

Ćwiczenie 1

Wprowadzenie

Bibliografia

- [1] Specyfikacja mikrokontrolera LM4F232H5QD - <http://www.ti.com/lit/ds/symlink/tm4c123gh6pge.pdf>
- [2] Instrukcja obsługi płyty EasyMxPROv7 <https://www.scribd.com/document/311182202/Easymx-Pro-v7-Stellaris-Manual-v102>
- [3] Materiały z warsztatów TI http://software-dl.ti.com/trainingTTO/trainingTTO_public_sw/GSW-TM4C123G-LaunchPad/TM4C123G_LaunchPad_Workshop_Workbook.pdf
- [4] Instrukcja obsługi biblioteki TivaWare Peripheral Driver Library <https://www.ti.com/lit/pdf/spmu298>
- [5] Instrukcja obsługi biblioteki TivaWare USB Library <https://www.ti.com/lit/pdf/spmu297>
- [6] Instrukcja obsługi biblioteki TivaWare Graphics Library <https://www.ti.com/lit/pdf/spmu300>
- [7] Instrukcja obsługi biblioteki TivaWare Sensor Library <https://www.ti.com/lit/pdf/spmu371>

Mikroprocesor TM4C123GH6PGE

Mikroprocesor TM4C123GH6PGE (dawniej LM4F232H5QD) (rys.1) należy do rodziny procesorów opartych na rdzeniu ARM Cortex-M4. Procesor ten jest jednym z wielu procesorów produkowanych przez firmę Texas Instruments (rys. 2). Charakteryzują go dobre parametry, wystarczające do podstawowych zastosowań domowych oraz przemysłowych, przy zachowaniu niskiej ceny.

Mikroprocesory z rodziny TM4C dzięki zastosowaniu rdzenia ARM M4, dodatkowych elementów (tj. wydajne konwertery analogowo cyfrowe, moduły komunikacji szeregowej, kontroler USB, generatory PWM) oraz możliwości pracy w trybie oszczędzania energii mają zastosowanie w różnorodnych aplikacjach np.:

- Urządzenia typu *hand-held* o niskim poborze mocy
- Sprzęt do gier
- Systemy monitoringu oraz sterowania
- Sterowanie ruchem
- Urządzenia medyczne
- Urządzenia pomiarowe i testowe
- Automatyka
- Bezpieczeństwo
- Rozwiązania typu SmartGrid
- Sterowanie oświetleniem
- Transport



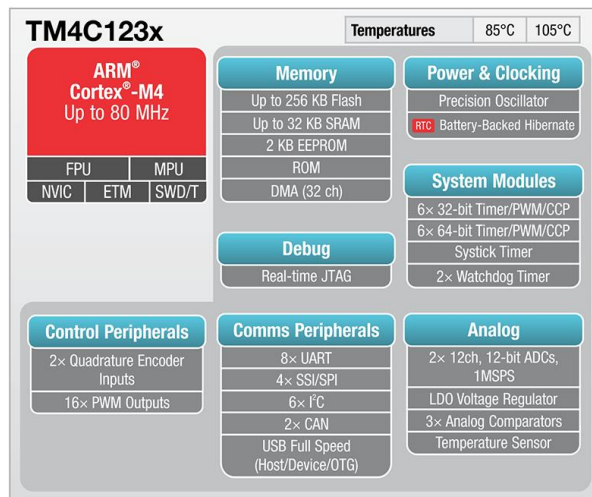
Rys. 1. Mikrokontroler TM4C123GH6PGE

Microcontrollers (MCUs)		ARM®-based Processors			Digital Signal Processors (DSP)	
16-bit Ultra Low Power MCU	32-bit Real-Time MCUs	32-bit ARM® MCU	32-bit ARM® Processors for Performance Applications	Application Processors	Singlecore DSP	Multicore DSP
<ul style="list-style-type: none"> • MSP430™ 	<ul style="list-style-type: none"> • C2000™ 	<ul style="list-style-type: none"> • Tiva™ C Series Cortex™-M4 • TMS570 Cortex™-R4 • RM4 Cortex™-R4F 	<ul style="list-style-type: none"> • Sitara Cortex®-A8, Cortex®-A9 and ARM9® • KeyStone Cortex®-A15 and Cortex®-A15 + DSP 	<ul style="list-style-type: none"> • OMAP™ Processors with Cortex™-A8, A9 and A15 	<ul style="list-style-type: none"> • C6000™ Power Optimized • C5000™ Ultra Low Power 	<ul style="list-style-type: none"> • KeyStone Multicore • KeyStone Multicore DSP+ARM • C6000™ Multicore
Frequency	Frequency	Frequency	Frequency	Frequency	Frequency	Frequency
Up to 25MHz	40MHz to 300MHz	Up to 220MHz	300MHz to 5 GHz	Up to 1.7 GHz	Up to 1.5 GHz Floating DSP + Video Accelerators + FFT Accelerators	Up to 15GHz Multicore, fixed/ floating + Accelerators
Memory (KB)	Memory (KB)	Memory (KB)	Memory (KB)	Memory (KB)	Memory (KB)	Memory (KB)
Flash 1KB to 256KB	Flash, RAM 16KB to 512KB	Flash 64 KB to 3MB	Up to 32KB I/D cache 256 KB L2, LPDDR1, DDR2/3 support	Up to 32KB i/D L1 caches and 2 MB L2 cache	L2 Cache mDDR, DDR2/DDR3 Up to 320 KB RAM Up to 256 KB ROM	Up to 4 MB SL2, 32 KB L1, 1MB L2
Key Attributes	Key Attributes	Key Attributes	Key Attributes	Key Attributes	Key Attributes	Key Attributes
Analog I/O, ADC, LCD, USB	PWM, ADC, CAN, SPI, I2C	USB, ENET MAC+PHY, CAN, ADC, PWM, SPI	USB 3.0 +PHY, Gigabit Ethernet MAC, PCIe+PHY, SATA+PHY, CAN, PRU-ICSS, 3D graphics, touchscreen	USB2/3, eMMC, SD/MMC, I2S, SPI, UART, HDMI, SATA, multiple camera, multiple display	USB 2.0 +PHY, Gb EMAC, PCIe+PHY, SATA+PHY, CAN, PRU ADC, SPI McBSP, I2C, LCD, On-chip regulators	C66x DSP, ARM® Cortex™-A15, RapidIO, PCIe, 10GigE, USB 3.0, Hyperlink, DDR3, NetCP
Applications	Applications	Applications	Applications	Applications	Applications	Applications
Measurement, sensing, general purpose	Industrial automation, retail applications/ point-of sale, HMI (Human Machine Interface), smart grid, cloud infrastructure	Motion control, HMI, industrial automation, automotive, safety	Industrial automation, retail applications/point-of sale, HMI (Human Machine Interface), smart grid, home and building automation.	Automotive, industrial automation, enterprise and mobile consumer	Portable audio/voice, fingerprint biometrics, portable medical, video, security, conferencing, test and measurement	Machine vision, video infrastructure, medical imaging, mission critical, base stations
\$0.25 to \$9.00 USD	\$1.85 to \$20.00 USD	\$1.00 to \$8.00 USD	\$5.00 to \$140.00 USD	\$5.00 to \$200.00 USD	\$1.99 to \$200.00 USD	\$1.99 to \$200.00 USD

Rys. 2. Portfolio procesorów Texas Instruments

Główne cechy procesora TM4C123GH6PGE to (rys. 3):

- 32-bitowy rdzeń ARM Cortex M4
- Niski pobór mocy
 - Moduł hibernacji
- Częstotliwość taktowania wynosząca 80 MHz
 - Elastyczny system taktowania
- Pamięć
 - 256 KB pamięci flash
 - 32 KB pamięci SRAM (*Static Random Access Memory*)
 - 2 KB pamięci EEPROM (*Electrically-Erasable Programmable Read-Only Memory*)
 - Wewnętrzna pamięć ROM (*Read Only Memory*) z wbudowanym oprogramowaniem TivaWare (dawniej StellarisWare)
- Jednostka zmiennoprzecinkowa
 - Jednocyklowe mnożenie oraz sprzętowy dzielnik
 - Zgodna ze standardem IEEE754
- Łączność szeregową
 - 8 układów UART (*Universal Asynchronous Receiver/Transmitter*)
 - 4 moduły SSI (*Synchronous Serial Interface*)
 - 6 modułów I²C (*Inter-Integrated Circuit*)
 - 2 kontrolery CAN (*Controller Area Network*) 2.0 A/B
 - Kontroler USB
- Timery
 - 12 układów timera GPTM (*General-Purpose Timer*) (6 działających na 16/32 bitach oraz 6 działających na 32/64 bitach)
 - 2 timery typu watchdog
- 14 fizycznych układów GPIO (*General-Purpose Input/Output*)
- 2 moduły PWM (*Pulse Wave Modulator*), każdy z czterema układami generatora, dające łącznie 16 wyjść PWM
- 2 moduły QEI (*Quadrature Encoder Interface*)
- 2 moduły 12 bitowych konwerterów analogowo cyfrowych o częstotliwości próbkowania miliona próbek na sekundę
- 32 kanałowy interfejs μ DMA (*Ultra-Direct Memory Access*)
- Kontroler przerwań NVIC (*Nested-Vectored Interrupt Controller*)
 - 7 wyjątków
 - 71 źródeł przerwań
 - 8 programowalnych poziomów priorytetów



Rys. 3. Cechy mikrokontrolera TM4C123GH6PGE

Płyta EasyMxPRO™v7 for STELLARIS® ARM®



Rys. 4. Płyta EasyMx PRO™ v7 for Stellaris® ARM®

Płyta EasyMx PRO™ v7 for Stellaris® ARM® (rys. 4) jest bogato wyposażoną platformą przeznaczoną do programowania mikrokontrolerów z rodziny Stellaris® ARM® Cortex™-M3 i Cortex™-M4. Zawiera wiele wbudowanych modułów pozwalających na tworzenie różnorodnych aplikacji. Są to m.in. ekran TFT (*Thin-film transistor*) wraz z panelem dotykowym, kodek mp3, czytnik kart pamięci microSD, łącza USB, brzęczyk piezoelektryczny, złącze Ethernet oraz CAN. Znajdujący się na płycie programator i debugger mikroProg™ może współpracować z ponad 270 mikrokontrolerami z rodziny ARM.

Budowa płyty EasyMxPRO

Płyta EasyMxPRO wyposażona jest w wiele modułów pozwalających na tworzenie szerokiego zakresu aplikacji.

Grupy wejść/wyjść

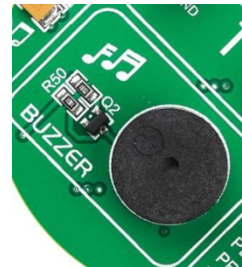
Na płycie znajdują się 8 grup wejść/wyjść. Każdą grupę tworzą: złącze portu, trójstanowy przełącznik DIP, przyciski oraz diody LED.

- **Trójstanowy przełącznik DIP**
Służy do włączania/wyłączania rezystorów typu *pull-up* (pozycja do góry) lub *pull-down* (pozycja do dołu) dla danej nóżki na danym porcie.
- **Złącze portu**
Umożliwia podłączenie różnego rodzaju akcesoriów z damską wersją złącza IDC10.
- **Przyciski**
Służą do zmiany stanu wszystkich wejść mikrokontrolera. Przełącznik SW16 pozwala na wybór, jaki stan logiczny ma zostać podany po naciśnięciu przycisku na odpowiednim pinie mikrokontrolera. Ustawienie pozycji „do góry” spowoduje podanie logicznej jedynki na wejście mikrokontrolera, a pozycji „na dół” - logicznego zera. Przełącznik w środkowej pozycji wyłącza działanie przycisku.
- **Diody LED**
Służą do wskazywania stanu logicznego na danym pinie danego portu. Dioda pozostaje zapalona, jeśli na pin podana jest logiczna jedynka. Do sterowania i włączania diod na każdym z portów służy przełącznik SW15.



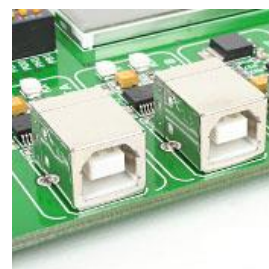
Brzęczyk piezoelektryczny

Brzęczyk piezoelektryczny, znajdujący się na płycie EasyMxPRO, podłączony jest do pinu PA6 mikrokontrolera. Jego włączenie odbywa się za pomocą przełącznika SW12.1. Mikrokontroler może tworzyć dźwięk generując sygnał PWM będący serią logicznych jedynek i zer. Częstotliwość sygnału określa wysokość tonu generowanego dźwięku, a wypełnienie sygnału PWM determinuje głośność dźwięku. Do wytworzenia sygnału PWM służą moduły PWM dostępne w mikrokontrolerze.



Złącza USB UART

Płyta EasyMxPRO posiada 2 niezależne złącza USB UART. UART jest jednym z najczęściej spotykanych narzędzi do wymiany danych pomiędzy mikrokontrolerem a urządzeniami zewnętrznymi. Nowoczesne komputery zazwyczaj nie są już wyposażone w złącza i kontrolery RS-232 ani UART; zamieniono je na złącza i kontrolery USB. Jednak, komunikacja UART jest nadal możliwa dzięki zastosowaniu konwerterów UART-USB. Płyta EasyMxPRO wyposażona jest w kontroler FT232RL, który umożliwia konwersję sygnałów UART do odpowiedniego standardu USB.



W celu uruchomienia komunikacji USB-UART na złączu A należy ustawić przełączniki SW10.1 oraz SW10.2 na pozycję włączoną. W ten sposób do linii PA0 i PA1 podłączone zostają linie nadawcze i odbiorcze.

W celu uruchomienia komunikacji USB-UART na złączu B należy ustawić przełączniki SW10.3 oraz SW10.4 na pozycję włączoną. W ten sposób do linii PD2 i PD3 podłączone zostają linie nadawcze i odbiorcze.

UWAGA: Podczas używania komunikacji USB-UART należy odłączyć wszystkie urządzenia oraz dodatkowe płytki, które mogłyby zakłócać sygnały i potencjalnie zniekształcić przesyłane/odbierane dane.

Złącze USB HOST

Płyta EasyMxPRO zawiera złącze USB HOST dla wtyczki USB typu A. Złącze to umożliwia zestawienie połączenia z zewnętrznym urządzeniem, np. klawiaturą USB bądź myszką USB. Złącze służy także jako źródło napięcia o wartości 5V. Linie danych USB są bezpośrednio podłączone do mikrokontrolera. Istnieje możliwość włączenia/wyłączenia zasilania przez USB przez odpowiednie przedstawienie przełącznika SW10.7.



Złącze komunikacji USB

Płyta EasyMxPRO posiada złącze USB DEVICE dostosowane do wtyczki USB typu B, które umożliwia ustanowienie połączenia z docelowym hostem np. komputerem PC, laptopem itp. Pozwala na stworzenie urządzenia USB typu *slave*.

Wykrycie czy płyta podłączona jest do urządzenia HOST odbywa się za pomocą linii VBUS, która jest doprowadzona do pinu PB1 mikrokontrolera. Podłączenie linii VCC urządzenia USB do pinu PB1 odbywa się z wykorzystaniem przełącznika SW10.8.



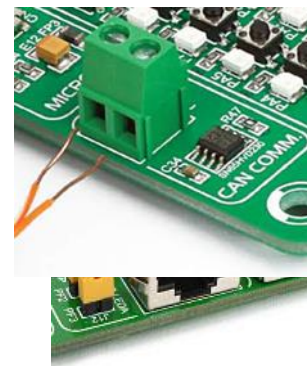
Złącze komunikacji Ethernet

Płyta EasyMxPRO posiada standardowe złącze RJ-45 do komunikacji z wykorzystaniem standardu Ethernet. Złącze to umożliwia podłączanie do płyty komputera, rutera lub innego urządzenia obsługującego standard Ethernet. Wszystkie niezbędne linie ethernetowe (TPOUT+, TPOUT-, TPIN+, TPIN-) są bezpośrednio podłączone do mikrokontrolera. W celach sygnalizacyjnych przy złączu umiejscowiono dwie diody LED, które można aktywować zworkami J11 i J12.

Złącze komunikacji CAN

Płyta EasyMxPRO wyposażona jest w układ nadawczo-odbiorczy standardu CAN, wykorzystywanym w pojazdach do komunikacji pomiędzy urządzeniami bez obecności hosta. Pomimo że CAN, który jest protokołem opartym na komunikatach, został zaprojektowany do zastosowań motoryzacyjnych, jest on także wykorzystywany w automatyce czy też w sprzęcie medycznym.

W celu umożliwienia komunikacji CAN, należy ustawić przełączniki SW10.5 oraz SW10.6 na pozycję włączoną. Połączy to linie nadawczą (TX) oraz odbiorczą (RX) do odpowiednich nóżek mikrokontrolera.



Wejścia/wyjścia audio

Na płycie EasyMxPRO znajduje się wysokiej klasy kodek audio VS1053. W jednym układzie zawarto dekodery Ogg Vorbis/MP3/AAC/WMA/FLAC/WAV/MIDI oraz koder PCM/IMA ADPCM/Ogg Vorbis. Kodek VS1053 otrzymuje wejściowy strumień bitów poprzez szeregową magistralę wejściową, którą monitoruje jako urządzenie typu *slave*. Strumień wejściowy jest dekodowany i poddany cyfrowej kontroli głośności, by następnie zostać poddanym na wielobitowy, nadpróbkowujący przetwornik cyfrowo analogowy typu sigma-delta. Dekodowanie jest sterowane poprzez szeregową magistralę kontrolną. Oprócz podstawowego trybu dekodowania, możliwe jest także dodanie efektów DSP.

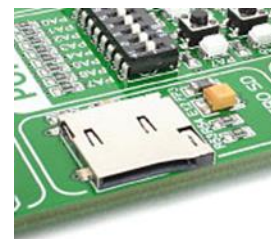
W celu wykorzystania modułu Audio I/O należy podłączyć linie danych oraz sterowania Audio mikrokontrolera do kodeka VS1053. W tym celu należy ustawić przełączniki SW13.1-SW13.3 oraz SW14.1-SW14.4 na pozycję włączoną. Dzięki temu linie danych SPI (ang. *Serial Peripheral Interface*) zostają połączone z nóżkami PA5, PA4 i PA2 mikrokontrolera, a także linie sterujące audio oraz linie wyboru układu do nóżek PF4, PF5, PF1 oraz PF0.



Wejście kart microSD

Płyta EasyMXPRO posiada czytnik kart pamięci microSD wykorzystujący standardowy interfejs użytkownika oparty na SPI.

Aby uzyskać dostęp do kart microSD należy uaktywnić linie komunikacyjne SPI wykorzystując przełączniki SW13.1 - SW13.3, a także linie „Chip Select” i „Card Detect” wykorzystując przełączniki SW13.8 i SW13.7



Wyświetlacz TFT z panelem dotykowym

Płyta EasyMxPRO wyposażona w kolorowy, podświetlany diodami LED wyświetlacz TFT z panelem dotykowym o rozdzielczości 320x240 pikseli. Każdy piksel może wyświetlać ponad 262 tys. różnych kolorów. Wyświetlacz podłączony jest do mikrokontrolera poprzez równoległy 8-bitowy interfejs 8080 oraz dodatkowe linie sterujące. Płyta posiada także sterownik podświetlenia, który umożliwia regulację jasności za pomocą sygnału PWM.



Panel dotykowy to szklana powierzchnia pokryta dwoma warstwami rezystancyjnego materiału. Kiedy dotykamy panel, warstwa zewnętrzna naciska na warstwę wewnętrzną a kontroler mierzy siłę nacisku oraz określa położenie dotyku. Płyta EasyMxPRO wyposażona jest w kontroler panelu dotykowego a także podłączenie dla czteroprzewodowych rezystancyjnych paneli dotykowych. Kontroler jest w stanie bardzo precyzyjnie określić siłę nacisku w danym punkcie przedstawiając współrzędne miejsca dotyku w postaci wartości napięcia elektrycznego.

Wyświetlacz włącza się za pomocą przełączników SW11.1 - SW11.8 oraz SW12.2 - SW12.6. Podświetlenie można włączyć na dwa sposoby. Pierwszy to użycie przełącznika SW12.7, które włączy podświetlenie z maksymalną wartością jasności. Drugi to wykorzystanie sygnału PWM dzięki czemu można dostosować jasność ekranu. Aby umożliwić sterowanie jasnością poprzez sygnał PWM, należy ustawić przełączniki SW12.7 i SW12.8 na pozycję włączoną. Panel dotykowy włączany jest za pomocą przełączników SW14.5 - SW14.8, które łączą linie READ-X i READ-Y z analogowymi wejściami mikrokontrolera PB4 i PB5, oraz linie DRIVEA i DRIVEB z cyfrowymi wyjściami mikrokontrolera PE0 i PE1. Przy korzystaniu z panelu dotykowego należy się upewnić, że dodatkowe urządzenia zewnętrzne oraz diody LED i rezystory *pull-down* i *pull-up* są odłączone z linii interfejsu. W przeciwnym razie mogą zakłócać integralność sygnałów danych.

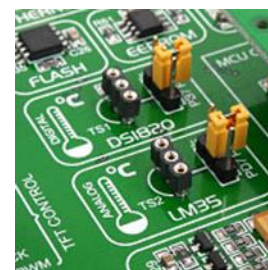
Dżojstik

Na płycie EasyMxPRO znajduje się dżojstik z pięcioma opcjami położenia (góra, dół, lewo, prawo oraz środek). Każda pozycja działa na zasadzie przycisku połączonych do odpowiedniej nóżki (są to nóżki: PB0, PE5, PB7, PE4 i PH2). Przed użyciem dżojstika należy ustawić odpowiednie przełączniki z grup wejść/wyjść na pozycję *pull-up*. Naciśnięcie dżojstika powoduje podłączenie odpowiedniej nóżki mikrokontrolera do linii GND, co może zostać wykryte przez odpowiednie oprogramowanie.



Czujniki temperatury

Płyta EasyMxPRO pozwala na podłączenie cyfrowego (DS18B20) i analogowego (LM35) czujnika temperatury. Czujnik DS18B20 pozwala mierzyć temperaturę (z 9-bitową rozdzielczością) w zakresie -55°C do 128°C z dokładnością 0,5°C (dla temperatur z zakresu -10°C do 88°C). Na wyznaczenie temperatury potrzeba maksymalnie 750 ms. Czujnik LM35 pozwala mierzyć temperaturę w zakresie 2°C do 150°C. Działanie czujnika LM35 oparte jest na liniowej zależności napięcia wyjściowego czujnika z temperaturą w stopniach Celsjusza. Napięcie rośnie w tempie +10 mV/°C przy poborze prądu mniejszym niż 60 μ A dzięki czemu efekt auto-ogrzewania jest zminimalizowany i wynosi mniej niż 0,1°C.



Czujnik DS18B20 podłączony jest poprzez interfejs 1-wire do nóżek PB7 lub PD4 mikrokontrolera, wybór nóżki odbywa się poprzez zworkę J8. Dane z czujnika LM35 można odczytać z nóżek PD4 lub PD7, wybór

nóżki odbywa się przez zworkę J10. Przy wkładaniu czujnika należy zwrócić uwagę na odpowiednie podłączenie. Należy także pamiętać o odłączeniu niepotrzebnych elementów, które mogą zakłócać odczyty.

Szeregowa pamięć Flash

Płyta EasyMxPRO posiada szeregową pamięć Flash M25P80 o pojemności 8 Megabitów, zorganizowanej w 16 sektorów po 256 stron każdy. Każda strona to 256 bajtów. Cała pamięć może być rozpatrywana jako 4096 stron lub 1048576 bajtów. Szeregowa pamięć Flash wykorzystuje interfejs SPI do komunikacji z mikroprocesorem. Maksymalna częstotliwość dla polecenia odczytu to 40 MHz. W celu podłączenia pamięci Flash do mikrokontrolera należy włączyć przełączniki SW13.1-SW13.3 oraz SW13.6, co spowoduje podłączenie linii SPI do nóżek PA5, PA4, PA2 i PA7 mikrokontrolera.

Pamięć EEPROM

Na płycie EasyMxPRO znajdują się pamięć EEPROM o pojemności 1024 bajtów. Pamięć EEPROM jest zapasową pamięcią, w której przechowywane dane pozostają mimo utraty napięcia. Do komunikacji wykorzystywany jest interfejs I²C. Na pamięci można wykonywać 16 bajtowe operacje odczytu i zapisu.

W celu podłączenia pamięci EEPROM należy uaktywnić przełączniki SW13.4 oraz SW13.5. Przed użyciem należy pamiętać o odłączeniu innych elementów z linii interfejsu, w przeciwnym razie dane mogą zostać przekłamane.

Wejścia konwertera A/C

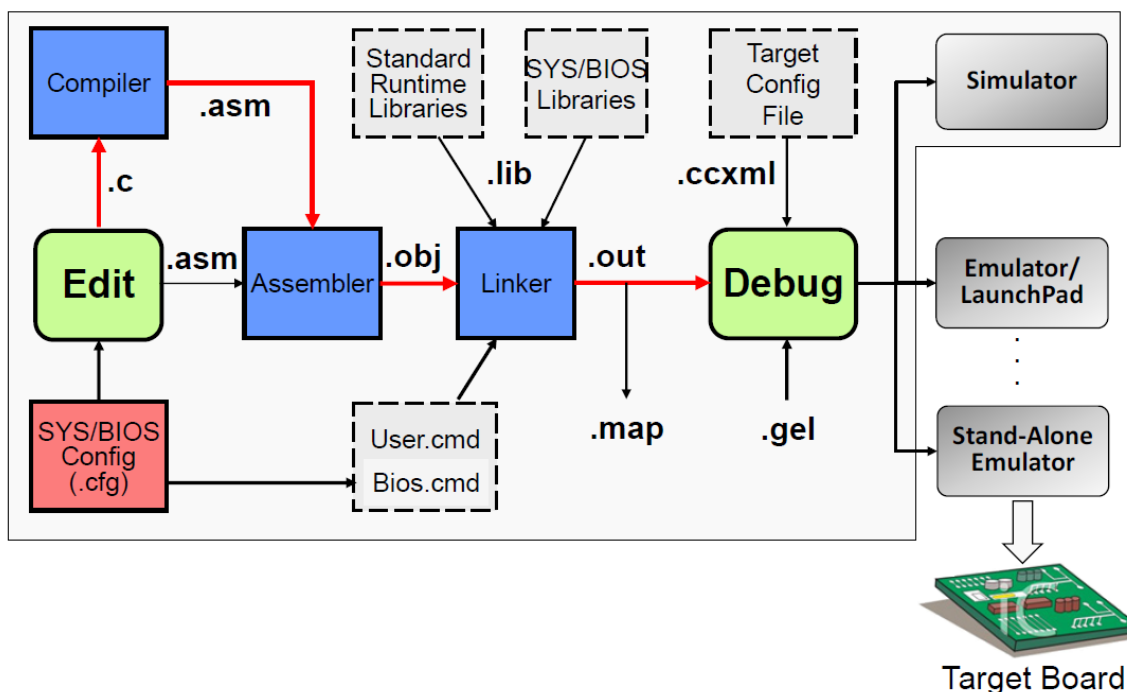
Płyta EasyMxPRO posiada interfejs do przetwornika analogowo cyfrowego w postaci potencjometru. Pozwala to na symulowanie różnych napięć wejściowych z zakresu GND do VCC. Wartości napięcia mogą być podane na jedno z pięciu analogowych wejść mikrokontrolera.

W celu połączenia wyjścia potencjometru z analogowymi wejściami mikrokontrolera PE7, PE6, PE5, PE4 lub PD7, należy umieścić zworkę J9 w odpowiedniej pozycji.



Code Composer Studio

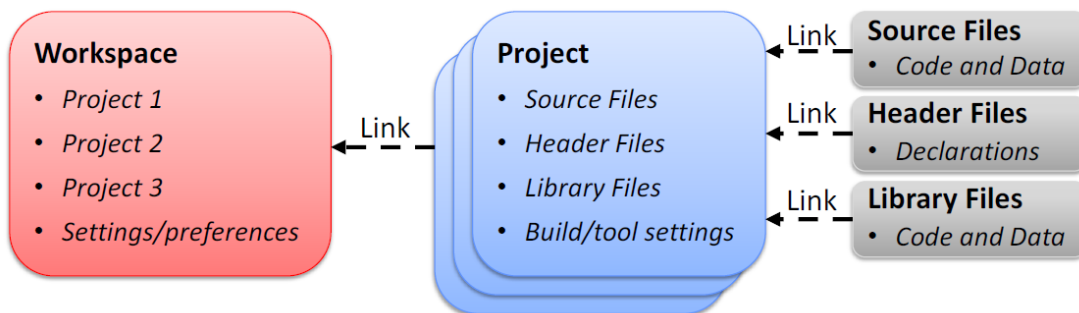
Code Composer Studio™ (CCS) jest zintegrowanym środowiskiem programistycznym (IDE, *Integrated Development Environment*) przeznaczonym dla rodzin procesorów firmy Texas Instruments, pozwalającym na tworzenie i debugowanie aplikacji systemów wbudowanych (Embedded). Na rys. 5. przedstawiono przegląd najważniejszych elementów funkcjonalnych CCS, a są to m.in. kompilatory dla każdego mikrokontrolera firmy TI, edytor kodu źródłowego, debugger, symulatory i wiele innych.



Rys. 5. Przegląd elementów funkcjonalnych oprogramowania Code Composer Studio

Code Composer Studio oparte jest na platformie Eclipse, która jest doskonałym narzędziem do tworzenia środowiska programistycznego i powoli staje się podstawową platformą wykorzystywaną przez dostawców oprogramowania dla systemów wbudowanych. CCS łączy bowiem zalety Eclipse z zaawansowanymi możliwościami debugowania systemów wbudowanych.

Praca na platformie Eclipse odbywa się w tzw. przestrzeni roboczej (*Workspace*). Przestrzeń robocza to miejsce, w którym znajdują się wszystkie projekty, które zostały w jej ramach utworzone. Projekt natomiast zawiera pliki z kodem źródłowym, pliki nagłówkowe oraz powiązane pliki bibliotek, które tworzą program. Zarówno przestrzeń robocza jak i pojedyncze projekty mają swoje oddzielne zbiory ustawień. Przy czym ustawienia przestrzeni roboczej będą miały wpływ na wszystkie znajdujące się w niej projekty. Struktura platformy Eclipse została przedstawiona na rys. 6.



Rys. 6. Struktura platformy Eclipse

Biblioteka TivaWare

Oprogramowanie TivaWare jest zbiorem narzędzi, mającym na celu uproszczenie oraz przyspieszenie pracy nad aplikacjami przeznaczonymi dla mikroprocesorów z rodziny Tiva. Wszystkie narzędzia oprogramowania TivaWare są darmowe oraz umożliwiają wykorzystanie we własnych aplikacjach na zasadach *royalty-free*. Całość biblioteki napisana jest w języku C co pozwala na tworzenie aplikacji w sposób łatwy i wydajny.

W skład oprogramowania TivaWare wchodzi m.in. :

- Biblioteki (Peripheral [4], USB [5], Graphics [6], Sensor [7])
- Przykłady kodów dla wielu zestawów testowych i różnych procesorów z rodziny TM4C123x
- Dokumentacje

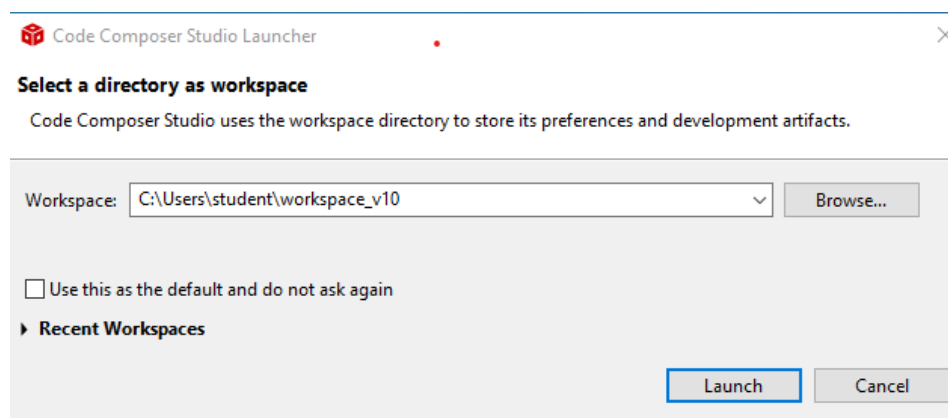
Przebieg ćwiczenia

Pierwsze uruchomienie

1. Uruchomić program CCS np. klikając na ikonę programu ze wstążki

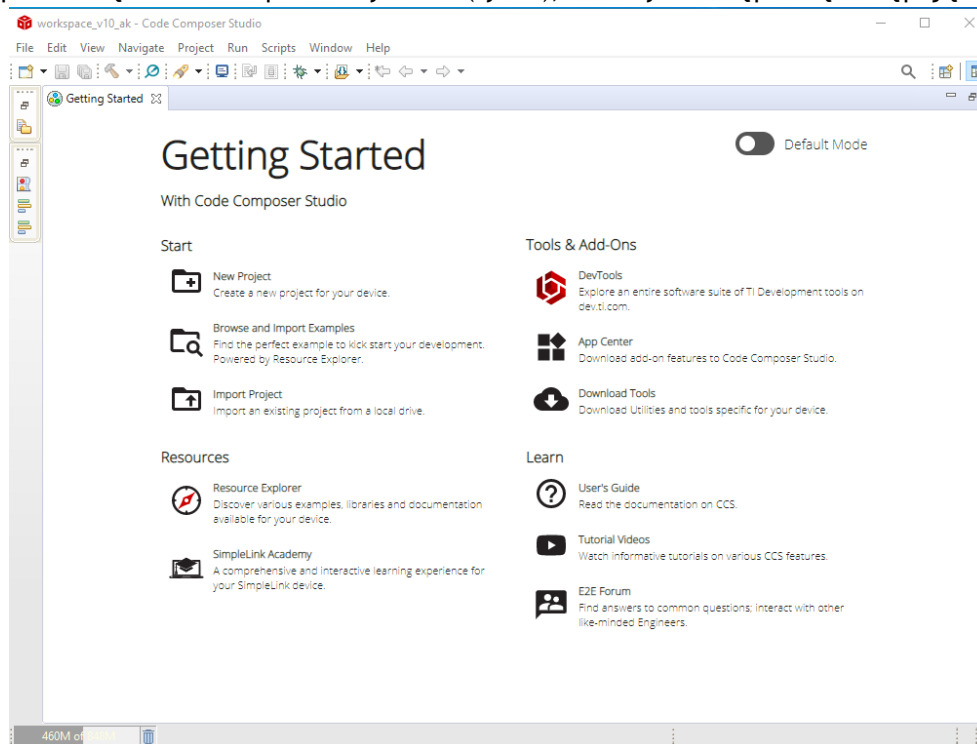


2. W polu wyboru przestrzeni roboczej (*Workspace*) (rys. 7) wybrać nową przestrzeń roboczą klikając w przycisk *Browse....* We wskazanym przez prowadzącego miejscu stworzyć nowy katalog o nazwie *Imię_Nazwisko_Grupa* oraz wewnątrz katalog *Workspace*. Wybrać właśnie utworzony katalog *Workspace*, jako nową przestrzeń roboczą i potwierdzić wybór przyciskiem OK.



Rys. 7. Okno wyboru przestrzeni roboczej

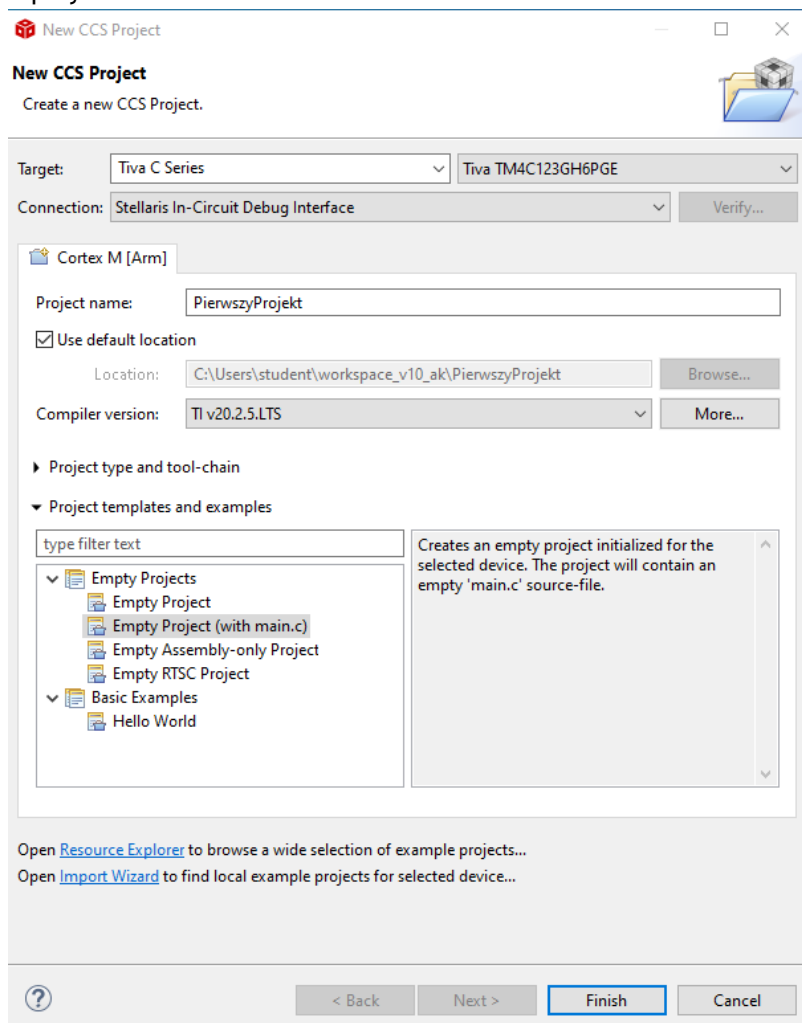
3. Zapoznać się z ekranem powitalnym CCS (rys. 8), w którym dostępne są następujące opcje:



Rys. 8. Ekran powitalny Code Composer Studio

Pierwszy projekt

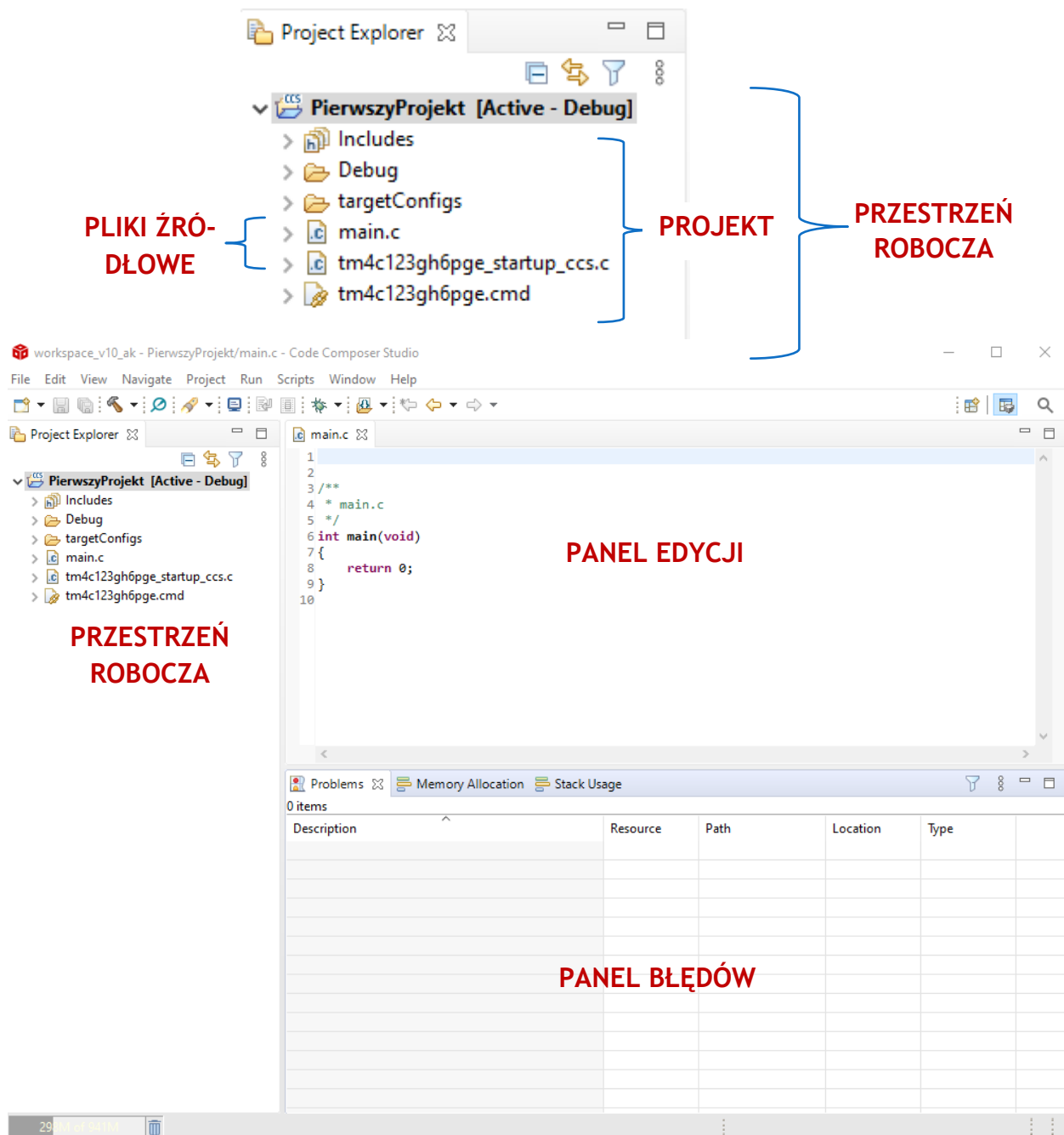
1. Utworzyć nowy projekt klikając w przycisk *New Project* na ekranie powitalnym lub poprzez kliknięcie *File->New->CCS Project*
2. W oknie *New CSS Project* (rys. 9) wpisać nazwę projektu w polu *Project name*, w grupie *Device* wybrać odpowiedni mikroprocesor: **Tiva TM4C123GH6PGE** (wpisanie 123G w polu *Target* pozwala na zmniejszenie listy urządzeń i łatwiejsze odnalezienie odpowiedniego urządzenia), *Connection* - **Stellaris In-Circuit Debug Interface**. Wybrać szablon *Empty Project (with main.c)*. Potwierdzić przyciskiem *Finish*.



Rys. 9. Okno *New CSS Project*

3. Zamknąć panel *TI Resource Explorer* (o ile nie zrobiliśmy tego wcześniej) i zapoznać się z zawartością panelu *Project Explorer* oraz perspektywą edycji (rys 10.). Okno *Project Explorer* można otworzyć klikając na odpowiednią ikonę.

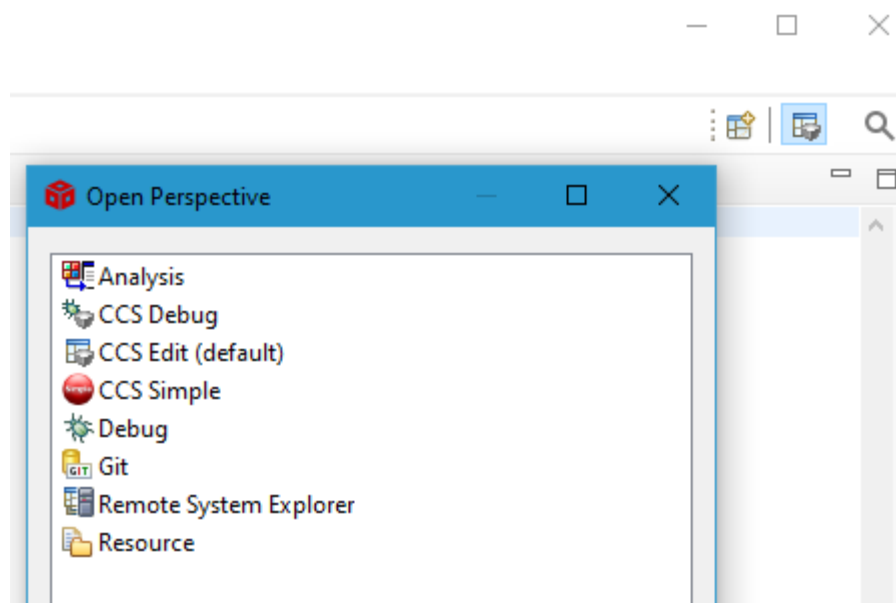




Rys. 10. Perspektywa edycji oraz zakładka *Project Explorer*

W środowisku można przelatać się pomiędzy różnymi perspektywami projektu - w tym celu należy wybrać odpowiednią ikonkę w prawym górnym rogu ekranu. W większości przypadków podczas zajęć będziemy wykorzystywać dwie perspektywy - Edit i Debug. Pierwsza z nich przystosowana jest do pisanie kodu, druga do jego debugowania.

UWAGA: położenie, rozmiar i inne parametry okien w ramach perspektywy można ustawiać indywidualnie według własnego uznania.

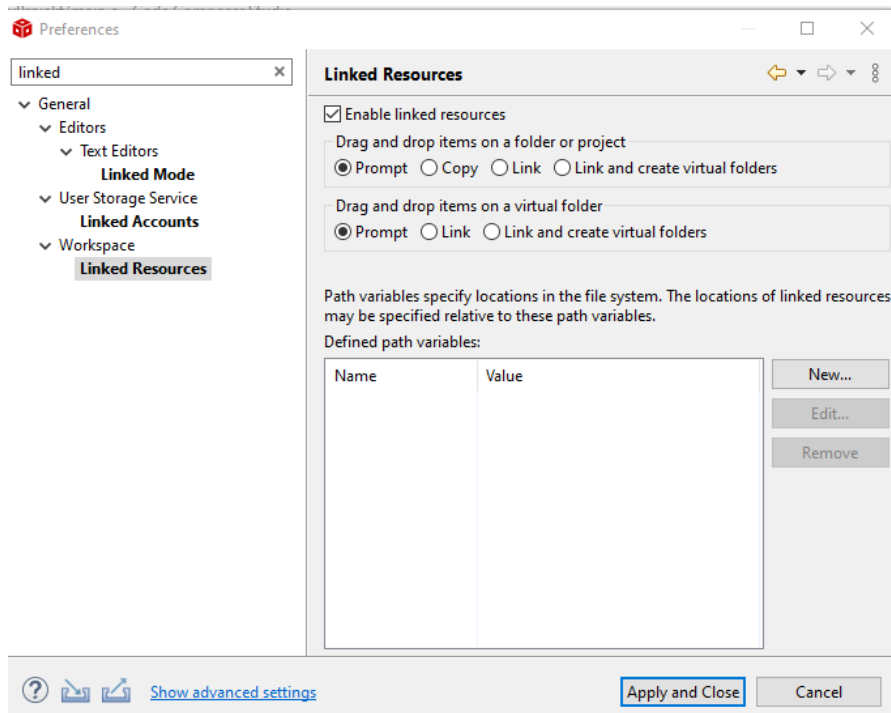


Okno wyboru perspektywy projektu

Zmienne Path i Build

Każdy z projektów w programie CCS posiada 2 specjalne typy zmiennych, są to: *Path Variables* oraz *Build Variables*. Zmienne typu *Path* są wykorzystywane do przechowywania ścieżek bazowych do plików podłączanych do projektów. Zmienne typu *Build* są wykorzystywane do przechowywania ścieżek do bibliotek oraz innych plików. Zmienne te mogą mieć zakres przestrzeni roboczej (powiązane z wszystkimi projektami w przestrzeni roboczej) bądź też zakres pojedynczego projektu. Ustawienie tych zmiennych można zrealizować na dwa sposoby. Pierwszy z nich to sposób manualny, w którym dla każdego projektu trzeba ustawiać zmienne osobno. Drugi sposób to sposób automatyczny z wykorzystaniem pliku vars.ini. Pozwala także na ustawienie zmiennych dla wszystkich projektów znajdujących się w przestrzeni roboczej. Ze względu na większą użyteczność drugiej metody, w dalszej pracy będziemy się nią posługiwać. Opis manualnego ustawiania zmiennych można znaleźć w [2] i [3].

1. Otwórz plik vars.ini znajdujący się w katalogu c:\TI\TivaWare. Wybierz *File->Open File* odnajdź plik vars.ini, po czym kliknij *Open*.
2. Zapoznaj się z zawartością pliku vars.ini. Zawiera on zmienną TIVA_WARE.
3. Wybierz *Window->Preferences*. W oknie dialogowym wpisz „linked” i kliknij w *Linked Resources* (rys. 11)

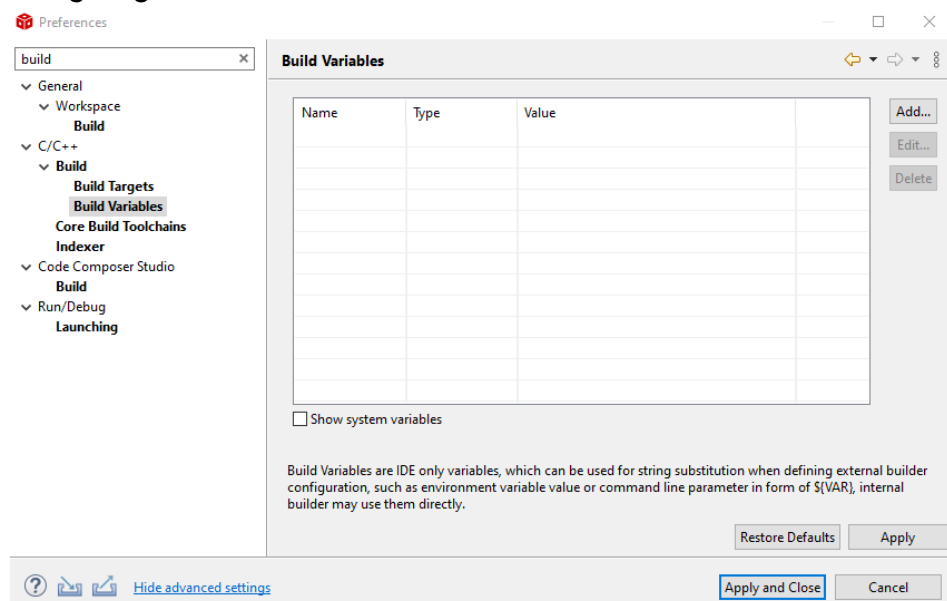


Rys. 11. Okno dialogowe Window->Preferences.

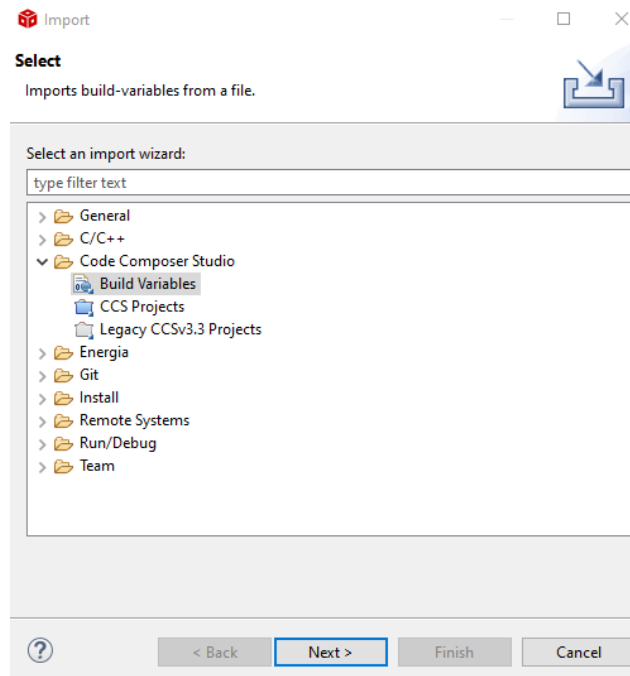
W ten sposób możemy dodać lub wyświetlić wszystkie zmienne typu *Path* w aktualnej przestrzeni roboczej. Aktualnie nie ma żadnej takiej zmiennej.

4. W tym samym oknie dialogowym wpisz „build” i kliknij w *Build Variables*. W tym miejscu można dodać lub wyświetlić zmienne typu *Build*. Tak samo jak w przypadku zmiennych typu *Path* nie mamy jeszcze zdefiniowanych zmiennych typu *Build*.

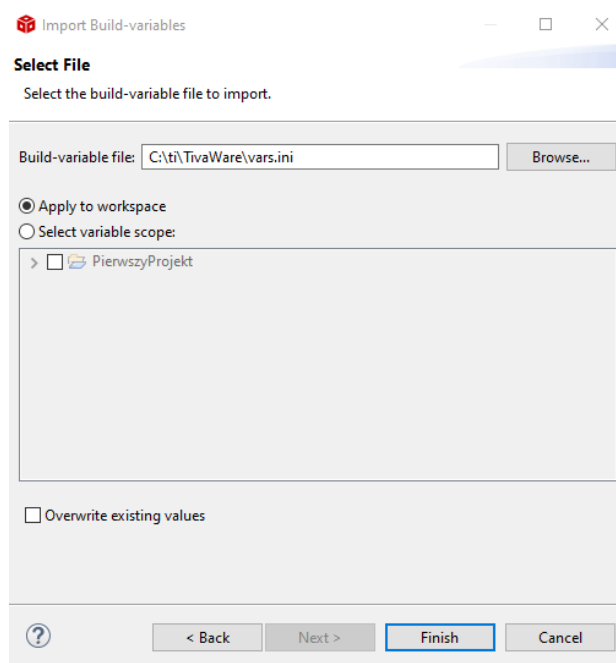
UWAGA: należy włączyć ustawienia zaawansowane (*Show advanced settings*) u dołu okna dialogowego



5. Zamknij okno właściwości klikając w *Cancel*.
6. Zaimportuj plik vars.ini. Wybierz File->Import. W oknie dialogowym wybierz *Build Variables* (rys. 12) i kliknij w *Next*.



Rys. 12. Okno dialogowe Import.



Rys. 13. Odnajdywanie ścieżki do pliku vars.ini.

7. Odnajdź plik vars.ini (rys. 13) i kliknij w *Finish*.
8. Powtórz kroki 3 i 4. Zweryfikuj czy zmienne Path i Build zostały poprawnie ustawione.

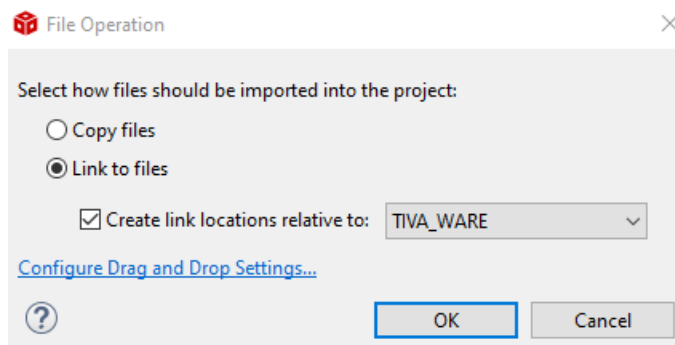
UWAGI DOTYCZĄCE VARS.INI

Zmiana przestrzeni roboczej oznacza konieczność ponownego zaimportowania pliku vars.ini.
Zmiana ścieżki instalacji biblioteki TivaWare wymaga aktualizacji pliku vars.ini.


Dodawanie plików do projektu

Każdy projekt powinien posiadać pliki źródłowe oraz łącza do bibliotek, z których korzysta. Główny plik źródłowy (main.c) został utworzony wraz z projektem, natomiast łącza do pliku biblioteki należy dodać ręcznie:

1. Dodaj połączenie do biblioteki *driverlib* należącej do TivaWare. Wybierz *Project->Add Files*.
2. Odnajdź plik *driverlib.lib* znajdujący się w katalogu *C:\ti\TivaWare\driverlib\ccs\Debug* i kliknij otwórz.
3. W oknie dialogowym (rys. 14) zaznacz *Link to files* a w polu *Create link locations relative to* wybierz wcześniej utworzoną zmienną *TIVA_WARE*. Kliknij OK.



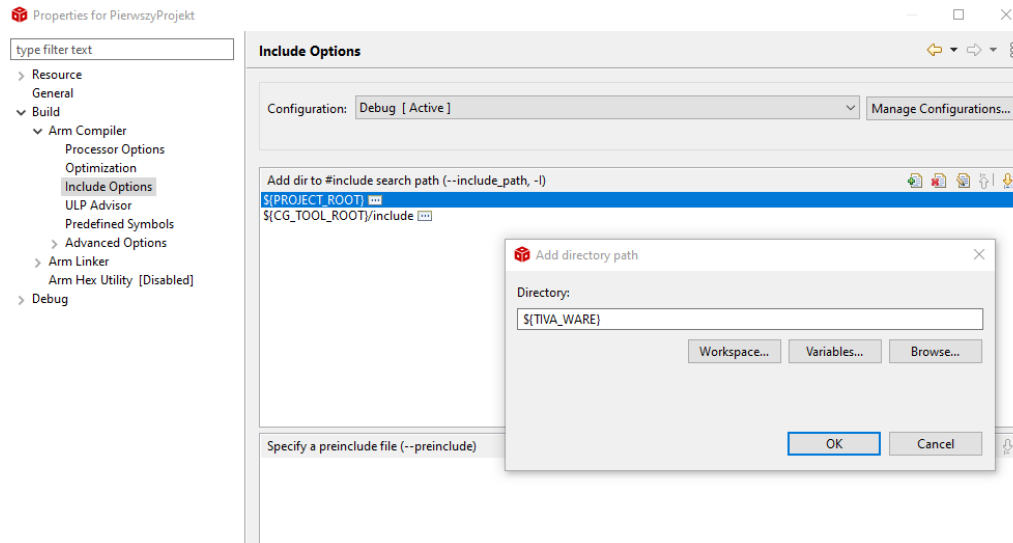
Rys. 14. Okno dialogowe File Operation

4. Przeanalizuj zawartość panelu *Project Explorer*. W strukturze projektu pojawił się plik *driverlib.lib* z ikoną . Ikona ta oznacza, że plik ten jest podłączony do projektu, ale nie znajduje się w nim bezpośrednio.
5. W pliku *main.c* dodaj następujące pliki nagłówkowe:

```
#include "stdint.h"
#include "stdbool.h"
#include "inc/hw_types.h"
#include "inc/hw_memmap.h"
#include "driverlib/sysctl.h"
#include "driverlib/gpio.h"
#include "driverlib/rom.h"
```

Więcej informacji o tych bibliotekach zostanie przekazane w trakcie następnych zajęć.

9. Przy niektórych plikach mogło pojawić się podkreślenie oraz znak zapytania na pasku po lewej stronie edytora. Najeżdżając na znak zapytania można podejrzec co jest przyczyną zaznaczenia danej linii. W tym przypadku mamy do czynienia z niewłaściwym podłączeniem pliku nagłówkowego. Aby naprawić ten błąd we właściwościach projektu należy wskazać gdzie szukać tych plików. Klikając prawym przyciskiem myszy na projekt wybieramy *Properties*.
10. Wybrać *Build->ARM Compiler ->Include Options*
11. Dodać wartość `${TIVA_WARE}` w sekcji *Add dir to #include search path* przez naciśnięcie przycisku *Add* (rys. 15). Zamknij właściwości klikając dwa razy w *OK*.
12. Znaki zapytania przy plikach nagłówkowych powinny zniknąć.



Rys. 15. Ustawienie ścieżki w *Include Options*.

Definiowanie kompilatora i urządzenia

Pliki nagłówkowe dodane w poprzednim punkcie są zaprojektowane do pracy z różnymi typami urządzeń oraz kompilatorów. Aby można było korzystać z funkcji wbudowanych w pamięci ROM mikroprocesora użytkownik musi zdefiniować z jakiego kompilatora i z jakiego urządzenia korzysta.

1. Otworzyć właściwości projektu i w zakładce *Build->ARM Compiler->Predefined Symbols* w sekcji *Pre-define NAME* nacisnąć przycisk *Add* (zielony krzyżyk) (rys. 16) i wpisać:

```
CCS="CCS"
```

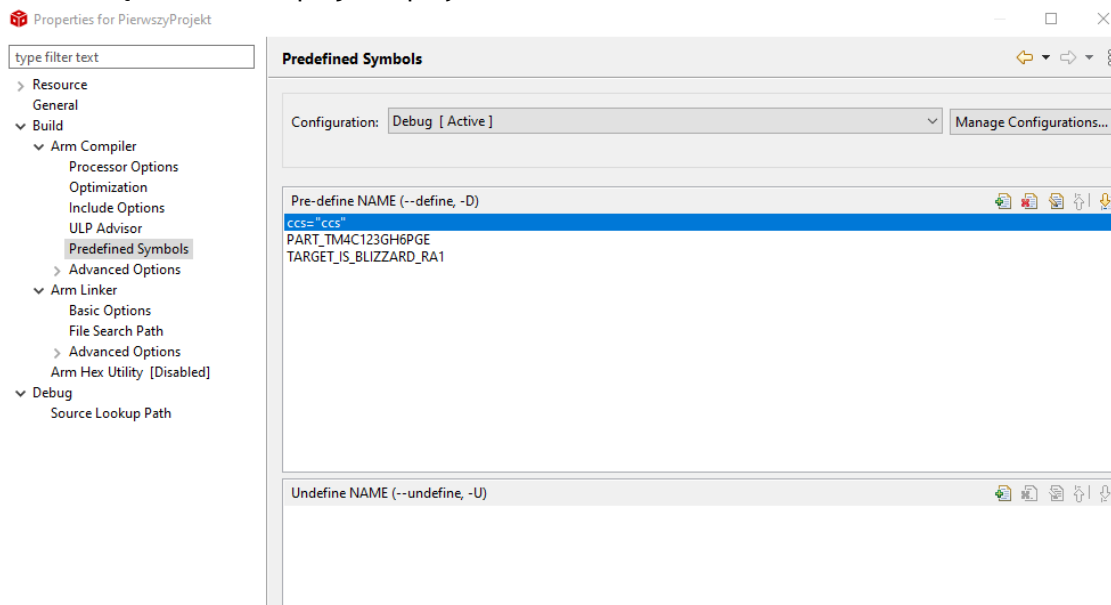
Ponownie nacisnąć przycisk *Add* i wpisać:

```
PART_TM4C123GH6PGE
```

Ponownie nacisnąć przycisk *Add* i wpisać:

```
TARGET_IS_BLIZZARD_RA1
```

2. Zamknąć właściwości projektu przyciskiem OK.

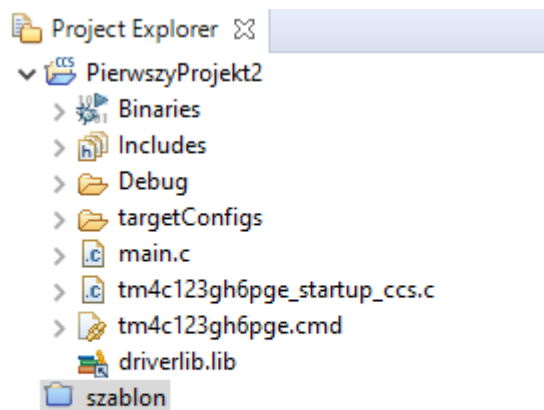


Rys. 16. Definiowanie kompilatora i urządzenia.

Utworzony projekt będzie służył jako szablon do realizacji zadań. Pozwoli to na uniknięcie kłopotu z dostosowywaniem ustawień przy tworzeniu każdego nowego projektu. W celu utworzenia nowego projektu na podstawie szablonu należy:

1. Skopiować szablon klikając na niego prawym przyciskiem myszy i wybierając polecenie *Copy* lub przez kombinację klawiszy Ctrl+C
2. Wkleić szablon w oknie Project Explorer poprzez kliknięcie prawym przyciskiem myszy i wybranie polecenia *Paste* lub przez kombinację klawiszy Ctrl+V.
3. W oknie dialogowym wpisać nową nazwę projektu.

W sytuacji posiadania kilku projektów jednocześnie w przestrzeni roboczej, zaleca się, aby nieużywane projektu po prostu zamknąć (po kliknięciu prawym przyciskiem myszy wybrać Close Project). W oknie eksploratora projektów zamknięty projekt oznaczony jest na niebiesko.



Pierwszy program



1. W pliku main.c w nowo utworzonym folderze projektu wpisać kod:

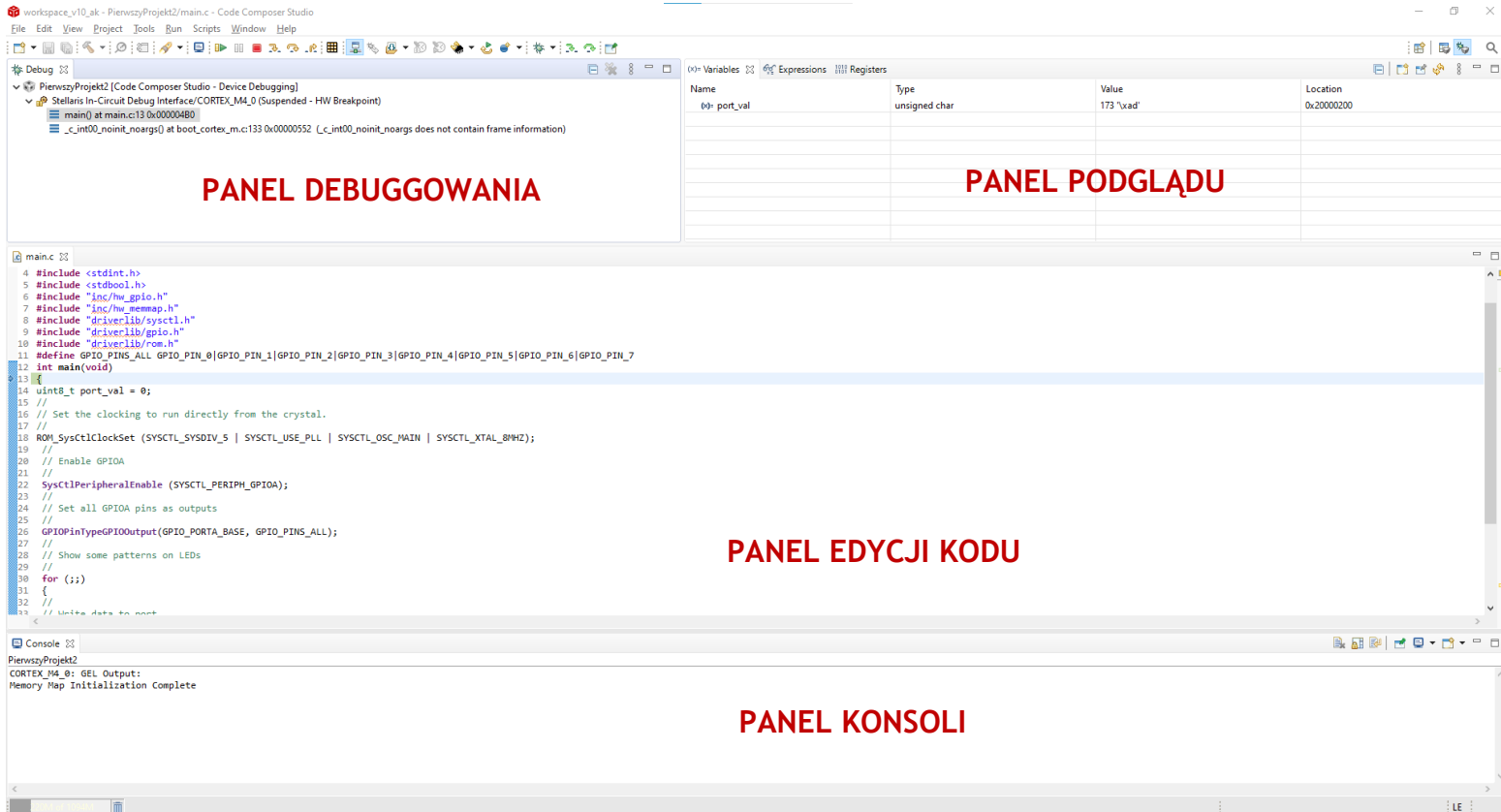
```
1.  /*
2.  main.c
3.  */
4.  #include <stdint.h>
5.  #include <stdbool.h>
6.  #include "inc/hw_gpio.h"
7.  #include "inc/hw_memmap.h"
8.  #include "driverlib/sysctl.h"
9.  #include "driverlib/gpio.h"
10. #include "driverlib/rom.h"
11.
12. #define GPIO_PINS_ALL
13.     GPIO_PIN_0|GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3|GPIO_PIN_4|GPIO_PIN_5|GPIO_PIN_6|GPIO_PIN_7
14.
15. int main(void)
16. {
17.     uint8_t port_val = 0;
18.
19.     //
20.     // Set the clocking to run directly from the crystal.
21.     //
22.     ROM_SysCtlClockSet (SYSCTL_SYSDIV_5 | SYSCTL_USE_PLL | SYSCTL_OSC_MAIN | SYSCTL_XTAL_8MHZ);
23.
24.     //
25.     // Enable GPIOA
26.     //
27.     SysCtlPeripheralEnable (SYSCTL_PERIPH_GPIOA);
28.
29.     //
30.     // Set all GPIOA pins as outputs
31.     //
32.     GPIOPinTypeGPIOOutput(GPIO_PORTA_BASE, GPIO_PINS_ALL);
33.
34.     //
35.     // Show some patterns on LEDs
36.     //
37.     for (;;)
38.     {
39.         //
40.         // Write data to port
41.         //
42.         GPIOPinWrite (GPIO_PORTA_BASE, 0xFF, port_val++);
43.         //
44.         // Delay for a while so changes can be visible
45.         //
46.         SysCtlDelay(SysCtlClockGet() / 2);
47.     }
48.     return 0;
49. }
```

Program ten zapala w zależności od wartości zmiennej port_val odpowiednie diody na porcie A. Najważniejsze elementy programu to:


- Linia 18 - ustawienie zegara z wykorzystaniem wbudowanej w ROM mikrokontrolera funkcji
- Linia 22 - włączenie portu GPIO A
- Linia 26 - ustawienie typu wszystkich nóżek portu GPIO A na wyjścia
- Linia 35 - wpisane na wybrane przez drugi argument funkcji nóżki portu A wartości zmiennej port_val
- Linia 39 - wprowadzenie opóźnienia pomiędzy kolejnymi iteracjami pętli.

Dokładny opis użytych w programie funkcji można znaleźć w [4]. Funkcje te zostaną również omówione w trakcie następnych zajęć.


2. Włączyć płytę EasyMxPRO i ustawić przełącznik SW15.1 na pozycję włączoną.
3. W programie CCS nacisnąć przycisk *Build*  w celu weryfikacji poprawności kodu.
4. Jeżeli operacja Build powiodła się, nacisnąć przycisk *Debug* . W wyniku program zostanie uruchomiony i zmiana ulegnie perspektywa z perspektywy edycji na perspektywę debuggowania (rys. 17), a program zatrzyma się na pierwszej linii funkcji main.
5. Zapoznaj się z perspektywą debuggowania.



Rys. 17. Widok perspektywy debuggowania

6. By kontynuować działanie programu, należy nacisnąć przycisk *Resume*  lub użyć przycisków analizy krok po kroku



7. Zweryfikować działanie programu na płycie EasyMxPRO.
8. Ustawić „Breakpoint” w programie na linii GPIOPinWrite poprzez dwukrotne kliknięcie na pasku na lewo od numeru linii. Jeżeli breakpoint został poprawnie dodany na pasku powinna się pojawić ikona .
9. Uruchomić program i nacisnąć przycisk *Resume*.
10. Program powinien zatrzymać się na ustawiony Breakpoint.


UWAGA

Wykorzystywany sterownik ICD1 nie pozwala na dodawanie lub usuwanie breakpointów podczas pracy procesora.

11. Kliknij w przycisk *Resume* kilka razy i zaobserwuj zmiany na płycie EasyMxPro. Nie wyłączaj programu.

Podgląd zmiennych, pamięci oraz rejestrów

Użytkownik może podglądać zarówno zawartość zmiennych użytych w programie jak i zawartość pamięci mikroprocesora oraz rejestrów. W celu podglądu zmiennych wykonaj następujące operacje:

1. Przejdź do zakładki *Expressions* w panelu Podglądu w perspektywie debuggowania.
2. Odnajdź zmienną `port_val`. Zaznacz ją i kliknij prawym przyciskiem myszy. W menu wybierz *Add Watch Expression*, a następnie w oknie dialogowym kliknij OK. W zakładce *Expressions* powinien pojawić się podgląd zmiennej `port_val`.
3. Zmienna `port_val` jest typu `uint8_t` należy więc zmienić format wyświetlania na liczbowy. W tym celu kliknij prawym przyciskiem myszy na wiersz w zakładce *Expressions* odpowiadający zmiennej `port_val`. Z menu wybierz *Number Format->Decimal*.
4. Klikając w przycisk *Resume* i zatrzymując się na ustawionym breakpointie zaobserwuj zmiany wartości zmiennej `port_val`.
5. Zakończ działanie programu naciskając przycisk . CCS powinno automatycznie przełączyć się na perspektywę edycji.

UWAGI DOTYCZĄCE PERSPEKTYW

Zarówno perspektywę edycji jak i perspektywę debuggowania można dowolnie modyfikować oraz dodawać nowe elementy poprzez menu *Window*. W wyniku modyfikacji często zdarza się, że programista nie może odnaleźć wcześniej wykorzystywanego elementu. Może on wtedy przywrócić stan perspektywy na domyślny poprzez wybranie polecenia *Window->Reset Perspective*.