Topics Blogs People Forum Shop Contest Help

Enter or register

EasyElectronics.ru Community

All Collective Personal TOP

Good Bad

Поиск

STM8L – I/O Lines

STM8

PCBWay
HIGH-QUALITY PCB
ONLY $5 FOR 10 PIECES
• Rogers, HDI, aluminum and rigid-flex PCB are available now
• Production time 24 hours
PCB ASS
Free shipping
ONLY
Compon
Quality a

As always, it is easier to start with the simplest, namely with the input-output lines. Knowing how to work with them will allow us to blink LEDs and query the state of the button. And as an example, working with the LCD indicator from the Nokia 1100 cell phone.

General-Purpose I/O Lines
Several modes of operation of the I/O lines are possible: Block diagram: According to the block diagram, five registers are used to control the operating modes and data input/output; the register structure is declared in the library file:
image

image

```
/**
  * @brief General Purpose I/Os (GPIO)
  */
typedef struct GPIO_struct
{
  __IO uint8_t ODR; /*!< Output Data Register */
  __IO uint8_t IDR; /*!< Input Data Register */
  __IO uint8_t DDR; /*!< Data Direction Register */
  __IO uint8_t CR1; /*!< Configuration Register 1 */
  __IO uint8_t CR2; /*!< Configuration Register 2 */
}
GPIO_TypeDef;
```

Base addresses for registers:

```
#define GPIOA_BASE(uint16_t)0x5000
#define GPIOB_BASE(uint16_t)0x5005
#define GPIOC_BASE(uint16_t)0x500A
#define GPIOD_BASE(uint16_t)0x500F
#define GPIOE_BASE(uint16_t)0x5014
#define GPIOF_BASE(uint16_t)0x5019
```

Live

Comments Publications

penzet → Sprint Layout in OS X 18 → Software for electronics engineer

Vga → EmBitz 6 → Software for electronics engineer

Vga → Rail-to-rail: ideal operational amplifier or a clever marketing ploy? 1 → Theory, measurements and calculations

Flint → A little more about 1-wire + UART 56 → Hardware connection to the computer.

penzet → Cross-platform terminal - SerIO 3.x 25 → Software for electronics engineer

Gornist → PIP regulator 2 → Algorithms and software solutions

whoim → CC1101, Treatise on Tracing 89 → Blog im. khomin

OlegG → Clock on Bluetooth LE module 8 → Cypress PSoC

Technicum505SU → Nuances of PWM control of a DC motor by a microcontroller 3 → Theory, measurements and calculations

sunjob → DDS synthesizer AD9833 88 → Blog named after grand1987

DIHALT → W801, LCD screen and fly in the ointment 2 → Detail

nictrace → New Arduino-compatible board. 20 → FPGA

sunjob → Connecting the ARM GCC compiler to CLion 1 → Software for electronics engineers

Vga → CRC32: on STM32 as on PC or on PC as on STM32. 58 → STM32

dmitrij999 → Capturing images from a USB camera using STM32 6 → STM32

Gilaks → Expanding the capabilities of a simple MC up to an ADC on 2 or 1 pin. 8 → Theory, measurements and calculations

sva_omsk → Lithium ECAD - Russian PCB CAD 40 → Software for electronics engineer

x893 → W801 - budget controller with Wi-Fi 4 → Details

podkassetnik → Changing the standard instrument cluster lighting of Logan-like cars 5 → Automotive electronics

```
#define GPIOG_BASE(uint16_t)0x501E
#define GPIOH_BASE(uint16_t)0x5023
#define GPIOI_BASE(uint16_t)0x5028
```

Based on the operating modes, I wrote macros for working with input/output lines:

```
/*
 * File: GPIO.h
 * Date: 23.01.2011
 */

#ifndef MCU_GPIO_H_
#define MCU_GPIO_H_

// PIN_TEST A, 3, HIGH, INPUT_FLOATING_WITHOUT_INTERRUPT

#define INPUT_FLOATING_WITHOUT_INTERRUPT INPUT_FLOATING_WITHOUT_INTERRUP
#define INPUT_FLOATING_WITH_INTERRUPT INPUT_FLOATING_WITH_INTERRUPT
#define INPUT_PULL_UP_WITHOUT_INTERRUPT INPUT_PULL_UP_WITHOUT_INTERRUPT
#define INPUT_PULL_UP_WITH_INTERRUPT INPUT_PULL_UP_WITH_INTERRUPT
#define OUTPUT_OPEN_DRAIN_NOSPEED_LIMIT OUTPUT_OPEN_DRAIN_NOSPEED_LIMIT
#define OUTPUT_OPEN_DRAIN_SPEED_LIMIT_10MHZ OUTPUT_OPEN_DRAIN_SPEED_LIMI
#define OUTPUT_PUSH_PULL_NOSPEED_LIMIT OUTPUT_PUSH_PULL_NOSPEED_LIMIT
#define OUTPUT_PUSH_PULL_SPEED_LIMIT_10MHZ OUTPUT_PUSH_PULL_SPEED_LIMIT_

//-------------------------------------------------------------------
#define GPIO_PIN_CONFIGURATION_OUTPUT_PUSH_PULL_SPEED_LIMIT_10MHZ(PORT,
                        { \
                                GPIO##PORT->DDR |= (1 << PIN); \
                                GPIO##PORT->CR1 |= (1 << PIN); \
                                GPIO##PORT->CR2 |= (1 << PIN); \
                        }

#define GPIO_PIN_CONFIGURATION_OUTPUT_PUSH_PULL_NOSPEED_LIMIT(PORT, PIN,
                        { \
                                GPIO##PORT->DDR |= (1 << PIN); \
                                GPIO##PORT->CR1 |= (1 << PIN); \
                                GPIO##PORT->CR2 &= (uint8_t)(~(1 << PIN)
                        }

#define GPIO_PIN_CONFIGURATION_OUTPUT_OPEN_DRAIN_SPEED_LIMIT_10MHZ(PORT,
                        { \
                                GPIO##PORT->DDR |= (1 << PIN); \
                                GPIO##PORT->CR1 &= (uint8_t)(~(1 << PIN)
                                GPIO##PORT->CR2 |= (1 << PIN); \
                        }

#define GPIO_PIN_CONFIGURATION_OUTPUT_OPEN_DRAIN_NOSPEED_LIMIT(PORT, PIN
                        { \
                                GPIO##PORT->DDR |= (1 << PIN); \
                                GPIO##PORT->CR1 &= (uint8_t)(~(1 << PIN)
                                GPIO##PORT->CR2 &= (uint8_t)(~(1 << PIN)
                        }

#define GPIO_PIN_CONFIGURATION_INPUT_PULL_UP_WITH_INTERRUPT(PORT, PIN, L
                        { \
                                GPIO##PORT->DDR &= (uint8_t)(~(1 << PIN)
                                GPIO##PORT->CR1 |= (1 << PIN); \
                                GPIO##PORT->CR2 |= (1 << PIN); \
                        }

#define GPIO_PIN_CONFIGURATION_INPUT_PULL_UP_WITHOUT_INTERRUPT(PORT, PIN
                        { \
```

Vga → ROPS (Rem Object Pascal Script) - Embedded interpreter of the Pascal language. Plugin PSImport_Classes 3 → Algorithms and software solutions

Full broadcast | RSS

1-Wire Altera arduino ARM Assembler Atmel AVR C++ compel DIY enc28j60 ethernet FPGA gcc I2C IAR KEIL LaunchPad LCD led linux LPCXpresso MSP430 nxp PCB PIC pinboard2 RS-485 RTOS STM32 STM8 STM8L TI UART USB algorithm assembler ADC library power unit detail display idea tool contest competition2 LUT microcontrollers for beginners review Debug board soldering iron printed circuit board pay FPGA crafts purchases programmer programming Light-emitting diode software scheme circuit design Technologies smart House photoresist freebie crap Watch humor

Blogs

Top

AVR	38.98
STM8	37.92
Garbage truck	29.53
STM32	28.46
Detail	24.63
Connection of hardware to the computer.	24.04
Circuit design	18.15
Smart House	17.75
MSP430	17.13
LPC1xxx	14.79

All blogs

```
                                GPIO##PORT->DDR &= (uint8_t)(~(1 << PIN)
                                GPIO##PORT->CR1 |= (1 << PIN); \
                                GPIO##PORT->CR2 &= (uint8_t)(~(1 << PIN)
                        }

#define GPIO_PIN_CONFIGURATION_INPUT_FLOATING_WITH_INTERRUPT(PORT, PIN,
                        { \
                                GPIO##PORT->DDR &= (uint8_t)(~(1 << PIN)
                                GPIO##PORT->CR1 &= (uint8_t)(~(1 << PIN)
                                GPIO##PORT->CR2 |= (1 << PIN); \
                        }

#define GPIO_PIN_CONFIGURATION_INPUT_FLOATING_WITHOUT_INTERRUPT(PORT, PI
                        { \
                                GPIO##PORT->DDR &= (uint8_t)(~(1 << PIN)
                                GPIO##PORT->CR1 &= (uint8_t)(~(1 << PIN)
                                GPIO##PORT->CR2 &= (uint8_t)(~(1 << PIN)
                        }

//----------------------------------------------------------------------
#define GPIO_PIN_CONFIGURATION(PORT, PIN, LEVEL, MODE) \
                        { \
                          GPIO_PIN_CONFIGURATION_##MODE(PORT, PIN, LEVEL
                        }

//----------------------------------------------------------------------
#define GPIO_PIN_OFF_HIGH(PORT, PIN, LEVEL, MODE) \
                        { GPIO##PORT->ODR &= (uint8_t)((uint8_t)(~(1 <<

#define GPIO_PIN_OFF_LOW(PORT, PIN, LEVEL, MODE) \
                        { GPIO##PORT->ODR |= (1 << PIN); }

#define GPIO_PIN_OFF(PORT, PIN, LEVEL, MODE) \
                        { GPIO_PIN_OFF_##LEVEL(PORT, PIN, LEVEL, MODE) }

//----------------------------------------------------------------------
#define GPIO_PIN_ON_HIGH(PORT, PIN, LEVEL, MODE) \
                        { GPIO##PORT->ODR |= (1 << PIN); }

#define GPIO_PIN_ON_LOW(PORT, PIN, LEVEL, MODE) \
                        { GPIO##PORT->ODR &= (uint8_t)(~(1 << PIN)); }

#define GPIO_PIN_ON(PORT, PIN, LEVEL, MODE) \
                        { GPIO_PIN_ON_##LEVEL(PORT, PIN, LEVEL, MODE) }

//----------------------------------------------------------------------
#define GPIO_PIN_SIGNAL_HIGH(PORT, PIN, LEVEL, MODE) \
                        ( (GPIO##PORT->IDR & (1 << PIN)) == (1 << PIN) )

#define GPIO_PIN_SIGNAL_LOW(PORT, PIN, LEVEL, MODE) \
                ( (GPIO##PORT->IDR & (1 << PIN)) != (1 << PIN) )

#define GPIO_PIN_SIGNAL(PORT, PIN, LEVEL, MODE) \
                        ( GPIO_PIN_SIGNAL_##LEVEL(PORT, PIN, LEVEL, MODE

//----------------------------------------------------------------------
#define PIN_CONFIGURATION(PIN_DESCRIPTION) GPIO_PIN_CONFIGURATION(PIN_DE
#define PIN_ON(PIN_DESCRIPTION) GPIO_PIN_ON(PIN_DESCRIPTION)
#define PIN_OFF(PIN_DESCRIPTION) GPIO_PIN_OFF(PIN_DESCRIPTION)
#define PIN_SIGNAL(PIN_DESCRIPTION) GPIO_PIN_SIGNAL(PIN_DESCRIPTION)

#endif /* MCU_GPIO_H_ */
```

Note: the operating modes are declared at the very beginning, they are only necessary for auto-completion of the code. It is
very easy to work with them, it is enough to declare a line with the selected operating mode, for example, to control a green LED installed on the board, you can do it like this:

```
#define LED_GREEN E, 7, HIGH, OUTPUT_PUSH_PULL_SPEED_LIMIT_10MHZ
```

Next, configure the program and blink ;):

```
int main(void)
{
        PIN_CONFIGURATION(LED_GREEN);

        while (1)
        {
                PIN_ON(LED_GREEN);
                delay_ms(100);

                PIN_OFF(LED_GREEN);
                delay_ms(100);
        }

        return 0;
}
```

LCD from Nokia 1100
I have been looking for a small black and white indicator from a mobile phone to the controller of which you can easily solder conductors and I found it, this is the LCD from Nokia 1100. Inexpensive and can be easily bought in any cell phone repair shop or in a spare parts store. Rustling around the Internet showed that the controller for the LCD is PCF8814, according to the documentation it seems not so complicated, but for some reason I found only one source on the Internet (or rather many, but they all pointed to one http://www.sunbizhosting.com/~spiral/1100/). The original source used PIC. And somehow it is not written stylishly ;) I removed the unnecessary, added the necessary and here is what I got:
011

```
/*
 * File: lcd_nokia_1100.c
 * Date: 17.01.2011
 */

#include "main.h"

const uint8_t lcd_nokia1100_decode[] =
{ '0', '1', '2', '3', '4', '5', '6', '7', '8', '9', 'A', 'B', 'C', 'D',
                'F' };

const uint8_t lcd_nokia1100_font_5x8[96][5] =
{
        { 0x00, 0x00, 0x00, 0x00, 0x00 },// (space)
        { 0x00, 0x00, 0x5F, 0x00, 0x00 },// !
        { 0x00, 0x07, 0x00, 0x07, 0x00 },// "
        { 0x14, 0x7F, 0x14, 0x7F, 0x14 },// #
        { 0x24, 0x2A, 0x7F, 0x2A, 0x12 },// $
        { 0x23, 0x13, 0x08, 0x64, 0x62 },// %
        { 0x36, 0x49, 0x55, 0x22, 0x50 },// &
        { 0x00, 0x05, 0x03, 0x00, 0x00 },// '
        { 0x00, 0x1C, 0x22, 0x41, 0x00 },// (
        { 0x00, 0x41, 0x22, 0x1C, 0x00 },// )
        { 0x08, 0x2A, 0x1C, 0x2A, 0x08 },// *
        { 0x08, 0x08, 0x3E, 0x08, 0x08 },// +
        { 0x00, 0x50, 0x30, 0x00, 0x00 },// ,
        { 0x08, 0x08, 0x08, 0x08, 0x08 },// -
        { 0x00, 0x30, 0x30, 0x00, 0x00 },// .
        { 0x20, 0x10, 0x08, 0x04, 0x02 },// /
        { 0x3E, 0x51, 0x49, 0x45, 0x3E },// 0
        { 0x00, 0x42, 0x7F, 0x40, 0x00 },// 1
        { 0x42, 0x61, 0x51, 0x49, 0x46 },// 2
        { 0x21, 0x41, 0x45, 0x4B, 0x31 },// 3
        { 0x18, 0x14, 0x12, 0x7F, 0x10 },// 4
        { 0x27, 0x45, 0x45, 0x45, 0x39 },// 5
        { 0x3C, 0x4A, 0x49, 0x49, 0x30 },// 6
        { 0x01, 0x71, 0x09, 0x05, 0x03 },// 7
        { 0x36, 0x49, 0x49, 0x49, 0x36 },// 8
        { 0x06, 0x49, 0x49, 0x29, 0x1E },// 9
        { 0x00, 0x36, 0x36, 0x00, 0x00 },// :
        { 0x00, 0x56, 0x36, 0x00, 0x00 },// ;
        { 0x00, 0x08, 0x14, 0x22, 0x41 },// <
        { 0x14, 0x14, 0x14, 0x14, 0x14 },// =
        { 0x41, 0x22, 0x14, 0x08, 0x00 },// >
        { 0x02, 0x01, 0x51, 0x09, 0x06 },// ?
        { 0x32, 0x49, 0x79, 0x41, 0x3E },// @
        { 0x7E, 0x11, 0x11, 0x11, 0x7E },// A
        { 0x7F, 0x49, 0x49, 0x49, 0x36 },// B
        { 0x3E, 0x41, 0x41, 0x41, 0x22 },// C
        { 0x7F, 0x41, 0x41, 0x22, 0x1C },// D
        { 0x7F, 0x49, 0x49, 0x49, 0x41 },// E
        { 0x7F, 0x09, 0x09, 0x01, 0x01 },// F
        { 0x3E, 0x41, 0x41, 0x51, 0x32 },// G
        { 0x7F, 0x08, 0x08, 0x08, 0x7F },// H
        { 0x00, 0x41, 0x7F, 0x41, 0x00 },// I
        { 0x20, 0x40, 0x41, 0x3F, 0x01 },// J
        { 0x7F, 0x08, 0x14, 0x22, 0x41 },// K
        { 0x7F, 0x40, 0x40, 0x40, 0x40 },// L
        { 0x7F, 0x02, 0x04, 0x02, 0x7F },// M
        { 0x7F, 0x04, 0x08, 0x10, 0x7F },// N
        { 0x3E, 0x41, 0x41, 0x41, 0x3E },// O
        { 0x7F, 0x09, 0x09, 0x09, 0x06 },// P
        { 0x3E, 0x41, 0x51, 0x21, 0x5E },// Q
        { 0x7F, 0x09, 0x19, 0x29, 0x46 },// R
        { 0x46, 0x49, 0x49, 0x49, 0x31 },// S
        { 0x01, 0x01, 0x7F, 0x01, 0x01 },// T
        { 0x3F, 0x40, 0x40, 0x40, 0x3F },// U
        { 0x1F, 0x20, 0x40, 0x20, 0x1F },// V
        { 0x7F, 0x20, 0x18, 0x20, 0x7F },// W
        { 0x63, 0x14, 0x08, 0x14, 0x63 },// X
        { 0x03, 0x04, 0x78, 0x04, 0x03 },// Y
        { 0x61, 0x51, 0x49, 0x45, 0x43 },// Z
        { 0x00, 0x00, 0x7F, 0x41, 0x41 },// [
        { 0x02, 0x04, 0x08, 0x10, 0x20 },// "\"
        { 0x41, 0x41, 0x7F, 0x00, 0x00 },// ]
        { 0x04, 0x02, 0x01, 0x02, 0x04 },// ^
        { 0x40, 0x40, 0x40, 0x40, 0x40 },// _
        { 0x00, 0x01, 0x02, 0x04, 0x00 },// `
        { 0x20, 0x54, 0x54, 0x54, 0x78 },// a
        { 0x7F, 0x48, 0x44, 0x44, 0x38 },// b
        { 0x38, 0x44, 0x44, 0x44, 0x20 },// c
        { 0x38, 0x44, 0x44, 0x48, 0x7F },// d
        { 0x38, 0x54, 0x54, 0x54, 0x18 },// e
        { 0x08, 0x7E, 0x09, 0x01, 0x02 },// f
        { 0x08, 0x14, 0x54, 0x54, 0x3C },// g
        { 0x7F, 0x08, 0x04, 0x04, 0x78 },// h
        { 0x00, 0x44, 0x7D, 0x40, 0x00 },// i
        { 0x20, 0x40, 0x44, 0x3D, 0x00 },// j
        { 0x00, 0x7F, 0x10, 0x28, 0x44 },// k
        { 0x00, 0x41, 0x7F, 0x40, 0x00 },// l
        { 0x7C, 0x04, 0x18, 0x04, 0x78 },// m
        { 0x7C, 0x08, 0x04, 0x04, 0x78 },// n
        { 0x38, 0x44, 0x44, 0x44, 0x38 },// o
        { 0x7C, 0x14, 0x14, 0x14, 0x08 },// p
        { 0x08, 0x14, 0x14, 0x18, 0x7C },// q
        { 0x7C, 0x08, 0x04, 0x04, 0x08 },// r
        { 0x48, 0x54, 0x54, 0x54, 0x20 },// s
        { 0x04, 0x3F, 0x44, 0x40, 0x20 },// t
        { 0x3C, 0x40, 0x40, 0x20, 0x7C },// u
        { 0x1C, 0x20, 0x40, 0x20, 0x1C },// v
        { 0x3C, 0x40, 0x30, 0x40, 0x3C },// w
        { 0x44, 0x28, 0x10, 0x28, 0x44 },// x
        { 0x0C, 0x50, 0x50, 0x50, 0x3C },// y
        { 0x44, 0x64, 0x54, 0x4C, 0x44 },// z
        { 0x00, 0x08, 0x36, 0x41, 0x00 },// {
        { 0x00, 0x00, 0x7F, 0x00, 0x00 },// |
        { 0x00, 0x41, 0x36, 0x08, 0x00 },// }
        { 0x08, 0x08, 0x2A, 0x1C, 0x08 },// ->
        { 0x08, 0x1C, 0x2A, 0x08, 0x08 } // <-
};

enum
{
        LCD_NOKIA1100_SET_X_ADDRESS_LOWER_4BITS = 0x00,
        LCD_NOKIA1100_SET_X_ADDRESS_UPPER_3BITS = 0x10,
        LCD_NOKIA1100_SET_Y_ADDRESS = 0xB0,
        LCD_NOKIA1100_SET_START_ROW_ADDRESS_6BITS = 0x40,
        LCD_NOKIA1100_DIPLAY_OFF = 0xAE,
        LCD_NOKIA1100_DIPLAY_ON = 0xAF,
        LCD_NOKIA1100_PIXELS_ALL_OFF = 0xA4,
        LCD_NOKIA1100_PIXELS_ALL_ON = 0xA5,
        LCD_NOKIA1100_DISPLAY_INVERSE_ON = 0xA7,
        LCD_NOKIA1100_DISPLAY_INVERSE_OFF = 0xA6,
        LCD_NOKIA1100_CHARGE_PUMP_ON = 0x2F,
        LCD_NOKIA1100_CHARGE_PUMP_OFF = 0x28
} lcd_nokia1100_command;

//------------------------------------------------------------------------
void lcd_nokia1100_write(uint8_t data)
{
        uint8_t bit_number;

        for (bit_number = 8; bit_number-- > 0;)
        {
                PIN_OFF(LCD_NOKIA_1100_SCLK);

                if (data & 0x80)
                {
                        PIN_ON(LCD_NOKIA_1100_SDA);
                }
                else
                {
                        PIN_OFF(LCD_NOKIA_1100_SDA);
                }

                PIN_ON(LCD_NOKIA_1100_SCLK);

                data <<= 1;
        }
```

```
}
```

```
//----------------------------------------------------------------------
void lcd_nokia1100_write_command(uint8_t data)
{
        PIN_ON(LCD_NOKIA_1100_CS);

        PIN_OFF(LCD_NOKIA_1100_SCLK);
        PIN_OFF(LCD_NOKIA_1100_SDA);
        PIN_ON(LCD_NOKIA_1100_SCLK);

        lcd_nokia1100_write(data);

        PIN_OFF(LCD_NOKIA_1100_CS);
}

//----------------------------------------------------------------------
void lcd_nokia1100_write_data(uint8_t data)
{
        PIN_ON(LCD_NOKIA_1100_CS);

        PIN_OFF(LCD_NOKIA_1100_SCLK);
        PIN_ON(LCD_NOKIA_1100_SDA);
        PIN_ON(LCD_NOKIA_1100_SCLK);

        lcd_nokia1100_write(data);

        PIN_OFF(LCD_NOKIA_1100_CS);
}

//----------------------------------------------------------------------
void lcd_nokia1100_clear(void)
{
        uint8_t i;

        lcd_nokia1100_write_command(LCD_NOKIA1100_SET_START_ROW_ADDRESS_
        lcd_nokia1100_write_command(LCD_NOKIA1100_SET_X_ADDRESS_UPPER_3B
        lcd_nokia1100_write_command(LCD_NOKIA1100_SET_X_ADDRESS_LOWER_4B
        lcd_nokia1100_write_command(LCD_NOKIA1100_SET_Y_ADDRESS | 0x0);
        lcd_nokia1100_write_command(LCD_NOKIA1100_DIPLAY_OFF);

        for (i = 0; i < 216; i++)
        {
                lcd_nokia1100_write_data(0x00);
                lcd_nokia1100_write_data(0x00);
                lcd_nokia1100_write_data(0x00);
                lcd_nokia1100_write_data(0x00);
        }

        lcd_nokia1100_write_command(LCD_NOKIA1100_DIPLAY_ON);
}

//----------------------------------------------------------------------
void lcd_nokia1100_set_position(uint8_t row, uint8_t column)
{
        column *= 6;

        lcd_nokia1100_write_command(LCD_NOKIA1100_SET_Y_ADDRESS | (uint8
        lcd_nokia1100_write_command(LCD_NOKIA1100_SET_X_ADDRESS_UPPER_3B
                        | ((column >> 4) & 0x07));
        lcd_nokia1100_write_command(LCD_NOKIA1100_SET_X_ADDRESS_LOWER_4B
                        | (column & 0x0F));
}

//----------------------------------------------------------------------
```

```
void lcd_nokia1100_print_char(uint8_t symbol)
{
        uint8_t line_number;

        for (line_number = 0; line_number < 5; line_number++)
        {
                lcd_nokia1100_write_data(
                                lcd_nokia1100_font_5x8[symbol - 32][line
        }

        lcd_nokia1100_write_data(0x00);
}

//------------------------------------------------------------------------
void lcd_nokia1100_print_string(uint8_t * string)
{
        lcd_nokia1100_write_command(LCD_NOKIA1100_DIPLAY_OFF);

        while (*string)
        {
                lcd_nokia1100_print_char(*string++);
        }

        lcd_nokia1100_write_command(LCD_NOKIA1100_DIPLAY_ON);
}

//------------------------------------------------------------------------
void lcd_nokia1100_print_hex_xx(uint8_t data)
{
        lcd_nokia1100_print_char(lcd_nokia1100_decode[(data >> 4) & 0x0F
        lcd_nokia1100_print_char(lcd_nokia1100_decode[data & 0x0F]);
}

//------------------------------------------------------------------------
void lcd_nokia1100_print_dec_xxx(uint8_t data)
{
        lcd_nokia1100_print_char(lcd_nokia1100_decode[(data / 100) & 0x0
        lcd_nokia1100_print_char(lcd_nokia1100_decode[((data % 100) / 10
        lcd_nokia1100_print_char(lcd_nokia1100_decode[((data % 100) % 10
}

//------------------------------------------------------------------------
void lcd_nokia1100_print_dec_xx(uint8_t data)
{
        lcd_nokia1100_print_char(lcd_nokia1100_decode[((data % 100) / 10
        lcd_nokia1100_print_char(lcd_nokia1100_decode[((data % 100) % 10
}

//------------------------------------------------------------------------
void lcd_nokia1100_init(void)
{
        PIN_CONFIGURATION(LCD_NOKIA_1100_CS);
        PIN_CONFIGURATION(LCD_NOKIA_1100_RST);
        PIN_CONFIGURATION(LCD_NOKIA_1100_SCLK);
        PIN_CONFIGURATION(LCD_NOKIA_1100_SDA);

        PIN_ON(LCD_NOKIA_1100_RST);
        delay_ms(5);
        PIN_OFF(LCD_NOKIA_1100_RST);

        lcd_nokia1100_write_command(LCD_NOKIA1100_PIXELS_ALL_OFF);
        lcd_nokia1100_write_command(LCD_NOKIA1100_CHARGE_PUMP_ON);
        lcd_nokia1100_clear();
```

And a small example:

```
/*
 * File: main.c
 * Date: 23.01.2011
 */

#include "main.h"

//----------------------------------------------------------------------
void delay_ms(uint16_t time)
{
        volatile uint32_t i;

        while (time-- > 0)
        {
                i = 100;
                while (i-- > 0)
                {
                }
        }
}

//----------------------------------------------------------------------
int main(void)
{
        PIN_CONFIGURATION(LED_GREEN);

        lcd_nokia1100_init();
        lcd_nokia1100_set_position(7, 0);
        lcd_nokia1100_print_string("ZiBlog.ru @ 2011");
        lcd_nokia1100_set_position(0, 0);
        lcd_nokia1100_print_string("STM8L-Discovery");
        lcd_nokia1100_set_position(4, 1);
        lcd_nokia1100_print_string("LCD Nokia 1100");

        while (1)
        {
                PIN_ON(LED_GREEN);
                delay_ms(1000);

                PIN_OFF(LED_GREEN);
                delay_ms(1000);
        }

        return 0;
}
```

As a result, we get the following picture on the screen:
008

STM8 , NOKIA1100

+4 07 March 2011, 20:50 **ZiB**

Comments (36)

The GPIO system is similar to the STM32F.
PS INTERRUPT, not ITERRUPT.
PPS You should add about LCD to the tags.

0

Vga
08 March 2011, 07:15

Yeah, there is such a thing, at first I wrote for them, then I just corrected it for stm8.
I agree about spelling, this is my main problem, I often lose letters :)

0

ZiB
08 March 2011, 08:24 ↑

Thanks for the articles about STM8! I hope there will be a continuation!

0

beau monde
08 March 2011, 09:22

I have them on my site, but I can't copy and paste them directly. I'm used to writing using Windows Live Writer, you don't think about tags and other HTML stuff, but here you have to do everything manually...

0

ZiB
08 March 2011, 13:20 ↑

Live writer he himself inserts tags. Then just copy the ready html code.

But I still couldn't attach it to the site. Something worked, but it worked somehow crookedly :/

0

DIHALT
08 March 2011, 14:23 ↑

I tried a lot of things, but "visually" only it works fine, especially the insertion of pictures, code and files pleases. Directly copying is not possible due to the difference in tags and code formatting, in particular: tables are not digested in any way, text alignment does not work, with code highlighting it is not clear yet how, the editing window could be made larger, the preview at the bottom of the entry or in a separate window...

0

ZiB
08 March 2011, 14:29 ↑

brr. I don't like WLW.

0

vvzvlad
09 April 2011, 09:52 ↑

Unfortunately I haven't found anything better yet, can you suggest anything?

0

ZiB
09 April 2011, 09:59 ↑

for what, for the site? what are the advantages of WLW?

0

vvzvlad
09 April 2011, 10:01 ↑

yes, for the site.
I would like to have:
1) the ability to write offline
2) upload articles from the site and to the site
3) "write" and not deal with html
4) insert code, images, links + automatic upload to the server
5) automatic formatting of images + automatic watermark
what came to mind off the top of my head, WLW provides everything in abundance, I haven't seen any analogues yet.
p.s. I'll have to write about it sometime :)

0

ZiB
09 April 2011, 11:04 ↑

Thanks for the article on STM8.
Such a noob question: is there division in STM8? I'm very interested in what the compiler will turn these lines into:

0

angel5a
March 11, 2011, 10:55 PM

Sorry, sent earlier:
lcd_nokia1100_print_char(lcd_nokia1100_decode[((data % 100) / 10) & 0x0F]);

0

angel5a
March 11, 2011, 10:56 PM ↑

There is division, but the compiler does everything optimally so far (in my opinion).
If you divide two unsigned 16-bit numbers, then this is literally a few instructions, but when dividing eight-bit instructions, it uses its own lib which, although it uses hardware division, has a fairly large volume.
— unsigned int t_div1;
unsigned int t_div2;
t_div2 = t_div1 / 10;
20 cycles
— unsigned char t_div1;
unsigned int t_div2;
t_div2 = t_div1 / 10;
41 cycles
— unsigned char t_div1;
unsigned char t_div2;
t_div2 = t_div1 / 10;
41 cycles
— signed char t_div1;
signed char t_div2;
t_div2 = t_div1 / 10;
44 bars
- signed int t_div1;
signed int t_div2;
t_div2 = t_div1 / 10;
44 bars
- something like that...

0

ZiB
March 12, 2011, 08:57 ↑

Damn, I forgot the quotation marks:
"optimally" :)

0

ZiB
March 12, 2011, 08:57 ↑

a and unsigned long 132 cycles, signed long 137 cycles

0

ZiB
March 12, 2011, 09:01 ↑

Thank you for such valuable information, otherwise I would have gone off once again to build my division bicycles.

0

angel5a
March 12, 2011, 4:39 PM ↑

You're welcome :)
Although the topic is not very good, judging by the logs on my site, people like STM32 more.
I'm more interested in the STM8L series because of its low consumption.

0

ZiB
March 12, 2011, 6:11 PM ↑

0

Well, ARMs are interesting of course, but I'm tormented by LPC. Well, I consider stm8l as a replacement for ARMs, where high computing power is not needed and low cost is needed. What is especially good is the possibility of battery power.

angel5a
March 12, 2011, 19:33 ↑

ZiB,

Can you add a connection diagram?
... and especially with the pinout of the LSD from 1100.

I tried to do the same, through interrupts from T4.

0

Valio
09 April 2011, 05:20

see at the bottom of the page
ziblog.ru/2011/01/25/stm8l-ndash-linii-vvoda-vyivoda/

0

ZiB
09 April 2011, 08:31 ↑

Am I the only one who thinks the code for working with the screen reminds me of software i2c?

0

crab
March 28, 2012, 15:11

No, there are two of you. To the rest, it reminds of SPI. :)

0

angel5a
March 28, 2012, 15:43 ↑

No, what I mean is that this controller has hardware i2c.

0

crab
March 28, 2012, 15:54 ↑

Looked more closely. Shit, I'm being stupid, I apologize =_=.

0

crab
March 28, 2012, 15:55 ↑

You're welcome :)
There are quite a few displays with an i2c interface, only old Siemens/Motorolas.

0

angel5a
March 28, 2012, 4:03 PM ↑

I didn't quite understand, but your VDD and VDDI are tightly soldered together?
And it doesn't require any external components? Because 3310 requires an external capacitor.

0

phantom_lord
09 April 2012, 10:37

Yes, that's right.
Look at this:
ziblog.ru/2011/06/19/soglasovanie-urovney/

0

ZiB
09 April 2012, 10:50 ↑

Excellent. I just bought myself these displays yesterday. Because the 3310, firstly, requires an extra external component (condenser), and secondly, has 1 more line (D/C). Plus, there is also a backlight. Only advantages all around))

0

phantom_lord
09 April 2012, 11:06

And the resolution is also higher. Here)))

0

phantom_lord
09 April 2012, 11:07

If only 3.3 would work stably (

0

ZiB
09 April 2012, 11:08

I don't get it?! Does it require power higher than 3.3?

0

phantom_lord
09 April 2012, 11:10

Mine require 2.9 volts
and burn out at 3.3, they say it's because they're "CHINA"...

0

ZiB
09 April 2012, 11:33

Ah, well, less is not more. Power it through a diode and that's it. Although I agree that burning out from extra 0.4V is not good.

0

phantom_lord
09 April 2012, 11:35

But thanks for warning me, otherwise I am 3.3B and was going to apply to them.

0

phantom_lord
09 April 2012, 11:36

That's why I sent a link to this topic)

0

ZiB
09 April 2012, 12:12

I'll have to take a look then))) I thought it was another 5=>3.3 converter. But this is how it actually turned out)))

0

phantom_lord
09 April 2012, 12:22

Only registered and authorized users can leave comments.

Design by — Studio XeoArt

© Powered by LiveStreet CMS