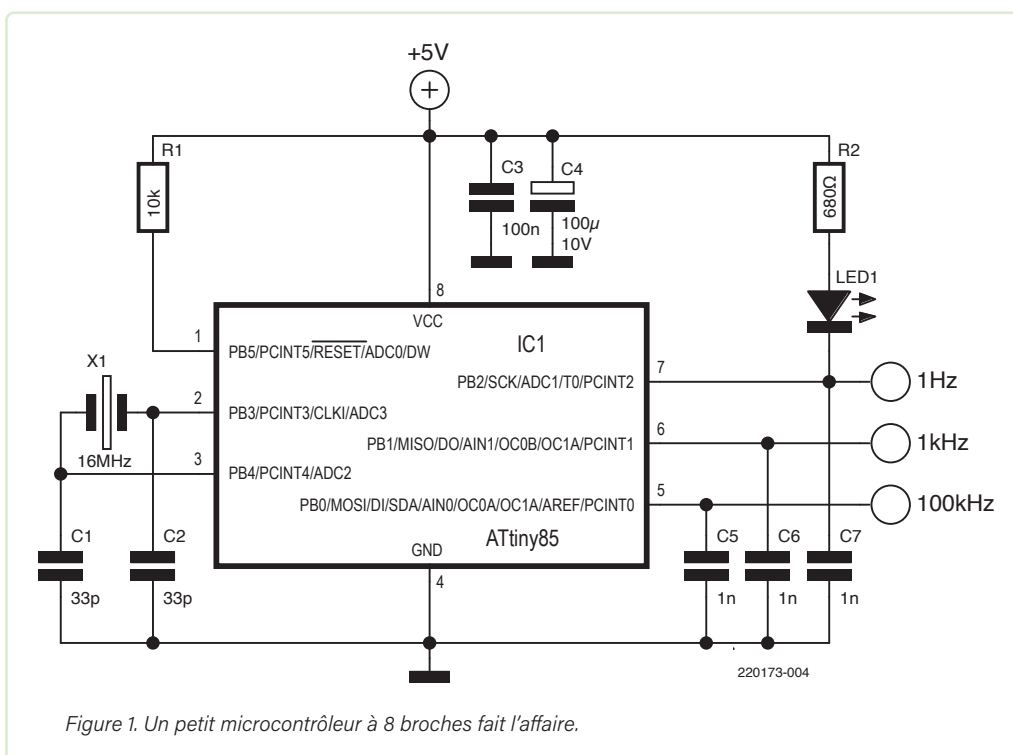


45

référence de fréquence avec ATtiny

génère des ondes carrées de 1 Hz, 1 kHz et 100 kHz



Antonello Della Pia (Italie)

Avec ce circuit utilisant juste un microcontrôleur ATtiny85, vous pouvez obtenir simultanément trois ondes carrées de fréquences différentes, définies par vous-même. Le logiciel du projet est plutôt simple.

Pour mes expériences avec les filtres à condensateurs commutés, j'avais besoin de deux signaux carrés stables et précis — un à une fréquence de 1000 Hz pour l'entrée du filtre et un autre à 100 kHz pour l'horloge du filtre. Comme je ne voulais pas utiliser un circuit complexe avec plusieurs oscillateurs et diviseurs de fréquence, j'ai trouvé une solution simple. Elle permet d'obtenir simultanément trois ondes carrées de fréquences différentes, définies par l'utilisateur, grâce à un microcontrôleur ATtiny85 (**figure 1**).

L'ATtiny85 possède deux timers/compteurs qui peuvent être contrôlés individuellement. Il est assez facile d'obtenir deux fréquences différentes en réglant leur prédiviseur et registres compteurs respectifs selon les tableaux donnés dans la fiche technique. Le Timer/Counter0 bascule la sortie PB0 pour produire une onde carrée de 100 kHz. De même, Timer/Counter1 génère le signal 1 kHz sur le port PB1. La stabilité et la précision suffisante sont garanties par un quartz de 16 MHz. Le logiciel utilisé pour ce projet a été écrit pour Arduino. Les deux timers

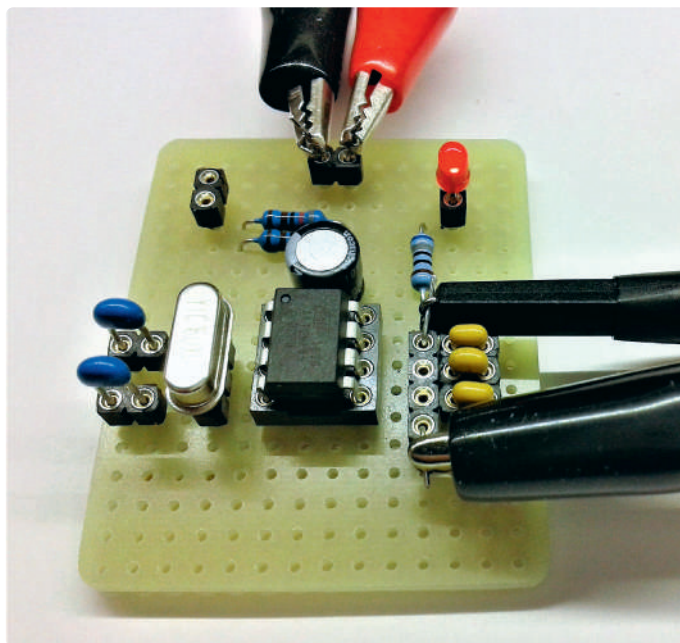


Figure 2. Le prototype a été construit sur une platine perforée.

sont complètement paramétrés dans la fonction *setup* du sketch, et donc la fonction *loop* peut rester vide. Pour la rendre quand même utile, j'ai décidé de l'utiliser pour générer une onde carrée de 1 Hz sur le port PB2. Ceci est facile à réaliser en basculant la broche tous les 500 ms.

Compter les cycles d'horloge

Cependant, comme les deux timers sont déjà utilisés, la fonction de retard (*delay*) d'Arduino ne fonctionne plus, et j'avais donc besoin d'une autre solution pour créer un délai de 500 ms. J'en ai trouvé une dans la fonction spéciale `__builtin_avr_delay_cycles`, qui fait partie des fonctions intégrées du compilateur GCC AVR. Cette fonction produit un retard en comptant les cycles d'horloge au lieu d'utiliser un timer. Son utilisation est assez simple puisque vous ne devez spécifier que le nombre de cycles d'horloge à attendre. Avec un quartz de 16 MHz, un cycle d'horloge (également appelé tick) dure 62,5 ns, donc 500 ms

correspondent à 8 000 000 cycles. Avec cette fonction, nous pouvons obtenir une plage de fréquences allant de 0,5 Hz (16 000 000 cycles) jusqu'à 500 kHz (11 cycles). Notez que le nombre de cycles ne peut pas être une variable, il doit être une constante (c'est-à-dire, écrit comme un nombre).

Dans le circuit de test, PB2 fait également clignoter une LED (figure 2). Les condensateurs connectés entre les sorties et la masse ne sont pas obligatoires, mais ils réduisent le bruit pour fournir une forme d'onde plus nette. La figure 3 montre les trois formes d'ondes disponibles aux sorties.

J'ai écrit et compilé le programme en utilisant l'Arduino IDE 1.8.19 avec l'ATtiny Core 1.5.2 de Spence Konde installé. J'ai utilisé un programmeur USBasp pour programmer le microcontrôleur. Le sketch utilise seulement 340 bytes de mémoire de programme, donc un ATtiny25 serait aussi adéquat. Il est possible de modifier et de recompiler facilement le fichier fourni, *ATtiny85_3_Square_Waves.ino*, qui contient des commentaires et des informations supplémentaires. Le fichier HEX est également disponible [1].

220173-04



Related Products

- W. A. Smith, *Explore ATtiny Microcontrollers using C and Assembly Language* (Elektor 2022) (SKU 20007) www.elektor.fr/20007

↓ Télécharger le projet



www.elektormagazine.fr/summer-circuits-22

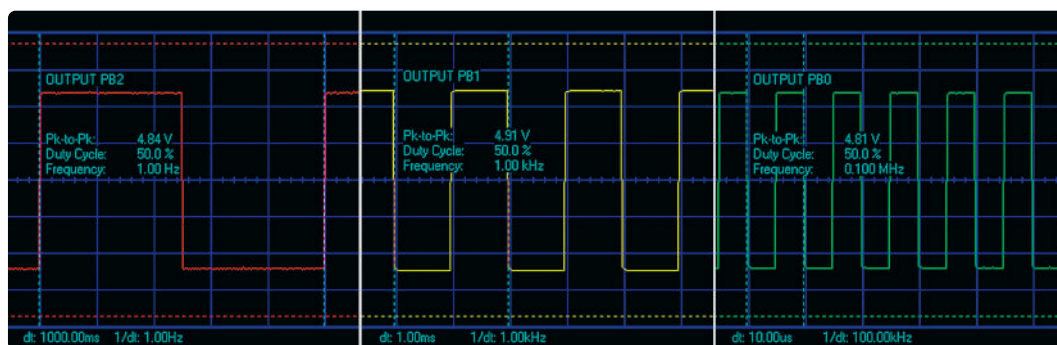


Figure 3. Les trois signaux de sortie générés par le circuit.

LIENS

[1] Téléchargements pour cet article : <https://www.elektormagazine.fr/summer-circuits-22>