

Koło Naukowe Robotyków KoNaR

HC12 I CODEWARRIOR
NIE ZA KRÓTKIE WPROWADZENIE DO TEMATU

Politechnika Wrocławska
Wydział Elektroniki

Na potrzeby zajęć
Systemy Mikroprocesorowe w Automatyce

Autor:

MATEUSZ CHOLEWIŃSKI

15 czerwca 2009

Spis treści

1	Wprowadzenie	2
1.1	Mikrokontroler HC12	2
1.2	Moduł HC12 i płytką testową	3
1.3	Interfejs BDM	3
1.4	Code Warrior	3
1.4.1	Wiadomości ogólne	3
1.4.2	Instalacja	3
1.4.3	Processor Expert i Device Initialization	4
2	Projekt	5
2.1	Tworzenie i parametry projektu	5
2.2	Wygląd środowiska	5
3	Przykładowe implementacje	8
3.1	Świecąca dioda	8
3.2	Przerwanie cykliczne	9
3.3	Przetwornik ADC	10
3.4	Generator sygnału PWM	11
4	Podsumowanie	14

Rozdział 1

Wprowadzenie

1.1 Mikrokontroler HC12

HC12 jest szesnastobitowym mikrokontrolerem wyprodukowanym przez Freescale Semiconductors [1]. Został zaprojektowany i po raz pierwszy wyprodukowany w połowie lat 90-tych. Jest naturalnym rozwinięciem modelu HC11 i warto nadmienić, iż wykazuje kompatybilność z programami napisanymi na HC11. Główne różnice to parę dodatkowych instrukcji, szybszy zegar i większa ilość pamięci.

Poniżej przedstawiono najistotniejsze cechy wyróżniające mikrokontroler HC9S12A64 [2]:

- 16 bitowa jednostka centralna CPU12
- 64 kb pamięci FLASH
- 4 kb pamięci RAM
- 1 kb pamięci EEPROM
- 8 kanałowy, 10-o bitowy przetwornik ADC
- 16-o bitowy, 8-o kanałowy timer
- modulator PWM - 7 kanałów 8-o bitowych.
- interfejsy SCI (szeregowy asynchroniczny), SPI (szeregowy synchroniczny), magistrala I^2C
- interfejs BDM (Background Debug Mode)
- wbudowane instrukcje FuzzLogic

Z punktu widzenia użytkownika i programowania urządzenia, najpotężniejszym narzędziem jest interfejs BDM. Za jego pomocą możemy, w czasie trwania programu, podglądać wartości zmiennych, a co więcej, nawet je zmieniać. Pozwala to na łatwe i szybkie odnajdowanie błędów w napisanym programie.

Należy także wspomnieć o dużej ilości portów, co daje dużą elastyczność i ogromne możliwości. Możemy wykorzystać wyświetlacz LCD, generować sygnały PWM, świecić linijką diodową, połączyć się z komputerem, a mimo tego pozostaje jeszcze wiele niewykorzystanych wejść/wyjść.

1.2 Moduł HC12 i płytką testowa

Podczas demonstracji wykorzystywany był moduł z mikrokontrolerem HC12, zaprojektowany przez dra Marka Kabałę i dra Marka Wnuka[4]. Sam moduł stanowi autonomiczną część, która (po zaprogramowaniu) wymaga tylko 5V zasilania.

Do nauki programowania i testowania aplikacji wykonano płytkę, opartą o płytkę uniwersalną.

Płytką zawiera:

- sekcję zasilania, opartą o mostek prostowniczy (zapobiega błędnej polaryzacji wtyczki zasilania) i stabilizator LM7805; dodatkowo napięcie +5V jak i masa zostały wyciągnięte na igły,
- sekcję komunikacji, opartą o układ MAX232N,
- zestaw diód LED,
- 2 potencjometry do testowania przetwornika ADC i do wysterowania kontrastu wyświetlacza LCD,
- złącze wyświetlacza LCD stowarzyszone z portem A.

Dodatkowo PortA i PortB mikrokontrolera został wyciągnięty na igły. Takie rozwiązanie pozwala na szybkie łączenie i modyfikację połączeń między mikrokontrolerami i innymi urządzeniami.

1.3 Interfejs BDM

Do debugowania i programowania wykorzystywany jest interfejs TBDML autorstwa Roberta Kuczaja. Podłączany jest do komputera za pomocą interfejsu USB. Jedną z ważniejszych zalet takiego rozwiązania, jest możliwość zasilania modułu wprost z programatora (do 500 mA w standardowym porcie, teoretycznie 3A w Power USB). Więcej w [3]

1.4 Code Warrior

1.4.1 Wiadomości ogólne

CodeWarrior jest zintegrowanym środowiskiem programistycznym stworzonym przez firmę Metrowerks. Dziś rozwija je i sprzedaje Freescale Semiconductors. Sama nazwa pochodzi z podtytułu "Mad Max 2: The Road Warrior". Należy zauważyć że jest bardzo trafna.

Pakiet zawiera wiele pomocnych narzędzi - podczas tworzenia nowego projektu jesteśmy zasypywani różnymi propozycjami. Jednocześnie sam nie jest wolny od błędów - pozostawianie przezroczystych pasków i artefakty to coś normalnego.

1.4.2 Instalacja

Tak naprawdę istnieje wiele wersji Code Warrior'ów. Należy zawsze pamiętać, by ściągnąć ten odpowiedni. W przypadku naszego mikrokontrolera musimy ściągnąć... Code Warrior dla HC12. Sama instalacja przebiega jak w większości programów na Windows. Niestety, darmowa jest tylko wersja dla Windows - wersja na Linuxa jest płatna. Można jednak uruchomić CW w wirtualnej maszynie, choć nie wiadomo jak będzie np. z RS.

1.4.3 Processor Expert i Device Initialization

Code Warrior nie byłby tak niezwykłym środowiskiem, gdyby nie Processor Expert (PE) i Device Initialization (DI). Są to nakładki, ułatwiające pisanie programów. Ułatwienie w PE sprowadza się do wybierania myszką interesującego nas peryferium, ustawienia opcji i przeciągnięcia požądanej funkcji danego peryferium do pliku projektu. Device Initialization pozwala tylko na łatwą inicjalizację peryferiów - wszystkie funkcje dostarczane przez PE musimy już sami napisać.

PE pozwala nam na nie pamiętanie do czego służy dany rejestr lub co trzeba robić po kolei, aby uruchomić przetwornik ADC. Pewna znajomość rejestrów i zasad postępowania jest już potrzebna przy wykorzystywaniu DI. Obie nakładki wprowadzają niesamowity komfort pracy. Dla początkujących najlepszy jest PE (można zawsze podejrzeć co tak naprawdę robi funkcja wygenerowana w PE), dlatego też właśnie jemu będzie poświęcony ten raport.

Należy jednakże mieć świadomość, że cedowanie wszystkich aspektów na automatyczne narzędzie, pociąga za sobą skutki, mogące być dosyć nieprzyjemne. PE w generowanych plikach wstawia komentarze **Write your code here** lub **Write local variables definitions here**. Należy się mocno trzymać tych wskazówek, gdyż umieszczenie kodu powyżej tego zapisku skutkuje utratą tegoż kodu podczas kompilacji.

Również czasem nie jest zaznaczone, gdzie można pisać. Jednakże po pewnym czasie nabiera się wyczucia, co gdzie można umieścić.

Kod generowany przez PE nie zawsze jest optymalny lub nawet dobry. Osobiście miałem przypadki, gdy funkcja nie działa poprawnie i dopiero wstawienie pustej linii przed lub warunku *if(1)* pomagało. Należy zawsze o tym pamiętać, szczególnie podczas pisania dużych projektów.

Rozdział 2

Projekt

2.1 Tworzenie i parametry projektu

Po instalacji i uruchomieniu Code Warriora wyskakuje monit przedstawiony na rysunku 2.1.

Wybieramy opcję **Create New Project**. Czekają nas teraz szereg pytań, na które odpowiadamy domyślnie, poza następującymi:

- pytanie o Processor Expert (rysunek 2.2); odpowiadamy **Yes**,
- wybór urządzenia łączącego mikrokontroler z CW (rysunek 2.3); wybieramy tbdml jeśli wykorzystujemy interfejs zaprojektowany przez Roberta Kuczaję,
- ostatnie pytanie dotyczy obudowy mikrokontrolera; wykorzystując opisywany powyżej moduł, wybieramy wersję 80-o nóżkową (rysunek 2.4).

Po ustawieniu wszystkich opcji, CW wygeneruje niezbędne pliki i przygotuje dla nas środowisko.

2.2 Wygląd środowiska

Na rysunku 2.5 przedstawiony jest wygląd przykładowego, pustego jeszcze projektu. W lewej części zebrane są najważniejsze elementy naszego projektu. Na niebiesko zaznaczony jest przycisk **Make**. Z jego pomocą kompilujemy nasz projekt. Na czerwono oznaczony jest przycisk który powoduje kompilację, wysłanie programu i przejście w tryb debugowania. Na początku należy kliknąć przycisk **Make**; stworzony zostanie plik o takiej samej nazwie jak nasz projekt i plik **Events.c**, który zawiera obsługę przerw. Wszystkie te pliki pojawiają się w polu zaznaczonym seledynową ramką.

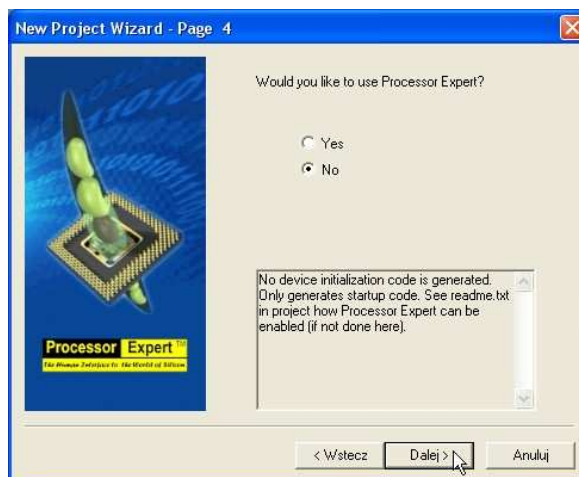
Środkowe okno zawiera ustawienia samego mikrokontrolera. W polu **Clock settings -> Input Clock -> Clock frequency** ustawiamy prędkość mikrokontrolera. Prędkość wewnętrznej szyny danych jest dwukrotnie mniejsza. W polu **Clock settings -> Input Clock -> Clock type** ustawiamy źródło zegara mikrokontrolera.

W prawym górnym rogu znajduje się rysunek poglądowy mikrokontrolera. Jeśli powiększymy okno to pokażą się opisy każdego pinu wraz z funkcją jaką aktualnie pełni.

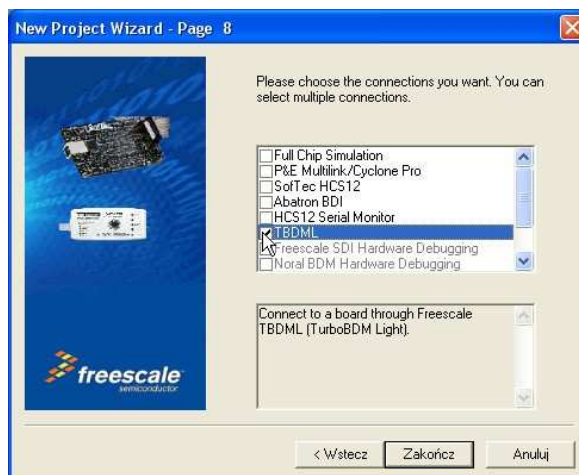
W prawym dolnym rogu, w oknie **Bean Selector** przedstawione są wszystkie peryferia mikrokontrolera. Wybieramy z menu pożądaną funkcję i klikamy dwukrotnie. Funkcja i odpowiednie pliki zostają dodane do projektu.



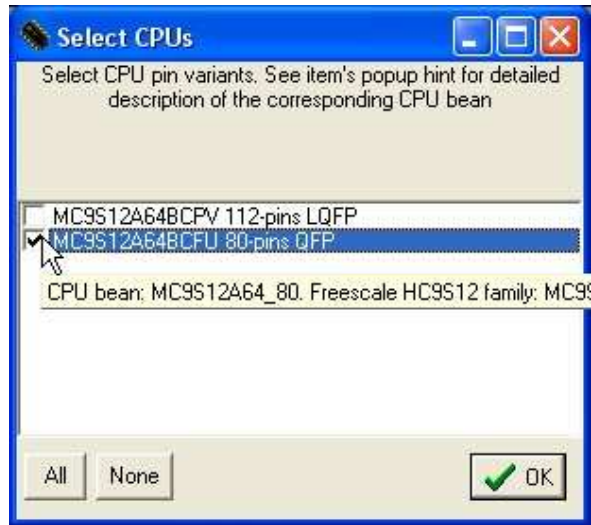
Rysunek 2.1: Ekran powitalny



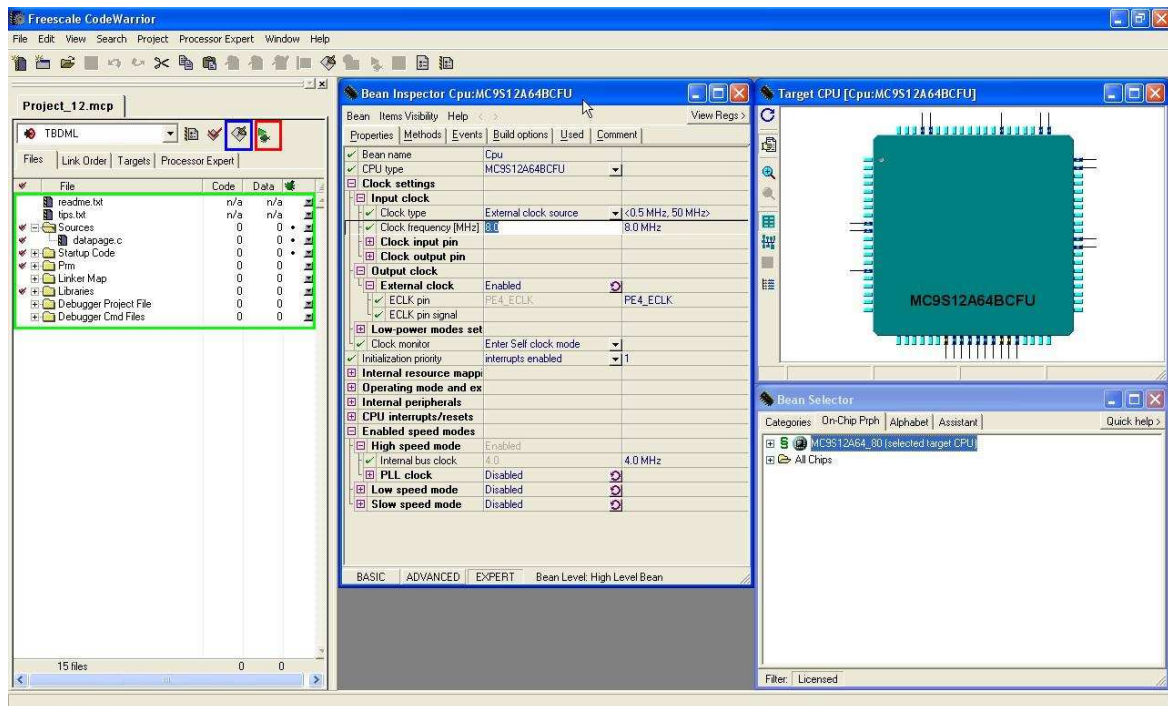
Rysunek 2.2: Processor Expert



Rysunek 2.3: Urządzenie łączące mikrokontroler z PC



Rysunek 2.4: Wybór opakowania



Rysunek 2.5: Menu główne tuż po ustawieniach

Rozdział 3

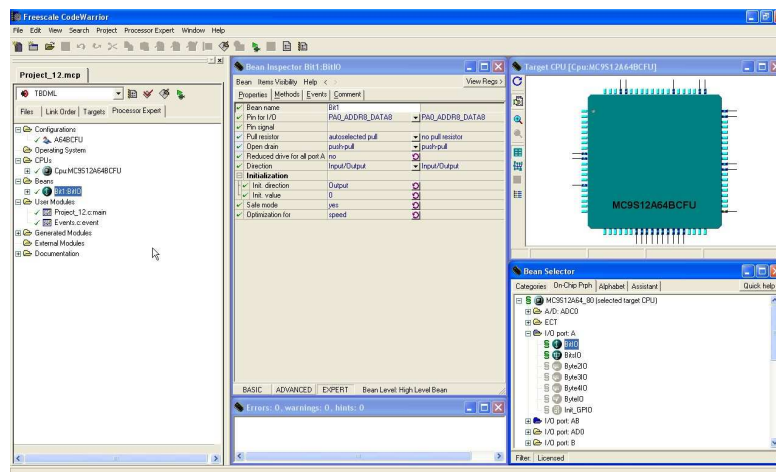
Przykładowe implementacje

3.1 Świecąca dioda

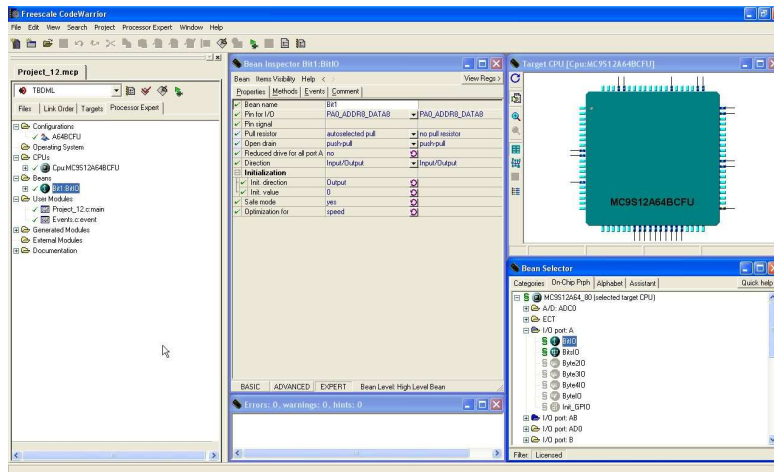
Przed przystąpieniem do pracy musimy zadbać, aby jedna z nóżek diody była podłączona do wybranej przez nas nóżki mikrokontrolera (należy pamiętać o wstawieniu rezystora ograniczającego, np 220Ω). Musimy także przyjąć konwencję jakim stanem świecimy: wysokim czy niskim. Dobrym zwyczajem jest uruchamianiem urządzeń stanem niskim. Większość mikrokontrolerów zaraz po włączeniu wystawia na swoje porty logiczne 1, co w przypadku doń podłączonego silnika skutkuje natychmiastowym, niekontrolowanym uruchomieniem urządzenia. Należy wspomnieć, iż CW pozwala na ustawienie początkowego stanu danej nóżki.

Załóżmy, że dioda podłączona jest do portu A, do wyjścia 0. Wybieramy w **Bean Selector I/O port: A**. W menu widać poszczególne beany. Mamy do wyboru kontrolowanie pojedynczego bita, kilka bitów jak i całe bajty. Wybieramy **BitIO** i ustawiamy opcje tak jak na rysunku 3.1. Bean jest tak ustawiony, że po wgraniu programu na mikrokontroler, dioda od razu zaświeci.

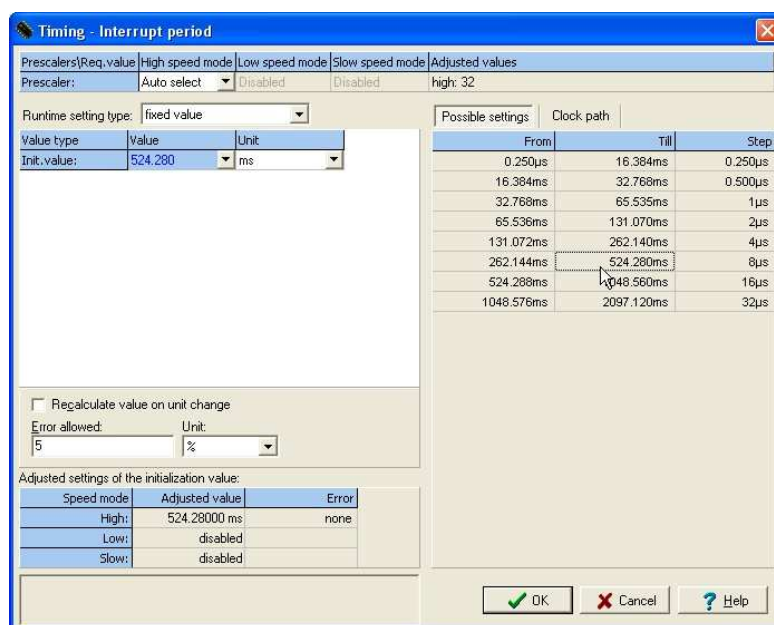
Jeśli teraz klikniemy na przycisk oznaczony plusem przy **Bit1:BitIO** w oknie po lewej stronie, to rozwinie się menu funkcji w jakie wyposażony jest nasz bean. Po najechaniu kursorem myszki na dowolną funkcję, uzyskamy podpowiedź co dana funkcja robi. Odznaczmy **NegVal** - funkcja ta pozwala na zanegowanie stanu na porcie. Będzie przydatna przy obsłudze przezwania cyklicznego. Wróćmy do opcji i ustawmy wartość początkową na 1 - po uruchomieniu, dioda nie będzie świecić. Aby teraz zmusić ją do pracy, należy w oknie po lewej stronie, w menu **User Modules** wybrać plik **Project_x.c:main**, gdzie x to numer naszego projektu. Jest to plik główny projektu. Odnajdujemy komentarz **Write your code here**. Powracamy w lewym



Rysunek 3.1: Wybór beana



Rysunek 3.2: Bean przerwania cyklicznego

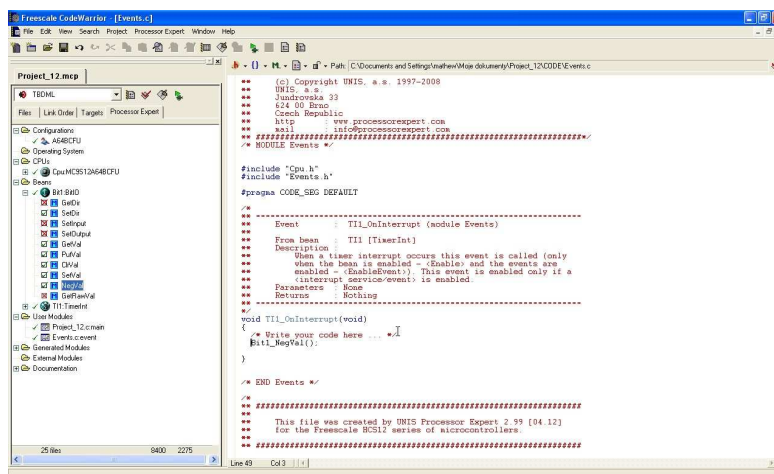


Rysunek 3.3: Ustawienia przerwania cyklicznego

oknie do naszego beana **Bit1:BitIO**, rozwijamy menu i odnajdujemy **ClrVal**. Przeciągamy tą funkcję do głównego pliku projektu w miejsce, o którym wcześniej wspomniałem i... naciskamy przycisk z zieloną strzałką (kompilacja i wgranie programu). Pojawi się wiele okien, na pytanie o wymazanie pamięci odpowiadamy twierdząco zaznaczając opcję "Nie pytaj mnie więcej". Gdy już wszystko się "ustabilizuje" klikamy przycisk z zieloną strzałką i cieszymy się widokiem świecącej diody.

3.2 Przerwanie cykliczne

W wielu projektach niezbędne jest wykorzystanie przerwania cyklicznego. Odpowiedni timer w mikrokontrolerze zostaje obciążony zadaniem liczenia do ustalonej wartości. Gdy już do niej doliczy zgłasza przerwanie, mikrokontroler przerywa wszelkie, aktualne zajęcia, magazynuje dane na stosie i zajmuje się obsługą przerwania. Po zakończeniu przerwania ściąga wszystkie dane ze stosu i kontynuuje poprzednie zadanie; timer zostaje wyrestowany i zaczyna liczyć od początku.



Rysunek 3.4: Plik przerwań - Events.c

Aby zaimplementować przerwanie cykliczne należy z **Bean Selector** wybrać zakładkę **ECT**, a następnie **TimerInt**. Po dwukrotnym kliknięciu na beana, pojawią się jego opcje. Należy wybrać czas, co jaki ma występować przerwanie. Jest to pokazane na rysunku 3.3. Należy teraz skompilować projekt. Następnie w zakładce **User Modules** wybieramy plik **Events.c** (3.4). Zawiera on funkcje obsługi przerwań. W opcjach beana przerwania zdefiniowana jest nazwa przerwania, jakie timer generuje. Nazwę tą odnajdujemy w pliku **Events.c** i poniżej znanego komentarza **Write your code here** przeciągamy funkcję **NegValue** z beana obsługi diody. Kompilujemy program i wgrywamy na mikrokontroler. Uruchamiamy debugowanie i oto płytka już do nas mruga.

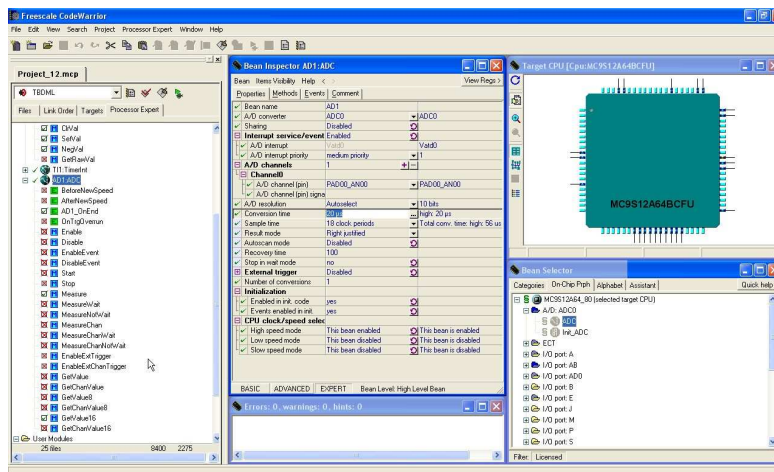
3.3 Przetwornik ADC

Przetwornik ADC służy do pomiaru napięcia. Sygnał analogowy napięcia jest przetwarzany na cyfrowy o wartości od 0 do 1023 (dla 10-bitowego przetwornika). W naszym przypadku 0 to 0V, 1023 to 5V. Przed przystąpieniem do programowania należy zadbać aby pin VRL był zwarty z pinem VSS(GND), a VRH z VDD(+5V). Pin VRH to “górną” granicą napięcia jakie chcemy badać, a VRL to “dolną” granicą.

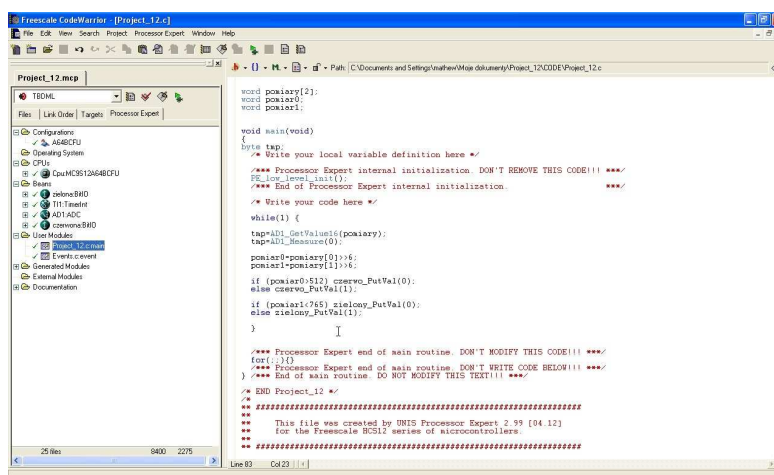
Z menu **Bean Selector**, zgodnie z rysunkiem 3.5 wybieramy przetwornik i ustawiamy ile kanałów chcemy wykorzystywać. Musimy także ustalić czas konwersji. W kodzie, przedstawionym później, założona jest obsługa 2-óch kanałów; czas konwersji można przyjąć $20\mu\text{s}$, gdyż gwarantuje to wynik nieobarczony błędem związanym ze zbyt krótkim czasem konwertowania wartości. Ważna jest również opcja **Result mode**. Mówi ona czy wynik ma być wyrównany do bardziej znaczących czy do mniej znaczących bitów. Ustawiamy wersję **right**.

Do otrzymywania wyników pomiaru wykorzystujemy funkcję **GetValueX**, gdzie X oznacza 8 lub 16 bitów. Należy zadbać, aby zmienna do której pobieramy wynik, miała odpowiednią wielkość. Na rysunku 3.6 zaproponowana jest zmienna typu word. Jeśli przeciągnęliśmy funkcję **GetValueX** do głównego pliku projektu, to po najechaniu na nią wskaźnikiem i kliknięciu prawym przyciskiem, mamy możliwość podejrzenia deklaracji funkcji. Nie trzeba chyba wspominać jak bardzo przydaje się ta opcja. Proponuję przepisać kod z 3.6, aby łatwiej było zrozumieć metodę działania przetwornika i uzyskiwania wartości. Dodatkowo w kodzie zaimplementowano załączanie się diód przy odpowiednich wartościach mierzonego napięcia. Jeśli mamy uruchomione przerwanie cykliczne, w którym migamy diodą, należy odpowiednią liniijkę zakomentować w celu wyeliminowania dziwnego zachowania diody.

Interfejs BDM daje możliwość podglądania wartości zmiennych w czasie działania programu.



Rysunek 3.5: Bean przetwornika analog-cyfra



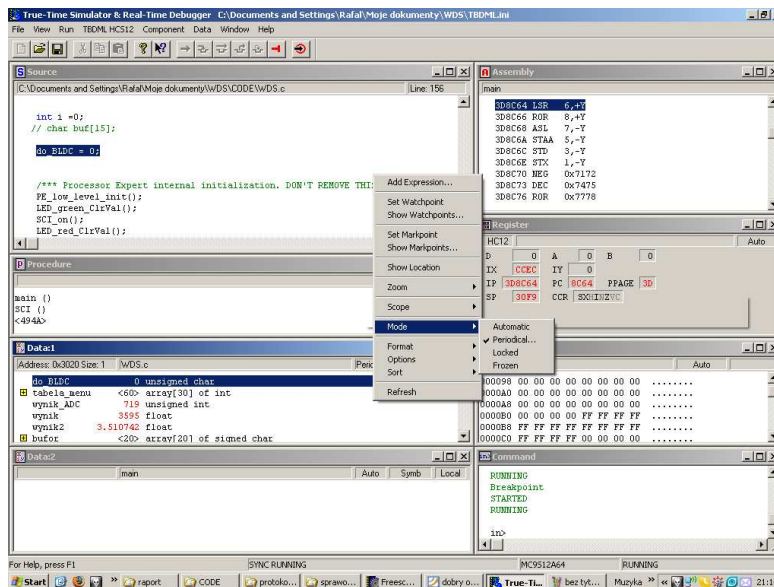
Rysunek 3.6: Programik do przetwornika ADC

Proponuję zmienne przechowujące zmierzone wartości uczynić globalnymi. Po wgraniu zaproponowanego programu przechodzimy do opcji debugowania. W oknie (3.7 - pochodzi z innego projektu) **Data 1** mamy wypisane wszelkie zmienne globalne, wraz z wartościami. Klikając prawym przyciskiem myszy i wybierając **Mode -> Periodical** ustawiamy jak często odświeżają się wartości zmiennych w tym oknie. Aplikacja na przetwornik została napisana, aby obsłużyć dalmierz laserowy Sharp. Z pomocą dokumentacji sensora, możemy dobrać tak progi, że czerwona dioda świeci się w odległości 30 cm lub więcej, zielona - 50 cm itd.

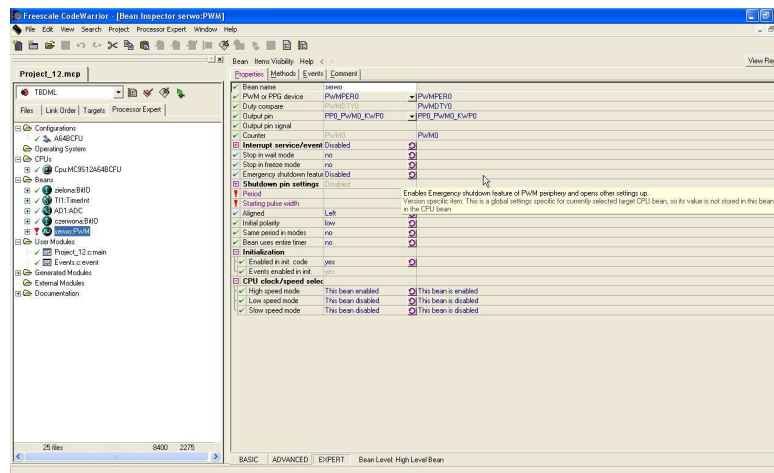
3.4 Generator sygnału PWM

Mikrokontroler jest wyposażony w sprzętowy generator sygnałów PWM. Za pomocą tegoż sygnału możemy sterować serwomechanizmem, lub silnikiem poprzez mostek H. PWM można scharakteryzować dwoma parametrami: okresem trwania i długością stanu wysokiego. Więcej informacji można znaleźć w internecie

Aby wykorzystać generator PWM, należy dodać odpowiedniego beana. Znajdziemy go w **Bean Selector** w sekcji **PWM**. Po dodaniu ukazuje się okno przedstawione na 3.8. Widać wykrzykniki przy polach **Period** i **Starting Pulse Width**. Standardowe serwomechanizmy zawierają zintegrowane mostki, działające dobrze na częstotliwości ok 50Hz. Wybieramy częstotliwość najbliższą tej wartości. Sterując wypełnieniem sygnału (stosunkiem długości czasu stanu wyso-



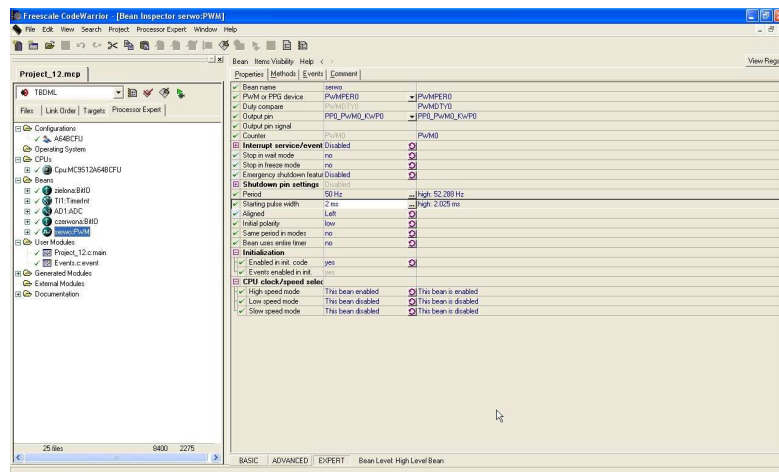
Rysunek 3.7: Opcje podglądania zmiennych



Rysunek 3.8: Bean generatora PWM

kiego do trwania całego okresu) powodujemy obrót w lewo, w prawo lub po prostu utrzymywanie pozycji przez serwo. Należy też pamiętać o ustawieniu odpowiedniego pinu jako wyjścia generatora PWM.

Na rysunku 3.9 przedstawiony jest przykład konfiguracji.



Rysunek 3.9: Ustawienia dla standardowego serwo mechanizmu

Rozdział 4

Podsumowanie

Niniejszy dokument powstał w ramach seminarium z Systemów Mikroprocesorowych w Automatyce. Jest on opisem demonstracji, przeprowadzonej na zajęciach seminaryjnych z przedmiotu Systemy Mikroprocesorowe w Automatyce. Dodatkowo ma być dokumentem pozwalającym na szybkie i łatwe wprowadzenie do programowania mikrokontrolerów firmy Freescale Semiconductors.

W dokumencie omówiono podstawowe aplikacje na mikrokontrolerze; internet pełen jest rozwiązań jak np. poradzić sobie z wyświetlaczem LCD. Przede wszystkim: warto eksperymentować i wymyślać sobie coraz to nowsze zadania.

Powodzenia!

Bibliografia

- [1] www.freescale.com
- [2] http://www.datasheetcatalog.com/datasheets_pdf/M/C/9/S/MC9S12A64.shtml
- [3] http://rab.ict.pwr.wroc.pl/mw/pdfs/osbdm_tbdml.pdf
- [4] <http://www.konar.ict.pwr.wroc.pl/module.php?op=download&cmd=click&id=112>