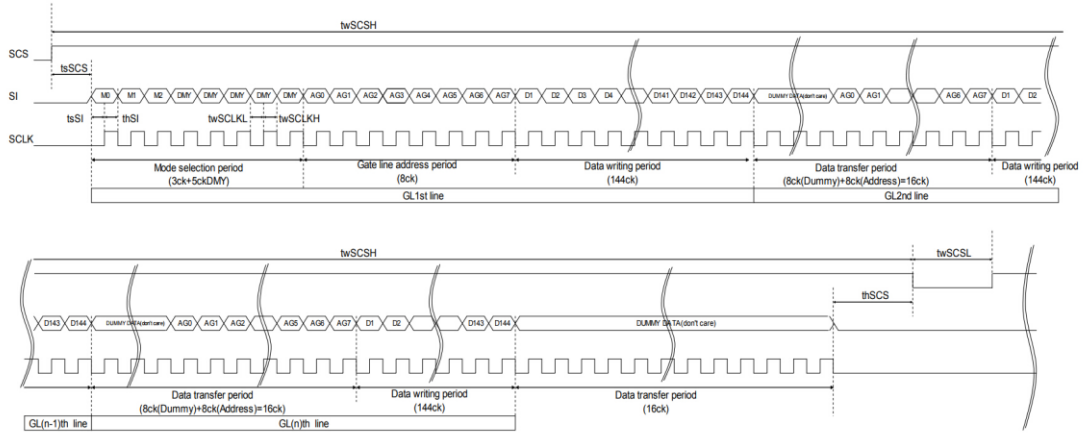
Update one line (M0 = “H”, M2 = “L”)



- M0: Mode flag. Set for “H”. Data update mode (Memory internal data update)
  - When “L”, display mode (maintain memory internal data).
- M1: Frame inversion flag.
  - When “H”, outputs VCOM=”H”, and when “L”, outputs VCOM=”L”.
  - When EXTMODE=”H”, it can be “H” or “L”.
- M2: All clear flag
  - All Clear Mode to execute clear.
- DUMMY DATA: Dummy data. It can be “H” or “L” (“L” is recommended.)
  - Data writ period: Data is being stored in 1st latch block of binary driver on panel
  - Data transfer period: Data written in 1st latch is being transferred (written) to pixel internal memory circuit.
- M1: Frame inversion flag is enabled when EXTMODE=”L”
- When SCS becomes “L”, M0 and M2 are cleared.

DATA UPDATE (MULTIPLE LINES)

Updates arbitrary multiple lines data. (M0=" H" 、M2 = " L")

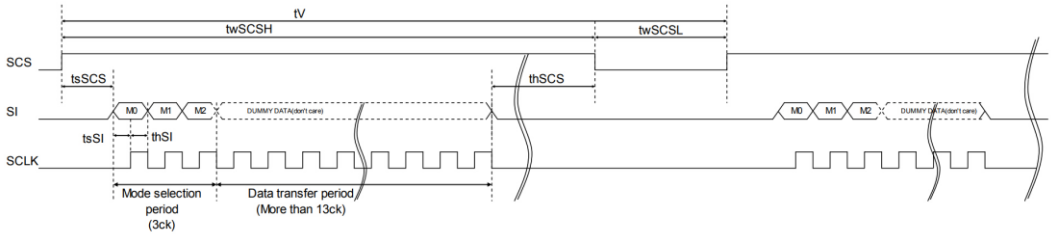


Almost same as one-line mode. The only difference is that: during data transferring period:

- Data transfer period:
  - For example, during GL2nd line data transfer period, GL 2nd line address is latched and GL1st line data is transferred from 1st latch to pixel internal memory circuit at the same time.

DISPLAY MODE

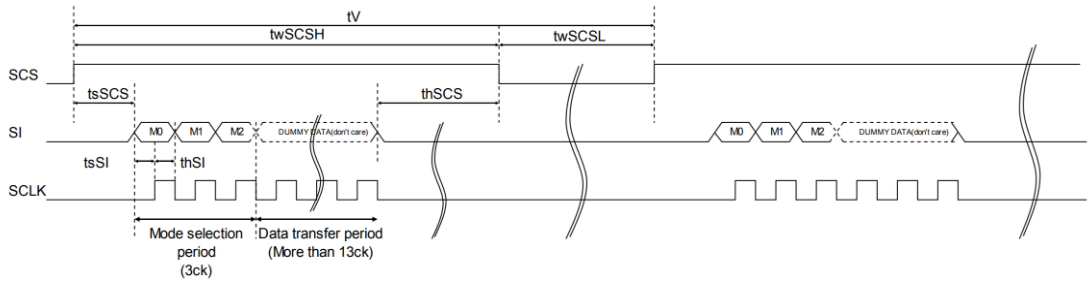
Maintains memory internal data (maintains current display). (M0=" L" 、M2 = " L")



M0: Mode flag. Set for "H". Data update mode (Memory internal data update)

ALL CLEAR MODE

Clears memory internal data and writes white. (M0=" L" 、M2 = " H")



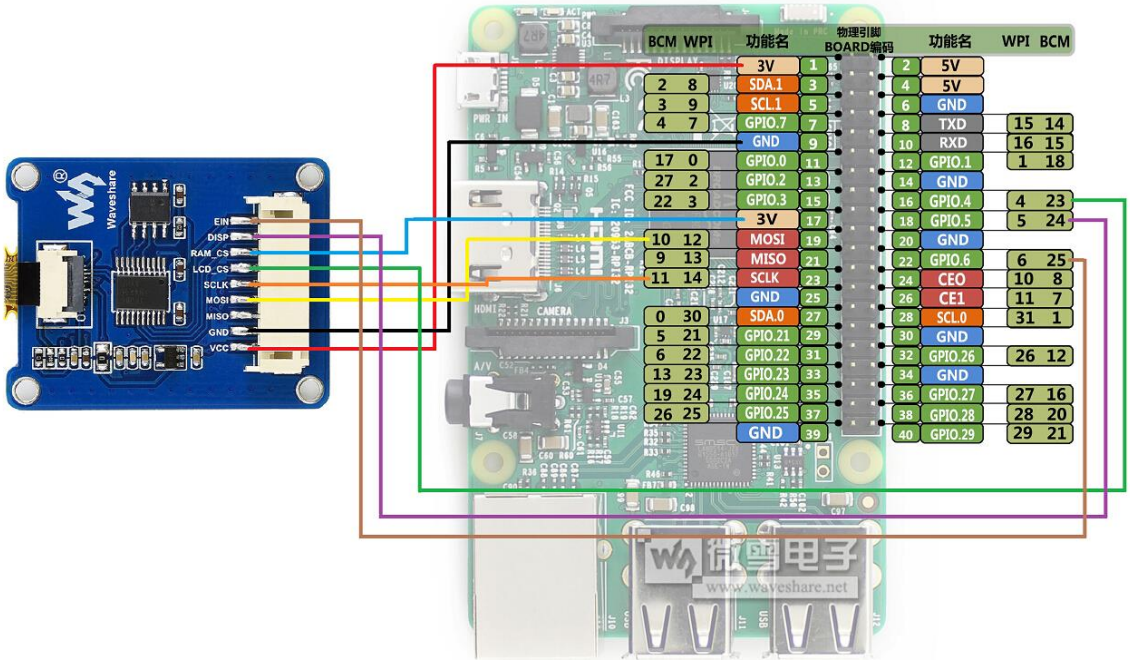
M2: All clear flag. Set it "H"

DEMO CODES

RASPBERRY PI

HARDWARE CONNECTION

PIN	SYMBOL	Raspberry Pi (BCM)
1	VCC	3.3V
2	GND	GND
3	MISO	N.C.
4	MOSI	10 (MOSI)
5	SCLK	11(SCLK)
6	LCD_CS	23
7	RAM_CS	3.3V
8	DISP	24
9	EIN	25



SOFTWARE

1. Enable SPI interface

   a) Method 1: modify config.txt file, uncomment line **dtparam=spi=on**

   Run command: **sudo nano /boot/config.txt** to open the file and modify

   b) Method 2: configure it in raspi-config page.

   Run command: **sudo raspi-config** to open the configuration page

   Click **Interfacing Options->SPI->Yes**

2. Install bcm2835 libraries:

   a) Download libraries from: http://www.airspayce.com/mikem/bcm2835/

   b) Install it

```
1. tar zxvf bcm2835-x.xx.tar.gz
2. cd bcm2835-x.xx
3. ./configure
4. make
5. sudo make check
```

```
6. sudo make install
```

3. Download demo code from github and run

```
1. git clone https://github.com/waveshare/MemoryLCD.git
2. cd MemoryLCD/
3. make
4. sudo ./mem_lcd
```

demo.h, demo.c: example codes

MemoryLCD.h, MemoryLCD.c: driver codes of LCD

pic.c: picture data

ugui*.*: open gui libraries uGUI

ARDUINO

HARDWARE CONNECTION

PIN	SYMBOL	Arduino
1	VCC	5V
2	GND	GND
3	MISO	D12
4	MOSI	D11
5	SCLK	D13
6	LCD_CS	D7
7	RAM_CS	D9
8	DISP	D8
9	EIN	D6

【Note】There are many version of Arduino board. The board described herein is Arduino UNO R3 and its compatible board. The operating voltage of Arduino UNO R3 is 5V, so, Vcc is connected to 5V (above table). If the board you use is 3.3V working voltage, you need to change the Vcc to 3.3V

SOFTWARE

1. Make sure you have installed newest Arduino IDE in your PC.
2. Download demo code from wiki.
3. Copy folder Memory-LCD to ...\Arduino\libraries
4. Open MonoDemo.ino, compile and download it to your Arduino board/

STM32

HARDWARE CONNECTION

PIN	SYMBOL	STM32
1	VCC	3.3V
2	GND	GND
3	MISO	N.C.
4	MOSI	PA7
5	SCLK	PA5
6	LCD_CS	PA8
7	RAM_CS	3.3V

8	DISP	PA9
9	EIN	PC7

【Note】The demo code is based on STM32F429IGTx

SOFTWARE

1. Make sure you have installed Keil-IDE in your PC
2. Download demo code from wiki
3. Open STM32 project, compile and download to your STM32 board