Laboratorium Robotyki

CW-HC08 *
Programowanie mikrokontrolera MC9S08QD4

Jan Kędzierski
Marek Wnuk

Wrocław 2009

*Dokument stanowi instrukcję do ćwiczenia w ramach kursu *Systemy mikroprocesorowe w automatyce II*.

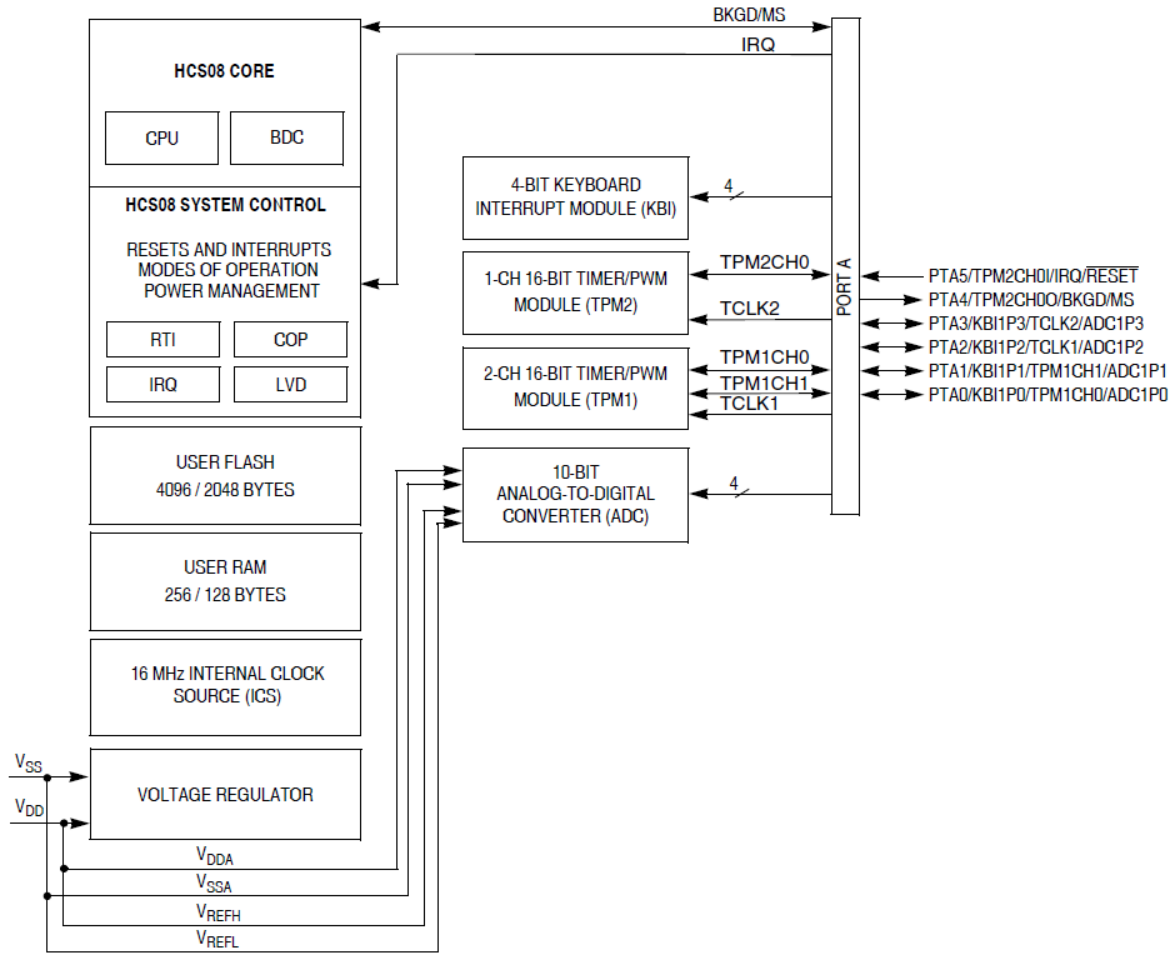
Spis treści

1 Cel ćwiczenia

Celem ćwiczenia jest ogólne zapoznanie się z 8-bitowym mikrokontrolerem z rodziny HCS08 oraz ze środowiskiem programistycznym CodeWarrior Development Studio firmy Freescale Semiconductors [1]. W ramach ćwiczenia należy wygenerować przy pomocy narzędzia Processor Expert [4] kod inicjujący układ, następnie uzupełnić go tak, aby układ realizował odpowiednie funkcje proponowane w niniejszej instrukcji.

2 Wprowadzenie

Układ MC9S08QD4 [2] należy do tanich, wydajnych, 8-bitowych mikrokontrolerów rodziny HCS08. Układy te są dostępne w różnych obudowach, z kilkoma rodzajami pamięci o różnych wielkościach. Programowanie i uruchamianie mikrokontrolera wykonuje się przy pomocy interfejsu BDM (Background Debug Module). Wbudowany w układ system debagowania pozwala na śledzenie pracy układu instrukcja po instrukcji. Diagram blokowy[†]mikrokontrolera przedstawiono na rysunku 1.



Rysunek 1: Diagram blokowy MC9S08QD4

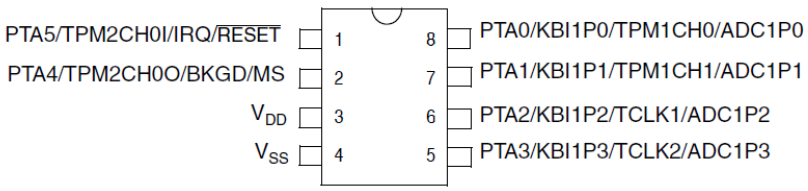
Własności mikrokontrolera:

- 8-bitowa jednostka centralna CPU :

[†]Diagram zaczerpnięto z dokumentacji producenta [2]

  - częstotliwość pracy zegara: 16MHz,
  - zestaw instrukcji z HC08 wzbogacony o instrukcje BGND,
  - system debugowania,
- zasoby pamięciowe:
  - rozmiar pamięci flash: 4096 bajty,
  - rozmiar pamięci RAM: 256 bajty,
- tryby pracy z oszczędzaniem energii,
- wewnętrzne źródło zegara (ICS),
- wbudowane systemy zabezpieczeń:
  - COP watchdog (Computer Operates Properly),
  - wykrycie spadku napięcia zasilającego,
  - wykrywanie nielegalnych instrukcji,
  - wykrywanie nielegalnych adresów,
  - blok zabezpieczeń pamięci flash,
- 4-kanałowy, 10-bitowy przetwornik analogowo-cyfrowy z funkcją porównanie (compare),
- 16-bitowy system timerów:
  - 16-bitowy licznik z 7-bitowym preskalerem,
  - 3 kanały (każdy może być IC - Input Capture lub OC - Output Compare),
  - dwa tryby PWM: Edge-Aligned, Center-Aligned,
  - możliwość resetowania licznika modulo,
- linie wej/wyj:
  - układ przerwań (KBI) dedykowany dla klawiatury (4 wejścia),
  - 4 linie wej/wyj,
  - zewnętrzne źródło przerwań (IRQ),
  - jednoliniowy interfejs uruchomieniowy BDM (Background Debug Mode) z pułapkami sprzętowymi,
- zasilanie 2,7V - 5V (max 30mA).

Mikrokontroler do poprawnej pracy nie wymaga żadnych zewnętrznych elementów. Na rysunku 2 przedstawiono opis poszczególnych linii układu[‡].

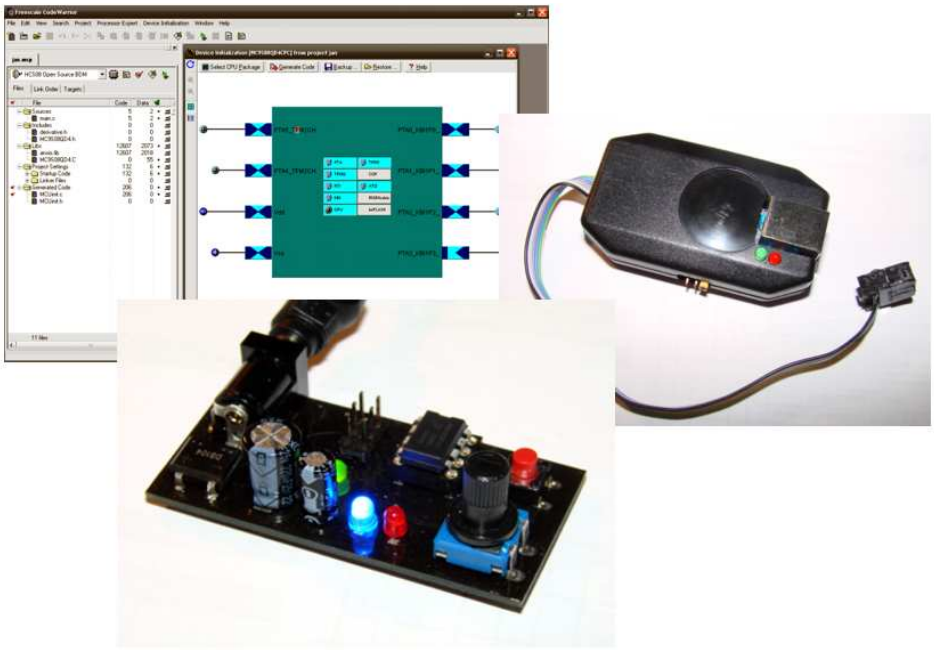


Rysunek 2: Opis linii układu MC9S08QD4

[‡]Opis zaczerpnięto z dokumentacji producenta [2]

3 Opis stanowiska do ćwiczeń

W skład stanowiska do ćwiczeń wchodzą: płytka testowa, interfejs BDM [3], zasilacz, komputer PC wraz z oprogramowaniem [4] oraz instrukcja. Przed przystąpieniem do ćwiczenia należy sprawdzić kompletność stanowiska.



Rysunek 3: Stanowisko do programowania układu MC9S08QD4

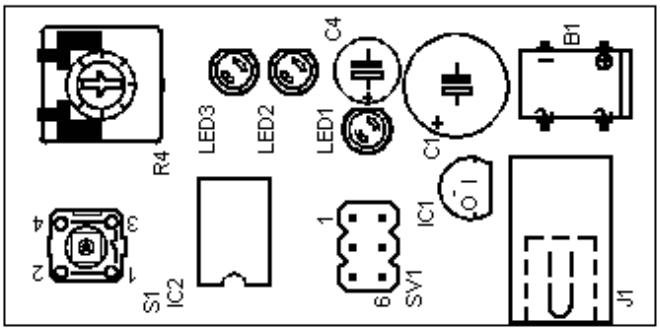
Płytka testowa

Głównym elementem płytki laboratoryjnej jest układ MC9S08QD4. Umieszczono na niej także zasilacz 5V oraz kilka drobnych elementów pozwalających zwizualizować działanie poszczególnych peryferii mikrokontrolera.

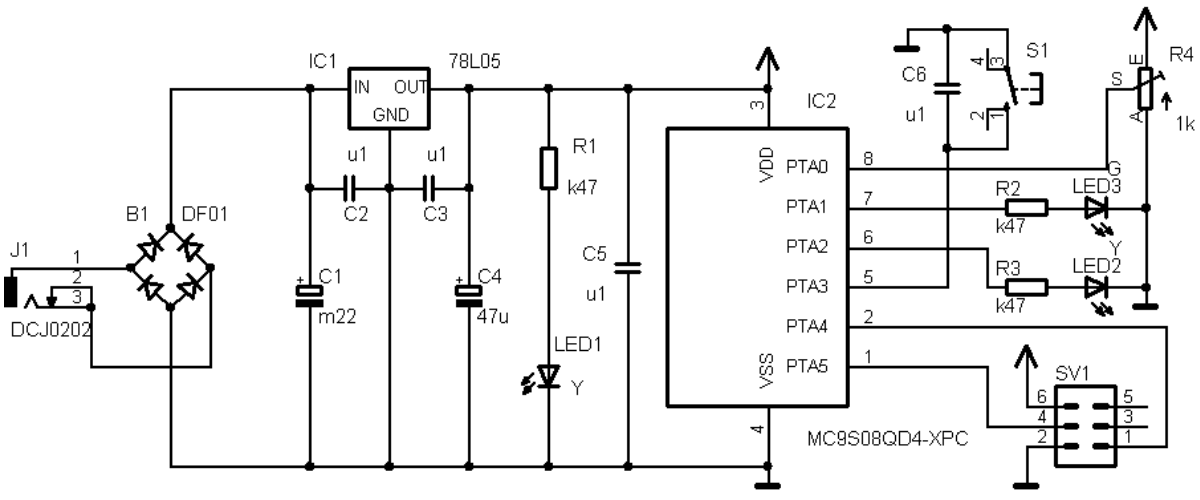
Opis płytki testowej:

- PTA0 - potencjometr 1kΩ,
- PTA1 - dioda LED czerwona,
- PTA2 - dioda LED niebieska,
- PTA3 - klawisz,
- PTA4, PTA5 - interfejs BDM.

Na rysunkach 4 i 5 przedstawiono rozmieszczenie elementów na płytce oraz schemat ideowy płytki.



Rysunek 4: Rozmieszczenie elementów na płytce laboratoryjnej



Rysunek 5: Schemat płytki laboratoryjnej

4 Przebieg ćwiczenia

Przed przystąpieniem do ćwiczenia należy utworzyć podkartotekę o nazwie **EXi_g** (gdzie: g - oznaczenie grupy A..J, i - numer ćwiczenia) i pozostawić w niej (i tylko w niej!) pliki będące wynikiem (lub ilustracją) poszczególnych etapów zajęć. Opis modyfikacji programów i ich działania należy zamieścić jako komentarz w tekstach źródłowych programów. Proszę pamiętać o podaniu nazwisk.

Każdy projekt powinien znajdować się w osobnej podkartotece. Oceniane będą tylko projekty nazwane zgodnie z zrealizowanym zadaniem np. **Zad1, Zad2,...**.

Zad1: Oprogramowaniem CodeWarrior Development Studio

Otrzymany od prowadzącego przykładowy projekt kopiujemy do katalogu Zad1. Następnie uruchamiamy środowisko programistyczne CodeWarrior Development Studio. Otwieramy skopiowany projekt **zad1.mcp**.

W pierwszej kolejności należy wykonać **Make** i uruchomić projekt. Prowadzący wyjaśni, jak powinien działać poprawnie uruchomiony przykładowy projekt. Po sprawdzeniu należy zapoznać się z oknami środowiska programistycznego: główne okno projektu, okno Processor Expert, a po uruchomieniu projektu z debugerem.

Do oceny należy pozostawić projekt z ustawioną pułapką i dodanym komentarzem w miejscu wskazanym przez prowadzącego.

Zad2: Cykliczne przerwanie RTI, moduł obsługi klawiatury KBI

W zadaniu 2 należy utworzyć nowy projekt przy użyciu kreatora. Po zamknięciu poprzedniego projektu i utworzeniu katalogu Zad2 w menu **Plik**, należy wybrać **New Project**. Pojawi się okno kreatora, w którym należy postępować wg instrukcji:

- **Device and Connection:** wybieramy układ *MC9S08QD4* i *HCS08 Open Source BDM*,
- **Project Parameters:** wybieramy *C*, ustawiamy ścieżkę i nazwę projektu,
- **Add Additional Files:** nie dodajemy żadnych plików,
- **Processor Expert:** wybieramy *Device Initialization*,
- **C/C++ Options:** pozostawiamy bez zmian,
- **PC Lint:** wybieramy *No*.

Po wykonaniu powyższych kroków pojawią się podobne okna jak w Zad1. W ćwiczeniu tym, należy skonfigurować dwa moduły. Pierwszy z nich, to cykliczne przerwanie RTI. Częstotliwość przerwania poda prowadzący. Drugi, to moduł obsługujący czteroklawiszową klawiaturę KBI. Aby móc zaprezentować działanie modułów na diodach LED, należy dodatkowo skonfigurować moduł PTA (port A).

Konfiguracja modułu PTA:

- **Settings**
  - **Port control:** wybieramy *Individual pins*,
- **Pins**
  - **Pin*x*:** konfigurujemy te linie, do których podłączono diody LED i wybieramy *Enabled*
    - **Direction:** wybieramy *Output*,

        * **Output value:** wybieramy *1*,
        * **Drive strenght:** wybieramy *High*.

Podobnie konfigurujemy PTA dla drugiej diody. Przed przystąpieniem do dalszej, części zadania można przetestować konfigurację PTA. W tym celu, w oknie **Device initlization** klikamy *Generate code* i zatwierdzamy klikając *Generate*. **Processor Expert** wygeneruje kod odpowiedzialny za poprawną inicjalizację mikrokontrolera. Po zapoznaniu się z wygenerowanym kodem, uruchamiamy *Debuger*.Jeżeli obydwie diody się zapaliły, można przejść do konfiguracji modułów RTI i KBI.

Konfiguracja modułu RTI:

- **Settings**
  - **Clock settings:**
    * **Clock select:** pozostawiamy *Internal oscillator*,
    * **Prescaler:** częstotliwość przerwania wybieramy zgodnie z zaleceniem prowadzącego,
- **Interrupts:**
  - **Real-Time Interrupt:** wybieramy *Enabled*.

Klikamy *Generate code* i zatwierdzamy klikając *Generate*. Konfiguracja RTI jest już zakończona. Naturalnie, aby móc oglądać efekt działania cyklicznego przerwania, należy w pliku źródłowym odszukać funkcję wywoływaną przez przerwanie i zapisać instrukcje, które np. będą zaświecały i gasiły dowolną diodę LED. W funkcji tej potwierdzamy obsłużone przerwanie przy pomocy instrukcji:

```
SRTISC|=(0x40); //ustaw 1 (RTIACK) w rejestrze SRTISC
```

UWAGA: Device initialization nie generuje kodu obsługującego przerwanie. W każdej funkcji wywoływanej przez dane przerwanie, należy „ręcznie" zgasić flagę. W przeciwnym wypadku program nigdy nie opuści przerwania.

Konfiguracja modułu KBI:

- **Settings**
  - **Triggering sensitivity:** wybieramy *edge and level*,
- **Pins**
  - **Pin*x*:** konfigurujemy tę linię, do której podłączono klawisz i wybieramy *Enabled*,
    * **Pull resistor:** wybieramy *pull up*,
- **Interrupts:**
  - **Keyboard request:**
    * **Keyboard request interrupt:** wybieramy *Enabled*.

Ponownie klikamy *Generate code* i zatwierdzamy klikając *Generate*. Konfiguracja KBI jest już zakończona. W pliku źródłowym, należy odszukać funkcję wywoływaną przez przerwanie i zapisać instrukcje, które np. będą zaświecały i gasiły dowolną diodę LED. W funkcji tej, potwierdzamy obsłużone przerwanie przy pomocy instrukcji:

```
KBISC|=(0x04); //ustaw 1 (KBACK) w rejestrze KBISC
```

Do oceny należy pozostawić projekt z komentarzami wyjaśniającymi powyższą konfigurację wszystkich modułów. Komentarze powinny znajdować się na początku każdej funkcji, której one dotyczą.

Zad3: Funkcje czasowo-licznikowe TPMx, przetwornik analogowo cyfrowy ATD

W zadaniu 3 przy pomocy kreatora można utworzyć na nowo projekt lub skopiować dotychczasowy do utworzonego katalogu Zad3. W ćwiczeniu tym należy uruchomić kolejne dwa moduły. Pierwszy z nich konfigurujemy tak, aby pracował jako modulator PWM o częstotliwości i wypełnieniu zadanym przez prowadzącego. Efekt pracy modulatora należy zwizualizować na diodzie LED. Drugi moduł konfigurujemy tak, aby pracował jako 8 - bitowy przetwornik A/C na kanale, do którego podłączono potencjometr. Wartość zmierzoną należy odpowiednio przeskalować i zadać w postaci wypełnienia na wcześniej skonfigurowany kanał PWM. W rezultacie uzyskamy regulację jasności świecenia diody LED przy pomocy potencjometru.

Konfiguracja modułu TPM1:

- **Settings**
  - **Clock settings:**
    - **Clock source select:** pozostawiamy *Fixed system clock*,
    - **Prescaler:** wybieramy na podstawie zadanej częstotliwości,
    - **Modulo counter:** wpisujemy na podstawie zadanej częstotliwości,
  - **Aligned:** pozostawiamy na *left*,
  - **Channels:** ustawiamy *1*,
    - **Channel:**
      - **Capture/compare device:** wybieramy *TPM11*,
      - **Settings:**
      - »**Mode:** wybieramy *PWM*,
      - »» **PWM output action:** wybieramy *Set output on compare*,
      - »» **Channel compare capture:** wpisujemy na podstawie zadanego wypełnienia,
      - **Pin:**
      - »**Channel pin:** wybieramy *PTA1_KBI1P1_TPM1CH1_ADC1P1*.

Przed przystąpieniem do dalszej części zadania, można przetestować konfigurację TPM1.

Konfiguracja modułu ATD:

- **Settings**
  - **Clock settings:**
    - **Input clock select:** pozostawiamy *BusClk*,
    - **Prescaler:** pozostawiamy *1*,
  - **Conversion mode:** wybieramy *Continous conversion*,
  - **Result data format:** pozostawiamy *8-bit right*,
- **Pins**
  - **ADC Input Pins:** klikamy *1*,

        * **Pin:** wybieramy pin, do którego podłączono potencjometr,

- **Initialization**
    - **Initial channel select:** wybieramy kanał, do którego podłączono potencjometr.

Zmierzoną przy pomocy przetwornika A/C wartość podajemy jako wypełnienie sygnału PWM. W tym celu należy np. w programie głównym użyć następujących instrukcji:

```
TPMC1VH=0; // TPMC1V jest 16 bitowy (dwa bajty L i H)
TPMC1VL=ADCRL;
```

Do oceny należy pozostawić projekt z komentarzami wyjaśniającymi powyższą konfigurację wszystkich modułów. Komentarze powinny znajdować się na początku każdej funkcji, której one dotyczą.

Literatura

[1] *www.freescale.com*, Freescale Semiconductors

[2] *MC9S08QD4 Datasheet*, Rev. 4, 9/2008, Freescale Semiconductors

[3] KUCZAJ R. *Interfejs OSBDM/TBDML*, Wrocław 2008, http://www.konar.ict.pwr.wroc.pl/uploads/download/hc12/osbdm_tbdml.pdf

[4] *CodeWarrior Development Studio for Microcontrollers, Quick Start*, Freescale Semiconductors