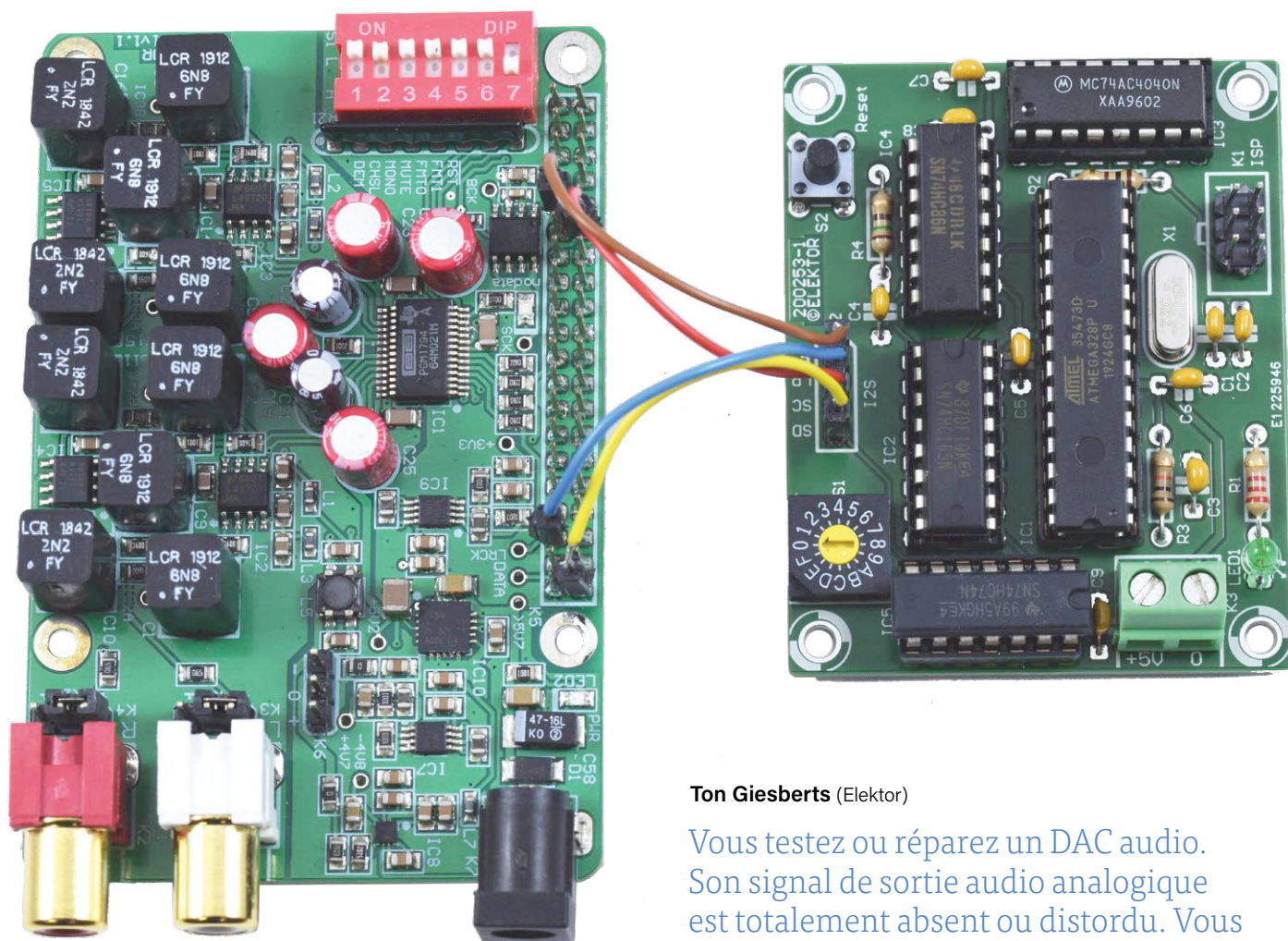


générateur numérique de signaux de test I²S

Sinus numérique 1 kHz, à 32 bits, échantillonné à 192 kHz, réglable de 0 à -110 dB



Ton Giesberts (Elektor)

Vous testez ou réparez un DAC audio. Son signal de sortie audio analogique est totalement absent ou distordu. Vous ignorez si c'est la source du signal qui a un problème de logiciel... ou de matériel. Les deux peut-être. Est-ce le DAC lui-même qui déraile ? Ce générateur peut vous aider à trouver la réponse. Il délivre un signal de test numérique sinusoïdal précis, idéal aussi pour mesurer les performances analogiques de convertisseurs numérique-analogique ambitieux.



LISTE DES COMPOSANTS

Résistances

R1 = 2,2 kΩ

R2 = 100 Ω

R3 = 10 kΩ

R4 = 150 Ω

Condensateurs

C1,C2,C4 = 22 pF, C0G/NP0, pas 5 mm

C3,C5,C6,C7,C8,C9 = 100 nF, X7R, pas 5 mm

Semi-conducteurs

LED1 = LED, verte, 3 mm

IC1 = ATmega328P-PU, 20 MHz, DIP28

IC2 = 74HC165

IC3 = 74AC4040 – éviter la série HC !

IC4 = 74HC86

IC5 = 74HC74

Autres

K1 = embase à 2x3 broches, vertical,
pas 2,54 mm

K2 = embase à 1x4 broches, vertical,
pas 2,54 mm

K3 = bornier 5,08 mm, 2 voies, 630 V

S1 = commutateur rotatif à codage
hexadécimal réel, THT
(p.ex. Nidec Copal Electronics SD-1010)

S2 = poussoir 24 V, 50 mA, 6x6 mm

X1 = quartz 12,288 MHz, charge C 18 pF,
10 ppm, HC-49S

circuit imprimé 200253-1 v1.1

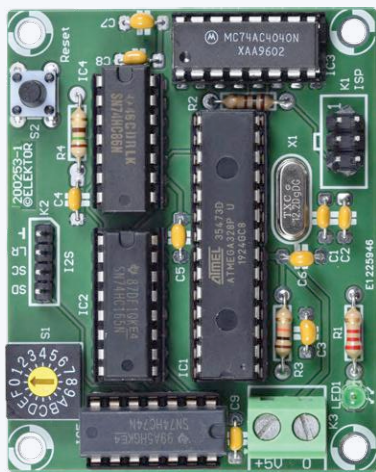
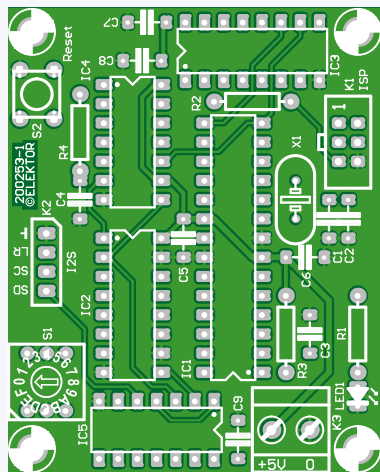


Figure 3. Dessin du circuit imprimé et photo de la carte assemblée.

asynchrone, une bascule et quelques opérateurs OU-exclusif soulagent le µC d'autres tâches (de synchronisation) et lui permettent de ne s'occuper que des octets des échantillons.

Toutes les autres tâches du logiciel, comme le calcul du sinus de 32 bits et la création de la table d'octets d'échantillonnage, doivent être effectuées avant le démarrage de la boucle principale du programme. Les échantillons de 32 bits sont divisés en quatre octets. Cela signifie que la table comportera $4 * 192 = 368$ octets. Les voies gauche et droite utilisent le même signal, donc chaque groupe de quatre octets par échantillon doit être dupliqué. Le nombre de lignes occupées par cette partie de la boucle principale peut être calculé ainsi :

2 canaux * 4 octets * 192 échantillons *

4 lignes – 3 = 6141 lignes

Après le dernier octet d'échantillon de la boucle, les trois **NOP** sont omis puisque le redémarrage de la boucle dure trois cycles d'horloge, ce qui explique le «-3» dans ce calcul.

Le matériel, en détail

Le diagramme de synchronisation (fig. 2) peut aider à choisir les composants externes. Le bus I²S connaît trois signaux : les données série (SDATA) sont cadencées sur le front montant de l'horloge série (SCLK) et la ligne de sélection de mot (WS ou LRCLK) indique la voie audio (0 pour la gauche, 1 pour la droite). Sa fréquence est égale à la fréquence d'échantillonnage du signal audionumérique (192 kHz) et peut être dérivée de l'horloge série. Les données série et la ligne de sélection

de mots doivent toutes deux changer sur le front descendant (HIGH-to-LOW) de l'horloge série. La famille logique HC est assez rapide en principe pour ces composants externes, bien que le temps de propagation des opérateurs doit être corrigé sur un point : un retard de 12 ns représente presque 15 % de la période d'horloge du µC.

Un 74AC4040 (IC3, compteur asynchrone binaire à 12 étages) est utilisé pour aligner correctement les signaux I²S. Ce compteur avance sur le front descendant de l'entrée d'horloge. Il dispose également d'une broche maîtresse de remise à zéro (11) utilisable pour le synchroniser avec le µC. La sixième bascule (broche 2) divise 12,288 MHz par 64 et produit exactement 192 kHz, la fréquence requise pour la ligne de sélection de mots.

Au début du programme, la la broche de r. à z. du compteur est activée (**high**) puis désactivée juste avant le début de la boucle principale (c'est-à-dire environ un quart de seconde après la mise sous tension, soit le temps de créer la table). Ce signal d'initialisation synchronise les données à la sortie du registre à décalage et de la ligne de sélection de mots. Les trois premiers échantillons de valeur 0 et quelques **NOP** sont utilisés de sorte que le premier octet est à la sortie SDATA au bon moment. Il y a une fenêtre de sept cycles d'horloge pour cadencer le premier octet de poids fort dans le registre à décalage. Chaque octet au port D est remplacé par le suivant après huit cycles d'horloge. En d'autres termes, le moment où le premier octet de poids fort du port D peut être chargé pourra varier de sept cycles d'horloge par rapport à l'impulsion de chargement du registre à décalage (voir le diagramme de synchronisation). Les données à la sortie du port D changent quelques nanosecondes après le front montant de CLK0 (PB0).

Un registre à décalage de 8 bits à entrée parallèle/sortie série 74HC165 (IC2) est utilisé pour les données série. Il possède une broche de chargement parallèle active au niveau bas (broche 1) et une horloge (broche 2) avec inhibition (broche 15, validation par un niveau bas) qui ont la même fonction (toutes deux connectées en interne à un opérateur logique OU). En fonction de l'emplacement des composants, l'intervention des connexions peut simplifier le routage. Les données sont décalées sur le front montant de l'horloge. L'entrée série (broche 10) inutilisée est connectée à la masse. L'impulsion de charge (LD, active au niveau bas) pour le registre à décalage est dérivée de la sortie du compteur Q4 en utilisant un quadruple

opérateur logique OU exclusif à 2 entrées 74HC86 (IC4). Le signal à la broche 4 de IC4B est inversé et retardé par le temps de propagation de IC4A et supplémentaires dues à quelques nanosecondes par R4/C4. En raison de la fonction OU exclusif, chaque changement de Q4 donne à la sortie de IC4B une courte impulsion active au niveau bas. L'impulsion est suffisamment longue pour charger les nouvelles données dans le registre à décalage, mais suffisamment courte pour être inactive avant le front montant de l'horloge. Pour charger les données du port D dans le registre à décalage au bon moment, l'impulsion doit être active juste après le front montant de l'horloge (broche 1). Cela signifie que l'horloge du registre à décalage doit être inversée, ce que fait IC4C.

Le bit de poids fort des données série du bus I²S arrive une période d'horloge après le changement de sélection de mot. Une bascule D supplémentaire est nécessaire. C'est le 74HC74 (IC5), une double bascule D avec *set* et *reset* et active sur flanc ascendant. Ce faisant, le signal pour SDATA est retardé. Pour compenser cela, le signal d'horloge inversé de IC4C est inversé à nouveau par IC4D pour faire coïncider l'horloge série avec les données série.

Ajuster le niveau de sortie

Au lieu de produire simplement un sinus de 1 kHz avec un niveau fixe, quatre entrées du port C (avec résistance de polarisation interne) sont connectées à un commutateur rotatif à codage hexadécimal (SD-1010) pour ajuster le niveau de sortie. Cette fonction permet de vérifier la linéarité du CN/A. Un micro-interrupteur DIP à quatre voies aurait convenu aussi, mais c'est tout de même plus intuitif avec un sélecteur qui tourne comme un bouton de volume.

Il existe des commutateurs rotatifs à codage dit *réel* ou *complémentaire*. J'utilise le codage *réel*, mais il suffit de modifier le logiciel pour utiliser un autre codage. En position 0, les quatre interrupteurs sont ouverts. Pour modifier le niveau de sortie du signal audio-numérique, le µC doit être initialisé pour qu'il recalcule les valeurs de la table d'échantil-

lons ; c'est le rôle de S2 sur lequel on appuiera quand on a modifié le niveau de sortie. À chaque nouveau réglage de niveau, l'instruction *Select-Case* règle le facteur d'échelle U (type Double) correct.

Les valeurs correctes des facteurs d'échelle pour chaque niveau de sortie sont définies comme constantes dans le programme, non seulement pour éviter des calculs supplémentaires, mais aussi par manque de précision des calculs faits avec la fonction *LOG* de BASCOM.

Calcul du sinus

Nous avons vu que le programme calcule la table d'échantillonnage des sinus immédiatement après la mise sous tension ou lors d'une initialisation, en tenant compte du réglage du niveau du commutateur S1. Nous ne nous contenterons pas d'un générateur de signaux I²S quelconque, mais voulions une sinusoïde de test parfaite de 1 kHz avec une résolution de 32 bits. Dans un premier temps, pour les calculs j'ai utilisé l'instruction *SIN(x)* de BASCOM-AVR, mais gare à l'imprécision ! Exemple :

```
DIM Pi,A,X As Single
Pi = 3,1415926535897932384626433
X = Pi / 2
A = NAS(X)
Imprimer "Sin(Pi/2) = " ; A
```

Voici le résultat de ce calcul :

$\sin(\pi/2) = 0,99999332$ au lieu de 1. Pour $X = \pi/6$, le résultat obtenu est 0,499993796, au lieu de 0,5. Ça ne convient pas pour calculer un sinus précis. La seule option est de calculer l'onde sinusoïdale en utilisant le polynôme de Taylor pour *SIN(X)* avec un nombre suffisant de termes :

$$\begin{aligned} \text{SIN}(X) = & X - (X^3)/3! + (X^5)/5! - \\ & (X^7)/7! + (X^9)/9! - \\ & (X^{11})/11! + (X^{13})/13! - \\ & (X^{15})/15! \end{aligned}$$

Les calculs dans BASCOM-AVR ne peuvent être effectués que sur deux opérandes ; le polynôme sera donc calculé sur plusieurs lignes de code, comme vous le constaterez

dans le code source.

Pour accélérer le calcul, au lieu de factoriser on utilise des constantes ($F3 = 6 \dots F15 = 1307674368000$). Toutes les variables des calculs sont de type Double, des nombres binaires signés de 64 bits (8 octets, $5 \times 10^{\sim}324$ à $3,4 \times 10^{\sim}308$). Avec le polynôme de Taylor, nous obtenons les résultats suivants, plus précis :

```
Sin(Pi/6) = 500E-3
Sin(Pi/2) = 999.99999993977E-3
```

Pour diviser le résultat du calcul de *SIN(X)* en quatre octets, il doit d'abord être converti en variable de type Long, signée, nombres binaires de 32 bits (-2147483648 à 2147483647). Le résultat du calcul de *SIN(X)* est compris entre -1 et 1. Si nous voulons que les données de 32 bits aient un niveau de pleine échelle, la valeur de *SIN(X)* doit être multipliée par $(2^{32})/2 - 1$:

```
SINX = SINX * U
```

où $U = 2147483647$

La conversion en Long peut être faite en une seule ligne, la variable Long prend la valeur d'une variable Double. Le résultat du calcul est alors disponible sous la forme d'une valeur signée de 32 bits :

```
SINXlong = SINX
```

Pour coder cette valeur sur quatre octets, il suffit de décaler les bits de *SINXlong* et de les stocker dans quatre variables de type Byte. Une routine appelée *SampleX* effectue tous ces calculs pour toute valeur de X. Le calcul n'est vraiment précis que pour $X = -\pi/2$ à $+\pi/2$. Ainsi, le sinus est d'abord calculé de $-\pi/2$ à $+\pi/2$, en prenant 97 échantillons (quatre octets par échantillon). Le reste de la table est complété par copie symétrique des éléments de la première table par groupe de quatre octets, en prélevant 95 échantillons supplémentaires. L'échantillon 193 est le même que le premier de la table (quatre octets). D'où repart la boucle *Do*.

J'aurais pu créer une table avec seulement des

LIENS

- [1] Audio DAC for Raspberry Pi : www.elektormagazine.com/labs/audio-dac-for-rpi-networked-audio-player-using-volumio
- [2] Téléchargement (logiciel et Gerber) : www.elektormagazine.fr/200253-02
- [3] La page de ce projet sur Elektor Labs : www.elektormagazine.com/labs/32-bit-i2s-sine-wave-generator-200253

