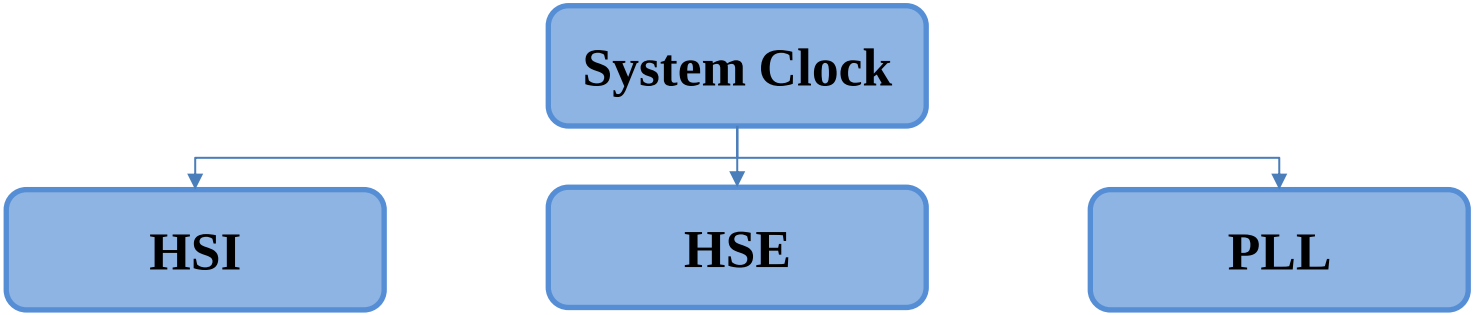
Iran University of
Science and Technology

Microcontroller

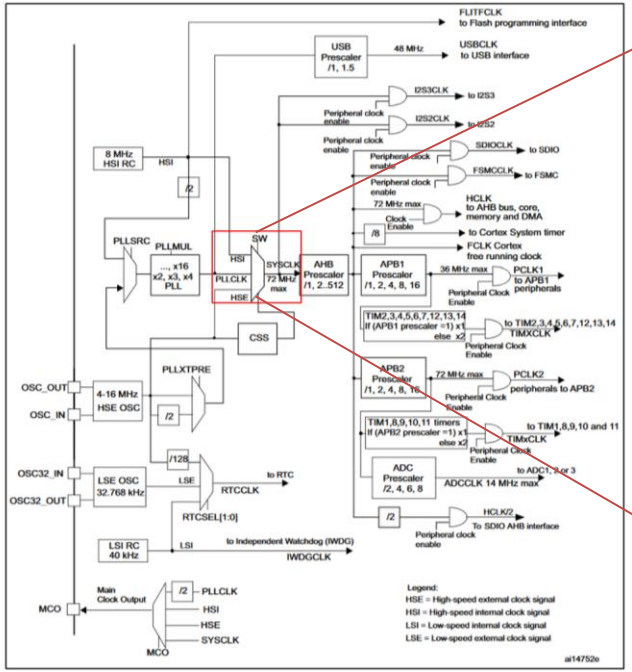
Department of Electrical Engineering
Iran University of Science and Technology

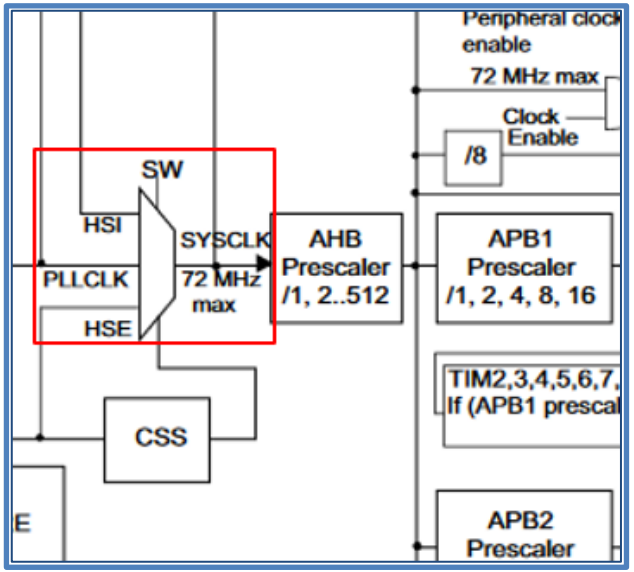
Clock configuration

Three different clock sources can be used to drive the system clock (SYSCLK).

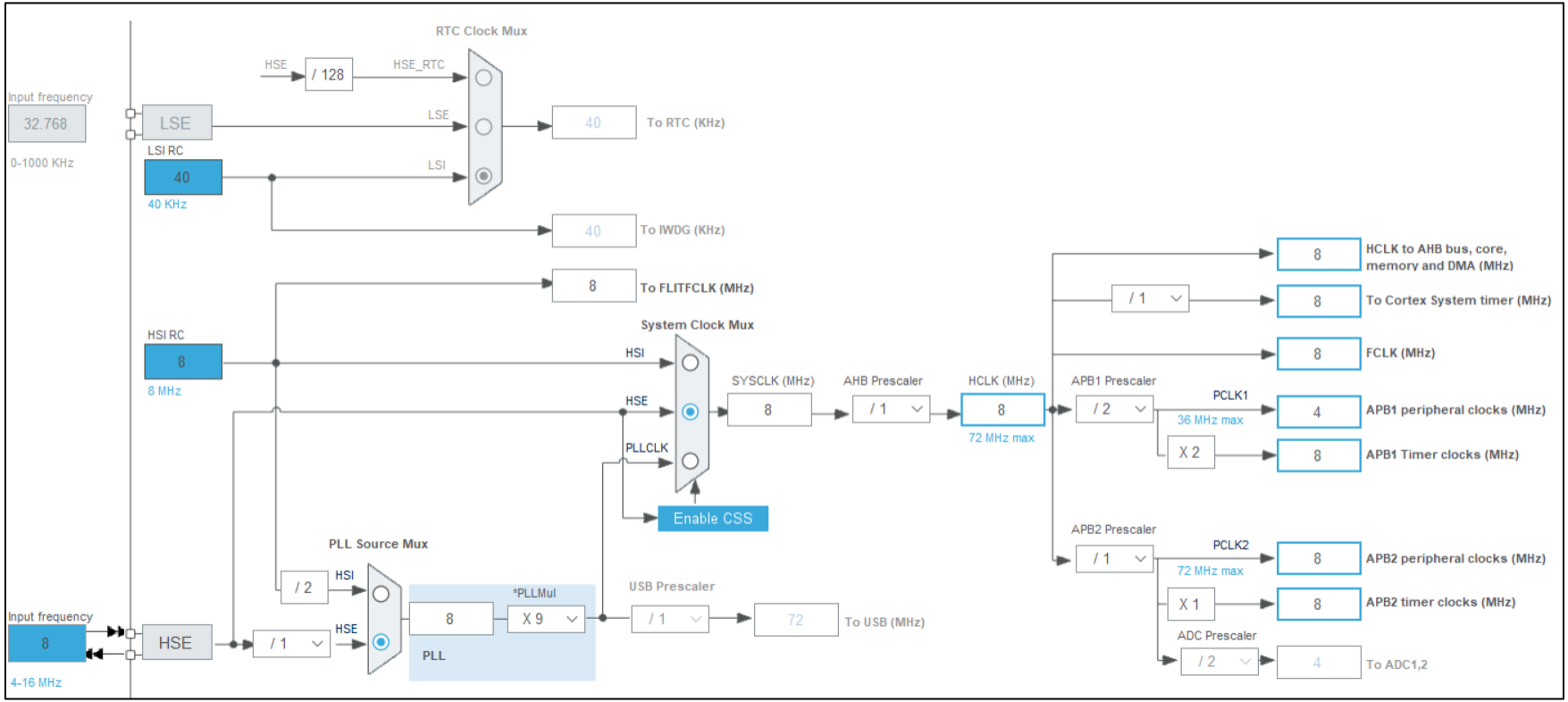


Clock tree


FLITFCLK
to Flash programming interface
USB Prescaler /1, 1.5
48 MHz
USBCLK
to USB interface
I2S3CLK
to I2S3
I2S2CLK
to I2S2
Peripheral clock enable
SDIOCLK
to SDIO
FSMCCLK
to FSMC
8 MHz HSI RC
HSI
/2
PLLSRC
PLLMUL
... x16
x2, x3, x4
PLL
SW
PLLCLK
HSE
SYSCLK
72 MHz max
AHB Prescaler /1, 2..512
APB1 Prescaler /1, 2, 4, 8, 16
HCLK
to AHB bus, core, memory and DMA
Clock Enable
/8
to Cortex System timer
FCLK Cortex free running clock
36 MHz max
PCLK1
to APB1 peripherals
Peripheral Clock Enable
TIM2,3,4,5,6,7,12,13,14
If (APB1 prescaler =1) x1
else x2
to TIM2,3,4,5,6,7,12,13,14
TIMXCLK
CSS
APB2 Prescaler /1, 2, 4, 8, 16
72 MHz max
PCLK2
peripherals to APB2
PLLXTPRE
OSC_OUT
OSC_IN
4-16 MHz
HSE OSC
/2
TIM1,8,9,10,11 timers
If (APB2 prescaler =1) x1
else x2
to TIM1,8,9,10 and 11
TIMxCLK
/128
OSC32_IN
OSC32_OUT
LSE OSC
32.768 kHz
LSE
to RTC
RTCCLK
RTCSEL[1:0]
ADC Prescaler /2, 4, 6, 8
to ADC1, 2 or 3
ADCCLK 14 MHz max
/2
HCLK/2
to SDIO AHB interface
LSI RC
40 kHz
LSI
to Independent Watchdog (IWDG)
IWDGCLK
Main Clock Output
MCO
/2
PLLCLK
HSI
HSE
SYSCLK
Legend:
HSE = High-speed external clock signal
HSI = High-speed internal clock signal
LSI = Low-speed internal clock signal
LSE = Low-speed external clock signal
ai14752e


Peripheral clock enable
72 MHz max
Clock Enable
/8
SW
HSI
PLLCLK
HSE
SYSCLK
72 MHz max
AHB Prescaler /1, 2..512
APB1 Prescaler /1, 2, 4, 8, 16
TIM2,3,4,5,6,7,
If (APB1 prescal
CSS
APB2 Prescaler

Clock tree


RTC Clock Mux
HSE
/ 128
HSE_RTC
LSE
LSE
40
To RTC (KHz)
LSI
LSI RC
40
40 KHz
40
To IWDG (KHz)
Input frequency
32.768
0-1000 KHz
8
To FLITFCLK (MHz)
System Clock Mux
HSI RC
8
8 MHz
HSI
HSE
PLLCLK
SYSCLK (MHz)
8
AHB Prescaler
/ 1
HCLK (MHz)
8
72 MHz max
Enable CSS
APB1 Prescaler
/ 2
PCLK1
36 MHz max
X 2
8
HCLK to AHB bus, core, memory and DMA (MHz)
/ 1
8
To Cortex System timer (MHz)
8
FCLK (MHz)
4
APB1 peripheral clocks (MHz)
8
APB1 Timer clocks (MHz)
APB2 Prescaler
/ 1
PCLK2
72 MHz max
8
APB2 peripheral clocks (MHz)
X 1
8
APB2 timer clocks (MHz)
ADC Prescaler
/ 2
4
To ADC1,2
PLL Source Mux
/ 2
HSI
HSE
/ 1
HSE
*PLLMul
8
X 9
PLL
USB Prescaler
/ 1
72
To USB (MHz)
Input frequency
8
4-16 MHz
HSE

Main buses

Three different clock sources can be used to drive the system clock (SYSCLK).

AHB

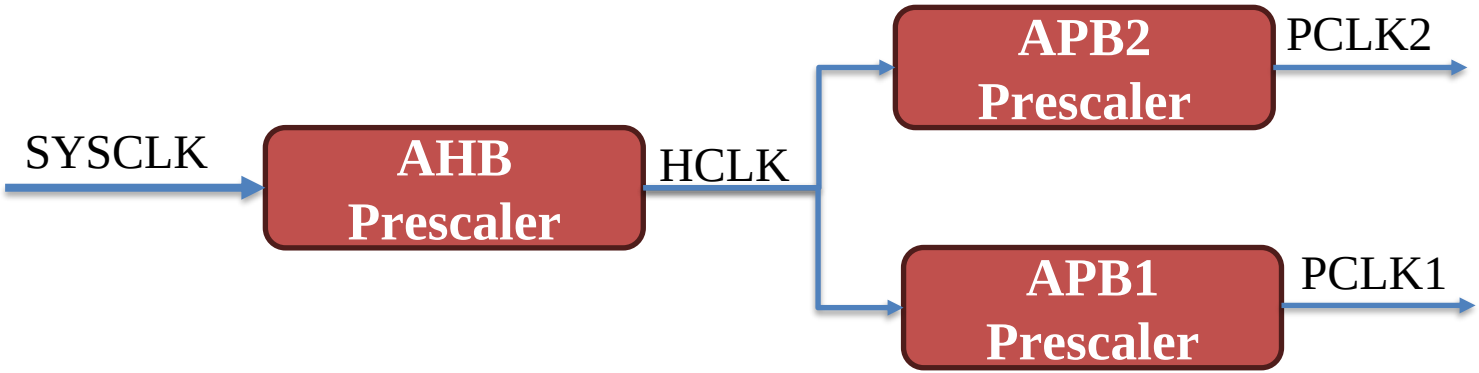
APB2

APB1

- Transfers data between the CPU and high-speed peripherals
- The maximum frequency of the AHB and the APB2 domains is 72 MHz
- The maximum allowed frequency of the APB1 domain is 36 MHz

Main buses

Prescaler : divide the clock frequency
PLL : synthesize higher frequencies from a lower-frequency clock source



Library

IUST
Iran University of Science and Technology

main.c

```
#include <stdio.h>
#include “Blink.h”

int main()
{
   ….



   ….
   return 0;
}
```

Blink.c

```
#include “Blink.h”

void Blink_func(void)
{

}
```

Blink.h

```
#ifndef    __Blink_H
#define    __Blink_H

#include <string.h>
#define  LED  GPIO_PIN_13

void Blink_func(void);

#endif
```

Library

ClkConfig.h

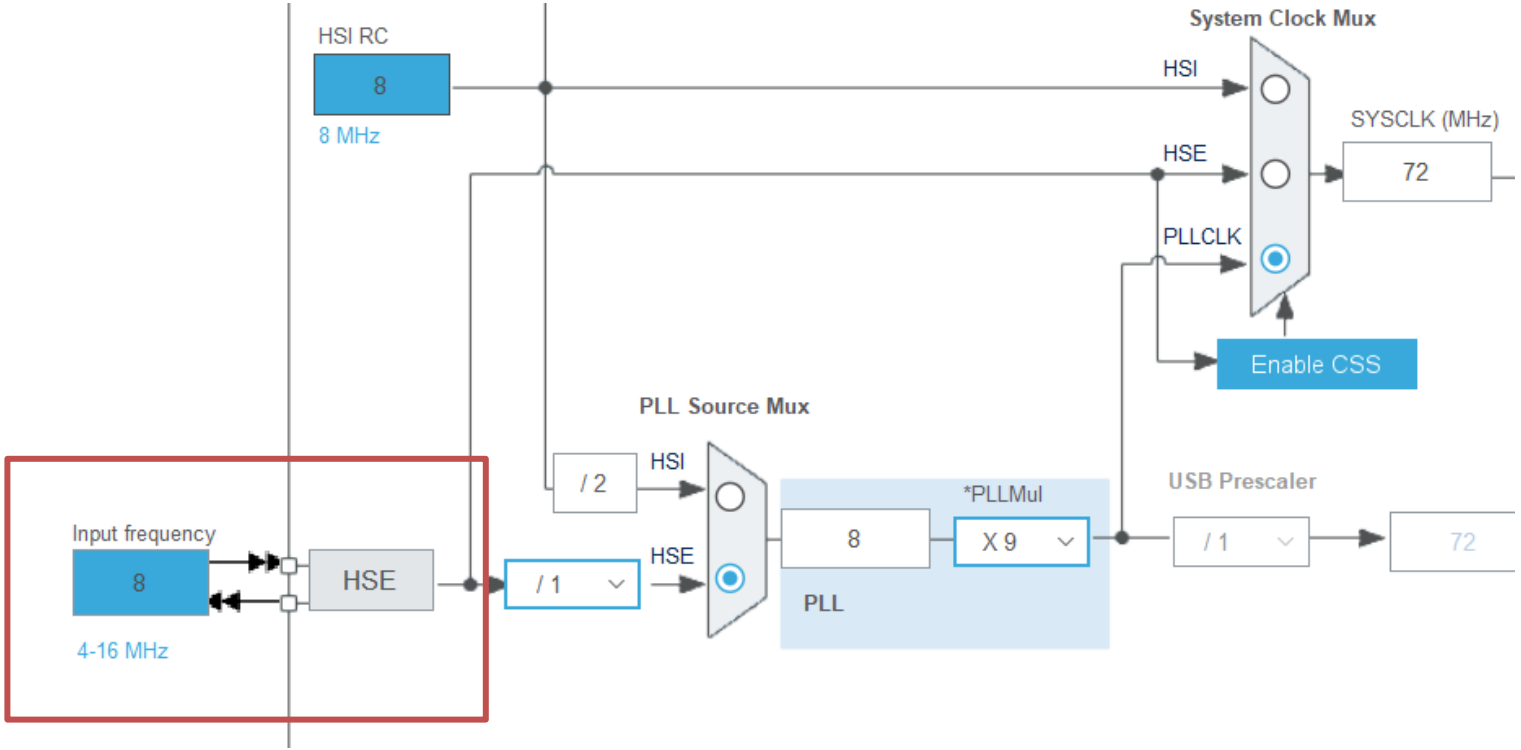
```
#ifndef __ClkConfig _H
#define __ClkConfig _H
#include "stm32f1xx.h"

void system_config(void);
void clock_config (void);



#endif
```

Library

ClkConfig.c

```
#include “ClkConfig.h”

void system_config(void){
//our code comes here!

}

void clock_config(void){
//our code comes here!

}
```

HSE ENABLE


HSI RC
8
8 MHz
System Clock Mux
HSI
HSE
PLLCLK
SYSCLK (MHz)
72
Enable CSS
PLL Source Mux
/ 2
HSI
HSE
/ 1
Input frequency
8
4-16 MHz
HSE
8
*PLLMul
X 9
PLL
USB Prescaler
/ 1
72

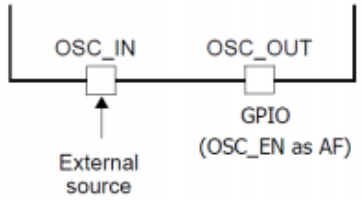
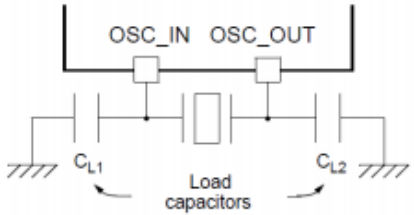
RCC_CR :HSEBYP

External high-speed clock bypass
0: external 4-16 MHz oscillator not bypassed
1: external 4-16 MHz oscillator bypassed with external clock

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved						PLL RDY	PLLON	Reserved				CSS ON	HSE BYP	HSE RDY	HSE ON
						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	HSION
r	r	r	r	r	r	r	r	rw	rw	rw	rw	rw		r	rw

RCC_CR :HSEBYP

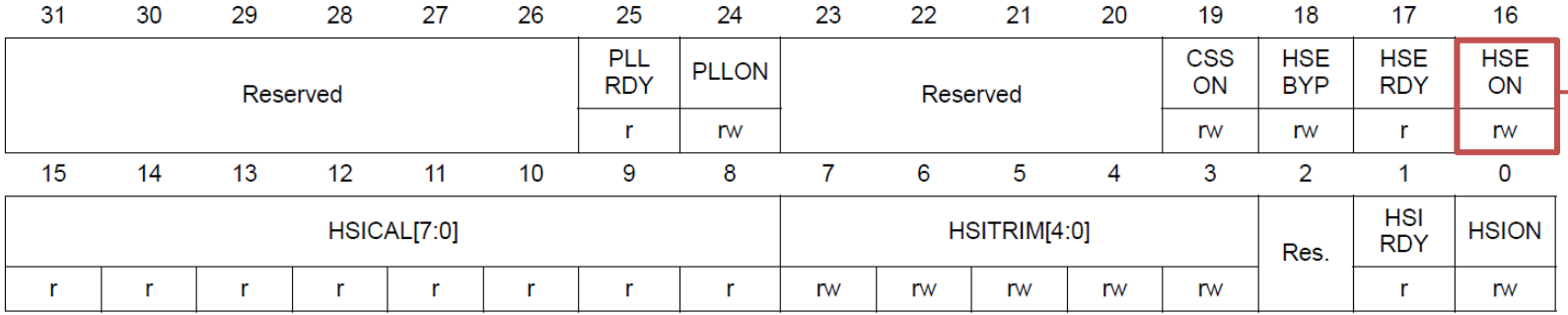
Clock source	Hardware configuration
External clock	
Crystal/Ceramic resonators	

RCC_CR :HSEBYP

```
#include “ClkConfig.h”

void clock_config(void){

RCC->CR &= ~RCC_CR_HSEBYP;

}
```

RCC_CR :HSEON

HSE clock enable
0: HSE oscillator OFF
1: HSE oscillator ON



31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved						PLL RDY	PLLON	Reserved				CSS ON	HSE BYP	HSE RDY	HSE ON
						r	rw					rw	rw	r	rw

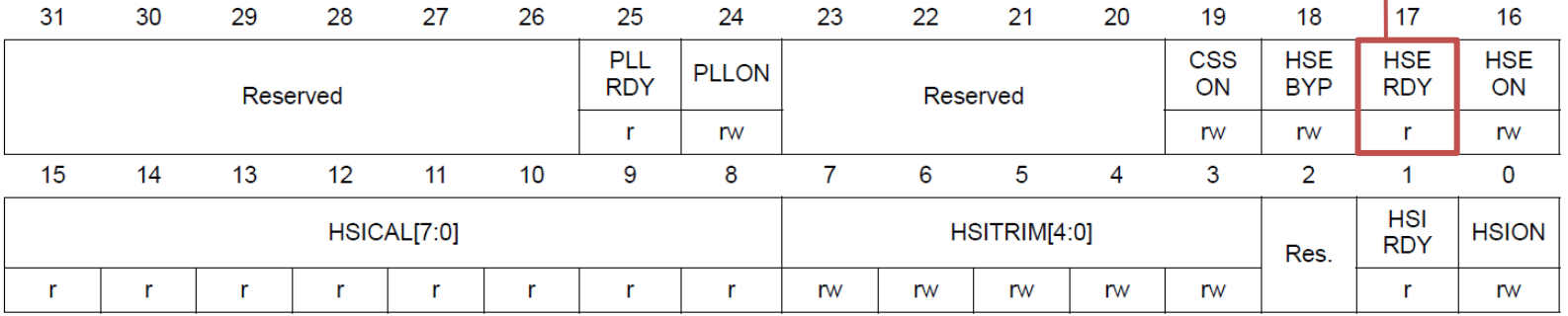
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
HSICAL[7:0]								HSITRIM[4:0]					Res.	HSI RDY	HSION
r	r	r	r	r	r	r	r	rw	rw	rw	rw	rw		r	rw

RCC_CR :HSEON

```
#include “ClkConfig.h”

Void clock_config(void){
RCC->CR &= ~RCC_CR_HSEBYP;
RCC->CR |= RCC_CR_HSEON;

}
```

RCC_CR :HSERDY

External high-speed clock ready flag
0: HSE oscillator not ready
1: HSE oscillator ready



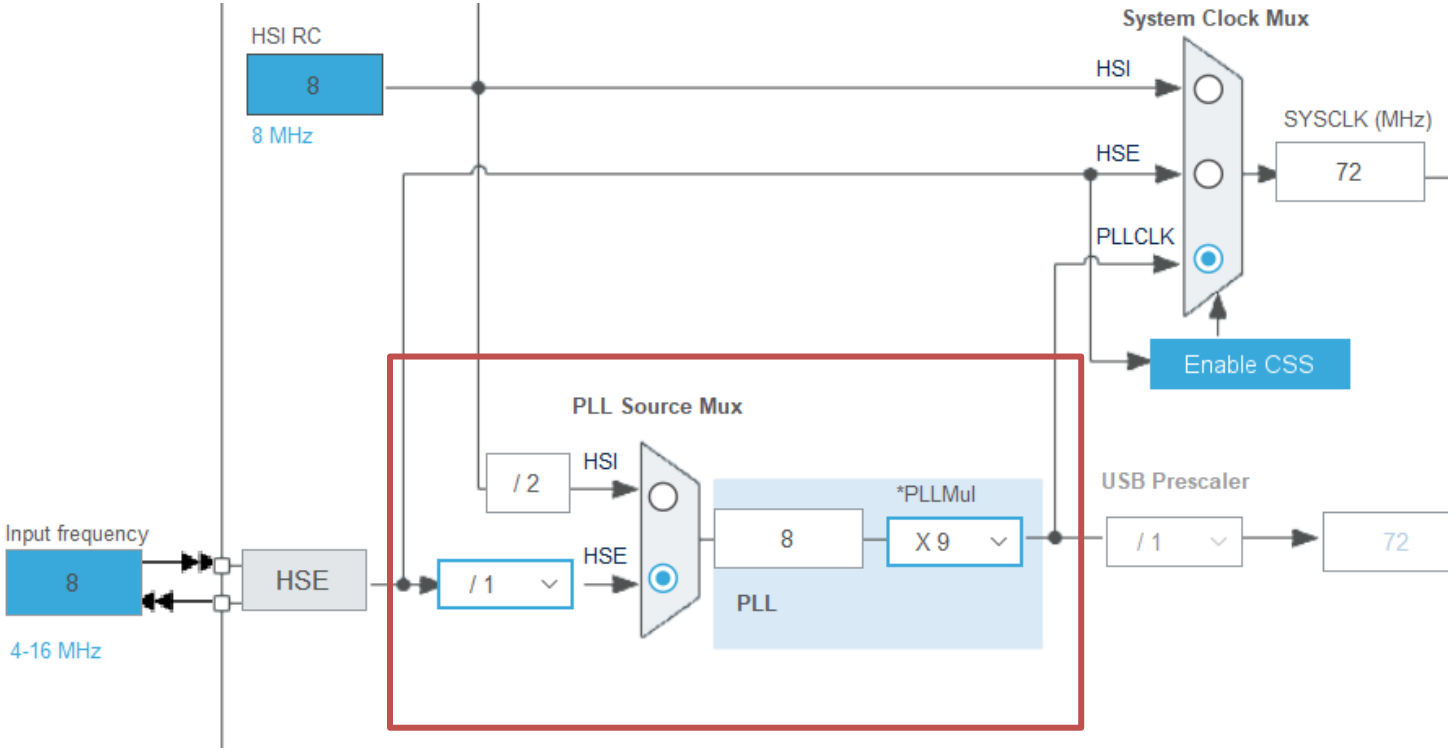
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved						PLL RDY	PLLON	Reserved				CSS ON	HSE BYP	HSE RDY	HSE ON
						r	rw					rw	rw	r	rw

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	HSION
r	r	r	r	r	r	r	r	rw	rw	rw	rw	rw		r	rw

RCC_CR :HSERDY

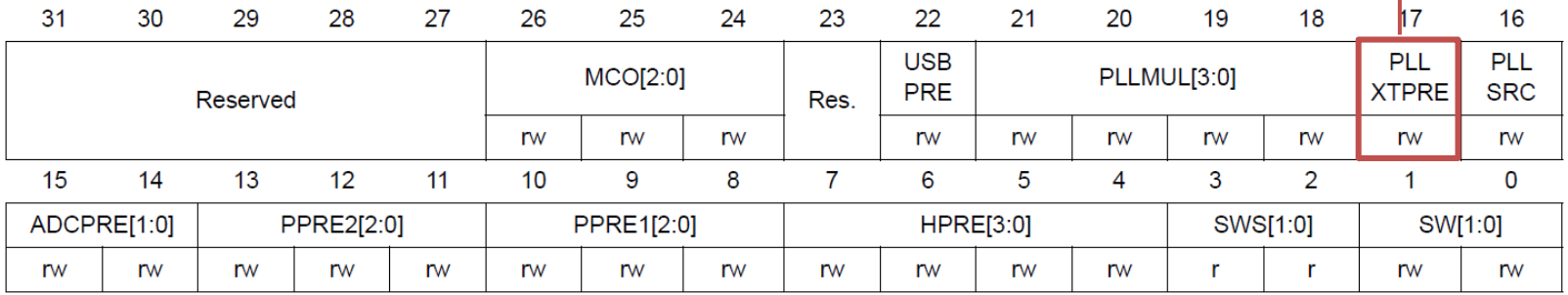
```
#include “ClkConfig.h”

Void clock_config(void){
RCC->CR &= ~RCC_CR_HSEBYP;
RCC->CR |= RCC_CR_HSEON;

 while ((RCC->CR & RCC_CR_HSERDY) == 0x00) {
}

}
```

PLL CONFIGURATIONS



RCC_CFGR :PLLXTPRE:

HSE divider for PLL entry
0: HSE clock not divided
1: HSE clock divided by 2



31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved					MCO[2:0]			Res.	USB PRE	PLLMUL[3:0]				PLL XTPRE	PLL SRC
					rw	rw	rw		rw	rw	rw	rw	rw	rw	rw

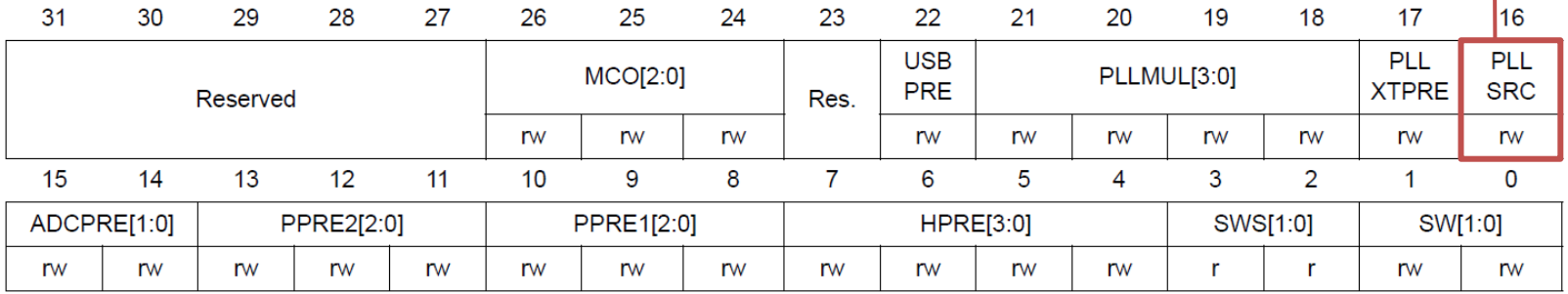
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADCPRE[1:0]		PPRE2[2:0]			PPRE1[2:0]			HPRE[3:0]				SWS[1:0]		SW[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	r	r	rw	rw

RCC_CFGR :PLLXTPRE:

```
#include “ClkConfig.h”

Void clock_config(void){
…
 while ((RCC->CR & RCC_CR_HSERDY) == 0x00) {}
 if(RCC->CR & RCC_CR_HSERDY) != 0x0){
    RCC->CFGR &= ~RCC_CFGR_PLLXTPRE;
}

}
```

RCC_CFGR :PLLSRC

PLL entry clock source
0: HSI oscillator clock / 2 selected as PLL input clock
1: HSE oscillator clock selected as PLL input clock



31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved					MCO[2:0]			Res.	USB PRE	PLLMUL[3:0]				PLL XTPRE	PLL SRC
					rw	rw	rw		rw	rw	rw	rw	rw	rw	rw

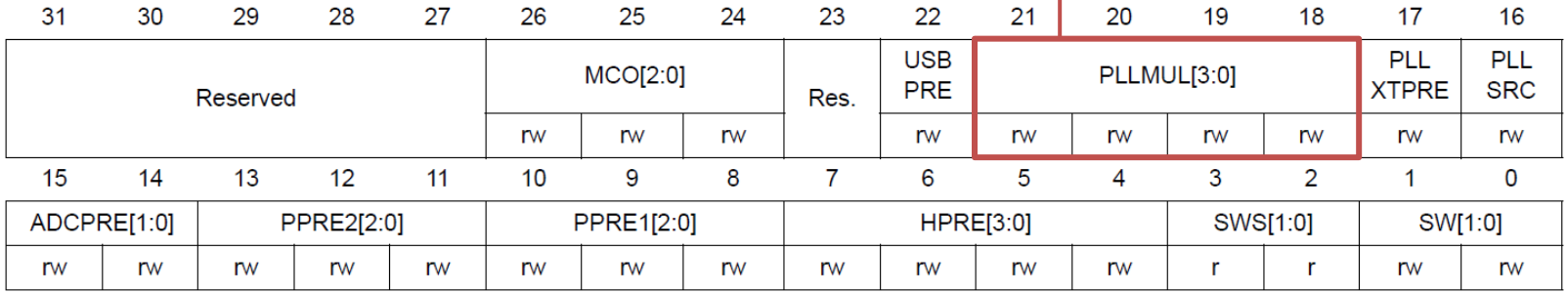
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADCPRE[1:0]		PPRE2[2:0]			PPRE1[2:0]			HPRE[3:0]				SWS[1:0]		SW[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	r	r	rw	rw

RCC_CFGR :PLLSRC

```
#include “ClkConfig.h”

Void clock_config(void){
…
 while ((RCC->CR & RCC_CR_HSERDY) == 0x00) {}
 if(RCC->CR & RCC_CR_HSERDY) != 0x0){
    RCC->CFGR &= ~RCC_CFGR_PLLXTPRE;
    RCC->CFGR |= RCC_CFGR_PLLSRC;
}

}
```

RCC_CFGR :PLLMULL

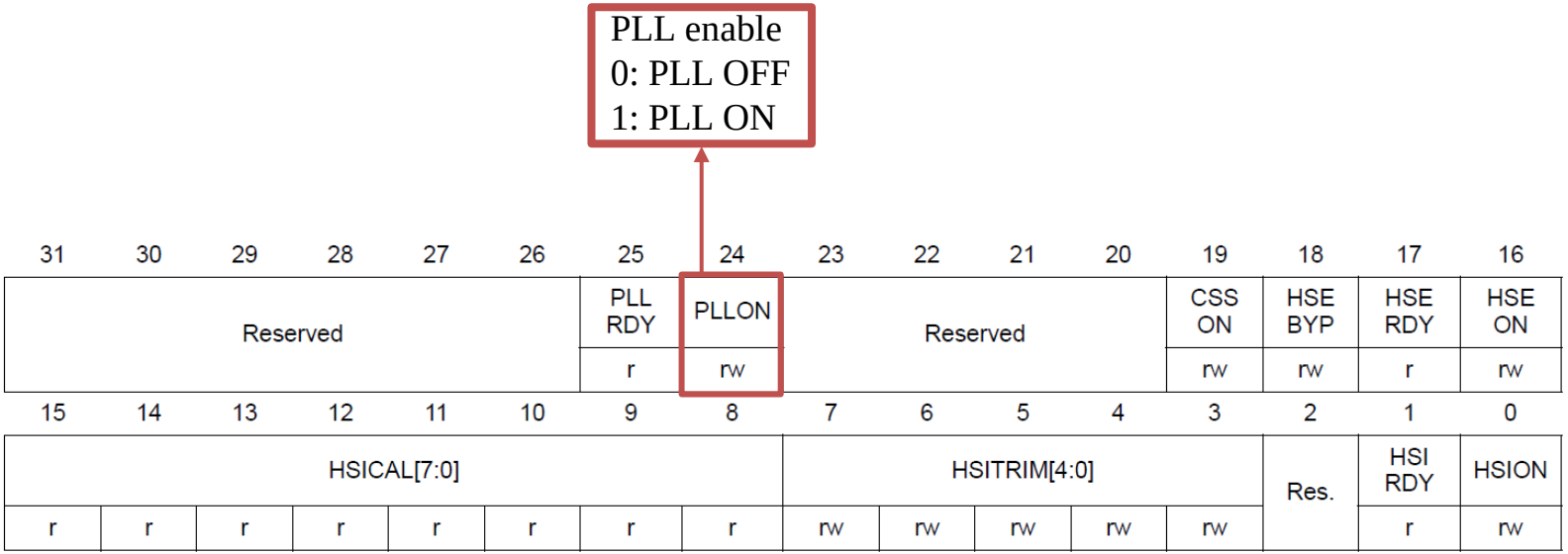
PLL multiplication factor
0111: PLL input clock x 9



31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved					MCO[2:0]			Res.	USB PRE	PLLMUL[3:0]				PLL XTPRE	PLL SRC
					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
ADCPRE[1:0]		PPRE2[2:0]			PPRE1[2:0]			HPRE[3:0]				SWS[1:0]		SW[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	r	r	rw	rw

RCC_CR :PLLON


PLL enable
0: PLL OFF
1: PLL ON
31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16
Reserved
PLL RDY
PLLON
Reserved
CSS ON
HSE BYP
HSE RDY
HSE ON
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
HSION
r r r r r r r r rw rw rw rw rw
r
rw

RCC_CR :PLLRDY

PLL clock ready flag
0: PLL unlocked
1: PLL locked

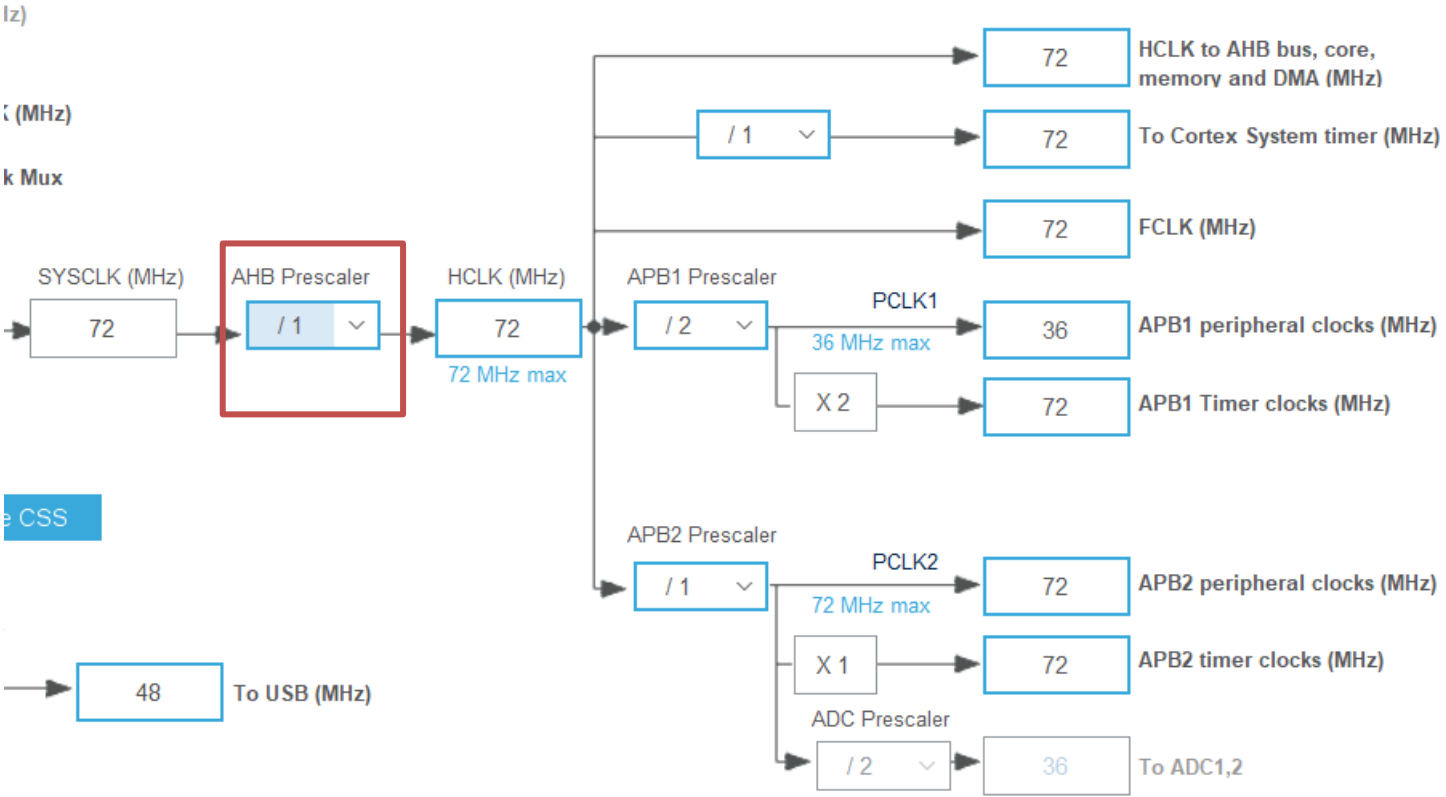
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved						PLL RDY	PLLON	Reserved				CSS ON	HSE BYP	HSE RDY	HSE ON
						r	rw					rw	rw	r	rw

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	HSION
r	r	r	r	r	r	r	r	rw	rw	rw	rw	rw		r	rw

RCC_CR :PLLRDY

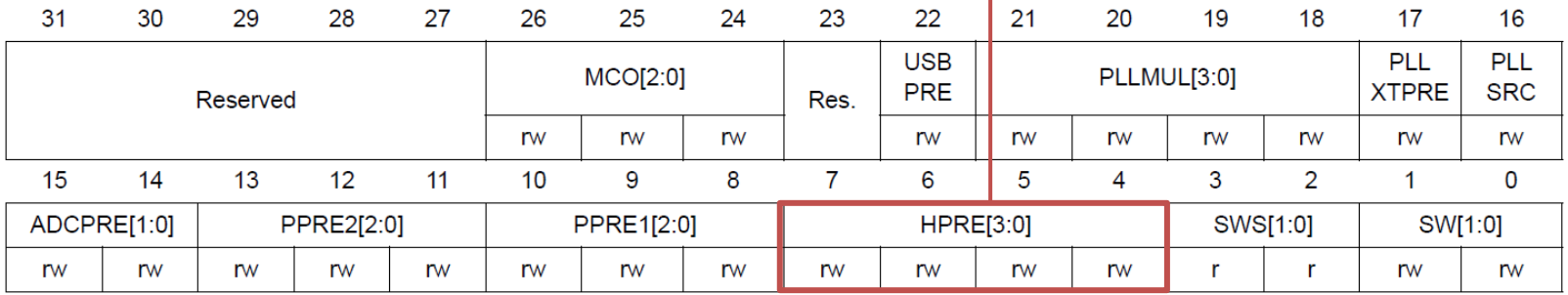
```
#include “ClkConfig.h”

Void clock_config(void){
…
if(RCC->CR & RCC_CR_HSERDY) != 0x0){
    …
 while( (RCC->CR & RCC_CR_PLLRDY)==0x0 );
}

}
```

RCC_CFGR :HPRE



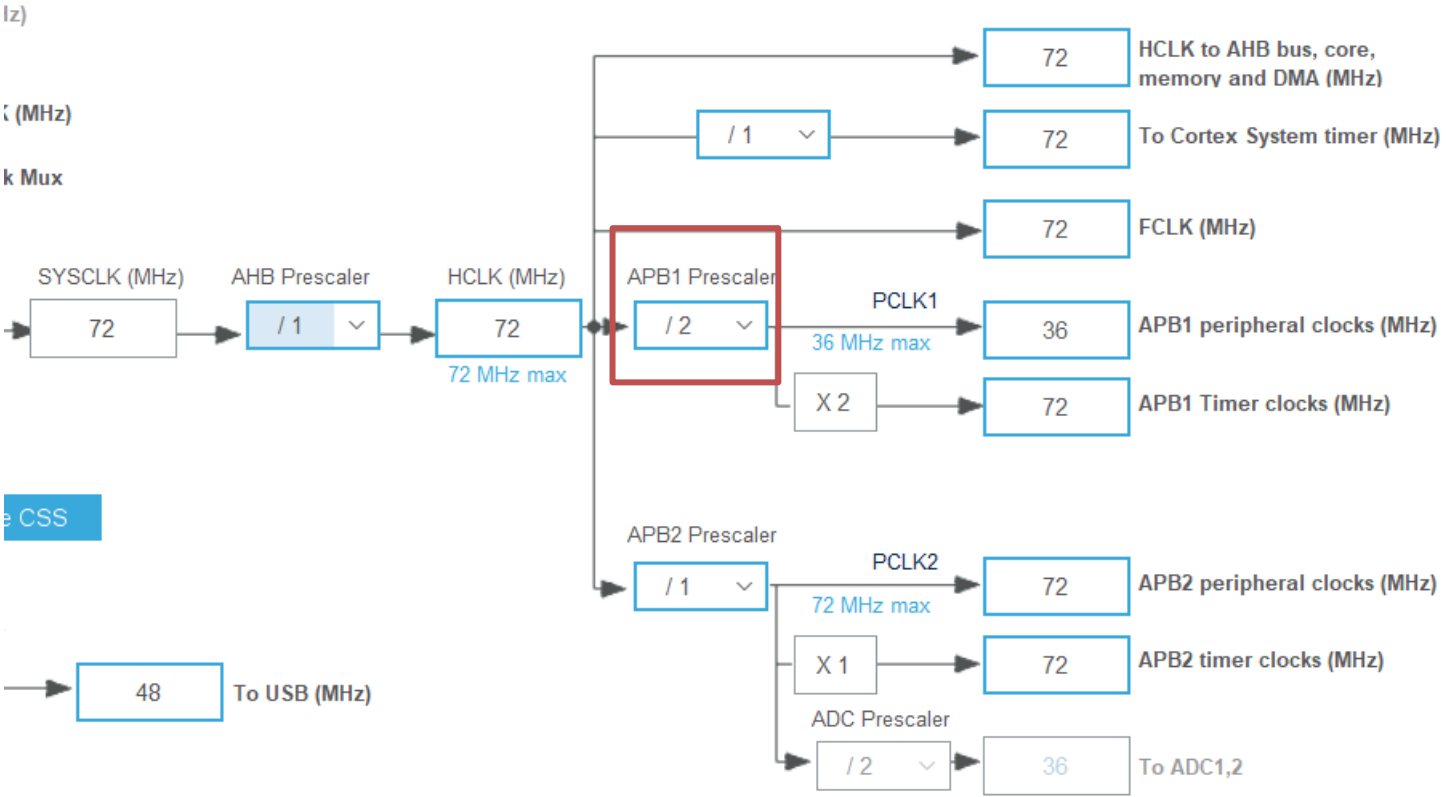
RCC_CFGR :HPRE

AHB prescaler0xx: HCLK not divided
0xxx: SYSCLK not divided
1000: SYSCLK divided by 2
…



Warning: the software has to set correctly these bits to not exceed 36 MHz on this domain

RCC_CFGR :PPRE1



RCC_CFGR :PPRE1

APB low-speed prescaler (APB1)
0xx: HCLK not divided
100: HCLK divided by 2
101: HCLK divided by 4
110: HCLK divided by 8
111: HCLK divided by 16

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved					MCO[2:0]			Res.	USB PRE	PLLMUL[3:0]				PLL XTPRE	PLL SRC
					rw	rw	rw		rw	rw	rw	rw	rw	rw	rw

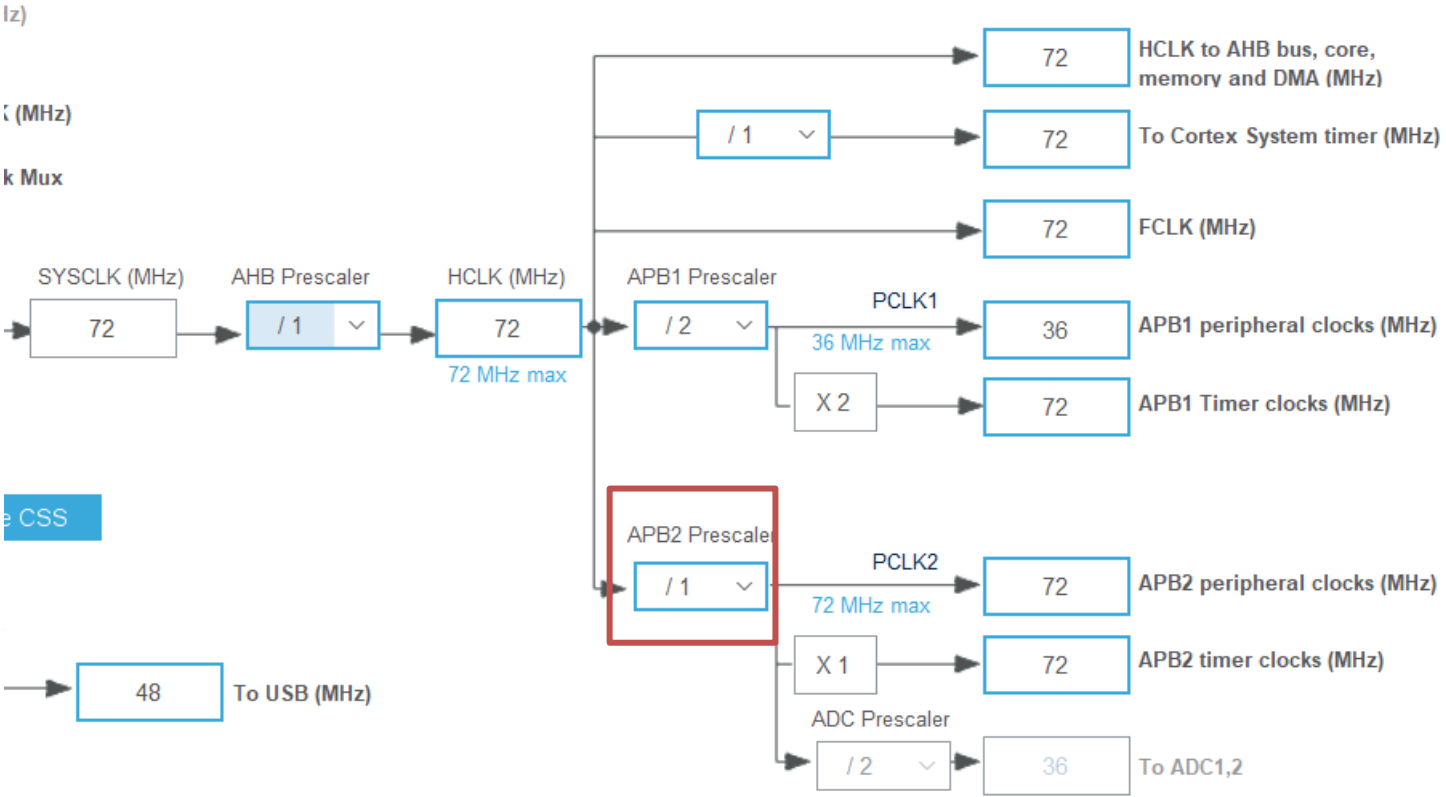
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADCPRE[1:0]		PPRE2[2:0]			PPRE1[2:0]			HPRE[3:0]				SWS[1:0]		SW[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	r	r	rw	rw

Warning: the software has to set correctly these bits to not exceed 36 MHz on this domain

RCC_CFGR :PPRE1

```
#include “ClkConfig.h”

Void clock_config(void){
…
RCC->CFGR &= ~RCC_CFGR_PPRE1;
 RCC->CFGR |=    RCC_CFGR_PPRE1_2;
// or RCC->CFGR |=    RCC_CFGR_PPRE1_DIV2;

}

}
```

RCC_CFGR :PPRE2



RCC_CFGR :PPRE2

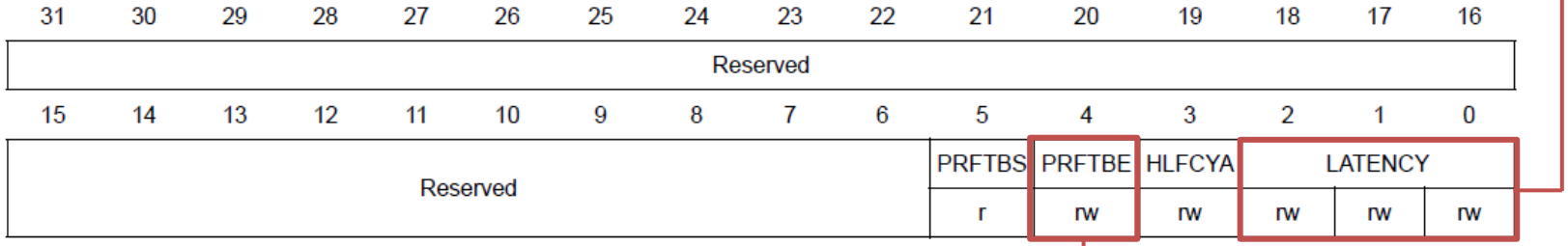
APB high-speed prescaler (APB2)
0xx: HCLK not divided
100: HCLK divided by 2
101: HCLK divided by 4
110: HCLK divided by 8
111: HCLK divided by 16

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved					MCO[2:0]			Res.	USB PRE	PLLMUL[3:0]				PLL XTPRE	PLL SRC
					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
ADCPRE[1:0]		PPRE2[2:0]			PPRE1[2:0]			HPRE[3:0]				SWS[1:0]		SW[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	r	r	rw	rw

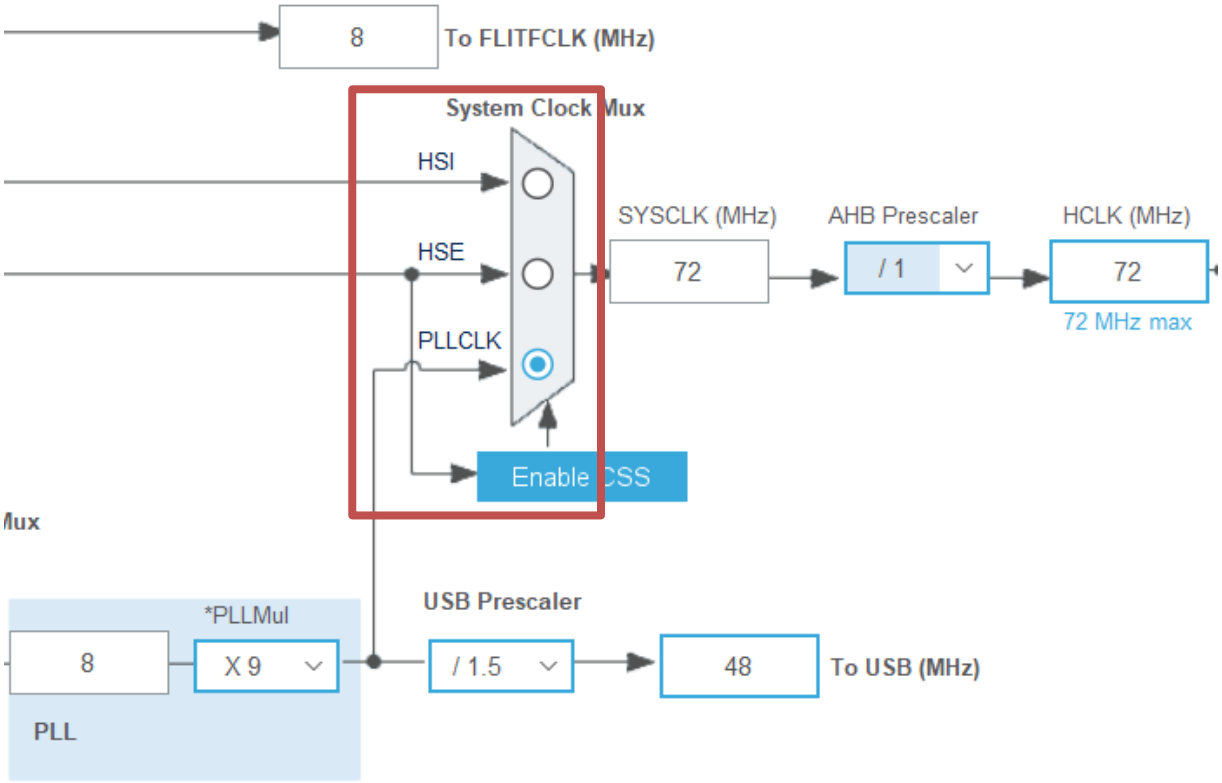
FLASH_ACR :LATENCY, PRFTBE

Latency
000 Zero wait state, if 0 < SYSCLK< 24 MHz
001 One wait state, if 24 MHz < SYSCLK < 48 MHz
010 Two wait states, if 48 MHz < SYSCLK < 72 MHz



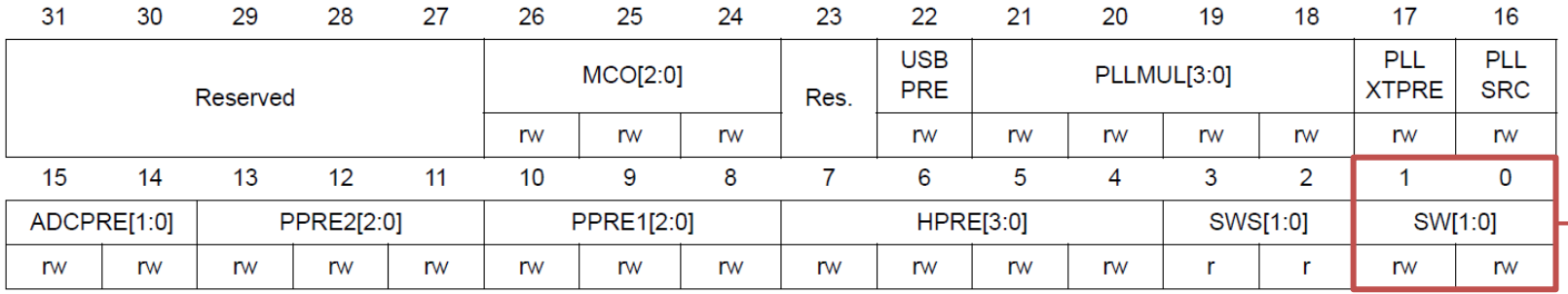
Prefetch buffer enable
0: Prefetch is disabled
1: Prefetch is enabled

RCC_CFGR :SW


8
To FLITFCLK (MHz)
System Clock Mux
HSI
HSE
PLLCLK
SYSCLK (MHz)
72
AHB Prescaler
/ 1
HCLK (MHz)
72
72 MHz max
Enable CSS
lux
*PLLMul
8
X 9
PLL
USB Prescaler
/ 1.5
48
To USB (MHz)

RCC_CFGR :SW

System clock switch
00: HSI selected as system clock
01: HSE selected as system clock
10: PLL selected as system clock
11: not allowed

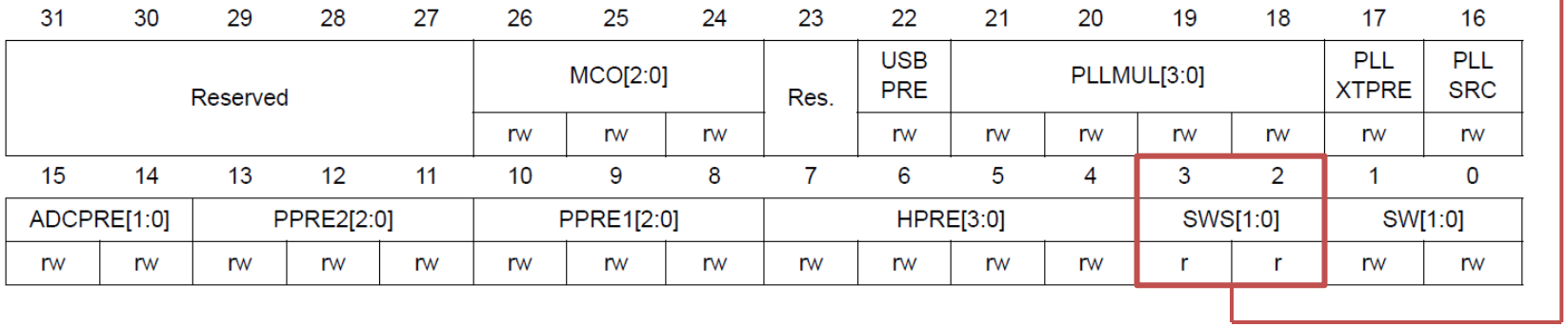


RCC_CFGR :SW

```
#include “ClkConfig.h”

Void clock_config(void){
…
if(RCC->CR & RCC_CR_HSERDY) != 0x0){
    …
 RCC->CFGR &= ~RCC_CFGR_SW;
 RCC->CFGR  |=  RCC_CFGR_SW_PLL;
 }

}
```

RCC_CFGR :SWS

System clock switch status
00: HSI oscillator used as system clock
01: HSE oscillator used as system clock
10: PLL used as system clock
11: not applicable

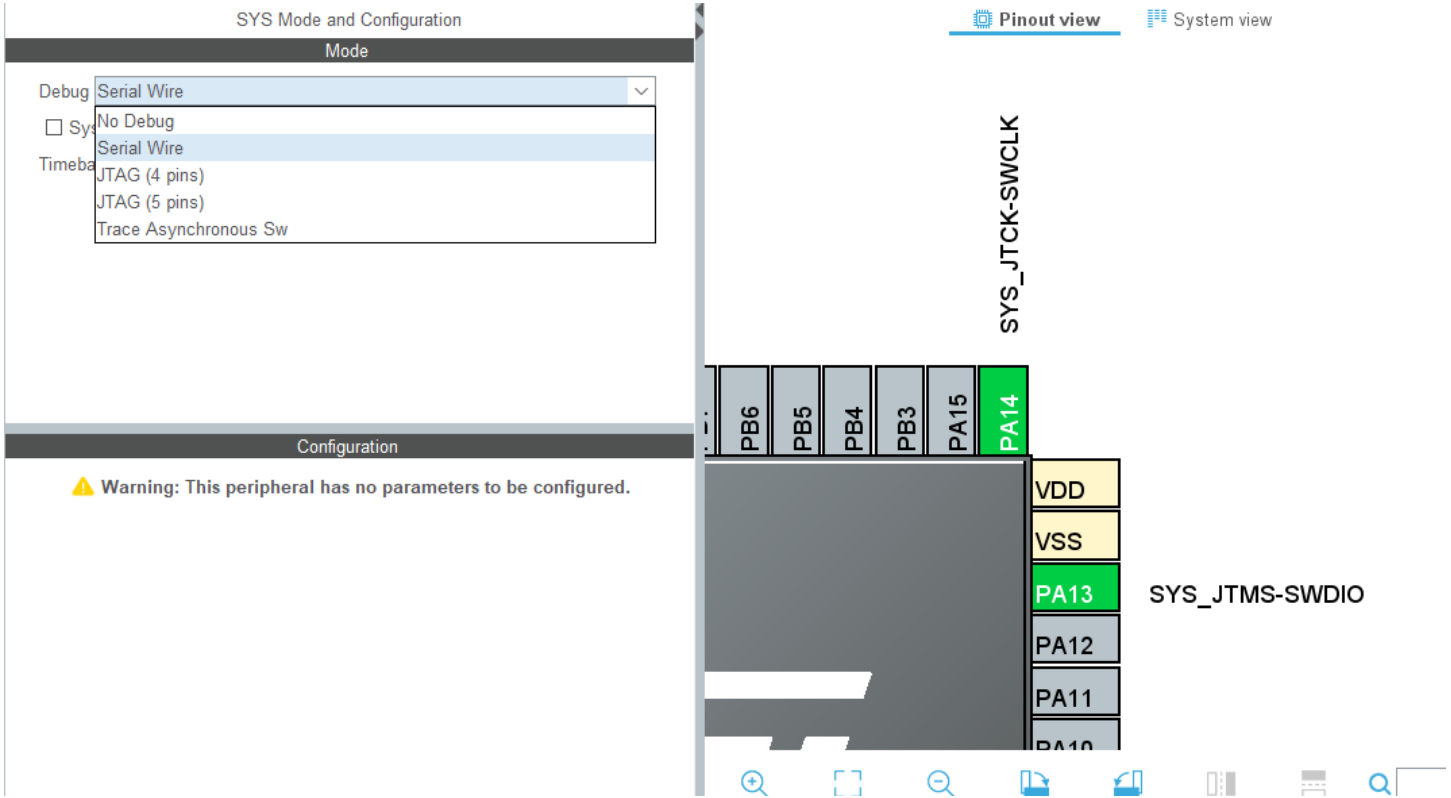


RCC_CFGR :SWS

```
#include “ClkConfig.h”

Void clock_config(void){
…
 RCC->CFGR &= ~RCC_CFGR_SW;
 RCC->CFGR  |=  RCC_CFGR_SW_PLL;
 while( (RCC->CFGR & RCC_CFGR_SWS) != 0x08);
 }

}
```

SYSTEM CONFIGS

DEBUG :SERIAL WIRE



RCC_APB2ENR :AFIOEN

Alternate function IO clock enable
0: Alternate Function IO clock disabled
1: Alternate Function IO clock enabled

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved										TIM11 EN	TIM10 EN	TIM9 EN	Reserved		
										rw	rw	rw			

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADC3 EN	USART 1EN	TIM8 EN	SPI1 EN	TIM1 EN	ADC2 EN	ADC1 EN	IOPG EN	IOPF EN	IOPE EN	IOPD EN	IOPC EN	IOPB EN	IOPA EN	Res.	AFIO EN
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		rw

RCC_APB2ENR :AFIOEN

```
#include “ClkConfig.h”

Void system_config(void){
 RCC->APB2ENR |= RCC_APB2ENR_AFIOEN;

}
```

AFIO_MAPR :SWJ_CFG[2:0]

Alternate function IO clock enable
000: Full SWJ (JTAG-DP + SW-DP): Reset State
001: Full SWJ (JTAG-DP + SW-DP) but without NJTRST
010: JTAG-DP Disabled and SW-DP Enabled
100: JTAG-DP Disabled and SW-DP Disabled

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved					SWJ_CFG[2:0]			Reserved			ADC2_ETRGREG_REMAP	ADC2_ETRGINJ_REMAP	ADC1_ETRGREG_REMAP	ADC1_ETRGINJ_REMAP	TIM5CH4_IREMAP
					w	w	w				rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PD01_REMAP	CAN_REMAP[1:0]		TIM4_REMAP	TIM3_REMAP[1:0]		TIM2_REMAP[1:0]		TIM1_REMAP[1:0]		USART3_REMAP[1:0]		USART2_REMAP	USART1_REMAP	I2C1_REMAP	SPI1_REMAP
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

AFIO_MAPR :SWJ_CFG[2:0]

```
#include “ClkConfig.h”

Void system_config(void){
 RCC->APB2ENR |= RCC_APB2ENR_AFIOEN;
 AFIO->MAPR &= ~AFIO_MAPR_SWJ_CNF;
 AFIO->MAPR |= AFIO_MAPR_SWJ_CNF_1;



}
```