Arduino – comparatorul intern

Ce este comparatorul ?

In electronica digitala, comparatorul este un mic dispozitiv care compara 2 tensiuni sau curenti si produce un semnal care iti spune care din cele doua este mai mare. Acesta este rolul de baza al comparatorului. El este format din doua intrari analogice si o singura iesire digitala. Intrarile analogice sunt marcate cu V+ si V-, iar iesirea digitala cu Vo. Iesirea poate genera doar 2 stari si anume: „1“ logic daca intrarea V+ este mai mare decat intrarea V- sau poate genera „0“ logic daca intrarea V+ este mai mica decat intrarea V-.

In aceeasi maniera functioneaza si comparatorul placii Arduino si daca vrei sa descoperi in ce situatii poti sa il folosesti, atunci viziteaza link-urile de mai jos:

http://www.electronicsblog.net/very-simple-arduino-capacitance-meter-using-16-bit-timer-and-analog-comparator/

Poti sa iti construiesti un capacimetru folosind o placa Arduino, comparatorul intern si timer-ul placii sau un osciloscop digital:

http://www.instructables.com/id/Girino-Fast-Arduino-Oscilloscope/?ALLSTEPS

Afla ca exista si o librarie Arduino care te poate ajuta foarte mult in proiectele tale:

http://www.leonardomiliani.com/2012/analogcomp-una-libreria-per-gestire-il-comparatore-analogico/?lang=en

Cum genereaza intreruperi comparatorul intern ?

Pe langa faptul ca poate compara 2 semnale asta fiind functia lui, comparatorul poate genera si intreruperi. Asta inseamna ca poti scrie un program care sa execute „ceva“, iar atunci cand apare o

situatie exterioara, spre exemplu: iesirea unui senzor a depasit o valoare de referinta, atunci placa Arduino poate sa raspunda prompt la intrerupere, opreste executia normala a programului si executa codul asociat intreruperii.

In programarea de nivel scazut (low level) poti sa spui ca microcontroller-ul executa rutina ISR (Interrupt Service Routine).

Iata cum arata o rutina ISR care deserveste o intrerupere generata de comparatorul intern:

```
ISR(ANALOG_COMP_vect)
{
   digitalWrite(13, !digitalRead(13));
}
```

Structura intreruperii este simpla si anume: ISR este o instructiune macro pe care compilatorul o detecteaza automat si o interpreteaza ca si rutina ISR. Asa ca ori de cate ori comparatorul intern genereaza o intrerupere, microcontroller-ul opreste executia normala a programului si executa toate liniile de cod din aceasta rutina. Apoi se intoarce de unde a plecat si continua cu executia programului principal.

Link-ul de mai jos te ajuta sa descoperi si alte intreruperi pe care le poti gestiona cu rutina ISR, pentru ca nu doar comparatorul intern poate genera intreruperi, dar mai sunt si alte module interne:

http://www.nongnu.org/avr-libc/user-manual/group__avr__interrupts.html

Cum se configureaza comparatorul intern ?

La nivel de cod, configurarea comparatorului este simpla. Configurarea este totusi necesara pentru functionarea corecta a comparatorului. Tot ce trebuie sa faci este sa accesezi un registru si sa setezi anumiti biti. Lucrul asta il vei face in rutina setup(). Spre exemplu iata cum se configureaza comparatorul intern:

```
void setup() {
  pinMode(7,INPUT);
  pinMode(13, OUTPUT);
  digitalWrite(13, LOW);
  Serial.begin(9600);
  ACSR = B01011010;
}
```

Linia de cod cu care nu esti tocmai familiar este ACSR = B01011010. Indicatorul B din fata sirului de biti ii indica compilatorului ca are de-a face cu un sir de biti si nu o valoare numerica. ACSR este registrul de status si control al comparatorului analogic si are urmatoarea structura de biti:

ACD, ACBG, ACO, ACI, ACIE, ACIC, ACIS1, ACIS0

Fiecare bit are rolul lui. Daca te uiti la linia de cod de mai sus vei observa ca placa seteaza niste biti pe „1“ si alti biti ii pastreaza pe „0“.

Prin setarea bitului ACBG, placa Arduino va selecta singura tensiunea de referinta pentru intrarea pozitiva a comparatorului. Adu-ti aminte ca un comparator are 2 intrari (una pozitiva, cealalta negativa) si o iesire. Practic placa Arduino va alege singura o tensiune de prag pentru intrarea pozitiva care este de aproximativ 1,2V. Intrarea negativa este accesibila la pinul digital 7. La acest pin tu vei aplica o tensiune analogica pe care doresti ca placa Arduino sa o compare cu tensiunea de referinta (1,2V).

Prin setarea bitului ACI se sterge bitul de flag al intreruperii. Lucrul asta te ajuta pentru ca in primul moment de functionare al placii pot aparea intreruperi false si chiar daca vor aparea, bitul flag este sters deocamdata si placa nu raspunde la aceste intreruperi.

Prin setarea bitului ACIE (Analog Comparator Interrupt Enable) se activeaza intreruperea comparatorului intern asa ca ori de cate ori apare o tranzitie sau o schimbare la iesirea comparatorului, acesta va genera o intrerupere, iar placa Arduino va executa rutina ISR.

Prin setarea bitilor ACIS1 si ACIS0, comparatorul va genera intreruperi pe frontul descrescator al iesirii, adica atunci cand iesirea trece din „1“ in „0“ logic.

Totusi ce valori are iesirea comparatorului ?

Intotdeauna iesirea va fi „1“ daca tensiunea de referinta este mai mare decat tensiunea aplicata pe pinul digital 7. In schimb iesirea comuta in „0“ daca tensiunea de pe pinul 7 creste si depaseste tensiunea de referinta. Cu alte cuvinte, cand are loc aceasta tranzitie din „1“ in „0“, comparatorul genereaza o intrerupere, iar placa Arduino executa rutina ISR asociata.

Cum testez codul pe o placa Arduino ?

Tot ce trebuie sa faci este sa incarci sketch-ul de mai jos. Copiaza codul sursa si incarca-l in placa Arduino.

```
void setup() {
   pinMode(7,INPUT);
   pinMode(13, OUTPUT);
   digitalWrite(13, LOW);
   Serial.begin(9600);
   ACSR = B01011010;
}

void loop() {
}
```

```
ISR(ANALOG_COMP_vect) {
  digitalWrite(13, !digitalRead(13));
}
```

Folosindu-te de un fir, conecteaza pinul marcat cu 5V la pinul digital 7 (atentie mare sa nu il conectezi la pinul VIN deoarece poate duce la defectarea placii). Apoi incepe si comuta firul de pe pinul 5V pe pinul GND. De fiecare data cand comuti, poti observa ca se aprinde led-ul, adica se executa linia de cod digitalWrite(13, !digitalRead(13)); din interiorul rutinei ISR.

Iata un alt exemplu:

```
void setup() {
  Serial.begin(9600);
  ACSR =
    (0 << ACD) |
    (0 << ACBG) |
    (0 << ACO) |
    (1 << ACI) |
    (1 << ACIE) |
    (0 << ACIC) |
    (1 << ACIS1) |
    (1 << ACIS0);

  pinMode(13, OUTPUT);
  pinMode(9, OUTPUT);
}


void loop() {

  if(ACSR & (1 << ACO))
    digitalWrite(9, HIGH);
  else
    digitalWrite(9, LOW);

  delay(100);
}

ISR(ANALOG_COMP_vect ) {
  digitalWrite(13, !digitalRead(13));
}
```

De aceasta data poti observa o forma noua de scriere a registrului ACSR. Spre deosebire de sketch-ul precedent acesta compara nivelele semnalelor aduse pe pinii digitali 6 si 7 (mai sus programul compara tensiunea de pe pinul 7 cu tensiunea de referinta interna).

Spre exemplu, poti folosi acest sketch pentru a regla temperatura intr-o incinta. Pentru asta vei avea nevoie de urmatoarele componente:

- O placa Arduino UNO
- Un tranzistor TIP122 brick - http://www.robofun.ro/bricks/tranzistor-tip122-brick
- sau o placa 2 relee SPDT - http://www.robofun.ro/placa_2_relee_12vdc
- Un microventilator - http://www.robofun.ro/electronice/generale/microventilator-12V
- Un alimentator extern - http://www.robofun.ro/surse_de_alimentare/alimentatoare
- Un senzor de temperatura brick - http://www.robofun.ro/bricks/senzor-temperatura-brick
- Rezistoare pentru a construi un divizor rezistiv - http://www.robofun.ro/electronice/rezistoare

De ce tranzistorul TIP122 brick ? Pentru ca tranzistorul iti permite sa comanzi o sarcina de putere mare direct cu placa Arduino. Pe de alta parte, poti opta pentru placa cu 2 relee SPDT daca doresti sa comanzi nu unul, ci doi consumatori care se alimenteaza in curent alternativ.

Cum se conecteaza componentele de mai sus ?

- Mai intai vei incepe cu ventilatorul sau sursa de putere pe care doresti sa o comanzi. Firele ventilatorului se conecteaza in conectorul marcat cu MOTOR de pe placa tranzistorului.
- Pentru tranzistorul TIP122: pinii marcati cu VIN si GND se conecteaza la bornele „+“ si „-“ ale sursei de alimentare, iar pinul marcat cu IN la pinul digital 13 de pe placa Arduino.
- Pentru senzorul de temperatura brick: pinii VCC si GND la pinii 5V si GND de pe placa Arduino, iar pinul OUT de pe senzorul brick la pinul digital 7.
- Divizorul rezistiv ? Aici trebuie sa descoperi cum sa conectezi 2 rezistori la pinul digital 6, pentru ca aici vei alege tensiunea de prag la care se va declansa ventilatorul.

Totusi iti raman foarte multe lucruri de facut. Spre exemplu, trebuie sa descoperi ce tensiune scoate senzorul brick, trebuie sa calculezi divizorul rezistiv si sa modifici sketch-ul de mai sus pentru a efectua comanda ventilatorului cu grade variabile de turatie.