Arduinokurs

Kurstillfälle 4

CW-generering

Det här kan ses som överkurs men kan ändå vara roligt för att kunna generera CW på ett ”enkelt” sätt. Det blir en hel del nytt men vi tar det steg för steg…

Som alla vet gäller följande för CW
En kort = 1 tidsenhet
En lång = 3 tidsenheter
Mellanrummet mellan tecknen för samma bokstav/tecken = 1 tidsenhet
Mellanrummet mellan bokstav/tecken = 3 tidsenhet
Ordmellanrum = 7 tidsenheter.

CW är binärt och enkelt att generera i en dator. En etta betyder ton (nyckel nedtryckt) och en nolla betyder tonuppehåll (nyckeln ej nedtryckt). Beroende på vad man vill nyckla väljs en ton eller en digital utgång. En etta kan alltså antingen lägga ut en ton eller en ”etta” på en portpinne. Vi lägger ut en ton.

Det första som måste göras är att omvandla vår text till CW och koda den så att vi kan lägga in den i programmet. Det finns program som ordnar saken men vi gör det manuellt.

Vi tar ordet HEJ som exempel.
HEJ = …. . .--- och det skrivs om till ettor och nollor enligt specen ovan.

1	0	1	0	1	0	1	0	0	0	1	0	0	0	1	0	1	1	1	0	1	1	1	0	1	1	1	0	0	0	0	0

Vi har en 8-bitars processor och då måste vi alltså spara datan med multiplar på 8 och det blir i det här fallet 4 st. 8-bitars tal där sista talet fylls ut med nollor.
De 4 talen är:
10101010 = AA_{HEX}
00100010 = 22_{HEX}
11101110 = EE_{HEX}
11100000 = $E0_{HEX}$

Implementering

Hur gör vi nu för att få ut CW av AA22EEE0 ?

Huvudprogram och initiering ser ut som följer.

```
#define buzzer 2              //Summern kopplas till Digitalutgång 2

void setup()
{
}
void loop()
{
  sendCW("\xAA\x22\xEE\xE0 ");

  while(1){}
}
```

Nytt här är sendCW(). sendCW() är en sk. funktion. Vi har tidigare stött på funktionerna void setup(), void loop() och Serial.begin(9600).
En funktion är en programslinga som anropas (körs) genom att man skriver namnet och bifogar ev. data inom parenteserna. En funktion kan även returnera data.

Vår funktion heter sendCW och vill ha text att sändas bifogad som en sträng bestående av hexadecimala data. Hexadecimala tal skrivs med \x före talet.
Datan läses mha. pekare. Pekare anses ofta som svårt att förstå men vi behöver inte gå in på djupet utan det räcker med följande förklaring. Pekaren pekar på den bifogade strängens första tecken, i det här fallet AA_{HEX}. Ökas pekaren pekas nästa värde ut i strängen, 22_{HEX}.
Det som återstår att göras är att ta fram ettor och nollor ur den bifogade datan. Det görs genom en skiftfunktion. Skift är enkelt och kraftfullt. Det är dessutom en grundläggande funktion i alla datorer oavsett slag. Det finns vänster och högerskift. Vi skiftar till vänster.

Vi tar första byten AA_{HEX} som exempel för att visa skiftoperationen. AA_{HEX} = 10101010 binärt.
Vid skift flyttas alla bitar ett steg till vänster och nollor fylls på från höger.
Ex.
10101010 skiftas åt vänster med 1 position
01010100 skiftar vi en gång till fås
10101000 osv…

För att veta om summern ska vara på eller av måste vi nu läsa av ettor och nollor. Det görs genom **bitvis OCH**. Vi repeterar OCH-funktionen

AND		f
0	0	0
0	1	0
1	0	0
1	1	1

Vi får en ”etta” endast om båda ingående talen är ettor!

Om vi applicerar och-funktionen på datan med masken 10000000 fås följande:
(En mask är det vi sätter in att jämföra mot)

1	0	1	0	1	0	1	0
&	&	&	&	&	&	&	&
1	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0

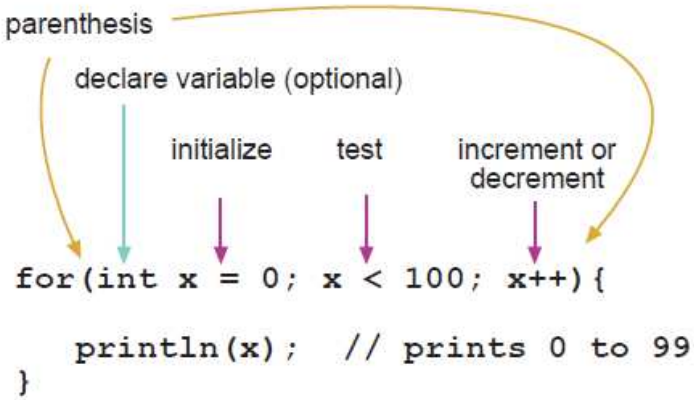
Vi får svaret 10000000 efter att ha kört OCH-funktionen på varje bit. Skiftas nu 1010101010 åt vänster får vi enligt ovan 01010100. Vi kör samma OCH-funktion igen och får:

0	1	0	1	0	1	0	0
&	&	&	&	&	&	&	&
1	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

Vi får svaret 00000000 eller enklare 0.

Mha. en if-sats kan vi nu ta reda på om det är en etta eller nolla vi har på första positionen i talet men det behövs även en räknare för att hålla ordning på vilken bit som behandlas.

for-satsen



for-satsen används för att upprepa kommandon ett bestämt antal gånger, som i exemplet ovan 99 gånger.
X är från början satt till noll.
X jämförs med 100 och om X är mindre än 100 utförs instruktionen inom måsvingarna – annars inte. Efter att instruktionen println är utförd ökas X med 1 (X++ är samma som X = X + 1). X är alltså lika med 2 vid andra varvet i loopen.

while-satsen

```
while(n)
{
}
```

while-satsen utförs om det inom parenteserna är sant. Är n sant utförs det inom måsvingarna, annars inte.

Nu kan vi titta på sendCW-funktionen.

```
void sendCW(const char* s)              //Bifoga en 8-bitars sträng
{
  unsigned char i, temp;

  while(*s)                             //Läs tecken för tecken tills strängen är slut
  {
    temp = *s;                          //temp = första tecknet i strängen
    for(i = 0; i < 8; i++)              //räkna från 0 till 7, alltså 8 steg
    {
      if(temp & 0x80)                   //maska ut första biten i strängen och
      {
        tone(buzzer,800);               //om resultatet är ett – summer på
      }
      Else                              //annars
      {
        noTone(buzzer);                 //Stäng av summer
      }
      temp <<= 1;                       //Skifta ett steg åt vänster
      delay(70);                        //Fördröjning – CW-hastighet
    }
    *s++;                               //Öka pekaren med ett = nästa tecken i strängen
  }
  noTone(buzzer);                       //Stäng av summer
}//sendCW----------
```

Det kompletta programmet finns på sista sidan.

```
#define buzzer 2              //Summern kopplas till Digitalutgång 2

void setup()
{
}
void loop()
{
  sendCW("\xAA\x22\xEE\xE0");

  while(1){}
}

void sendCW(const char* s)              //Bifoga en 8-bitars sträng
{
  unsigned char i, temp;

  while(*s)                             //Läs tecken för tecken tills strängen är slut
  {
    temp = *s;                          //temp = första tecknet i strängen
    for(i = 0; i < 8; i++)              //räkna från 0 till 7, alltså 8 steg
    {
      if(temp & 0x80)                   //maska ut första biten i strängen och
      {
        tone(buzzer,800);               //om resultatet är ett – summer på
      }
      else                              //annars
      {
        noTone(buzzer);                 //Stäng av summer
      }
      temp <<= 1;                       //Skifta ett steg åt vänster
      delay(70);                        //Fördröjning – CW-hastighet
    }
    *s++;                               //Öka pekaren med ett = nästa tecken i strängen
  }
  noTone(buzzer);                       //Stäng av summer
}//sendCW----------
```