



# CREDO School

HANDS ON LAB

KATARZYNA SMELCERZ

# CREDO SCHOOL - HARDWARE

Sesja została przygotowana dla osób początkujących w programowaniu sprzętowym. Poświęcona jest mikrokontrolerowi **STM32L152 RB**. Oprogramowanie będzie pisane w środowisku IAR **Embedded Workbench**. Wszystkie zadania będą wykonywane na podstawie materiałów zawartych w bieżącym dokumencie.

## OPROGRAMOWANIE I DOKUMENTACJA

W mailu, który był wysłany znajduje się lista oprogramowania koniecznego do zainstalowania.

Dokumentacja:

- Opis zestawu starter-kit, schemat: [http://www.st.com/st-web-ui/static/active/en/resource/technical/document/user\\_manual/DM00027954.pdf](http://www.st.com/st-web-ui/static/active/en/resource/technical/document/user_manual/DM00027954.pdf)
- Kompilator dla rdzenia ARM: <https://www.iar.com/iar-embedded-workbench/arm/>
- Opis rdzenia ARM STM32L152 <http://www.st.com/st-web-ui/static/active/en/resource/technical/document/datasheet/CD00277537.pdf>
- Dokumentacja do środowiska STM32CubeMX: [http://www.st.com/st-web-ui/static/active/en/resource/technical/document/data\\_brief/DM00103564.pdf](http://www.st.com/st-web-ui/static/active/en/resource/technical/document/data_brief/DM00103564.pdf)

# 1. PODSTAWOWE WIADOMOŚCI DOTYCZĄCE MIKROKONTROLERA ARM STM32L152 RB

Mikrokontroler ARM STM32L152 RB, znajdujący się w zestawie starter-kit, wykorzystywanym na laboratorium, należy do rodziny L. Znacznik L w nazwie, oznacza rodzinę Low Power, czyli produkty przeznaczone do projektów, w których niskie zużycie energii (np. pochodzącej z baterii) jest priorytetem. Często jest jednak tak, że w zamian za niskie zużycie energii otrzymujemy mniejszą moc obliczeniową, tak też i jest tym razem. Jeśli prześledzimy na stronie ST rodzinę rdzeni, pod względem mocy obliczeniowej, łatwo jest zauważyć, że istnieją rdzenie, które mają szybsze zegary i większe pamięci, np. rodzina F.

Procesory niskomocowe mają często dodatkowe wyposażenie, niespotykane w innych rodzinach. Tak też jest w tym wypadku, procesor STM32L152 ma wbudowany sterownik ekranu LCD. Niski pobór mocy oznacza często mniejszą moc obliczeniową, ale wciąż wielokrotnie większą niż możliwości procesorów 8 i 16 bitowych.

Poniżej przedstawiono porównanie rdzeni serii L1.



Rys 1.1 Porównanie rdzeni serii L1, źródło: ST.com

Ogólna charakterystyka rdzenia STM32L152 RB:

- Niskie zużycie mocy, 1.65 V – 3.6 V zasilanie
- 0.3  $\mu$ A Standby mode
- 0.9  $\mu$ A Standby mode +ko RTC
- 0.57  $\mu$ A Stop mode
- 1.2  $\mu$ A Stop mode + RTC
- 9  $\mu$ A Low-power run mode
- 214  $\mu$ A/MHz Run mode
- 10 nA ultra-low I/O
- < 8  $\mu$ s czas wybudzenia
- Rdzeń: ARM®Cortex®-M3 32-bit CPU
- Od 32 kHz do 32 MHz, częstotliwość zegara

## 2. KRÓTKI OPIS ZESTAWU STARTER-KIT - STM32L-DISCOVERY



1.2 Zestaw ewaluacyjny STM32L-Discovery

Zestaw ewaluacyjny STM32L-Discovery jest wyposażony w rdzeń STM32L152 RB, który został opisany w poprzednim rozdziale. Nagrywanie programu odbywa się za pomocą ST-Link, który jest wbudowany w płytke.

Można również go podłączyć jako urządzenie zewnętrzne, za pomocą 4 pinów. Zasilanie jest dostarczone przez kabel USB.

Płytką, na której będą wykonywane zadania laboratoryjne jest między innymi wyposażona w:


- Wyświetlacz LCD
- 4 diody
- 2 przyciski
- Czujnik dotykowy + 4 dotykowe klawisze
- Pomiar prądu zasilania

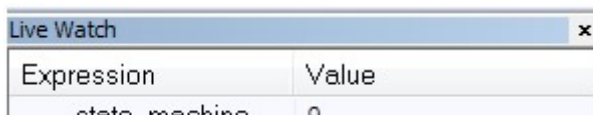
Więcej szczegółów znajduje się w dokumentacji dostarczonej przez producenta.

### 3. KONFIGURACJA I PODSTAWOWE FUNKCJE ŚRODOWISKA IAR EMBEDDED WORKBENCH

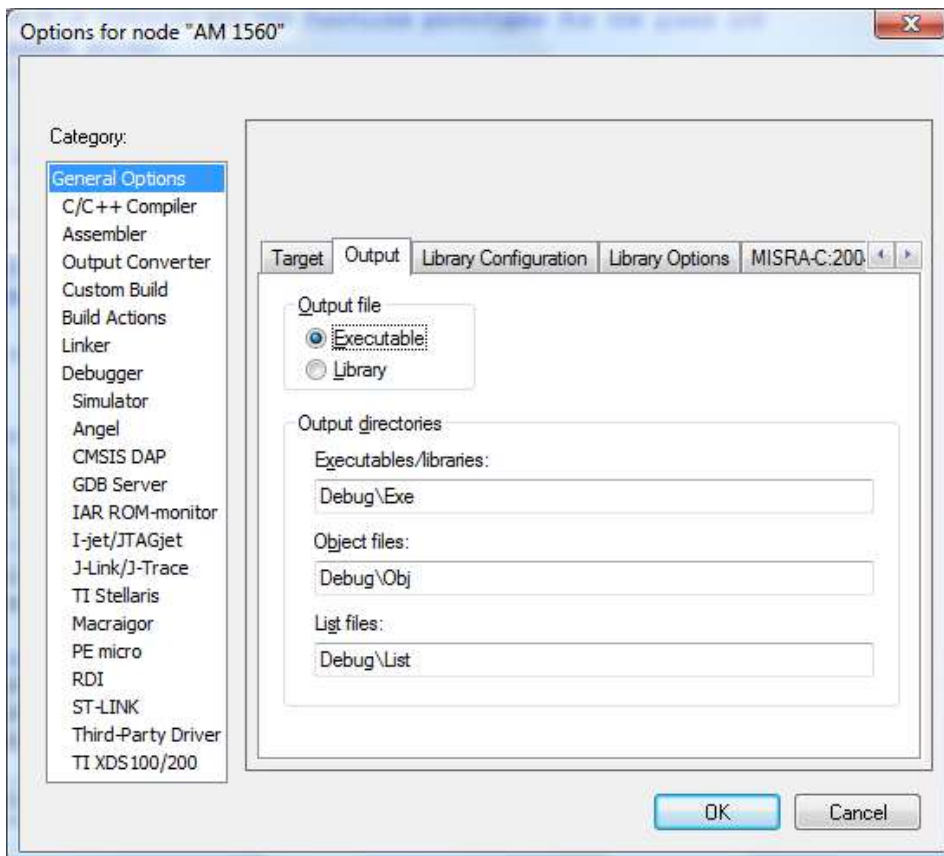
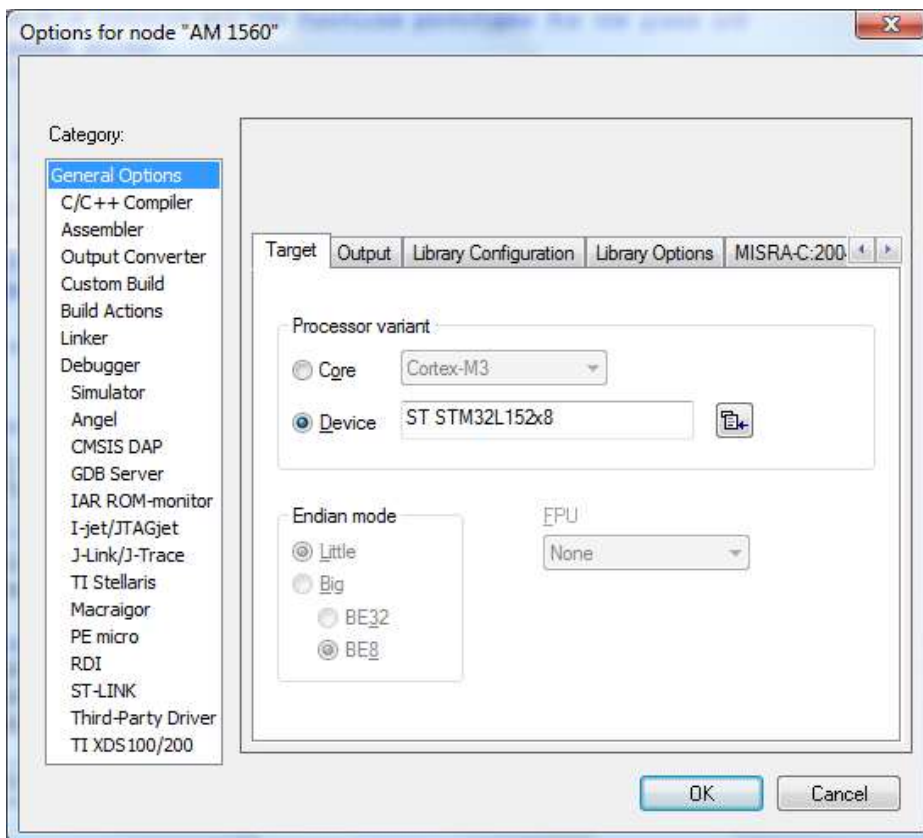
Kompilator IAR wymaga konfiguracji do pracy pod konkretnym rdzeniem. Wszystkie ustawienia są przedstawione na poniższych print screenach. Nagranie programu odbywa przyciskiem „Download and

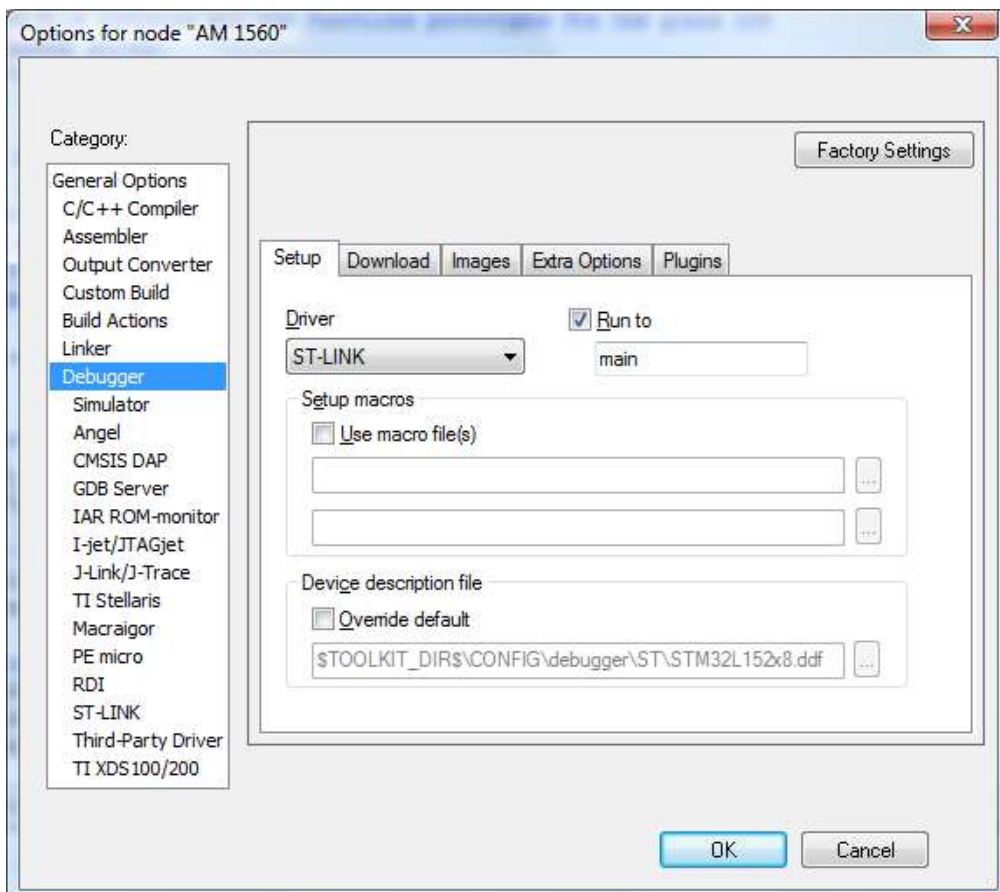
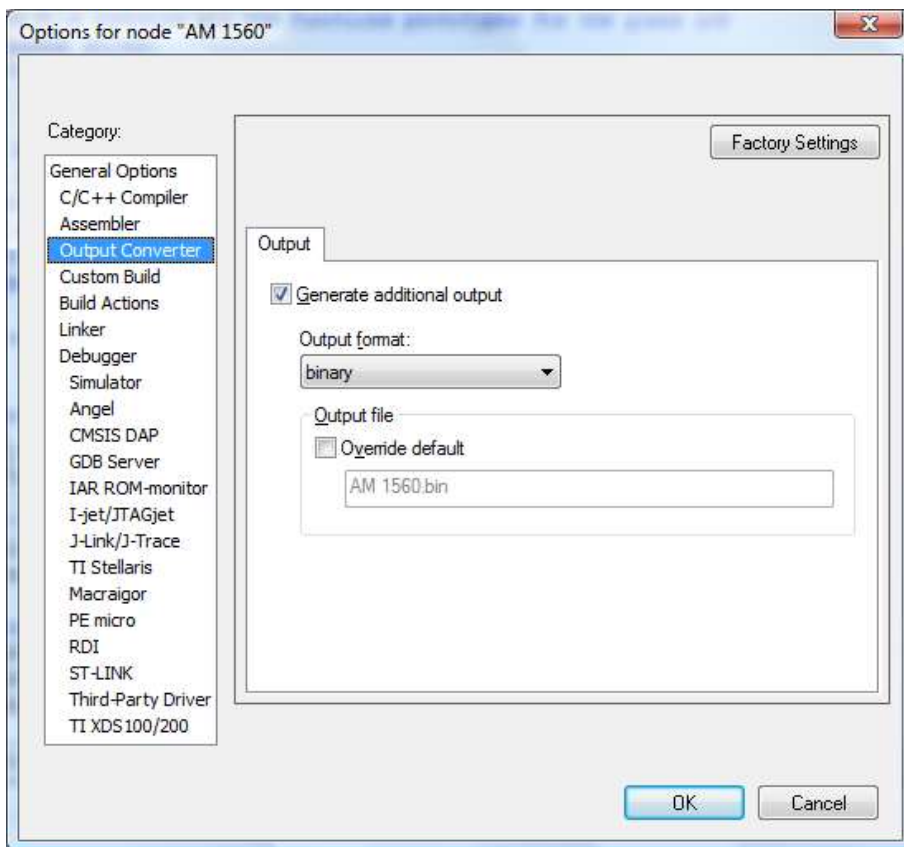
Debug” , uruchomienie „go” , pozostałe funkcje debuggера „step over”, „step into” itd również są

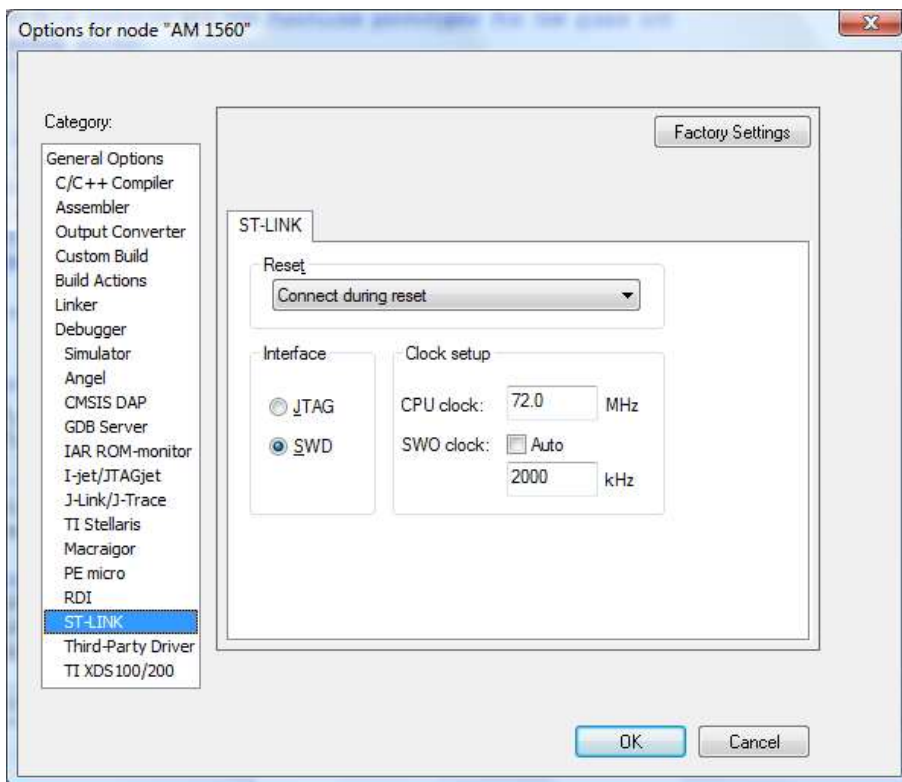
dostępne . Podgląd zmiennych odbywa się w trybie debuggowania w oknach Watch i Locals – należy je wywołać w zakładce „View”, na górnej belce kompilatora.



Expression	Value
state_machine	0







## 4. PODSTAWOWE WIADOMOŚCI DOTYCZĄCE ŚRODOWISKA STM32CUBEMX

Środowisko Cube, jest środowiskiem graficznym, służącym do generowania kodu w języku C. Wygenerowany kod, jest kompatybilny z takimi kompilatorami, jak Keil, IAR oraz GCC. Środowisko to, bardzo ułatwia konfigurację peryferii, portów IO, czy też zegarów.

Ręczna konfiguracja jest mozolna i łatwo w niej popełnić błędy.

## 5. PORTY WEJŚCIA/WYJŚCIA

Porty w omawianym na laboratorium mikrokontrolerze zazwyczaj występują w postaci 16 bitowej. Tak jak we wszystkich architekturach, tak i tu, służą do komunikacji ze światem zewnętrznym. Port może być skonfigurowany w różnych trybach, takich jak te przedstawione poniżej.

```
typedef enum {
    GPIO_Mode_IN   = 0x00, /*!< GPIO Input Mode */
    GPIO_Mode_OUT  = 0x01, /*!< GPIO Output Mode */
    GPIO_Mode_AF   = 0x02, /*!< GPIO Alternate function Mode */
    GPIO_Mode_AN   = 0x03 /*!< GPIO Analog Mode */
} GPIOMode_TypeDef;
```



W mikrokontrolerach ARM port konfiguruje się jako strukturę. Opis tej struktury jest zawarty w bibliotece `stm32l1xx_gpio.h`, dołączonej do projektu.

```
typedef struct
{
    uint32_t GPIO_Pin;          /*!< Specifies the GPIO pins to be configured.
                                This parameter can be any value of @ref GPIO_pins_define */

    GPIOMode_TypeDef GPIO_Mode; /*!< Specifies the operating mode for the
                                selected pins. This parameter can be a value
                                of @ref GPIOMode_TypeDef */

    GPIOSpeed_TypeDef GPIO_Speed; /*!< Specifies the speed for the selected pins.
                                This parameter can be a value of @ref
                                GPIOSpeed_TypeDef */

    GPIOType_TypeDef GPIO_OType; /*!< Specifies the operating output type for the
                                selected pins. This parameter can be a
                                value of @ref GPIOType_TypeDef */

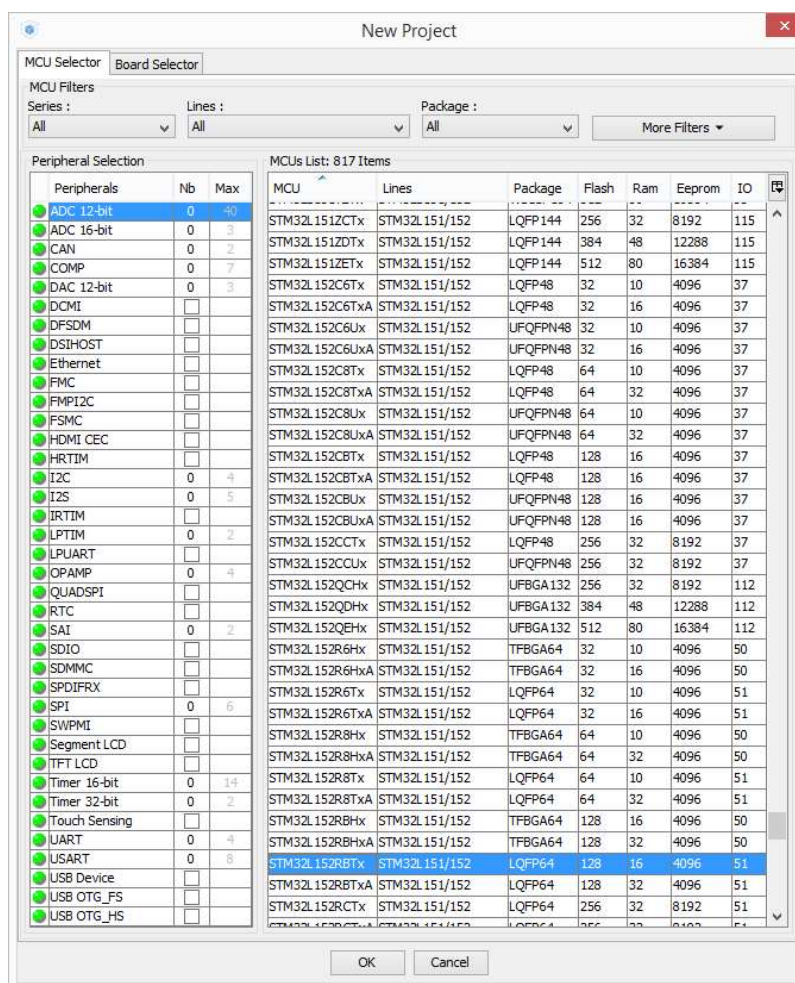
    GPIOPuPd_TypeDef GPIO_PuPd; /*!< Specifies the operating Pull-up/Pull down
                                for the selected pins. This parameter can
                                be a value of @ref GPIOPuPd_TypeDef */
} GPIO_InitTypeDef;
```

**UWAGA!** Jak ustawić konkretny Pin dla danego portu w stan wysoki lub niski, jest opisane w pliku `stm32l1xx_hal_gpio.c`, dołączanym automatycznie przez Cube na etapie generowania kodu w języku C.

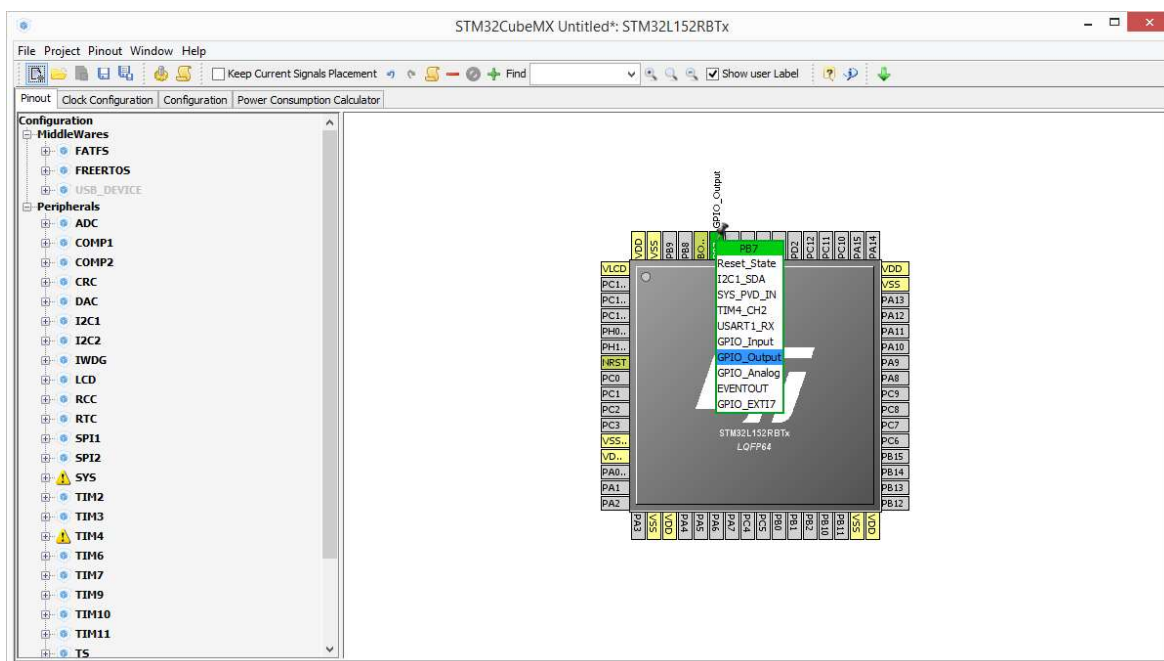
## 6.KONFIGURACJA PORTÓW W ŚRODOWISKU CUBE

Jak widać na Listingach zawartych w poprzednim rozdziale, konfiguracja portów jest dosyć żmudnym i czasochłonnym procesem. Środowisko Cube dostarcza programiście narzędzia, które pomagają znacznie zaoszczędzić czas. Poniżej przedstawiono przykładową konfigurację portu oraz konfigurację Cube dla naszego rdzenia.

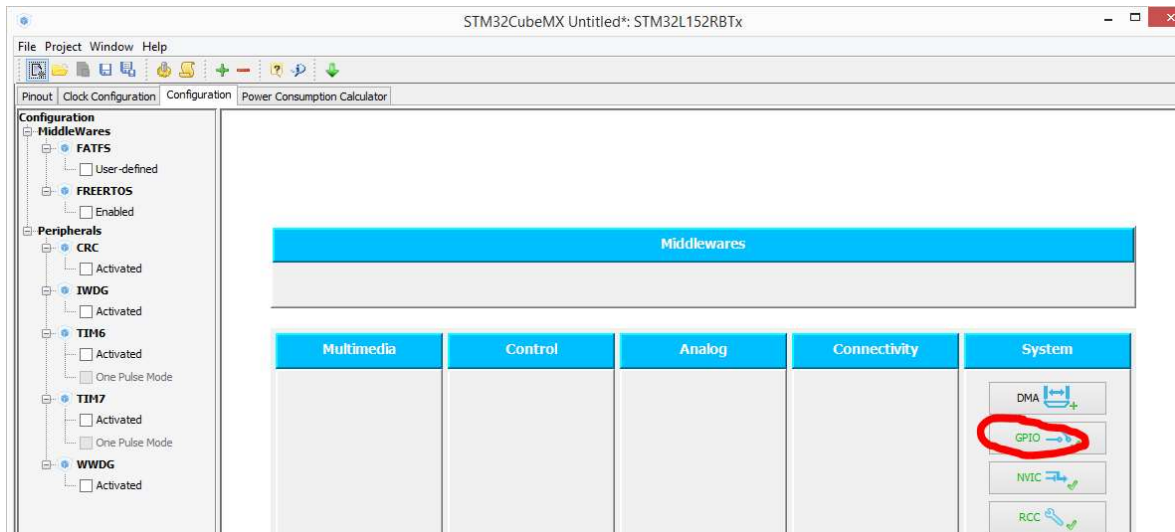
Kliknij „new project”, wybierz odpowiedni rdzeń, zgodnie z nr układu na płytce:



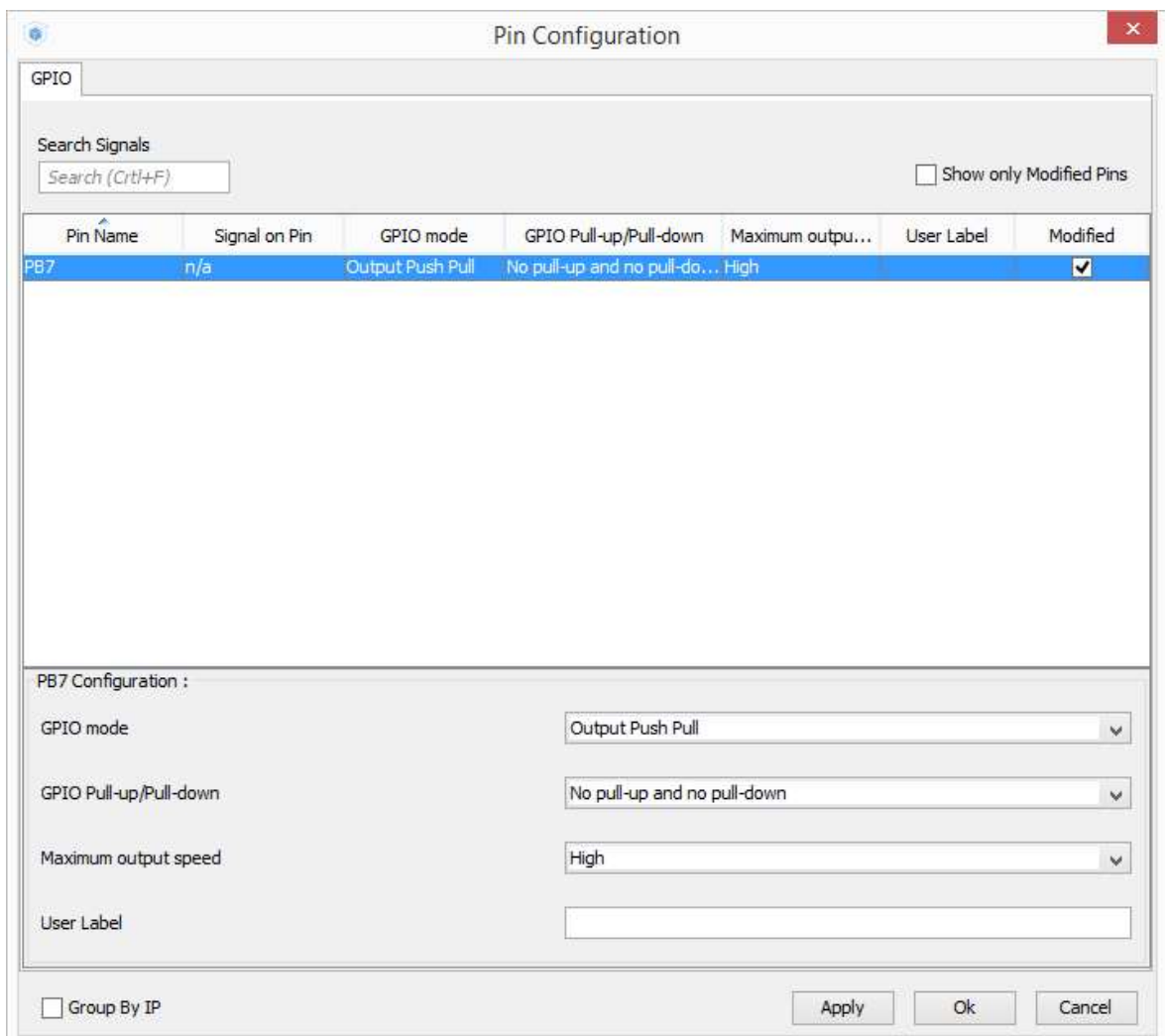
## 1.1 Wybranie rdzenia



## 1.2 Konfiguracja pinów - ustawienie pinu jako wyjście

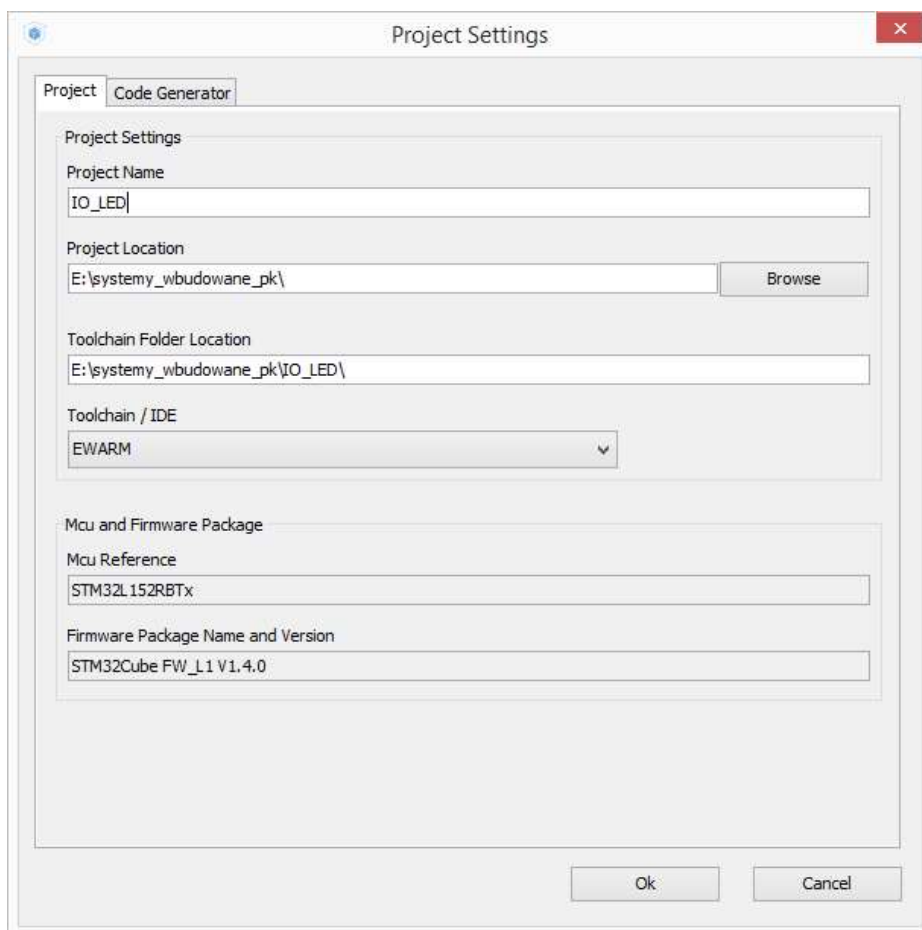


## 1.3 Dalsza konfiguracja pinów



#### 1.4 Tryb, prędkość i pozostałe opcje pinu

Generowanie kodu w języku C:



Project Settings

Project Code Generator

Project Settings

Project Name  
IO\_LED

Project Location  
E:\systemy\_wbudowane\_pk\ Browse

Toolchain Folder Location  
E:\systemy\_wbudowane\_pk\IO\_LED\

Toolchain / IDE  
EWARM

Mcu and Firmware Package

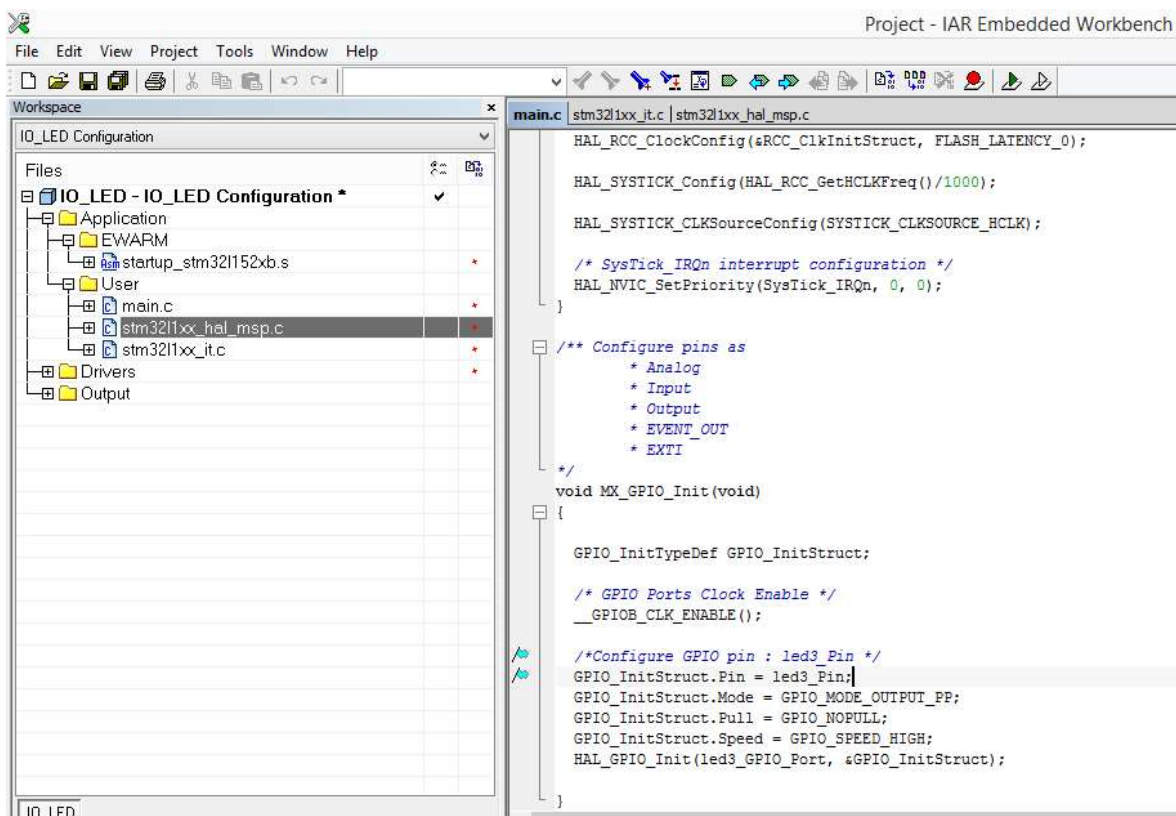
Mcu Reference  
STM32L152RBTx

Firmware Package Name and Version  
STM32Cube FW\_L1 V1.4.0

Ok Cancel

#### 1.5 Zrzut po wygenerowaniu kodu w języku C - nazwa projektu dla IAR

Po naciśnięciu przycisku OK, zostaje uruchomione środowisko IAR. Znajdują się w nim wygenerowane pliki w języku C, zgodne z konfiguracją ustawioną w środowisku CUBE.



1.6 Zrzut z IAR - kod programu

## 7.TIMERY

Rdzeń ARM STM32L152 RB jest wyposażony w 10 timerów. Sześć 16-bitowych, do 4 kanałów IC/OC/PWM oraz dwa 16-bitowe podstawowe timery i 2 do obsługi watchdoga.

Poniżej znajduje się proste zestawienie wszystkich Timerów wraz z ich najważniejszymi funkcjami.

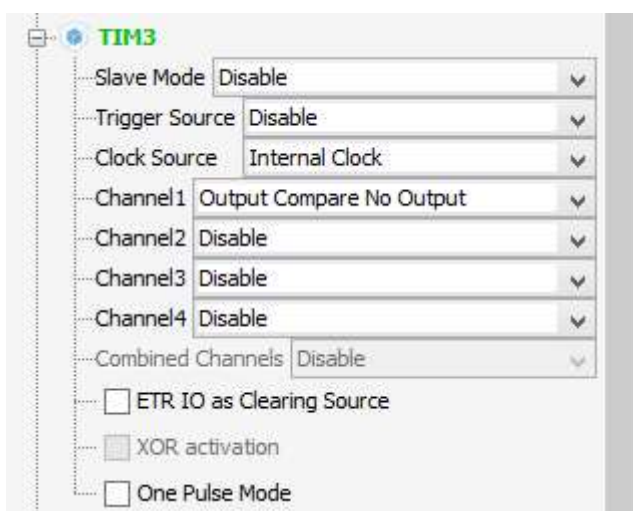
Timer	x-bitowy	Typ licznika	Prescaler	Obsługa DMA	Capture/compare Ilość kanałów
TIM2, TIM3, TIM4	16-bit	Up, down, up/down	Int 1-65536	Tak	4
TIM9	16-bit	Up	int 1-65536	Nie	2
TIM10, TIM11	16-bit	Up	int 1-65536	Nie	1
TIM6, TIM7	16-bit	Up	int 1-65536	Tak	0

## 8. NVIC (NESTED VECTOR INTERRUPT CONTROLLER)

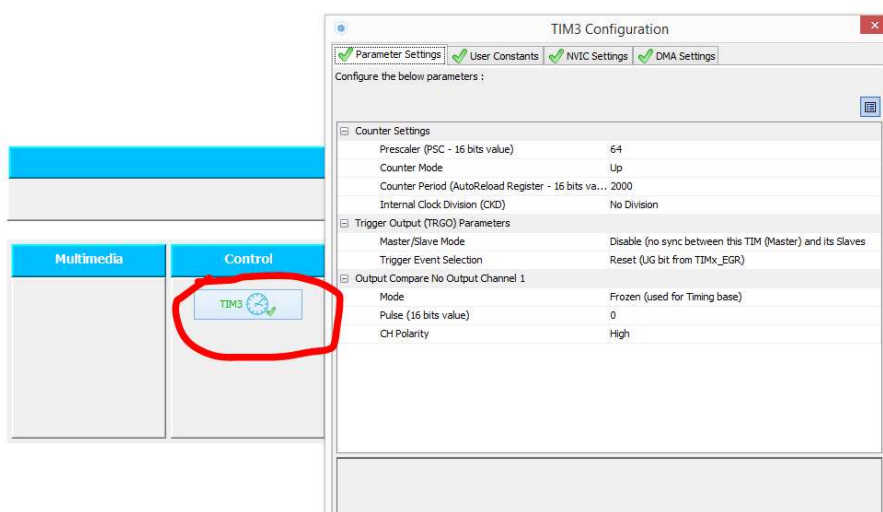
Rdzenie ARM wyposażone są w NVIC (Nested Vector Interrupt Controller). Jest to rozwiązanie sprzętowe pozwalające na obsługę przerwania o wyższym priorytecie, nawet jeśli w danej chwili jest wykonywane inne przerwanie, bez interwencji CPU. Więcej szczegółów znajduje się w dokumentacji.

## 9. KONFIGURACJA TIMERÓW W ŚRODOWISKU CUBE

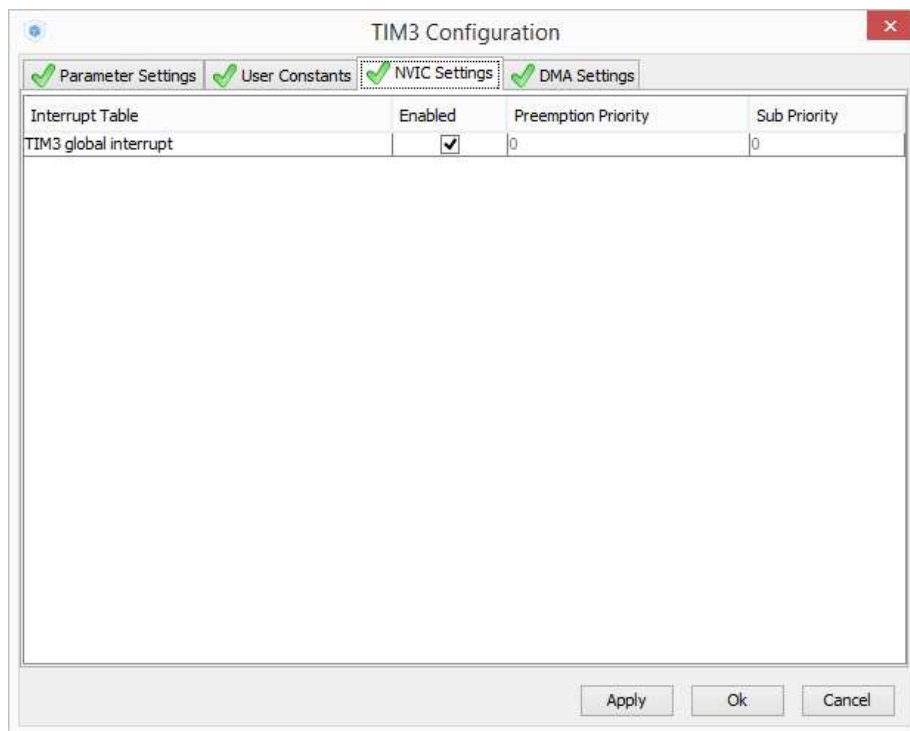
Poniższe print screeny przedstawiają konfigurację Timera 3.



Zrzut 1.1 Ustawienie TIM3 z zegarem wewnętrznym w odpowiednim trybie na kanale 1 (sygnał nie jest wyprowadzony na pin, ale może być wykorzystany, np. do przerwania)



Zrzut 1.2 Ustawienie odpowiednich parametrów TIM3





Zrzut 1.3 Włączenie przerwania od TIM3

Aby uruchomić Timer 3 oraz PWM należy dopisać funkcje, w wygenerowanym kodzie w języku C, w kompilatorze IAR przedstawione poniżej:

```
HAL_TIM_Base_Start_IT(&htim3);           //Start TIM3

HAL_TIM_PWM_Start_IT(&htim4, TIM_CHANNEL_2); //Start PWM, kanał2
```

## ZADANIA PRAKTYCZNE

1. Należy przeanalizować pracę mikrokontrolera poprzez analizę kodu dostarczonego przez prowadzącego. Należy uruchomić kompilator IAR dla ARM, następnie nagrać program na płytkę ewaluacyjną za pomocą kabla USB. Zadaniem jest zaobserwowanie przebiegu programu, ustawienie pracy krokowej w kompilatorze, oraz uruchomienie podglądu zmiennej  $n$  w oknie „Locals”. Po zatrzymaniu programu , a następnie jego wznowieniu  można zaobserwować zmianę wartości.

1.1 Proszę zmienić górną wartość zmiennej  $n$ , jaki ma to wpływ na świecenie diody?

2. W programie STM32CubeMX proszę skonfigurować pin obsługujący diodę LED4. Należy sprawdzić na schemacie płytki ewaluacyjnej, do którego portu podpięta jest dioda i następnie odpowiednio skonfigurować ten port, tak aby dioda świeciła się światłem ciągłym. Wygenerowany kod przez Cube, należy otworzyć w środowisku IAR, przekompilować i nagrać na płytkę.

2.1 W programie STM32CubeMX proszę skonfigurować odpowiednio porty obsługujące **dwie diody LED** oraz przycisk **USER**. Należy sprawdzić na schemacie płytki ewaluacyjnej, do którego portu są one podpięte, następnie odpowiednio skonfigurować te porty. Wygenerowany kod należy przez Cube, należy otworzyć w środowisku IAR, przekompilować i nagrać na płytkę. W środowisku IAR należy dopisać kod, tak aby po naciśnięciu przycisku, pojawiała się jakaś zmiana na diodach (według uznania). Przycisk jest konfigurowany podobnie jak dioda, z tą różnicą, że jest wejściem, nie wyjściem.

3. Proszę skonfigurować Timer 3 do wygenerowania przerwania o dowolnym okresie. Proszę nagrać program na płytkę i w podglądzie zmiennych w IAR dla rejestrów (Register) sprawdzić działanie Timera. Proszę wskazać rejestr będący licznikiem.

3.1 Proszę skonfigurować Timer 3 do wygenerowania przerwania o okresie 3000 cykli zegarowych. Proszę w przerwaniu inkrementować dowolną zmienną „czas” typu short. Proszę nagrać program na płytkę i w podglądzie zmiennych w IAR dla rejestrów (Register) sprawdzić działanie Timera poprzez sprawdzenie, czy jego licznik jest inkrementowany. Następnie proszę sprawdzić, czy program wchodzi w przerwanie, albo przez ustawienie breakpointa, albo przez podgląd zmiennej „czas”, w podglądzie zmiennych.



