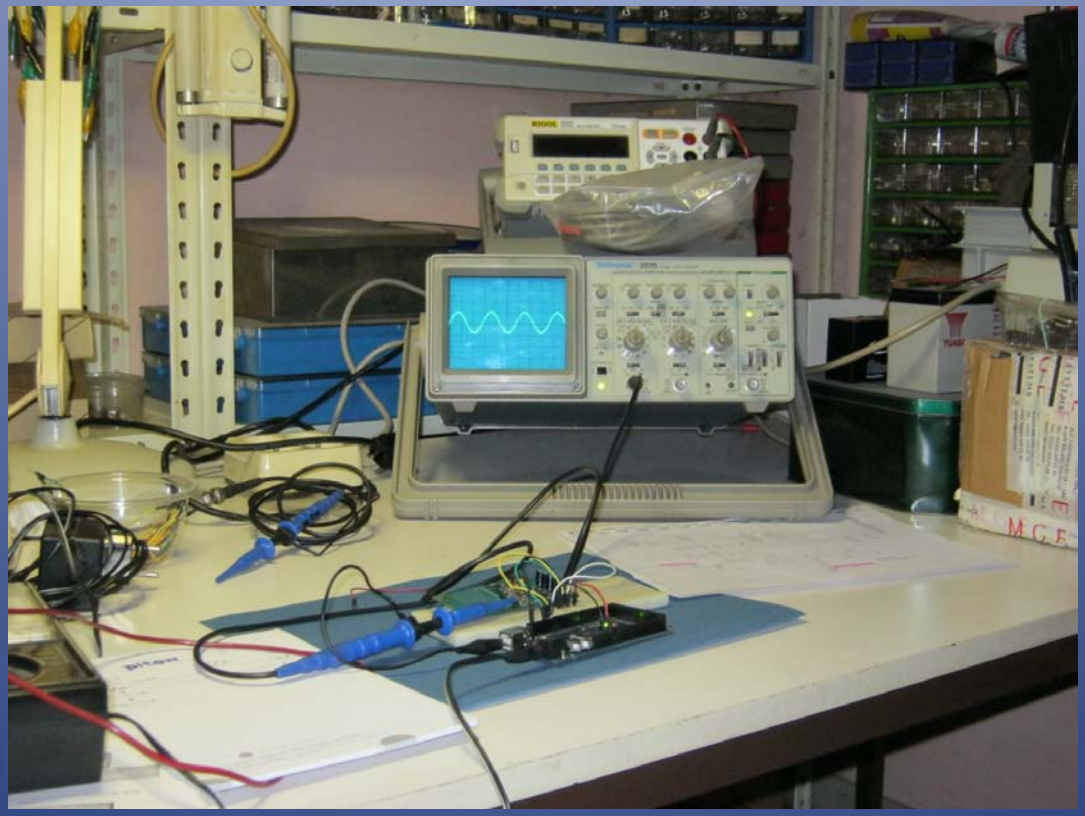
www.arduino.cc

Arduino en DDS

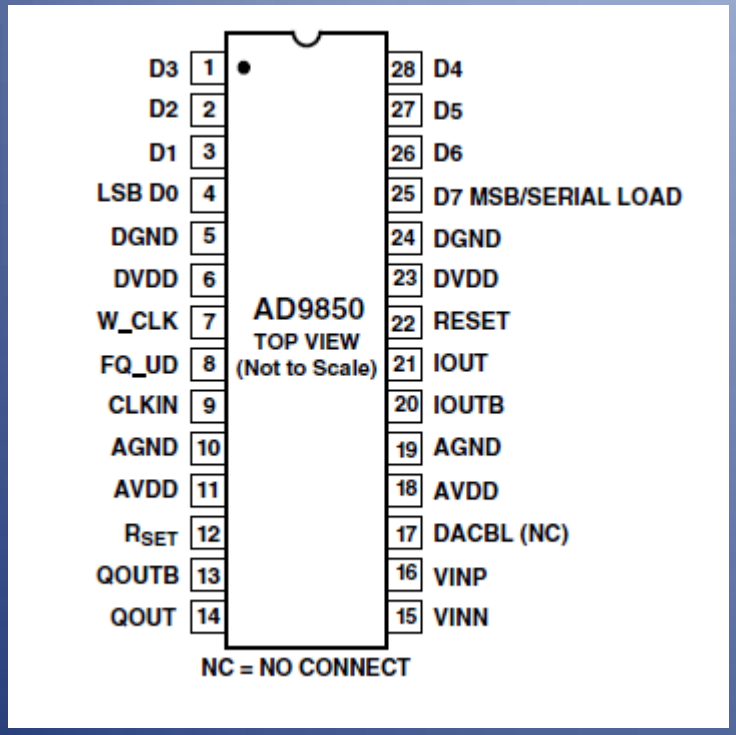


DDS chips

- ✓ DDS = Direct Digital (frequency) Synthesis
- ✓ Output = sinusvormig signaal
- ✓ Maximum frequentie = ½ klokfrequentie
- ✓ Frequentie bepaald door ‘tuning word’
- ✓ Grootste fabrikant: Analog Devices (AD9xxx)

DDS chips

Voorbeeld: AD9850 (Analog Devices)



CLKIN = systeemklok, max. 125 MHz

DGND, DVDD: voeding digitaal

AGND, AVDD: voeding analoog

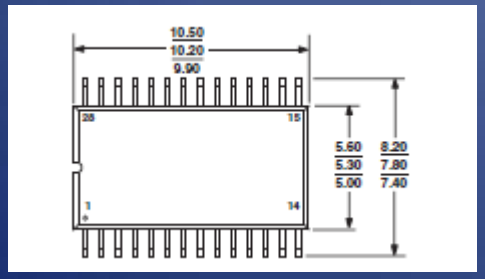
D7, W_CLK, FQ_UD
tuning word upload (32 bits)

IOUT = analoge signaaluitgang

Max. frequentie: 125/2 = 62,5 MHz

Frequentiestap: tot 0,0291 Hz

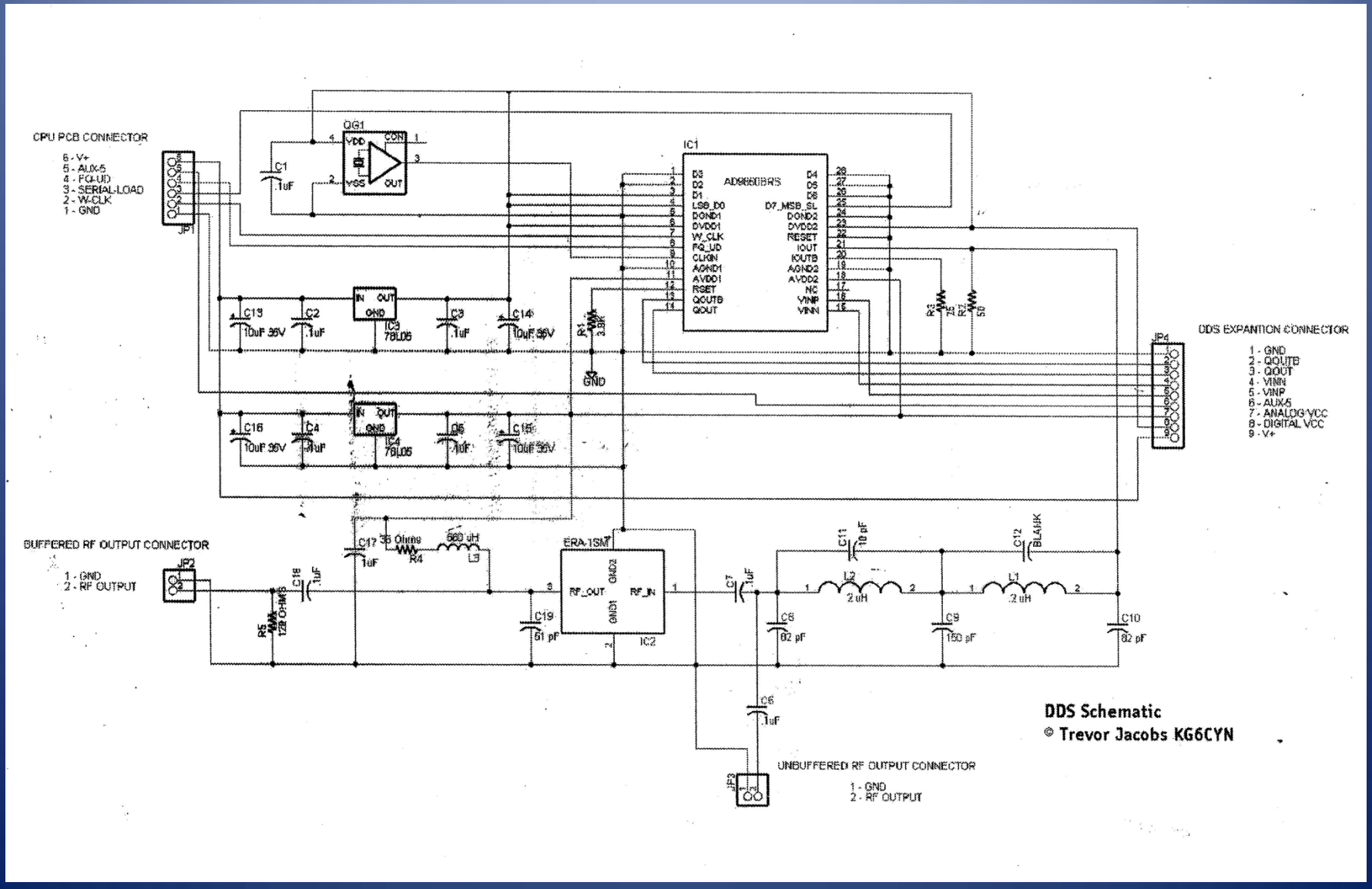
Frequentievariatie: tot 23 M per seconde



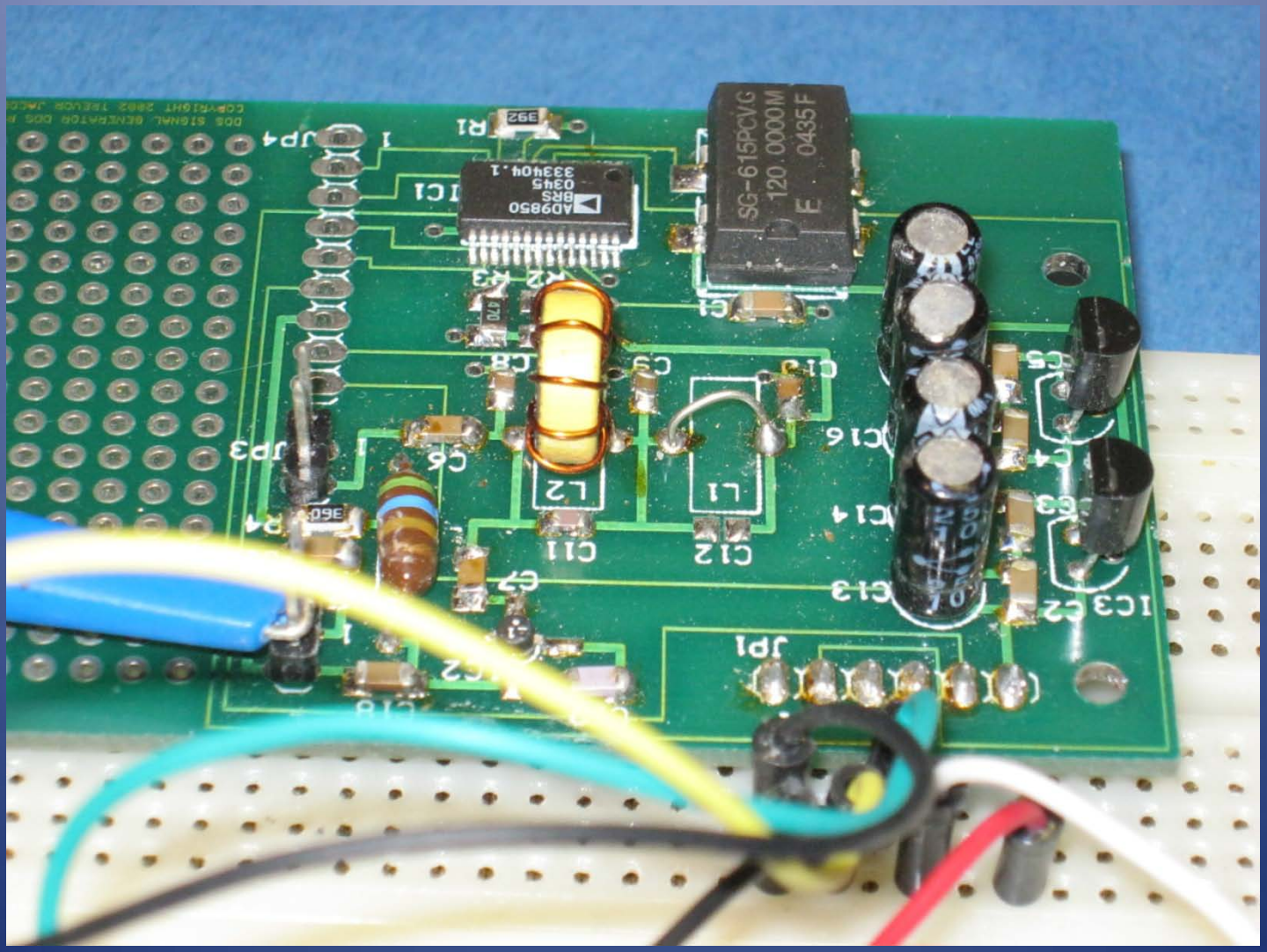
28-SSOP (Shrink Small Outline Package)

DDS chip AD9850

Voorbeeld DDS-board



DDS chip AD9850



DDS chip AD9850

Het 'tuning word' Δ Phase bepaalt de uitgangsfrequentie:

$$f_{OUT} = (\Delta \text{ Phase} \times CLKIN) / 2^{32}$$

Of:

$$\Delta \text{ Phase} = (f_{OUT} \times 2^{32}) / CLKIN$$

Δ Phase = waarde van het tuning word
CLKIN = frequentie van de systeemklok (bv. 120 MHz)
f_{OUT} = uitgangsfrequentie

Voorbeeld voor 3,6 MHz:

Δ Phase = $(3{,}6 \times 2^{32}) / 120$

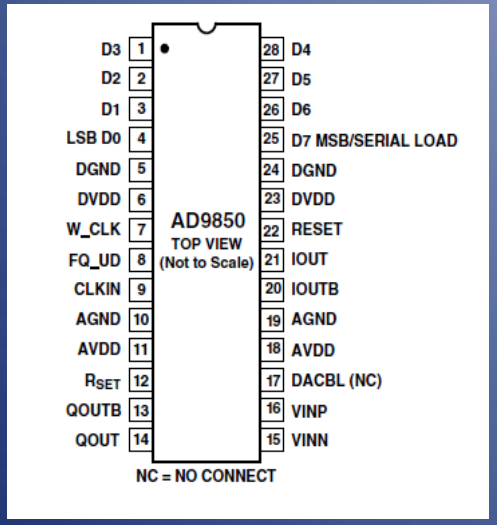
Δ Phase = 128849018,88 ~ 128849019

fOUT = (128849019 x 120) / 232 = 3,600000003

Tuning woord binair: 11110101110000101000111011

DDS chip AD9850

Hoe wordt het ‘tuning word’ geladen?



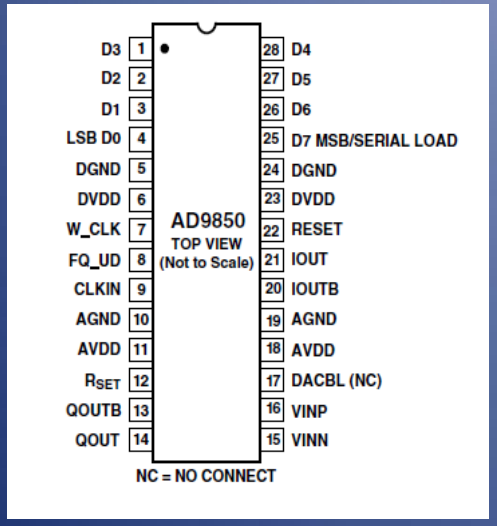
2 mogelijke manieren

✓ **Parallel: 5 x 8 bits via D0-D7 pennen. Wordt hier niet verder beschouwd.**

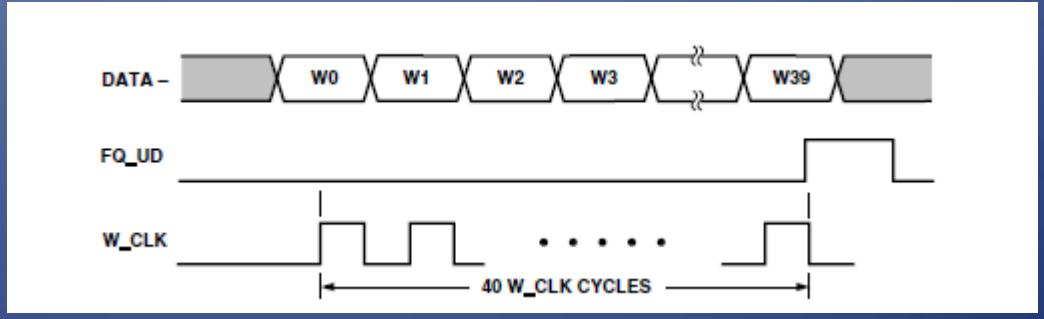
✓ **Serieel: 40 bits, bit per bit via D7, in de juiste volgorde en met de juiste timing.**

DDS chip AD9850

Serieel laden van het tuning woord: volgorde en timing



1. 40-bit data bit per bit laden via pen 25 D7
2. Volgorde LSB (least significant bit) first: laagstbeduidende bit eerst
3. Timing: 1 bit per W_CLK puls, na 40 bits FQ_UD puls



DDS chip AD9850

Serieel laden van het tuning woord: 40-bit?

W0	Freq-b0 (LSB)	W14	Freq-b14	W28	Freq-b28
W1	Freq-b1	W15	Freq-b15	W29	Freq-b29
W2	Freq-b2	W16	Freq-b16	W30	Freq-b30
W3	Freq-b3	W17	Freq-b17	W31	Freq-b31 (MSB)
W4	Freq-b4	W18	Freq-b18	W32	Control
W5	Freq-b5	W19	Freq-b19	W33	Control
W6	Freq-b6	W20	Freq-b20	W34	Power-Down
W7	Freq-b7	W21	Freq-b21	W35	Phase-b0 (LSB)
W8	Freq-b8	W22	Freq-b22	W36	Phase-b1
W9	Freq-b9	W23	Freq-b23	W37	Phase-b2
W10	Freq-b10	W24	Freq-b24	W38	Phase-b3
W11	Freq-b11	W25	Freq-b25	W39	Phase-b4 (MSB)
W12	Freq-b12	W26	Freq-b26		
W13	Freq-b13	W27	Freq-b27		

Bits 0-31 = tuning word (frequentie)

Bits 32-39 zijn controlebits: op 0 plaatsen (voor AD9850)

SWEEP9850

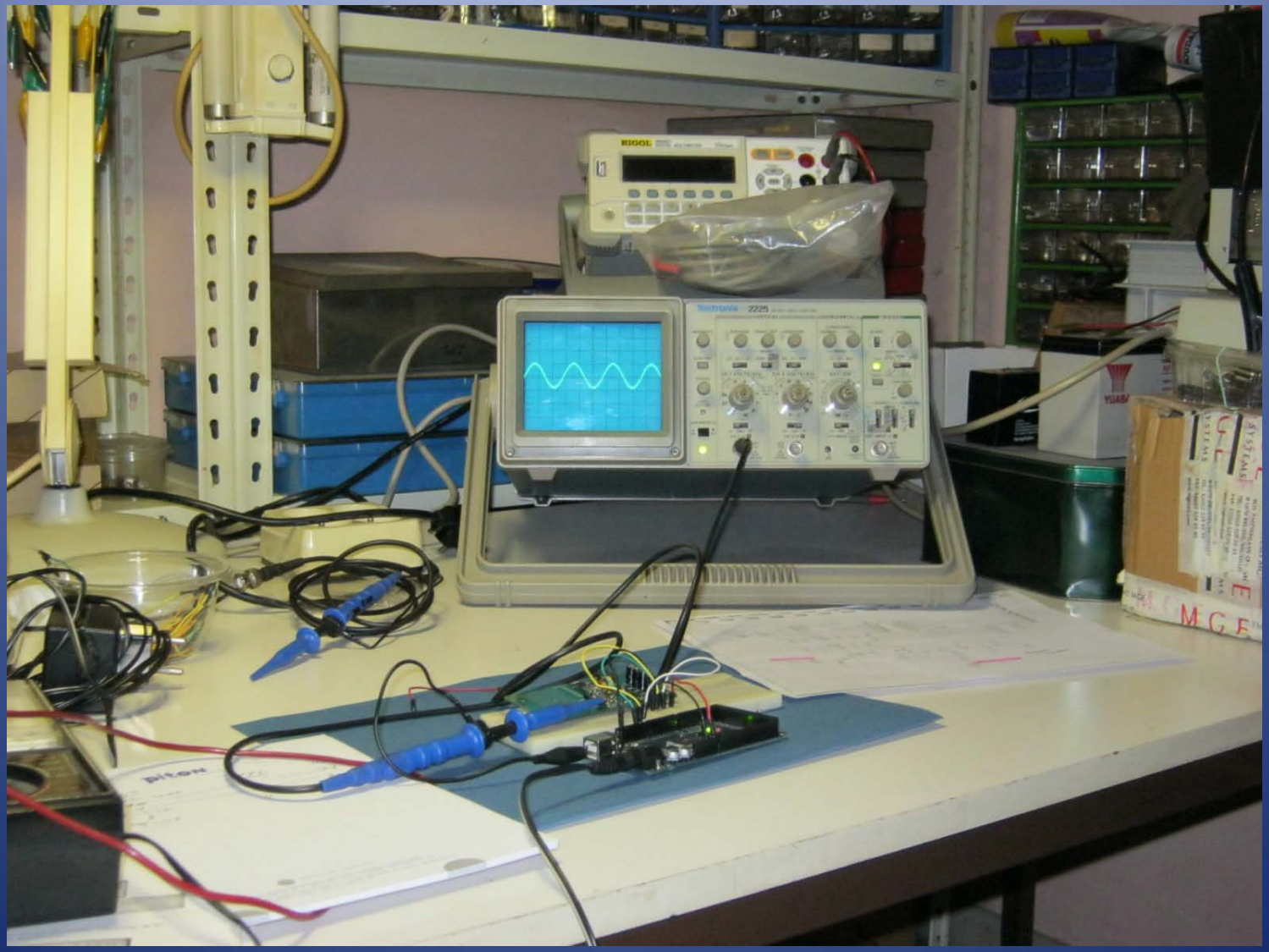
Opdracht:

Genereer frequenties tussen 1 en 10 kHz

Hardware setup

1. Verbind FQ_UD (pen 8) van de AD9850 met I/O 8 van de Arduino
2. Verbind W_CLK (pen 7) van de AD9850 met I/O 9 van de Arduino
3. Verbind D7 (pen 25) van de AD9850 met I/O 10 van de Arduino
4. Sluit een scoop aan op Iout (pen 21) van de AD9850
5. Verbind de Arduino met de USB-poort van de PC

SWEEP9850



SWEEP9850

Genereer frequenties tussen 1 en 10 kHz

1. Start de Arduino IDE
2. Check Tools/Board en Tools/Serial
3. Voer de volgende lijnen in:

```
// Frequentiesturing van een AD9850 DDS door het berekenen en serieel laden van een 40-bit datawoord
#define DDS_CLOCK 120000000    // frequentie van de DDS-klok (in Hz)
byte LOAD = 8;              // I/O 8 is verbonden met FQ_UD van de DDS
byte CLOCK = 9;             // I/O 9 is verbonden met W_CLK van de DDS
byte DATA = 10;             // I/O 10 is verbonden met D7 van de DDS
```

- ✓ Alles na // is commentaar (veelvuldig gebruiken!)
- ✓ #define DDS_CLOCK 120000000: tijdens de compilatieslag zal DDS_CLOCK in de broncode vervangen worden door 120000000, de frequentie van de systeemklok.
- ✓ Byte LOAD = 8 → LOAD is de naam voor de bytewaarde 8

SWEEP9850

4. Voeg de volgende regels toe:

```
void setup()              // deze instructies worden eenmaal
uitgevoerd
{
// Plaats alle I/O pennen in OUTPUT mode
  pinMode (DATA, OUTPUT);
  pinMode (CLOCK, OUTPUT);
  pinMode (LOAD, OUTPUT);
}
```

- ✓ pinMode() bepaalt of de Arduino I/O-poort in kwestie zich moet gedragen als input- of outputpoort. In dit geval sturen al deze poorten data naar de AD9850, dus OUTPUT.
- ✓ Deze instructies worden <u>eenmaal</u> uitgevoerd

SWEEP9850

5. Voeg de volgende regels toe:

```
void loop()              // deze instructies worden doorlopend
uitgevoerd
{
 // Doe een frequentiezwaai tussen 1 en 10 kHz in stappen van 1 Hz
 for(unsigned long freq = 1000; freq < 10000; freq++)
 {
  sendFrequency(freq);
  delay(2);
 }
}
```

- ✓ void loop() : deze instructies worden continu uitgevoerd
- ✓ for-loop:

  for (vanaf; test; stap)

  { instructies}

  Voer de instructies uit vanaf freq = 1000 zolang freq < 10000

  Verhoog freq met 1 na elke lusdoorloop

SWEEP9850

6. Voeg de volgende regels toe:

```
void sendFrequency(unsigned long frequency)
{
// Bereken het datawoord
  unsigned long tuning_word = (frequency * pow(2, 32)) / DDS_CLOCK;
// Zet W_CLK laag
  digitalWrite(CLOCK, LOW);
// Zet FQ_UD laag
  digitalWrite(LOAD, LOW);
// Klok de eerste 8 bits in het register van de DDS (W0-W7)
  shiftOut(DATA, CLOCK, LSBFIRST, tuning_word);
// Klok de volgende 8 bits (W8-W15)
  shiftOut(DATA, CLOCK, LSBFIRST, tuning_word >> 8);
// Klok de volgende 8 bits (W16-W23)
  shiftOut(DATA, CLOCK, LSBFIRST, tuning_word >> 16);
// Klok de volgende 8 bits (W24-W31)
  shiftOut(DATA, CLOCK, LSBFIRST, tuning_word >> 24);
// Klok de volgende 8 bits = 0 (W32-W39)
  shiftOut(DATA, CLOCK, LSBFIRST, 0x0);
// Zet FQ_UD hoog om het datawoord door te schuiven in de DDS (= nieuwe frequentie)
  digitalWrite(LOAD, HIGH); // Take load pin high again
}
```

SWEEP9850

unsigned long tuning_word = (frequency * pow(2, 32)) / DDS_CLOCK

Bereken het 'tuning word': Δ Phase = (f_{OUT} x 2^{32}) / CLKIN

digitalWrite(CLOCK, LOW)
digitalWrite(LOAD, LOW)

Plaats de poorten CLOCK en LOAD op LOW

shiftOut(DATA, CLOCK, LSBFIRST, tuning_word)

shiftOut: shifts out a byte of data one bit at a time. Starts from either the most (i.e. the leftmost) or least (rightmost) significant bit. Each bit is written in turn to a data pin, after which a clock pin is pulsed (taken high, then low) to indicate that the bit is available.

shiftOut(DATA, CLOCK, LSBFIRST, tuning_word >> n)

>> n: schuift het datawoord n bits naar rechts

shiftOut(DATA, CLOCK, LSBFIRST, 0x0)

Niet vergeten om de controlebits 32-39 (waarde 0) te laden

digitalWrite(LOAD, HIGH)

Ten slotte: FQ_UD hoog plaatsen om het datawoord door te schuiven

Datatypes

unsigned long ?
byte ?

datatype	bereik	opslag
boolean	false = 0 true = elke andere waarde	1 byte
byte	0-255	1 byte
int	-32768 32767	2 bytes
unsigned int	0 65535	2 bytes
long	-2,147,483,648 2,147,483,647	4 bytes
unsigned long	0 4,294,967,295	4 bytes
char	0-255	1 byte

Andere datatypes: zie reference manual (selecteer Help /Reference in IDE)

HOP9850

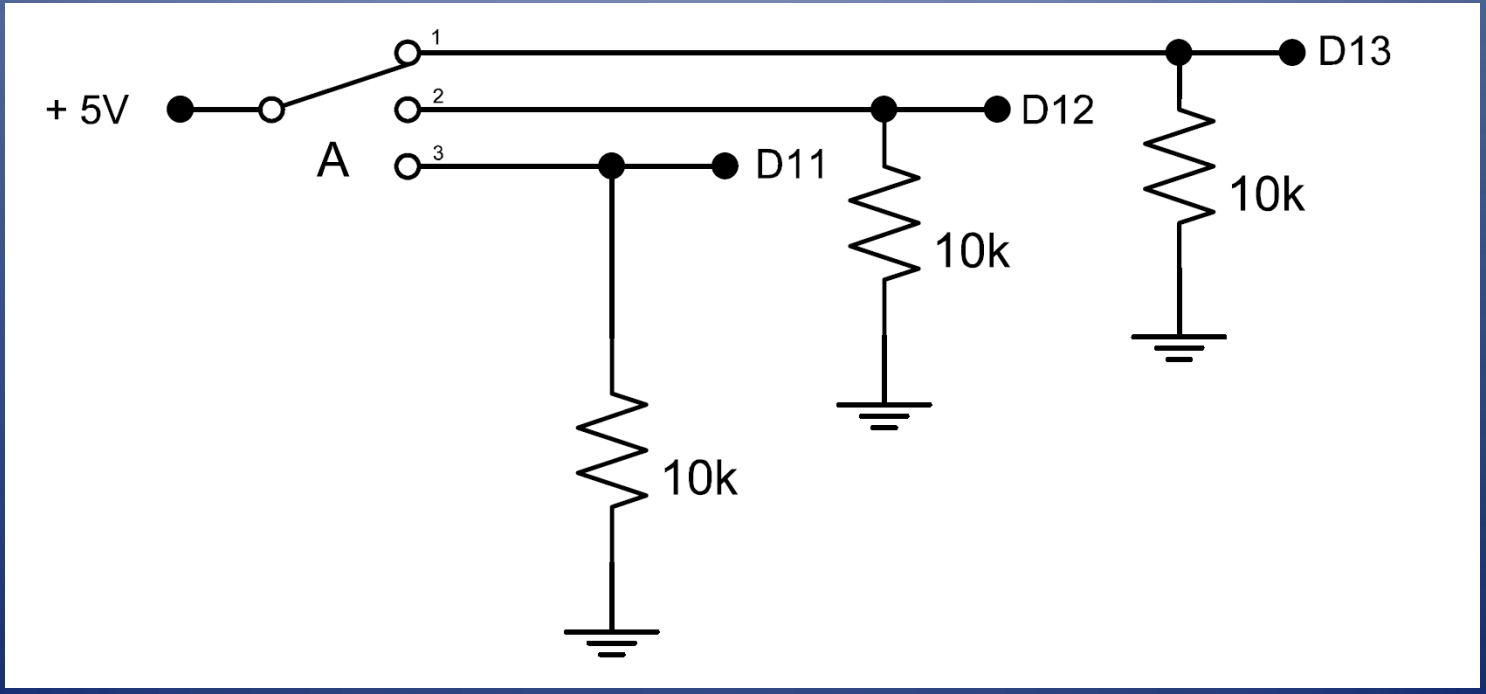
Opdracht:

Frequenties instellen met een schakelaar

Schakelstand A1 = 50,040 MHz

A2 = 50,041 MHz

A3 = 50,042 MHz



HOP9850

```
byte switchA1 = 11;          // I/O 11 is verbonden met schakelaar A, pen 1
byte switchA2 = 12;          // I/O 12 is verbonden met schakelaar A, pen 2
byte switchA3 = 13;          // I/O 13 is verbonden met schakelaar A, pen 3
unsigned long freq1 = 50050000;
unsigned long freq2 = 50051000;
unsigned long freq3 = 50052000;
int switchNu = 1;
int switchVorig = 0;
```

HOP9850

```
pinMode (switchA1, INPUT);   // via I/O 11 wordt de status van de schakelaar pen 1 gelezen
pinMode (switchA2, INPUT);   // via I/O 12 wordt de status van de schakelaar pen 2 gelezen
pinMode (switchA3, INPUT);   // via I/O 13 wordt de status van de schakelaar pen 3 gelezen
```

HOP9850

```
void loop()
{
  if (digitalRead(switchA1) != 0)  // schakelaar A1 hoog?
    switchNu = 1;
  if (digitalRead(switchA2) != 0)  // schakelaar A2 hoog?
    switchNu = 2;
  if (digitalRead(switchA3) != 0)  // schakelaar A3 hoog?
    switchNu = 3;
  if (switchNu != switchVorig)     // als de schakelstand gelijk is: niets uitvoeren
  {
    switch (switchNu)
    {
      case 1:                  // A1 is gesloten
        sendFrequency(freq1);
        break;
      case 2:                  // A2 is gesloten
        sendFrequency(freq2);
        break;
      case 3:                  // A3 is gesloten
        sendFrequency(freq3);
        break;
    }
    switchVorig = switchNu;
    delay (1000);
  }
}
```

FSKCW9850

Vakantietaak

Genereer “ON6MS JO10UX” in morse op 28,350 MHz, FSKCW shift 170 Hz, 8 wpm

- ✓ **Hardware setup: zoals bij Sweep9850 (3 lijnen)**
- ✓ **Software: zie infoblad**
- ✓ **Maak een analyse van het programma**


That's all Folks!