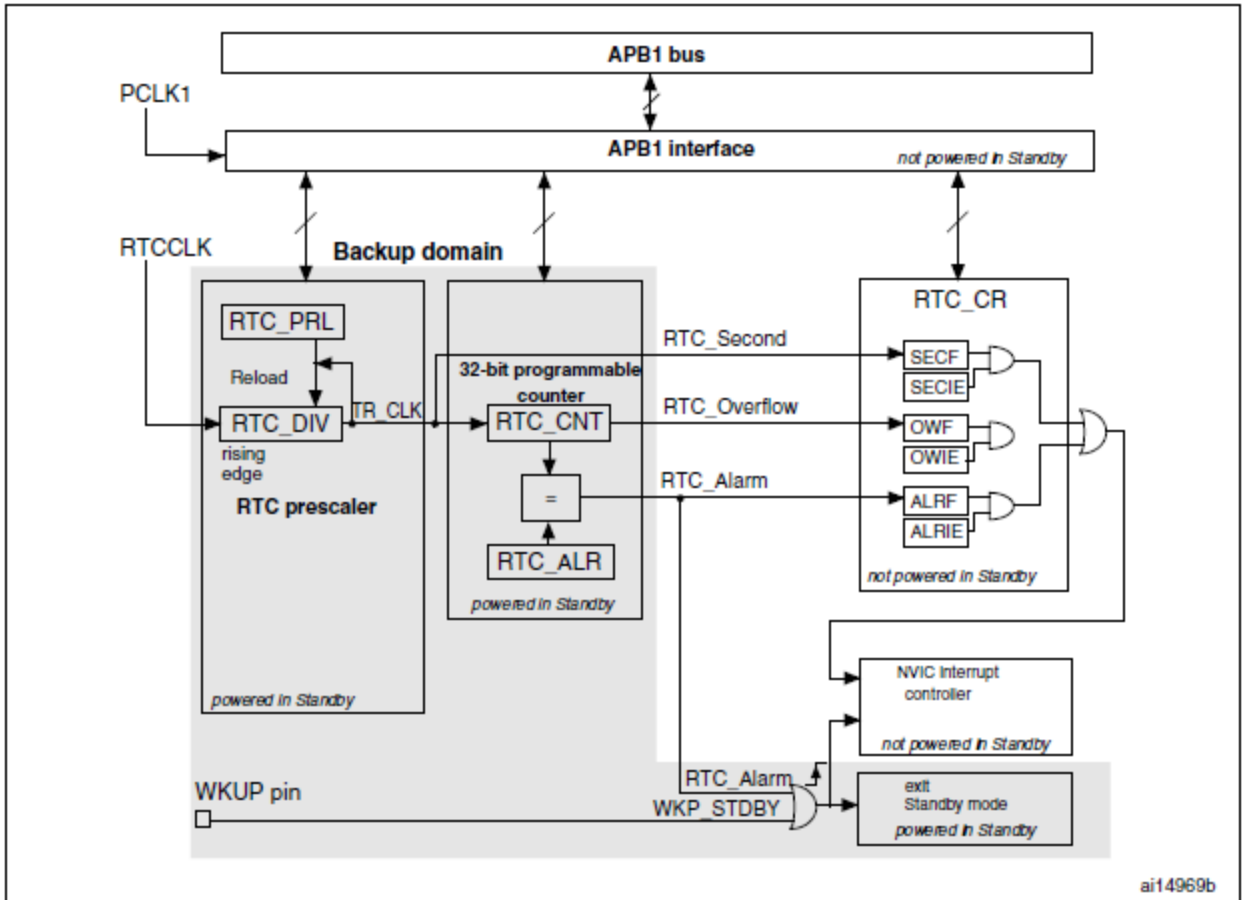
# Real-Time Clock

# RTC features

- Programmable prescaler
- 32-bit programmable counter
- RTC clock source could be any of the following:
  - HSE divided by 128
  - LSE
  - LSI

# RTC block diagram



# RTC overview

- RTC consists of two main units:
  - APB1 interface
  - RTC core
- APB1 is used to interface to APB1 bus
- APB1 interface contains a set of registers for read/write
- APB1 interface is clocked by APB1 bus clock

# RTC core

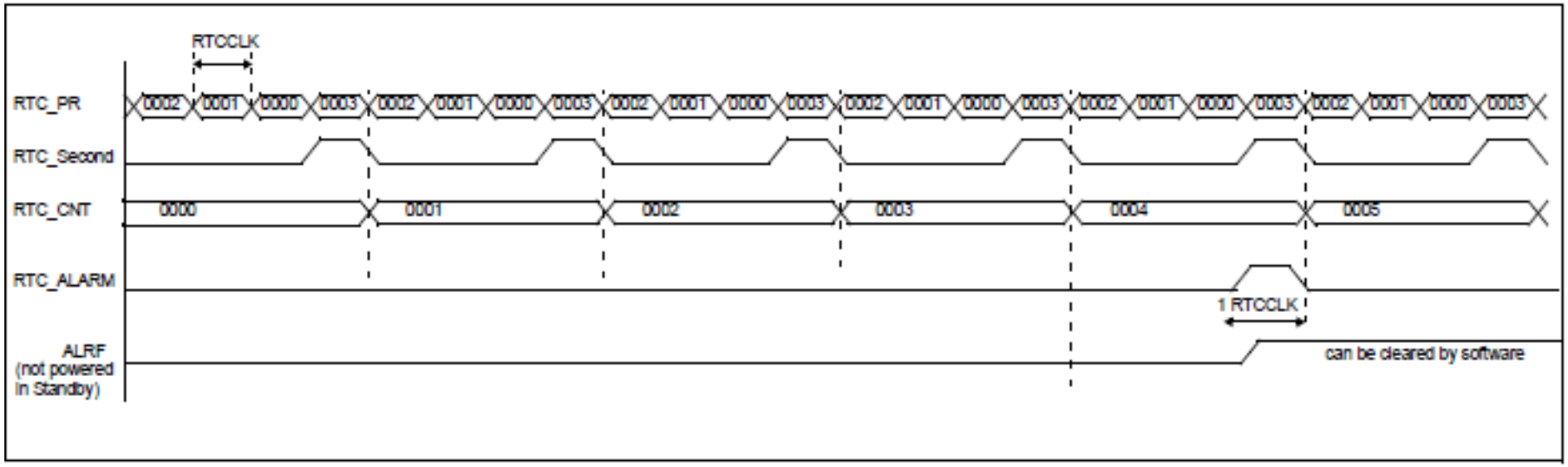
- It consists of two main blocks
  - RTC prescaler
  - 32 bit programmable counter
- RTC prescaler block which generates TR_CLK that can be programmable to have a period up to 1 second
- Every TR_CLK period, RTC generates an interrupt if it is enable in RTC_CR register

# RTC core

- The programmable counter can be initialized to current time

- Alarm interrupt can be set if the programmable counter equal to the value in RTC_ALR register and alarm interrupt is enable in RTC_CR register

# Alarm interrupt example PR=003 and ALARM=004



# Example of code

```
int main(){
  RCC_setup();
  GPIO_setup();
  NVIC_setup();
  RTC_setup();
  while(1);
}
```

```
void GPIO_setup(){
  GPIO_InitTypeDef GPIO_InitStructure;
  RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOC,ENABLE);
  // enable GPIOC clock
  // configure PC6 as output- push-pull connect with LED
  GPIO_InitStructure.GPIO_Pin = GPIO_Pin_6;
  GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
  GPIO_InitStructure.GPIO_Mode = GPIO_Mode_Out_PP;
  GPIO_Init(GPIOC, & GPIO_InitStructure);
}

void  NVIC_setup(){
  NVIC_InitTypeDef NVIC_InitStructure;
  NVIC_PriorityGroupConfig(NVIC_PriorityGroup_1);
  NVIC_InitStructure.NVIC_IRQChannel = RTC_IRQChannel;
  NVIC_InitStructure.NVIC_IRQChannelPreemptionPriority = 1;
  NVIC_InitStructure.NVIC_IRQChannelSubPriority = 0;
  NVIC_InitStructure.NVIC_IRQChannelCmd = ENABLE;
  NVIC_Init(&NVIC_InitStructure);
}
```

```
void RTC_setup(){
   RCC_APB1PeriphClockCmd(RCc_aPB1Periph_PWR,ENABLE);  // Enable clock
   PWR_BackupAccessCmd(ENABLE);  // for access to RTC module
   RCC_LSEConfig(RCC_LSE_ON);       // Enable LSE
   while(RCC_GetFlagStatus(RCC_FLAG_LSERDY) == RESET);  // wait til LSE ready
   RCC_RTCCLKConfig(RCC_RTCCLKSource_LSE);  //select LSE as RTC clock source
   RCC_RTCCLKCmd(ENABLE);  // Enable RTC Clock
   RCC_WaitForSynchro();  // wait for RTC register Synchronize
   RCC_WaitForLastTask();  // wait until last write operation on RTC has finished
   RCC_ITConfig(RTC_IT_SEC, ENABLE);  // enable RTC second
   RCC_WaitForLastTask();  // wait until last write operation on RTC has finished
   RCC_SetPrescaler(32767); // RTC period = RTCCLK/RTC_PR = 32768/(32767+1)
   RCC_WaitForLastTask();  // wait until last write operation on RTC has finished
}
```

```
void RTC_IRQHandler(void){
   if(RTC_GetITStatus(RTC_IT_SEC) != RESET){
       GPIO_WriteBit(GPIOC,GPIO_Pin_6, (BitAction)(1-
      GPIO_ReadOutputDataBit(GPIOC,GPIO_Pin_6))
      RTC_WaitforLastTask();
       RTC_Clear_PendingBit(RTC_IT_SEC);  // clear RTC second interrupt
   }
}
```

# Questions?