Projekt wprowadzający: Generacja sygnału PWM

Krzysztof Kowol w ramach SKN Fusion

23 lutego 2017

1 Określenie problemu

Zaprogramować mikrokontroler MSP430 tak, aby na wyjściu cyfrowym generował napięciowy sygnał PWM o wybranej częstotliwości i wypełnieniu. Do wyjścia podpiąć diodę i sprawdzić, czy ze zmianą wypełnienia sygnału PWM zmienia się natężenie jej światła. Zmianę wypełnienia można zrealizować poprzez wpisanie nowej wartości stałej w kodzie programu i ponowną kompilację lub zadawanie nowych wartości przy wykorzystaniu komunikacji szeregowej.

1.1 Cel główny:

1. Zmiana wypełnienia sygnału prostokątnego.
2. Zmiana częstotliwości sygnału.

1.2 Cel poboczny:

1. Komunikacja szeregowa z PC.

2 Realizacja

Prace związane z realizacją projektu zostały wykonane w oparciu o układ ewaluacyjny MSP430 LaunchPad v1.5 z mikrokontrolerem MSP430g2553. Kod w języku C z plikiem nagłówkowym msp430g2553.h, kompilowany przy użyciu msp430-gcc. Programowanie przy użyciu mspdebug.

2.1 Zmiana wypełnienia sygnału

Zmianę wypełnienia sygnału zrealizowano w oparciu o Timer_A mikrokontrolera. Zostało to wykonane na dwa sposoby.

PWM programowy Programowo, bazując na przerwaniach w oparciu o rejestry porównawcze CCR0 i CCR1. W tym przypadku należy mieć na uwadze dwie rzeczy. Pierwszą jest występowanie oddzielnych wektorów przerwań dla ww. rejestrów. Obsługa przrwań dla CCR0:

```
#pragma vector = TIMER0_A0_VECTOR //CCR0 interrupt vector
__interrupt void CCR0_Interrupt(void)
{
      P1OUT |= BIT0; //set LED1
}
```

Obsługa pozostałych przrwań Timera_A:

```
#pragma vector = TIMER0_A1_VECTOR //Timer0,TAIFG interrupt vector
__interrupt void TimerA(void)
{
  switch(TAIV)
  {
    case 0x002: // CCR1
    {
                P1OUT &= ~BIT0; //reset LED1
    }
                break;
    case 0x004: //CCR2 not used
    {
    }
                break;
    case 0x00A: //TAR overflow interrupt not used

                break;
  }
}
```

Druga rzecz na którą należy zrócić uwage to domyślnie nieaktywne przerwania. Zezwolenie przerwań odbywa się poprzez ustawienie odpowiednich flag w rejestrach TACCTLx. Przykład dla CCR0:

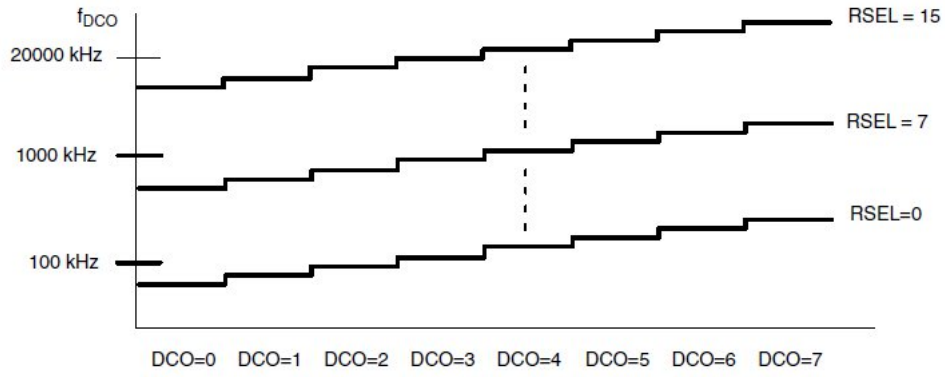
```
TACCTL0 |= CCIE; // Enable CCR0 interrupt
```

Wprowadzając mikrokontroler w stan energooszczędny (Low Power Mode), również należy pamiętać o zapewnieniu obsługi przerwań.

```
_BIS_SR(LPM0_bits + GIE); // Sleep in LPM0 with interrupts enabled
```

PWM sprzętowy Sprzętowo, konfigurując rejestrem PxSEL odpowiedni pin jako wyjście z Timera. Konfiguracja sprzętowego PWMa ogranicza się do konfiguracji Timera oraz pinów zgodnie z dokumentacją.

Rysunek 1: Charakterystyka DCO



2.2 Zmiana częstotliwości sygnału

Timer_A został skonfigurowany tak, że jako źródło sygnału zegarowego został wybrany SMCLK. SMCLK może być taktowany z oscylatora DCO, którego parametry pracy można modyfikować z poziomu programu w trakcie pracy układu. Charakterystyka DCO, patrz rys.1. Częstotliwość pracy DCO określa się poprzez zakres i stopień w rejestrach DCOCTL(step) oraz BCSCTL1(range).

2.3 Komunikacja szeregowa z PC

Tej części projektu nie została wykonana we względu na problem z doborem źródła taktowania dla USCI. Do taktowania timera używany jest sygnał SMCLK o zmiennej częstotliwości pracy ze względnu na DCO. Jednak moduł komunikacji szeregowej USCI musi być taktowany innym sygnałem o stałej częstotliwości np. ACLK, w celu spełnienia zależności czasowych transmisji.

3 Źródła

1. TI MSP430x2xx Family User's Guide
2. TI MSP430G2x53 Datasheet
3. http://xanthium.in/brief-introduction-msp430g2xxx-using-ti-launchpad-development-board
4. http://www.embedds.com/configuring-the-dco-of-msp430/