```
/*
   Simple DDS Signal Generator
   2017/6/20 by morecat_lab
   based on
   http://interface.khm.de/index.php/lab/interfaces-advanced/arduino-dds-sinewave-generator/
   KHM 2009 /  Martin Nawrath
   Kunsthochschule fuer Medien Koeln
   Academy of Media Arts Cologne
*/

/***
   dk2jk 04 2020
   modifiziert
   nur ein ausgang
   CTCSS frequenzen wie MX-315 encoder
   Kanalwahl durch pins[10:5] entsprechend CX-315 pins [6:1]
   nach Frequenztabelle aus CX-315 Datenblatt "cx_315_v1.h"
*/

#include "avr/pgmspace.h"
#include "Arduino.h"
#include "cx_315_v1.h" //kanaltabelle wie MX-315 decoder
#define PTT      2 // CTCSS einschalten
#define SINOUT   3 // CTCSS ausgang
#define TRIGGER  A5 // trigger sinus
int kanal_pin[] = { 5, 6, 7, 8, 9, 10}; // CTCSS Kanal Code
#define LED      13

#include "sinus.h" // sinus[]

#define cbi(sfr, bit) (_SFR_BYTE(sfr) &= ~_BV(bit))
#define sbi(sfr, bit) (_SFR_BYTE(sfr) |= _BV(bit))
#define REFCLK (31376.6)

// fuer interrupt routine volatile !
volatile unsigned long phase_accu;
volatile unsigned long phase_increment;
volatile byte          phase_index;


static inline void disable_timer0() {
  cbi (TIMSK0, TOIE0);
  // disable Timer0 !!! delay() is now not available
  // damit 1ms -IRQ nicht stoert !!!
}
static inline void enable_timer() {
  sbi (TIMSK2, TOIE2);
}
static inline void disable_timer() {
  cbi (TIMSK2, TOIE2);
}

byte liesKanal()
{ byte y = 0;
  int i;
  for (i = 5; i >= 0; i--)
  { y = y + (digitalRead(kanal_pin[i]) << i);
  }
  return (y & 0x3f);
}

float code_to_frequenz(byte code)
{ // in frequenztabelle nach code suchen
  int i;
  bool gefunden = false;
  for (i = 0; i < TABELLENLAENGE; i++)
  { if ( frequenztabelle[i].code == code)
    { gefunden = true;
      break;
    }
  }
  return gefunden ? frequenztabelle[i].fq : 1000.0;
}

unsigned long tick(int i)
{ return pow(2, 32) * code_to_frequenz(i) / REFCLK;
}

void setup_SineFreq(int fq_index) {
  disable_timer();
  phase_increment = tick(fq_index);
  phase_accu = 0;
  enable_timer();
}

void Setup_timer2() {
  // set prscaler to 1, PWM mode to phase correct PWM,  16000000/510 = 31372.55 Hz
  clock
  TCCR2A = (1 << COM2A1) | (0 << COM2A0) | ( 1 << COM2B1) | ( 0 << COM2B0) | ( 0 <<
  WGM21) | ( 1 << WGM20);
  // Timer2 Clock Prescaler to : 1 =>  16000000/510 = 31372.55 Hz clock
  TCCR2B = (0 << WGM22) | (0 << CS22) | ( 0 << CS21) | ( 1 << CS20);
}

// Timer2 Interrupt Service at 31372,550 KHz = 32uSec
ISR(TIMER2_OVF_vect) {
  phase_accu = phase_accu + phase_increment; // soft DDS, phase accu with 32 bits
  phase_index = phase_accu >> 24;   // use upper 8 bits for phase accu as frequency
  information
  // read value fron ROM sine table and send to PWM DAC
  OCR2B = pgm_read_byte_near(sinus + phase_index);
  if (OCR2B < sinus[0])
  { // output digital by PWM info // compare a = 128...255
    digitalWrite(TRIGGER, HIGH);
  } else {
    digitalWrite(TRIGGER, LOW);
  }
}

void setup()
{ pinMode(LED, OUTPUT);
  pinMode(SINOUT, OUTPUT);
  pinMode(TRIGGER, OUTPUT);
  pinMode(PTT, INPUT); // high active
  for (int i = 0; i < 6; i++)
  { pinMode(kanal_pin[i], INPUT_PULLUP);
  }
  disable_timer0();
  Setup_timer2();
  setup_SineFreq(63);
}

void loop() {
  static byte alt = 0;
  static byte neu = 1;
  static bool en_alt = true;
  static bool en_neu = false;
  neu = liesKanal();
  if ( neu == alt)
  { // nix zu tun
  }
  else
  { setup_SineFreq( neu);
    alt = neu;
  }
  en_neu = digitalRead(PTT);
  if ( en_neu == en_alt)
  { //nix zu tun
  }
  else
  { en_alt = en_neu;
```

```
      if (en_neu == 0)
      { disable_timer();
      }
      if ( en_neu == 1)
      { enable_timer();
      }
    }
  }

```