```
// ------------------------------------------------------------
//          HUSK RETTE FILNAVN OG DATO i ver 1 og ver 2          -
char ver1[ ] = "DDS_2_VFO_v_0_24";  // FilNavn til startdisplay    -
char ver2[ ] = "OZ1LN_26_06_2023";  // OZ1LN+Dato til startdisplay -
// ------------------------------------------------------------
// 15-04-2023-010 Formateret udskrift af frekvenser til display
// 06-06-2023-011 Ændret kanalstep, start og udskrift. VFO og RX frekvens vises
// 23-06-2023-019 Ændret så frekvensvalg sker på RX-frekvens, og øvrige
frekvenser beregnes.
// 26-06-2123-021 Sammenskriv af AD9851 og Si5351 sketch, til nyt testprint



// OBS OBS Der er ikke tjek på om syntesen er i lås. Fejlstart er set ;o(

#include <LiquidCrystal.h>
LiquidCrystal lcd(12, 11, 10, 9, 8, 7);
// Display forbundet med D12 til RS, D11 til E, RW til stel.
// D10 til display D4, D9 til D 5, D8 til D6 og D7 til D7.

// ---------- Kun til DSS Si5351 ------------------------------|
#include "si5351.h"        // hjælpebibliotek til Si5351        |
Si5351 si5351;             //                                   |
#define XT_CAL_F  139000   // Si5351 Krystal frekvens justering |
// Si5351 justeres til at give nøjagtig frekvens.               |
// Forøgelse vil forminske udgangsfrekvensen og omvendt. 139000 |
// ------------------------------------------------------------|

// deklaration af pins ----------------------------------------
const int dds_9851_LOAD    = A1;  // AD9851
const int dds_9851_CLOCK   = A2;  // AD9851
const int dds_9851_DATA    = A3;  // AD9851

const int KNAP_1  = 6;  // D6
const int KNAP_2  = 5;  // D5
const int KNAP_3  = 4;  // D4
const int KNAP_4  = 3;  // D3
const int KNAP_5  = 2;  // D2
const int KNAP_6 = 14;  // A0

const int blinkled = 13;     // D13 pin 13 er LED på NANO

boolean Knap1 = 0;
boolean Knap2 = 0;
boolean Knap3 = 0;
boolean Knap4 = 0;
boolean Knap5 = 0;
boolean Knap6 = 0;

long cal = XT_CAL_F; // bruges til finjustering af Si5351 output frekvens

unsigned long freq_valg  = 145500000; // Start frekvens 145,500 MHz
unsigned long freq_mf     = 10700000; // mellemfrekvens 10700000 Hz
unsigned long freq_space  =   600000; // repeaterspace  00600000 Hz
unsigned long freq_step1  =    12500; // Kanalstep 12,5 kHz
unsigned long freq_step2  =  1000000; // Kanalstep 1 MHz
unsigned long last_freq = freq_valg + 12500; // Display og AD opdateres ved start
unsigned long freq_lcd;
unsigned long freq_dds;
// ------------------------------------------------------
// Setup() kører kun 1 gang ved programstart
// ------------------------------------------------------
void setup()
{
  // DEFINE INPUTs
  pinMode (KNAP_1, INPUT_PULLUP);
```

```
  pinMode (KNAP_2, INPUT_PULLUP);
  pinMode (KNAP_3, INPUT_PULLUP);
  pinMode (KNAP_4, INPUT_PULLUP);
  pinMode (KNAP_5, INPUT_PULLUP);
  pinMode (KNAP_6, INPUT_PULLUP);

  // DEFINE OUTPUTs
  pinMode (dds_9851_DATA,  OUTPUT);
  pinMode (dds_9851_CLOCK, OUTPUT);
  pinMode (dds_9851_LOAD,  OUTPUT);

  lcd.begin(16, 2);              // start 16*2 display
  lcd.setCursor(0, 0);           // øverste linje
  lcd.print(ver1);               // print ver1
  lcd.setCursor(0, 1);           // nederste linje
  lcd.print(ver2);               // ver2

  delay(2000);                   // vent 2 sekunder

  lcd.setCursor(0, 0);           // øverste linje
  lcd.print("               "); // Display ryddes
  lcd.setCursor(0, 1);           // nederste linje
  lcd.print("               "); // Display ryddes

  pinMode(blinkled , OUTPUT);   // LED på NANO
  digitalWrite(blinkled , LOW);

// ---------- Kun til DSS Si5351 ----------------------------------|
si5351.init(SI5351_CRYSTAL_LOAD_8PF, 0, 0);     // Start af Si5351 |
si5351.set_correction(cal, SI5351_PLL_INPUT_XO); // Frkv justering |
si5351.drive_strength(SI5351_CLK0, SI5351_DRIVE_2MA); // out power |
si5351.output_enable(SI5351_CLK0, 1);   // 1 - Aktiv / 0 - Slukket |
si5351.output_enable(SI5351_CLK1, 0);   // 1 - Aktiv / 0 - Slukket |

si5351.output_enable(SI5351_CLK2, 0);   // 1 - Aktiv / 0 - Slukket |
// ---------- Kun til DSS Si5351 ----------------------------------|

}
// ---------------------Her begynder funktioner-------------
// Test_Input()
// ---------------------------------------------------------
void Test_Input()
{
  Knap1  = !digitalRead(KNAP_1); // input inverteres
  Knap2  = !digitalRead(KNAP_2); // input inverteres
  Knap3  = !digitalRead(KNAP_3); // input inverteres
  Knap4  = !digitalRead(KNAP_4); // input inverteres
  Knap5  = !digitalRead(KNAP_5); // input inverteres
  Knap6  = !digitalRead(KNAP_6); // input inverteres

 if (Knap1 == 1)
   {
    freq_valg = freq_valg + freq_step1; // step1 er + 12.5 kHz
    delay(300);  // simpel debounce
   }
 if (Knap2 == 1)
   {
    freq_valg = freq_valg - freq_step1; // step1 er - 12.5 kHz
    delay(300);  // simpel debounce
   }
 if (Knap3 == 1)
   {
    freq_valg = freq_valg + freq_step2; // step2 er + 1 MHz
    delay(300);  // simpel debounce
   }
 if (Knap4 == 1)
```

```
    {
     freq_valg = freq_valg - freq_step2; // step2 er - 1 MHz)
     delay(300);  // simpel debounce
    }
 if (Knap5 == 1)
    {
     freq_valg = 145000000;    // sæt frekvens til 145,000 MHz
     delay(300);  // simpel debounce
    }
  if (Knap6 == 1)
    {
     freq_valg = 145500000;    // sæt frekvens til 145,500 MHz
     delay(300);  // simpel debounce
    }
  }

// ---------------------------------------------------------
  void format_lcd()
    // Frekvensen formatteres med 2 decimal punkter, MHz and KHz
    {
    char str[12];   // Split frekvens-strengen op, indsæt punkter
    sprintf(str, "%010lu", freq_lcd);
    str[0] = str[1];
    str[1] = str[2];
    str[2] = str[3];
    str[3] = '.';
    str[4] = str[4];
    str[5] = str[5];
    str[6] = str[6];
    str[7] = ' ';
    str[8] = 'M';
    str[9] = 'H';
    str[10] = 'z';
    str[11] = ' ';

    if (freq_lcd < 10000000) {str[0] = ' '; str[1] = ' ';}
    if (freq_lcd < 100000000) str[0] = ' ';

    lcd.print(str);
    }

// ---------------------------------------------------------
// Vis VFO frekvenser i LCD display
// ---------------------------------------------------------
void vis_frekvens()
{
  lcd.clear();

  lcd.setCursor(0, 0);
  lcd.print("RX  ");
  freq_lcd = freq_valg;
  format_lcd();

  lcd.setCursor(0, 1);
  lcd.print("VFO ");
  freq_lcd = freq_dds;
  format_lcd();

}
// ---------------------------------------------------------------

void dds_5351()
// data freq_dds * 100 sendes til Si5351
{
  si5351.set_freq(freq_dds * 100ULL, SI5351_CLK0);
}
```

```
// ------------------------------------------------------------

void dds_9851()
// dds instruction write - takes unsigned long frequency in Hz
{
  int last8;
  unsigned long DDSLong;
  unsigned long Bitmask32 = 1;
  // 32 bit bit mask '0000 0000 0000 0000 0000 0000 0000 0001'

  byte Bitmask8 = 1;           // 8 bit bit mask '0000 0001'
  // shift bitmasks left 1 bit at a time and AND them bitwise with a value
  // to simply determine if an unknown individual bit within a data
  // structure is a 1 or a 0

  byte FirstBit = 1;

  float clock_frequency = 179983220;
  // Dette er krystalfrekvensen på AD9851 modulet. 180000000
  // Tilpasses, så udgangsfrekvensen er korrekt, målt med en god frekvenstæller.
  // Højere værdi giver lavere udgangsfrekvens.
  // ved test afgjort at der ved dette modul skal stå 1179983200 for at passe.

  float twoE32 = pow (2, 32); // this is 2 to the power of 32 (which is quite a
lot)

  DDSLong = ((twoE32 * (freq_dds)) / clock_frequency);
  // this calculates the first 32 bits of the 40 bit DDS instruction

  // now we itterate through the first 32 bits one at a time, determine if the
  // individual bits are 1 or 0 and write a HIGH or LOW as appropriate

  for (Bitmask32 = 1; Bitmask32 > 0; Bitmask32 <<= 1)
   // iterate through 32 bits of DDSLong
  {
    if (DDSLong & Bitmask32)                  // if bitwise AND resolves to true
      digitalWrite(dds_9851_DATA, HIGH);
    else                                      // if bitwise AND resolves to false
      digitalWrite(dds_9851_DATA, LOW);

    // after every single bit toggle clock pin for the DDS to receive the data
bit

    digitalWrite(dds_9851_CLOCK, HIGH);
    // Clock data in by setting clock pin high then low
    delayMicroseconds(1);
    digitalWrite(dds_9851_CLOCK, LOW);

  }

  // now send the final 8 bits to complete the 40 bit instruction
  // the AD9851 datasheet explains this well, but we need 1000 0000
  // because we want to use the clock multiplier
  // so here we are going to look 8 times and send a 1 the first time round then
7 0s
  // 10 out of 10 for niftyness but 0 out of 10 for readability:

  for (Bitmask8 = 1; Bitmask8 > 0; Bitmask8 <<= 1)
  { // iterate through last 8 bits of 40 bit instruction to DDS
    if (Bitmask8 & FirstBit)
      // 1.st bit of remaining 8 needs to be 1 to enable clock multiplier
      digitalWrite(dds_9851_DATA, HIGH);
    else
      digitalWrite(dds_9851_DATA, LOW);

    // after every single bit toggle clock pin for DDS to receive the data bit
```

```
    digitalWrite(dds_9851_CLOCK, HIGH);
    // Clock data in by setting clock pin high then low
    delayMicroseconds(1);
    digitalWrite(dds_9851_CLOCK, LOW);
  }

  // and once all 40 bits have been sent
  // finally we toggle the load bit to say we are done
  // and let the AD9851 do its stuff

  digitalWrite (dds_9851_LOAD, HIGH);   // Pulse DDS update
  delayMicroseconds(1);
  digitalWrite (dds_9851_LOAD, LOW);    // to execute previous instruction set

    return;
}

//************************************************************
// HOVEDPROGRAM. Her står den kode, som kører i uendelig sløjfe.
//************************************************************
void loop()
{
  Test_Input();

  if (freq_valg != last_freq)  // Hvis freq er ændret opdateres DDS og Display
  {
    freq_dds = (freq_valg / 3); // Her kommer regnestykker for MF, Space etc

    vis_frekvens();       // skriv frekvenser i display

    dds_9851();            // send DDS kodestreng til AD9851
    dds_5351();            // send DDS kodestreng til Si5351

    digitalWrite(13, HIGH); // Blink med LED for at vise opdateringen
    delay(50);
    digitalWrite(13, LOW);

    last_freq = freq_valg ;    // sæt last_freq lig med freq_valg
  }
}
// ------Program slut--------------------
```