

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 kiệm diện tích thiết kế mạch.

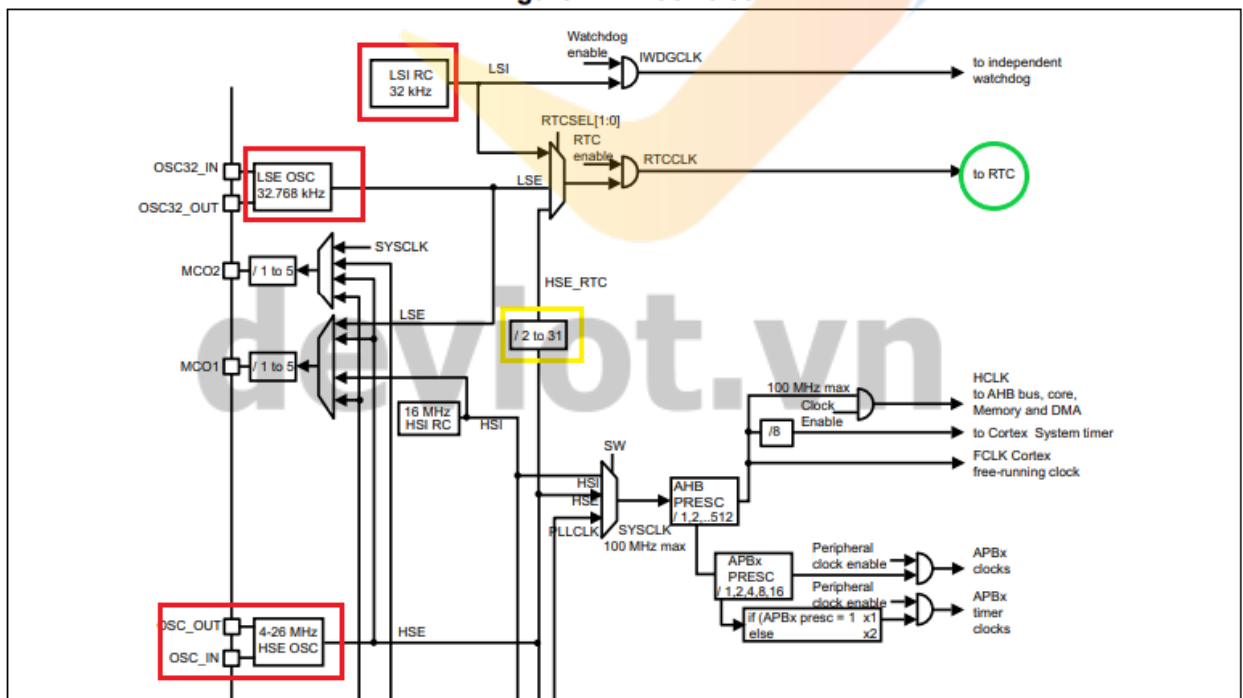
Nhược điểm: Bộ RTC trong chip STM32F4 sử dụng từ Clock từ các bộ LSI, LSE, HSE. Nếu sử dụng LSI làm bộ nguồn Clock thì đây là bộ clock nội và sai số, vì vậy trong quá trình hoạt động thì khi chúng ta đọc thời gian sẽ bị sai lệch.

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 thạch anh ngoài tốc độ thấp 32.768khz.

Figure 12. Clock tree



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 2^{32} 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.

Address offset: 0x08

Backup domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved								COE	OSEL[1:0]		POL		BKP	SUB1H	ADD1H
								r/w	r/w	r/w	r/w	r/w	r/w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TSIE	WUTIE	ALRBIE	ALRAIE	TSE	WUTE	ALRBE	ALRAE	DCE	FMT	BYPHAD	REFCKON	TSEDGE	WUCKSEL[2:0]		
r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w

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.

Address offset: 0x00

Backup domain reset value: 0x0000 0000

System reset: 0x0000 0000 when BYPSHAD = 0. Not affected when BYPSHAD = 1.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved									PM	HT[1:0]		HU[3:0]			
									rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	MNT[2:0]			MNU[3:0]				Res.	ST[2:0]			SU[3:0]			
	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw

Thanh ghi chứa thời gian ở định dạng Binary Code Decima

3. RTC_DR – RTC date register.

Address offset: 0x04

Backup domain reset value: 0x0000 2101

System reset: 0x0000 2101 when BYPSHAD = 0. Not affected when BYPSHAD = 1.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved									YT[3:0]			YU[3:0]			
									rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
WDU[2:0]				MT	MU[3:0]				Reserved		DT[1:0]		DU[3:0]		
rw	rw	rw	rw	rw	rw	rw	rw	rw			rw	rw	rw	rw	rw

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

Address offset: 0x0C

Backup domain reset value: 0x0000 0007

System reset value: Not affected except INIT, INITF and RSF which are cleared to 0.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															RECAL PF
															r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	TAMP 1F	TSOVF	TSF	WUTF	ALRBF	ALRAF	INIT	INITF	RSF	INITS	SHPF	WUT WF	ALRB WF	ALRA WF
		rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rw	r	rc_w0	r	r	r	r	r



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

The screenshot shows the 'New Project' wizard in STM32CubeMX. The 'MCU/MPU Selector' tab is active. In the 'Part Number Search' field, 'F411VE' is entered. The 'Core' dropdown is set to 'ARM Cortex-M4'. The 'Check/Uncheck All' button is visible. The 'Features' tab for 'STM32F411VE' is selected, showing details like 'High-performance access line, ARM Cortex-M4 core with DSP and FPU, 512 Kbytes Flash, 100 MHz CPU, ART Accelerator'. The 'Unit Price for 10kU (US\$): 3.242' and 'Board: 32F411EDISCOVERY' are also displayed. A table at the bottom lists available MCUs/MPUs, with 'STM32F411VE' highlighted.

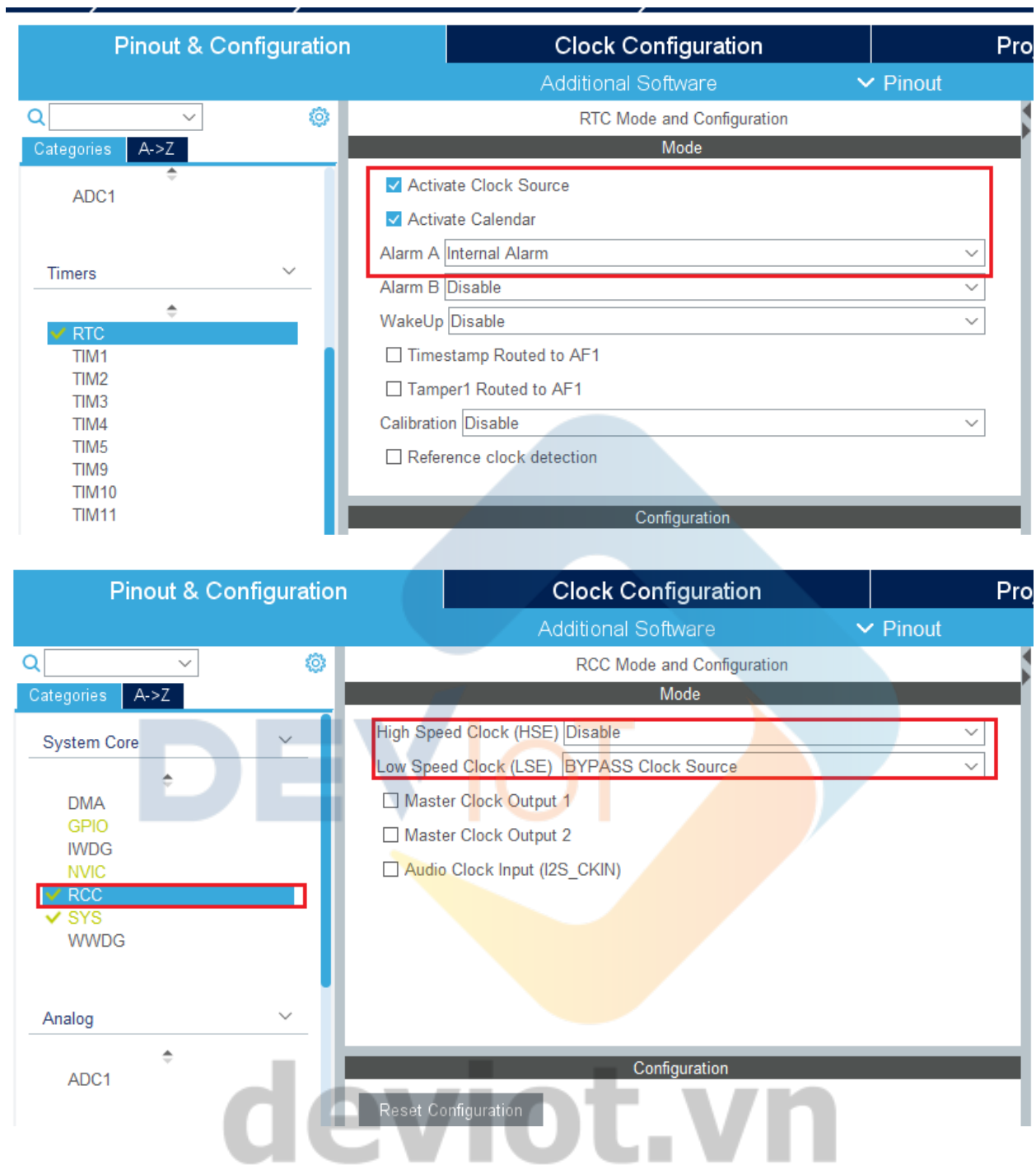
Part No	Reference	Marketing St	Unit Price for 10k	Board	Package	Flash	RAM	IO	Freq	USB_H
STM32F411VE	STM32F411V...	Active	3.242		UFBGA...	512 kBy...	128 kBy...	81	100 M...	0
STM32F411VE	STM32F411V...	Active	3.242	32F411EDISCOVERY	LQFP100	512 kBy...	128 kBy...	81	100 M...	0

Cấu hình Chip Debug bằng mode SWD

The screenshot shows the 'Pinout & Configuration' window in STM32CubeMX. The 'Debug' dropdown is set to 'Serial Wire'. The 'SYS Mode and Configuration' section shows 'SYS' selected. The 'Pinout' view shows the pin configuration for the STM32L476RGTx LQFP64 package. The 'SYS_JTCK-SWCLK' and 'SYS_JTMS-SWDIO' pins are highlighted in red.

Warning: This IP has no parameters to be configured.

Kích hoạt nguồn Clock và Calendar 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

RTC Mode and Configuration

Configuration

Reset Configuration

Parameter SettingsUser ConstantsNVIC Settings

Configure the below parameters :

Search (Ctrl+F)

General

Hour FormatHourformat 24

Asynchronous Predivider value127

Synchronous Predivider value255

Calendar Time

Data FormatBCD data format

Hours23

Minutes59

Seconds40

Day Light Saving: value of hour adjustmentDaylightsaving None

Store OperationStoreoperation Reset

Calendar Date

Week DayMonday

MonthJanuary

RTC Mode and Configuration

Configuration

Reset Configuration

Parameter SettingsUser ConstantsNVIC Settings

Calendar Date

Week DaySaturday

MonthMarch

Date14

Year20

Alarm A

Hours0

Minutes0

Seconds20

Sub Seconds0

Alarm Mask Date Week dayDisable

Alarm Mask HoursDisable

Alarm Mask MinutesDisable

Alarm Mask SecondsDisable

Alarm Sub Second MaskAll Alarm SS fields are masked.

Alarm Date Week Day SelDate

Alarm Date1

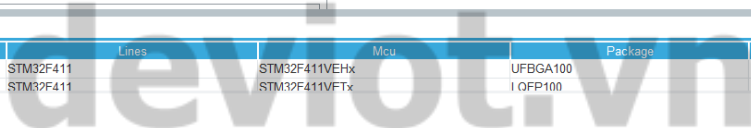
deviot.vn

Alarm A	
Hours	0
Minutes	0
Seconds	0
Sub Seconds	0
Alarm Mask Date Week day	Disable
Alarm Mask Hours	Disable
Alarm Mask Minutes	Disable
Alarm Mask Seconds	Disable
Alarm Sub Second Mask	All Alarm SS fields are masked.
Alarm Date Week Day Sel	Date
Alarm Date	15

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

Additional Software	Pinout	RTC Mode and Configuration
Configuration		
Reset Configuration		
Parameter Settings	User Constants	NVIC Settings
NVIC Interrupt Table		Enabled
RTC alarms A and B interrupt through EXTI line 17		0
		Preemption Priority
		Sub Priority

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



Pinout & Configuration	Clock Configuration	Project Manager
Project	STM32Cube Firmware Library Package <input type="radio"/> Copy all used libraries into the project folder <input checked="" type="radio"/> Copy only the necessary library files <input type="radio"/> Add necessary library files as reference in the toolchain project configuration file	
Code Generator	Generated files <input type="checkbox"/> Generate peripheral initialization as a pair of 'c/h' files per peripheral <input type="checkbox"/> Backup previously generated files when re-generating <input checked="" type="checkbox"/> Keep User Code when re-generating <input checked="" type="checkbox"/> Delete previously generated files when not re-generated	
Advanced Settings	HAL Settings <input type="checkbox"/> Set all free pins as analog (to optimize the power consumption) <input type="checkbox"/> Enable Full Assert	
	Template Settings Select a template to generate customized code Settings...	

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 từ bộ đếm

```
void HAL_RTC_AlarmAEventCallback(RTC_HandleTypeDef *hrtc)  
{  
    HAL_GPIO_TogglePin(GPIOD,GPIO_PIN_15);  
}
```

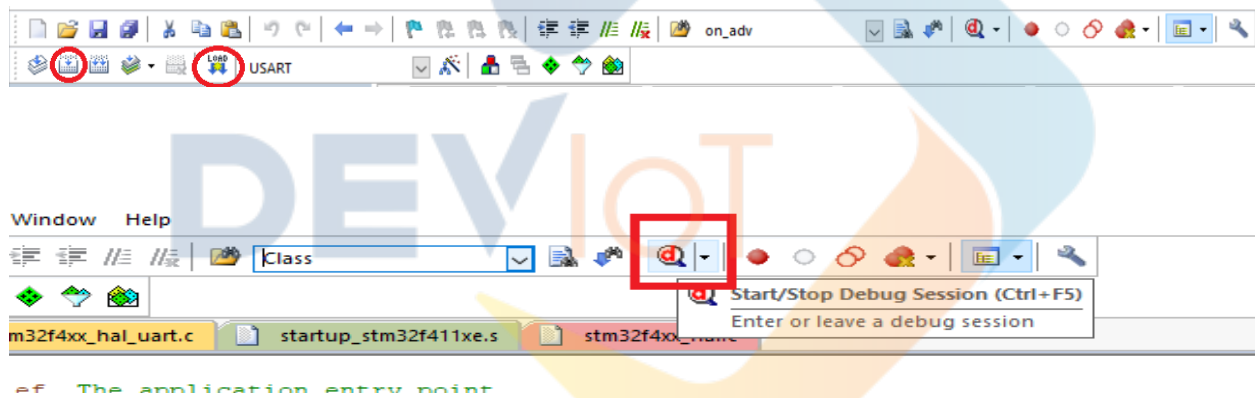
Nhảy vào sự kiện Alarm khi chạy đúng đến thời gian đã set

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



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

deviot.vn

Disassembly

400: while((HAL_GetTick() - tickstart) < wait)

401: {

402: }

0x080002D0 F000F8DC BL.W HAL_GetTick (0x0800048C)

main.c

stm32f4xx_hal_rtc.c

startup_stm32f411xe.s

stm32f4xx_hal.c

394 /* Add a freq to guarantee minimum wait */

395 if (wait < HAL_MAX_DELAY)

396 {

397 wait += (uint32_t)(uwTickFreq);

398 }

399 }

400 while((HAL_GetTick() - tickstart) < wait)

401 {

402 }

403 }

404 }

405 /**

406 * @brief Suspend Tick increment.

407 * @note In the default implementation, SysTick timer is the source of t

408 * used to generate interrupts at regular time intervals. Once HAL_

409 * is called, the SysTick interrupt will be disabled and so Tick in

410 * is suspended.

411 * @note This function is declared as __weak to be overwritten in case of

412 * implementations in user file.

413 * @retval None

414 */

DA

0A

Watch 1

Name	Value	Type
sTime.Hours	23	uchar
sTime.Minutes	59	uchar
sTime.Seconds	45	uchar
sDate.Date	14	uchar
sDate.Month	3	uchar
sDate.WeekDay	6	uchar
sDate.Year	20	uchar
sTime.Hours	23	uchar

<Enter expression>

Khoảng vài phút sau đó



Disassembly

```

326: }
0x08000490 4770 BX 1r
0x08000492 0000 DCW 0x0000
0x08000494 0004 DCW 0x0004

```

main.c | stm324xx_hal_rtc.c | startup_stm32411xe.s | **stm324xx_hal.c**

```

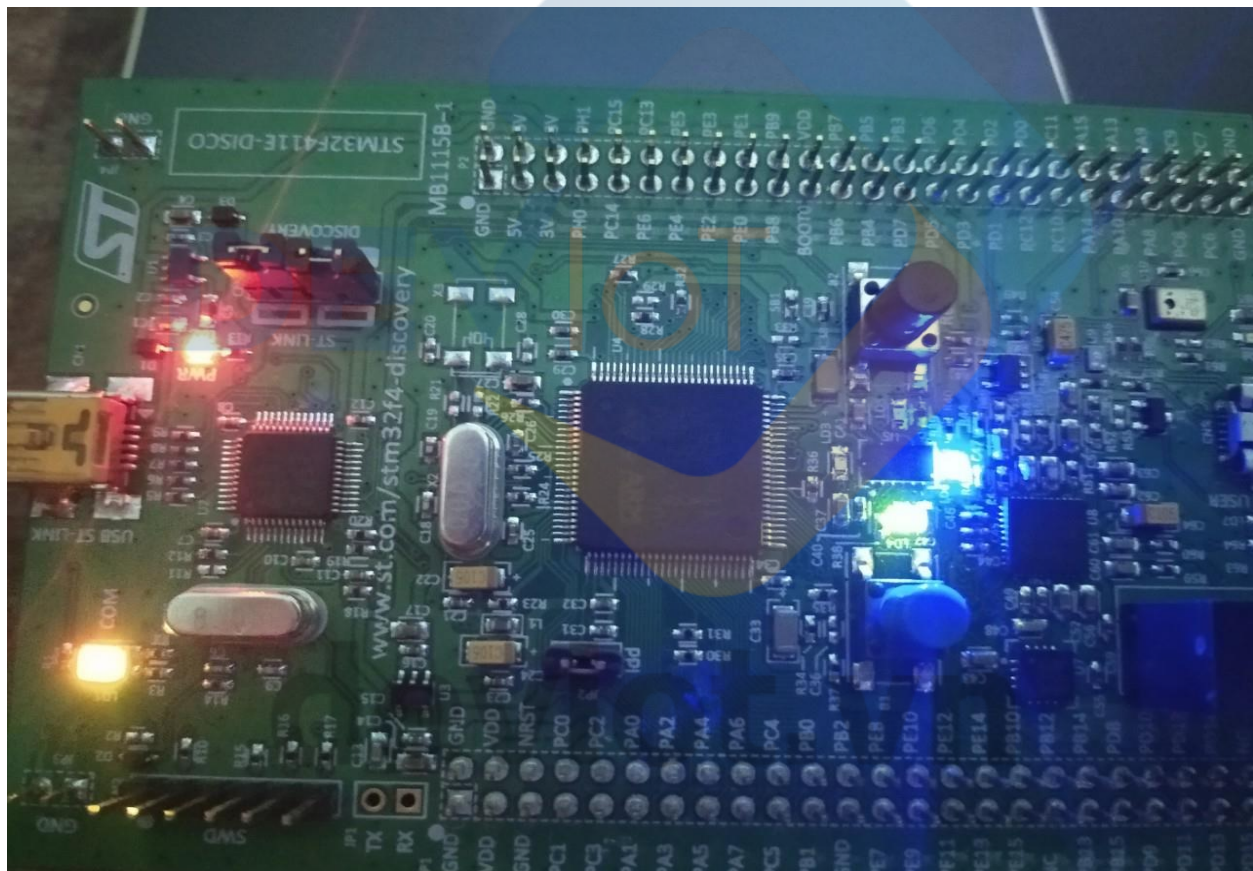
320 * implementations in user file.
321 * @retval tick value
322 */
323 weak uint32_t HAL_GetTick(void)
324 {
325     return uwTick;
326 }
327
328 /**
329 * @brief This function returns a tick priority.
330 * @retval tick priority
331 */
332 uint32_t HAL_GetTickPrio(void)
333 {
334     return uwTickPrio;
335 }
336
337 /**
338 * @brief Set new tick Freq.
339 * @retval Status
340 */

```

Watch 1

Name	Value	Type
✓ sTime.Hours	0	uchar
✓ sTime.Minutes	3	uchar
✓ sTime.Seconds	10	uchar
✓ sDate.Date	15	uchar
✓ sDate.Month	3	uchar
✓ sDate.WeekDay	7	uchar
✓ sDate.Year	20	uchar
✓ sTime.Hours	0	uchar

<Enter expression>



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

DEVIOT - CÙNG NHAU HỌC LẬP TRÌNH IOT

📌 Website: deviot.vn

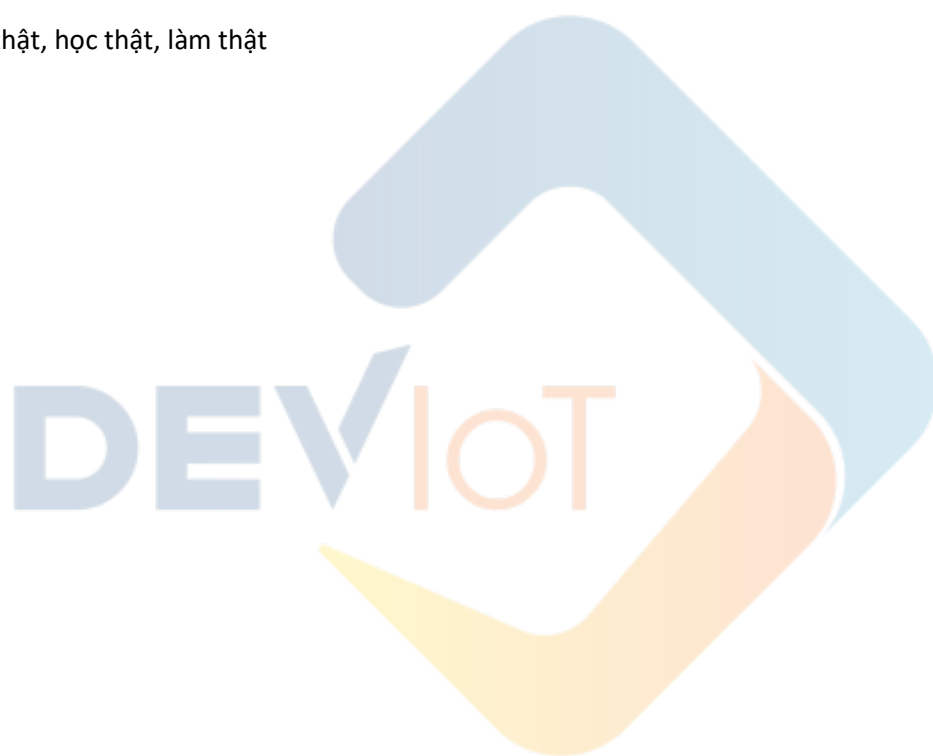
📌 Fanpage: Deviot - Thời sự kỹ thuật & IoT

📌 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



deviot.vn