Real-time Clock(RTC) trên STM32F4

## **Kiến thức cần chuẩn bị**

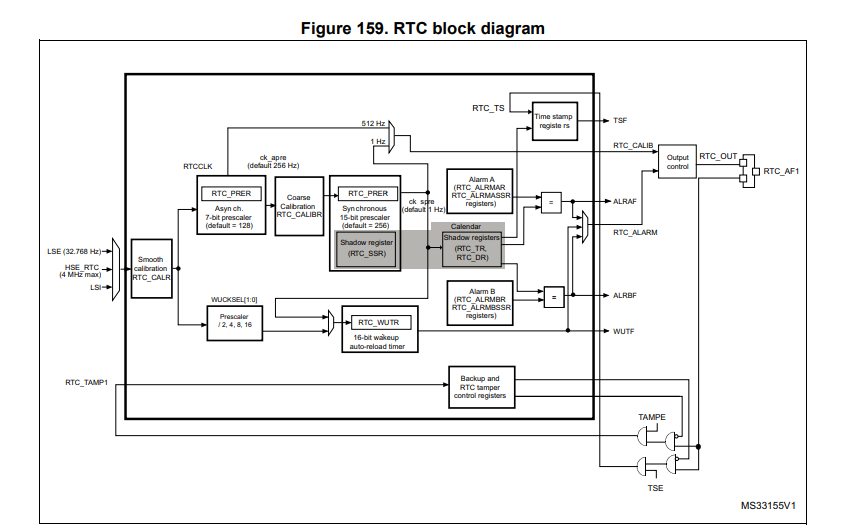
## 1 . Real-time Clock

**The Real-time Clock (RTC) là bộ đếm thời gian độc lập ,ta có thể hiểu đơn giản như một chiếc đồng hồ bình thường** . RTC cung cấp lịch , thời gian trong ngày, hai Alarm (A và B)có thể lập trình và lập trình định kỳ cờ wakeup với khả năng ngắt. RTC cũng bao gồm một đơn vị đánh thức tự động để quản lý Low Power Modes.

Hai thanh ghi 32 bit chứa giây, phút, giờ (định dạng 12 hoặc 24 giờ), ngày (ngày của tuần), ngày (ngày trong tháng), tháng và năm, được biểu thị ở định dạng thập phân được mã hóa nhị phân (BCD). Giá trị giây phụ cũng có sẵn ở định dạng nhị phân. Bù cho các tháng 28-, 29- (năm nhuận), 30- và 31 ngày được thực hiện tự động. Các thanh ghi 32 bit bổ sung chứa các subseconds báo động có thể lập trình, giây, phút, giờ, ngày và ngày trong tuần .

Một tính năng hiệu chuẩn kỹ thuật số có sẵn để bù cho bất kỳ sai lệch nào trong bộ tạo dao động tinh thể hoạt động chính xác.

Miễn là điện áp cung cấp vẫn nằm trong phạm vi hoạt động, RTC không bao giờ dừng lại, Bất kể trạng thái Chip (Run Mode , Low Power Modes hoặc được Reset).



Với các loại module hiện có trên thị trường như DS3231, DS1307… chúng ta phải dùng thêm IC để đọc được dữ liệu thời gian về ngày, tháng, năm, giờ, phút, giây và các loại IC này đều sử dụng giao thức I2C để đọc/ghi dữ liệu.

Còn đối với chip STM32F4 ở bên trong nó đã tích hợp sẵn một bộ thời gian thực.

**Ưu và nhược điểm khi sử dụng bộ RTC trong chip STM32F4**:

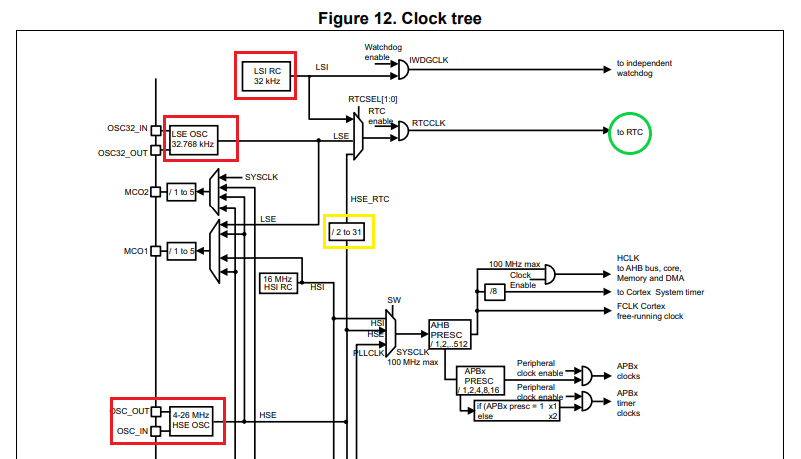
*Ưu điểm*: Không phải tốn chi phí cho bất kì IC RTC nào vì đã được tích hợp sẵn, tiết kiểm diện tích thiết kế mạch.

*Nhược điểm*: Bộ RTC trong chip STM32F4 sử dụng từ Clock từ các bộ LSI, LSE, HSE. Nếu sử dụng LSI làm bộ nguồn Clock thì đây là bộ clock nội và sai số , vì vậy trong quá trình hoạt động thì khi chúng ta đọc thời gian sẽ bị sai lệch .

Một số ứng dụng chính mà bộ RTC mang lại là làm đồng hồ, mạch kiểm soát thời gian, báo thức, counter…Bộ RTC này sử dụng timer độc lập, tách biệt với các bộ timer khác .

**Nguồn clock cấp cho bộ RTC hoạt động :**

* HSE : sử dụng thạch anh ngoài tốc độ cao, từ 8MHZ sẽ được chia từ 2 đến 31 lần để ra tần số hoạt động (rất cao)
* LSI RC: sử dụng bộ giao động RC nội tốc độ 32Khz.
* LSE : sử dụng thạc anh ngoài tốc độ thấp 32.768khz.



Thạch anh ngoài giúp bộ MCU hoạt động chính xác và tiết kiệm năng lượng hơn so với bộ giao động RC nội . Khi cần Backup data khi mất nguồn trên chân VDD thì cần có 2 điều kiện là sử dụng thạch anh ngoài và có điện áp trên chân VBAT. Nhưng trên Chip ta không cần sử dụng đến thạch anh ngoài mà thường sử dụng clock nội

**Các chức năng cơ bản của bộ RTC:**

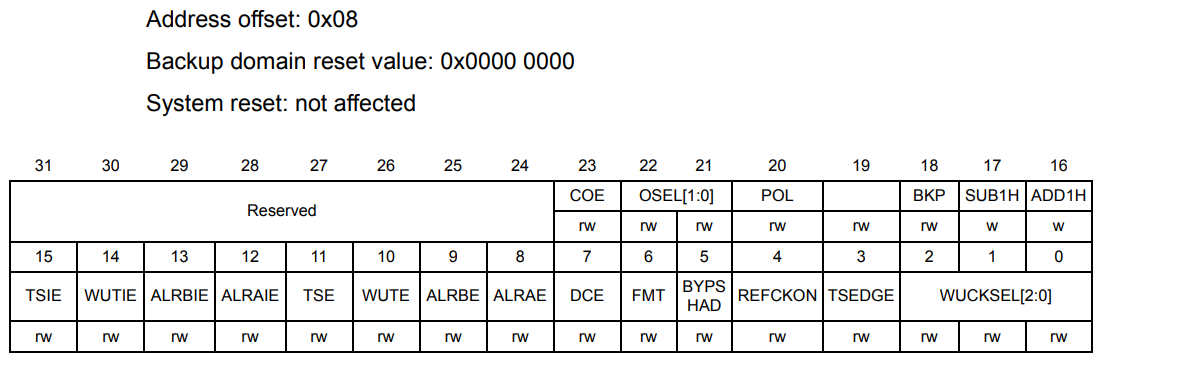
* Bộ chia clock lên đến 20 bit, giúp bộ RTC hoạt động chính xác.
* Độ phân giải của timer RTC lên đến 32 bit tức là 232 giây mới tràn và cần reset lại.
* 3 clock source có thể được sử dụng.
* 2 loại Reset RTC riêng biệt.
* Có các ngắt hỗ trợ là : ngắt Alarm, ngắt mỗi giây, ngắt tràn.

Hãy cùng phân tích một số thanh ghi quan trọng trong RTC.

Các thanh ghi ngày và giờ của RTC được truy cập thông qua các ***Shadow register*** được đồng bộ hóa với PCLK1 ( APB1 clock ). APB1 Interface giúp cho Core có thể đọc ghi dữ liệu đến các thanh ghi trong bộ RTC thông qua APB1 bus. Ngoài ra, APB1 interface sẽ được APB1 bus clock trong quá trình giao tiếp dữ liệu.

**Một số thanh ghi quan trọng trong bài:**

**1. RTC\_CR – RTC control register**.



COE: Kích hoạt hoặc Vô hiệu hóa chuẩn đầu ra

OSEL[1:0]: Output selection (chọn cờ được định tuyến đến RTC\_ALARM output)

POL: Output polarity (sử dụng để định cấu hình mức logic xuất ra của RTC\_ALARM output)

COSEL: Calibration output selection

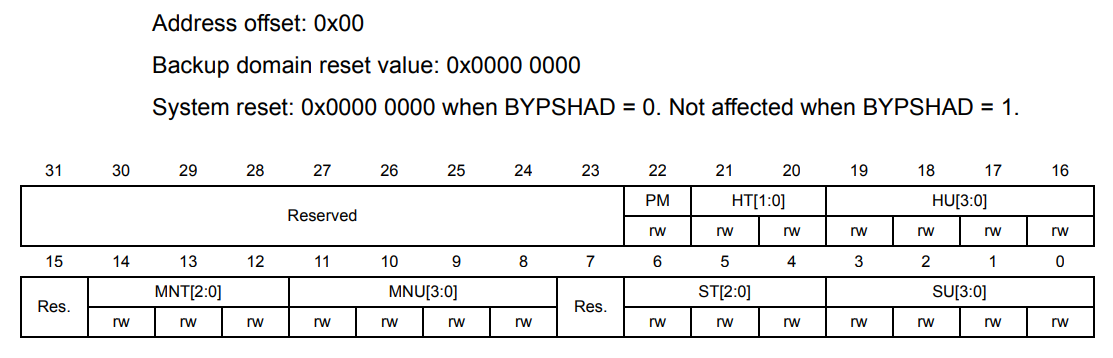
BKP: Backup

SUB1H: Subtract 1 hour (winter time change)

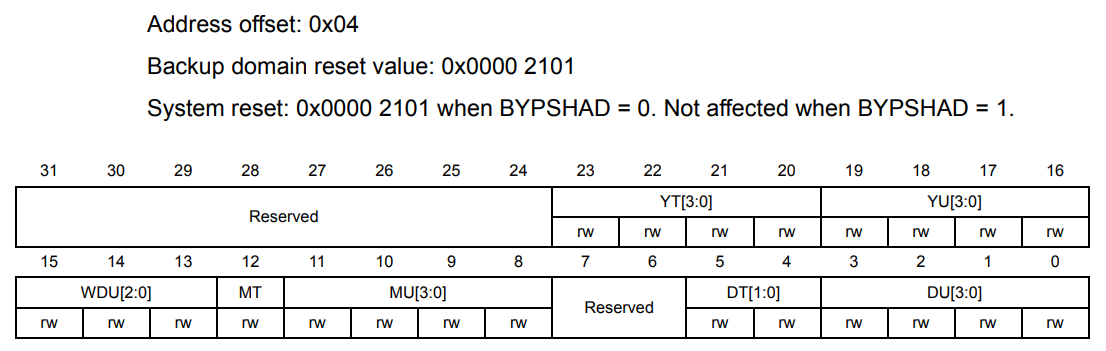
.... Tham khảo thêm các chức năng của thanh ghi tại page 489 tài liệu của ST

<https://www.st.com/content/ccc/resource/technical/document/reference_manual/9b/53/39/1c/f7/01/4a/79/DM00119316.pdf/files/DM00119316.pdf/jcr:content/translations/en.DM00119316.pdf>

**2. RTC\_TR – RTC time register.**

  
Thanh ghi chứa thời gian ở định dạng Binary Code Decima

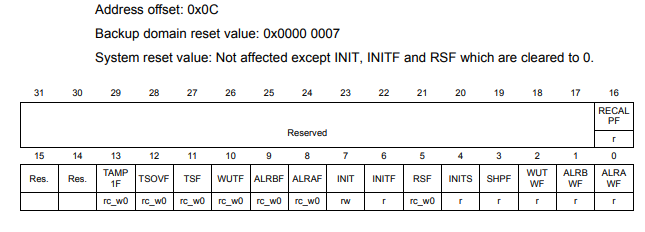
**3. RTC\_DR – RTC date register.**



Thanh ghi chứa ngày dương lịch ở định dạng Binary Code Decima

**4 . RTC\_ISR - RTC initialization and status register**

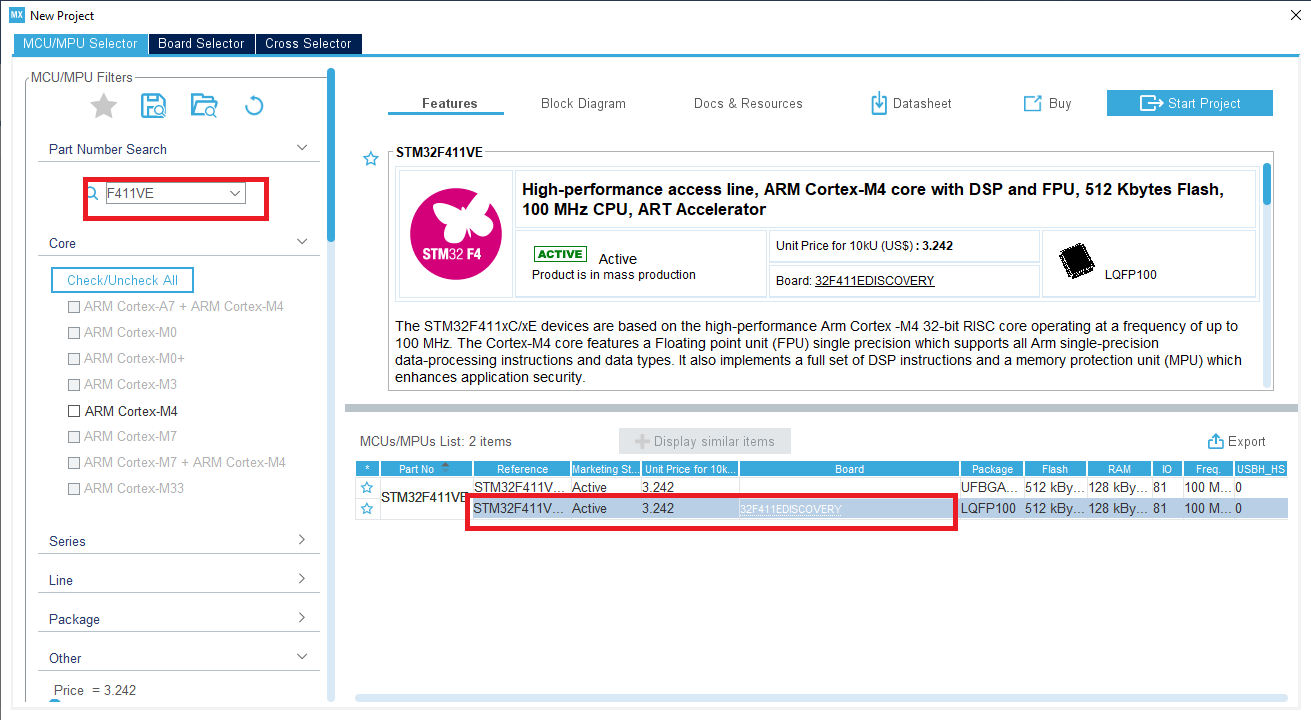
Thanh ghi khởi tạo và trạng thái RTC



## **Lập trình**

Mở phần mềm STM CubeMX, chọn dòng chip bạn sử dụng. Ở đây mình chọn chip STM32F411VE

Đối với các dòng chip STM32 đời 4, tất cả mọi câu lệnh khi sử dụng thư viện HAL đều giống nhau. Chỉ khác nhau phần cấu hình Clock phụ thuộc riêng vào mỗi Chip

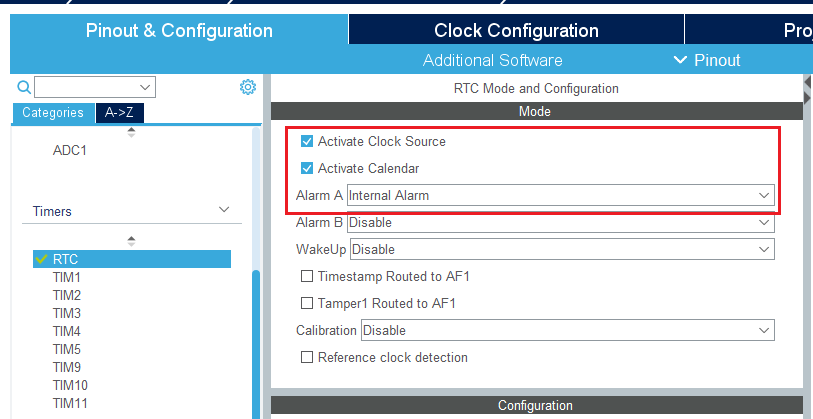


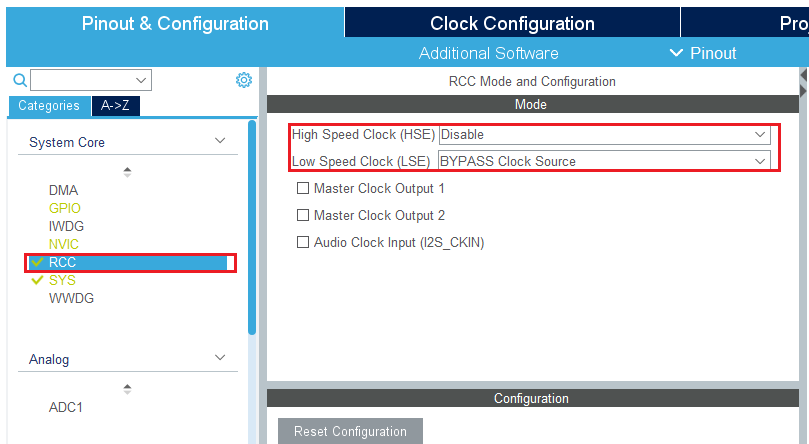
Cấu hình Chip Debug bằng mode SWD

A screenshot of a social media post

Description automatically generated

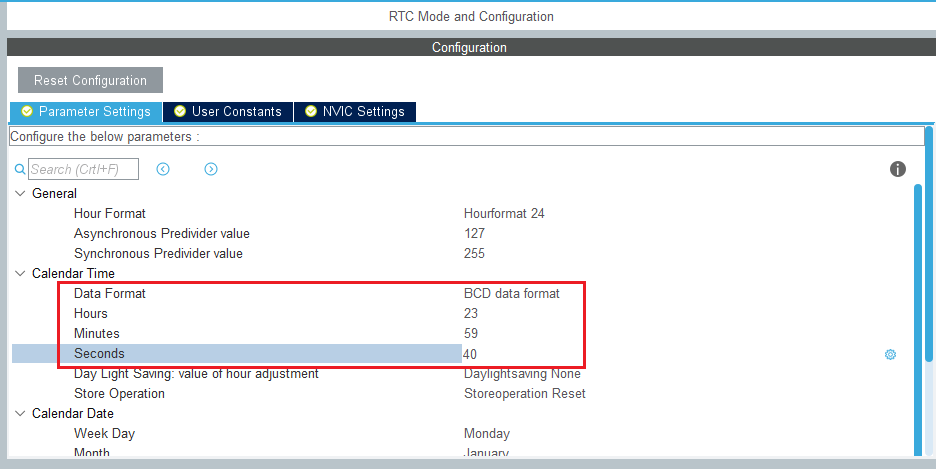
Kích hoạt nguồn Clock và Calender cho RTC và sẽ chọn Clock source là Thạch anh ngoài tốc độ thấp hoặc nội Clock LSI . Với những bài cần độ chính xác cao và tiết kiệm năng lượng ta sẽ dùng LSE nhưng với ví dụ đơn giản này chúng ta sẽ dùng LSI.

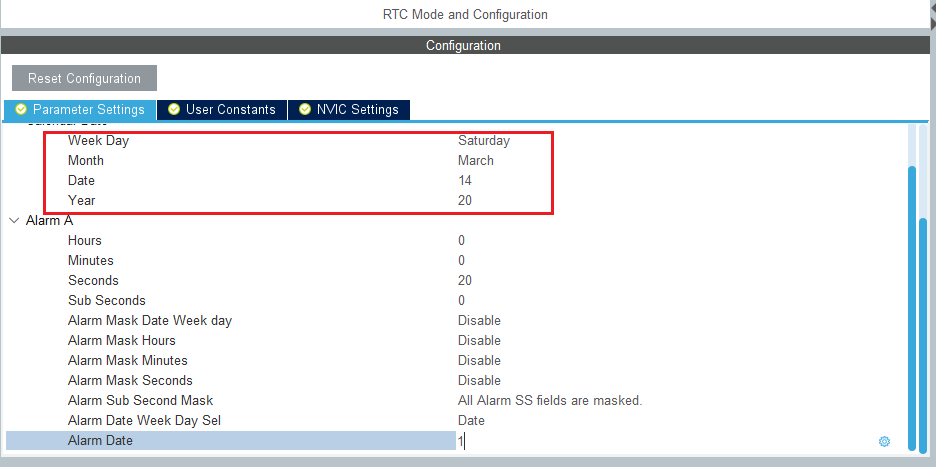


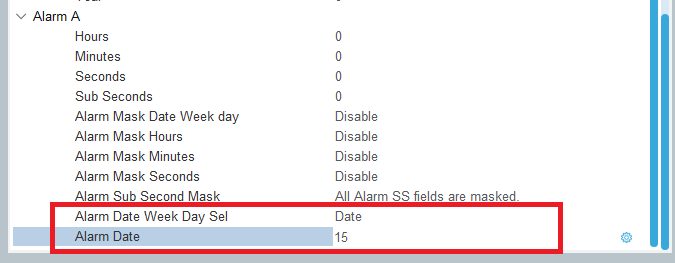


Sau đó cấu hình chi tiết cho RTC . Định dạng giờ là 24 truyền dữ liệu 8 bit

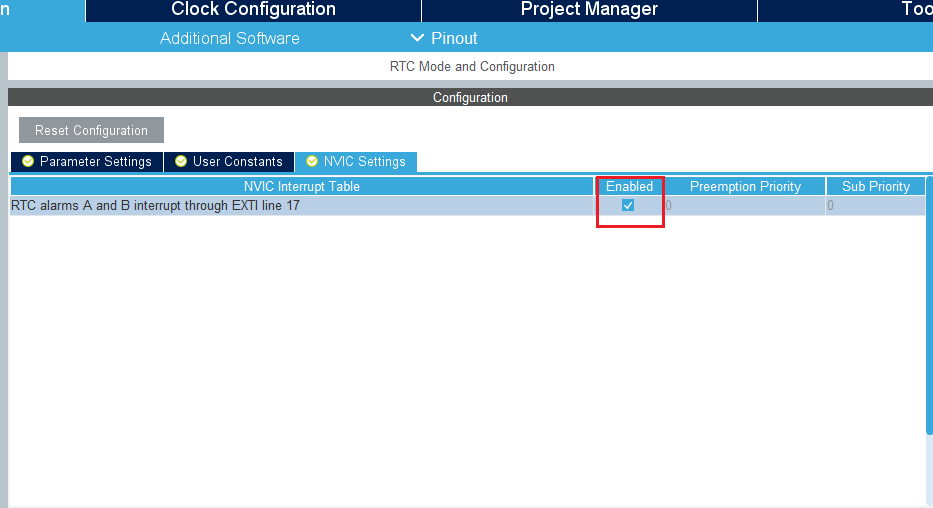
Định dạng dữ liệu là BCD nghĩa là Binary-code Decimal : Số thập phân được mã hóa dưới dạng nhị phân . Set ngày và giờ bắt đầu để lịch đếm . Chú ý nhỏ là định dạng năm trong RTC là 20xx nghĩa là mình sẽ điền 2 số cuối của năm



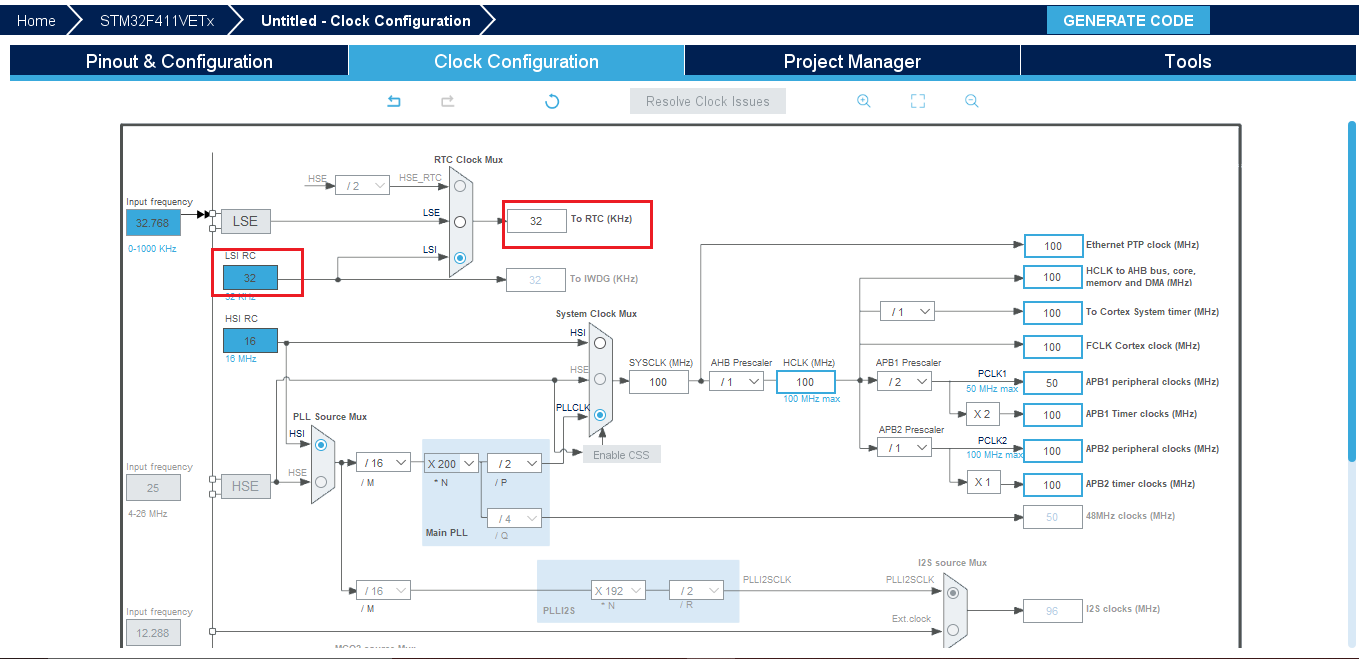




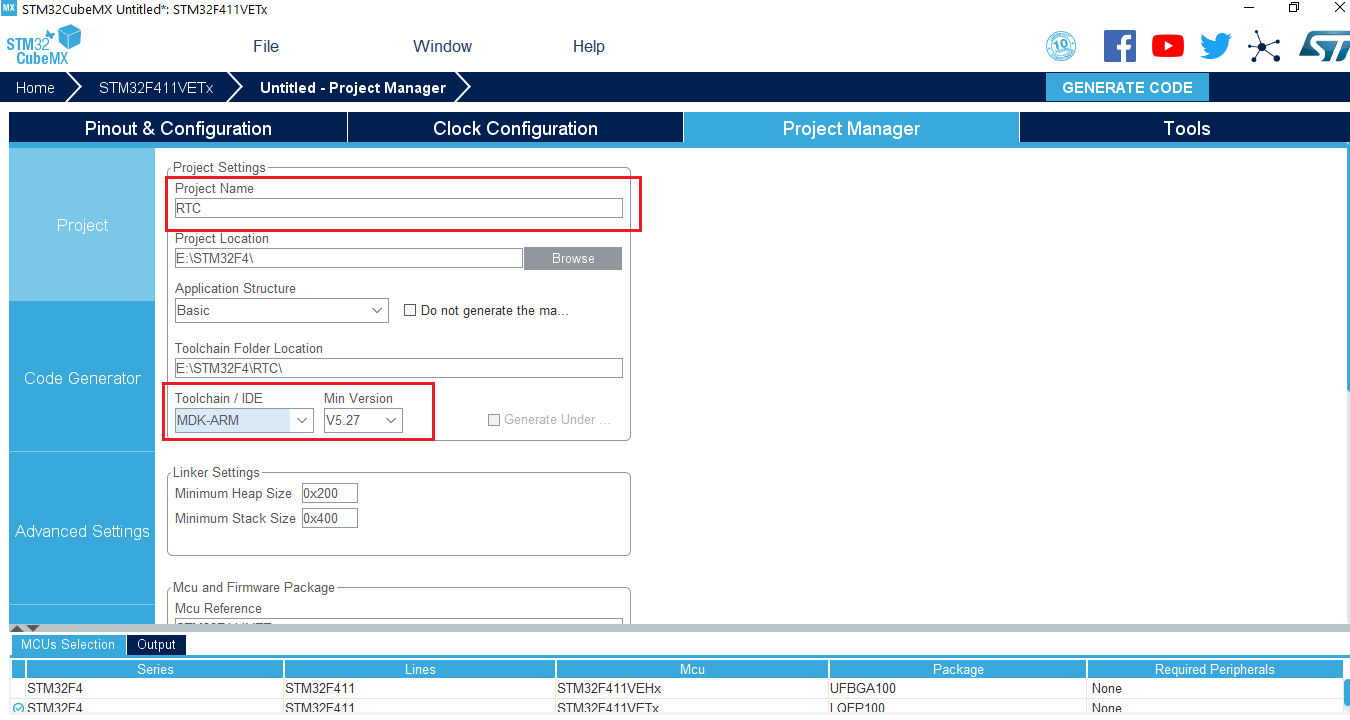
Cho phép ngắt ở Alarm ta có thể chọn thời gian để RTC nhảy vào hàm ngắt Alarm . Ở đây mình chỉ chọn Alarm Date để Demo cho các bạn khi đến 0:0:15 ngày 15



Ta sẽ sử dụng nguồn clock nội LSI 32KHz cấp cho RTC



Click vào Project sau đó Setting , đặt tên cho project , chọn công cụ lập trình Kiel



A screenshot of a social media post

Description automatically generated

Chọn những thư viện cần thiết để sinh code nhanh hơn và giảm dung lượng Project nhé.

Khởi tạo RTC với các thuộc tính đã đặt mà Cube tự sinh ra

static void MX\_RTC\_Init**(**void**)**

**{**

RTC\_TimeTypeDef sTime **=** **{**0**};**

RTC\_DateTypeDef sDate **=** **{**0**};**

RTC\_AlarmTypeDef sAlarm **=** **{**0**};**

hrtc**.**Instance **=** RTC**;**

hrtc**.**Init**.**HourFormat **=** RTC\_HOURFORMAT\_24**;**

hrtc**.**Init**.**AsynchPrediv **=** 127**;**

hrtc**.**Init**.**SynchPrediv **=** 255**;**

hrtc**.**Init**.**OutPut **=** RTC\_OUTPUT\_DISABLE**;**

hrtc**.**Init**.**OutPutPolarity **=** RTC\_OUTPUT\_POLARITY\_HIGH**;**

hrtc**.**Init**.**OutPutType **=** RTC\_OUTPUT\_TYPE\_OPENDRAIN**;**

**if** **(**HAL\_RTC\_Init**(&**hrtc**)** **!=** HAL\_OK**)**

**{**

Error\_Handler**();**

**}**

sTime**.**Hours **=** 0x23**;**

sTime**.**Minutes **=** 0x59**;**

sTime**.**Seconds **=** 0x40**;**

sTime**.**DayLightSaving **=** RTC\_DAYLIGHTSAVING\_NONE**;**

sTime**.**StoreOperation **=** RTC\_STOREOPERATION\_RESET**;**

**if** **(**HAL\_RTC\_SetTime**(&**hrtc**,** **&**sTime**,** RTC\_FORMAT\_BCD**)** **!=** HAL\_OK**)**

**{**

Error\_Handler**();**

**}**

sDate**.**WeekDay **=** RTC\_WEEKDAY\_SATURDAY**;**

sDate**.**Month **=** RTC\_MONTH\_MARCH**;**

sDate**.**Date **=** 0x14**;**

sDate**.**Year **=** 0x20**;**

**if** **(**HAL\_RTC\_SetDate**(&**hrtc**,** **&**sDate**,** RTC\_FORMAT\_BCD**)** **!=** HAL\_OK**)**

**{**

Error\_Handler**();**

**}**

Enable Alarm với 20s

sAlarm**.**AlarmTime**.**Hours **=** 0x0**;**

sAlarm**.**AlarmTime**.**Minutes **=** 0x0**;**

sAlarm**.**AlarmTime**.**Seconds **=** 0x15**; // Thời gian sẽ nhảy vào hàm ngắt**

sAlarm**.**AlarmTime**.**SubSeconds **=** 0x0**;**

sAlarm**.**AlarmTime**.**DayLightSaving **=** RTC\_DAYLIGHTSAVING\_NONE**;**

sAlarm**.**AlarmTime**.**StoreOperation **=** RTC\_STOREOPERATION\_RESET**;**

sAlarm**.**AlarmMask **=** RTC\_ALARMMASK\_NONE**;**

sAlarm**.**AlarmSubSecondMask **=** RTC\_ALARMSUBSECONDMASK\_ALL**;**

sAlarm**.**AlarmDateWeekDaySel **=** RTC\_ALARMDATEWEEKDAYSEL\_DATE**;**

sAlarm**.**AlarmDateWeekDay **=** 0x15**;**

sAlarm**.**Alarm **=** RTC\_ALARM\_A**;**

**if** **(**HAL\_RTC\_SetAlarm\_IT**(&**hrtc**,** **&**sAlarm**,** RTC\_FORMAT\_BCD**)** **!=** HAL\_OK**)**

**{**

Error\_Handler**();**

**}**

Ở AlarmDateWeekDay = 15

RTC\_TimeTypeDef sTime**;**

RTC\_DateTypeDef sDate**;**

Khai báo TypeDef cho ngày và thời gian

Trong int main() ta khai báo giờ và ngày tháng set ban đầu cho RTC , trong thư viện stm32f4xx\_hal\_rtc.c lấy thời gian và ngày giờ liên tục *HAL\_TRC\_GetTime();* và *HAL\_RTC\_GetDate();*

sTime**.**Hours**=**23**;**

sTime**.**Minutes**=**59**;**

sTime**.**Seconds**=**40**;**

HAL\_RTC\_SetTime**(&**hrtc**,&**sTime**,**RTC\_FORMAT\_BIN**);**

// set date

sDate**.**Date**=**14**;**

sDate**.**Month**=**RTC\_MONTH\_MARCH**;**

sDate**.**WeekDay **=** RTC\_WEEKDAY\_SATURDAY**;**

sDate**.**Year **=** 20**;**

HAL\_RTC\_SetDate**(&**hrtc**,&**sDate**,**RTC\_FORMAT\_BIN**);**

**while** **(**1**)**

**{**

HAL\_RTC\_GetTime**(&**hrtc**,&**sTime**,**RTC\_FORMAT\_BIN**);**

HAL\_RTC\_GetDate**(&**hrtc**,&**sDate**,**RTC\_FORMAT\_BIN**);**

HAL\_GPIO\_TogglePin**(**GPIOD**,**GPIO\_PIN\_12**);**

HAL\_Delay**(**200**);**

**}**

Bắt đầu đếm từ 23:59:40 ngày thứ 7 14/3/2020

Sau đó sẽ cho hàm getTime và getDate để nhận thời gian liên tục từu bộ đếm

void HAL\_RTC\_AlarmAEventCallback**(**RTC\_HandleTypeDef **\***hrtc**)**

**{**

HAL\_GPIO\_TogglePin**(**GPIOD**,**GPIO\_PIN\_15**);**

**}**

Nhảy vào sự kiện Alarm khi chạy đúng đến thời gian đã set

Mình sẽ giải thích một số hàm

HAL\_RTC\_GetTime**(&**hrtc**,&**sTime**,**RTC\_FORMAT\_BIN**);**

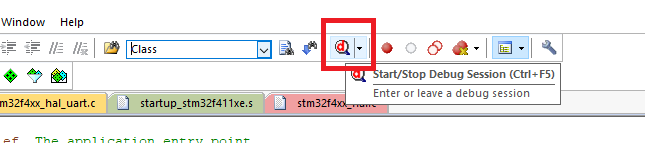
Định dạng thời gian cho RTC truyền và liên tục lấy thời gian trong vòng lặp while

Tương tự với HAL\_RTC\_GetDate();

Tiến hành Build Code và Nạp chương trình xuống KIT

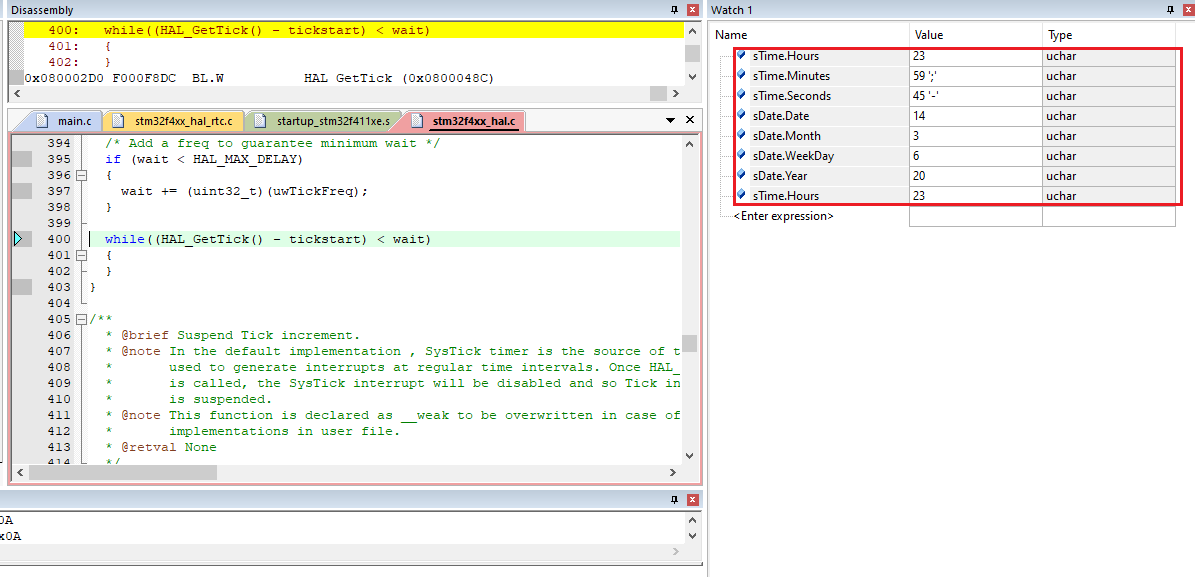
A screenshot of a social media post

Description automatically generated



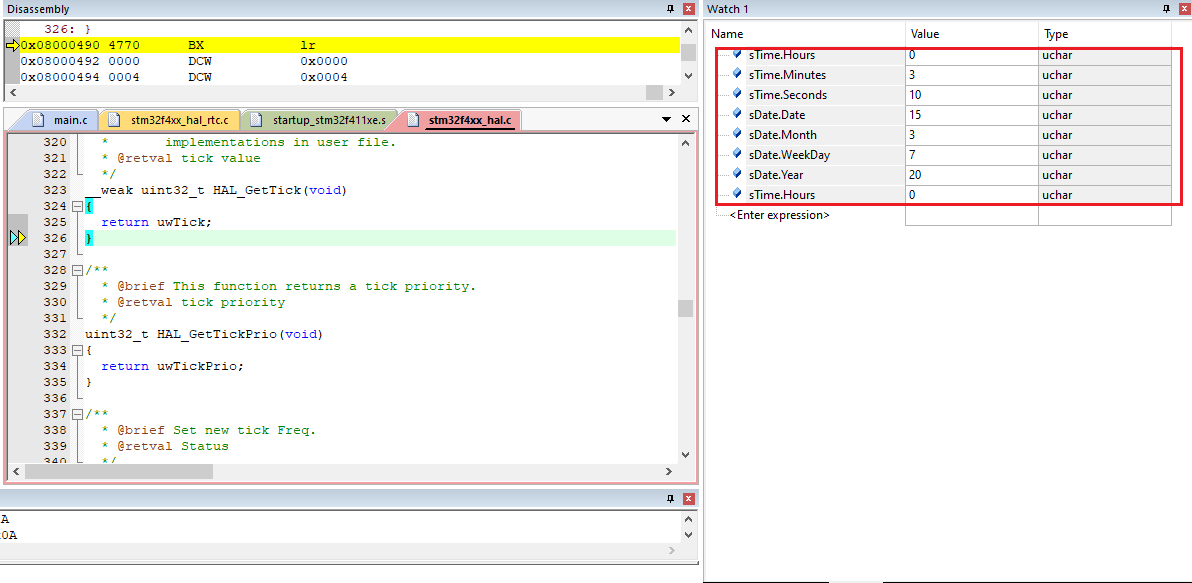
Chuyển các biến cần theo dõi lên Watch

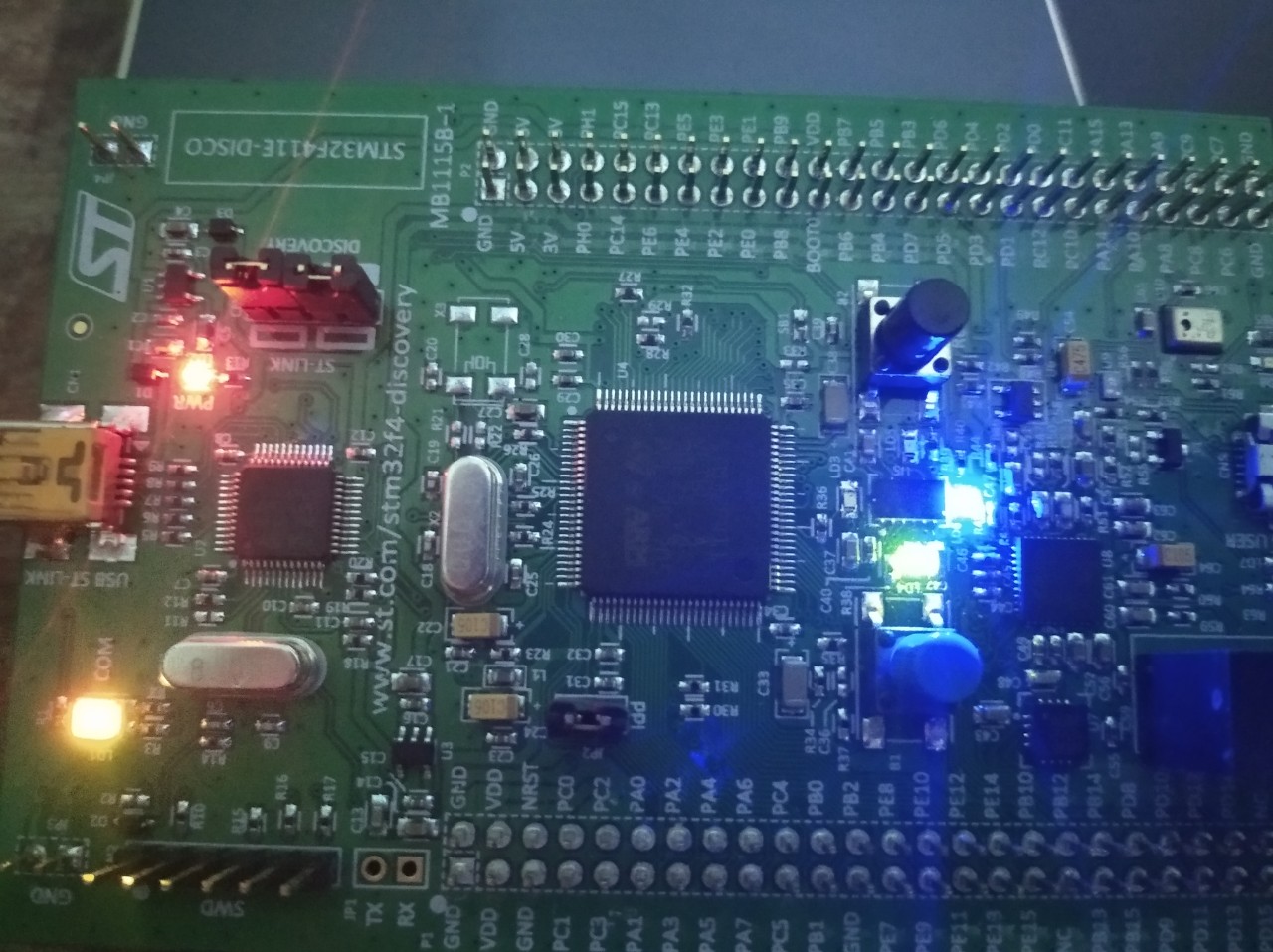
Ta thấy các biến thời gian đã chạy và bắt đầu từ đúng giá trị ta set sau đó tăng dần lên



Khoảng vài phút sau đó







Sau khi qua giây 20 đèn PD15 đã sáng báo xử lý đã ngảy vào hàm ngắt của AlarmA

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

DEVIOT - CÙNG NHAU HỌC LẬP TRÌNH IOT

📌 Website: deviot.vn

📌 FanPage: Deviot - Thời sự kỹ thuật & IoT

📌 Group: Deviot - Cùng nhau học lập trình IOT

📌 Hotline: 0969.666.522

📌 Address: Số 101C, Xã Đàn 2

📌 Đào tạo thật, học thật, làm thật