esp32-pcd8544

Release

Sep 23, 2017

Contents

This is the documentation for ESP32-PCD8544 (esp32-pcd8544). ESP32-PCD8544 is a driver library implementation for PCD8544 LCD controller a.k.a. Nokia5110 LCD.

CHAPTER 1

Contents

1.1 Get Started

This document is intended to introduce Nokia 5110 LCD as cheap graphic LCD for ESP-IDF users.

1.1.1 Introduction

PCD8544 is a model name of graphic controller integrated into Nokia 5110's LCD. Nokia 5110 is old-style GSM mobile phones that was available in 1998-2001.

Nokia 5110 seems widely sold over the world in GSM era. But people were already upgraded from them to smartphones now 2010s. Recent years, aimed at DIY electronics developers (a.k.a. makers), some Chinese manufacturers started disassemble no-more-used Nokia 5110's LCD with controllers, then hand-assembled modules with dedicated PCB + LED. From the surrrounding background, we can get these types of LCD module in very cheap cost (around $2 in Aliexpress.com) comparing with general graphic LCD's price (around $7-$15).

Wholesale nokia 5110 Gallery - Buy Low Price nokia 5110 Lots on Aliexpress.com

As far as I know, this is the cheapest graphic LCD generally availble in the world. It have small monochrome 84x48 pixels. But we can still expect that it works and enjoyable for electronic DIY projects: we can display sensor values plot them to small graphs, render monochrome raster graphics, etc by integrating an Nokia 5110 LCD to our project.

The problem was there were no available libraries dedicated to ESP-IDF (Espressif IoT Devlopment Framework) is . I could found some libraries works in Arduino framework, but nor ESP-IDF native SPI support or efficient implementation. That's why I started to developing ESP32-PCD8544 library.

1.1.2 Warning: Assemble Quality of Nokia 5110 LCD

Though Nokia 5110 LCD is very cheap graphic LCD, I exprienced around 50% failure rate based on 4 orders from different shops. There are two type of PCB (red one and blue one), and some manufacturers are behind shops. There are variations on assemble quality by manufacturers but we cannot choose manufacterer directory. As far as I know, I recommend (but of cource no guarantee) following shops because I could purchese working indivisuals.

LCD Module LCD5110 Board Blue Screen Backlight Display Module 5110 Wholesale-in LCD Modules from Electronic Components & Supplies on Aliexpress.com | Alibaba Group

LCD Module Display Monitor White backlight adapter PCB 84*48 84x84 5110 Screen For Arduino-in LCD Modules from Electronic Components & Supplies on Aliexpress.com | Alibaba Group

Note that in Aliexpress.com allow us to open dispute to the shop which we purchesed items. If delivered Nokia 5110 LCD was are not working individual, you can open dispute with evidence (photos) and then Aliexpress will judge that it should be refunded.

1.1.3 Related Links

- Zero Cost 84×48 Graphical LCD for the Freedom Board | MCU on Eclipse (external site)
- How to use the Nokia 5110 LCD Module at Arduino (external site)
- genindex

1.2 API Reference

1.2.1 ESP32-PCD8544

Overview

ESP32-PCD8544 is dedicated driver implementation of PCD8544 LCD controller (a.k.a Nokia 5110 LCD) for Espressif IoT Development Framework.

Application Examples

- "Hello world" example - hello_pcd8544
- Draw lines via frame buffer - draw_lines
- Positioning and formatting texts - print_texts
- Drawing bitmap images - draw_bitmaps
- Miscellaneous mode controls - control_modes
- Plotting touch sensor values - plot_touch_sensor
- Plotting high frequency analog microphone input with DMA - plot_microphone_input

(more example projects will be written here)

API Reference

Header File

- include/pcd8544.h

Functions

void **pcd8544_init**(*pcd8544_config_t* *config)
Initialize PCD8544 device.

Initialize PCD8544 as SPI slave. You can make some operations via APIs offered from ESP32-PCD8544 until pcd8544_free() is called.

Parameters

- `config`: Configuration of PCD8544 SPI

void **pcd8544_set_backlight**(bool *on*)
Set LCD backlight status.

Set LCD backlight enabled / disabled.

Attention To use this function correctly you have to initialize `.is_backlight_common_anode` member in *pcd8544_config_t* according to the model of your Nokia 5100 LCD Module (red one or blue one, or you will get opposite result.

Parameters

- `on`: boolean parameter to activate or deactivate LED for LCD.

void **pcd8544_set_powerdown_mode**(bool *powerdown*)
Control power-down mode / chip-active mode status of PCD8544.

PCD8544 provides `power-down` mode that consume only very low current on inactive status. This function control its status. In power-down mode, only 1.5uA current is consumed based on datasheet.

Attention On entering power-down mode, backlight goes off for to reduce current. Similary, when back to chip-active mode backlight goes on.

Attention To enter power-down mode correctly, content of frame buffer and LCD DRAM is cleared before entering power-down mode. Thus you have to recover LCD contents by yourself after back to chip-active mode.

Parameters

- `powerdown`: boolean parameter to activate or deactivate power-down mode.

void **pcd8544_set_display_mode**(*pcd8544_display_mode* *mode*)
Control display mode of the LCD.

PCD8544 provides four display modes. `normal`, `all segments on`, `inverted` and `blank`. This function controls PCD8544 between modes which defined on pcd8544_display_mode.

`normal` mode, DDRAM (Display Data RAM) as-is applied to LCD content.

`inverted` mode makes oppsite on / off state of LCD segments.

In `all segments on` mode, you can confirm electrical connection by setting and very high contrast seting.

`blank` mode make make all segments off.

Attention After pcd8544_init() called, display mode goes `normal`.

Attention `blank` mode does not erase LCD contents. It is hold in DDRAM (Display Data RAM) and still and can be recovered by switching to other modes.

Parameters

- `mode`: pcd8544_display_mode value for entering other display mode.

void **pcd8544_set_contrast**(uint8_t *vop*)
Control contrast of the LCD.

PCD8544 have contrast control functionality. You can access this by setting contrast value. Optimal parameter may different from each individuals. pcd8544_init() resets contrast to empilically good default but you can tune via this function.

Attention pcd8544_init() calls this function internally with empilically good default (vop=20) so you don't need to tune contrast manually. You can still tune contrast against different indivisual characteristics.

Parameters

- `vop`: Operation voltage parameter for LCD (0~127). In default configuration options, range between 20 to 24 will be good choice.

void **pcd8544_clear_display**()
Clear all LCD content.

This function content in frame buffer of LCD.

Attention To confirm result on the LCD, you have to call pcd8544_finalize_frame_buf() after frame buffer writing operation is finished.

void **pcd8544_draw_line**(uint8_t *x0*, uint8_t *y0*, uint8_t *x1*, uint8_t *y1*)
Draw line to the LCD.

This function draw an line on the frame buffer. You can specify start and end points in geometory.

Attention To confirm result on the LCD, you have to call pcd8544_finalize_frame_buf() after frame buffer writing operation is finished.

Parameters

- `x0`: x of the start point.
- `y0`: y of the start point.
- `x1`: x of the end point.
- `y1`: y of the end point.

void **pcd8544_draw_rectangle**(uint8_t *x0*, uint8_t *y0*, uint8_t *x1*, uint8_t *y1*)
Draw rectrangle to the LCD.

This function draw rectangle on the frame buffer. You can specify upper-left and bottom-right points in geometory.

Attention To write line data to the DDRAM and update segments on the LCD, you have to call pcd8544_finalize_frame_buf() after all frame buffer writing operation is finished.

Parameters

- `x0`: x of the upper-left corner.
- `y0`: y of the upper-left corner.
- `x1`: x of the bottom-right corner.
- `y1`: y of the bottom-right corner.

void **pcd8544_set_pos**(uint8_t *row*, uint8_t *col*)
Set (row, col) position operation start position.

This function set start position of the next operation.

Attention Unlike draw_line() and draw rectangle() accepts (x,y) geometory as arguments, pcd8544_set_pos() accepts row(0-83) and column(0-6) dimentions. Thus, you can spcify only per 8 segments for vertical direction.

Parameters

- `row`: start row of the next operation (0-83)
- `col`: start column of the next operation (0-6)

void **pcd8544_draw_bitmap**(**const** uint8_t **bitmap*, uint8_t *rows*, uint8_t *cols*, bool *transparent*)
Draw arbitrary bitmap to the frame buffer.

Attention `*bitmap` buffer should be sized (rows * cols) bytes. and decomposed per `rows` bytes for drawing each columns.

Attention the Top-left corner position of the bitmap is determined by last pcd8544_set_pos() arguments..

Attention To write line data to the DDRAM and update segments on the LCD, you have to call pcd8544_finalize_frame_buf() after all frame buffer writing operation is finished.

Parameters

- `*bitmap`: pointer to a uint8_t buffer contains the bitmap data.
- `rows`: number of rows of the bitmap data.
- `cols`: number of cols of the bitmap data.
- `transparent`: boolean value for whether operate '0' bits as transparent or not.

void **pcd8544_finalize_frame_buf**()
Finish frame buffer operations of current frame and write to DDRAM of the LCD.

To avoid redundant data transfer via SPI bus, functions `pcd8544_clear_display()` `pcd8544_draw_line()`, pcd8544_draw_rectangle() and `pcd8544_draw_bitmaap()` theirselves does not transfer to the DDRAM. Alternatively, `pcd8544_finalize_frame_buf()` does this operation to apply frame buffer state to the LCD at once.

For each frame, you have to call this function after above functions are called.

void **pcd8544_puts**(**const** char **text*)
Print text to the LCD.

Like standard `puts()` function, you can write arbitrary text to the LCD via this function. Position of the Text is determined from last pcd8544_set_pos() call.

As this library includes 8x5 dots font, you can write up to 14 characters per row.

Attention If writing position of the text reached right-end of the LCD, writing position is automatically moves to the next column. When it was the bottom column (column 6), the next column is column 1.

Parameters

- `text`: `const char*` text to print.

void **pcd8544_printf**(**const** char *_format_, ...)
Print formatted text to the LCD.

Like standard `printf()` function, you can write arbitrary text with format. Position of the Text is determined from last pcd8544_set_pos() call.

Format string is full compatible with standard `printf()` as this function uses `spritntf()` internally.

Attention If writing position of the text reached right-end of the LCD, writing position is automatically. moves to the next column. When it was the bottom column (column 6), the next column is column 1.

Parameters

- `*format`: format string to print.
- `...`: arbitrary arguments for format string.

void **pcd8544_sync_and_gc**()
Sync background SPI transfer and free memory of buffers used for transfer.

ESP32-PCD8544 driver offers asyncronous SPI tranfer to allow users to other operations during SPI transaction. To achieve this, some internal data should be dynamically allocated and have to wait until SPI transaction finished. This function waits all SPI transfer queue finisshed, and free dynamically allocated buffers for them.

Attention Developer must call this function in the last of the frame operations, or it cause memory leak.

void **pcd8544_free**()
Free SPI related resource completely.

ESP32-PCD8544 utilize single SPI host and optionally DMA channel of ESP32 to drive PCD8544. Developer may want to re-use SPI host after some content has been written to the LCD. In that case, SPI device and bus occupied by `pcd8544_init()` can be freed with this function.

Attention Once `pcd8544_free()` is called, you have to call `pcd8544_init()` again to later use.

Structures

struct pcd8544_config_t
Confirugration for PCD8544.

Configurations for PCD8544 and Nokida 5110 LCD board.

Public Members

bool **is_backlight_common_anode**
configure installed PCD8544 board PCBs LED are common-anode or common-cathode

There are two type of PCB is selling store for Nokia5100 LCD module. They can be determained from its color; red one and green one.

Red one comes with common-anode LED so it this member should be `true`.

Blue one comes with common-cathode LED so it this member should be `false`.

spi_host_device_t **spi_host**
SPI host to be used for PCD8544.

You can choose from ESP32' logical 3 SPI channels `SPI_HOST`, `HSPI_HOST` and `VSPI_HOST`.

Attention Differnt SPI host have different native pin assignemnt for fast and low latency IO_MUX. So ESP23-PCD8544 offers different pin assignment according to `.spi_host` member. You can confirm which pins are assigned via UART serial output log in INFO lebel.

Attention If you want to use DMA channel for low latency and less flicker, choose HSPI_HOST or VSPI_HOST that is currently available for DMA transfer.

uint8_t **dma_chan**
DMA channel to use (1 or 2) or no use DMA channel (0).

ESP32 offers two DMA channel; 1 and 2. You can optionally use a DMA channel for less CPU load, low latency and high throughput SPI transfer. If you already using DMA channels for other use (like I2S-connected speaker, mic or image sensor), you can specify `.dma_chan = 0` to tell ESP32-PCD8544 not to use any DMA channel.

pcd8544_spi_pin_config_t ***spi_pin**
Manual SPI pin assignment configuration.

Althrough native SPI pin assignment for SPI host is recommended as it enables low latency SPI transfer by IOMMU, you can still specify any IO pin numbers for each SPI pins one by one, via setting pointer to `*spi_pin` member.

Attention All specified GPIO pins must be able to configured output pins. (see "2.2 Pin Description" in the ESP32 Datasheet)

Attention Keeping this member `NULL` allows ESP32-PCD8544 to auto-configure native SPI pins corresponding to selected SPI host. You can confirm which pins are assigned to control pins from UART serial output log (In INFO log level).

pcd8544_control_pin_config_t ***control_pin**
Manual control pin assignment configuration.

To drive PCD8544, it requires some additoinal control pins from tradisonal SPI pins; `DC`(Data/Command) pin, `RST`(Reset) pin and `BL`(Backlight) pin. You can specify them manually alternative to automatic configuration by ESP32-PCD8544.

Attention All specified IO pins should be able to configured output pins.

Attention Keeping this member `NULL` allows auto configuration of recommended control pin assignment by ESP32-PCD8544. You can confirm which pins are assigned to control pins from UART serial output log (In INFO log lavel).

Macros

PCD8544_FRAME_BUF_SIZE
Internal frame buffer size.

PCD8544_TRANS_QUEUE_SIZE
SPI Tranfer queue size for ESP32-PCD8544.

PCD8544_BIAS_SYSTEM
Bias System parameter for PCD8544.

It affects results of the contrast setting via `pcd8544_set_contrast(uint8_t vop);`

You may configure this parameter via `make menuconfig` if you have contrast range problem.

PCD8544_TEMPERATURE_COEFFICIENT
Temperature coefficient parameter for PCD8544.

It affects results of the contrast setting via `pcd8544_set_contrast(uint8_t vop);`, in low tempearature environment.

You may configure this parmeter via `make menuconfig` if you have contrast range problem.

PCD8544_MAX_TRANS_LENGTH_WITHOUT_DMA
Maximum buffer size (in bits) for non-DMA SPI transfer.

If you initialized pcd8544_init(config) with `config.dma_chan = 0`, SPI data transfer via DMA is disabled. In that case, Maximum transfer data size is limited to this size in bits. ESP32-PCD8544 decompose large size of data transfer requrest into multiple 256 bits of request to fit this limitation.

Attention Full frame buffer transfer cause 84x6=504 bytes = 4032 bits of request. In non-DMA environment, the request is decomposed into each 256 bits i.e. 16 data transfer request occurs. YOu may want to increase PCD8544_TRANS_QUEUE_SIZE from kconfig.

PCD8544_WARN_SUPPRESS_INTERVAL_MS
Interval for surpressing repeatedly displaying warnings in milliseconds.

To surpress queue size related warning, certain interval in milliseconds is used.

Type Definitions

typedef struct *pcd8544_config_t* **pcd8544_config_t**
Confirugration for PCD8544.

Configurations for PCD8544 and Nokida 5110 LCD board.

Enumerations

enum pcd8544_addressing_mode
Values:

PCD8544_ADDRESSING_MODE_HORIZONTAL
Make DDRAM address pointer to move from (row,col) like (0,0), (1,0), (2,0) . . .

(83,0), (0,1), (1,1) order for each data transfer.

PCD8544_ADDRESSING_MODE_VERTICAL
Make DDRAM address pointer to move from (row,col) like (0,0), (0,1), (0,2) . . .

(0,5), (1,0), (1,1) order for each data transfer.

enum pcd8544_display_mode
Values:

PCD8544_DISPLAY_MODE_BLANK = 0b00
Make LCD content to blank.

PCD8544_DISPLAY_MODE_ALL_SEGMENTS_ON = 0b01
Make all LCD segments on (for pin connection check)

PCD8544_DISPLAY_MODE_NORMAL = 0b10
Make LCD segments correspond to DDRAM data.

PCD8544_DISPLAY_MODE_INVERSE = 0b11
Make LCD segments opposite to DDRAM data.

Header File

- include/pcd8544_pin.h

Functions

const *pcd8544_spi_pin_config_t* ***pcd8544_native_spi_pin_config**(spi_host_device_t *host*)
Native pin configuration for SPI host.

In ESP32 Datasheet, native pin configuration that enables IOMUX faster and low latency data transfer method comparing to GPIO matrix is recommended. This function returns Native SPI pin assignnments for SPI host, that are defined in driver/spi_common.h of ESP-IDF.

Return *pcd8544_spi_pin_config_t* Native pin configuraion for SPI host

Parameters

- `host`: SPI host to request native pin configuration

const *pcd8544_control_pin_config_t* ***pcd8544_default_control_pin_config**(spi_host_device_t *host*)
(internal use) Get recommended control pin assignments for PCD8544 controller.

Like other SPI display controllers, PCD8544 requires some additional control pins against traditional SPI RST(Reset), BL(Backlight) and DC(Data/Control). These pins must be able to configures as output pins. Recommended pins (i.e. not reserved for other important functionality) are retunred for SPI host.

Return *pcd8544_control_pin_config_t* Recommended control pin configuraion for SPI host

Parameters

- `host`: SPI host to request control pin configuration

Structures

struct pcd8544_spi_pin_config_t
custom configuration for PCD8544 control pins

ESP32-PCD8544 offers automatic configuration for native SPI pins according to SPI host, to allow IOMUX fast and low latency SPI transfer. But you can specify them manually too. In that case initialize this structure and assign to the `spi_pin member of pcd8544_config_t, passed to`pcd8544_init(); ‘

For more detail, see “Chapter.4 IO_MUX and GPIO Matrix” in the ESP32 Technical Reference Manual.

Attention You have to specify IO pins that can be configured as output pin. See ESP32 datasheet for more detail.

Public Members

uint8_t **miso_io_num**
GPIO pin number for SPI MISO pin.

J This pin is actually not used in PCD8544, but have to be specified for compatibility.

uint8_t **mosi_io_num**
GPIO pin number for SPI MOSI pin.

GPIO pin number connected to the `Din` pin on Nokia 5110 LCD Module.

uint8_t **sclk_io_num**
GPIO pin number for SPI CLK pin.

GPIO pin number connected to the `Clk` pin on Nokia 5110 LCD Module.

uint8_t **spics_io_num**
GPIO pin number for SPI CS pin.

GPIO pin number connected to the `CE` pin on Nokia 5110 LCD Module.

struct pcd8544_control_pin_config_t
custom configuration for PCD8544 control pins

ESP32-PCD8544 offers recommended assignments for control pins. But you can specify them manually too.

Attention You have to specify IO pins that can be configured as output pin. See ESP32 datasheet for more detail.

Public Members

uint8_t **reset_io_num**
GPIO Pin number for Reset control pin.

GPIO pin number connected to the `RST` pin on Nokia 5110 LCD Module

uint8_t **dc_io_num**
GPIO Pin number for D/C control pin.

GPIO pin number connected to the `DC` pin on Nokia 5110 LCD Module

uint8_t **backlight_io_num**
GPIO Pin number for backlight control pin.

GPIO pin number connected to the `BL` pin on Nokia 5110 LCD Module

Type Definitions

typedef struct *pcd8544_spi_pin_config_t* **pcd8544_spi_pin_config_t**
custom configuration for PCD8544 control pins

ESP32-PCD8544 offers automatic configuration for native SPI pins according to SPI host, to allow IOMUX fast and low latency SPI transfer. But you can specify them manually too. In that case initialize this structure and assign to the `spi_pin member of` *`pcd8544_config_t`*`, passed to`pcd8544_init(); ‘

For more detail, see “Chapter.4 IO_MUX and GPIO Matrix” in the ESP32 Technical Reference Manual.

Attention You have to specify IO pins that can be configured as output pin. See ESP32 datasheet for more detail.

typedef struct *pcd8544_control_pin_config_t* **pcd8544_control_pin_config_t**
custom configuration for PCD8544 control pins

ESP32-PCD8544 offers recommended assignments for control pins. But you can specify them manually too.

Attention You have to specify IO pins that can be configured as output pin. See ESP32 datasheet for more detail.

1.2.2 Configuration Options

Introduction

ESP-IDF uses Kconfig system to provide a compile-time configuration mechanism. Kconfig is based around options of several types: integer, string, boolean. Kconfig files specify dependencies between options, default values of the options, the way the options are grouped together, etc.

Applications developers can use `make menuconfig` build target to edit components' configuration. This configuration is saved inside `sdkconfig` file in the project root directory. Based on `sdkconfig`, application build targets will generate `sdkconfig.h` file in the build directory, and will make sdkconfig options available to component makefiles.

Configuration Options Reference

Subsequent sections contain the list of available ESP32-PCD8544 options, automatically generated from Kconfig files. Note that depending on the options selected, some options listed here may not be visible by default in the interface of menuconfig.

By convention, all option names are upper case with underscores. When Kconfig generates sdkconfig and sdkconfig.h files, option names are prefixed with `CONFIG_`. So if an option `ENABLE_FOO` is defined in a Kconfig file and selected in menuconfig, then sdkconfig and sdkconfig.h files will have `CONFIG_ENABLE_FOO` defined. In this reference, option names are also prefixed with `CONFIG_`, same as in the source code.

Index

P