

SP-MASTER

display a simple graphics with vaguing pixel colors ont the

320 x 240 LCD on an ESP-WROVER-KIT.

#include (stdion)

include (stalib.n) # include (string.h)

include "freertas/Free RTOS. h"

include "freertos/task.h" #include "esp-system.h"

#include "driver/spi-master.h"

include "driver/gpio.h"

#include "pretty-effect-h" < local

11 Spi-device-transmit 11 spi-device-queve-trans

11 spi-device-trans-result

11 149341 / ST7789V:

define PIN NUM_MOUSO 25

define PIN_NUM_MOSI 23

#define PIN-NUM_CLK 19 # define PIN_NUM_CS 22

#define PIN-NUM DC 21

#define PIN_NUM_RST 18

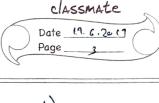
#define PIN-NUM_BCKI 5

#define PARALIEL LINES 16

typeday struct & wints t cmd; vinte + data [16]; vinte + data bytes; 3 lcd_init_cmd_t. typedel enom & LCD_TYPE_ILI =1, LCD_TYPE_ST, LCD TYPE MAX 3, type_lcd_t, DRAM ATTR static const led init condit st init ends [] = & DRAM ATTR static const led-init and + ili-init - emds[] = & 3, void Ica cond (spi device handle + spi, const vinte + cond) & expert ret; spi-transaction + t; memset (&t, 0, size of (t)). t.length = 8.

t. tx buffer = & cmd;

t-user = (void =) o;



ret = spi_device_polling_transmit(spi, &t); assert (ret == ESP-OK);

void led data (spi device handle & spi, const vinta - t *data, in espernt ret, Spi-transaction-t t;

if (len=o) return; memset (at, o, size of (1)). t. length = len x 2.

t. tx_buffer = data;

t. user = (void *) 1;

ret = spi device polling transmit (spi, &t); assert (ret == ESP_OK),

void led-epi-pre-transfer-collback (spi-transaction + * +) gs int dc = (int) t -> user,

gpio cet level (PIN_NUM_DC, dc);

vint32-t led get id (pi-device handle + spi) &

led-emd (spi, oxoy), spi-transaction t; memset (&t, o, size of (+)).

t.length = 8 x 3;

t. flags = SPI_TRANS_USE_RXDATA; t-user = (void *) 1; esperrt ret = spi-device - polling-transmit (spi, &t); ascert (ret == ESP.OK); return * (vint 32 t) t. rx - data; void led-init (spi-device-handle + spi) & int cmd = 0; const ladeinit and + * lad init ands; apio- at direction (PIN-NUM DC GPIO MODE OUTRUT gpio set direction (PIN_ NUM_RST, GIPIO MODE_ OUTPUT) apio_set_direction (PIN_NUM_BCKL, GPIO_MODE_ONTPUT). apib_set level (PIN-NUM-RT, 0). vTaskDelay (100/portTICK_RATE_MS). gpio-set_level (PIN_NUM_RCT, 1). VTack Delay (100 / portTick RATE_MS), wint82 t led-id = led-get id (spi); int led detected type = 0; int lad-type; printf ("LCD ID: XOEX In" (cd-id). if [led_id = = 0) { led_ detected_ type = 100 TYPE - 111 print ("IL19341 detected. in"). lad_detected_type = KCD-TYPE-SP; printf ("ST77891 detected in").

Classmate # ifdet CONFIGELCO-TYPE AUTO Icd type = Icd detected type; # elif defined (CONNFIG - LCD - TYPE ST 7789V) south print ("Kconfig: force CONFIG_LCD-TYPE lcd-type = LCD_TYPE_ST; # elif defined CONFIG LCD-TYPE ILIA341 print ("Kconfig: force CONFIG. LCD. TYPE 1619341. \n"); lcd-type = LCD_TYPE_ILI; # endit ef (lcd-type == LCD-TYPE_ST) & print ("LCD ST7789V initialization ("); led_init_emds = st_init_emds; 3 else & print ("LCD 149341 initialization. \n"); led-init emds = cli-init-emds: while (Icd_init-cmds [cmd] databytes 1= 0xFF)

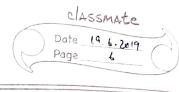
led and (spi, led-init ands tend] and)

ef (led-init- and [cma] data bytes & 0x80) &

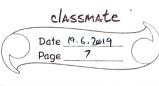
VTask Delagy (100 / portTICK_RATE_MS);

apia set level (PIN_NUM, BCKL, 0).

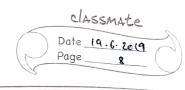
led-data Capi, led-init and s [cmd]. data gotes BONIF)



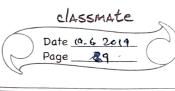
static void send-lines (spi-device-handle-t spi, int gos, until t espert ret: i'n memony int n; & Stack > static spi-transaction-t trans[6]; for (x=0, x < 6, x++) & memset (& transIx], o, size of (spi-transaction-t); if ((nx1) == 0) & trans [n]. length = 8; trans [x] user = (void +) 0; 3 else & trans [x], length = 8 ox 4; trans[x], user = (void x) 1: Ł transtal flags = CPI TRANS DEE TX DATA; trans[0123] .. trons [4]. tx-data [0] = 0x2c; trans [s]. tx-buffer = line data; frons [5] length = 320 x 2 x 8 x PARALLEL LINES. transts] flags = 0; for (x=0, x < 6, x++) { ret = Spi_ device - queue trans (spi, strans[a], port MAY DELAY) assert (ret == ESP.OK); 4 3



Static void cend line finish (spi-device handle + epi) & Spi-transaction - + + trans; esp-err-t ret; for (cônt x=0, x(6, x++) } ret = Spi- device get_ trans_result (spi, &rtrans, portMAN DELAY) assert (ret == ESP_OK); 4 Static void display pretty colors (spi-device handle + spi) & vint 16-t * lines [2]; for (cint i=0; (12; 1+4) & lines [i] = heap_caps malloc (320 * PARALLEL LINES * size of (vintre + assert (lines [i] != NULL). int frame = 0; int sending line = -1. ont calc line = 6, while (i) & for (ont y=0; y<240; y+= PARALLEL LINES) & pretty-effect-calc lines (lines Calc line), y, frame, PARALLEL LINES if (sending-line = mouting send line finish (spi). Sending-line = cole_line; colc_line = Calc_line == i) ? 0:1; Send lines (spi, y lines [sending line]).



		Page &
×.	STORAGE W	FI GTA
y.,	OBJECTNE:	MOSQUITTO SERVER MOTT HELLO WORLD
r _{is} ,		FCLIPSE PUBLISH - SUBSCRIBE
5-12-		
y		
×-1,-		
۲.,,		void app_main () &
-		esperrat vet;
-		Spi-device-handle-t spi;
Y- 		Spi-bus config. + buschg = &
۲.,		miso-io-num = PIN-NUM-MISO,
· [mosi-i'o_ nom = PIN_NOM_Mosi
- 1 Fy		. scik_io_nom = PIN_NUM_CLK
 		· quadap-io_num = -1
	. '	· quadant_io_nom = -1,
, 		, max transfer_SZ = PARALLEL LINES + 320+2 + 2
- 		3 ,
· ·		spi-device interface config-t develg = &
7		# ifdet CONFICY_LCD_OVERCLOCK
- F.,		-clock-speed-hz = 26 ok 1000 ok 1000,
-		# else
-12		· clock-speed_hz= 10 of 1000 of 1000
-12		# endif
-1		·mode = 0,
12	+	. spics_io_num = PIN_NUM_cs,
-		queve size = 7
1		· pre cb= lcd_spi-pre_transfer_callback,
1		3 .
-		ret = opi-bus inHializa (HSPI_HOST &buschg, 1).
		0 1



ret = Spi_bus_add device (HSPI_HOST, Bdevctg, Bspi).

ESP_ERROR_CHECK (ret);

lcd_init (Spi);

ret = pretty_effect_init();

ESP_ERROR_CHECK (ret);

display_pretty_colors (spi);

3