Proseminar Computer and Music – Das C64 SID

Sebastian Jörn Geiss

17. Januar 2018

Inhaltsverzeichnis

1 Einleitung

Die folgende Ausarbeitung handelt von dem Commodore 64 SID. Das SID war ein fester Bestandteil des im Jahre 1984 vorgestellten Commodore 64 (im weiteren Verlauf als C64 bezeichnet). Der C64 ist ein damals oft verkaufter Heimcomputer. Im Jahre 1989 wurde er in Deutschland bereits über 1 Million mal verkauft. [3] Aus einer E-Mail von Commodore Angestellten Dr. Peter Kittel geht hervor, dass der C64 über 3 Millionen mal verkauft worden ist.
SID steht für Sound Interface Device. Dieses ist im C64 für die Musik zuständig. Im weiteren Verlauf wird dargestellt, wieso das SID ein sogenannter Meilenstein in der Geschichte der Musikprozessoren in Heimcomputern darstellte und wie man Töne mithilfe des Soundchips generieren und bearbeiten kann. Zusätzlich wird der Aufbau erläutert.

2 Historischer Kontext

Um darzustellen, wie sich das C64 SID im historischen Kontext einordnen lässt, empfiehlt sich der direkte Vergleich zu einem anderen, damals verbreiteten Chip. Hierzu eignet sich der PoKEY Chip aus dem Atari 400 bzw 800.
Der Name PoKEY kommt daher, dass mit dem Chip auch die Tastatur gesteuert werden konnte.

Tabelle 1: Gegenüberstellung der beiden Musikprozessoren:

C64 SID	**PoKEY**
3 Stimmen	4 Stimmen
16 Bit Frequenzteiler	8 Bit Frequenzteiler
7 Oktaven	3 Oktaven
4 Waveforms + Kombination	2 Waveforms
ADSR Kurve für jede einzelne Stimme	
Ringmodulator und Filter	

[4] [2]

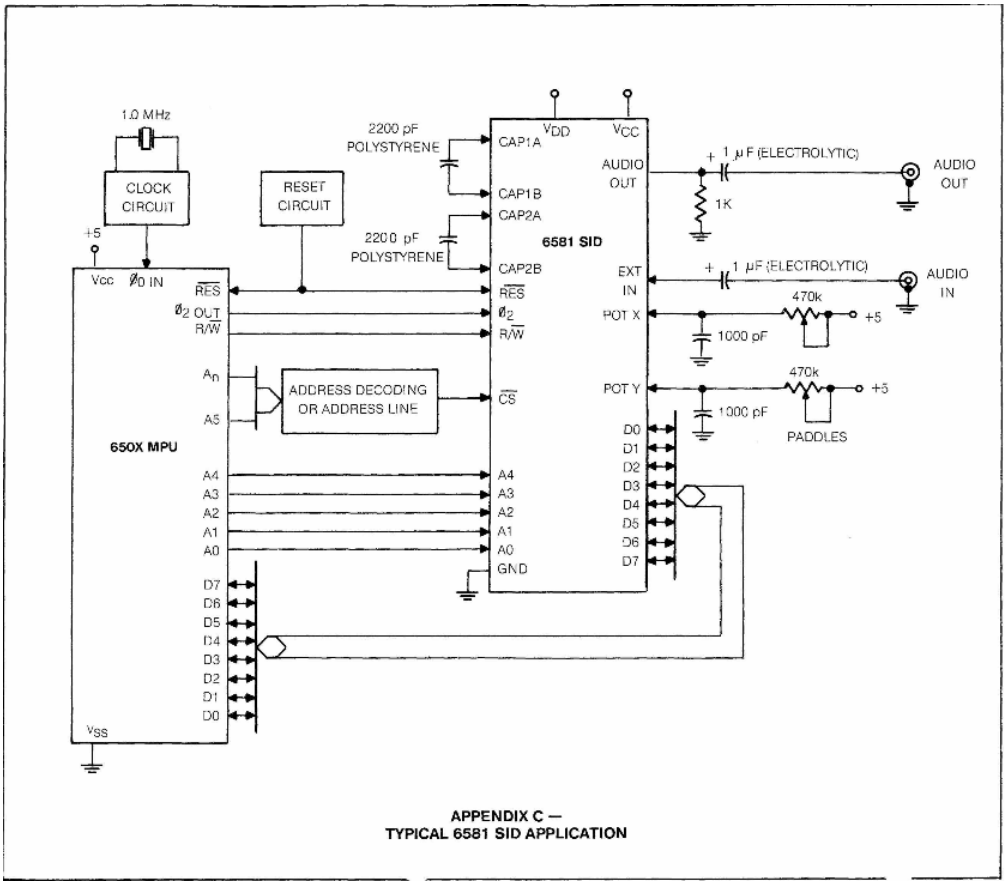
Aus der Tabelle geht hervor, dass der PoKEY Chip zwar über mehr Stimmen verfügt, aber ansonsten dem C64 SID unterliegt. Dies wird beispielsweise in der Anzahl verschiedener Waveforms, der Anzahl Oktaven und dem Vorhandensein der Ringmodulation deutlich.
Die Anzahl Oktaven des SID ist dabei mit Vorsicht zu interpretieren, da die sehr tiefen Töne lediglich ein Brummen, bzw die sehr hohen Töne lediglich ein Piepen darstellen.

In dem Interview mit Bob Yannes, einem Entwickler des Chips, geführt von Andreas Varga, tätigte dieser folgende Aussage:

"I thought the sound chips on the market (including those in the Atari computers) were primitive and obviously had been designed by people who knew nothing about music. As I said previously, I was attempting to create a synthesizer chip which could be used in professional synthesizers." [5]

Yannes behauptet, dass er einen sehr viel professionelleren Chip für Heimcomputer geschaffen hätte, als alle anderen jemals zuvor. Zusätzlich hätten die Entwickler der anderen Soundchips keine Ahnung von Musik gehabt, so seine Aussage.
Zwar sticht das SID klar aus der Konkurrenz hervor, dennoch sollte man die Aussage kritisch hinterfragen, dass die Konkurrenz schlicht keine Ahnung hatte.

3 Aufbau des SID

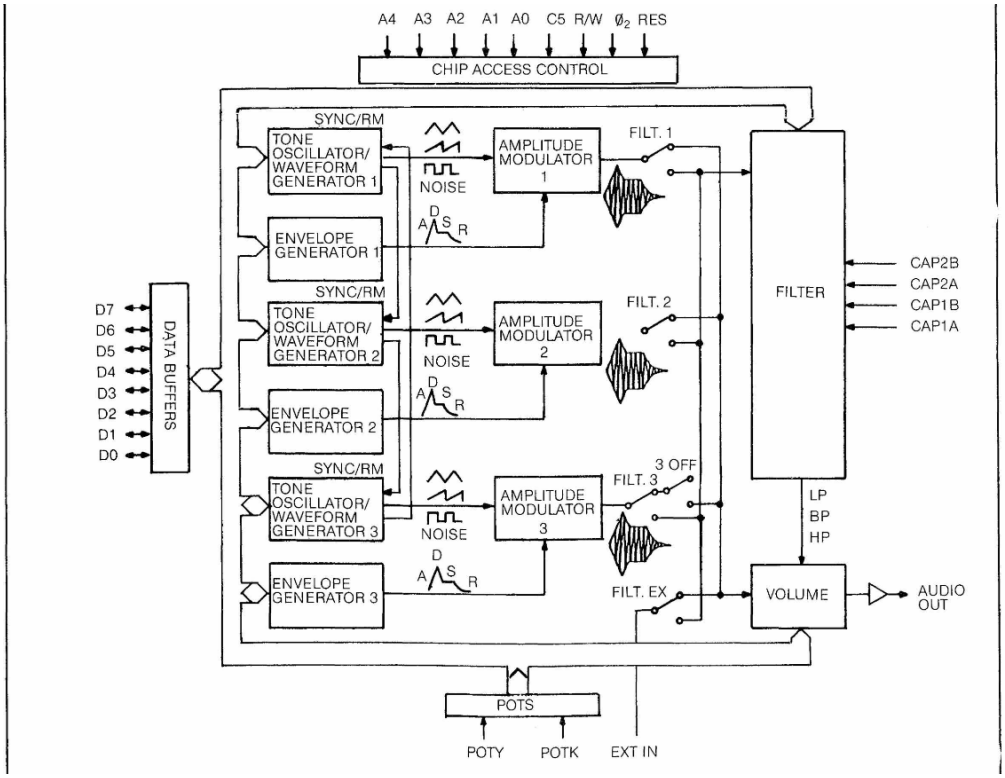


APPENDIX C —
TYPICAL 6581 SID APPLICATION

[2]

Abbildung 1: Aufbau des C64

Aus der Abbildung 1 geht hervor, über welche Eingänge und Ausgänge, bzw. über welche Verbindungen zur MPU das SID verfügt.
So wird deutlich, dass D0 bis D7 für den Inhalt der zu beschreibenden Register zuständig sind. Zusätzlich sind A0 bis A4 für die Adresse im SID zuständig.
Mit PotX und PotY verfügt das SID über die Möglichkeit, externe Steuerungspaddles anzuschließen. Diese konnten früher zum C64 dazugekauft werden.
Mit EXT IN konnte man beispielsweise eine E-Gitarre oder einen weiteren SID anschließen. Die Kombination mehrerer SID Chips diente hauptsächlich dazu, dass mehr als 3 Stimmen auf einmal ausgegeben werden konnten.
CAP1A, CAP2A, CAP1B, CAP2B verbanden einen Kondensator mit dem SID. Dieser war für den Filter zuständig.



[2]

Abbildung 2: Beispielsituation

Zuerst fällt in Abbildung 2 sicher auf, dass die Stimme 2 abgeschaltet, bzw. aus ist. Man erkennt zusätzlich, dass die Stimme 1 in dieser Momentaufnahme gefiltert wird, die Stimme 3 aber nicht. Dazu sieht man sich besonders die rechte Seite der Abbildung 2 an.
Gut zusehen ist, dass der Oszillator für die Wellenform da ist, während der Envelope Generator den ADSR Verlauf regelt.
Da alle Stimmen, bevor sie ausgegeben werden, durch VOL verlaufen, sieht man deutlich, dass die Lautstärke lediglich für alle Stimmen gleichzeitig eingestellt werden kann. Dies wird später auch noch einmal erklärt.

Durch den Aufbau kann man erste Rückschlüsse auf die Programmierung ziehen. So erkennt man in der Abbildung 2 , dass die Ringmodulation der Stimmen einer vorgegebenen Reihenfolge folgt, sodass Stimme 1 die Stimme 3 modulieren kann, die Stimme 2 die Stimme 1 modulieren kann und die Stimme 3 die Stimme 2 modulieren kann.

4 Ausgabe eines simplen Tons

	Address					Reg #	Data									
	A4	A3	A2	A1	A0	(Hex)	D7	D6	D5	D4	D3	D2	D1	D0	Reg Name	Reg Type
															VOICE 1	
0	0	0	0	0	0	00	F7	F6	F5	F4	F3	F2	F1	F0	Freq Lo	Write-only
1	0	0	0	0	1	01	F15	F14	F13	F12	F11	F10	F9	F8	Freq Hi	Write-only
2	0	0	0	1	0	02	PW7	PW6	PW5	PW4	PW3	PW2	PW1	PW0	PW LO	Write-only
3	0	0	0	1	1	03	—	—	—	—	PW11	PW10	PW9	PW8	PW HI	Write-only
4	0	0	1	0	0	04	NOISE	⊓⊓	◿◺	∧∧	TEST	RING MOD	SYNC	GATE	Control Reg	Write-only
5	0	0	1	0	1	05	ATK3	ATK2	ATK1	ATK0	DCY3	DCY2	DCY1	DCY0	Attack/Decay	Write-only
6	0	0	1	1	0	06	STN3	STN2	STN1	STN0	RIS3	RIS2	RIS1	RIS0	Sustain/Release	Write-only

[2]

Abbildung 3: Registertabelle Stimme 1

In der Abbildung 3 erkennt man, dass die Frequenz für die Stimme 16 Bit groß ist. Dies ist erkennbar dadurch, dass FreqLo und FreqHi zusammen die Frequenz ergeben. Da der C64 ein 8 Bit System war, schreibt man die Frequenz in 2 Register. Diese sind als Register 0 und 1 gekennzeichnet. Das Starttegister ist 54272 [3]. Demnach ist das 0. Register bei 54272, das erste bei 54273, etc. Register 2 und 3 stellen die Pulsweite der möglichen Rechteckwelle dar. Auf Register 4, 5 und 6 wird später Bezug genommen.

```
1       lda #$4
2       sta  $D400+1
3       lda #$89
4       sta  $D400
5       lda #%00001100
6       sta  $D400+5
7       lda #%11111111
8       sta  $D400+6
9       lda #%00001111
10      sta  $D400+24
11      lda #%00100001
12      sta  $D400+4
```

Code 1

Dieses Codebeispiel zeigt die Ausgabe eines simplen Tons. Dazu werden verschiedene Schritte ausgeführt. In den ersten beiden Schritten wird die Highfrequenz gesetzt, im folgenden die Lowfrequenz. Anschließend setzt man Attack, Decay, Sustain und Release. In Zeile 9 und 10 wird die Lautstärke gesetzt. In Zeile 12 wird eine Wellenform ausgewählt und das Gate Bit auf 1 gesetzt, wodurch der Ton erklingt.

Um eine Note akustisch darzustellen, hilft die Tabelle aus der Programmers Reference aus.

MUSICAL NOTE		OSCILLATOR FREQ		
NOTE	OCTAVE	DECIMAL	HI	LOW
0	C–0	268	1	12
1	C#–0	284	1	28
2	D–0	301	1	45
3	D#–0	318	1	62
4	E–0	337	1	81
5	F–0	358	1	102
6	F#–0	379	1	123
7	G–0	401	1	145
8	G#–0	425	1	169
9	A–0	451	1	195
10	A#–0	477	1	221
11	B–0	506	1	250
16	C–1	536	2	24

[1]

Abbildung 4: Notentabelle

Wenn man in Abbildung 4 die Werte von C-0 und C-1 vergleicht, fällt auf, dass man von C-1 auf C-0 kommt, wenn man den Wert durch 2 teilt. Demnach kann die Operation mit einem Rechtsshift durchgeführt werden. Um Speicherplatz zu sparen, bietet es sich daher an, lediglich die oberste Oktave zu implementieren. Dazu benötigt man die Spalte HI und LOW, welche für den Wert des Hibytes bzw Lowbytes stehen.

5 Attack, Decay, Sustain, Release

Mit Hilfe der Envelope Generatoren kann man für jede Stimme einzeln die Dauer von Attack, Decay, Sustain und Release (nachfolgend als ADSR bezeichnet) des Tons bestimmen. Die nötigen Einträge in das Register kann man dem offiziellen Datenblatt des SIDs entnehmen:

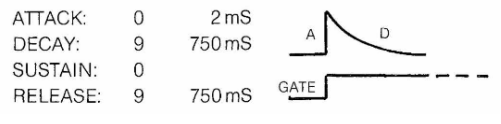
TABLE 2 — ENVELOPE RATES

VALUE		ATTACK RATE	DECAY/RELEASE RATE
DEC	(HEX)	(Time/Cycle)	(Time/Cycle)
0	(0)	2 mS	6 mS
1	(1)	8 mS	24 mS
2	(2)	16 mS	48 mS
3	(3)	24 mS	72 mS
4	(4)	38 mS	114 mS
5	(5)	56 mS	168 mS
6	(6)	68 mS	204 mS
7	(7)	80 mS	240 mS
8	(8)	100 mS	300 mS
9	(9)	250 mS	750 mS
10	(A)	500 mS	1.5 S
11	(B)	800 mS	2.4 S
12	(C)	1 S	3 S
13	(D)	3 S	9 S
14	(E)	5 S	15 S
15	(F)	8 S	24 S

[2]

Abbildung 5: Envelope Rates

Die ADSR-Werte sind stets kleiner als 16, da jeweils 2 hintereinander in ein Byte geschrieben werden. So teilen sich Attack und Decay, sowie Sustain und Release jeweils ein Byte. Der Attackwert wird demnach mit 16 multipliziert und auf den Decaywert addiert. Anschließend wird das Byte in Register 5 geschrieben.



[2]

Abbildung 6: Beispielverlauf

```
5       lda  #%00001001
6       sta   $D400+5
7       lda  #%00001001
8       sta   $D400+6
```

Dieser Assemblercode sorgt für einen Ton wie in Abbildung 6.

6 Wellenformen

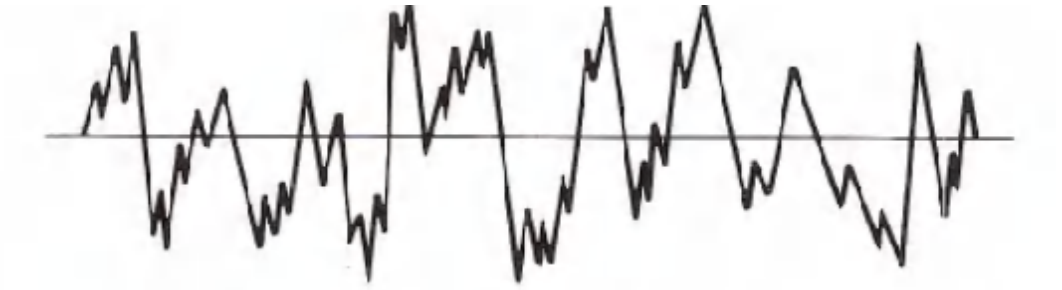
Mit dem C64 ist es möglich, verschiedene Wellenformen für die Töne zu benutzen. Der Registertabelle der Stimme 1 (zu finden in Kapitel 4) kann man entnehmen, dass die Dreieckwelle, die Rechteckwelle, die Sägezahnwelle und Noise zur verfügung stehen. Diese können auch kombiniert werden.

```
11      lda  #%00100001
12      sta   $D400+4
```

Dieses Codebeispiel aktiviert die Sägezahnwellenform und schaltet zusätzlich das Gate auf 1, wodurch ein Ton erklingt.
Die beiden Register 2 und 3 sind für die Pulsweite der Rechteckwelle zuständig.

Mit Hilfe der verschiedenen Wellen in Kombination zu dem ADSR-Verlauf ist es möglich, Instrumente zu simulieren. So eignet sich die Dreieckswellenform gut für ein Xylophon.

Da der C64 auch zum Spielen benutzt wurde, gab es neben der Musik auch die Anforderung Sounds darzustellen. Dazu eignete sich die Wellenform Noise.



[2]

Abbildung 7: Verlauf der Noisewelle

Der Ton erklingt durch diese Welle sehr dumpf. Die Verwendung von Noise bot sich an, um Geräusche, wie beispielsweise Explosionen, darzustellen [3]
Zusätzlich ist es möglich, die verschiedenen Wellenformen zu mischen. Dies erlaubt weitere Möglichkeiten in der Anwendung, wobei es sich am meisten anbietet zwischen den verschiedenen Wellenformen zu wechseln.

7 Filter

Mit dem SID hat man verschiedene Möglichkeiten, den ausgegebenen Ton zu filtern. So gibt es neben einem Tiefpassfilter und einem Hochpassfilter auch einen Bandpassfilter.
Die Grafik "Beispielsituation" in Kapitel 3 zeigt, dass der Filter nicht pro Stimme einstellbar ist. Demnach kann nicht eine Stimme mit einem Tiefpassfilter und eine andere mit einem Hochpassfilter gefiltert werden. Es können aber beide mit dem gleichen Filter gefiltert werden.

															Filter		
21	1	0	1	0	1	15	—	—	—	—	—	FC2	FC1	FC0	FC LO	Write-only	
22	1	0	1	1	0	16	FC10	FC9	FC8	FC7	FC6	FC5	FC4	FC3	FC HI	Write-only	
23	1	0	1	1	1	17	RES3	RES2	RES1	RES0	Filt EX	Filt 3	Filt 2	Filt 1	RES/Filt	Write-only	
24	1	1	0	0	0	18	3 OFF	HP	BP	LP	VOL3	VOL2	VOL1	VOL0	Mode/Vol	Write-only	

[2]

Abbildung 8: Registertabelle des Filters

Aus der Registertabelle in Abbildung 8 kann man ablesen, dass die Frequenz des Filters mit Hilfe der Register 21 und 22 eingestellt wird. Die Resonsanz des Filters und die Stimme, welche gefiltert werden soll, wird über Register 23 eingestellt. Zusätzlich kann im 3. Bit von Register 23 der externe Eingang gefiltert werden.
In Register 24 wird die Lautstärke für alle Stimmen eingestellt. Zusätzlich wählt man hier aus, ob man den Filter als Tiefpass, Hochpass oder Bandpass einstellen möchte.
Folgender Beispielcode zeigt, wie man den Filter benutzen kann:

```
15 POKES+24,26
18 POKES+23,1
19 POKES+22,10
```

Beispielcode 4

8 Abschluss

Zusammengefasst erkennt man, dass das SID schon viele Möglichkeiten bot, Musik auszugeben. Heutzutage nicht mehr zeitgemäß, bot das C64 SID damals fortschrittliche Möglichkeiten, mit seinen Filtern, 3 Stimmen, Wellenformen und dem ADSR Verlauf. Dennoch begeistern Künstler auf etablierten Onlinevideoplattformen bis heute mit ihren Kompositionen auf dem C64.

Literatur

[1] Commodore. C64 programmers reference. 1982.

[2] Commodore. Mos 6581 sound interface device (sid) datenblatt. 1982.

[3] Hecht. Das große commodore 64 buch. 1989.

[4] Avery Lee. Altirra hardware reference manual. 2017.

[5] Andreas Varga. Interview bob yannes.