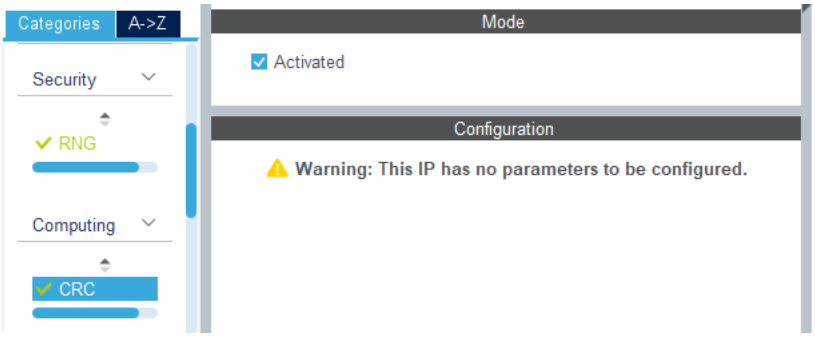
Systemy Wbudowane

Laboratorium 6:

Wykorzystanie CRC, RNG, RTC i praca samodzielna

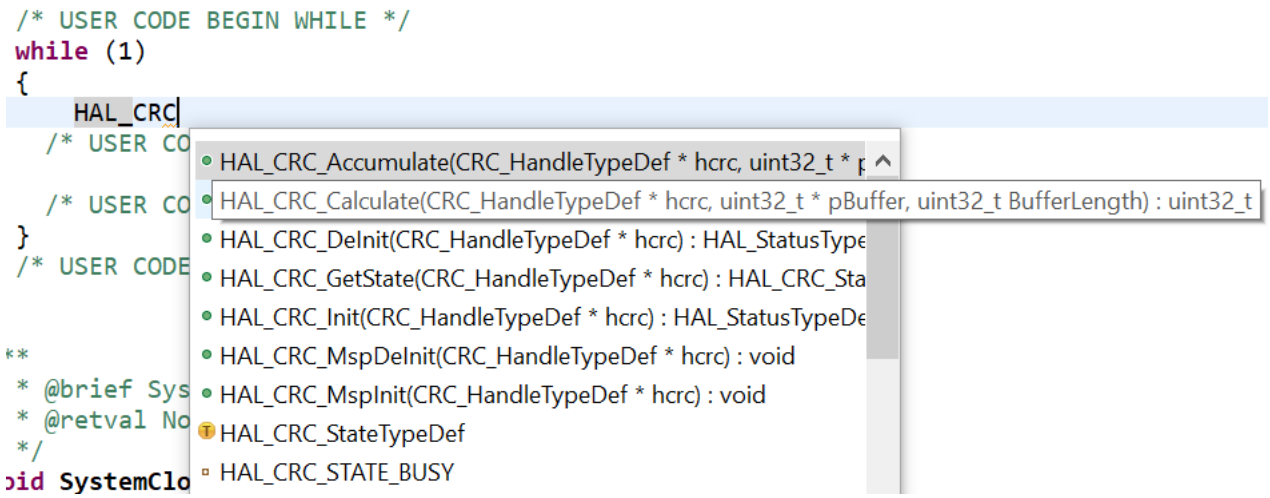
6.1 CRC i RNG

Płytki STM32F429ZI posiadają wbudowany sprzętowy generator sumy kontrolnej CRC oraz generator liczb losowych RNG. W celu ich włączenia wystarczy aktywować odpowiednie elementy w konfiguratorze:



Rysunek 6.1: Aktywacja CRC

Biblioteki HAL posiadają zestaw metod pozwalających na wykorzystanie CRC i RNG. Aby je odnaleźć samodzielnie wystarczy po wygenerowaniu kodu wpisać HAL_CRC i wcisnąć Ctrl+Spacja:



Rysunek 6.2: Podpowiedź z listą metod dotyczących sprzętowego generatora CRC

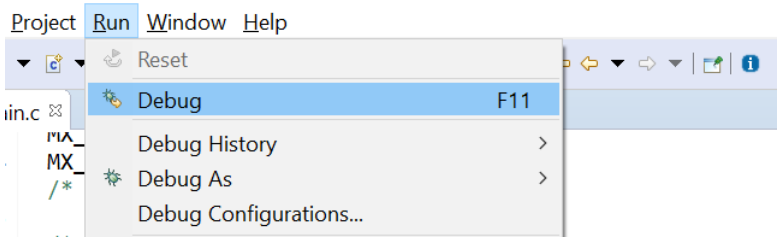
Na Rysunku 6.2 można zauważyć metodę *HAL_CRC_Calculate(...);* oraz parametry jakie przyjmuje i zwraca. Pierwszym z nich jest referencja do obiektu (a właściwie struktury) związanego z generatorem (&hcrc), drugim jest bufor z danymi dla których będzie obliczona suma CRC, trzecim jest długość bufora, zwracana przez metodę jest natomiast wartość nieujemna całkowita 32-bitowa (typu uint32_t). Można samemu wywnioskować że taka metoda będzie właściwa do liczenia sumy kontrolnej i wykorzystać ją następująco:

```
/* USER CODE BEGIN WHILE */
while (1)
{

  /* USER CODE END WHILE */

  /* USER CODE BEGIN 3 */
    uint32_t vals[3] = { 1749, 1567, 1438 };
    uint32_t crc = HAL_CRC_Calculate(&hcrc, (uint32_t*)vals, 3);
    uint32_t test = crc;
}
/* USER CODE END 3 *
}
```

Expression	Type	Value
crc	uint32_t	1711426083

Rysunek 6.3: Wykorzystanie sprzętowego generatora sumy kontrolnej CRC

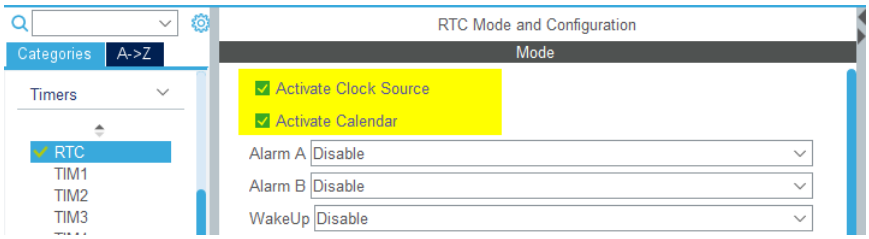
Aby otrzymać podgląd wartości zmiennej tak jak na rysunku powyżej, należy uruchomić program w trybie debugowania (Run → Debug) umieszczając w odpowiednim miejscu pułapkę (opis debugowania - Laboratorium 1).



Rysunek 6.4: Włączenie trybu debugowania

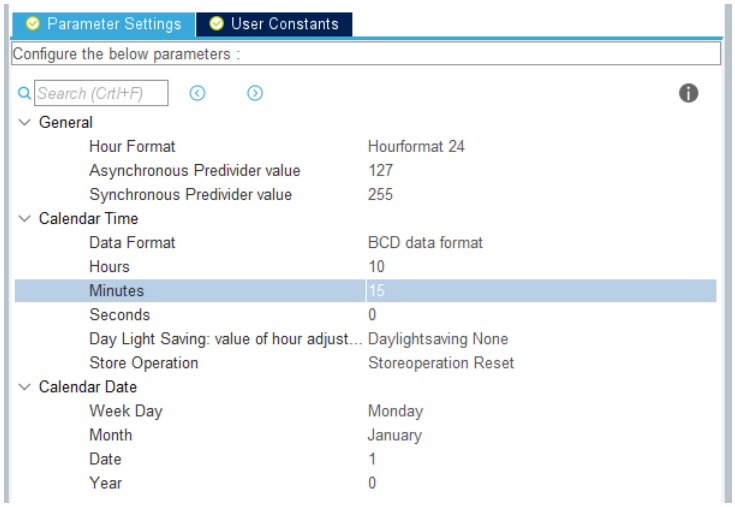
6.2 RTC

RTC (*ang. Real Time Clock*) czyli zegar czasu rzeczywistego jest podzespołem zliczającym czas w tle. Zasada działania RTC jest taka, aby umożliwiał on liczenie czasu, nawet gdy główne źródło zasilania płytki jest odłączone (źródłem zasilania staje się wówczas bateria, podobnie jak ma to miejsce np. w płytach głównych komputerów). Niestety płytki rozwojowe STM32F429ZI nie są domyślnie wyposażone w taką baterię (istnieje jednak możliwość jej podłączenia), a więc zliczany czas zegara RTC nie zostaje zapamiętany po odłączeniu zasilania. Podstawowa konfiguracja RTC wymaga włączenia następujących opcji:



Rysunek 6.5: Aktywacja RTC

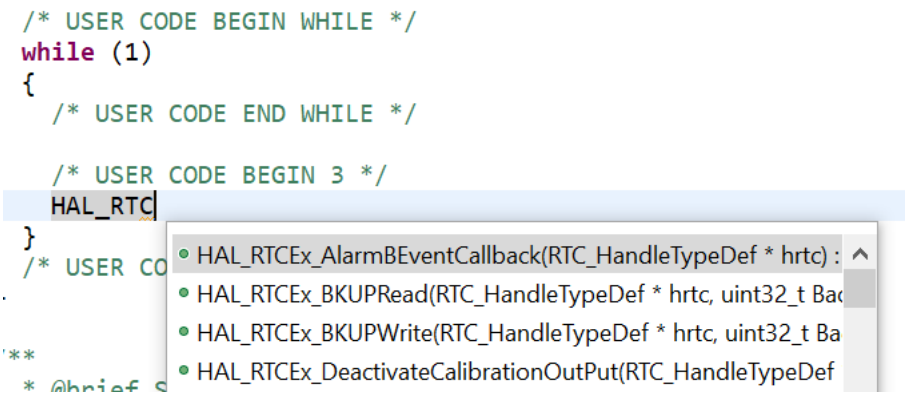
RTC można także skonfigurować ustawiając w nim alarmy (Rysunek 6.5), format daty oraz godzinę początkową (ustawianą w przypadku utraty zasilania lub wgrania programu):



Rysunek 6.6: Konfiguracja RTC

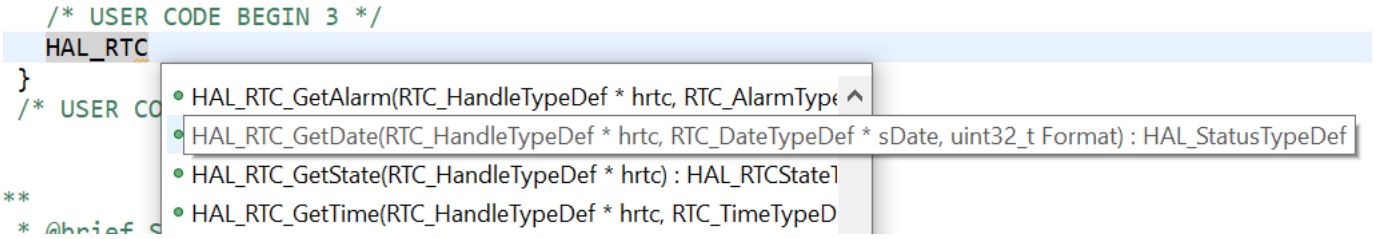
Po wygenerowaniu kodu ustawiona konfiguracja RTC znajduje się w metodzie *MX_RTC_Init(void);*. Znajduje się tam także kod ustawiający godzinę i datę (można go tymczasowo zakomentować aby sprawdzić czy zegar działa niezależnie od resetu programu) i sekcja kodu przeznaczona do sprawdzania kopii zapasowej czasu zegara.

Analogicznie do przykładu z CRC, wciśnięcie Ctrl+Spacja rozwinie listę metod biblioteki HAL dla danego elementu:



Rysunek 6.7: Podpowiedź z listą metod dotyczących RTC

Dopisek Ex (*ang. Extended*) oznacza listę metod dodatkowych, właściwe metody znajdują się w tym przypadku na dole listy:



Rysunek 6.8: Podpowiedź z listą metod dotyczących RTC - metody bazowe

Na Rysunku 6.8 widać metodę GetTime, powinna ona pozwolić zwrócić czas z RTC, posiada ona jednak szereg parametrów wejściowych w postaci struktur. Aby dowiedzieć się więcej o metodzie należy przytrzymać przycisk Ctrl i kliknąć na nazwę metody:

```
HAL_StatusTypeDef HAL_RTC_GetTime(RTC_HandleTypeDef *hrtc, RTC_TimeTypeDef *sTime, uint32_t Format)
{
  uint32_t tmpreg = 0U;

  /* Check the parameters */
  assert_param(IS_RTC_FORMAT(Format));

  /* Get subseconds structure field from the corresponding register */
  sTime->SubSeconds = (uint32_t)(hrtc->Instance->SSR);

  /* Get SecondFraction structure field from the corresponding register field*/
  sTime->SecondFraction = (uint32_t)(hrtc->Instance->PRER & RTC_PRER_PREDIV_S);
```

Rysunek 6.9: Metoda *HAL_RTC_GetTime(...);*

Analogicznie można postąpić z typami struktur i np. funkcją IS_RTC_FORMAT:

```
800 #define IS_RTC_FORMAT(FORMAT) (((FORMAT) == RTC_FORMAT_BIN) || ((FORMAT) == RTC_FORMAT_BCD))
801 #define IS_RTC_YEAR(YEAR)             ((YEAR) <= 99U)
802 #define IS_RTC_MONTH(MONTH)           (((MONTH) >= 1U) && ((MONTH) <= 12U))
803 #define IS_RTC_DATE(DATE)             (((DATE) >= 1U) && ((DATE) <= 31U))
804 #define IS_RTC_WEEKDAY(WEEKDAY) (((WEEKDAY) == RTC_WEEKDAY_MONDAY)    || \
```

Rysunek 6.10: Metoda *IS_RTC_FORMAT(...);*

Podpowiedzi są dostępne także przy najechaniu kursorem na typy danych:

```
HAL_StatusTypeDef HAL_RTC_GetTime(RTC_HandleTypeDef *hrtc, RTC_TimeTypeDef *sTime
/**
  * @brief  HAL Status structures definition
  */
typedef enum
{
  HAL_OK       = 0x00U,
  HAL_ERROR    = 0x01U,
  HAL_BUSY     = 0x02U,
  HAL_TIMEOUT  = 0x03U
} HAL_StatusTypeDef;
Press 'F2' for focus
                                                  orresponding register */
                                                  e->SSR);
                                                  he corresponding register field*/
                                                  tance->PRER & RTC_PRER_PREDIV_S);
  /* Get the TR register */
  tmpreg = (uint32_t)(hrtc->Instance->TR & RTC_TR_RESERVED_MASK);
```

Rysunek 6.11: Podpowiedzi dostępne po najechaniu kursorem na strukturę

Analizując kod powyższej metody i jej składowych można już napisać poprawny kod umożliwiający jej wywołanie:

```
HAL_StatusTypeDef result;
RTC_TimeTypeDef sTime;
result = HAL_RTC_GetTime(&hrtc, &sTime, RTC_FORMAT_BIN);
if (result == HAL_OK) {
    uint8_t sec = sTime.Seconds;
    uint8_t min = sTime.Minutes;
    uint8_t hou = sTime.Hours;
}
```

6.3 Ćwiczenie 1

Celem ćwiczenia jest wykorzystanie generatora liczb losowych RNG. W celu realizacji ćwiczenia należy:

- Utworzyć nowy projekt i skonfigurować generator RNG (wystarczy sama jego aktywacja)
- Sprawdzić dostępne metody z biblioteki HAL powiązane z generatorem RNG. Odnaleźć metodę generującą liczbę losową
- Wygenerować liczbę losową i wyświetlić jej podgląd w trybie debugowania

6.4 Ćwiczenie 2

Celem ćwiczenia jest wykorzystanie przetwornika ADC. W celu realizacji ćwiczenia należy:

- Poprosić prowadzącego o potencjometr i głośnik
- Przy pomocy konfiguratora graficznego skonfigurować przetwornik ADC2 wybierając dowolny kanał (od IN0 do IN15)
- Sprwadzić jaka linia sygnału jest wykorzystywana przez dany kanał
- Podpiąć potencjometr do płytki, pamiętając aby sygnał OUT potencjometru podłączyć pod znalezioną linię sygnału
- Wygenerować kod programu, wyszukać metodę HAL_ADC... odczytującą wartość z przetwornika ADC i przypisać wartość do dowolnej zmiennej (należy pamiętać że przed każdym odczytaniem wartości z przetwornika należy uruchomić rozpoczęcie konwersje metodą *HAL_ADC_Start(...);* i po jej wywołaniu odczekać około 15ms: *HAL_Delay(15);*)
- Wykorzystać odczytaną wartość i na jej podstawie ustawić ARR sygnału PWM wybranego licznika (TIMx->ARR, gdzie x oznacza numer licznika, pamiętać ustawić wypełnienie puslu TIMx->CRR1 na połowę TIMx->ARR)
- Licznik skonfigurować ustawiając PSC=15, Counter Period=1000 oraz Pulse=10 (należy pamiętać o uruchomieniu licznika metodą *HAL_TIM_PWM_Start(...);*)
- Sprawdzić czy operowanie potencjometrem rzeczywiście wpływa na generowany dźwięk, w razie problemów podejrzeć odczytywane wartości z przetwornika ADC w trybie debugowania

Dla tego laboratorium należy przygotować sprawozdanie zawierające odpowiednie kody źródłowe oraz zdjęcia przedstawiające działanie programów (dla wszystkich ćwiczeń)