CSE3103 Microprocessor and Microcontroller
STM32F446: CLOCK, GPIO, Timer & USART

Dr. Mosaddek Tushar, Professor

Computer Science and Engineering, University of Dhaka

January 30, 2024

Contents

Configure the Clock

Clock and Source

- HSI – Internal Crystal
- HSE – External Crystal – 8MHz (for STM32F446)
- External Clock Input
- PLL clock source
- Sysclock – RTC – Timing
- Clock Power Enable
- $SystemCoreClock = ((INPUT\_CLOCK(HSE\_OR\_HSI\_IN\_HZ)/PLL\_M) * PLL\_N)/PLL\_P$

Clock Pre-scalar

- PLL_M, PLL_N, PLL_P

Peripheral Bus Must Clock Enable

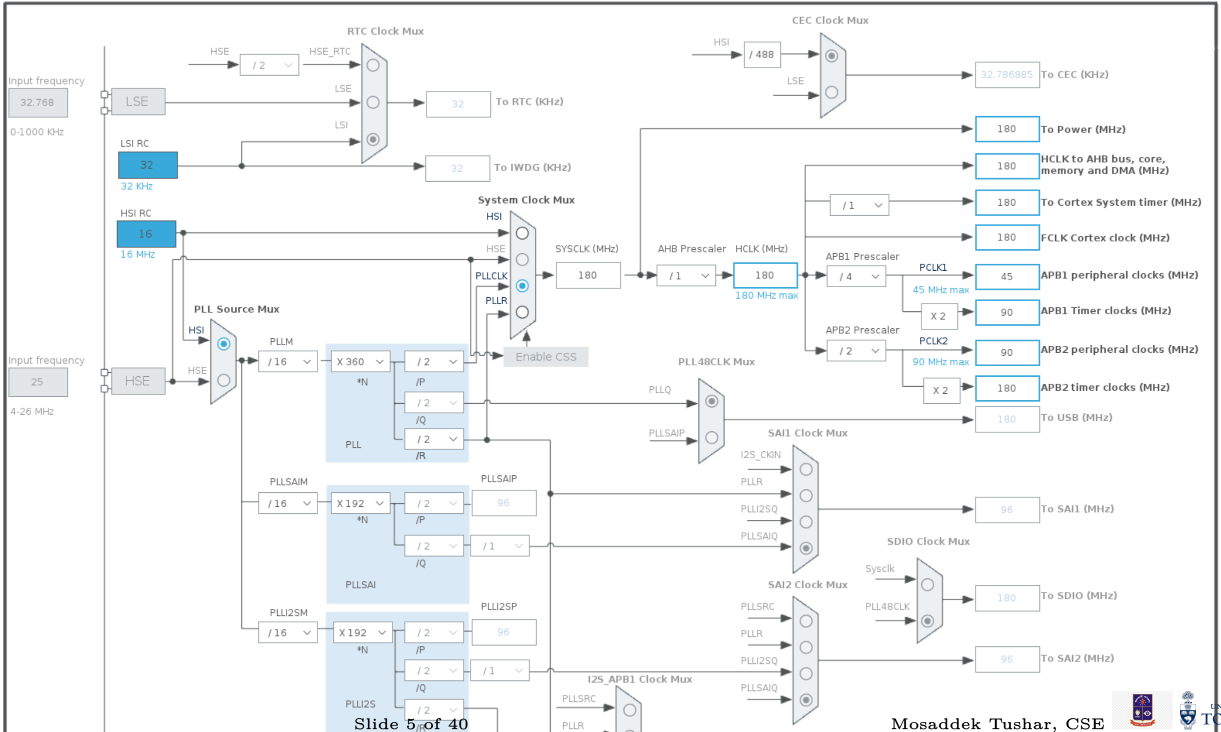
- AHB1 – 180 MHz (max)
- APB1 – 45 MHz
- APB2 – 90 MHz

Example Clock Configuration – Clock for all and clock power

Example Clock to be set (Specification)

- AHB – 180 MHz
- APB1 – 45MHz
- APB2 – 90 MHz
- HSE external crystal 8 MHz
- Use PLL clock as system clock source
- Set FLASH Memory latency and Cache enabled Access
- Enable clock power

Clock Architecture



Clock Registers

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	PLLSAI RDY	PLLSAI ON	PLLI2S RDY	PLLI2S ON	PLL RDY	PLL ON	Res.	Res.	Res.	Res.	CSS ON	HSE BYP	HSE RDY	HSE ON
		r	rw	r	rw	r	rw					rw	rw	r	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
HSICAL[7:0]								HSITRIM[4:0]					Res.	HSI RDY	HSI ON
r	r	r	r	r	r	r	r	rw	rw	rw	rw	rw		r	rw

Figure 2: RCC_CR – Control Register

- HSI ON, HSE ON, PLL ON, and . . . to enable oscillator and check the status ([x] RDY)
- HSI clock calibration (HSICAL)
- HSI Trimming to adjust variation of frequency and temperature

– –**Setting** —

- RCC_CR[16] = 1
- Check if Bit - 17 is set.

Later After PLL setting is complete

- RCC_CR[24] = 1
- check if bit - 25 is set.

PLL CLK CKT Configuration Register

***For detail see reference manual of stm32f446re

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	PLLR[2:0]			PLLQ[3:0]				Res.	PLLSRC	Res.	Res.	Res.	Res.	PLLP[1:0]	
	rw	rw	rw	rw	rw	rw	rw		rw					rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	PLLN[8:0]									PLLM[5:0]					
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Figure 3: RCC PLLCFGR Register – PLL configuration

- Prescaler: PLLR, PLLQ, PLLP, PLLN, PLLM –
  - PLLR – Main PLL division factor for I2Ss, SAIs, SYSTEM and SPDIF-Rx clocks ($2 \leq PLLR \leq 7$)
  - PLLQ – Main PLL (PLL) division factor for USB OTG FS, SDIOclocks ($2 \leq PLLQ \leq 15$)
  - PLLP [2-bit] – Main PLL (PLL) division factor for main system clock [‘00’→2, ‘01’→4, ‘10’→6, or ‘11’→8]
  - PLLN – Main multiplication factor $50 \leq PLLN \leq 432$
  - PLLM – Division factor for the main PLL (PLL) input clock $2 \leq PLLM \leq 63$
- PLLSRC – PLL Clock source HSE or HSI oscillator clock selected

Settings . . . can you do it? Try . . .

RCC Clock Configuration Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MCO2[1:0]		MCO2 PRE[2:0]			MCO1 PRE[2:0]			Res.	MCO1		RTCPRE[4:0]				
rw		rw	rw	rw	rw	rw	rw		rw		rw	rw	rw	rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
PPRE2[2:0]			PPRE1[2:0]			Res.	Res.	HPRE[3:0]				SWS[1:0]		SW[1:0]	
rw	rw	rw	rw	rw	rw			rw	rw	rw	rw	r	r	rw	rw

Figure 4: RCC clock configuration register (RCC_CFGR)

- HPRE, PPRE1, PPRE2 – AHB, APB1, APB2 prescaler – Think and link values with PLLCLK
- Whst is RTCPRE, find out.
- Micrcontroller clock output ?? Other bits find it in reference manual
- SWS – System Clock Switch Status
- SW – 2-bits: system clock switch HSE, HSI, PLL_P or PLL_R

RCC – RSTR Reset Peripherals AHB1, AHB2, AHB3, APB1, APB2

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	OTGHS RST	Res.	Res.	Res.	Res.	Res.	Res.	DMA2 RST	DMA1 RST	Res.	Res.	Res.	Res.	Res.
		rw							rw	rw					

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	CRC RST	Res.	Res.	Res.	Res.	GPIOH RST	GPIOG RST	GPIOF RST	GPIOE RST	GPIOD RST	GPIOC RST	GPIOB RST	GPIOA RST
			rw					rw	rw	rw	rw	rw	rw	rw	rw

Figure 5: RCC AHB1 Reset Register for connected peripheral registers

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	DAC RST	PWR RST	CECRST	CAN2 RST	CAN1 RST	FMPI2C1 RST	I2C3 RST	I2C2 RST	I2C1 RST	UART5 RST	UART4 RST	UART3 RST	UART2 RST	SPDIFRX RST
		rw	rw	rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SPI3 RST	SPI2 RST	Res.	Res.	WWDG RST	Res.	Res.	TIM14 RST	TIM13 RST	TIM12 RST	TIM7 RST	TIM6 RST	TIM5 RST	TIM4 RST	TIM3 RST	TIM2 RST
rw	rw			rw			rw	rw	rw	rw	rw	rw	rw	rw	rw

Figure 6: APB1 Connected Peripheral Reset Register (APB1RSTR)

Peripheral Clock Enable Registers – for AHB1, AHB2, AHB3, APB1, APB2

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	OTGHS ULPIEN	OTGHS EN	Res.	Res.	Res.	Res.	Res.	Res.	DMA2 EN	DMA1 EN	Res.	Res.	BKP SRAMEN	Res.	Res.
	rw	rw							rw	rw			rw		
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	CRC EN	Res.	Res.	Res.	Res.	GPIOH EN	GPIOG EN	GPIOF EN	GPIOE EN	GPIOD EN	GPIOC EN	GPIOB EN	GPIOA EN
			rw					rw	rw	rw	rw	rw	rw	rw	rw

Figure 7: RCC AHB1 Enable register for Connected Peripherals

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	DAC EN	PWR EN	CEC EN	CAN2 EN	CAN1 EN	FMPI2C1 EN	I2C3 EN	I2C2 EN	I2C1 EN	UART5 EN	UART4 EN	USART3 EN	USART2 EN	SPDIFRX EN
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
SPI3 EN	SPI2 EN	Res.	Res.	WWDG EN	Res.	Res.	TIM14 EN	TIM13 EN	TIM12 EN	TIM7 EN	TIM6 EN	TIM5 EN	TIM4 EN	TIM3 EN	TIM2 EN
rw	rw			rw			rw	rw	rw	rw	rw	rw	rw	rw	rw

Figure 8: RCC APB1 Enable Register for Connected Peripherals

Peripheral Clock Low Power mode Enable Registers – for AHB1, AHB2, AHB3, APB1, APB2

******More registers – use it on demand (Reference Manual)**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	OTGHS ULPI LPEN	OTGHS LPEN	Res.	Res.	Res.	Res.	Res.	Res.	DMA2 LPEN	DMA1 LPEN	Res.	Res.	BKP SRAM LPEN	SRAM2 LPEN	SRAM1 LPEN
	rw	rw							rw	rw			rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FLITF LPEN	Res.	Res.	CRC LPEN	Res.	Res.	Res.	Res.	GPIOH LPEN	GPIOG LPEN	GPIOF LPEN	GPIOE LPEN	GPIOD LPEN	GPIOC LPEN	GPIOB LPEN	GPIOA LPEN
rw			rw					rw	rw	rw	rw	rw	rw	rw	rw

Figure 9: RCC AHB1 peripheral clock enable in low power mode register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SAI2 LPEN	SAI1 LPEN	Res.	Res.	Res.	TIM11 LPEN	TIM10 LPEN	TIM9 LPEN
								rw	rw				rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	SYSCFG LPEN	SPI4 LPEN	SPI1 LPEN	SDIO LPEN	ADC3 LPEN	ADC2 LPEN	ADC1 LPEN	Res.	Res.	USART6 LPEN	USART1 LPEN	Res.	Res.	TIM8 LPEN	TIM1 LPEN
	rw	rw	rw	rw	rw	rw	rw			rw	rw			rw	rw

Figure 10: RCC APB2 Clock Enable Register for Low power mode

Flash Memory Configuration and Latency

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	DCRST	ICRST	DCEN	ICEN	PRFTEN	Res.	Res.	Res.	Res.	LATENCY			
			rw	w	rw	rw	rw					rw	rw	rw	rw

Figure 11: Flash Memory Access Control Registers (FLASH_ACR)

*****There are more registers – use according to your need**

Power Control Register – Clock/Peripheral Needs Power

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FISSR	FMSSR	UDEN[1:0]		ODSWEN	ODEN
										rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VOS[1:0]		ADCDC1	Res.	MRUDS	LPUDS	FPDS	DBP	PLS[2:0]			PVDE	CSBF	CWUF	PDDS	LPDS
rw	rw	rw		rw	rw	rw	rw	rw	rw	rw	rw	rc_w1	rc_w1	rw	rw

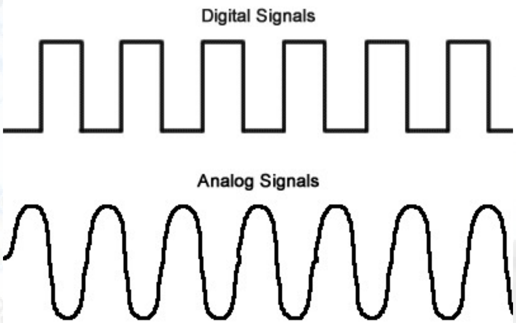
Figure 12: Power Control Register (Power SRC MNGT connected to APB1)

Bit 14-15 Regulator voltage scaling (1,2, and 3) output selection – See Reference manual. These bits can be modified while PLL is off

General Purpose Input output

General purpose input output

- Input/Output pins are the entrances of the microprocessor to communicate with the outside world.
  - Analog In/Out Communication
  - Digital In/Out communication



GPIO Registers Set (All 32-bit Registers)

- Four Configuration Registers
  - GPIO Mode (IN or OUT) register
  - GPIO output type
  - GPIO Output Speed
  - Output PushPull or Open Drain
- Two Data Registers
  - GPIO Input Data Register
  - GPIO Output Data Register
- One lock register
- Two Alternate Function Register

GPIO Functional Description

- Input floating
- Input pull-up
- Input-pull-down
- Analog
- Output open-drain with pull-up or pull-down capability

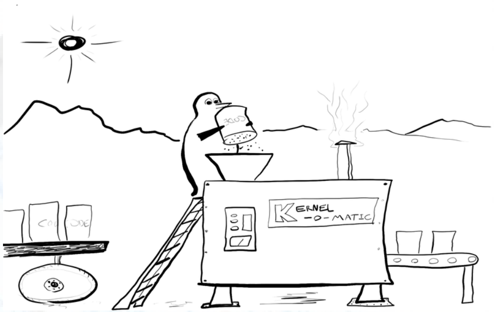


Figure 13: Input and Output

GPIO Functional Description (cont.)

- Output push-pull with pull-up or pull-down capability
- Alternate function push-pull with pull-up or pull-down capability
- Alternate function open-drain with pull-up or pull-down capabilit

GPIO Mode Register

The mode register is used to configure GPIOx either for input or output

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MODER15[1:0]		MODER14[1:0]		MODER13[1:0]		MODER12[1:0]		MODER11[1:0]		MODER10[1:0]		MODER9[1:0]		MODER8[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
MODER7[1:0]		MODER6[1:0]		MODER5[1:0]		MODER4[1:0]		MODER3[1:0]		MODER2[1:0]		MODER1[1:0]		MODER0[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Figure 14: GPIOx Mode Register (32-bits)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
OT15	OT14	OT13	OT12	OT11	OT10	OT9	OT8	OT7	OT6	OT5	OT4	OT3	OT2	OT1	OT0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Figure 15: GPIOx Configuration Register for setting output type (PushPull or Open Drain)

GPIOx Output Type

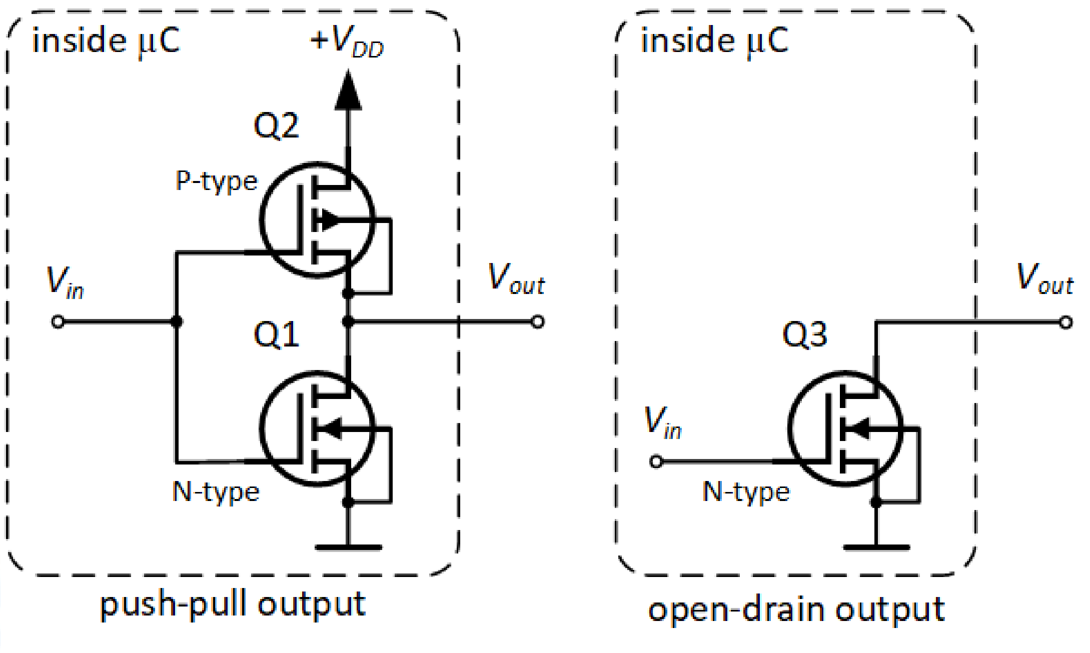


Figure 16: GPIOx output circuit

Pull up and Pull Down Resistor

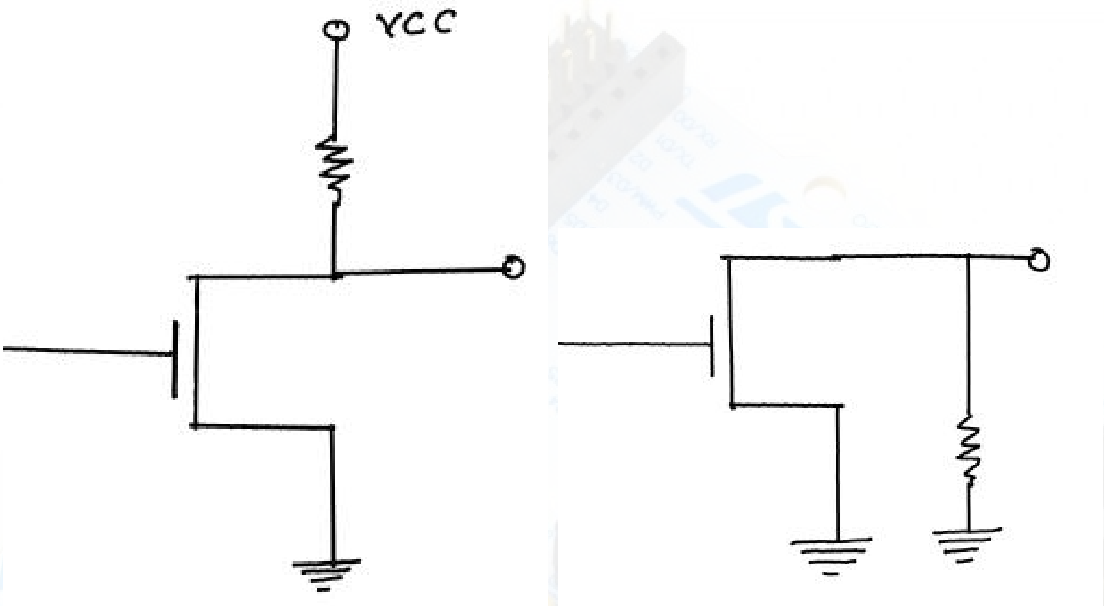


Figure 17: Pull Up and Pull Down resistor

GPIOx Output Speed Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
OSPEEDR15 [1:0]		OSPEEDR14 [1:0]		OSPEEDR13 [1:0]		OSPEEDR12 [1:0]		OSPEEDR11 [1:0]		OSPEEDR10 [1:0]		OSPEEDR9 [1:0]		OSPEEDR8 [1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
OSPEEDR7 [1:0]		OSPEEDR6 [1:0]		OSPEEDR5 [1:0]		OSPEEDR4 [1:0]		OSPEEDR3[1:0]		OSPEEDR2 [1:0]		OSPEEDR1 [1:0]		OSPEEDR0 1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Figure 18: GPIOx Speed Configuration Register (00→ Low Speed, 00→ Medium Speed, 10→ Fast Speed, 11→ High Speed; See Datasheet for detail)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PUPDR15[1:0]		PUPDR14[1:0]		PUPDR13[1:0]		PUPDR12[1:0]		PUPDR11[1:0]		PUPDR10[1:0]		PUPDR9[1:0]		PUPDR8[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
PUPDR7[1:0]		PUPDR6[1:0]		PUPDR5[1:0]		PUPDR4[1:0]		PUPDR3[1:0]		PUPDR2[1:0]		PUPDR1[1:0]		PUPDR0[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Figure 19: Pull up and pull down register for setting internal pull-up and pull-down resistor

GPIO input and output data register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
IDR15	IDR14	IDR13	IDR12	IDR11	IDR10	IDR9	IDR8	IDR7	IDR6	IDR5	IDR4	IDR3	IDR2	IDR1	IDR0
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Figure 20: GPIOx Input Data Register IDR; Get ‘1’ when input is active or on

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
ODR15	ODR14	ODR13	ODR12	ODR11	ODR10	ODR9	ODR8	ODR7	ODR6	ODR5	ODR4	ODR3	ODR2	ODR1	ODR0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Figure 21: GPIOx Output Data Register ODR (Each bit dedicated to a GPIOx pin); Pin output is high when the corresponding bit is set

GPIOx Set Reset Register (BSRR)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BR15	BR14	BR13	BR12	BR11	BR10	BR9	BR8	BR7	BR6	BR5	BR4	BR3	BR2	BR1	BR0
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BS15	BS14	BS13	BS12	BS11	BS10	BS9	BS8	BS7	BS6	BS5	BS4	BS3	BS2	BS1	BS0
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Figure 22: GPIOx Set and Reset Register (first 16 bit for set, higher 16-bits for reset); If set corresponding ODR bit is set and pin output is high

Alternate Function for GPIOx

Two 32-bit Register for 15-pin (GPIOx)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
AFRL7[3:0]				AFRL6[3:0]				AFRL5[3:0]				AFRL4[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AFRL3[3:0]				AFRL2[3:0]				AFRL1[3:0]				AFRL0[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Figure 23: GPIOx Alternate Function for USART, I2C, SPI and so on (See datasheet for alternate function of a peripheral operation, such as USART1: '0xAF7'); Dedicated for GPIOx pins from 0 to 7

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
AFRH15[3:0]				AFRH14[3:0]				AFRH13[3:0]				AFRH12[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AFRH11[3:0]				AFRH10[3:0]				AFRH9[3:0]				AFRH8[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Figure 24: GPIOx Alternate function High Register for GPIOx pins from 8 to 15

Timer of STM32F446re

What are the purposes of timer

- Specialized type clock used to measure the time interval
- Count upward and downward from zero or specified time interval to generate a time delay
- It can be seen as a counter of oscillator clock or clock counter
- uses the frequency of the internal clock and generates a delay
- operated by incrementing a register value in every machine cycle

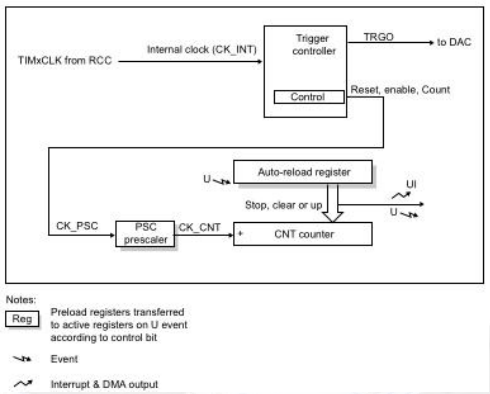
Timer ??

- Counts the clock feed into
- Prescaler divide the system clock (TIMx_PSC)
- Auto Reload Register (TIMx_ARR)
- Counter Register to keep the current count (TIMx_CNT)

Use of Time

- DAC Trigger driver
- Delay or Sleep and timer interrupt
- PWM generation
- So on . . .

$$CK\_CNT = \frac{f_{ck\_psc}}{PSC \geq 1}$$



Type of Timers

Timer type	Timer	Counter resolution	Counter type	Prescaler factor	DMA request generation	Capture/ compare channels	Complementary output	Max interface clock (MHz)	Max timer clock (MHz)[1]
Advanced-control	TIM1, TIM8	16-bit	Up, Down, Up/down	Any integer between 1 and 65536	Yes	4	Yes	90	180
General purpose	TIM2, TIM5	32-bit	Up, Down, Up/down	Any integer between 1 and 65536	Yes	4	No	45	90/180
	TIM3, TIM4	16-bit	Up, Down, Up/down	Any integer between 1 and 65536	Yes	4	No	45	90/180
	TIM9	16-bit	Up	Any integer between 1 and 65536	No	2	No	90	180
	TIM10, TIM11	16-bit	Up	Any integer between 1 and 65536	No	1	No	90	180
	TIM12	16-bit	Up	Any integer between 1 and 65536	No	2	No	45	90/180
	TIM13, TIM14	16-bit	Up	Any integer between 1 and 65536	No	1	No	45	90/180
Basic	TIM6, TIM7	16-bit	Up	Any integer between 1 and 65536	Yes	0	No	45	90/180

Basic Timer

Basic Timer (TIM6 & TIM7)

- Used to drive the DAC and Waveform generator
- Generic 16-bit time base
- Support Independent 16-bit DMA request generator
- Auto-Reload counter driven by a programmable prescaler (Clock Frequency between factor between 1 and 65536)
- interrupt and DMA generation for counter overflow (TIMx_DIER)→ UDE, UIE bits

Register Sets:

- Counter Register – TIMx_CNT
- Prescaler Register – TIMx_PSC
- Auto-Reload Register – TIMx_ARR use auto-reload preload enable bit (ARPE) in the TIMx_CR1 when UDIS is '0'
- TIMx_SR – Status register → UIF update interrupt flag for register updating
  - bit CEN in TIMx_CR1 enabled for clock to counter
- Control and other Registers
  - TIMx_CR1, TIMx_CR2, TIMx_DIER

Basic Timer Registers (TIM6 & TIM7)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ARPE	Res.	Res.	Res.	OPM	URS	UDIS	CEN
								rw				rw	rw	rw	rw

Figure 25: CEN: counter enable, UDIS: Auto reload preload, One-pulse mode (OPM), URS: Update Request Source (update interrupt and DMA request)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CNT[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Figure 26: Counter Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PSC[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Figure 27: Prescaler (PSC) Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ARR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Figure 28: Timer auto reload register

Timer Counting and Update

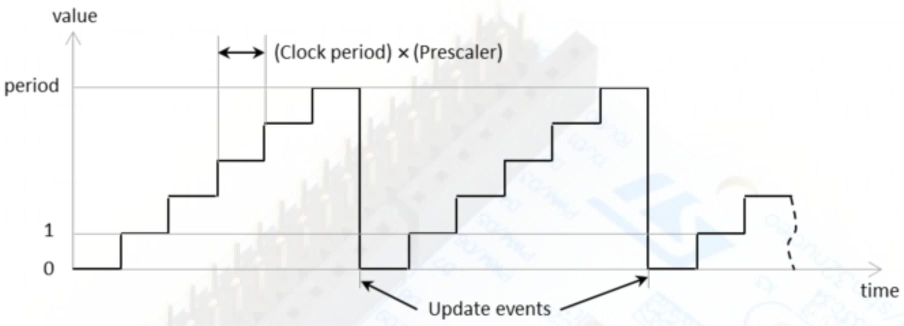


Figure 29: Timer Counting, update and Reload

General purpose timer

TIM2 to TIM5 are general purpose timer (See the Reference manual)

- 16-bit (TIM3 and TIM4) or 32-bit (TIM2 & TIM5) up, down, up/down counter
- 16-bit programmable prescaler divide (on the fly) the counter clock by the factor of 1 to 65536
- 4-independent channel (a) input capture (b) output capture (c) PWM generation (d) One-pulse mode output
- interrupt/DMA generation (a) counter overflow (b) trigger event (counter start, stop or initialization) (c) input capture/output compare
- trigger input for external clock cycle

TIM9 to TIM14 are general purpose timer (different from the above)

- 16-bit auto-reload upcounter
- 16-bit programmable prescaler divide (on the fly)
- 2-independent channel (a) input capture (b) output capture (c) PWM generation (d) One-pulse mode output

Advanced Timers (TIM1 & TIM8)

Extracted from the reference manual (follow Youtube videos)

- 16-bit up/down autoreload counter
- 16-bit programmable prescaler
- 4-independent channel (a) input capture (b) output capture (c) PWM generation (d) One-pulse mode output
- Complementary output
- synchronous circuit for the timer with external signals to interconnect several timers
- repetition counter to update the timer registers only after given number of cycles of the counter
- break input to put the timer's output signal in reset state or in known state
- interrupt/DMA generation (a) counter overflow (b) trigger event (counter start, stop or initialization) (c) input capture/output compare (d) break input
- support incremental encoder and Hall-sensor for positioning purpose
- trigger input for external clock or cycle by cycle current management

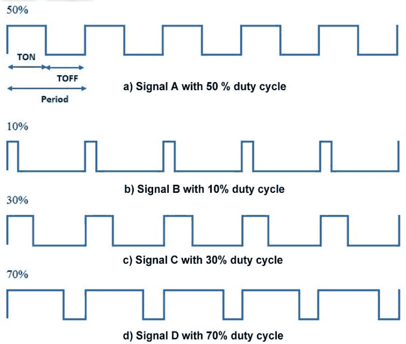
PWM – Pulse Width Modulation

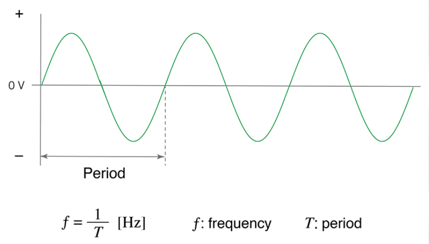
- To control the analog circuit
- One of the important function of timer
- Analog Signal – Values varies continuously
- Digital signal can only be either high or low

PWM consists Two main Component

- Duty cycle – describe the amount of time the signal is high
- Frequency – how fast the PWM completes a cycle

Duty cycle





We discuss PWM Detail in the next week

USART – Universal Synchronous/Asynchronous Receiver and Transmitter

USART – Two mode of communication

- Asynchronous Transmission and Reception
  - No clock is sent to slave from master
  - $Rx_m \leftarrow Tx_s$ and $Tx_m \rightarrow Rx_s$
  - No clock transmission
  - example: RS232
- Synchronous Transmission and Reception
  - $SDA_m \leftrightarrow SDA_s$ – data transmission line
  - $CLK_m \rightarrow CLK_s$ – clock transmission line
  - Example: I^2C, SPI
- STM32 USART Total 6-USART
  - (1,2,3,& 6)– Full USART
  - Can only be used for UART (UART4 & 5)
  - use NRZ standard digital signaling
  - USART1& 6 – Maximum Baud Rate: 11.25 Mbits/s
  - USART2,3, 4, 5 – Maximum Baud Rate: 5.62 Mbits/s

Asynchronous Communication

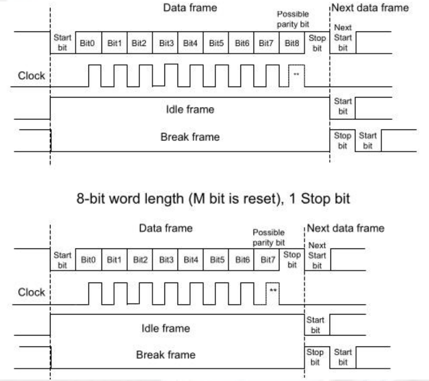
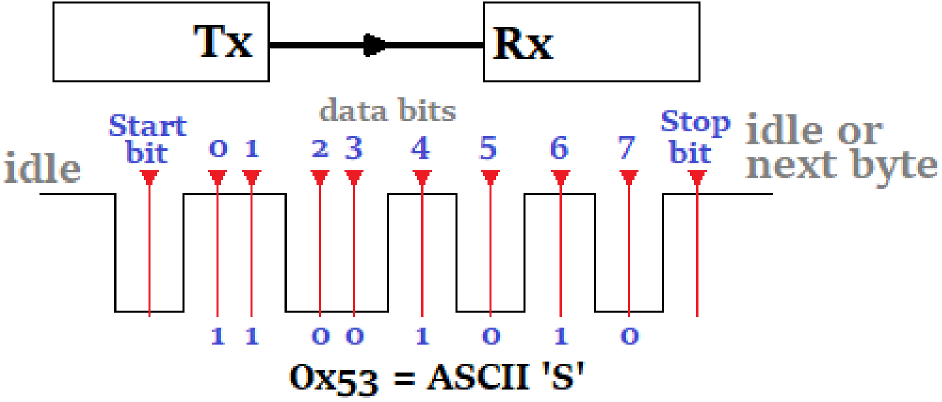


Figure 30: USART Signaling

UART Configuration

- No clock require to transmit from transmitter to receiver
- Data Transmission and Reception – Baud Rate
- Start, Stop bit and Data length
- Interrupt and DMA enabled or Disable
- Transmit and Receive data using Data Register
- Using shift register for transmission and reception (DR) of data
- Check Status for transmission and Reception of data



STM32 USART Configuration

Step To configure USART

1. Enable USART clock (the APBx Bus)
2. Enable USART by writing the UE bit in USART_CR1 register to 1
3. Program the *M* bit in USART_CR1 to define the word length
4. Program the number of stop bits in USART_CR2
5. Select DMA enable (DMAT) in USART_CR3 for multiple communication
6. Select the desired baud rate using the USART_BRR register
7. Set the TE and RE bit in USARTx_CR1 to send and receive USART frame
8. Write the data to send in the USARTx_DR register – clear TXE bit of SR register
9. After writing the last data into the USARTx_DR register, wait until TC=1

Enable USART clock (the APBx Bus)

Enable Clock for USARTx

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	DAC EN	PWR EN	CEC EN	CAN2 EN	CAN1 EN	FMPI2C1 EN	I2C3 EN	I2C2 EN	I2C1 EN	UART5 EN	UART4 EN	USART3 EN	USART2 EN	SPDIFRX EN
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
SPI3 EN	SPI2 EN	Res.	Res.	WWDG EN	Res.	Res.	TIM14 EN	TIM13 EN	TIM12 EN	TIM7 EN	TIM6 EN	TIM5 EN	TIM4 EN	TIM3 EN	TIM2 EN
rw	rw			rw			rw	rw	rw	rw	rw	rw	rw	rw	rw

Figure 31: Enable Clock for USART ($RCC \rightarrow APB1ENR$)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SAI2 EN	SAI1 EN	Res.	Res.	Res.	TIM11 EN	TIM10 EN	TIM9 EN
								rw	rw				rw	rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	SYSCFG EN	SPI4 EN	SPI1 EN	SDIO EN	ADC3 EN	ADC2 EN	ADC1 EN	Res.	Res.	USART6 EN	USART1 EN	Res.	Res.	TIM8 EN	TIM1 EN
	rw	rw	rw	rw	rw	rw	rw			rw	rw			rw	rw

Figure 32: $RCC \rightarrow APB2ENR$ for USART1& 6

Back to Configuration content

Enable USART by writing the UE bit in USARTx_CR1 register to 1

Enable USART, first reset as $USART\_CR1 = 0$ and then set UE-bit before any configuration

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OVER8	Res.	UE	M	WAKE	PCE	PS	PEIE	TXEIE	TCIE	RXNEIE	IDLEIE	TE	RE	RWU	SBK
rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Figure 33: USARTx Control Register1

- Set UE, M for word length: '0' for 8-bit and '1' for 9-bit
- PCE: Parity enable, PS: parity selection (odd or even)
- TXEIE: TXE interrupt enable, TCIE: Transmission complete interrupt enable and RXNEIE: RXNE (receive buffer not empty) interrupt enable
- TE: Transmitter enable, RE: Receiver enable
- SBK: Send break – a low equal to the USART frame length

Back to Configuration content

Program the number of stop bits in USARTx_CR2 & USARTx˙CR3

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	LINEN	STOP[1:0]		CLKEN	CPOL	CPHA	LBCL	Res.	LBDIE	LBDL	Res.	ADD[3:0]			
	rw	rw	rw	rw	rw	rw	rw		rw	rw		rw	rw	rw	rw

Figure 34: USARTx_CR2Bits 13:12 STOP: STOP bits. ‘00’ - one stop bit, ‘01’ - 0.5 stop bit, ‘10’ – 2 stop bit, ‘11’: 1.5 stop bit

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	ONEBIT	CTSIE	CTSE	RTSE	DMAT	DMAR	SCEN	NACK	HDSEL	IRLP	IREN	EIE
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Figure 35: USARTx˙CR3 for DMA Transmitter (DMAT) and receiver (DMAR) enable.

Back to Configuration content

Select the desired baud rate using the USARTx_BRR register

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DIV_Mantissa[11:0]												DIV_Fraction[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Figure 36: Baud rate register (USARTx_BRR)

DIV_Mantissa[11:0]: mantissa of USARTDIV
Bits 3:0 DIV_Fraction[3:0]: fraction of USARTDIV
Baud Rate calculation

$$Tx/Rx\ Baud = \frac{f_{ck}}{8 \times (2 - OVER8) \times USARTDIV} \tag{1}$$

To program USARTDIV example: 103.1678
$DIV\_fraction = 1678 \times 16 \equiv 0x3$
$DIV\_Mantissa = 103 = 0x67$

Back to Configuration content

Enable USARTx Transmission and Reception

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
OVER8	Res.	UE	M	WAKE	PCE	PS	PEIE	TXEIE	TCIE	RXNEIE	IDLEIE	TE	RE	RWU	SBK
rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Figure 37: USARTx Control Register1

- Enable RE and TE bit (set)
- See interrupt service routine: USARTx_IRQHandler. See table 38 in reference manual for interrupt.
- To enable Parity Interrupt set PEIE-bit
- To enable transmission interrupt enable set TXEIE – interrupt when TXE (transmission not empty) is true.
- To enable interrupt for data receive, set RXNEIE-bit. an interrupt when RXNE (receiver buffer not empty)
- Transmission complete Interrupt set TCIE-bit.
- see for other interrupts

Status Register

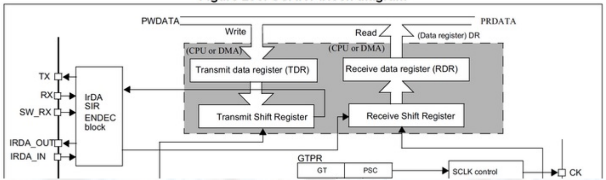
In the interrupt, service routing checks the status register for the cause of the interrupt.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	CTS	LBD	TXE	TC	RXNE	IDLE	ORE	NF	FE	PE
						rc_w0	rc_w0	r	rc_w0	rc_w0	r	r	r	r	r

Figure 38: Status Register (USARTx_SR)

- Clear the interrupt (RXNE, TXE)in SR register
- example find here `https://www.keil.com/download/docs/359.asp`
- TXE-bit is set when content of DR register transferred to Shift Register. Shift register will start sending to the TX line
- TC flag is set after send a stop-bit is sent (after writing the last data byte wait for TC)
- During Receive RXNE bit is set. A read from data register clear data register (DR) and RXNE-bit

Figure 296. USART block diagram



Back to Configuration content