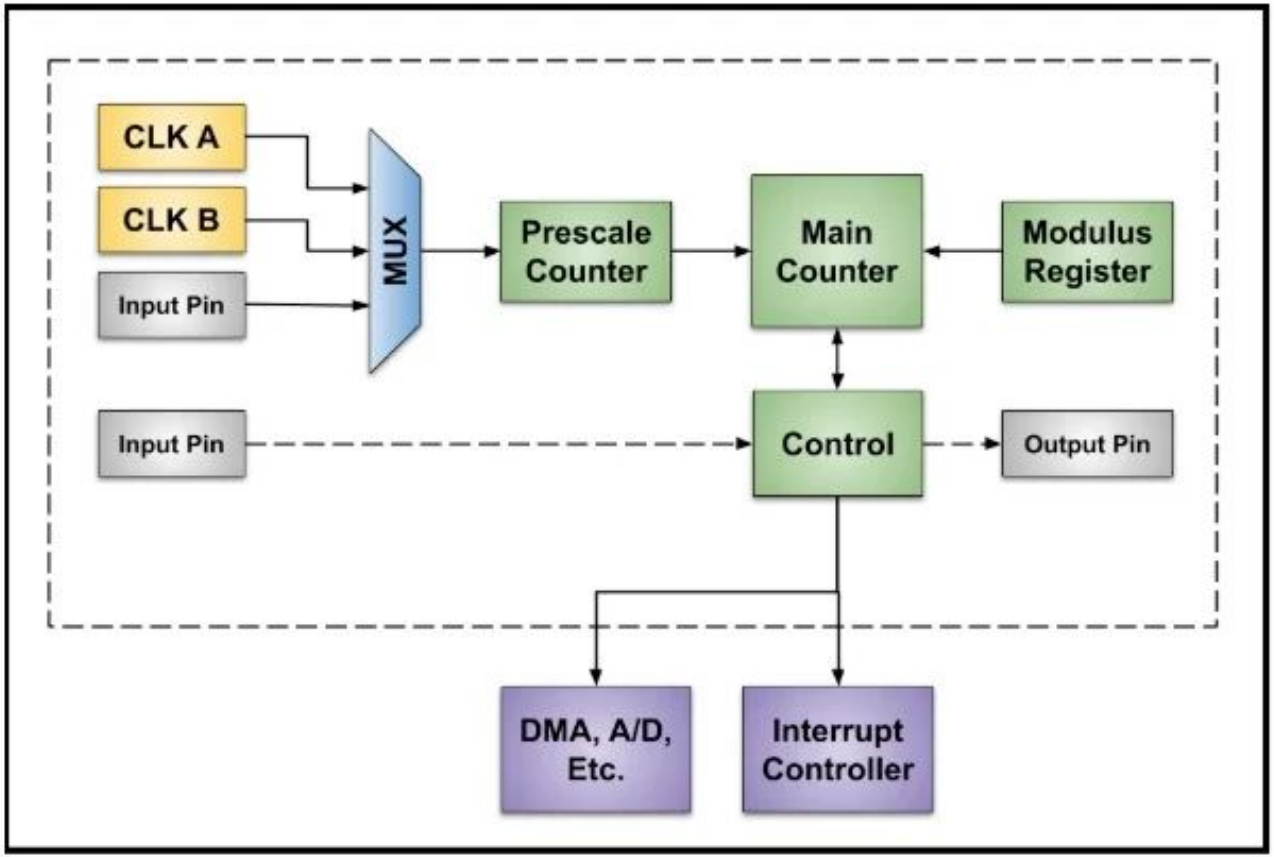
Microcontroller

~Tapadyuti Baral

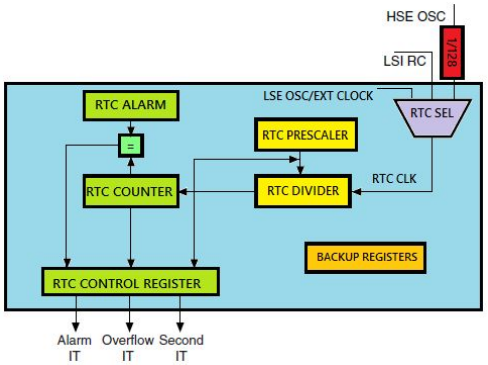
Topics :

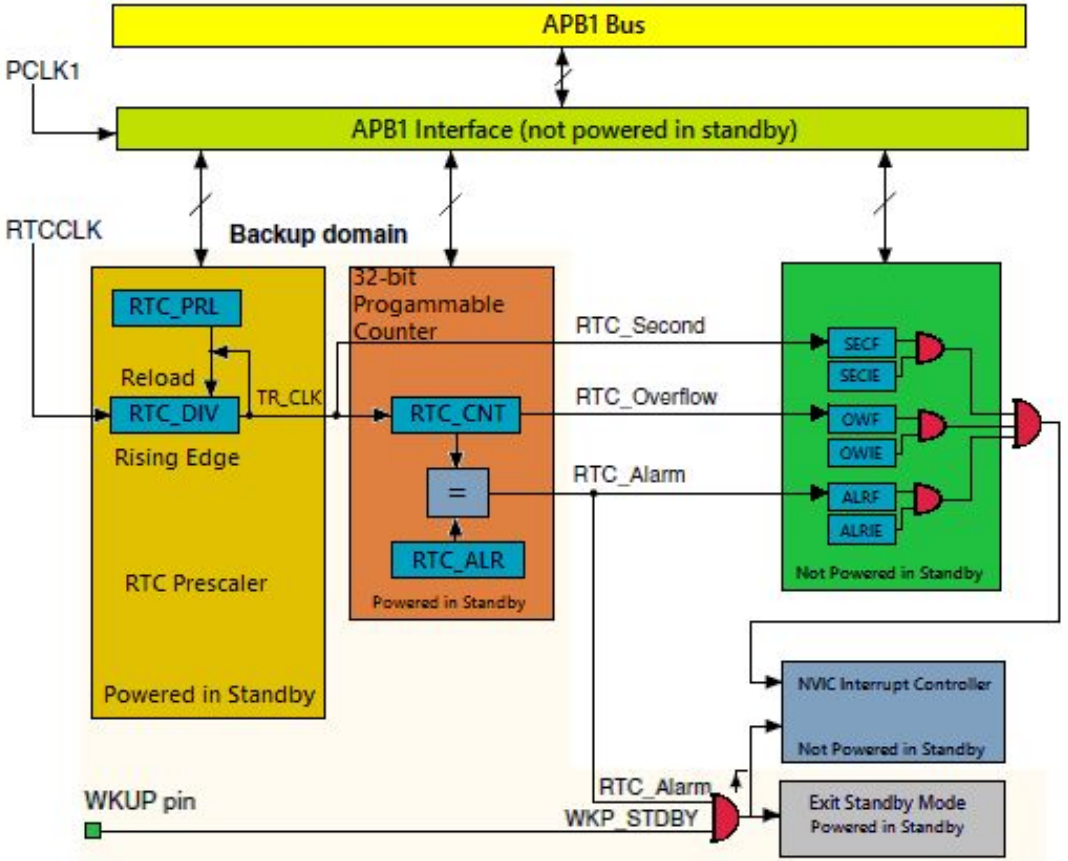
RTC & PWM

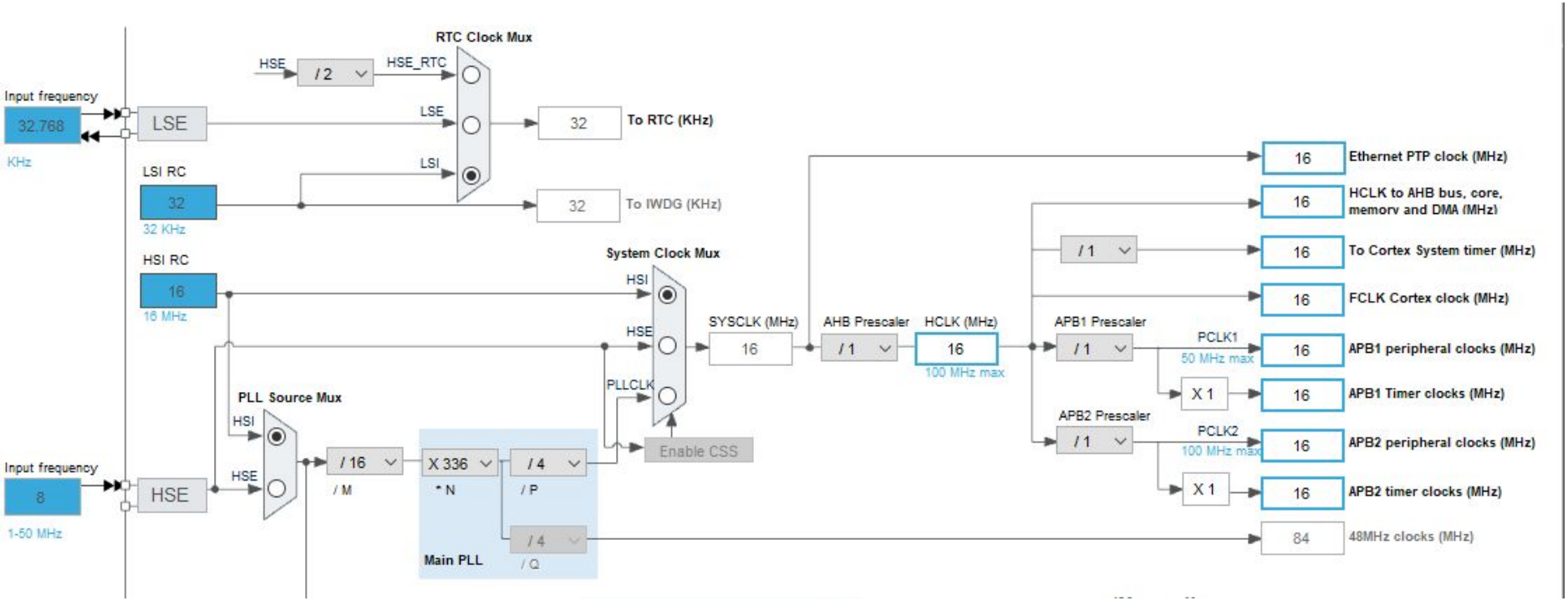

CLK A
CLK B
Input Pin
MUX
Prescale Counter
Main Counter
Modulus Register
Input Pin
Control
Output Pin
DMA, A/D, Etc.
Interrupt Controller

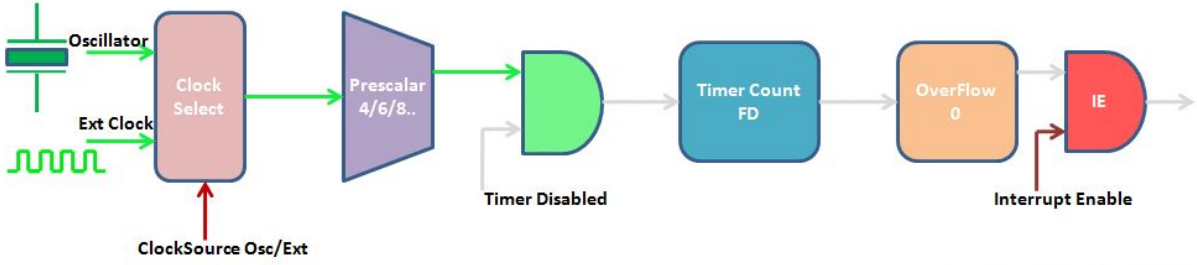
RTC

Provide Date and Time accurately

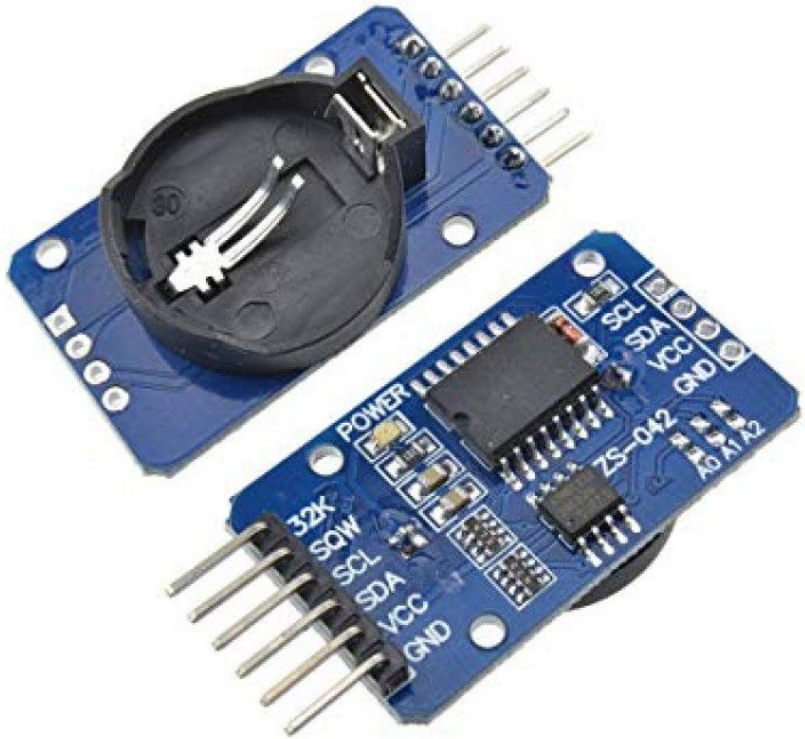


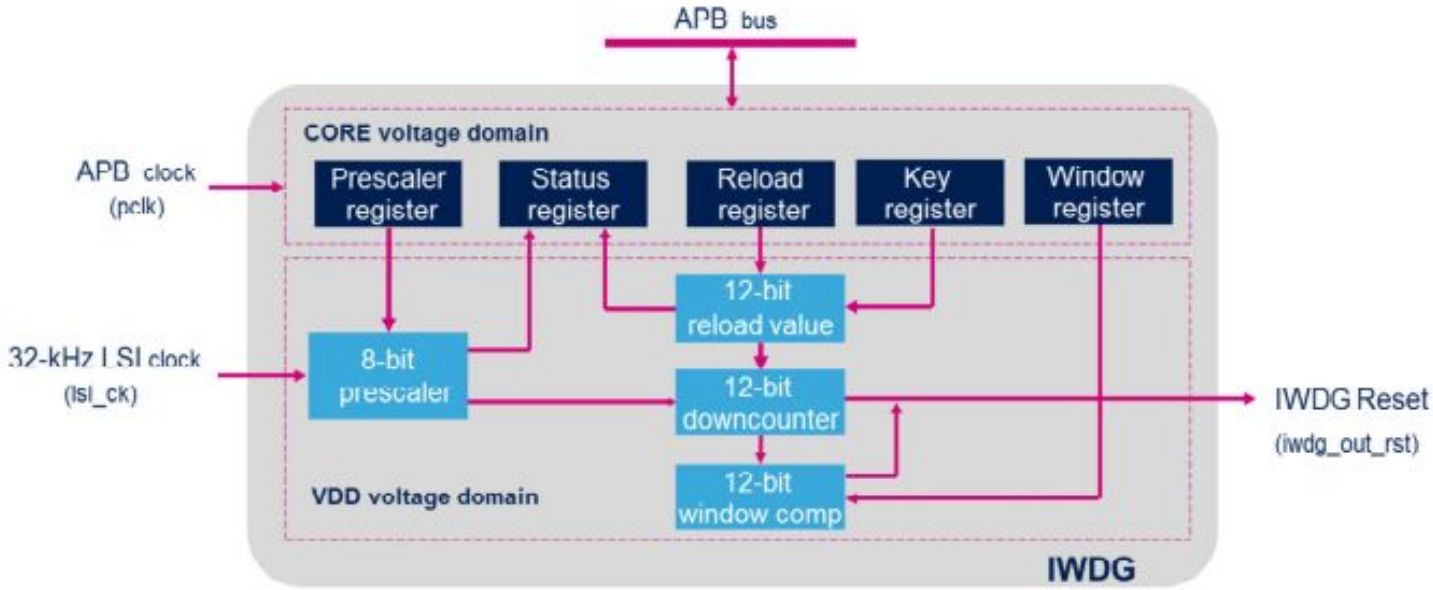



RTC Clock Mux
HSE
/ 2
HSE_RTC
Input frequency
32.768
KHz
LSE
LSE
32
To RTC (KHz)
LSI RC
32
32 KHz
LSI
32
To IWDG (KHz)
HSI RC
16
16 MHz
System Clock Mux
HSI
HSE
PLLCLK
SYSCLK (MHz)
16
AHB Prescaler
/ 1
HCLK (MHz)
16
100 MHz max
16
Ethernet PTP clock (MHz)
16
HCLK to AHB bus, core, memory and DMA (MHz)
/ 1
16
To Cortex System timer (MHz)
16
FCLK Cortex clock (MHz)
APB1 Prescaler
/ 1
PCLK1
50 MHz max
16
APB1 peripheral clocks (MHz)
X 1
16
APB1 Timer clocks (MHz)
APB2 Prescaler
/ 1
PCLK2
100 MHz max
16
APB2 peripheral clocks (MHz)
X 1
16
APB2 timer clocks (MHz)
84
48MHz clocks (MHz)
PLL Source Mux
HSI
HSE
Input frequency
8
1-50 MHz
HSE
/ 16
/ M
X 336
* N
/ 4
/ P
/ 4
/ Q
Main PLL
Enable CSS


Oscillator
Ext Clock
Clock Select
ClockSource Osc/Ext
Prescalar 4/6/8..
Timer Disabled
Timer Count FD
OverFlow 0
IE
Interrupt Enable

An external RTC allows you to keep the clock running while the microcontroller is powered down. Save on other external components when using a small RTC with integrated functionality.





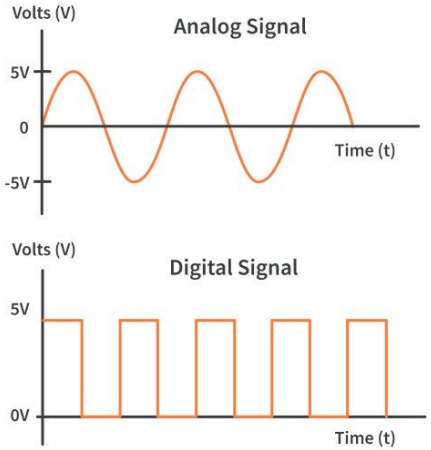
Two clocks are needed:

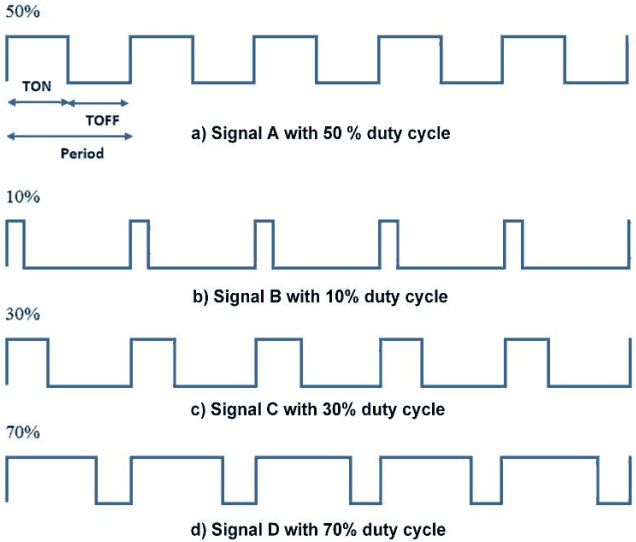
1. The APB clock is required in order to access registers
2. The LSI clock is required for the functional part of the watchdog

Independent watchdog (IWDG)

- It triggers a reset sequence when it is not refreshed within the expected time-window (optional)
- Since its clock is an independent 32-kHz low-speed internal RC oscillator (LSI), it remains active even if the main clock fails.
- Once enabled, it forces the activation of the low-speed internal oscillator, and it can only be disabled by a reset.
- One of the main benefits for applications is its ability to run independently from the main clock
- It is clocked by a 32-kHz RC oscillator which cannot be disabled when the independent watchdog is enabled.

Pulse Width Modulation

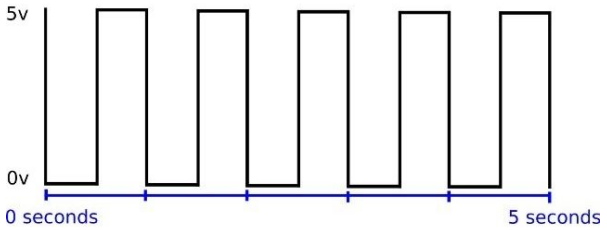





```
Duty Cycle =Turn ON time/ (Turn ON time + Turn OFF time)
```

```
Frequency = 1/Time Period
 Time Period = On time + Off time
```

Pulse Width Modulation



- **Frequency of a PWM**
- **How to convert PWM signals into Analog Voltage?**
- **How to calculate the output voltage of PWM signal?**

Setup code —

```
uint16_t pwm = 0;

HAL_TIM_PWM_Start(&htim2, TIM_CHANNEL_1);
```

Operational code —-

```
while(pwm < htim2.Init.Period)
  {
        pwm+=100;
        __HAL_TIM_SET_COMPARE(&htim2, TIM_CHANNEL_1, pwm);
        HAL_Delay(1);  //wait for 1 ms
  }

  while(pwm > 0)
  {
        pwm-=100;
        __HAL_TIM_SET_COMPARE(&htim2, TIM_CHANNEL_1, pwm);
        HAL_Delay(1);  //wait for 1 ms
  }
```

Configuring/Program internal RTC

Setup code —-

```
#include <stdio.h>
#include <string.h>

char time[30];
char date[30];

RTC_TimeTypeDef sTime = {0};
RTC_DateTypeDef sDate = {0};
```

Operational code —-

```
HAL_RTC_GetTime(&hrtc, &sTime, RTC_FORMAT_BIN);
HAL_RTC_GetDate(&hrtc, &sDate, RTC_FORMAT_BIN);

sprintf(date, "Date:%02d.%02d.%02d\t",
sDate.Date,sDate.Month,sDate.Year);
sprintf(time, "Time: %02d.%02d.%02d\r\n",
sTime.Hours,sTime.Minutes,sTime.Seconds);

HAL_UART_Transmit(&huart2, (uint8_t *)date, sizeof(date), 300);
HAL_UART_Transmit(&huart2, (uint8_t *)time, sizeof(time), 300);
HAL_Delay(1000);
```