Källkod

```
#include <avr/io.h>
#include <stdint.h>
#include <avr/interrupt.h>
#include <stdbool.h>

//Setup of the sinus wave, the size of the array as well as an index counter
    .
int sineWaveSampleRate = 30;
int currentIndex = 0;
int sineWaveValues[30] = {62, 61, 59, 56, 52, 47, 41, 34, 28, 21, 16, 10, 6,
    3, 1, 0, 1, 3, 6, 10, 16, 21, 28, 34, 41, 47, 52, 56, 59, 61};

//Boolean value to keep track of whether frequency or volume should be read
   next by the ADC
bool readFrequencyNext = true;

uint8_t frequency = 0;
uint8_t volume = 0;

//Declaration of initialization-methods
void adc_init();
void timer0_init();
void adc_read_frequency();

int main(void)
{
  //Run initialization-methods
  timer0_init();
  adc_init();

  //Enable interrupts
  sei();

  //Start the first AD-conversion.
  adc_read_frequency();

  while (1)
  {
    //Empty while-loop
  }

  return 0;
}

//ADC initialization-method
void adc_init() {
  //Enables the ADC, including enabling the interrupt and
```

```
  //setting it to work at 1/128 of the processors clock frequency
  ADCSRA = 0b10001111;
  ADMUX |= (1 << ADLAR); //Turning ADLAR to get 8-bit values from the ADC
   instead of 10-bit values
  DDRA = 0b00001100; //Sets data direction for the I/O pins on PORT A
  DDRD = 0xff; //Sets the data direction for the I/O pins on PORT D
}

//Timer initialization-method
void timer0_init() {
  TCCR0A = 0x00; //Sets the counter value to zero
  TCCR0B = 0b00000010; //Sets the prescaler to a division factor of 8
  TIMSK0 = 0b00000010; //Enables Compare Match interrupt
}

void adc_read_frequency()
{
  //Set so that the frequency is the first thing to be read
  readFrequencyNext = true;
  //Changes the ADMUX register to read from the correct potentiometer
  ADMUX |= (0 << MUX0);
  //Starts the conversion process and waits for interrupt.
  ADCSRA |= (1 << ADSC);
}

//Interrupt routine for the ADC
ISR (ADC_vect) {
  //If the frequency is to be read, set the frequency, then change so that
   the volume is read the next time
  if (readFrequencyNext) {
    frequency = ADCH;
    readFrequencyNext = false;
  }
  else
  {
    volume = ADCH/64;
    readFrequencyNext = true;
  }

  //Toggle the channel from which to read and start the next conversion
  ADMUX ^= (1<<MUX0); //Switch channgel
  ADCSRA |= (1<<ADSC); //Start conversion
}

//Interrupt routine for the counter which decides the frequency of the
   output signal
ISR (TIMER0_COMPA_vect)
{
```

```
  //Set the next value to which the counter will count before triggering the
     next interrupt
  OCR0A = frequency;

  //Sets PORT D to the value in the current index of the sineWaveValues
   array
  PORTD = volume*sineWaveValues[currentIndex];

  //Go to the next value in the sin-wave-array
  currentIndex = (currentIndex + 1) % (sineWaveSampleRate);

  TCNT0 = 0x00; //Resets the timer
}
```