ThePicoSid – SID Ersatz mit RP2040 (RaspberryPi Pico)

von Thorsten Kattanek - Berlin, 17.09.2023

Funktionen der beiden Pico Cores

Core0

- Initialisierung (System Clock setzten, SID Emulator)

- Core1 starten

- PIO’s programmieren

- hier werden die Write Daten per IRQ0 entgegen genommen

- auf Reset prüfen und SID Emulation resetten

Core1

- Initialisierung des PWM Ausgangs

- Der SID wird hier im IRQ des PWM emuliert. 24 Zyklen pro Sample

Reset

Die C64 RESET Leitung ist am Pico GPIO02 angeschlossen. Dafür wird ein Interrupt eingesetzt der auf die steigende und fallende Flanke des GPIO’s reagiert. Bei fallender Flanke ist ein Reset am C64 aufgetreten. (Normalerweise muß 10 Zyklen lang die Resetleitung auf Low gewesen sein). Es wird jetzt der Audio Ausgang ausgeschaltet und es werden alle Register auf 0 gesetzt (SidReset). Kommt eine steigende Flanke so wird der Audio Ausgang wieder eingeschaltet.

Write Data

Wann werden Daten vom C64 zum SID gesendet und wie können wir diese hier abgreifen. Dazu gibt es die 3 Leitungen CLK, RW und CS.

CLK = Der Systemtakt des C64
RW = Read and Write
CS = Chip Select

Der PIO vom Pico ist so programmiert das er auf die Steigende Flanke des CLK Signals wartet und dann nach 31 Zyklen überprüft ob CS = 0 und RW = 0 ist. Wenn diese Bedingung erfüllt ist, liest er die Daten vom Adressbus und Datenbus. Diese Daten werden dann per IRQ0 an dem Pico Core#0 übergeben und dieser übergibt die Daten an die SID Emulation per „SidWriteReg“.

PWM Ausgabe

Die PWM Ausgabe erfolgt mit einer Samplerate von 41223Hz und eine Bitbreite von 11 Bit.

Berechnung der PWM Frequenz: $\frac{248000000\,Hz}{2048*(2+\frac{15}{16})}=41223\,Hz$

Der C64 (PAL) Takt beträgt: 985248Hz.

Das macht dann $985248\,Hz/41223\,Hz=23.9\,Zyklen$ pro Sample. Es werden aber 24 Zyklen pro Sample berechnet. Das macht einen Fehler von plus 0,41841 %.

Das ganze wird per IRQ aufgerufen, also pro Sample, wo dann 24 Sid Zyklen emuliert werden und die Ausgabe per „pwm_set_gpio_level“ auf den PWM Kanal übertragen wird.

Wichtig an der Stelle: Die Audio Ausgabe der SID Emulation ist „signed“ also Vorzeichen behaftet, der PWM arbeitet aber mit „unsigned“ als Vorzeichenlos. Das muss man berücksichtigen!

Hier die Umrechnung:
```
uint16_t out = ((SidFilterOut() >> 4) + 32768) / (float)0xffff * 0x7ff;
```