STM8 microcontrollers. Input/output ports.

STM8

STM8 microcontrollers. Input/output ports.

Hello,
Today we will figure out how the STM8S I/O ports are arranged.
The number of I/O lines, naturally, varies among different controller models, and ranges from 16 (in a twenty-pin case) to 68 for microcontrollers in an LQFP-80 case. At the same time, the port lines are not the same and have different load capacities. Just to clarify, the STM8 ports are eight-bit, each port pin can be configured individually.
As an example, let's take the datasheet for the microcontroller installed in the STM8S Discovery board - STM8S105C6, in case anyone has forgotten. By the way, the principle of building documentation at ST is similar to TI's approach - there is one reference manual for the entire family, which describes all possible peripheral devices, and separate short datasheets for each crystal with a description of the pinout, peripheral content and electrical characteristics. In this case, we have a datasheet for the entire STM8S105xx family. Let's go to page number **9** and look at the table:

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

Device	STM8S105C6	STM8S105C4	STM8S105S6	STM8S105S4	STM8S105
Pin count	48	48	44	44	32
Maximum number of GPIOs	38	38	34	34	25
Ext. Interrupt pins	35	35	31	31	23
Timer CAPCOM channels	9	9	8	8	8
Timer complementary outputs	3	3	3	3	3
A/D Converter channels	10	10	9	9	7
High sink I/Os	16	16	15	15	12
Medium density Flash Program memory (bytes)	32K	16K	32K	16K	32K
Data EEPROM (bytes)	1024	1024	1024	1024	1024
RAM (bytes)	2K	2K	2K	2K	2K

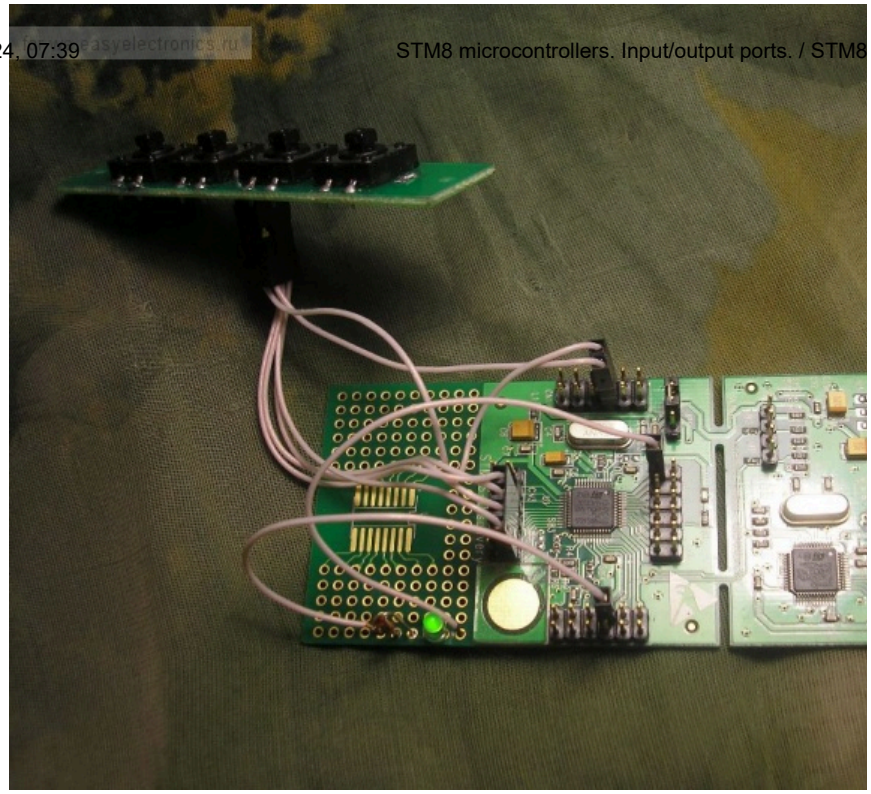
Obviously, our microcontroller has 48 pins, of which we can use 38 as input-output lines, and out of these thirty-eight pins, sixteen have an increased load capacity.
In addition, the processor pins differ in some other parameters. As an example, let's figure out what parameters the pin has, to which the LED is connected on the Dicsovery board. In this document , on page **14** , we look at the module diagram and see that the LED is connected to pin PD0 with number 41. Now let's open the previous datasheet for the controller itself and first look at table 5 on page **21.** This is the legend for table number 6, which starts on page **25.** We find the line in it that corresponds to pin number forty-one.

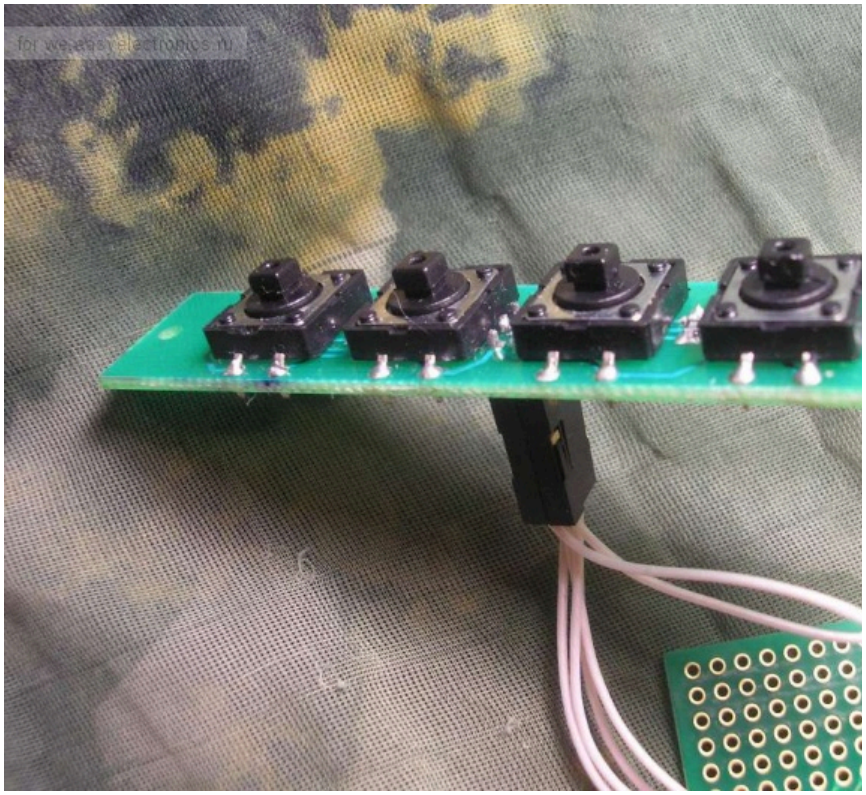
Pin number				Pin name	Type	Input			Output				Main function (after reset)	Defa func
LQFP48	LQFP44	LQFP32/ VFQFPN32/ UFQFPN32	SDIP32			floating	wpu	Ext. interrupt	High sink	Speed	OD	PP		
41	37	25	30	PD0/ TIM3_ CH2 [TIM1_ BKIN] [CLK_ CCO]	I/O	X	X	X	HS	O3	X	X	Port D0	Time char

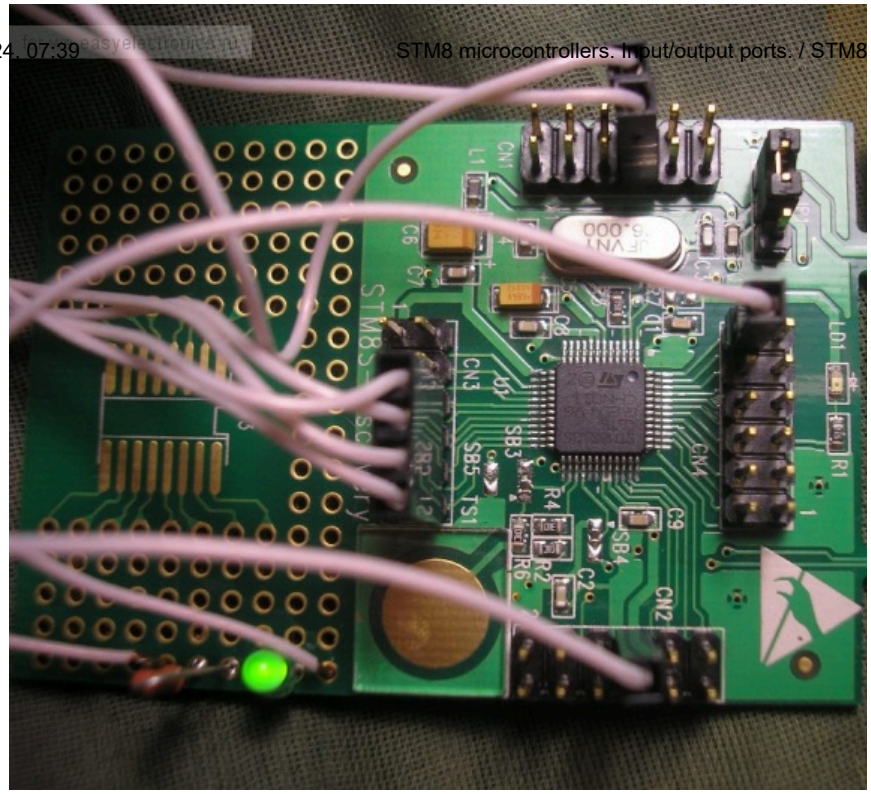
And now we will decipher it:
- this pin is in a floating state after reset (there is an underlined cross in the floating column);
— this pin has an increased load capacity (in the High Sink column, the designation is HS);
— for this pin, you can select the maximum switching speed of 2 or 10 MHz, by default 2 MHz (in the Speed column, the designation is O3);
— in addition to its main function, this pin can perform the functions of the second channel for Timer3, stop for Timer1, or output the clock signal of the microcontroller (columns Default alternate functions and Alternate functions after remap).
The main work with the input-output ports is carried out through five registers for each port. Let's consider each of them in more detail:
Register **Px_ODR** — *Port x output data register*. Data output register, analogous to the PORTx register in AVR. If the port pin is configured as an output, writing to the corresponding bit of the Px_ODR register leads to a corresponding change in the electrical state of the port pin. **Px_IDR** register — *Port x pin input register* . Data read register, analogous to the PINx register in AVR. **Px_DDR** register — *Port x data direction register* . Data direction register, analogous to the DDRx register in AVR. When writing a one to any of the register bits, the corresponding port pin is configured as an output. When writing a zero, the port pin is configured as an input. **Px_CR1** register — *Port x control register 1* . Port control register. AVR does not have an analog of

this register. This register configures the electrical properties of the port pins. If the pin is configured in input mode, writing a zero to the corresponding bit of the Px_CR1 register will enable the built-in pull-up to power. Otherwise, the input is floating. If the pin is configured as an output, writing zero to the corresponding bit of the Px_CR1 register will switch the pin to pseudo-open collector mode. Otherwise, the pin behaves as push-pull. More information about the types of inputs and outputs and how to use them is available here . **Px_CR2** register — *Port x control register 2* . Another port control register. If the pin is configured as an input, the corresponding bits enable (one) or disable (zero) the generation of an external interrupt. For a pin configured as an output, these bits select the maximum switching speed. Zero limits the speed to 2 MHz, one - to 10 MHz. For our convenience, ST has summarized all this in one table (it has already been cited in this blog, but I will repeat it): Now let's move on to practice. I soldered another LED and a pull-up for it on the breadboard area and connected it to the power supply and the PD7 pin. I connected a button to the PB0 pin, which is pulled up to ground and switches power to the pin when pressed. As you can see in the photo, I have a four-button keyboard board, but we will use one. All components of the circuit are connected using connectors. To find out what is on which connector, look at the pinout tables on page **11** in the documentation for the Discovery board. For example, the PB0 pin is located on the CN 3 block under number 10. In the photo - connections to the blocks: Let's write a simple program that constantly analyzes the state of the button. If the button is released, one LED is constantly on, if the button is pressed, it goes out and another one lights up.

Mode	DDR bit	CR1 bit	CR2 bit	Function	Pull-up	P-buffer	to
Input	0	0	0	Floating without interrupt	Off	Off	
	0	1	0	Pull-up without interrupt	On		
	0	0	1	Floating with interrupt	Off		
	0	1	1	Pull-up with interrupt	On		C
Output	1	0	0	Open drain output	Off	Off	
	1	1	0	Push pull output		On	
	1	x	1	Output speed limited to 10 MHz		Depends on CR1 bit	
	1	x	x	True open drain (on specific pins)	Not Implemented		Not plem (see







```
#include "iostm8.h"      // подключение заголовочного файла с объявления

int main( void )         // Основная программа
{

  PD_DDR_bit.DDR0 = 1;   // Ножка PD0 конфигурируется на вывод
  PD_CR1_bit.C10 = 1;   // Выход типа Push-pull
  PD_CR2_bit.C20 = 1;   // Скорость переключения - до 10 МГц.

  PD_DDR_bit.DDR7 = 1;   // Ножка PD7 конфигурируется на вывод
  PD_CR1_bit.C17 = 1;   // Выход типа Push-pull
  PD_CR2_bit.C27 = 1;   // Скорость переключения - до 10 МГц.

  PB_DDR_bit.DDR0 = 0;   // Ножка PB0 конфигурируется на ввод
  PB_CR1_bit.C10 = 0;   // Выход плавающий - у меня установлен подтягива
  PB_CR2_bit.C20 = 0;   // Прерывание отключено


  while(1)              // Бесконечный цикл
  {
        if (PB_IDR_bit.IDR0 == 1) // Проверяем состояние кнопки
        {
          PD_ODR_bit.ODR0 = 1;    // Зажигаем нужные светодиоды
          PD_ODR_bit.ODR7 = 0;    // в зависимости от состояния кнопки
        } else
        {
          PD_ODR_bit.ODR0 = 0;
          PD_ODR_bit.ODR7 = 1;
        }
  }

}
```

And a short video from Captain Obvious about how this program works. I

apologize in advance for the poor quality of the video.

That's all for today, and in the next article we will look at timers.

STM8 , contest , microcontrollers , programming

+4 March 14, 2011, 11:02 PM **Kalvenolt**

Comments (45)

RSS Collapse / Expand

Everything is fine, plus to you. But why the hell stick such a big, gaudy ST logo in every post. And also upload it again every time :) 0

DIHALT
March 15, 2011, 00:58

Unlike Masteram, ST doesn't give us any perks for advertising :) 0

DIHALT
March 15, 2011, 00:58

I like it)
And the article is immediately visible)
But since the boss asks, I'll remove it. 0

Kalvenolt
March 15, 2011, 10:33

Overall, it is informative and accessible, but there is one remark: 0

> *And now let's decipher:*
> *- this pin is in a floating state after reset;*
> *- this pin has an increased load capacity;*
> *- for this pin, you can select the maximum switching speed of 2 or 10 MHz (default 2 MHz);*
> *- in addition to its main function, this pin can perform the functions of the second channel for Timer3, stop for Timer1, or output the clock signal of the microcontroller.*

It is not obvious which point was derived from what. Perhaps when you open the datasheet and study it, it will become intuitively clear, but it would not hurt to describe it in more detail here, since we have already touched on this point.
It is just that the rest of the material, I repeat, is presented very clearly, and here you have to guess.

angel5a
March 17, 2011, 01:02

0

Added. I just didn't want to fill the article with screenshots from the datasheet.

Kalvenolt
March 17, 2011, 11:16

Good day, gentlemen. To be honest, I didn't understand the meaning of the pins with "increased load capacity". How is that? Regular pins can output/receive 20 mA, and those - more? And how much is more? No matter how hard I looked, I couldn't find it. 0

RxDTxD
08 May 2011, 10:39

normal I/O lines at 3.3 volts 4 mA supply
lines with increased load at 3.3 volts 10 mA supply
see section
10.3.6 I/O port pin characteristics
www.st.com/internet/com/TECHNICAL_RESOURCES/TECHNICAL_LITERATURE/DATASHEET/CD 0

ZiB
08 May 2011, 11:13

Another question, in the stm8s_gpio.h file there is an explanation for the 4th bit in the GPIO configuration port: 0

```
/*
 Bits definitions:
 - Bit 7: 0 = INPUT mode
          1 = OUTPUT mode
 - Bit 6: 0 = FLOAT (input) or OPEN-DRAIN (output)
          1 = PULL-UP (input) or PUSH-PULL (output)
 - Bit 5: 0 = No external interrupt (input) or No slope control (output)
          1 = External interrupt (input) or Slow control enabled (output)
 - Bit 4: 0 = Low level (output)
          1 = High level (output push-pull) or HI-Z (output open-drain)
 */
 GPIO_MODE_IN_FL_NO_IT      = (u8)0b00000000, /*! Input floating, no exte
 GPIO_MODE_IN_PU_NO_IT      = (u8)0b01000000, /*! Input pull-up, no exter
 GPIO_MODE_IN_FL_IT         = (u8)0b00100000, /*! Input floating, externa
 GPIO_MODE_IN_PU_IT         = (u8)0b01100000, /*! Input pull-up, external
 GPIO_MODE_OUT_OD_LOW_FAST  = (u8)0b10000000, /*! Output open-drain, low
 GPIO_MODE_OUT_PP_LOW_FAST  = (u8)0b11000000, /*! Output push-pull, low l
 GPIO_MODE_OUT_OD_LOW_SLOW  = (u8)0b10100000, /*! Output open-drain, low
 GPIO_MODE_OUT_PP_LOW_SLOW  = (u8)0b11100000, /*! Output push-pull, low l
 GPIO_MODE_OUT_OD_HIZ_FAST  = (u8)0b10010000, /*! Output open-drain, high
 GPIO_MODE_OUT_PP_HIGH_FAST = (u8)0b11010000, /*! Output push-pull, high
 GPIO_MODE_OUT_OD_HIZ_SLOW  = (u8)0b10110000, /*! Output open-drain, high
 GPIO_MODE_OUT_PP_HIGH_SLOW = (u8)0b11110000  /*! Output push-pull, high
```

What is the point of setting this bit from the point of view of circuitry?

does it put the pin into a high-impedance state (so as not to interfere with others)?
That is, does it only make sense when configuring the port pin to an output?

Valio
08 June 2011, 16:24

I soldered another LED and a pull-up for it on the breadboard area and connected it to the power supply and the PD7 pin... in this case, the diode lights up when the controller gives 0 to the port... right? Why not connect it to the controller and through the pull-up to the ground? Then it will light up when the port gives 1...? 0

Onion
07 February 2012, 13:36

Answers to questions: yes. Because of the moon phase. yes. 0

angel5a

07 February 2012, 14:08

Because of the moon phase.

Rather, it's a habit from the times when N-channel amplifiers in micros were much better than P-channel amplifiers.

0

Vga
07 February 2012, 15:02

and OK too :)

0

angel5a
07 February 2012, 15:41

Well, NPN were better than PNP.
In short, most chips pulled down better, or even pulled only down. That's where the habit comes from :)

0

Vga
07 February 2012, 15:57

Hello!
Please help me... I have the code

0

```
/*
        STM8S-DISCOVERY minimal blink for Cosmic C compiler
        LED is connected to high sink pin PD0 (active low)
*/
#include <iostm8s105.h> // register defines

typedef unsigned short uint16_t;
#define LED_BIT 0
#define nop()                {_asm("nop\n");}  /* No Operation */

void main()
{
        uint16_t d = 0;

        PD_DDR_bit.DDR0 = 1; // Set to output
        PD_CR1_bit.C10 |= 1; // Push-pull output


        for (;;)
        {
                // dummy delay loop


                for (d = 0; d < 40000; d++)
                {
                        nop();
                }


                PD_ODR_bit.ODR0 ^= 1;
        }
}
```

Errors are thrown:

```
----------- Project led - STM8 Cosmic - Configuration Debug -------------

Compiling main.c...
cxstm8 +debug -pxp -no -l +mods0 -pp -i"C:\Program Files (x86)\COSMIC\CXST
#error cpstm8 main.c:15(12+4) bad struct/union operand
#error cpstm8 main.c:16(12+3) bad struct/union operand
#error cpstm8 main.c:30(13+4) bad struct/union operand
#error cpstm8 main.c:16(1+10) PD_CR1_bit undefined
#error cpstm8 main.c:15(1+10) PD_DDR_bit undefined
#error cpstm8 main.c:30(2+10) PD_ODR_bit undefined
```

```
main.c:
The command: "cxstm8 +debug -pxp -no-log+mods0 -nn -i"C:\Program Files (x
exit code=1.

led.elf - 8 error(s), 0 warning(s)
```

I understand that the variables are not declared in the .h file. I tried to include iostm8.h, but my iostm8.h does not have them either.

sanek776
15 February 2012, 07:30

And if you write like this:

0

```
#include <iostm8s105.h> // register defines

typedef unsigned short uint16_t;
#define LED_BIT 0
#define nop()                {_asm("nop\n");}  /* No Operation */

void main()
{
        uint16_t d = 0;

        PD_DDR = 1; // Set to output
        PD_CR1 |= 1; // Push-pull output


        for (;;)
        {
                // dummy delay loop


                for (d = 0; d < 40000; d++)
                {
                        nop();
                }


                PD_ODR ^= 1;
        }
}
```

It all compiles, but I don't understand how to control different legs. Let's say I want to change these bits on PD2. How do I do that?

sanek776
15 February 2012, 07:36

0

```
PD_ODR ^= (1<<bitnum);
```

But it's better to find definitions of structures or write them yourself. It's more convenient after all.

angel5a
15 February 2012, 08:31

0

I understand that it would be better to find definitions... but I can’t find them anywhere...

sanek776
15 February 2012, 08:44

0

2 sanek776 — headers come with every C compiler, so most likely you're doing something wrong...
P.S. It would be nice to specify which compiler you're using, otherwise it's like "reading coffee grounds"

ChipKiller
15 February 2012, 14:49

I use STVD + Cosmic 0

sanek776
15 February 2012, 17:06

... you just need to **explicitly** include the header in the project - #include <iostm8s105.h> in the *.c file is not enough 0

ChipKiller
15 February 2012, 18:00

um... that's strange... where is this done, can you tell me? 0

sanek776
15 February 2012, 18:36

This is done in exactly the same way as adding a file to a project - through the "Workspace" window 0

ChipKiller
15 February 2012, 18:56

I already had this file in the project. But just in case, I added it again, but no reaction. I looked into the .h file itself, but there are no descriptions of these bits. 0

sanek776
15 February 2012, 19:08

miracles don't happen - I use Cosmic myself and everything is OK 0

ChipKiller
15 February 2012, 19:25

Can you send me your .h file. So that I know for sure that this is me portal. If not sloeno 0

sanek776
15 February 2012, 19:36

the file is located in the stvd\inklude\ folder and is called stm8s105c6.h 0

```
#include <iostm8s105.h> // register defines
```

why iostm8s105.h is a mystery...

ChipKiller
15 February 2012, 20:02

Help please!
I don't understand what's going on. Today I tried to flash mk on discovery, but the red SD on st-link didn't light up and stm32 which is there started to heat up like crazy and the debugger gave an error that the device is not responding 0

sanek776
19 February 2012, 08:20

Естьл проблема… Обнаружилось, что вывод, подключенный к каналу PWM таймера, работает как push-pull вне зависимости от состояния Px_CR1. Кто-нибудь знает, это можно как-то обойти (получив open-drain PWM) или нет? 0

_YS_
05 июня 2012, 00:39

Диод шоттки на выход повесить… не???))) 0

HHIMERA
05 июня 2012, 00:59

С диодом каждый дурак сможет. Я-то думал использовать эту прикольную фишечку STM8… 0

_YS_
05 июня 2012, 19:59

Сомневаюсь)))))
Дурак и МК спалит, и ШИМ не заведёт…)))))))))))))) 0

HHIMERA
05 июня 2012, 21:06

Yeah, я завел ШИМ, я не дурак. Утешили. xDDD 0

Что с внешними компонентами можно, это понятно. Я думал, что за счет такой прикольной фишки, как работа порта в режиме открытого коллектора удастся сократить количество рассыпухи на плате… Но нет.

Да, я перепроверил. В обычном режиме все ОК, открытый коллетор. Включаю ШИМ-таймер, не меняя настройки Px_CR1 — автоматом включается push-pull. Значит, и правда так задумано разработчиками ST.

_YS_
05 июня 2012, 21:35

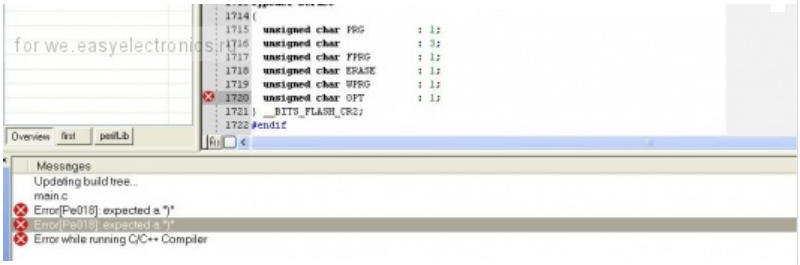
Диодом, как уже посоветовали. Некоторая переферия сама настраивает выводы для себя. Возможно таймер относится к числу такой переферии. Это долбанутый участок стм8, фиг найдешь кто настраивает и как, а кто нет. 0

angel5a
05 июня 2012, 10:28

Вот да, нигде не нашел. Ни в reference manual, ни в даташите на сам МК, ни в аппноуте по ШИМу. 0

_YS_
05 июня 2012, 20:00

Ошибка в iostm8.h: 0



Как это исправить?

nightmare
26 октября 2012, 19:48

И еще вопрос: как залить прошивку без отладки? пункт меню Project->Download всегда не активный. 0

nightmare
26 октября 2012, 19:55

Насчет Project->Download не знаю, но можно взять ST-шный прошиватор и запихнуть его в меню Tools. 0

Vga
26 октября 2012, 22:21

Не компилировал это и не знаю, но нет ли где в коде макроса OPT? 0

teplofizik

26 октября 2012, 23:44

Если заинклюдить в такой последовательности, то получаем 360 варнингов, но подобнй ошибки как на скрине нет и всё работает.
#include «iostm8.h»
#define STM8S105
#include <stm8s.h>

0

nightmare
27 октября 2012, 00:16

А так ли необходимо два хедера от разных разработчиков для одного и того же?
Как минимум пример отвратительного кода.

0

angel5a
27 октября 2012, 01:23

Как минимум пример из статьи без второго хедера не работает.

0

nightmare
27 октября 2012, 12:06

*без первого

0

nightmare
27 октября 2012, 12:06

Первый нужен из-за взятых из него структур данных **PD_DDR_bit** и иже с ним. В статье собственно по этомы и подключен **iostm8.h**.
А вот **stm8s.h** уже воткнули вы. Зачем?
Вы используйте либо базовую библиотеку IAR'a, либо STD PL.
Мешать их не следует.

0

angel5a
27 октября 2012, 16:53

Так мне удобнее с портами работать через регистры, а не с помощью тех библиотечных функций. Раз не следует, тогда не буду.

0

nightmare
27 октября 2012, 17:38

There were messages above about STDV+Cosmic not wanting to work with bit operations like:
PD_ODR_bit.ODR0 ^= 1;

I also encountered this. I added the stm8s105c6.h file to the project, it didn't help. There are no such descriptions in this file, only PD_ODR worked, but bits didn't work...

0

MikeF
June 24, 2015, 15:27

Only registered and authorized users can leave comments.