Funkcje czasowe (*multifunction timer*)

- TO (*Timer Overflow*) – przerwanie cykliczne przy przepełnieniu licznika TCNT
- RT (*Real–Time*) – przerwanie cykliczne o programowalnym okresie
- COP (*Computer Operates Properly watchdog*) – sprzętowy nadzór poprawnej pracy programu
- OC (*Output Compare*) – generowanie sygnałów przez porównanie licznika TCNT z programowalnym rejestrem (4-5 kanałów)
- IC (*Input Capture*) – zapamiętywanie stanu TCNT przy wybranym zboczu sygnału wejściowego (3-4 kanały)
- PA (*Pulse Accumulator*) – zliczanie zdarzeń zewnętrznych lub bramkowanego sygnałem zewnętrznym zegara wewnętrznego

Opóźnienia z TO (*polling*)

```
* definicje stalych
PRE16   EQU     %11         bity PR1:PR0 - podzial przez 16
TOF     EQU     %10000000   maska bitu TOF w TFLG2
COUNT   EQU     20          zadana ilosc przepelnien
* Dla czestotliwosci zegara E = 2MHz (T_e = 0.5 [us]):
* T = 20 * (16 * 2^16 * 0.5) [us] = 10.48576 [s]
* inicjalizacja wstepnego dzielnika dla TCNT (PRx w TMSK2)
        LDAA    #PRE16      podzial E przez 16
        STAA    TMSK2       PR1:PR0

*************************************************************
* petla oczekiwania na zadana ilosc przepelnien
LOOP    LDAB    #COUNT      licznik przepelnien w AccB
        LDAA    #TOF        maska bitu TOF w AccA
TOLOOP  BITA    TFLG2       sprawdzenie flagi TOF
        BEQ     TOLOOP      oczekiwanie na przepelnienie
        STAA    TFLG2       kasowanie flagi TOF
        DECB                odliczanie COUNT razy
        BNE     LOOP        czekanie na wyzerowanie licznika
* dalszy ciag programu po opoznieniu
        JSR     ZADANIE     wywolanie podprogramu
        BRA     LOOP        powrot do czekania
```

Przerwania cykliczne (TOI)

```
*************************************************************
* ustawienie licznika przerwan w pamieci RAM
        LDAA    #COUNT      zadana liczba powtorzen
        STAA    COUNTER     ustawienie licznika
* inicjalizacja przerwan od TO
        LDAA    #TOF        maska bitu TOF
        STAA    TFLG2       kasowanie flagi TOF
        STAA    TMSK2       zezwolenie na przerwanie TOI
        CLI                 odblokowanie przerwan w CPU (CCR)
* petla bez wyjscia (reszta w obsludze przerwania)
DLOOP   BRA     DLOOP

* procedura obslugi przerwania TOI
TOISVC  DEC     COUNTER     odliczanie przerwan
        BNE     TOEXIT      wyjscie z obslugi TOI
* tu: wykonanie wlasciwego zadania cyklicznego
        JSR     ZADANIE     wywolanie podprogramu
        LDAA    #COUNT      zadana liczba powtorzen
        STAA    COUNTER     ustawienie licznika
* zgaszenie flagi przerwania
TOEXIT  LDAA    #TOF        maska bitu TOF w AccA
        STAA    TFLG2       kasowanie flagi TOF
        RTI                 powrot z przerwania
```

Programowalne przerwania cykliczne (RTI)

```
* definicje stalych
RTD8    EQU     %11         bity RTR1:RTR0 - podzial przez 8
RTIF    EQU     %10000000   maska bitu RTIF w TFLG2
* Dla czestotliwosci zegara E = 2MHz (T_e = 0.5 [us]):
* T = (8 * 2^13 * 0.5) [us] = 32.768 [ms]
* ustawienie dzielnika dla RTI (RTRx w PACTL)
        LDAA    #RTD8       podzial E przez 8*2^13
        STAA    PACTL       RTR1:RTR0

* inicjalizacja przerwan RTI
        LDAA    #RTIF       maska bitu RTIF
        STAA    TFLG2       kasowanie flagi RTIF
        STAA    TMSK2       zezwolenie na przerwanie RTI
        CLI                 odblokowanie przerwan w CPU (CCR)
* petla bez wyjscia (reszta w obsludze przerwania)
DLOOP   BRA     DLOOP

* procedura obslugi przerwania RTI
RTISVC  JSR     ZADANIE     wywolanie podprogramu
* zgaszenie flagi przerwania
        LDAA    #RTIF       maska bitu RTIF w AccA
        STAA    TFLG2       kasowanie flagi RTIF
        RTI                 powrot z przerwania
```

Przerwania cykliczne z OC

```
OC2F    EQU     %01000000   maska bitu OC2F w TFLG1
TIMDEL  EQU     20000       stala dla opoznienia 10 [ms]
* T = TIMDEL * T_e = (20000 * 0.5) [us] = 10 [ms]
* przygotowanie pierwszego przerwania OC2
        LDD     TCNT        odczyt licznika TCNT
        ADD     #TIMDEL     przesuniecie o 10 [ms]
        STD     TOC2        przerwanie za ok. 10 [ms] !
* odblokowanie przerwan
        LDAA    #OC2F       maska OC2F (i OC2I)
        STAA    TFLG1       kasowanie flagi OC2F
        STAA    TMSK1       odblokowanie OC2I
        CLI                 odblokowanie przerwan w CPU
* petla bez wyjscia
DLOOP   BRA     DLOOP

* procedura obslugi przerwania OC2
OC2SVC  JSR     ZADANIE     wywolanie podprogramu
        LDD     TOC2        odczyt rejestru komparatora
        ADD     #TIMDEL     przesuniecie o 10 [ms]
        STD     TOC2        nastepne przerwanie za 10 [ms]
        LDAA    #OC2F       maska bitu OC2F w AccA
        STAA    TFLG1       kasowanie flagi OC2F
        RTI                 powrot z przerwania
```

Generowanie przebiegu przy pomocy OC

```
* inicjalizacja OC2 (PA6)
        LDAA    #%01000000  OM2=0, OL2=1 - toggle
        STAA    TCTL1       tryb pracy PA6
        LDD     TCNT        odczyt licznika TCNT
        ADD     HPERIOD     zadany okres [us]
        STD     TOC2
* odblokowanie przerwan
        LDAA    #OC2F       maska OC2F (i OC2I)
        STAA    TFLG1       kasowanie flagi OC2F
        STAA    TMSK1       odblokowanie OC2I
        CLI                 odblokowanie przerwan w CPU
* petla bez wyjscia (lub inna petla glowna)
DLOOP   BRA     DLOOP

* procedura obslugi przerwania OC2
OC2SVC  LDD     TOC2        odczyt rejestru komparatora
        ADD     HPERIOD     przesuniecie o pol okresu
        STD     TOC2
        LDAA    #OC2F       maska bitu OC2F w AccA
        STAA    TFLG1       kasowanie flagi OC2F
        RTI                 powrot z przerwania

        ORG     RAM
HPERIOD RMB     2           wartosc okresu [us]
```

Pomiar czasu przy pomocy IC (1/2)

```
* inicjalizacja pseudo-wektora przerwan
        LDAA    #$7E         kod JMP (EXT)
        STAA    PVIC1        pseudo-wektor IC1
        LDX     #IC1SVC      adres obslugi przerwania IC1
        STX     PVIC1+1      skok do obslugi przygotowany
PWSTRT  LDX     #$1000       wskaznik bloku rejestrow
        LDAA    #%00010000   EDG1B:EDG1A = 0:1
        STAA    TCTL2,X      IC1 - zbocze narastajace
        LDAA    #$FF
        STAA    IC1MOD       FF-IC1 wylaczone; 0-/; 1-\
        CLR     IC1FIN       to nie koniec...
        BCLR    TFLG1,X,$FB zerowanie IC1F
        BSET    TMSK1,X,$04 wlaczenie przerwan IC1 (HW)
        CLI                  zezwolenie na przerwania w CPU
WLOOP   LDAA    IC1FIN       sprawdzenie flagi konca
        BEQ     WLOOP        petla oczekiwania na koniec
        SEI                  koniec impulsu, blokada przerwan
* wyliczenie dlugosci w [us]
        LDD     PWVAL        ilosc taktow (0.5uS/takt)
        LSRD                 16-bitowe dzielenie przez 2
        BCC     NORND        zaokraglic ?
        ADDD    #1           tak!
NORND   BRA     NORND        wynik gotowy w AccD
```

Pomiar czasu przy pomocy IC (2/2)

```
* procedura obslugi przerwania IC1
IC1SVC  LDX     #$1000      adres bazowy bloku rejestrow
        INC     IC1MOD      $FF->0 => pierwsze,
        BNE     TRAILE      0->1   => drugie zbocze
* obsluga narastajacego zbocza
        LDD     TIC1,X      odczyt czasu
        STD     PWVAL       zachowany wynik
* przestawienie IC1 na zbocze opadajce
        BCLR    TCTL2,X,$30 EDG1B:EDG1A->0:0
        BSET    TCTL2,X,$20 EDG1B:EDG1A->1:0
        BRA     IC1OUT      wyjscie z przerwania
* obsluga opadajacego zbocza
TRAILE  LDD     TIC1,X      odczyt czasu
        SUBD    PWVAL       odstep od poprzedniego zbocza
        STD     PWVAL       zapis wyniku
        BCLR    TCTL2,X,$30 wylaczenie IC1
        INC     IC1FIN      flaga konca impulsu
IC1OUT  BCLR    TFLG1,X,$FB kasowanie IC1F
        RTI                 powrot z przerwania

        ORG     RAM
PWVAL   RMB     2           wynik pomiaru
IC1MOD  RMB     1           stan procesu pomiaru
IC1FIN  RMB     1           flaga zakonczenia pomiaru
```

PWM przy pomocy OC (1/2)

```
OC2F    EQU     %01000000   maska bitu OC2F w TFLG1
PWM1P   EQU     10          stala dla wypelnienia 1%
PWMPER  EQU     100*PWM1P   stala dla okresu (100%)
* ustawienie bitu portu A i sposobu zmieniania (toggle)
        LDAA    #%10000000  OM2=1, OL2=0 - ustaw LOW
        STAA    TCTL1       pierwsze ustawienia wyjcia
        LDAA    #OC2F       maska bitu OC2
        STAA    CFORC       wymuszenie ustawienia LOW
        LDAA    #%01000000  OM2=0, OL2=1 - toggle
        STAA    TCTL1       normalny tryb pracy
* przygotowanie pierwszego przerwania OC2
        LDAA    #100
        SUBA    PWM         stan niski
        LDAB    #PWM1P      stala dla 1%
        MUL
        ADD     TCNT        biezacy stan licznika TCNT
        STD     TOC2        przerwanie po stanie niskim
* odblokowanie przerwan
        LDAA    #OC2F       maska OC2F (i OC2I)
        STAA    TFLG1       kasowanie flagi OC2F
        STAA    TMSK1       odblokowanie OC2I
        CLI                 odblokowanie przerwan w CPU
* petla bez wyjscia (lub inna petla glowna)
DLOOP   BRA     DLOOP
```

PWM przy pomocy OC (2/2)

```
* procedura obslugi przerwania OC2
OC2SVC  LDAA    PORTA       port timer-a
        BITA    #OC2F       sprawdzenie stanu wyjscia
        BEQ     OC2LO       obsluga stanu niskiego
* tu: stan wysoki
        LDAA    LMOTOR      zadane wypelnienie
        BRA     OC2COM      dalej wspolny kod
* tu: stan niski
OC2LO   LDAA    #100        okres
        SUBA    LMOTOR      obliczenie czasu niskiego
* wspolny kod
OC2COM  LDAB    #PWM1P      stala dla 1%
        MUL                 stala czasowa
        ADD     TOC2        przesuniecie komparatora
        STD     TOC2
        LDAA    #OC2F       maska bitu OC2F w AccA
        STAA    TFLG1       kasowanie flagi OC2F
        RTI                 powrot z przerwania

        ORG     RAM
LMOTOR  RMB     1           zmienna: wypelnienie [%]
```

ADC + PWM = robocik swiatłolubny

```
* inicjalizacja ADC
        LDAA    #%10010000  ADPU + DLY
        STAA    OPTION      tylko raz po restarcie
        JSR     DELAY
        LDAA    #%00110000  SCAN + MULT, bank 0
        STAA    ADCTL
* tu - inicjalizacja PWM na OC2 i OC3
*
* petla glowna:
* patrz -> mysl -> dzialaj
main
* patrz:
*   odczyt czujnikow swiatla
        LDAA    ADR1        prawy odczyt w AccA
        LDAB    ADR2        lewy odczyt w AccB
* mysl:
*   przetworzenie obrazu na decyzje o predkosci
*   i kierunku ruchu silnikow
        JSR     mysl        reakcja na odczyty
* dzialaj:
*   ustawianie predkosci silnikow w obsludze PWM
*   w przerwaniach od OC2 i OC3
        JMP     main        petla bez konca
```

```
mysl    CMPA    #100        prog zadowolenia
        BLO     work        zle, trzeba sie poruszac
        CMPB    #100
        BLO     work        zle, trzeba sie poruszac
* o, jak jasno, tu mi dobrze!
        CLR     LMOTOR      lewy stop
        CLR     RMOTOR      prawy stop
        RTS                 koniec myslenia!
* znow trzeba pracowac :(
work    CMPA    #50         dolny prog jasnosci
        BLO     look        zbyt ciemno
        CMPB    #50
        BHS     appr        cos widac
* ciemnosc widze, rozejrze sie!
look    LDAA    #50         predkosc [%]
        STAA    LMOTOR      pol naprzod
        STAA    RMOTOR      pol wstecz
        LDAA    #%10        lewe >, prawe <
        STAA    PORTB,X
        RTS                 koniec myslenia!
* o, cos widze, podjade blizej!
appr    STAA    LMOTOR      prawe oko -> lewy silnik
        STAB    RMOTOR      lewe oko -> prawy silnik
        CLR     PORTB,X     oba naprzod
        RTS                 koniec myslenia!
```