TAPIT
BUILD THE FUTURE WITH US!
Learning - Research - Sharing Community

KHÓA HỌC
LẬP TRÌNH VI ĐIỀU KHIỂN

Giảng viên
NGUYỄN HUỲNH NHẬT THƯƠNG

LỊCH HỌC:
Tại Đà Nẵng: 19h30 - 22h00 thứ 2 và T6

ĐỊA ĐIỂM:
Online qua nền tảng Zoom

MODULE 11
. REAL TIME CLOCK (RTC)

REAL TIME CLOCK (RTC)

1. The RTC provides an automatic wakeup to manage all low-power modes
2. The real-time clock (RTC) is an independent BCD timer/counter. The RTC provides a time of-day clock/calendar with programmable alarm interrupts
3. The RTC includes also a periodic programmable wakeup flag with interrupt capability
4. Two 32-bit registers contain the seconds, minutes, hours (12- or 24-hour format), day (day of week), date (day of month), month, and year, expressed in binary coded decimal format (BCD)
5. Compensations for 28-, 29- (leap year), 30-, and 31-day months are performed automatically. Daylight saving time compensation can also be performed
6. As long as the supply voltage remains in the operating range, the RTC never stops, regardless of the device status (Run mode, low-power mode or under reset)
7. Two alarms: Alarm A, Alarm B
8. 16 x 32-bit backup registers

RTC LOW-POWER MODES

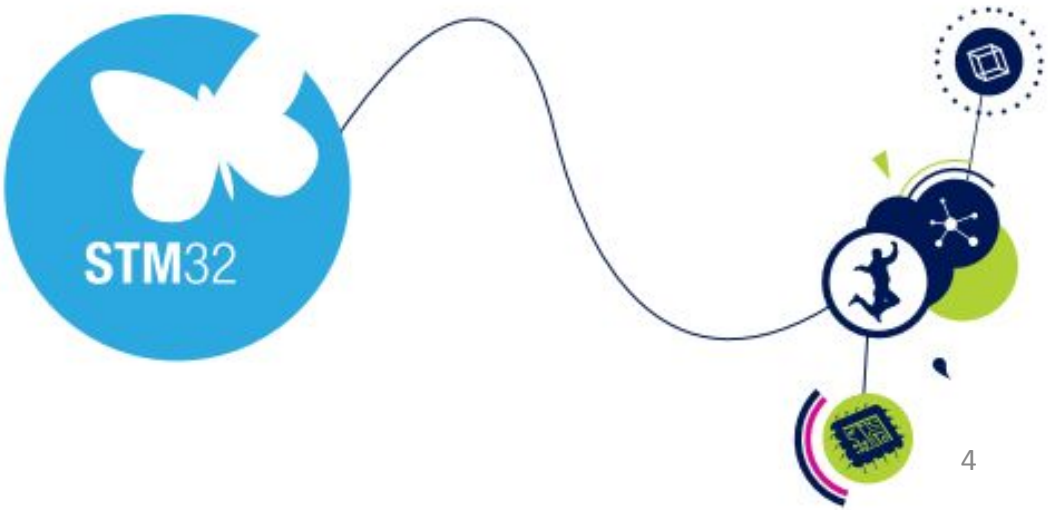
Mode	Description
Sleep	No effect RTC interrupts cause the device to exit the Sleep mode.
Stop	The RTC remains active when the RTC clock source is LSE or LSI. RTC alarm, RTC tamper event, RTC timestamp event, and RTC Wakeup cause the device to exit the Stop mode.
Standby	The RTC remains active when the RTC clock source is LSE or LSI. RTC alarm, RTC tamper event, RTC timestamp event, and RTC Wakeup cause the device to exit the Standby mode.

RTC REGISTERS

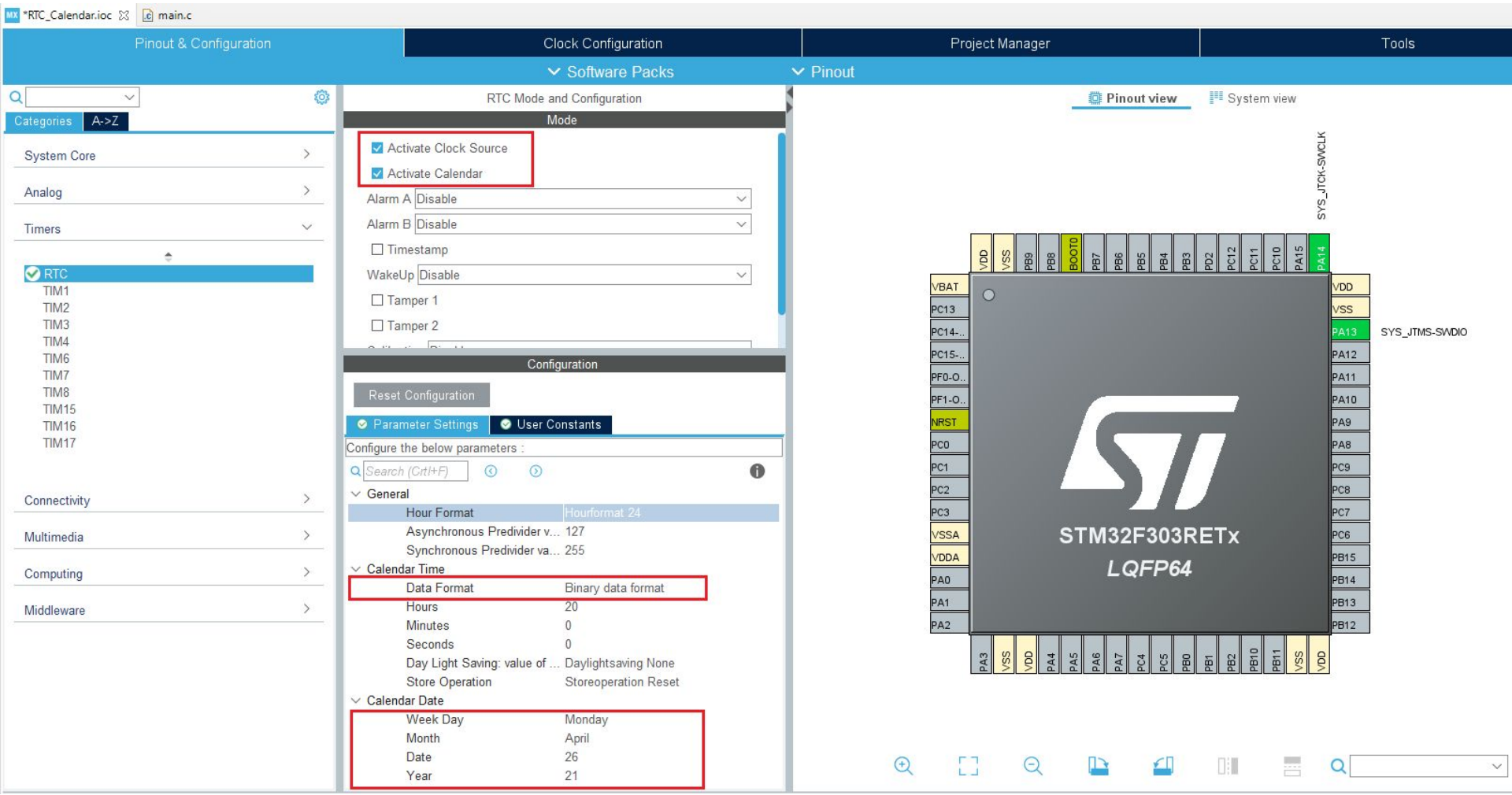
- RTC time register (RTC_TR) // giờ phút giây
- RTC date register (RTC_DR) // ngày tháng năm, thứ
- RTC control register (RTC_CR)
- RTC alarm A register (RTC_ALRMAR)
- RTC alarm B register (RTC_ALRMBR)
- RTC backup registers (RTC_BKPxR)

MODULE 8

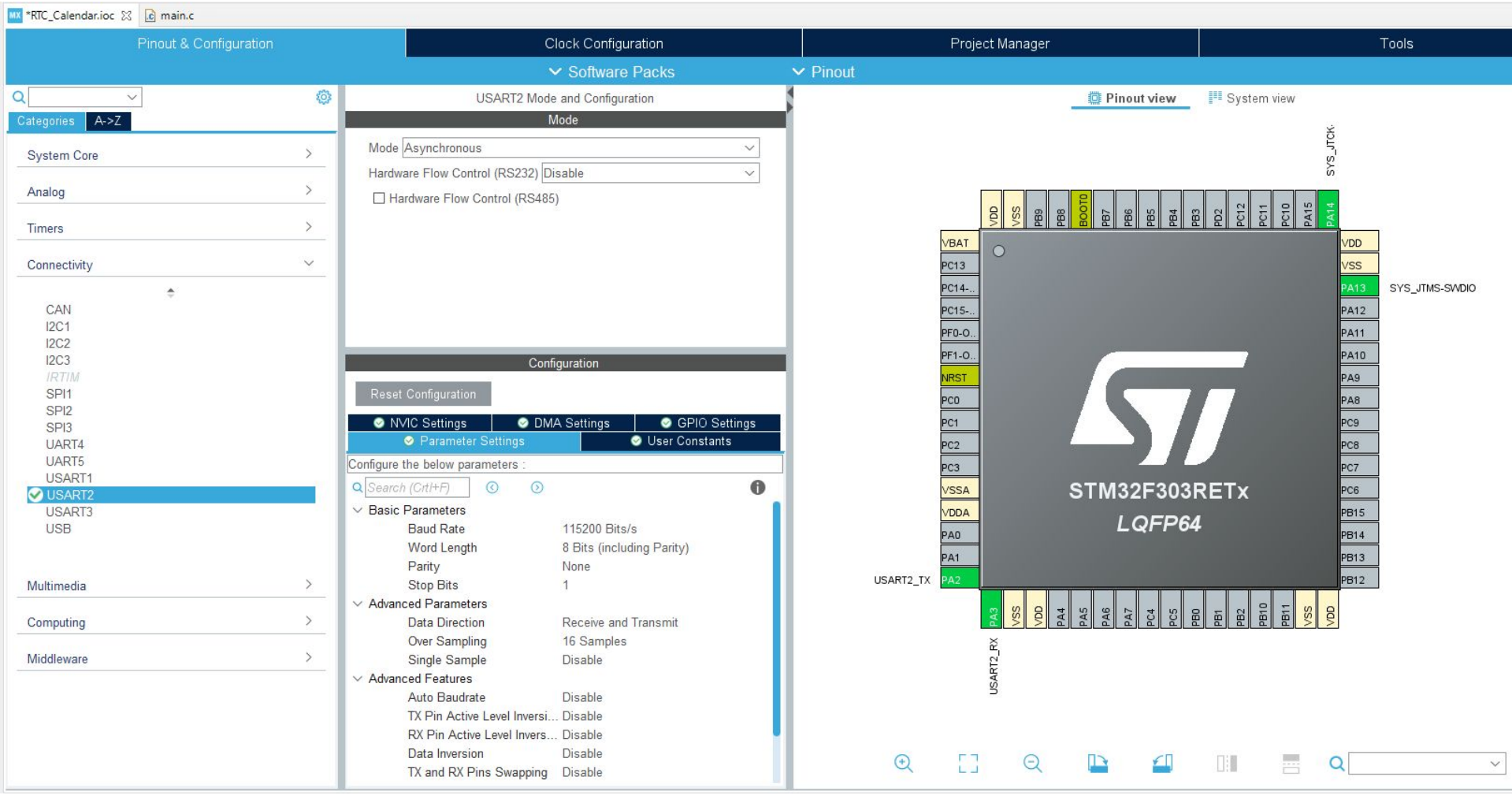
- THỰC HÀNH RTC



GIAO DIỆN CẤU HÌNH



GIAO DIỆN CẤU HÌNH



GIAO DIỆN C/C++

```
/* USER CODE BEGIN Includes */
#include "string.h"
#include "stdio.h"
```

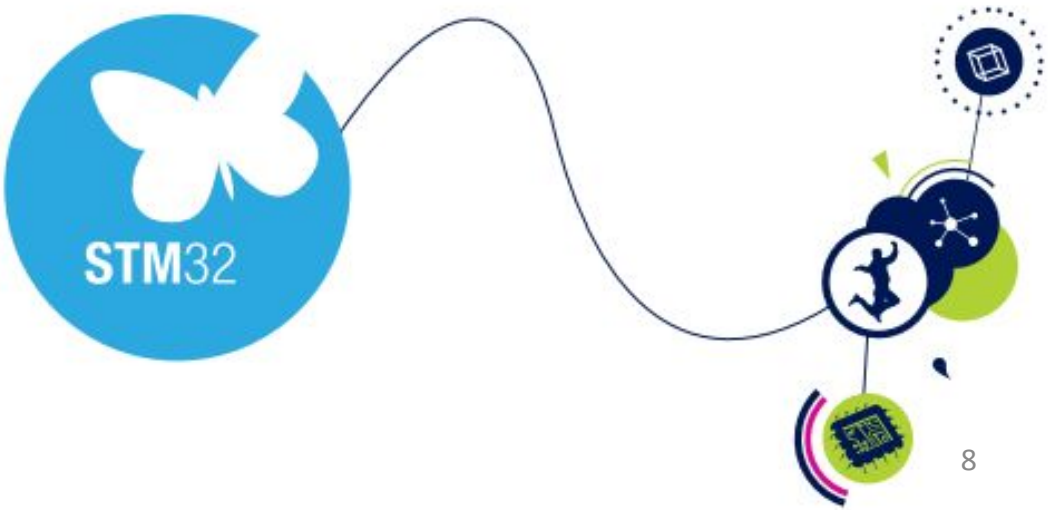
```
/* USER CODE BEGIN PV */
char showTime[50] = {0};
char showDate[50] = {0};
RTC_DateTypeDef datestructure;
RTC_TimeTypeDef timestructure;
```

```
/* USER CODE BEGIN WHILE */
 while (1)
 {
      HAL_RTC_GetTime(&hrtc, &timestructure, RTC_FORMAT_BIN);
      HAL_RTC_GetDate(&hrtc, &datestructure, RTC_FORMAT_BIN);
      sprintf(showTime, "Time: %02d:%02d:%02d\n", timestructure.Hours,
    timestructure.Minutes, timestructure.Seconds);
      sprintf(showDate, "Date: %02d-%02d-%4d\n", datestructure.Date,
    datestructure.Month, 2000 + datestructure.Year);
      HAL_UART_Transmit(&huart2, (uint8_t*)showTime, strlen(showTime), 100);
      HAL_UART_Transmit(&huart2, (uint8_t*)showDate, strlen(showDate), 100);
      HAL_Delay(1000);
       /* USER CODE END WHILE */
}
```

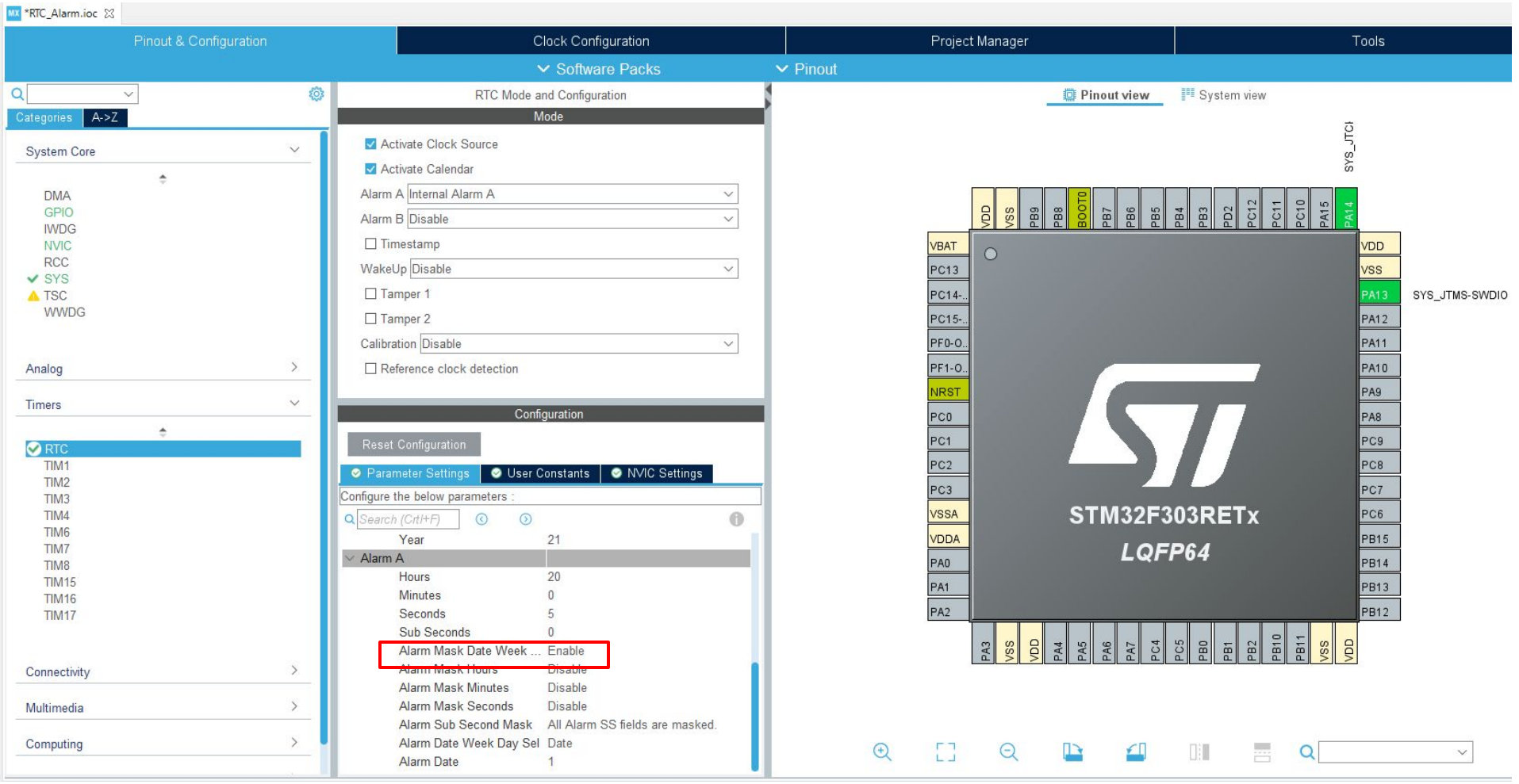
TAPiT

MODULE 8

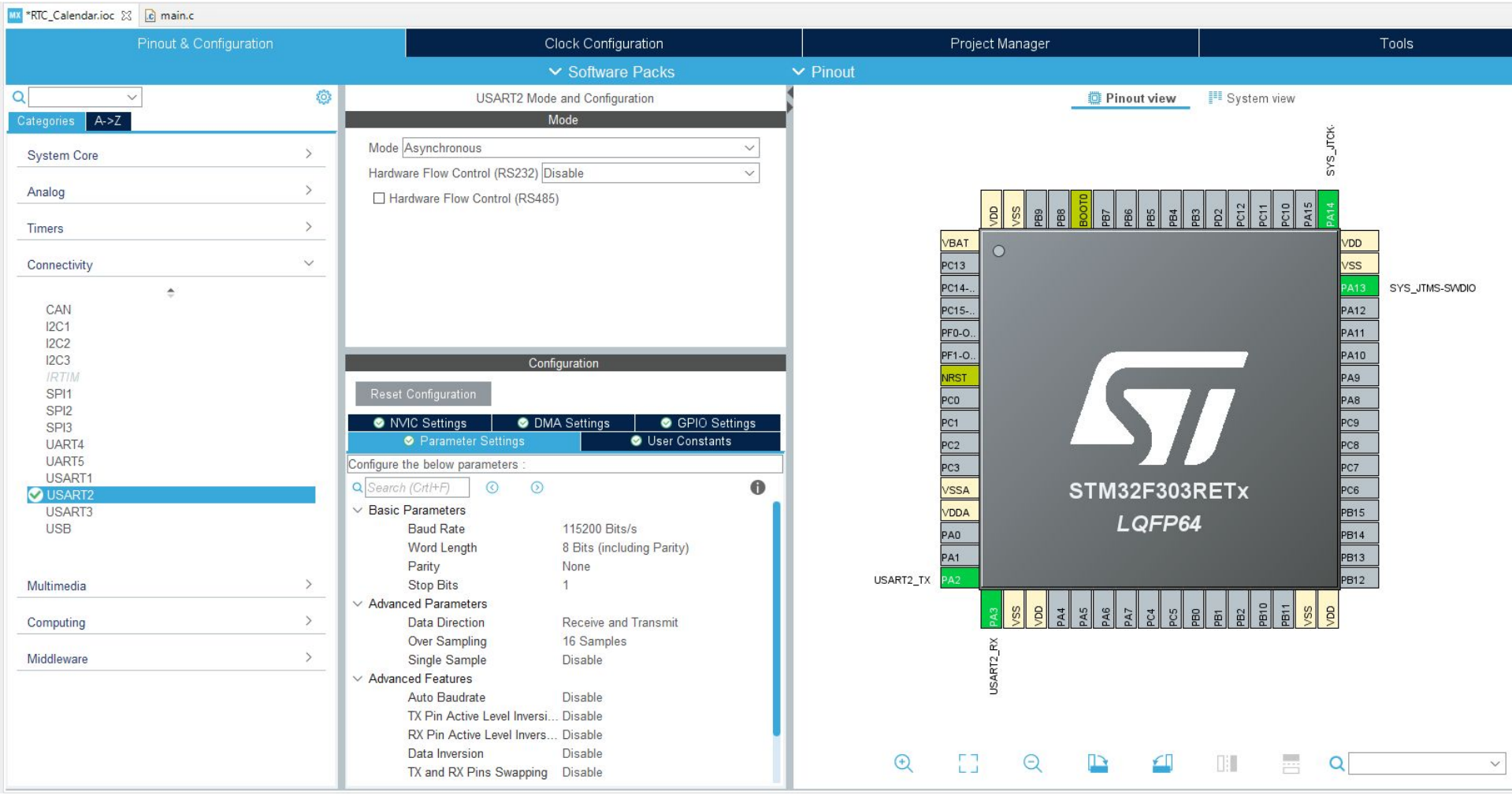
- THỰC HÀNH RTC ALARM



GIAO DIỆN CẤU HÌNH



GIAO DIỆN CẤU HÌNH



GIAO DIỆN C/C++

```
/* USER CODE BEGIN Includes */
#include "string.h"
#include "stdio.h"
```

```
/* USER CODE BEGIN PV */
char showTime[50] = {0};
char showDate[50] = {0};
RTC_DateTypeDef datestructure;
RTC_TimeTypeDef timestructure;
```

```
/* USER CODE BEGIN 0 */
void HAL_RTC_AlarmAEventCallback(RTC_HandleTypeDef *hrtc)
{
	HAL_RTC_GetTime(hrtc, &timestructure, RTC_FORMAT_BIN);
	HAL_RTC_GetDate(hrtc, &datestructure, RTC_FORMAT_BIN);
	sprintf(showTime, "Time: %02d:%02d:%02d\n", timestructure.Hours,
	timestructure.Minutes, timestructure.Seconds);
	sprintf(showDate, "Date: %02d-%02d-%4d\n", datestructure.Date,
	datestructure.Month, 2000 + datestructure.Year);
	HAL_UART_Transmit(&huart2, (uint8_t*)showTime, strlen(showTime), 100);
	HAL_UART_Transmit(&huart2, (uint8_t*)showDate, strlen(showDate), 100);
}
```

Định kỳ 1s (sử dụng delay trong while1). In thời gian lên UART Hercules với định dạng:
Time:hhmmss (có xuống hàng)
Date:ddmtmtyy (có xuống hàng)

Cài đặt thời gian Rtc: Trên hercules gửi xuống:

Timecfg:hhmmss (có kết thúc câu)
Datecfg:ddmtmtyy(có kết thúc câu)

backup register có giá trị ban đầu bằng 0.
Trong hàm mx rtc init (được gọi ở main, trước while 1.)

Đọc giá trị của 1 backup register nào đó và so sánh xem nó có khác với giá trị 1 hay không.

Nếu khác thì
Init RTC cài đặt thời gian vào ngoại vi RTC. để cho rtc bắt đầu chạy từ thời gian cấu hình.

Sau đó ghi giá trị backup register đó bằng 1.

Vậy thì lần khởi động sau đó, khi hàm mx rtc init được gọi thì , hàm read backup register sẽ trả về giá trị 1. Mà 1 thì == 1, mà nếu bằng 1 thì k init thời gian lại thêm 1 lần nữa.

Instructor

Eng. Nguyen Huynh Nhat Thuong