Laboratorium Robotyki

Programowanie mikrokontrolera MC9S08LL64 w środowisku CodeWarrior*

Jan Kędzierski
Marek Wnuk

ver.1.1
Wrocław 2013

*Dokument stanowi instrukcję do ćwiczenia w ramach kursu *Sterowniki robotów*.

Przebieg ćwiczenia

Celem ćwiczenia jest ogólne zapoznanie się z 8-bitowym mikrokontrolerem z rodziny HCS08 oraz 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 realizował odpowiednie funkcje proponowane w niniejszej instrukcji. Przed przystąpieniem do ćwiczenia, należy zapoznać się z zestawem ćwiczeniowym TWR-S08LL64 [3] w szczególności ze schematem ideowym modułu.

Zad.1 Tworzenie projektu oraz konfiguracja jednostki CPU

Tworzenie nowego projektu

W zadaniu 1 należy utworzyć nowy projekt przy użyciu kreatora. W tym celu wybieramy menu **Plik**, następnie **New Project**. Pojawi się okno kreatora, w którym należy postępować wg poniższych instrukcji:

- **Device and Connection:** wybieramy układ *MC9S08LL64* i *HCS08 FSL Open Source BDM*,
- **Project Parameters:** wybieramy *C*, ustawiamy ścieżkę i nazwę projektu,
- **Add Additional Files:** nie dodajemy żadnych plików,
- **Processor Expert:** wybieramy *Processor Expert*,
- **C/C++ Options:** pozostawiamy bez zmian,
- **PC Lint:** wybieramy *No* i zatwierdzamy klikając Finish.

Konfiguracja modułu CPU

Przed przystąpieniem do konfiguracji CPU, należy zapoznać się z dokumentacją układu [2] w szczególności z modułem ICS (Internal Clock Source) opisanym w 10 rozdziale, na stronie 155. Warto też przyjrzeć się w jaki sposób ICS dystrybuuje sygnały zegara dla poszczególnych peryferii. Opis ten można znaleźć na stronie 29.

Układ generowania zegara może pracować w kilku trybach: FEI, FEE, FBI, FBILP, FBE, FBELP, oraz stop. W ćwiczeniu tym należy skonfigurować ICS do pracy w trybie FEE (FLL Engaged External). Zatem zaprzęgamy do pracy generator DCO (Digitaly Controlled Oscillator) z pętlą FLL (Frequency-locked Loop), który taktowany jest zewnętrznym zegarem, w tym przypadku pochodzącym z oscylatora kwarcowego.

Po utworzeniu nowego projektu, Processor Expert automatycznie doda moduł (bean) CPU, który znajdziemy w zakładce projektu Processor Expert. Po dwukrotnym kliknięciu otworzy się okno konfiguracyjne (Component Inspector).

Zakładka Properties

- **Clock Settings**
  - **External clock:** wybieramy *Enabled* - na płytce zestawu laboratoryjnego znajduje się rezonator kwarcowy, o częstotliwości 32768Hz,
    - **Clock source:** wybieramy *External crystal*,
    - **Clock frequency [MHz]:** wpisujemy *0.032768*.
- **Enabled speed modes** - konfigurujemy tylko jeden tryb

- **High speed mode**
  - **High speed clock:** wybieramy *External*, czyli skonfigurowane powyżej źródło zegara,
  - **Bus freq. divider:** wybieramy *1* - dzielnik zegara magistrali (w tym przypadku wyjściowego z generatora z pętlą FLL),
  - **FLL mode:** wybieramy *Engaged* - zaprzęgamy do pracy generator z pętlą FLL,
    - **Ref. clock source** - ustawienia parametrów źródła zegara,
    - **Ref. clock divider:** wybieramy *1* - dzielnik zegara wejściowego dla FLL,
    - **DCO mode:** wybieramy *Fine tuned 32kHz* - uzyskanie maksymalnej częstotliwości pracy CPU ze źródła zegara o częstotliwości 32kHz,
    - **DCO fll mult. factor:** wybieramy *608* - uzyskanie częstotliwości DCO 19.92 MHz.

Powyższa procedura pozwala uzyskać częstotliwość pracy mikrokontrolera 19,92MHz oraz częstotliwość magistrali równej połowie częstotliwości CPU 9,96MHz. Warto zwrócić uwagę, że układ MC9S08LL64 może pracować z maksymalną częstotliwością 40MHz. Przestawienie trybu pracy DCO z *608* na *1216* pozwoli na uzyskanie częstotliwości taktowania CPU 39,85MHz. Podobny efekt można uzyskać bazując na wewnętrznym zegarze.

Zad.2 Konfiguracja linii I/O

Celem kolejnego zadania jest skonfigurowanie trzech linii wybranego portu do sterowania diodami LED. Przy wyborze portu oraz linii, należy posłużyć się schematem zestawu laboratoryjnego oraz opisem zworek (złącze JP11). Dokładny opis konfiguracji portów pracujących w trybie wej/wyj można znaleźć w dokumentacji [2] układu w rozdziale 6 na stronie 101.

W poprzednim zadaniu Processor Expert automatycznie dołączył do projektu moduł (bean) CPU. Wszystkie kolejne moduły powinno się wybrać według potrzeb. W tym celu posługujemy się bibliotekami znajdującymi się w oknie Componets Library. Moduły, w które jest wyposażony konfigurowany układ, można znaleźć w zakładce On-Chip Prph. Każdą wybraną linię należy skonfigurować jako wyjście, nadać jej adekwatną nazwę, parametry oraz wybrać minimalny zestaw przydatnych funkcji.

Konfiguracja pojedynczej linii:

W oknie Componets Library należy wybrać katalog z nazwą portu, do którego są podłączone diody LED. Następnie wybieramy moduł BitIO, który po dwukrotnym kliknięciu znajdzie się w zakładce Processor Expert aktualnie otwartego projektu. Konfigurację należy przeprowadzić, podobnie jak w przypadku CPU, w oknie Component Inspector.

zakładka Properties

- **Component name:** nadajemy nazwę adekwatną dla wybranej diody LED np. *LED1*,
- **Pin for I/O:** wybieramy linie do której podłączona jest np. dioda LED1,
- **Drive strenght:** wybieramy *Low* - zwiększona wydajność prądowa wybranej linii,
- **Initialization** - parametry startowe,
  - **Init value:** należy wybrać wartość dla której dioda po resecie nie będzie świecić.

zakładka Methods

Wybieramy minimalny zestaw funkcji potrzebnych do zaświecania i zgaszenia diody LED. Wybrana linia została skonfigurowana jako wyjście, zatem odznaczamy wszystkie funkcje, które służą do odczytania wartości na linii oraz zmieniające jej funkcje z wyjściowej na wejściową. W tym przypadku proponuje się zostawić tylko funkcję *PutVal*.

zakładka Events

Moduł ten nie generuje żadnych zdarzeń/przerwań.

Powyższą konfigurację należy przeprowadzić dla trzech wybranych diod LED. Powyższą konfigurację można przetestować wykorzystując wygenerowane funkcje. Układ programujemy przechodząc do trybu Debug.

Zad.3 Konfiguracja cyklicznego przerwania

W kolejnym zadaniu należy skonfigurować cykliczne przerwanie o wybranej częstotliwości, z możliwością jej zmiany z poziomu programu w zadanym zakresie. Mikrokontroler MC9S08LL64 posiada trzy źródła cyklicznych przerwań. Dwa z nich to uniwersalne, konfigurowalne moduły czasowo-licznikowe TPM1 oraz TPM2. Trzecie źródło to moduł zegara TOD (Time-of-Day). W niniejszym zadaniu należy posłużyć się wybranym modułem TPM. Dokładny opis konfiguracji modułu TPM można znaleźć w dokumentacji [2] układu w rozdziale 17 na stronie 327.

Podobnie jak w poprzednim zadaniu posługujemy się bibliotekami, znajdującymi się w oknie Componets Library. Z katalogu TMPx dołączamy do projektu moduł TimerInt.

zakładka Properties

- **Component name:** nadajemy nazwę modułu np. *MyClock*,
- **Interrupt period** otwieramy konfigurator klikając klawisz [...],
  - **Clock source:** źródło zegara pozostawiamy na *Auto select*,
  - **Prescaler:** preskaler zegara pozostawiamy na *Auto select*,
  - **Runtime setting type:** wybieramy *from time interval* - pozwoli to na zmianę częstotliwości przerwań z poziomu programu,
    - **Init value:** należy ustawić wartość początkową,
    - **Low limit:** należy ustawić wartość minimalną (maksymalna częstotliwość przerwań),
    - **High limit:** należy ustawić wartość maksymalną (minimalna częstotliwość przerwań),
  - **Min. resolution:** minimalny skok w wybranym powyżej zakresie.

zakładka Methods

Wybieramy minimalny zestaw funkcji potrzebnych jedynie do zmiany częstotliwości przerwań. Pozostawiamy tylko te funkcje, które dla wybranego zakresu mają sens użycia.

zakładka Events

- **Event module name:** pozostawiamy *Events* - jest to nazwa modułu w którym Processor Expert umieści funkcje obsługujące konfigurowane przerwanie,
- **OnInterrupt**

- **Event procedure name:** wpisujemy nazwę funkcji, którą wywoła konfigurowane przerwanie np. *MyClockInterrupt*

Konfiguracja przerwania jest już zakończona. W celu zwizualizowania efektu jego działania, można posłużyć się diodami LED oraz wygenerowanymi w poprzednich zadaniach funkcjami. W zakładce *Files*, aktualnie otwartego projektu, należy odszukać moduł, w którym Processor Expert umieścił funkcje wywoływaną przez skonfigurowane przerwanie. W tym przypadku będzie to plik *Event.c* znajdujący się w katalogu *User modules*. Po odszukaniu funkcji *MyClockInterrupt* możemy posłużyć się zaproponowanym poniżej kodem.

```
void MyClockInterrupt(void)
{
  /* Write your code here ... */
  static bool toggle=TRUE;
  toggle=!toggle;
  (void)LED3_PutVal(toggle);
}
```

Zad.4 Konfiguracja modułu klawiatury

Mikrokontroler MC9S08LL64 posiada wbudowany moduł KBI (Keyboard Interrupt) do obsługi 8-klawiszowej klawiatury. Płytka laboratoryjna zestawu TOWER wyposażona jest w cztery klawisze, z których trzy są podłączone do KBI, a czwarty do zewnętrznego źródła przerwań IRQ. Dokładny opis konfiguracji modułu KBI można znaleźć w dokumentacji [2] układu w rozdziale 7 na stronie 119.

Konfiguracja KBI odbywa się przy użyciu jedynie uproszczonego konfiguratora Device Initialization. Podobnie jak w poprzednim zadaniu należy posłużyć się bibliotekami znajdującymi się w oknie Componets Library. Z katalogu KBI dołączamy do projektu moduł Init_KBI.

zakładka Properties

- **Component names:** wpisujemy nazwę modułu np. *MyKey*,
- **Settings**
  - **Triggering sensitivity:** pozostawiamy *edge* - przerwanie zostanie wywołane jednorazowo po wystąpieniu zbocza na linii,
- **Pins**
  - **Pin*x*:** konfigurujemy te linie, do których podłączono klawisze i wybieramy *Enabled*,
    - **Pull resistor:** wybieramy *pull up* - podłączamy rezystor polaryzujący linię,
    - **Interrupt polarity:** pozostawiamy *low level and/or falling edge* - przerwanie zostanie wywołane po pojawieniu się zbocza opadającego,
- **Interrupts/DMA:**
  - **Keyboard request**,
    - **Keyboard request interrupt:** wybieramy *Enabled*,
    - **ISR name:** wpisujemy nazwę funkcji wywoływanej po naciśnięciu klawisza, np. *MyKeyInterrupt*.

UWAGA: Device initialization nie generuje kodu obsługującego przerwanie w module wskazanym przez programistę, jak to miało miejsce w przypadku TPM. Funkcję tę, należy odszukać w wygenerowanych przez Processor Expert plikach inicjalizujących moduł KBI i następnie samodzielnie umieścić ją w wybranym module.

Pliki inicjalizujące KBI, znajdują się w zakładce *Files* aktualnie otwartego projektu. Z katalogu *Generated Code* należy otworzyć wygenerowany przez Processor Expert plik, zawierający funkcje inicjalizujące oraz obsługujące przerwania modułu KBI. Dla powyższego przykładu kod ten może wyglądać następująco:

```
** ###################################################################
**
**  The interrupt service routine(s) must be implemented
**  by user in one of the following user modules.
**
**  If the "Generate ISR" option is enabled, Processor Expert generates
**  ISR templates in the CPU event module.
**
**  User modules:
**
** ###################################################################
ISR(MyKeyInterrupt)
{
  // NOTE: The routine should include the following actions to obtain
  //       correct functionality of the hardware.
  //
  //  Keyboard Interrupt Flag is cleared by writing a 1 to the
  //  KBACK control bit.
  // Example:   KBISC_KBACK = 0x01;
}
```

W celu zwizualizowania efektu działania klawiatury, można posłużyć się diodami LED oraz wygenerowanymi w poprzednich zadaniach funkcjami. Powyższy kod najwygodniej jest umieścić w pliku zawierającym pozostałe funkcje obsługujące przerwania. W tym przypadku będzie to plik *Event.c*, znajdujący się w katalogu *User modules*. Kod ten można zmodyfikować wg zaproponowanego poniżej przykładu:

```
ISR(MyKeyInterrupt)
{
   (void)LED1_PutVal(PTAD_PTAD6);
   (void)LED2_PutVal(PTAD_PTAD7);
   KBISC_KBACK = 0x01;
}
```

Należy także pamiętać o nagłówku powyższej funkcji, który powinien znaleźć się, w tym przypadku, w pliku *Event.h*. Poniżej przedstawiono przykład:

```
__interrupt void MyKeyInterrupt(void);
```

Zad.5 Generator sygnału PWM

W zadaniu tym należy skonfigurować, niewykorzystany, drugi moduł TPM. Tym razem będzie on pracował jako modulator PWM o wybranych parametrach (częstotliwość i wypełnienie). Efekt pracy modulatora należy zwizualizować na ostatniej niewykorzystanej diodzie LED. Z katalogu TMPx dołączamy do projektu moduł PWM.

zakładka Properties

- **Component names:** wpisujemy nazwę modułu np. *LED4*,
- **Output pin:** wybieramy linię z podłączoną ostatnią niewykorzystaną dioda LED,
- **Period:** wpisujemy okres sygnału PWM,
- **Starting pulse width:** wpisujemy początkowe wypełnienie,
- **Initial polarity:** wybieramy *low*,

zakładka Methods

Wybieramy minimalny zestaw funkcji, potrzebnych do sensownego zadawania wypełnienia konfigurowanego generatora sygnału PWM.

zakładka Events

Powyższa konfiguracja nie generuje zdarzeń/przerwań.

W celu zwizualizowania efektu działania generatora, można posłużyć się jedną z wygenerowanych funkcji, pozwalających na zadawanie wypełnienia sygnału PWM i umieścić ją w funkcji main() lub w funkcji obsługującej klawiaturę.

Zad.6 Konfiguracja przetwornika A/C

W ostatnim zadaniu należy skonfigurować i uruchomić zintegrowany 10-kanałowy, 12-bitowy przetwornik ADC. Dokładny opis konfiguracji ADC można znaleźć w dokumentacji [2] układu w rozdziale 11 na stronie 171.

Przed przystąpieniem do tego zadania, należy ponownie posłużyć się schematem zestawu laboratoryjnego TOWER. Warto zwrócić uwagę na użyte układy peryferyjne, stanowiące źródła sygnałów analogowych: potencjometr, czujnik światła oraz 3-osiowy akcelerometr. Pomiar wybranego kanału należy zwizualizować przy pomocy diody LED, podłączonej do generatora PWM.

zakładka Properties

- **Component name:** nadajemy nazwę modułu np. *MyAnalog*,
- **Interrupt service/event** wybieramy *Disabled* - brak przerwań,
- **A/D channels** wybieramy *1* - liczba kanałów pomiarowych,
  - **A/D channel (pin)** wybieramy linię pomiarową w zależności od wybranego źródła,
- **A/D resolution** wybieramy rozdzielczość pomiaru, tak aby precyzja pomiaru była sensownie wykorzystana,
- **Conversion time** wpisujemy czas pojedynczego pomiaru,

zakładka Methods

Wybieramy minimalny zestaw funkcji, potrzebnych jedynie do zainicjalizowania pomiaru oraz odczytania zmierzonej wartości.

zakładka Events

Powyższa konfiguracja nie generuje zdarzeń/przerwań.

Zmierzoną przy pomocy przetwornika A/C wartość, podajemy jako wypełnienie sygnału PWM. W tym celu należy, np. w programie głównym użyć wcześniej skonfigurowanych modułów oraz ich funkcji. Należy zwrócić uwagę, że funkcja *MyAnalog_GetValue16* zwraca wynik (8,10,12-bitowy) w postaci 16-bitowej zmiennej z wartością wyrównaną do lewej. Poniżej zaproponowano przykładowy kod.

```
for(;;) {
   (void)MyAnalog_Measure(TRUE);
   (void)MyAnalog_GetValue16(&measure);
   // dla pomiaru 10 bit dokonujemy przesunięcia o 6 bitów
  (void)LED4_SetDutyUS(measure>>6);
}
```

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] *MC9S08LL64 Reference Manual*, MC9S08LL64RM, Rev. 5, 10/2009, Freescale Semiconductors

[3] *TWR-S08LL64: MC9S08LL 8-bit Segment LCD Module*, www.freescale.com/webapp/sps/site/prod_summary.jsp?code=TWR-S08LL64

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