K U R Z F A S S U N G

(Rückseite kann auch beschrieben werden)

Bundesland:

Name/Anschrift/Telefon

Einzel bzw. Gruppensprecher | 2. Gruppenmitglied | 3. Gruppenmitglied

Heiko Purnhagen

Elsasser Str. 64 2800 Bremen

Tel.: (0421) 3 49 96 55

Alter: 16 Jahre

Schule/Betrieb/Anschrift

Gymnasium Horn

Ronzelenstr. 51 2800 Bremen

Fachgebiet und Thema:

Technik

Digitaler Synthesizer

In meiner Arbeit beschreibe ich die Entwicklung und Funktionsweise eines digitalen Synthesizers. Dieser ist nicht nur in der Lage, Töne künstlich zu erzeugen, sondern auch, sie zu verändern, zu verfremden, oder andere Effekte auf sie anzuwenden (Im augenblicklichen Ausbaustadium ist dieses jedoch nur mit Einschränkungen möglich. Es sind noch Erweiterungen geplant).

Die grundlegende Idee bestand darin, das System möglichst vollständig digital aufzubauen. So werden nur A/D- und D/A-Wandler und die dazugehörigen Tiefpässe und Verstärkerstufen als Analogschaltungen benötigt. Man könnte dabei - mit guten und teuren Wandlern - Signale in Studioqualität erzeugen und verändern oder, bei einer digitalen Aufnahme, die Signale ohne Umweg über Analogstufen direkt aufnehmen und ggf. dann direkt auf einem digitalen Tonträger (CD) speichern.

In meinem digitalen Synthesizer wird die Ausgangsspannung als Funktion der Zeit digital berechnet und mit einem D/A-Wandler in eine Spannung umgewandelt. Der Wandler erhält dabei 30 000 neue Werte pro Sekunde. Das Ausgangssignal wird dann durch einen Tiefpass mit hoher Flankensteilheit geschickt, der die treppenförmige Spannung des Wandlers glättet. So wird die berechnete Zahlenfolge in die analoge Ausgangsspannung des Synthesizers umgewandelt.

Das größte Problem war nun, die Ausgangsspannung mit einer so hohen Rate und möglichst einer Genauigkeit von 16 Bit in Echtzeit zu berechnen. Für diese Berechnungen benötige ich einen Digitalrechner, der einen Ausgangsspannungswert in etwa 30 us berechnen kann. Diese Rechenleistungen werden von sogenannten digitalen

Signalprozessoren (DSP) erfüllt. Sie dienen normalerweise zu Zwecken wie der Spracherkennung und -synthese, zur digitalen Filterung, Verarbeitung oder auch Erzeugung von Signalen bzw. für Steuerungszwecke. Für meine Zwecke schien mir dabei der Signalprozessor TMS 32010 am geeignetsten.

Einen Teil meines Wissens über digitale Signalverarbeitung konnte ich während eines Aufenthaltes in einem Labor für Sprachverarbeitung der Fraunhofer Gesellschaft in Stuttgart erwerben, der mir aufgrund einer Forschungspatenschaft möglich war. Ich habe dort an der Entwicklung eines leistungsfähigen mikroprogrammierbaren Signalverarbeitungsprozessors für Spracherkennungsanwendungen mitgearbeitet.

Zusätzlich zu dem schnellen Signalverarbeitungsrechner sind noch zwei weitere Mikrocomputer notwendig. Einer dient zur Berechnung der Eingangsparameter für den Signalprozessor. Der andere fragt das Keyboard ab.

Die Vielfalt der bei der Entwicklung dieses digitalen Synthesizers auftretenden Probleme machten seinen Aufbau zu einer schwierigen aber reizvollen und interessanten Arbeit.

D I G I T A L E R S Y N T H E S I Z E R

Ein Beitrag zum Wettbewerb " JUGEND FORSCHT " 1986

Gliederung

von Heiko Purnhagen, Bremen, Elsasser Str. 64

(geb. 2.4.1969) Schüler des Gymn. Horn, Bremen

1. Einleitung

In meiner Arbeit beschreibe ich die Entwicklung und Funktionsweise eines digitalen Synthesizers. Dieser ist nicht nur in der Lage, Töne künstlich zu erzeugen, sondern auch, sie zu verändern, zu verfremden, oder andere Effekte auf sie anzuwenden (Im augenblicklichen Ausbaustadium ist dieses jedoch nur mit Einschränkungen möglich. Es sind noch Erweiterungen geplant).
Die grundlegende Idee bestand darin, das System möglichst vollständig digital aufzubauen. So werden nur A/D- und D/A-Wandler und die dazugehörigen Tiefpässe und Verstärkerstufen als Analogschaltungen benötigt. Man könnte dabei - mit guten und teuren Wandlern - Signale in Studioqualität erzeugen und verändern oder, bei einer digitalen Aufnahme, die Signale ohne den Umweg über Analogstufen direkt aufnehmen und ggf. dann direkt auf einem digitalen Tonträger (CD) speichern.
Einen Teil meines Wissens über digitale Signalverarbeitung konnte ich während eines Aufenthaltes in einem Labor für Sprachverarbeitung der Fraunhofer-Gesellschaft in Stuttgart erwerben, der mir aufgrund einer Forschungspatenschaft möglich war. Ich habe dort an der Entwicklung eines leistungsfähigen mikroprogrammierbaren Signalverarbeitungsprozessors für Spracherkennungsanwendungen mitgearbeitet.

2. Funktionsprinzipien von Synthesizern

Die ersten Synthesizer arbeiteten nur mit analogen Modulen und waren wie eine Art Analogrechner aufgebaut. Die einzelnen Module wurden von (analogen) Spannungen gesteuert und konnten untereinander mit Kabeln oder Kreuzschienenverteilern verbunden werden bzw. waren fest miteinander verbunden. Die wichtigsten Module sind:

VCO - voltage controlled oscillator - ein spannungsgesteuerter (Frequenz) Oszillator mit verschiedenen Kurverformen,
VCA - voltage controlled amplifier - ein spannungsgesteuerter (Verstärkung) Verstärker,
VCF - voltage controlled filter - ein spannungsgesteuerter (Eckfrequenz) Filter (meist Tiefpass mit 12 - 24 dB/Oktave),
ENV - envelope generator - ein vom Keyboard gesteuerter Hüllkurvengenerator und
LFO - low frequency oscillator - ein langsamer Oszillator für Vibrato und ähnliche Effekte.
Zusätzlich gibt es natürlich auch Mischer, mit denen man mehrere Signale mixen kann.

Das Keyboard liefert dabei ein Key-Signal, das angibt, ob eine Taste gedrückt ist oder nicht, und eine Spannung, die mit meistens 1 V / Oktave die Tonhöhe angibt. So ist nur monophones Spiel möglich, also nur ein Ton zur Zeit. Will man dagegen polyphon spielen, benötigt man ein Keyboard mit mehreren solcher Ausgänge. An jeden dieser Ausgänge muß man dann wieder einen eigenen monophonen Tonerzeugungsblock anschließen.
Nachteile dieses Prinzips ist, daß große Mengen äußerst präziser analoger Bauelemente und ICs verwendet werden müssen, wobei auch das Temperaturverhalten beachtet werden muß. Besonders kritisch sind hier die VCO's.
Verschiedene Klangfarben lassen sich hier im wesentlichen dadurch realisieren, daß man ein obertonreiches Signal (z.B. einen Sägezahn) erzeugt und dann einen Teil der Oberwellen wieder wegfiltert.

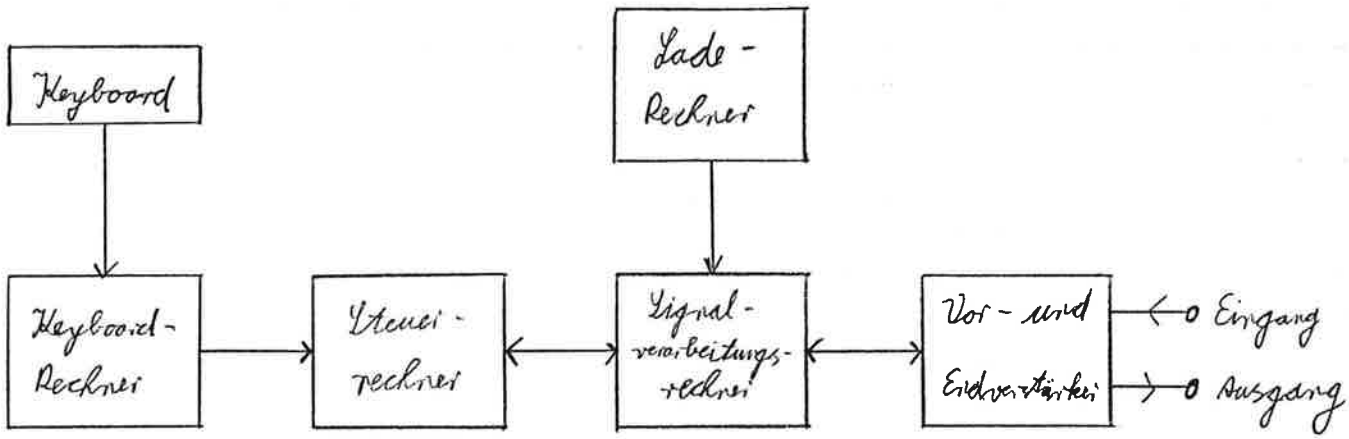
Seit einigen Jahren gibt es auch Synthesizer auf dem Markt, die - teilweise digital - verschiedene Klangfarben erzeugen, indem sie mehrere Sinusschwingungen sich gegenseitig frequenzmodulieren lassen.

3. Aufbau des digitalen Synthesizers

Ich habe nun versucht, einen vollständig digitalen Sythesizer zu bauen, der zunächst einen aus Modulen aufgebauten analogen Synthesizer simuliert. Ihn und seine Entwicklung werde ich im folgenden beschreiben. Dabei bin ich auf einige, teilweise sehr große, Probleme gestoßen, die ich, zusammen mit den jeweiligen Lösungsmöglichkeiten, an der betreffenden Stelle aufzeigen möchte.
Prinzipiell wird die Ausgangsspannung am Synthesizer als Funktion der Zeit digital berechnet. Diese wird dann mit einem D/A-Wandler in eine Spannung umgewandelt. Der Wandler erhält etwa 20 000 bis 40 000 neue Werte pro Sekunde. Das Ausgangssignal wird dann durch einen Tiefpass mit hoher Flankensteilheit geschickt, der das treppenförmige Signal vom Wandler glättet. So wird also die berechnete Zahlenfolge in die analoge Ausgangsspannung des Synthesizers ungewandelt.
Das größte Problem war nun, die Ausgangsspannung mit einer so hohen Rate und möglichst einer Genauigkeit von 16 Bit in Echtzeit zu berechnen. Für diese Berechnungen benötigte ich einen Digitalrechner, der die notwendigen Berechnungen für einen Ausgangsspannungswert in etwa 30 us ausführen kann. Diese ist die Periodendauer bei der von mir gewählten Wandelrate von 30KHz. Diese Rechenleistungen werden von sogenannten digitalen Signalprozessoren (DSP) erfüllt. Sie dienen normalerweise zu Zwecken, wie der Spracherkennung und -synthese, zur digitalen Filterung, Verarbeitung oder auch Erzeugung von Signalen bzw. für Steuerungszwecke. Für meine Zwecke schien mir dabei der Signalprozessor TMS 32010 am geeignetsten.

Mit dem TMS 32010 habe ich nun einen Signalverarbeitungsrechner aufgebaut und als Tongenerator programmiert. Er simuliert die VCO's, VCA's, Rauschgeneratoren und Mischer. Man könnte auf ihm auch VCF's und Effektgeräte bzw. Programme zur sogenannten FM-Synthese programmieren, jedoch wird es dann mit der zur Verfügung stehenden Rechenzeit wieder sehr knapp.
Die sich langsamer ändernden Steuerspannungen werden dagegen auf einem anderen Rechner berechnet. In meinem Fall ist dieses ein Rechner mit einer 8085 CPU mit 3 MHz Taktfrequenz. Seine Rechenleistung genügt, um mit 100 Hz bis 200 Hz mehrere (niederfrequente) LFO's, ENV's, VCA's und Mischer zu simulieren. Die Rate von nur 100 Hz bis 200 Hz, mit der neue Werte für die Oszillatoren berechnet werden, ruft, bei Verwendung eines Tricks (siehe 4.3.), keine hörbaren Störgeräusche hervor. Dieser Steuerrechner gibt über ein geeignetes Interface seine Informationen an den Signalverarbeitungsrechner weiter.
Aber auch der Steuerrechner selber benötigt nun wieder Informationen darüber, welche Tasten auf dem Keyboard gedrückt sind. Auch muß der Synthi-Spieler Parameter während des Spiels verändern können (z.B. Lautstärke und Modulationsstärke). Diese Informationen werden nun von einem dritten Rechner ermittelt, der das Keyboard und ggf. mehrere Regler abfragen kann und seine Information dann an den Steuerrechner weitergibt. Dieses geschieht z.Z. noch mit einer recht langsamen seriellen Schnittstelle (1200 Bd). Anstelle der Keyboardabfrage könnte man auch einen Sequencer programmieren, der dann kleine Melodien und Tonfolgen automatisch spielen kann.

Blockschaltbild des digitalen Synthesizers



4. Der Signalverarbeitungsrechner

Nachdem ich einen geeigneten Signalverarbeitungsprozessor, den TMS 32010, gefunden hatte, mußte ich mit ihm einen Rechner aufbauen. Der Speicher dieses Rechners muß von einem anderen Rechner aus ladbar sein. Dieser Lade-Rechner muß auch das Reset-Signal für den TMS 32010 erzeugen können. Auch ein A/D- und ein D/A-Wandler mußten an ihn angeschlossen werden. Zusätzlich war ein Interface notwendig, mit dem der Signalverarbeitunsprozessor Informationen von einem Steuerrechner übernehmen konnte.

4.1. Der Signalprozessor TMS 32010

Von den mir bekannten Signalprozessoren schien mir der TMS 32010 von Texas Instruments für meine Anwendung am besten geeignet, da er über die notwendige Rechenleistung verfügt, einfach zu programmieren ist und man an ihn direkt 4K Word RAM und bis zu 8 Ein- und Ausgabekanäle anschließen kann. Es können dabei sowohl Programme als auch Daten im RAM abgelegt werden, auf das er in normalerweise 200 ns zugreifen kann. Sein internes Daten-RAM hat eine Kapazität von 144 Words. So kann er, mit maximal 20 MHz getaktet, fast 5 MIPS (million instructions per second = Millionen Befehle pro Sekunde) ausführen. Der Befehlstakt liegt also bei 1/4 der Taktfrequenz.
Der TMS 32010 besitzt einen Hardware-Multiplizierer, der in 200 ns, also innerhalb eines Befehls, eine vorzeichenbehaftetete 16*16-Bit-Multiplikation ausführen kann und einen 32 Bit breiten Akkumulator und eine ebenso breite ALU. Er hat so beim Multiplizieren, einer bei der Signalverarbeitung sehr wichtigen Operation, mehr als die 1000-fache Rechengeschwindigkeit eines normalen 8-Bit Mikroprozessors (z.B. 8085 / 3 MHz). Einen Multiplikations-Akkumulier-Zyklus, die Hauptoperation eines digitalen Filters, kann er in normalerweise 400 ns ausführen.

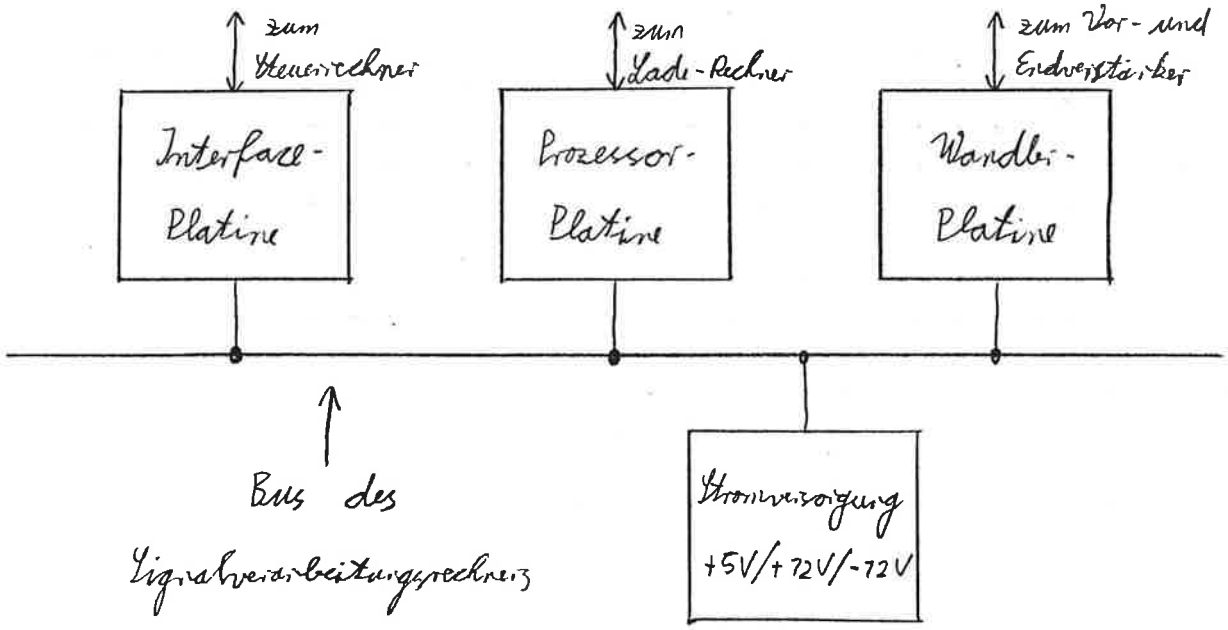
4.2. Hardware

Im folgenden beschreibe ich die Hardware des von mir entwickelten Signalverarbeitungsrechners.

4.2.1. Prinzipieller Aufbau

Da ich schon bei ersten überlegungen über den Aufbau des Signalverarbeitungsrechners festgestellt habe, daß er sich sicher nicht auf einer einzigen Euro-Karte (100 x 160 mm) unterbringen läßt, habe ich mich dazu entschlossen, ihn auf mehreren Euro-Karten aufzubauen und diese mit einer Buskarte (motherboard) untereinander zu verbinden. Als Steckverbindung habe ich normale 64-polige Messerleisten verwendet. Die Karten selber habe ich in der Fädeltechnik aufgebaut, da man so auch bei einer hohen Bestückungsdichte (die Prozessor-Platine hat 22 ICs) eine Karte in relativ kurzer Zeit verdrahten kann. Ich habe normalerweise zu Aufbau einer Karte 1 bis 2 Tage benötigt. Zum anderen kann man Verdrahtungsfehler auf gefädelten Platinen sehr leicht beheben.
Auf der Busplatine liegen neben den Versorgungsspannungen von 5 V, +12 V und -12 V und Masse der 16 Bit breite Datenbus, 16 Selectleitungen für die je 8 Ein- und Ausgabeports und die Steuersignale Reset, Clockout, INT und BIO. So kann man neben der Prozessor-Platine je nach Bedarf eine oder mehrere Ein- und Ausgabe-Platinen auf den Bus stecken. Im Augenblick sind dieses bei mir eine Wandler-Platine und eine Interface-Platine. Ich plane noch für Sampler-Anwendungen eine RAM-Platine mit etwa 64 bis 256 K Word DRAM, die über I/O-Ports angesprochen werden würde. Für alle Logik-ICs habe ich ICs aus der TTL-Serie 74LS... verwendet. Die Stromversorgung wird direkt in die Busplatine eingespeist und wird mit einfachen Linearreglern stabilisiert.

Blockschaltbild des Signalverarbeitungsrechners

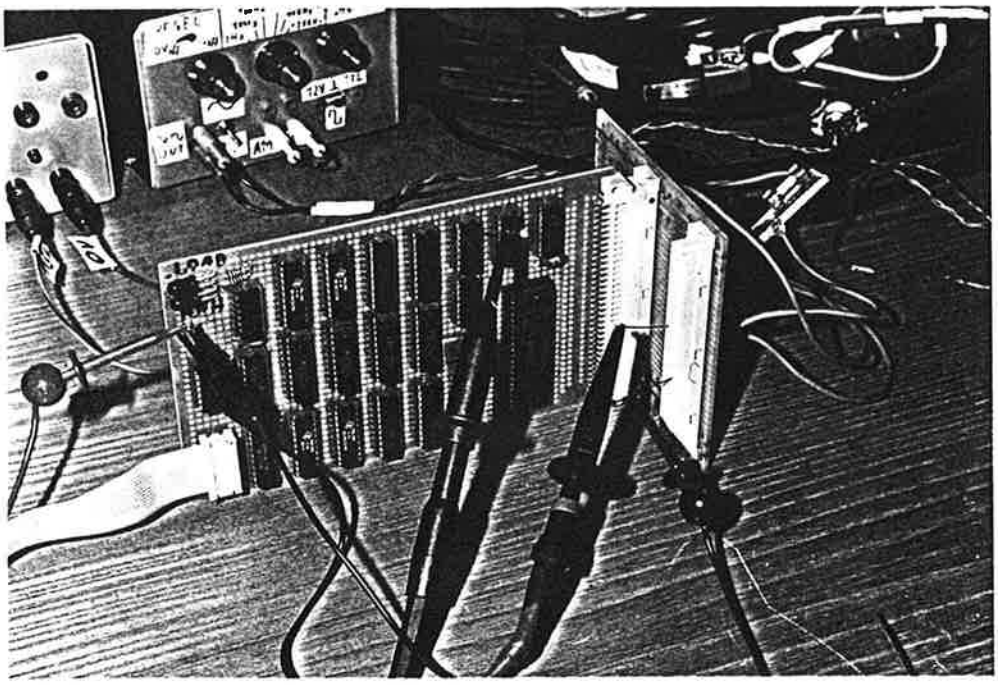


4.2.2. Die Prozessor-Platine

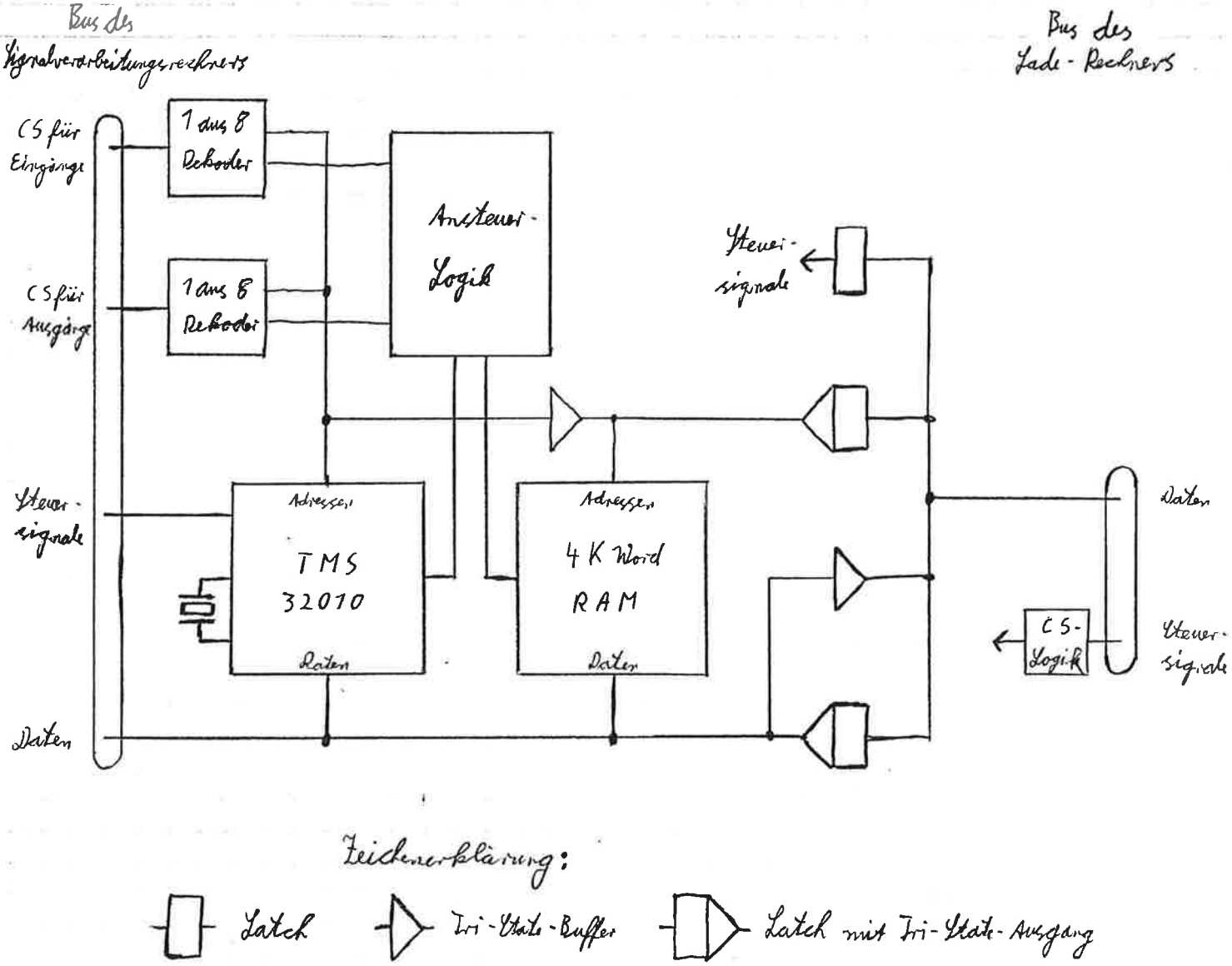
Auf der Prozessor-Platine habe ich neben dem Signalprozessors TMS 32010 selber auch das von ihm benötigte RAM und eine Interface untergebracht, mit dem ich dem TMS 32010 steuern kann und das RAM schreiben und lesen kann.
Da ich den TMS 32010 mit seiner maximalen Taktfrequenz von 20 MHz takten wollte, benötigte ich relativ schnelle RAMs. Da für einen

vollständigen Speicherzugriff auf das externe RAM insgesamt nur 200 ns zur Verfügung stehen, mußte es eine Zugriffszeit von weniger als 70 ns haben. Insgesamt wollte ich den Speicher gleich von Anfang an auf die vom TMS maximal adressierbare Kapazität von 4K Word ausbauen. Ich habe daher vier statische CMOS-RAMs HM6168HP-55 verwendet, die zu je 4K mal 4 Bit organisiert sind und eine Zugriffszeit von 55 ns haben. Diese RAMs kann man nun direkt an den TMS 32010 anschließen.
Da ich aber auch Ein- und Ausgabeeinheiten an den TMS anschließen wollte, mußte ich auch die Chip-Select-Signale für die maximal 8 Ein- und Ausgabeeinheiten erzeugen. Hierzu verwende ich zwei 1 aus 8 Dekoder-ICs. Deren insgesamt 16 Ausgänge und der Datenbus des TMS werden nun auf den 64-poligen Stecker zur Busplatine geführt. Zusätzlich führe ich auch die beiden Signale Reset und Clockout auf den Busstecker. Die beiden Eingänge INT und BIO habe ich auch am Busstecker angeschlossen. Sie haben je einen 10 K - Pullupwiderstand gegen 5 V. Das Signal am BIO-Eingang kann vom Signalprozessor über einen speziellen Befehl abgefragt und ausgewertet werden (siehe 4.2.4.).
Nun mußte aber auch der Lade-Recher auf das RAM des TMS zugreifen können und außerdem das Reset-Signal für den TMS erzeugen. Hierfür war ein Interface nötig, das ich auch auf der Prozessor-Platine aufgebaut habe. Es wird mit dem Lade-Rechner über ein 16-poliges Flachbandkabel verbunden, auf dem ein Teil der Signale des Busses des Laderechners liegen. über Bus kann nun der Lade-Rechner zum einen das Reset-Signal setzen und, wenn Reset aktiv ist, auch Daten und Adressen an das RAM des TMS anlegen und einschreiben bzw. ein Lese-Signal erzeugen und dann die Datenleitungen des RAMs abfragen. Der aktuelle Pegel des Reset-Signales wird mit zwei Leuchtdioden auf der Vorderseite der Platine angezeigt.

Test der Prozessor-Platine



Blockschaltbild der Prozessor-Platine



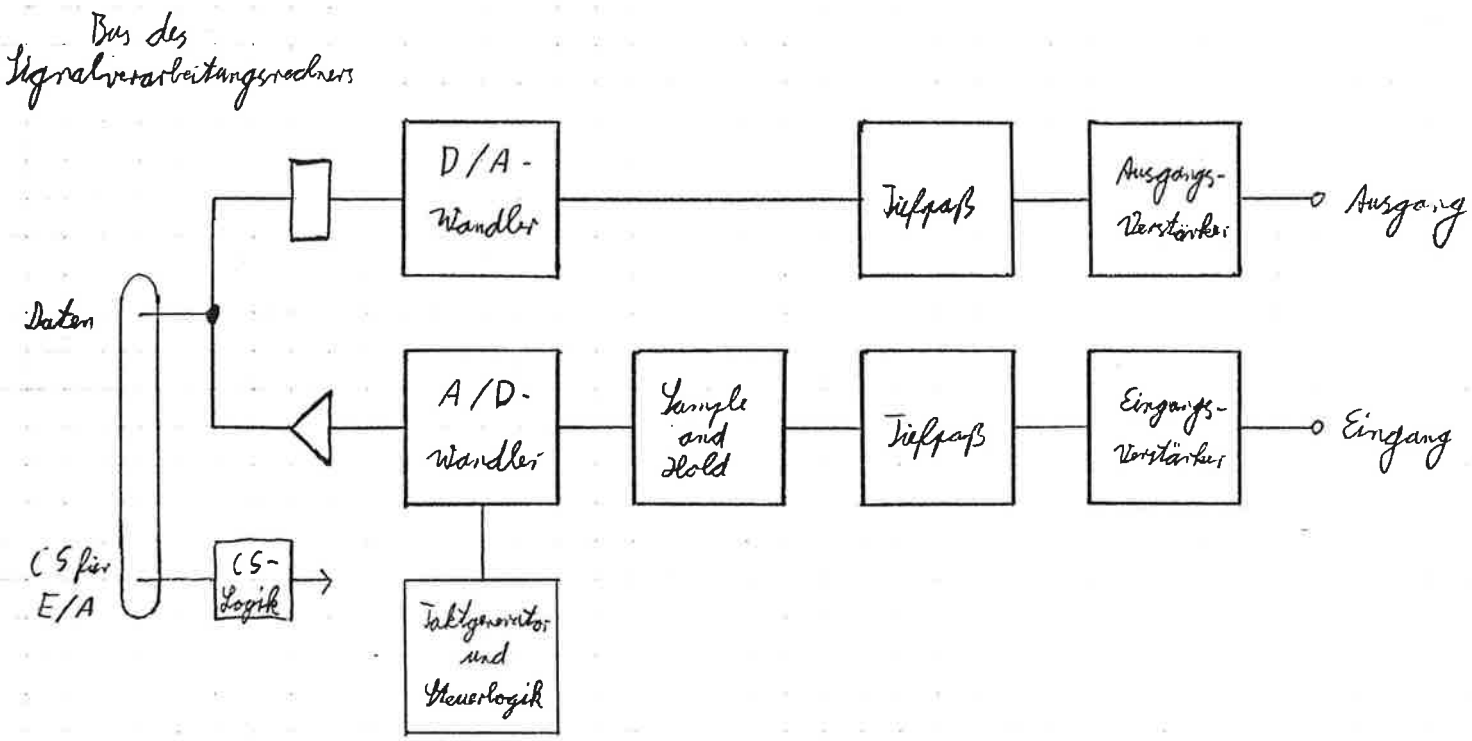
4.2.3. Die Wandler-Platine

Die Wandler-Platine dient dazu, die vom Signalprozessor berechnete digitale Ausgangsspannung in eine analoge Spannung zu wandeln und das treppenförmige Signal zu glätten. Zusätzlich kann eine analoge Spannung wieder in einen digitalen Wert gewandelt werden. Dieses benötige ich, wenn ich den Signalverarbeitungsrechner als Effektgerät verwenden will. Die Wandler-Platine ist mit der Prozessor-Platine über den Bus verbunden. Die auf dieser Platine befindlichen Tiefpässe dienen dazu, die Ein- und Ausgangssignale auf eine Bandbreite von der halben Abtastfrequenz zu begrenzen. Dieses ist aufgrund des Abtast-Theorems notwendig. Man kann auch sagen, das Ausgangs-Filter dient dazu, das treppenförmige Signal des D/A-Wandlers zu glätten.
Zunächst war es notwendig, passende D/A- und A/D-Wandler-ICs zu finden. Für den D/A-Wandler ist eine Auflösung von 12 Bit sinnvoll, da man mit dieser ein Signal mit einem Stör- zu Nutzsignalverhältnis (S/N) von bis zu 70 db erzeugen kann. Dieser Wert,

auch Dynamik genannt, liegt in der Größenordnung des Bandrauschens eines guten HiFi-Cassettendecks, reicht also für meine Zwecke vollständig aus. D/A-Wandler-ICs mit dieser Auflösung sind noch recht kostengünstig (ca. 35 DM). Ich habe mich hierbei für den DAC800P entschieden. Größere Probleme bereitete da schon die Auswahl des A/D-Wandlers. Bei einer Wandelzeit von weniger als 25 us (entsprechend einer Abtastrate von 40 KHz) kommen nur Wandler in Frage, die nach dem Succesive-Approximation-Prinzip arbeiten. Sie kosten aber, bei 12 Bit Auflösung, mehr als 100 DM, so daß ich mich vorläufig für das wesentlich günstigere 8-Bit-Wandler-IC ZN427 entschieden habe. Es wäre problemlos möglich, später einen A/D-Wandler mit höherer Auflösung zu verwenden, da keine Änderung der Software für den TMS 32010 notwendig wäre. Man könnte auch, bei veränderter Software und Verwendung von je zwei A/D- und D/A-Wandlern, stereophone Signale erzeugen und verändern.

Meine jetzige Wandler-Platine wird über je einen Ein- und einen Ausgang des TMS angesprochen. Die Portadresse kann über je eine Steckbrücke gewählt werden. Mit dem Chip-Select-Signal für den gewählten Ausgabeport wird der am Datenbus anliegende Wert in ein 16 Bit breites Latch übernommen. An seinen 12 höchstwertigen Ausgängen habe ich die Eingänge des D/A-Wandler-ICs angeschlossen. Die Ausgangsspannung des Wandlers wird dann über einen Butterworth-Tiefpaß 3. Ordnung geschickt. Seine Eckfrequenz liegt bei 15 KHz, und er besitzt eine Flankensteilheit von 18 db / OKtave. Danach folgt ein Pufferverstärker für den Ausgang mit einstellbarer Verstärkung. Er liefert eine Ausgangsspannung von etwa 1 Vss.
Die Eingangsstufe des A/D-Wandlers besteht nun zuerst aus einem Vorverstärker mit einer einstellbaren Verstärkung zwischen 1 und 100. Sein Ausgang ist mit dem Eingang eines weiteren Tiefpasses verbunden, der in der gleichen Weise aufgebaut ist, wie der Tiefpaß für den D/A-Wandler. Nachdem das Eingangssignal den für den A/D-Wandler notwendigen Pegel hat, folgt eine Sample-and-Hold-Schaltung. Sie ist notwendig, damit sich das Eingangssignal des Wandlers während der Wandlung nicht verändert. Ich habe ihn mit einem CMOS-Analogschalter, einem Kondensator und einem Impedanzwandler realisiert. Die Ausgangsspannung des Impedanzwandlers geht nun direkt an den Eingang des A/D-Wandler-ICs. Dieses wird mit einer Freqeunz von 1 MHz getaktet und benötigt so 10 us für eine Wandlung. Eine Wandlung wird immer dann gestartet, wenn gerade der zuletzt gewandelte Wert vom Signalprozessor gelesen wurde. Die Digital-Ausgänge des Wandlers gehen dann über einen Tri-State-Buffer an den Datenbus. Dieser Buffer wird bei einer Abfrage des A/D-Wandlers geöffnet. Für die insgesamt 5 Operationsverstärker auf der Wandler-Platine habe ich den CMOS-Typ ICL7611DCPA verwendet, da er sehr unkompliziert zu benutzen ist.
An die analogen Ein- und Ausgänge habe ich einen Kopfhörerverstärker und einen Mikrofonvorverstärker angeschlossen, die sich in einem eigenen Gehäuse befinden, das mit DIN-Steckverbindungen für ein Mikrofon und einen Verstärker, Mischpult oder Cassetten-Deck versehen ist.

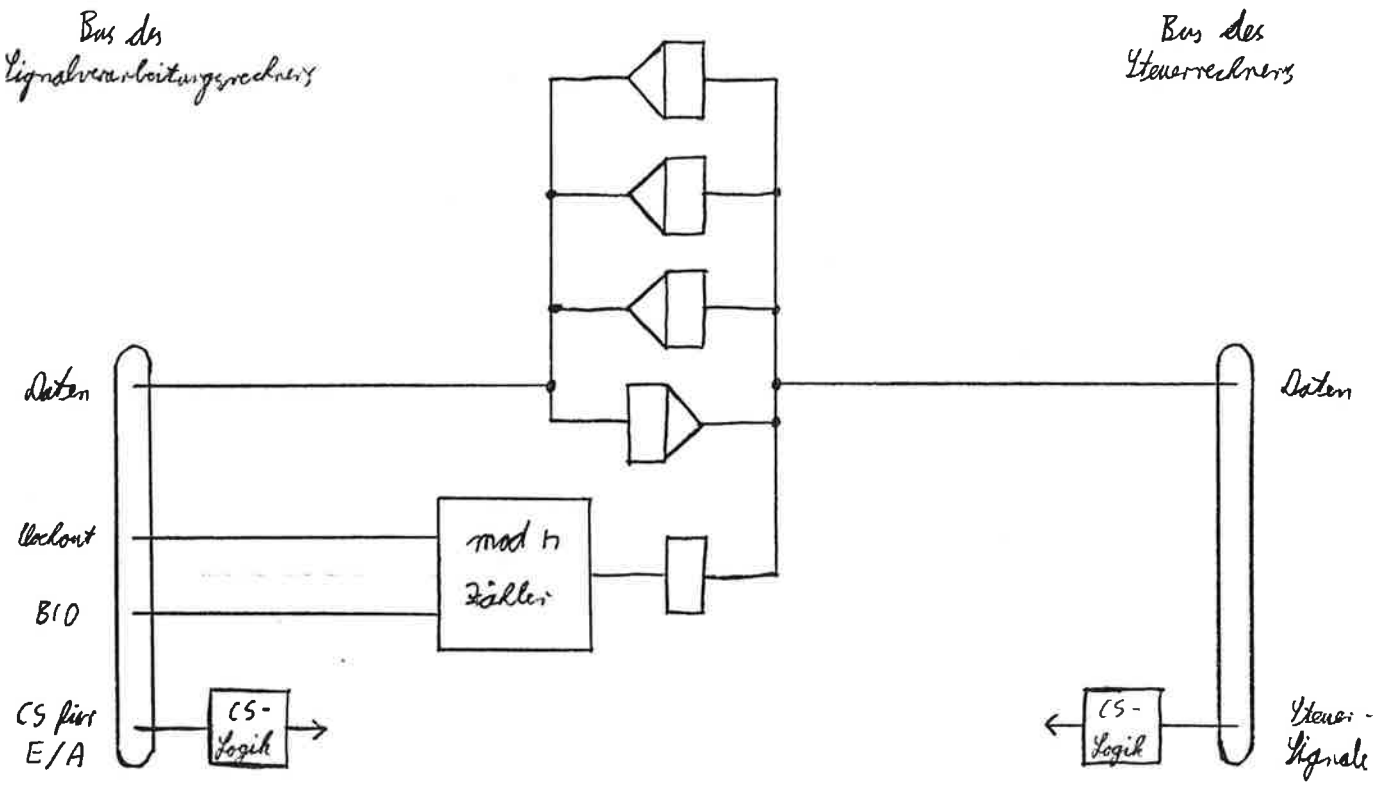
Blockschaltbild der Wandler-Platine



4.2.4. Die Interface-Platine

Die Interface-Platine dient zum Datenaustausch zwischen dem Signalverarbeitungsrechner und dem Steuerrechner. Zusätzlich habe ich auf ihr einen programmierbaren Teiler aufgebaut, der den Taktausgang des TMS von 5 MHz durch einen Faktor zwischen 1 und 4096 teilt. Er dient zur Erzeugung einer festen Wandelrate.
Die Interfacekarte wird mit dem Steuerrechner über ein 16-poliges Flachbandkabel verbunden, auf dem ein Teil des Busses des Steuerrechners liegt. Der Steuerrechner kann über dieses Kabel 4 je 16 Bit breite Latches beschreiben. Diese Latches haben Tri-State-Ausgänge. Die Chip-Select-Eingänge von drei ICs können über Steckbrücken mit je einer der 8 Eingangs-Chip-Select-Leitungen von der Prozessorplatine verbunden werden. Die Ausgänge dieser drei Latches sind mit dem Datenbus des TMS verbunden. Er kann so die in diesen Latches gespeicherten Daten lesen. Umgekehrt kann der TMS über einen Ausgabebefehl ein fünftes 16 Bit breites Latch beschreiben. Das verwendete Chip-Select-Signal kann wieder mit einer Steckbrücke ausgewählt werden. Dieses Latch kann dann von Steuerrechner gelesen werden. Das vierte vom Steuerrechner beschreibbare Latch liefert das Teilungsverhältnis für den Frequenzteiler. Den Ausgang des Teilers habe ich mit mit dem BIO-Eingang des TMS verbunden. Dieser kann sich nun, indem er diesen Eingang abfragt, damit synchronisieren und somit die Wandler mit einer konstaten Abtastrate bedienen. Normalerweise habe ich so eine Wandelrate von 30 KHz eingestellt.

Blockschaltbild der Interface-Platine



4.3. Software für den TMS 32010

Die Software für den TMS 32010 dient dazu, Töne oder Geräusche mit einer vom Steuerrechner vorgegeben Frequenz und Amplitude zu erzeugen, zu mischen und auf den D/A-Wandler auszugeben. Auch könnte die Software Effektgeräte simulieren, zur Zeit ist es jedoch nur möglich, den analogen Eingang zum Ausgang dazuzumischen.
Das Programm für den TMS darf aufgrund der Abtastrate von normalerweise 30 KHz nicht mehr als 33 us * 5 MHz = 165 Taktzyklen pro Durchlauf benötigen. Dieses entspricht etwa 150 Befehlen, da fast alle Befehle nur 1 Taktzyklus benötigen.

Um die Programme für den TMS nicht von Hand in den Speicher des Lade-Rechners eingeben zu müssen, habe ich mir eine Art Assembler geschrieben, dem man die einzelnen Befehle zwar noch in ASCII-Hex angeben muß, der aber in der Lage ist, Labels für Adressen zu benutzen, so daß man nicht beim Einfügen von Befehlen alle Sprungadressen neu eingeben muß. Dieses Programm, ich habe es Nanoassembler genannt, erhält als Eingabe ein Textfile, in dem die Befehle in Hex gespeichert sind. Es können auch Pseudooperationen für die Labelverarbeitung in diesem Text stehen. Zusätzlich ist es möglich, jede Zeile mit einem beliebigen Kommentar zu versehen. So schreibe ich hinter jedes Befehls-Word die jeweilige Mnemonik des Befehls. Der Nanoassembler speichert, nachdem er das Textfile ausgewertet hat, das übersetzte Programm in das RAM des TMS.

In meinen ersten Versuchen zu Tonerzeugung habe ich mich zunächst mit der Erzeugung von Sägezahn-Tönen beschäftigt. Dazu wird in jedem Durchlauf des TMS-Programmes ein Zähler um einen vorgegebenen Wert erhöht. Die im Zähler gespeicherte Zahl kann man nun auf den D/A-Wandler ausgeben. Man muß aber vorher das höchstwertigste Bit invertieren, um den Zählerstand, der vom TMS als Zahl in Zweierkomplement-Schreibweise behandelt wird, in eine Binary-Offset-Zahl zu verwandeln, die der Wandler als Eingangswert benötigt. Wenn man nun einen Freqenzwert wählt, der keine Potenz von 2 ist, hört man neben dem eigentlichen Ton auch "Störungen", deren Frequenzen unter der des Sägezahns liegen. Diese entstehen dadurch, daß man ein nicht bandbegrenztes Signal auf den Wandler gibt und so das Abtastheorem verletzt. Dieser Effekt wird um so stärker, je höher die Frequenz des Sägezahns wird. Eine Sägezahnschwingung enthält nämlich alle nur möglich Oberwellen. Die n-te Oberwelle hat dabei die n-fache Frequenz der Grundschwingung und deren 1/n-fache Amplitude.
Es ist also nicht sinnvoll, Töne zu erzeugen, deren Oberwellen über der halben Abtastfrequenz liegen. Da aber die Oberwellen im wesentlichen die Klangfarbe eines Tones bestimmen und daher unbedingt notwendig sind, Kann man sie nicht einfach außer acht lassen. Eine Möglichkeit, dieses Problem zu lösen, besteht darin, jeweils neben dem Grundton nur die Oberwellen, also Sinusschwingungen, zu erzeugen, die nicht oberhalb der halben Abtastrate liegen. Diese Lösung benötigt aber sehr viel Rechenzeit. Eine andere Möglichkeit besteht darin, die Töne anhand einer Tabelle zu erzeugen, in der die Kurvenform einer Periode mit den jeweils in einem bestimmten Frequenzbereich erlaubten Oberwellen enthalten ist. Man müßte dabei für jede Oktave die dazugehörige Kurvenform speichern oder auf hochfrequente Oberwellen ganz verzichten.

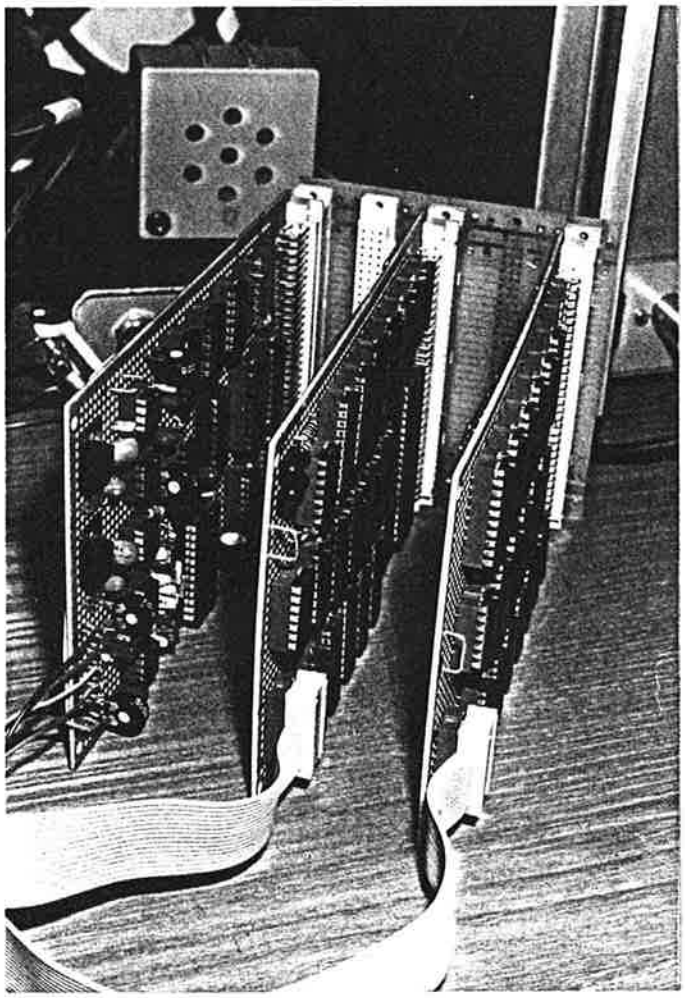
Zur Zeit verwende ich jedoch nur einzelne Sinusschwingungen, die ich anhand einer Tabelle mit 128 Sinuswerten erzeuge, zwischen denen linear interpoliert wird. Dieses Programm habe ich mit Kleinen Veränderungen nach einem Beispielprogramm von Texas Instruments geschrieben, da es vollständig in Bezug auf Ausführungszeit optimiert ist. Es erhöht einen Winkel-Zähler je einmal pro Durchlauf um einen frequenzbestimmenden Wert und ermittelt dann den Sinus dieses Winkels. Es benötigt 23 Befehle, also 4,6 us.
Ich habe auch einen Rauschgenerator programmiert, der ein rückgekoppeltes 16-Bit-Schieberegister simuliert und so eine pseudozufällige Zahlenfolge erzeugt.

Meine erste Version des Tongenerator-Programms für den TMS war nun in der Lage, gleichzeitig 3 Sinustöne, 1 Rauschsignal und den analogen Eingang mit verschiedenen Amplituden zu mischen, dabei auf einen überlauf zu testen und ggf. ein Clipping, also die Begrenzung auf den Höchst- und Tiefstwert des Wandlers, durchzuführen und auszugeben. Die benötigten Parameter werden über die Interface-Platine übergeben. Dieses sind je ein Amplituden- und Frequenzwert für die 3 Sinusgeneratoren und zwei weitere Amplitudenwerte für den Rauschgenerator und den Eingang. Dazu wird in ein Word des Interfaces die Nummer des zu ändernden Parameters (0-7) und in ein anderes der neue Parameterwert geschrieben. Dann wird für die Dauer eines Programmdurchlaufes eine Bit des dritten Words gesetzt. Das TMS-Programm fragt nun einmal pro Durchlauf dieses Word ab und übernimmt ggf. den neuen Wert in seinen Datenspeicher. Auf das vom Steuerrechner lesbare Word wird der Inhalt eines Zählers geschrieben, der einmal je Durchlauf um 1 erhöht wird, und mit dem dem Programm des Steuerrechners eine genaue Zeitbasis zur Verfügung gestellt wird. Da auch der Steuerechner

häufig eine 16*16-Bit-Multiplikation durchführen muß, ein Programm dazu aber etwa 500 us benötigen würde, habe ich das TMS-Programm so geschrieben, daß, falls gerade kein Parameter geändert wird oder der Zähler nicht abgefragt wird, das Produkt der in zwei Words des Interfaces gespeicherten Zahlen in das vom Steuerrechner lesbare Word geschrieben wird. Dieses Programm benötigte maximal etwa 32 us, nutzte die ihm zu Verfügung stehende Zeit also voll aus.

Als ich den Tongenerator nun zum ersten Mal vom Steuerrechner steuern ließ, stellte sich heraus, daß sich der erzeugte Ton sehr rauh anhörte, wenn man den Amplitudenwert eines Sinustones oder des Eingangs relativ schnell veränderte (Alle Parameter werden 100 - 200 mal pro Sekunde verändert; siehe 5.1.). Dieses kommt daher, daß man einen Sprung im Ausgangssignal erhält, den man als Knacken wahrnimmt, wenn man bei einem Signal, das gerade nicht den Wert Null hat, den Amplituden-Faktor ändert. Treten diese Knack-Geräusche häufig, eben 100 bis 200 mal pro Sekunde, auf, so hört sich das Signal rauh an. Dieses läßt sich verhindern, wenn man diese Sprünge im Amplituden-Faktor verkleinert. Dazu habe ich die Amplituden-Parameter für die Sinustöne und den Eingang über je einen digitalen Tiefpaß erster Ordnung geleitet, der sich auf dem TMS recht einfach programmieren läßt. Die Eckfrequenz dieses Tiefpasses liegt bei etwa 20 bis 50 Hz, so daß sich der Amplituden-Faktor nicht mehr sprunghaft sondern kontinuierlich ändert und sich das Ausgangssignal wieder "sauber" anhört. Da aber nun diese Filter selber wieder Rechenzeit benötigen, mußte ich die Anzahl der Sinusgeneratoren auf 2 reduzieren.

Der Signalverarbeitungsrechner in seinem momentanen Ausbaustadium



5. Der Steuerrechner

Der Steuerrechner dient dazu, die "Steuerspannungen" für das Tongenerator-Programm im Signalverarbeitungsrechner zu berechnen. Er ist z.Z. mit 8085 CPU / 3 MHz ausgestattet.

5.1. Prinzipielle Wirkungsweise

Die prinzipielle Wirkungsweise des Steuerrechners ähnelt sehr der des Signalverarbeitungsrechners. So werden in einem Programm die für die Tongeneratoren notwendigen Steuerwerte, die den Steuerspannungen bei einem analogen Synthesizer entsprechen, berechnet. Diese Berechnung geschieht etwa 100 bis 200 mal pro Sekunde. Dazu reicht die Rechenleistung eines normalen 8085 / 3 MHz Rechners aus. Die Verknüpfung der einzelnen im Steuerrechner simulierten Module geschieht anhand einer Tabelle. Die so berechneten Steuerwerte werden dann über die Interface-Platine an den Signalprozessor übergeben. Die recht niederige Rate, mit der die Parameter des Tongenerator-Programmes geändert werden, ruft dennoch, bei Verwendung eines kleines Tricks (siehe 4.3.), keine hörbaren Störgeräusche im Ausgangssignal des Synthesizers hervor.

5.2. Hardware

Die Hardware besteht momentan aus einem Mikrocomputer mit einer 8085 CPU / 3 MHz. Er ist über ein Flachbandkabel mit der Interface-Platine des Signalverarbeitungsrechners verbunden und über eine serielle Schnittstelle (1200 Bd) mit dem Rechner, der das Keyboard abfragt. Die aktuelle Version des Steuerrechner-Programms benötigt etwa 4K Byte Programmspeicher und etwa 2K Byte für Variablen. Ich beabsichtige, dafür später einen Z80-Einplatinenrechner zu benutzen. Auch wäre es sinnvoll, eine sehr schnelle parallele Schnittstelle anstelle der recht langsamen seriellen Schnittstelle zwischen Steuerrechner und Keyboard-Rechner zu verwenden.

5.3. Software

Die Software des Steuerrechners dient dazu, die verschiedenen Steuerwerte für den Signalprozessor zu berechnen. Das Programm wird dabei pro Sekunde z.Z. 150 mal durchlaufen. Als Zeitbasis für diese konstante Rate dient der im TMS programmierte und über die Interface-Platine lesbare Zähler.
Während eines Durchlaufes wird zunächst solange gewartet, bis der TMS-Zähler einen bestimmten Wert erreicht hat. Als nächstes werden die neuen Werte für die Parameter des Tongenerators berechnet. Ich wollte mich nicht auf eine feste Kombination der einzelnen, die Steuerparameter bestimmenden, Module beschränken. Daher habe ich in einer Tabelle, die pro Durchlauf einmal abgearbeitet wird, zu jedem verwendeten Modul angegeben, wo die benötigten Eingangswerte stehen und wo die Ausgangswerte abgespeichert werden sollen. Für diese Zwischenspeicherung habe ich ein Speicherfeld mit 256 Words vorgesehen. Alle Zahlen werden dabei im 16-Bit-ZweierKomplement-Format gespeichert. Die Frequenzinformationen für die Tongeneratoren werden dabei logarithmisch mit einer Auflösung von 4096 Werten je Oktave gespeichert. 0 entspricht dabei dem eingestrichenen C. Dieses hat, wie bei analogen Synthesizern, die meist mit 1 V / Oktave arbeiten, zum einen den Vorteil, daß man zum Ändern der Tonlage nur einen bestimmten Offset zum Frequenzwert dazu addieren muß. Zum anderen hat man in allen Frequenzbereichen die gleiche relative Auflösung.

Mit meinen Steuerrechner-Programm kann ich momentan folgende Module simulieren:

KON lädt eine Konstante in ein Element des Speicherfeldes.
IN kopiert den Inhalt eines Elementes des Eingabefelds in ein Element des Speicherfeldes.
OUT kopiert den Inhalt eines Elementes des Speicherfeldes in ein Element des Ausgabefeldes.
LFO simuliert einen LFO. Ein Element des Speicherfeldes gibt dabei die Frequenz an. Der aktuelle Wert des Ausgangs wird wird wieder in ein Element des Speicherfeldes geschrieben. Es sind die Kurvenformen Sinus, Dreieck, Sägezahn aufwärts, Sägezahn abwärts und Rechteck möglich.
ENV simuliert einen Hüllkurvengenerator. Ein Element des Speicherfeldes gibt dabei an, ob eine Taste gedrückt ist. Der Ausgangswert wird auf ein Element des Speicherfeldes geschrieben. Seine Hüllkurve wird durch 4 Anstiegs- bzw. Abfallraten und den 4 dazugehörigen Amplituden definiert.
ADD addiert die Inhalte zweier Elemente des Speicherfeldes und speichert die Summe wieder in ein Element des Speicherfeldes.
ADDK wie ADD, nur daß eine Konstante addiert wird.
MPY multipliziert die Inhalte zweier Elemente des Speicherfeldes und speichert die Summe wieder in ein Element des Speicherfeldes.
MPYK wie MPY, nur daß mit einer Konstante multipliziert wird.

Es ist im Augenblick möglich, maximal etwa 15 Module zu verwenden. Die 16*16-Bit-Multiplikation wird dabei vom Signalprozessor ausgeführt (siehe 4.3.).

Am Ende jedes Durchlaufes werden die neu berechneten Parameter im Ausgabefeld über die Interface-Platine dem Signalprozessor übergeben. Dabei werden die Frequenzinformationen, die im Steuerrechner logarithmisch gespeichert werden, anhand einer Tabelle in den entsprechenden Frequenzwert für das Tongeneratorprogramm umgerechnet. Zusätzlich wird abgefragt, ob an der seriellen Schnittstelle ein neuer Wert angekommen ist, also eine Taste auf dem Keyboard gedrückt oder losgelassen wurde. Ist dieses der Fall, wird dieser Wert umcodiert und im Eingabefeld gespeichert. Dort steht dann zum einen, ob eine Taste gedrückt ist und zum anderen welche Tonhöhe die gedrückte Taste hat (im 4096 Werte / Oktave Format).
Da man die Tabellen, die die Verknüpfung der Module angeben, momentan noch von Hand eingeben muß, plane ich, hierfür eine Art Assembler zu schreiben. Diesem müßte man dann die Verknüpfungsinformationen in einer mnemonischen Schreibweise in einem Textfile vorgeben.

6. Ansteuerung des Steuerrechners

Da der Steuerrechner nun auch noch auf Informationen angewiesen ist, die sich während des Spiels ändern, aber nicht über genug freie Rechenzeit verfügt, um diese selber zu ermitteln, wird ihm diese Aufgabe von einem anderen Rechner abgenommen. Die Hauptaufgabe dieses Rechners ist es, das Keyboard abzufragen, seine Daten zu dekodieren und über eine Schnittstelle dem Steuerrechner zu übergeben. Im Augenblick verwende ich dafür einen weiteren Mikrocomputer mit einer 8085 CPU / 3 MHz.

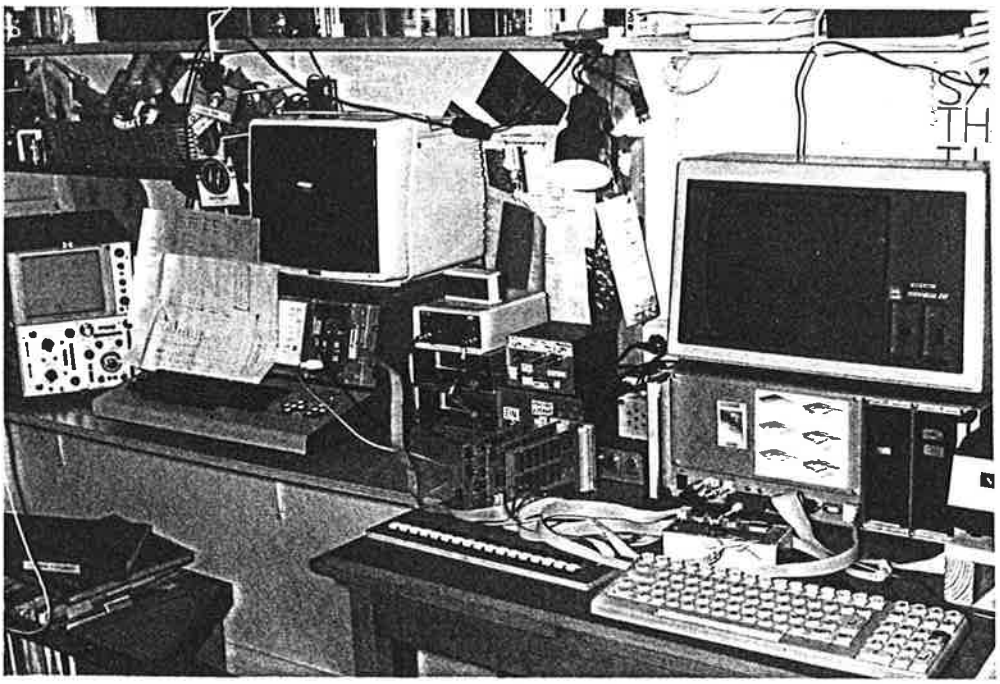
6.1. Keyboard

Für Testzwecke habe ich mir ein kleines Keyboard aus 28 "Digitast"-Tastern gebaut. Ich habe sie mit je einer Diode pro Taste in einer 8*4-Matrix verdrahtet. Diese habe ich über ein 8 Bit breites, paralleles Ein- und Ausgabeinterface an den Keyboardrechner angeschlossen. Dieser testet nun die Matrix ab und ordnet dann die gedrückte(n) Taste(n) einem oder mehreren Ausgängen zu, je nachdem ob monophon oder polyphon gespielt werden soll. Beim polyphonen Spiel habe ich dafür gesorgt, das jeder Ton maximal lang ausklingen kann. Die so ausgewerteten Tastendruckinformationen werden dann über eine serielle Schnittstelle dem Steuerrechner übergeben. Dieser simuliert für jeden der "Ausgänge" des Keyboards einen eigenen Tonerzeugungsblock.

6.2. Sequencer

Anstelle eines Keyboards könnten die Tasten-Informationen auch von einem Sequencer-Programm kommen. Diesem müßte man dann vorher ein Melodie oder Tonfolge in geeigneter Form vorgeben, und er würde diese dann in der gleichen Weise, als wenn sie auf dem Keyboard gespielt würden, an den Steuerrechner weitergeben.
Bei Verwendung eines schnelleren parallelen Interfaces könnte man auch die Stellung mehrerer Regler an den Steuerrechner weitergeben, mit denen man dann die Lautstärke oder Modulationsstärke oder Ähnliches verändern könnte.

Der digitale Synthesizer in seinem momentanen Ausbaustadium



7. Erweiterungsmöglichkeiten

Mit einer RAM-Platine für den Signalverarbeitungsrechner könnte man die Fähigkeiten des digitalen Synthesizers wesentlich erhöhen. Einige mögliche Effekte möchte ich im folgenden beschreiben.

7.1. Vorversuche für mögliche Effekte

Meine ersten Versuche mit digitaler Speicherung und Veränderung von Klängen und Geräuschen habe ich auf einem Rechner mit 8085

CPU, 3 MHz Takt und 64K Byte RAM durchgeführt. Ich habe mir dazu mit den 8 Bit Wandler-ICs ZN 427 und ZN 428 eine A/D- und D/A-Wandlerkarte aufgebaut, die ich über zwei I/O-Port's ansprechen kann. Ich kann so Geräusche mit einer Abtastrate von bis zu 50 KHz digitalisieren und bis zu einer Sekunde lange Geräusche speichern. Mit einem Programmpaket kann ich den Speicherinhalt in der Art eines Speicheroszilloskopes auf dem Bildschirm darstellen und mit einem Cursor auch Ausschnitte markieren. Diese Ausschnitte kann ich mit veränderlicher Abtastrate wieder ausgeben. Dieses ist schon eine Art Sampler.
Auch habe ich die Möglichkeit, mit einer Art FIFO für die gewandelten Werte eine Verzögerung von bis zu einer Sekunde zu erreichen. Wenn man nun noch das verzögerte Signal wieder in das Eingangssignal rückkoppelt, erhält man eine Art Nachhall bzw. Echo. Ein solches Effektgerät, "digital delay" genannt, findet in moderner Musik sehr oft Verwendung.

7.2. Tonerzeugung und Sampler

Ein großes Problem ist, daß ich zur Zeit noch keine oberwellenreiche Töne erzeugen kann. Eine Möglichkeit wäre, oberwellenreiche Schwingungen in die RAM-Platine zu schreiben und dann in der gleichen Weise wie jetzt die Sinusschwingungen auszulesen. Dabei muß man aber beachten, daß man das Abtasttheorem nicht verletzt (siehe 4.3.). Man könnte aber auch den natürlichen Klang eines Instrumentes mit dem A/D-Wandler aufnehmen und in der RAM-Platine ablegen. So könnte man realistische und den Orginalistrumenten sehr ähnliche Klänge erzeugen. Aber auch Geräusche oder Stimmen könnte man so aufnehmen und in der ggf. mit einer anderen Tonhöhe wieder ausgeben. Geräte mit diesen Fähigkeiten bezeichnet man normalerweise als Sampler.

7.3. Effekte

Zu den wichtigsten Effekten, die man mit einer RAM-Platine erzeugen könnte, gehören die Verzögerungs- und Echo-Effekte eines "digital delay" die ich in 7.1. kurz beschrieben habe. Man könnte aber auch solche Geräte wie einen Ringmodulator, Verzerrer, sowie weitere ähnliche Geräte simulieren.

8. Literatur und Hilfen

Benutzte Literatur:

1) TMS 32010 User's Guide, Texas Instruments, 1983

2) Precision Digital Sine-Wave Generation with the TMS 32010, Application Report, Texas Instruments, 1984

3) Datenblätter bzw. -bücher der übrigen verwendeten IC's

4) Beschreibung der benutzten Microcomputersysteme

Benutzte Hilfen:

Es wurden keine weitern Hilfen benutzt.

Danken möchte ich an dieser Stelle besonders der Firma Texas Instruments Deutschland GmbH, die mir einen Signalprozessors TMS 32010 kostenlos zur Verfügung stellte.