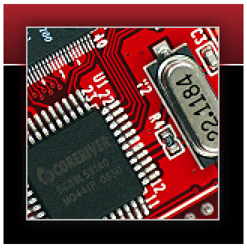
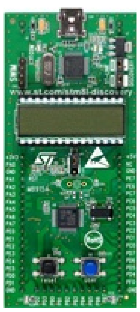


STM8L. UART Setup for Beginners

STM8



I didn't see any articles for beginners on the site about setting up and working with **UART** on **STM8L** , so I decided to make up for this shortcoming. Now I'll tell you how to set it up quickly and easily.
As a debug board with this microprocessor, I'll use **STM8L-Discovery** , which has its own **ST-Link debugger. We'll use IAR** as the programming environment .

Starting UART

So, our **STM8L152C6** processor in the default configuration is clocked by **HSI (High Speed Internal)** — a high-frequency internal RC generator with a frequency of 16 MHz. I do not have an external quartz, so we will use it.
When the microcontroller is turned on, the clock signal goes to the core and all the peripherals (timers, USART, etc.). However, in **STM8L,** clocking of all peripherals is disabled by default to reduce power consumption. Therefore, in order to work with **UART** and its registers, we first need to supply clocking to it.
The **CLK_CKDIVR**

register is used to control the system clock signal divider : We will use the maximum core frequency (16 MHz in our case) and set the last 3 least significant bits to zeros:

7	6	5	4	3	2	1
for we.easyelectronics.ru		Reserved			CKM[2:0]	
-	-	-	-	-	rw	rw

Bits 7:3 Reserved, must be kept cleared.

Bits 2:0 **CKM[2:0]**: System clock prescaler
000: System clock source/1
001: System clock source /2
010: System clock source /4
011: System clock source /8
100: System clock source /16
101: System clock source /32
110: System clock source /64
111: System clock source /128
These bits are written by software to define the system clock prescaling factor.

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

From Gorn → 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 tar spoon 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

Gilak → 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

```
CLK_CKDIVR=0x00; //делитель частоты 1
```

Next, we need to feed this frequency to the **USART** module to start working with it. The clock supply to the periphery is controlled by the **CLK_PCKENRx** register . A separate bit is responsible for turning on each device:

7	6	5	4	3	2	1
PCKEN1[7:0]						
rw	rw	rw	rw	rw	rw	rw

Peripheral clock gating bits (PCKEN 10 to PCKEN 17)

Control bit	Peripheral
PCKEN17	DAC
PCKEN16	BEEP
PCKEN15	USART1
PCKEN14	SPI1
PCKEN13	I2C1
PCKEN12	TIM4
PCKEN11	TIM3
PCKEN10	TIM2

As you can see from the picture above, to turn on **USART1,** you need to set the 5th bit of the **CLK_PCKENR1** register to one :

```
CLK_PCKENR1_bit.PCKEN15 = 1;
```

Now let's move on to setting up **USART1** . Based on the documentation for the stm8l-discovery board, we learn that pin **PC3** is **USART1_TX** , and pin **PC2** is **USART1_RX** . This means that pin **PC2** should be set to the input, and **PC3** to the output:

```
PC_DDR &= ~(1 << 2);  //PC2 RX USART1 receive (вход)
PC_DDR|=1<<3; //PC3 TX USART1 transmit (выход)
```

UART Setup

Let's configure the **USART_CR1** register :

7	6	5	4	3	2	1
R8	T8	USARTD	M	WAKE	PCEN	PS
rw	rw	rw	rw	rw	rw	rw

The word length in the frame is set by bit **M** (bit 4) of the **USART_CR1** register. If it is not set, the data length will be 8 bits. If it is set, it will be 9 bits. Let's set the standard receive/transmit mode to 8 data bits. The **PCEN** bit enables/disables the parity control mode.
The number of stop bits is configured in bits 4 and 5 of the **USART_CR3** register :

7	6	5	4	3	2	1
Reserved		STOP[1:0]		CLKEN	CPOL	CPHA
		rw		rw	rw	rw

```
00: 1 стоп бит
01: Зарезервировано
10: 2 стоп бита
11: 1.5 стоп бита (используется в режиме "Smartcard”)
```

The exchange rate (BaudRate) is determined by the value of the UART clock frequency divider register , which cannot be less than 16 and is calculated using the formula:

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 The other arduino ARM Assembler Atmel AVR C++ compel DIY enc28j60 ethernet FPGA gcc I2C AND KEIL LaunchPad LCD led linux LPCXpresso MSP430 nxp PCB PIC pinboard2 RS-485 RTOS STM32 STM8 STM8L OF UART USB algorithm assembler ADC the library power unit detail display an idea tool competition competition 2 ANGRY microcontrollers for beginners review Debug board soldering iron printed circuit board salary FPGA crafts purchases programmer programming Light-emitting diode software scheme circuit design Technologies smart House photoresist it's free crap Times humor

Blogs

Group

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

```
Tx/Rx_baud_rate = fSYSCLK / USART_DIV
```

To get the value of this divider, we must divide the clock frequency **fSYSCLK** by the desired exchange rate of 9600 bps (**Tx/Rx_baud_rate**). We will get the required value of the division coefficient **USART_DIV** :

```
16000000ГЦ/9600bps=1666,6 (0x0683 в 16-ричной с округлением)
```

The two bytes of **the USART_DIV** value are written first to the **USART_BRR2** register and then to **USART_BRR1** .

```
USART1_BRR2 = 0x03;
USART1_BRR1 = 0x68;
```

Setting **the UART** pins to **TX** and **RX** mode instead of general purpose ports is done using the **USART2_CR2** register :

7	6	5	4	3	2	1	
TIEN	TCIEN	RIEN	ILIEN	TEN	REN	RWU	S
rw	rw	rw	rw	rw	rw	rw	

```
USART1_CR2_bit.REN=1; //прием
USART1_CR2_bit.TEN=1; //передача
```

Data transfer mode

In this mode, the **USART_DR** register consists of the **TDR** buffer and the transmitter shift register, which transmits the frame directly to the **TX** line . As soon as we have data in the **USART_DR** register , it is immediately moved to the **TDR** buffer register , and from there to the shift register. The **TXE** flag is set in the **USART_SR** register when **the TDR** buffer content has been transferred to the shift register and transmission has begun. Therefore, the **TDR** buffer register is empty and the next data can be written to the **USART_DR** register without overwriting **the TDR** buffer register . This means that when trying to write to the **USART_DR** register when the **TXE** flag is cleared, the buffer register will not be empty (since the previous byte in the shift register has not finished sending), and the pending byte in the **TDR buffer will be overwritten. This flag generates an interrupt if the TIEN bit of the USART1_CR2** register is set .

```
0: Данные не переданы регистр сдвига
1: Данные переданы в сдвиговый регистр
```

If the frame transmission in the shift register is completely completed and the **TXE** flag is set (the **TDR** buffer is clear), the **TC flag of the USART_SR** register is set . This means that the transmission of any data via **UART** is completely completed . There is also an interrupt for this flag if the **TCIEN bit of the USART1_CR2** register is set .

```
0: Передача не завершена
1: Передача завершена
```

In order to transmit data, we must wait until the **TDR** buffer is empty and the **TXE flag of the USART_SR** register is set.

```
while(!(USART1_SR_bit.TXE)); //Ожидаем освобождения буферного регистра T
USART1_DR='F'; // Отправляем символ "F".
```

Status register (USART_SR)

Address offset: 0x00
Reset value: 0xC0

7	6	5	4	3	2	1
TXE	TC	RXNE	IDLE	OR	NF	FE
r	rc_w0	r	r	r	r	r

Data reception mode

Now let's look at receiving. In this mode, the **USART_DR** register consists of the **RDR** buffer and the receiver shift register reading data from the RX leg. When data from the shift register is moved to the **RDR buffer register, the RXNE flag of the USART_SR** register is set . This data becomes available in the **USART_DR register. The RXNE** flag is cleared when an attempt is made to read **the USART_DR** register . An interrupt is also generated if the **RIEN** bit is set in the **USART_CR2** register . We enable this interrupt:

```
USART1_CR2_RIEN=1; //Прерывание по приему
#pragma vector=USART_R_OR_vector
     __interrupt void USART1_RXE(void)
     {
          USART1_DR=USART1_DR; //Отправляем тоже, что и приняли (эхо)
     }
```

Example

Finally, putting it all together. Here is the complete code for initializing **the UART** , sending the "F" character at startup, and the "echo" receive mode.

```

#include <iostm8l152c6.h>
#include <intrinsics.h>  //Здесь описана функция __enable_interrupt ().

int main()
{
  CLK_CKDIVR=0x00; //делитель частоты 1
  CLK_PCKENR1_bit.PCKEN15 = 1; //Включаем тактирование модуля USART1

  PC_DDR &= ~(1 << 2); //PC2 RX USART1 receive (вход)
  PC_DDR|=1<<3; //PC3 TX USART1 transmit (выход)

  USART1_BRR2 = 0x03; //скорость 9600bps
  USART1_BRR1 = 0x68;

  USART1_CR2_bit.REN=1; //прием
  USART1_CR2_bit.TEN=1; //передача

  USART1_CR2_RIEN=1; //Прерывание по приему
  __enable_interrupt (); // Разрешаем прерывания.

  while(!(USART1_SR_bit.TXE)); //Ожидаем освобождения буферного регистра
  USART1_DR='F'; // Отправляем символ "F".

  while (1){} //беск. цикл

}

#pragma vector=USART_R_OR_vector
__interrupt void USART1_RXE(void)
{
          USART1_DR=USART1_DR; //Отправляем тоже, что и приняли (эхо)
}
```

UART, USART , STM8L, STM8L-Discovery

+6 April 13, 2014, 11:44 PM **Poloten4ik**

Comments (12)

RSS Collapse / Expand

Thanks for the info!

0

JokerDVB
April 14, 2014, 12:25

Thanks for the info, but there is some inaccuracy:

> *PC_DDR|=0<<2; //PC2 RX USART1 receive (вход)*

The entry makes no sense, because in fact it turns out PC_DDR |= 0; Which will not give any changes to PC_DDR.
And if the bit was set before, it will not be asked. So it is better to do it this way:
PC_DDR &= ~(1 << 2);

0

shevmax
April 14, 2014, 11:57 PM

Thank you, I'll take that into account.

0

Poloten4ik
April 15, 2014, 10:55 PM ↑

So fix it already - this shameful construction has been hanging for more than a year, and in two places.

0

Law
May 20, 2015, 9:29 pm ↑

Seriously, fix it!
Otherwise, newbies might do it, and if it doesn't work, they'll look for a cat where there isn't one!

0

zelya
02 February 2017, 08:58 ↑

He hasn't come here for two years.

0

Vga
02 February 2017, 16:31 ↑

Calm down, I'm here. I got an email with Achtung after your comments today. I'll fix it now)

0

Poloten4ik
02 February 2017, 21:21 ↑

Corrected.
It was originally

```
PC_DDR|=0<<2; //PC2 RX USART1 receive (вход)
  PC_DDR|=1<<3; //PC3 TX USART1 transmit (выход
```

Just in case... I hope I fixed everything like that. If something is wrong, shout

0

Poloten4ik
02 February 2017, 21:22 ↑

If I understand the logic of this register correctly, then the second line should not have been removed. Only the first one should have been replaced.

0

Vga
03 February 2017, 04:25 ↑

exactly.
the first one is now correct, it only resets 1 bit related to PC2,
and the second line sets 1 bit related to PC3.

0

zelya
03 February 2017, 14:51 ↑

Oops. Sorry.
Fixed .

0

Poloten4ik
03 February 2017, 20:34 ↑

Thank you for the article

0

megannn98
April 15, 2014, 07:56

Only registered and authorized users can leave comments.