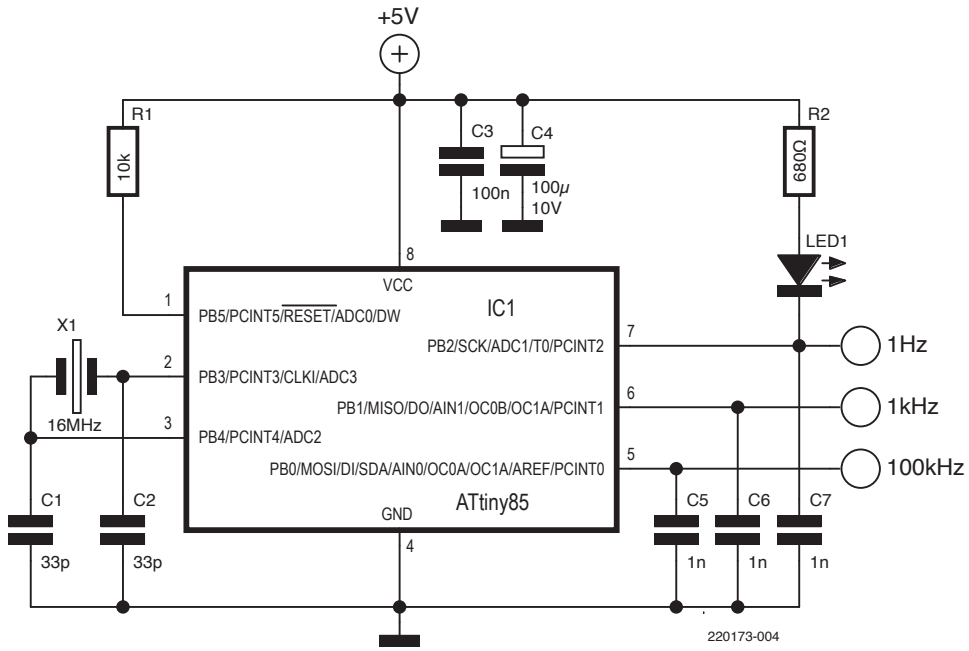
45

Mini-frequentiereferentie

genereert 1Hz-, 1kHz- en 100kHz-blokgolven



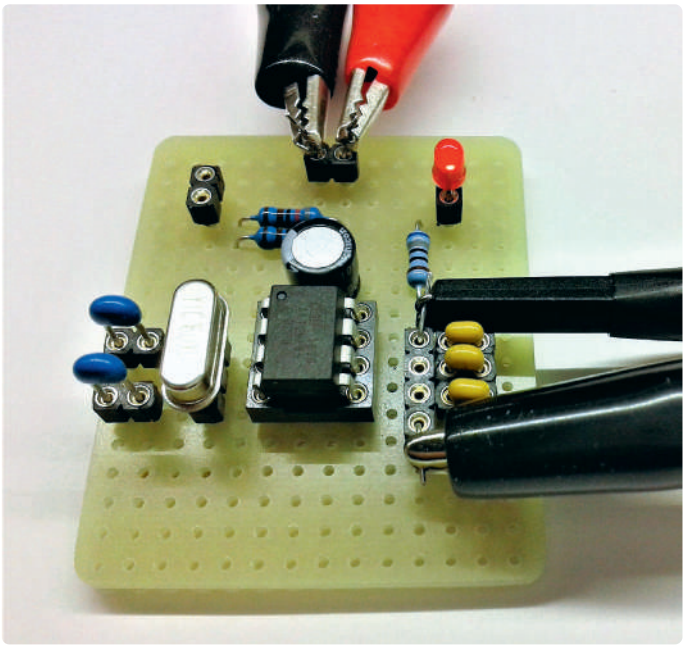
Figuur 1. Een kleine 8-pins microcontroller neemt alles voor zijn rekening.

Antonello Della Pia (Italië)

Met deze schakeling, opgebouwd rond een enkele ATtiny85-microcontroller, kunnen simultaan drie blokgolven met verschillende, door de gebruiker gedefinieerde frequenties worden gegenereerd. De software van het project is rechttoe-rechtaan.

Voor mijn experimenten met filters met geschakelde condensatoren had ik twee stabiele en nauwkeurige blokgolven nodig – één met een frequentie van 1000 Hz voor de ingang van het filter en een andere met een frequentie van 100 kHz voor de klok van het filter. Omdat ik geen ingewikkelde schakeling wilde met meerdere oscillatoren en frequentiedelers, heb ik een eenvoudige oplossing bedacht. Hiermee kunnen gelijktijdig drie blokgolven met verschillende, door de gebruiker gedefinieerde frequenties worden verkregen met een enkele ATtiny85-microcontroller (**figuur 1**).

De ATtiny85 heeft twee timers/counters die onafhankelijk van elkaar kunnen worden gebruikt. Het is vrij eenvoudig om twee verschillende frequenties te genereren door de betreffende prescaler- en tellerregisters in te stellen volgens de tabellen in de datasheet. Timer/Counter0 schakelt uitgang PB0 en produceert zo een 100kHz-blokgolf. Op dezelfde manier genereert Timer/Counter1 het 1kHz-signaal op poort PB1. De stabiliteit en de nauwkeurigheid (die voor mijn doeleinden goed genoeg is) worden gegarandeerd door het gebruik van een 16MHz-kristal.



Figuur 2. Het prototype op gaatjesprint.

De software voor het project is gemaakt in Arduino. De twee timers zijn volledig geconfigureerd in de functie *setup* van de sketch, en dus kan de functie *loop* leeg blijven. Om die toch iets nuttigs te laten doen, besloot ik om er een 1Hz-blokgolf mee te genereren op poort PB2. Dit is eenvoudig te realiseren door de pin elke 500 ms om te schakelen (toggle).

Klokcycli tellen

Omdat de beide timers echter al in gebruik zijn, werkt de Arduino-functie *delay* niet meer. Ik had dus een andere oplossing nodig om een vertraging van 500 ms te creëren. Ik vond er een in de speciale functie *__builtin_avr_delay_cycles*, die deel uitmaakt van de ingebouwde functies van de GCC AVR compiler. Deze functie produceert een vertraging door klokcycli te tellen in plaats van een timer te gebruiken. Het gebruik ervan is vrij eenvoudig, aangezien alleen het aantal klokcycli gedurende welke moet worden gewacht, hoeft te worden opgegeven.

Met een 16MHz-kristal duurt een klokcyclus (ook wel tick genoemd) 62,5 nanoseconden, dus 500 ms komt overeen met 8.000.000 cycli. Met deze functie kunnen we frequenties genereren in een bereik van 0,5 Hz (16.000.000 cycli) tot 500 kHz (11 cycli). Merk op dat het aantal cycli geen variabele mag zijn, het moet een constante zijn (dus geschreven als een getal).

In de testschakeling doet PB2 ook een LED knipperen (**figuur 2**). De condensatoren tussen de uitgangen en massa zijn niet verplicht, maar ze reduceren de ruis en zorgen voor een 'schonere' golfvorm. **Figuur 3** toont de drie golfvormen op de uitgangen.

Het programma werd geschreven en gecompileerd met behulp van de Arduino IDE 1.8.19, met de ATTiny Core 1.5.2 van Spence Konde geïnstalleerd. Ik heb een USBasp-programmer gebruikt om de MCU te flashen. De sketch gebruikt slechts 340 bytes aan programmageheugen, dus een ATtiny25 zou ook voldoen. Het bestand in de download (*ATtiny85_3_Square_Waves.ino*), dat van meer commentaar en informatie is voorzien, kan eenvoudig worden aangepast en opnieuw worden gecompileerd. Het HEX-bestand is ook beschikbaar [1].

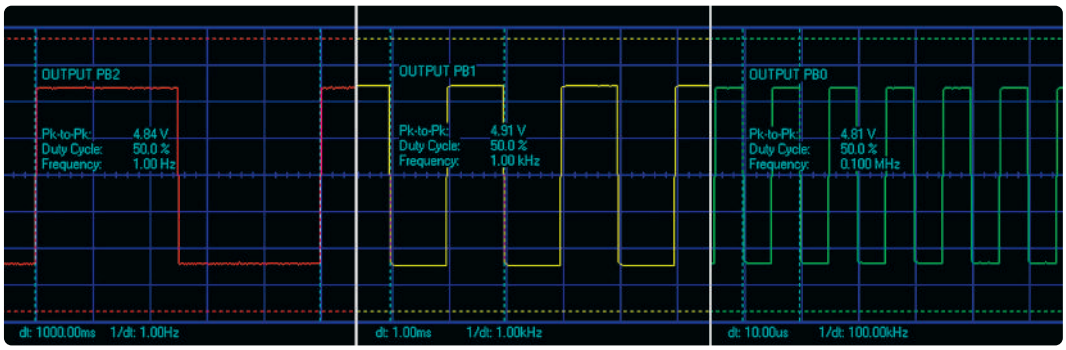
220173-03

Gerelateerde producten

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

Project-download

www.elektormagazine.nl/summer-circuits-22



Figuur 3. De drie uitgangssignalen die de schakeling produceert.

WEBLINKS

[1] Software-download: https://www.elektormagazine.com/summer-circuits-22