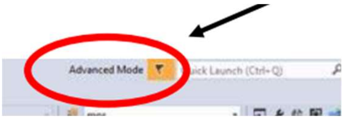
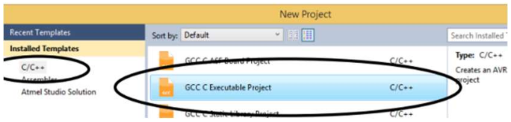
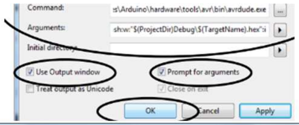
1. 

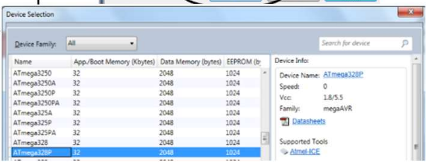
2. main menu -> Tools -> External Tools

3. main menu and create new project



Command: C:\Program Files (x86)\Arduino\hardware\tools\avr\bin\avrdude.exe
Argument: -C"C:\Program Files (x86)\Arduino\hardware\tools\avr\etc\avrdude.conf" -v -v -patmega328p -carduino -P\\.***COM3*** -b115200 -D -Uflash:w:"$(ProjectDir)Debug\$(TargetName).hex":i


Command: s\Arduino\hardware\tools\avr\bin\avrdude.exe
Arguments: sh:w:"$(ProjectDir)Debug\$(TargetName).hex":i
Initial directory:
☑ Use Output window ☑ Prompt for arguments
☐ Treat output as Unicode ☑ Close on exit

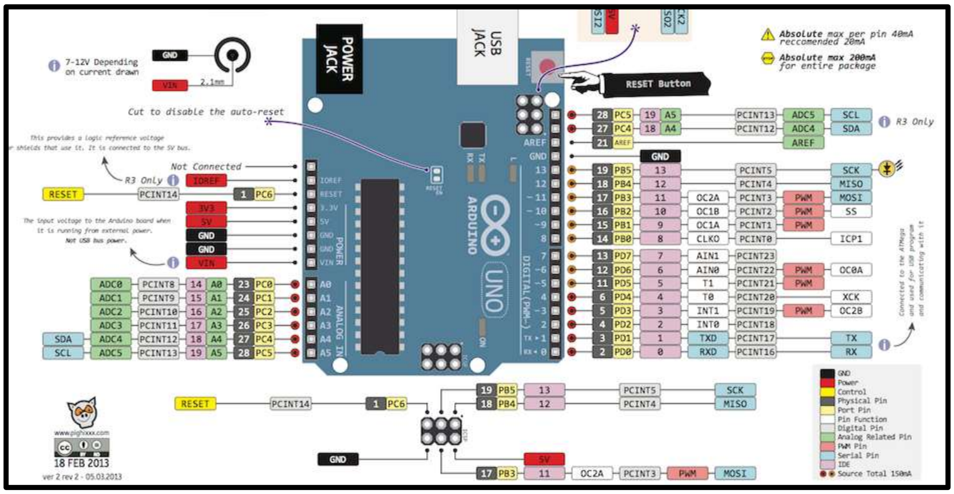

Device Selection
Device Family: All

Name	App./Boot Memory (Kbytes)	Data Memory (bytes)	EEPROM (b
ATmega3250	32	2048	1024
ATmega3250A	32	2048	1024
ATmega3250P	32	2048	1024
ATmega3250PA	32	2048	1024
ATmega325A	32	2048	1024
ATmega325P	32	2048	1024
ATmega325PA	32	2048	1024
ATmega328	32	2048	1024
ATmega328P	32	2048	1024

Device Info:
Device Name: ATmega328P
Speed: 0
Vcc: 1.8/5.5
Family: megaAVR
Datasheets
Supported Tools
Atmel-ICE

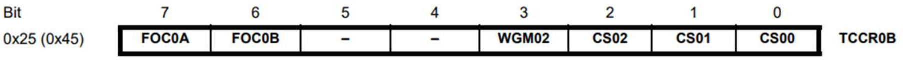
Basic GPIO output

```
#include <avr/io.h>

int main(void){
    DDRD = 0xFF; // Set all pins of
port D as outputs
    while (1){
        PORTD = 0x55; // Set value
of port D to 0b01010101
    }
}
```

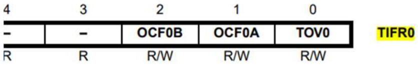

7-12V Depending on current drawn
GND
VIN
2.1mm
POWER JACK
USB JACK
RESET Button
Absolute max per pin 40mA reccomended 20mA
Absolute max 200mA for entire package
Cut to disable the auto-reset
This provides a logic reference voltage for shields that use it. It is connected to the 5V bus.
Not Connected
R3 Only
IOREF
RESET
PCINT14
1 PC6
3V3
5V
GND
GND
VIN
The input voltage to the Arduino board when it is running from external power. Not USB bus power.
ADC0 PCINT8 14 A0 23 PC0
ADC1 PCINT9 15 A1 24 PC1
ADC2 PCINT10 16 A2 25 PC2
ADC3 PCINT11 17 A3 26 PC3
SDA ADC4 PCINT12 18 A4 27 PC4
SCL ADC5 PCINT13 19 A5 28 PC5
ARDUINO
UNO
ANALOG IN
POWER
DIGITAL (PWM~)
AREF
28 PC5 19 A5 PCINT13 ADC5 SCL
27 PC4 18 A4 PCINT12 ADC4 SDA
21 AREF AREF
GND
19 PB5 13 PCINT5 SCK
18 PB4 12 PCINT4 MISO
17 PB3 11 OC2A PCINT3 PWM MOSI
16 PB2 10 OC1B PCINT2 PWM SS
15 PB1 9 OC1A PCINT1 PWM
14 PB0 8 CLKO PCINT0 ICP1
13 PD7 7 AIN1 PCINT23
12 PD6 6 AIN0 PCINT22 PWM OC0A
11 PD5 5 T1 PCINT21 PWM
6 PD4 4 T0 PCINT20 XCK
5 PD3 3 INT1 PCINT19 PWM OC2B
4 PD2 2 INT0 PCINT18
3 PD1 1 TXD PCINT17 TX
2 PD0 0 RXD PCINT16 RX
R3 Only
Connected to the ATmega and used for USB programming and communicating with it
RESET PCINT14 1 PC6
19 PB5 13 PCINT5 SCK
18 PB4 12 PCINT4 MISO
GND
5V
17 PB3 11 OC2A PCINT3 PWM MOSI
ICSP
GND
Power
Control
Physical Pin
Port Pin
Pin Function
Digital Pin
Analog Related Pin
PWM Pin
Serial Pin
IDE
Source Total 150mA
www.pighixxx.com
18 FEB 2013
ver 2 rev 2 - 05.03.2013

Basic GPIO input

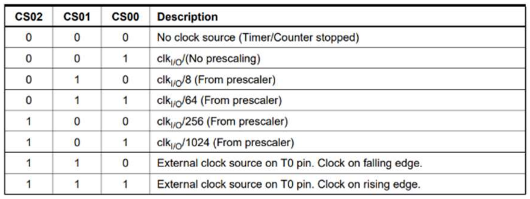
```
……
int main(void){
    DDRD = 0xFF;
    DDRB = 0x00;
    while (1){
        if ((PINB & (1<<PINB0))){ //
If input B0 is high
            PORTD = 0xFF;
        }else{
            PORTD = 0x00;
        }
    }
}
```



Bit	7	6	5	4	3	2	1	0	
0x25 (0x45)	FOC0A	FOC0B	–	–	WGM02	CS02	CS01	CS00	TCCR0B



4	3	2	1	0	
–	–	OCF0B	OCF0A	TOV0	TIFR0
R	R	R/W	R/W	R/W	



CS02	CS01	CS00	Description
0	0	0	No clock source (Timer/Counter stopped)
0	0	1	$clk_{I/O}$/(No prescaling)
0	1	0	$clk_{I/O}$/8 (From prescaler)
0	1	1	$clk_{I/O}$/64 (From prescaler)
1	0	0	$clk_{I/O}$/256 (From prescaler)
1	0	1	$clk_{I/O}$/1024 (From prescaler)
1	1	0	External clock source on T0 pin. Clock on falling edge.
1	1	1	External clock source on T0 pin. Clock on rising edge.

Delay using Timer 0 (non-CTC)

f = CLK(XTAL) / Prescaler / num of cnt

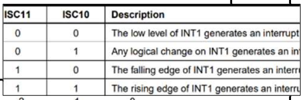
```
……
void delay_1s(void){
    char i = 62;
    for(;i>0;i--){
        TCNT0 = 0x04;
        TCCR0B = 0x05; // Timer Counter Control Register 0B
        while( (TIFR0&(1<<TOV0)) == 0x00 ){} // Wait Timer/Counter 0
Overflow Flag (TOV0)
        TCNT0 = 0x00; // Reset Timer/Counter Register
        TCCR0B = 0x00; // Stop the Timer/Counter 0
        TIFR0 = (1<<TOV0); // Clear the Timer/Counter 0 Overflow Flag by
writing 1
    }
}
```

Delay using Timer 0 (CTC)

```
……(for(){)
        TCNT0 = 0x00;
        TCCR0B = 0x0D; // WGM[2:0]=010: CTC
        OCR0A = 252; // Output compare target
        while( (TIFR0&(1<<OCF0A)) == 0x00 ){}
        TCCR0B = 0x00;
        TIFR0 = (1<<OCF0A);
……
```

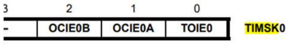
Timer 1 as external counter (a 16-bit counter)

```
……
    while (1){
        TCCR1B = 0x0F;
        OCR1A = 2; // Set OCF1A every 3 rising edge
        if (TIFR1&(1<<OCF1A)){
            PORTC ^= 0x01;
            TIFR1 |= (1<<OCF1A);
        }
    }
……
```

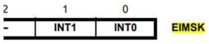


ISC11	ISC10	Description
0	0	The low level of INT1 generates an interrupt
0	1	Any logical change on INT1 generates an in
1	0	The falling edge of INT1 generates an interr
1	1	The rising edge of INT1 generates an interru

Timer interrupt



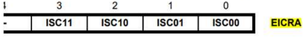
3	2	1	0	
-	OCIE0B	OCIE0A	TOIE0	TIMSK0



2	1	0	
-	INT1	INT0	EIMSK

```
#include <avr/io.h>
#include "avr/interrupt.h" // !! Don't forget !!

int ms = 0;
int main(void)
{
    DDRC |= 0x31;  // PC 0, 4, 5 as outputs
    PORTC = 0x00;

    TCCR0B = 0x0B;  // Timer 0: with prescaler 64, CTC, 1ms
    OCR0A = 250;
    TCCR1B = 0x0F;  // Timer 1: ext. CLK input, CTC
    OCR1A = 2;

    TIMSK0 = (1<<OCIE0A); // Timer 0 compare match A interrupt
    TIMSK1 = (1<<OCIE1A); // Timer 1 compare match A interrupt
    sei(); // Enable global interrupts !! Don't forget !!
    while (1){}
}

ISR(TIMER1_COMPA_vect){
    PORTC ^= 0x01;
}

ISR(TIMER0_COMPA_vect){ // TIMER0_OVF_vect -- TIMSK0=(1<<TOIE0)
    PORTC ^= 0x20;
    ms++;
……
```

External GPIO interrupt



3	2	1	0	
ISC11	ISC10	ISC01	ISC00	EICRA

```
……
#include "avr/interrupt.h" // !! Don't forget !!
……
    TCCR0B = 0x0B;  // Timer 0: prescaler 64, CTC, 1ms
    OCR0A = 250;

    TIMSK0 = (1<<OCIE0A); //Tim0 compare match A Intrupt
    EIMSK = 0x01;  // Enable external interrupt INT0
    EICRA |= 0x03;  // INT0 trigger on rising edge
    sei(); // Enable global interrupt !! Don't forget !!
……
ISR(INT0_vect){
    count++;
    if (count>=3){ count = 0; PORTC ^= 0x01;}
}
……
```

Bit	7	6	5	4	3	2	1	0	
	RXCn	TXCn	UDREn	FEn	DORn	UPEn	U2Xn	MPCMn	UCSRnA
Read/Write	R	R/W	R	R	R	R	R/W	R/W	
Initial Value	0	0	1	0	0	0	0	0	

R/TX complete, UDR Emp, Frame Err, Data OverRun, Parity Err, 2xspeed, mul-pr

Bit	7	6	5	4	3	2	1	0	
	RXCIEn	TXCIEn	UDRIEn	RXENn	TXENn	UCSZn2	RXB8n	TXB8n	UCSRnB
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R	R/W	

R/TX/C/UDRE

7	6	5	4	3	2	1	0	
UMSELn1	UMSELn0	UPMn1	UPMn0	USBSn	UCSZn1	UCSZn0	UCPOLn	UCSRnC
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

Intrupt EN….

UPMn1	UPMn0	Parity Mode
0	0	Disabled
0	1	Reserved
1	0	Enabled, Even Parity
1	1	Enabled, Odd Parity

UMSELn1	UMSELn0	Mode
0	0	Asynchronous USART
0	1	Synchronous USART
1	0	(Reserved)
1	1	Master SPI (MSPIM)[1]

UCSZn2	UCSZn1	UCSZn0	Character Size
0	0	0	5-bit
0	0	1	6-bit
0	1	0	7-bit
0	1	1	8-bit
1	1	1	9-bit
	else		Reserved

USBSn	Stop Bit(s)
0	1-bit
1	2-bit

UCPOLn	Transmitted Data Changed (Output of TxDn Pin)	Received Data Sar Pin)
0	Rising XCKn Edge	Falling XCKn Edge
1	Falling XCKn Edge	Rising XCKn Edge

Transmit string and receive string (USART0, polling)

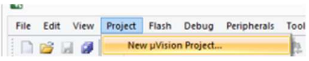
```
#include <avr/io.h>

void transmit_char(char ch){
    UDR0 = ch;
    while (!(UCSR0A&(1<<UDRE0))){}
}
void transmit_str(char ch[30], int length){
    for (int i=0; i<length; i++){
        UDR0 = ch[i];
        while (!(UCSR0A&(1<<UDRE0))){}
    }
}
int main(void)
{
    unsigned char rx_buffer[2] = "  ";
    UCSR0B = (1<<RXEN0) | (1<<TXEN0);
    UCSR0C = 0x06;
    UBRR0 = 208-1;
    ……
}
```

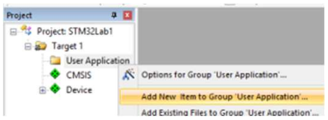
```
#include <avr/io.h>

int main(void)
{
    unsigned char c = 0;
    UCSR0B = (1<<RXEN0) | (1<<TXEN0); // Enable RX TX
    UCSR0C = 0x06;     // Asynchronous, 8 data bits, no parity, 1 stop bit
    UBRR0 = 208-1;  // baud rate 4800 (for 16MHz CLK)
    // UBRR(12bit)=(Fosc / (16*(Desired Baud Rate))-1
    while (1)
    {
        while (!(UCSR0A&(1<<RXC0))){} // Wait till a char received
        c = UDR0;  // Read the received char from UDR0
        UDR0 = c;  // Write (Transmit) the received char
        while (!(UCSR0A&(1<<UDRE0))){}; // Wait till UDR0 is empty (char transmitted)
    }
}
```

Trans/receive str (USART0, interrupt)

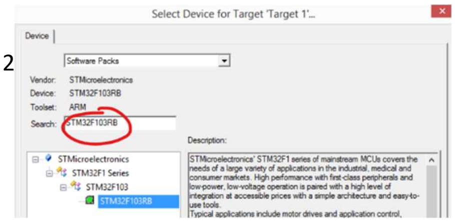
```
#include <avr/io.h>
#include <avr/interrupt.h>  // !! Don't forget !!
#include <string.h>  // !! if you want to use strcpy !!
void transmit_char(char ch){
    UDR0 = ch;
}
char tx_buffer[30] = "We are ready!";
ISR(USART_UDRE_vect){
    if (tx_buffer[0] != 0){  // If there's a char to trans
        transmit_char(tx_buffer[0]);
    }
    for(int i=0; i<29; i++){  // shift tx_buffer by 1
        tx_buffer[i] = tx_buffer[i+1];
    }
    tx_buffer[29] = 0;
}
ISR(USART_RX_vect){
    c = UDR0;
    rx_buffer[0] = rx_buffer[1];
    rx_buffer[1] = c;
    if (rx_buffer[0]=='H' && rx_buffer[1]=='i'){
        strcpy(tx_buffer, "Bye!");
    }
}
int main(void)
{
    UCSR0B = (1<<RXEN0)|(1<<TXEN0)|(1<<RXCIE0)|(1<<UDRIE0);
    ……
    sei(); // Enable global interrupt !! Don't forget !!
……
```

STM32

1


4

Project
Project: STM32Lab1
Target 1
User Application
CMSIS
Device
Options for Group 'User Application'…
Add New Item to Group 'User Application'…
Add Existing Files to Group 'User Application'…

(After programming)

2

Select Device for Target 'Target 1'…

Device

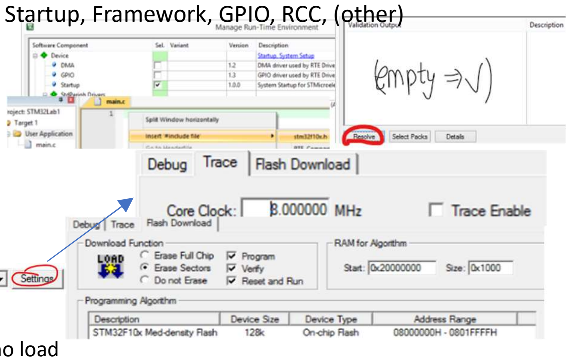
Software Packs

Vendor: STMicroelectronics
Device: STM32F103RB
Toolset: ARM
Search: STM32F103RB

STMicroelectronics
- STM32F1 Series
  - STM32F103
    - STM32F103RB

Description:
STMicroelectronics' STM32F1 series of mainstream MCUs covers the needs of a large variety of applications in the industrial, medical and consumer markets. High performance with first-class peripherals and low-power, low-voltage operation is paired with a high level of integration at accessible prices with a simple architecture and easy-to-use tools.
Typical applications include motor drives and application control,

3 Startup, Framework, GPIO, RCC, (other)

5

Manage Run-Time Environment

Software Component	Sel.	Variant	Version	Description
Device				Startup, System Setup
DMA	☐		1.2	DMA driver used by RTE Drive
GPIO	☐		1.3	GPIO driver used by RTE Drive
Startup	☑		1.0.0	System Startup for STMicroele
StdPeriph Drivers				

Validation Output / Description: (empty ⇒√)

Debug | Trace | Flash Download

Core Clock: 8.000000 MHz ☐ Trace Enable

Debug | Trace | Flash Download

Download Function: LOAD ○ Erase Full Chip ● Erase Sectors ○ Do not Erase ☑ Program ☑ Verify ☑ Reset and Run

RAM for Algorithm: Start: 0x20000000 Size: 0x1000

Programming Algorithm

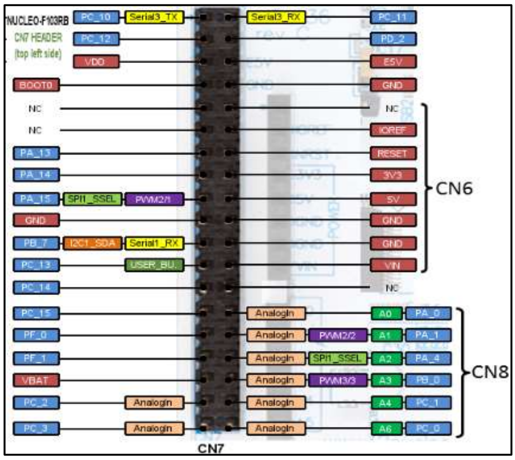
Description	Device Size	Device Type	Address Range
STM32F10x Med-density Flash	128k	On-chip Flash	08000000H - 0801FFFFH

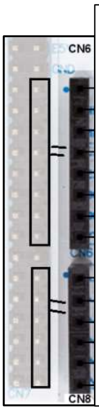
Flash ➔ Configure Flash Tools, Debug ➔ choose ST-Link Debugger

● Use: ST-Link Debugger

____

PA0~15, PB0~15, PC0~15, PD0~2 (51 Pins), PC13~15 Only one can be an output, 2 MHz, no load





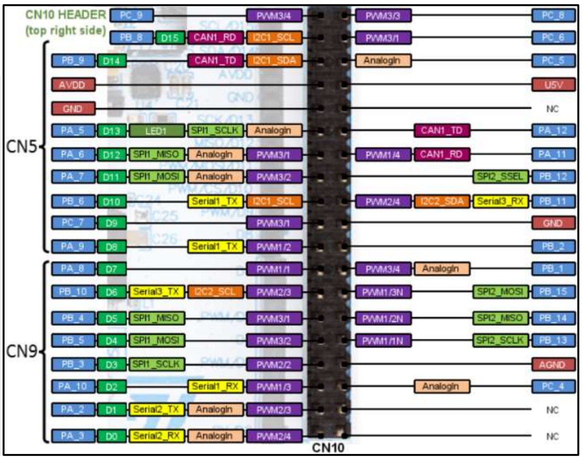
4 Timer:
 3GP,
 1Adv-Ctrl

Communica
 SPI (2)
 I2C (2)
 USART (3)
 USB (1)
 CAN (1)

GPIOs (51)
 With EXTI

syn ADC
 (12-bit)

Package:
LQFP64


CN10 HEADER
(top right side)
PC_9 PWM3/4
PWM3/3 PC_8
PB_8 D15 CAN1_RD I2C1_SCL
PWM3/1 PC_6
PB_9 D14 CAN1_TD I2C1_SDA
AnalogIn PC_5
AVDD
U5V
GND
NC
PA_5 D13 LED1 SPI1_SCLK AnalogIn
CAN1_TD PA_12
PA_6 D12 SPI1_MISO AnalogIn PWM3/1
PWM1/4 CAN1_RD PA_11
PA_7 D11 SPI1_MOSI AnalogIn PWM3/2
SPI2_SSEL PB_12
PB_6 D10 Serial1_TX I2C1_SCL
PWM2/4 I2C2_SDA Serial3_RX PB_11
PC_7 D9 PWM3/1
GND
PA_9 D8 Serial1_TX PWM1/2
PB_2
PA_8 D7 PWM1/1
PWM3/4 AnalogIn PB_1
PB_10 D6 Serial3_TX I2C2_SCL PWM2/3
PWM1/3N SPI2_MOSI PB_15
PB_4 D5 SPI1_MISO PWM3/1
PWM1/2N SPI2_MISO PB_14
PB_5 D4 SPI1_MOSI PWM3/2
PWM1/1N SPI2_SCLK PB_13
PB_3 D3 SPI1_SCLK PWM2/2
AGND
PA_10 D2 Serial1_RX PWM1/3
AnalogIn PC_4
PA_2 D1 Serial2_TX AnalogIn PWM2/3
NC
PA_3 D0 Serial2_RX AnalogIn PWM2/4
NC
CN5
CN9
CN10

C语言数据类型

关键字	位数	表示范围	stdint关键字	ST关键字
char	8	-128 ~ 127	int8_t	s8
unsigned char	8	0 ~ 255	uint8_t	u8
short	16	-32768 ~ 32767	int16_t	s16
unsigned short	16	0 ~ 65535	uint16_t	u16
int	32	-2147483648 ~ 2147483647	int32_t	s32
unsigned int	32	0 ~ 4294967295	uint32_t	u32
long	32	-2147483648 ~ 2147483647		
unsigned long	32	0 ~ 4294967295		
long long	64	-(2^64)/2 ~ (2^64)/2-1	int64_t	
unsigned long long	64	0 ~ (2^64)-1	uint64_t	
float	32	-3.4e38 ~ 3.4e38		
double	64	-1.7e308 ~ 1.7e308		

关键字： #define （宏定义）
用途：用一个命名代替一个值，便于理解；
　　提取经常出现的参数，便于修改
定义宏定义： #define ABC 12345
引用宏定义： int a = ABC; //等效于 int a = 12345;

关键字：typedef
将一个比较长的变量类型名换个名字，便于使用
定义 typedef： typedef unsigned char uint8_t;
引用 typedef： uint8_t a; //等效于 unsigned char a;

关键字：struct （数据打包，不同类型变量的集合）
定义结构体变量： struct {char x; int y; float z} StructName;
因为结构体变量类型较长，所以通常用 typedef 更改变量类型名
引用结构体成员： StructName.x = 'A'; StructName.y = 66;
或 pStructName->z = 1.23; //pStructName 为结构体的地址

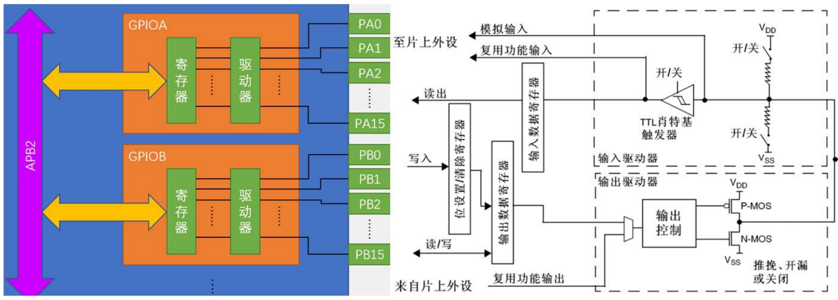
关键字：enum 定义一取值受限制的整型变量；宏定义的集合
定义枚举变量： enum{FALSE = 0, TRUE = 1} EnumName;
因为枚举变量类型较长，所以通常用 typedef 更改变量类型名
引用枚举成员： EnumName = FALSE; EnumName = TRUE;

```
int y = 7; //&: get mem addr
int *yPtr; //yPtr: y's mem addr
yPtr = &y; //yPtr points to y
*yPtr == 7; //*： get value
// aPtr == &a == 0012F580
// *aPtr == a == 7
// &aPtr == 2000
// *aPtr = 9 <=> (auto) a = 9
```

```
#define __IO volatile
```

volatile is a qualifier that is applied to a variable when it is declared. Tells the compiler variable may change at any time (without any action by the code the compiler finds nearby)

GPIO



模式名称	性质	特征
浮空输入	数字输入	可读取引脚电平，若引脚悬空，则电平不确定
上拉输入	数字输入	可读取引脚电平，内部连接上拉电阻，悬空时默认高电平
下拉输入	数字输入	可读取引脚电平，内部连接下拉电阻，悬空时默认低电平
模拟输入	模拟输入	GPIO无效，引脚直接接入内部ADC
开漏输出	数字输出	可输出引脚电平，高电平为高阻态，低电平接VSS
推挽输出	数字输出	可输出引脚电平，高电平接VDD，低电平接VSS
复用开漏输出	数字输出	由片上外设控制，高电平为高阻态，低电平接VSS
复用推挽输出	数字输出	由片上外设控制，高电平接VDD，低电平接VSS

APB2ENR

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	USAR T1EN	Res.	SPI1 EN	TIM1 EN	ADC2 EN	ADC1 EN	Reserved		IOPE EN	IOPD EN	IOPC EN	IOPB EN	IOPA EN	Res.	AFIO EN
	rw		rw	rw	rw	rw			rw	rw	rw	rw	rw		rw

LCKR – Lock Register

	6
Reserved	KK
	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LCK15	LCK14	LCK13	LCK12	LCK11	LCK10	LCK9	LCK8	LCK7	LCK6	LCK5	LCK4	LCK3	LCK2	LCK1	LCK0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

GPIO Output (Register Oriented)

```
#include "stm32f10x.h"                   // Device header
int main(void){
    RCC->APB2ENR |= RCC_APB2Periph_GPIOA; //Enable APB2 periph clock
    GPIOA->CRL &= ~0x00F00000; // clear the setting
    GPIOA->CRL |= 0<<22 | 2<<20; //GPIO_Mode_Out_PP, GPIO_Speed_2MHz
    while(1){
        GPIOA->BRR ^= 0x00; // Reset "Reset" Register
        GPIOA->BSRR |= 0x20; delay(1000); // Set A5 to 1 using BSRR
        GPIOA->BSRR &= 0x00; // Reset "Set" Register
```

Exhaustive Delay

```
void delay(int t) {               // delay by t ms
    int i; int j;
    for (i = 0; i < t; i++) {
        for (j=0;j<2666 ;j++)
        { GPIOA->BSRR ^= 0x00;} // do something to PA0

    }
}
```

Delay via Systick Interrupt (with GPIO I/O via ST lib)

```
static __IO uint32_t msTicks;
void DelayMs(uint32_t ms){
    msTicks = ms; // Reload us value
    while (msTicks){}; // Wait until msTicks reaches zero
}
// SysTick_Handler function will be called every 1 ms
void SysTick_Handler(){
    if (msTicks != 0){msTicks--;}
}
int main(void){
    RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOA, ENABLE);
    RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOC, ENABLE);


    GPIO_InitTypeDef GPIO_InitStructure;
    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_5;
    GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_Out_PP;
    GPIO_Init(GPIOA, &GPIO_InitStructure);

    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_13;
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IPD;
    GPIO_Init(GPIOC, &GPIO_InitStructure);

    SystemCoreClockUpdate(); // Update SystemCoreClock value
    sysTick_Config(SystemCoreClock / 1000);  // the SysTick timer overflow every 1 ms

    while(1){
        GPIO_WriteBit(GPIOA, GPIO_Pin_5,
Bit_SET^(BitAction)GPIO_ReadOutputDataBit(GPIOA, GPIO_Pin_5));
        DelayMs(1000);
        GPIO_WriteBit(GPIOA, GPIO_Pin_5,
(BitAction)(1^GPIO_ReadInputDataBit(GPIOC, GPIO_Pin_13)));
    }
}
```

GPIOx->CRL

Port bit configuration table

Configuration mode		CNF1	CNF0	MODE1	MODE0	PxODR register
General purpose output	Push-pull	0	0	01 10 11		0 or 1
	Open-drain		1			0 or 1
Alternate Function output	Push-pull	1	0			don't care
	Open-drain		1			don't care
Input	Analog	0	0	00		don't care
	Input floating		1			don't care
	Input pull-down	1	0			0
	Input pull-up					1

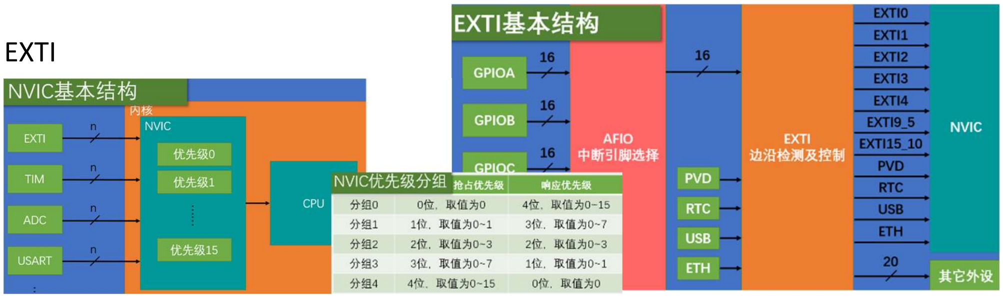
Output MODE bits

MODE[1:0]	Meaning
00	Reserved
01	Max. output speed 10 MHz
10	Max. output speed 2 MHz
11	Max. output speed 50 MHz

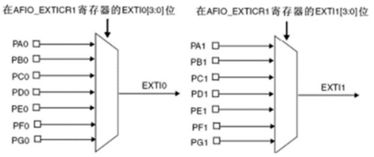
GPIO Read Button (Software debouncing)

```
uint8_t readKey(GPIO_TypeDef *GPIOx, uint16_t GPIO_Pin, uint8_t commonState){
    uint8_t pressed = 0;
    if (commonState ^ GPIO_ReadInputDataBit(GPIOx, GPIO_Pin)){
        DelayMs(20);
        while (commonState ^ GPIO_ReadInputDataBit(GPIOx, GPIO_Pin)){}
        DelayMs(20);
        pressed = 1;
    }
    return pressed;
}
```

EXTI

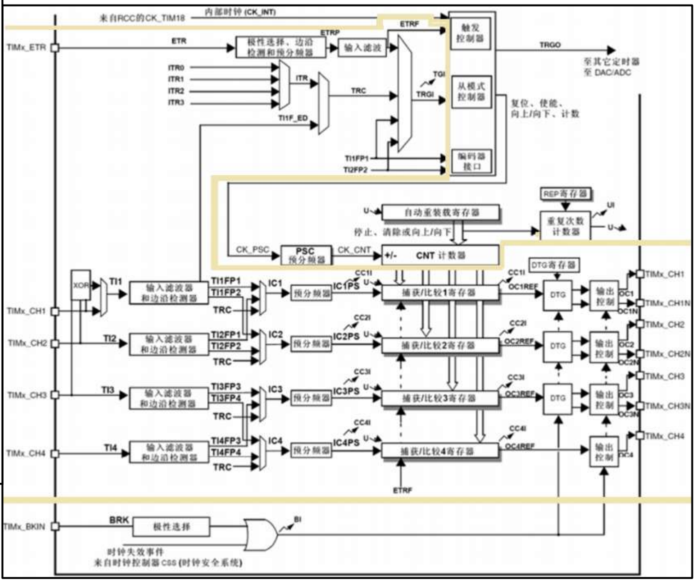


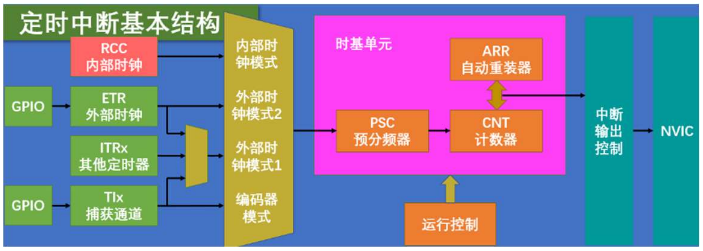
NVIC优先级分组	抢占优先级	响应优先级
分组0	0位，取值为0	4位，取值为0~15
分组1	1位，取值为0~1	3位，取值为0~7
分组2	2位，取值为0~3	2位，取值为0~3
分组3	3位，取值为0~7	1位，取值为0~1
分组4	4位，取值为0~15	0位，取值为0


在AFIO_EXTICR1寄存器的EXTI0[3:0]位
PA0
PB0
PC0
PD0
PE0
PF0
PG0
EXTI0
在AFIO_EXTICR1寄存器的EXTI1[3:0]位
PA1
PB1
PC1
PD1
PE1
PF1
PG1
EXTI1

```
void EXTI15_10_IRQHandler(void) {
    if (EXTI_GetITStatus(EXTI_Line13) != RESET) {
        ……
        EXTI_ClearITPendingBit(EXTI_Line13);
    }
}
int main(void){
    RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOA, ENABLE);
    RCC_APB2PeriphClockCmd(RCC_APB2Periph_AFIO, ENABLE);
    …… // GPIO config
    GPIO_EXTILineConfig(GPIO_PortSourceGPIOC, GPIO_PinSource13);
    EXTI_InitTypeDef EXTI_InitStruct;
    EXTI_InitStruct.EXTI_Line = EXTI_Line13;
    EXTI_InitStruct.EXTI_Mode = EXTI_Mode_Interrupt;
    EXTI_InitStruct.EXTI_Trigger = EXTI_Trigger_Rising;
    EXTI_InitStruct.EXTI_LineCmd = ENABLE;
    EXTI_Init(&EXTI_InitStruct);

    NVIC_InitTypeDef NVIC_InitStruct;
    NVIC_InitStruct.NVIC_IRQChannelCmd = ENABLE;
    NVIC_InitStruct.NVIC_IRQChannel = EXTI15_10_IRQn;
    NVIC_InitStruct.NVIC_IRQChannelPreemptionPriority = 0x02;
    NVIC_Init(&NVIC_InitStruct);
……
}
```

Timer

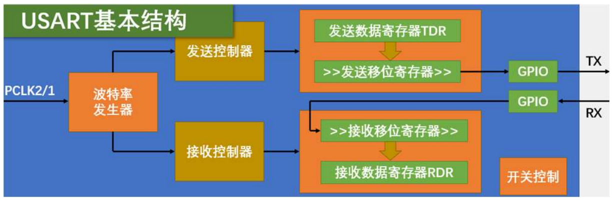




External Timer Clock

```
int main(void){
    RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOA, ENABLE);
    RCC_APB1PeriphClockCmd(RCC_APB1Periph_TIM2, ENABLE);
    ...... // GPIO config
    TIM_TimeBaseInitTypeDef timerInitStructure;
    timerInitStructure.TIM_Prescaler = 0;
    timerInitStructure.TIM_CounterMode =
TIM_CounterMode_Up;
    timerInitStructure.TIM_Period = 3-1;
    timerInitStructure.TIM_ClockDivision = 0;
    timerInitStructure.TIM_RepetitionCounter = 0;
    TIM_TimeBaseInit(TIM2, &timerInitStructure);
    TIM_Cmd(TIM2, ENABLE);
    TIM_TIxExternalClockConfig(TIM2,
TIM_TIxExternalCLK1Source_TI1, TIM_ICPolarity_Rising, 0);

    TIM_ITConfig(TIM2, TIM_IT_Update, ENABLE);
    NVIC_EnableIRQ(TIM2_IRQn);
    while(1){}
}
void TIM2_IRQHandler(void){
    if (TIM_GetITStatus(TIM2, TIM_IT_Update) != RESET) {
        ......
        TIM_ClearITPendingBit(TIM2, TIM_IT_Update);
    }
}
```

USART



USART TX/RX (via polling)

```
int main(void){
  RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOA|
RCC_APB2Periph_AFIO, ENABLE);
  // Tx pin
  GPIO_InitStructure.GPIO_Pin = GPIO_Pin_2;
  GPIO_InitStructure.GPIO_Speed = GPIO_Speed_2MHz;
  GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AF_PP;
  GPIO_Init(GPIOA, &GPIO_InitStructure);
  // Rx pin
  GPIO_InitStructure.GPIO_Pin = GPIO_Pin_3;
  GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IN_FLOATING;
  GPIO_Init(GPIOA, &GPIO_InitStructure);

  RCC_APB1PeriphClockCmd(RCC_APB1Periph_USART2,ENABLE);
  USART_InitTypeDef USART_InitStructure;
  USART_InitStructure.USART_BaudRate = 4800;
  USART_InitStructure.USART_WordLength = USART_WordLength_8b;
  USART_InitStructure.USART_StopBits = USART_StopBits_1;
  USART_InitStructure.USART_Parity = USART_Parity_No;
  USART_InitStructure.USART_HardwareFlowControl =
USART_HardwareFlowControl_None;
  USART_InitStructure.USART_Mode = USART_Mode_Rx | USART_Mode_Tx;
  USART_Init(USART2, &USART_InitStructure);
  USART_Cmd(USART2, ENABLE);

  uint8_t ch=0; uint8_t counter = 0;
  while(1){
    while(USART_GetFlagStatus(USART2, USART_FLAG_RXNE)==RESET){
      while(USART_GetFlagStatus(USART2, USART_FLAG_TC) == RESET);
      if(ch==0){USART_SendData(USART2, 'a');
      }else if (counter){USART_SendData(USART2, ch);counter--;}
    }
    if (ch==0){
      ch = USART_ReceiveData(USART2) & 0xFF;
      counter = 10;
    } //USART_ClearITPendingBit(USART2, USART_IT_RXNE); auto clears
  }
}
```