

Wir basteln uns ein Interface

Mit einem gut ausgestatteten Computer kann man eine Menge wichtiger Dinge tun: Man kann z.B. seinem Konterfei in der viele MB Arbeitsspeicher fressenden Bildbearbeitung einen Satz neuer Ohren verpassen, das Ganze in Photoqualität ausdrucken und mit dem Ergebnis Freunde und Bekannte beeindrucken – oder man kann sich in stundenlanger Kleinarbeit atemraubenden Lötdämpfen und aggressiven Ätzmitteln aussetzen, um mit Computerhilfe einige LED's zum Leuchten zu bringen. Hohn und Spott werden der Lohn für diese bahnbrechende Elektronikentwicklung sein - doch genau damit wollen wir uns hier befassen.

Interfaceschaltungen gab und gibt es für den Atari zugegebenermaßen schon eine erkleckliche Anzahl. Störend dabei ist aber in der Regel die Komplexität der Schaltungen, die dem Hobbybastler ein Verstehen der elektronischen Zusammenhänge unmöglich macht. Schaltungstechnische Spitzfindigkeiten, Spezialbauelemente und ellenlange Logikgleichungen für Gal's und Pal's tun ihr übriges, um den euphorischen Elektronikneuling in einen von Selbstzweifeln geplagten Nur-Konsumenten zu verwandeln. Mit einigen kleinen Basteleien (Schaltungsentwicklung wäre wohl zu dick aufgetragen) wollen wir diesem Trend entgegenwirken und dem engagierten Bastler das eine oder andere Erfolgserlebnis vermitteln.

Da stellt sich natürlich als erstes die Frage, was im Rahmen dieser Serie denn so alles gebaut werden soll: Prinzipiell alles, was mit Standardbauelementen realisierbar ist und keine fundierten Kenntnisse voraussetzt. Anregungen und erprobte Schaltungsvorschläge aus der Leserschaft sind herzlich willkommen, ebenso konstruktive Kritik und Verbesserungsvorschläge.

Heute geht es zunächst erst einmal darum, welche Ausrüstung man haben sollte und welche Bauelemente zu beschaffen sind, damit zukünftige Basteleien auch durchgeführt werden können. In der nächsten Ausgabe wollen wir dann tatsächlich einige LED's mit dem Rechner ansteuern und danach "Daten einlesen".

Das Erzeugen und Messen von Spannungen steht etwas weiter hinten auf dem Programm. Wie es dann weitergeht, hängt auch ganz wesentlich von den Wünschen und Anregungen aus Bastlerkreisen ab, fangen wir also erst einmal vorne an.

Anfangsüberlegung

Der auserwählte Rechner dürfte wohl klar sein: Der allseits beliebte Atari. Ob 520er oder TT, ob mit oder ohne Festplatte, viel oder wenig Ram – spielt alles keine Rolle, nur laufen muss er. Ein monochromer bzw. hochauflösender Monitor für die Programmiergeschichten wäre wohl nicht schlecht, ist aber nicht Bedingung. Weiteres Nachdenken erfordert die Entscheidung, welche Schnittstelle denn nun für unsere Zwecke missbraucht werden soll. Die parallele Schnittstelle scheidet aus, da sich sonst kein Drucker betreiben läßt. Die serielle Schnittstelle bietet uns, wie der Name schon vermuten lässt, lediglich einen seriellen Datenstrom, der über Schieberegister erst in eine für uns genehme Form gebracht werden müsste – viel zu aufwendig. Gleiches gilt für die Midi-Schnittstelle. Was bleibt also: genau, der Romport. Hier finden wir reichlich Daten- und Adressleitungen, die zudem

auch noch sehr einfach zu programmieren sind. Der Pferdefuß sei nicht verschwiegen: Der 40polige Slot, der an jedem Atari ab ST-Baureihe zu finden ist, hält für uns einen gar unüblichen Kontaktabstand von 2mm bereit. Ein entsprechender Romportstecker muss also selbst gefertigt werden (soviel zu "Standardbauteilen").

Unsere Schnittstelle

Der Romport, auch Modulschacht genannt, dient eigentlich der Aufnahme eines Programm- oder Diagnosemoduls, das, je nach Programmierung, sofort nach dem Einschalten abgearbeitet wird oder wie ein Massespeicher vom Desktop aus angesprochen werden kann. Wie der Name ROM (Read-Only-Memory) schon sagt, kann auf diese Schnittstelle nur lesend zugegriffen werden. Ein Schreibversuch wird vom Atari mit Bomben belohnt. Trotzdem ist es möglich über die Adressleitungen Daten "nach draußen" zu schaffen, aber mehr dazu später. Der Romport verfügt über 40 Kontakte, deren Signale sich in vier Gruppen unterteilen lassen:

- 1. Versorgungsleitungen 3 x Masse und 2 x +5 Volt
- 2. Adressleitungen 15 Stück (A1 bis A14)
- 3. Datenleitungen 16 Stück (D0 bis D15)
- 4. Steuerleitungen /ROM3, /ROM4, /UDS und /LDS

Die Signale am Romport:

1	3	5	7	9	11	13	15	17	19	21	23	25	27	29	31	33	35	37	39
2	4	6	8	10	12	14	16	18	20	22	24	26	28	30	32	34	36	38	40
1		+5V																	
3		D 14																	
5		D 12																	
7		D 10																	
9		D 8																	
11		D 6																	
13		D 4																	
15		D 2																	
17		D 0																	
19		A 13																	
21		A 8																	
23		A 7																	
25		A 6																	
27		A 5																	
29		A 11																	
31		/ROM3																	
33		/ROM4																	
35		/UDS																	
37		/LDS																	
2		+5V																	
4		D 15																	
6		D 13																	
8		D 11																	
10		D 9																	
12		D 7																	
14		D 5																	
16		D 3																	
18		D 1																	
20		A 15																	
22		A 14																	
24		A 9																	
26		A 10																	
28		A 12																	
30		A 4																	
32		A 3																	
34		A 2																	
36		A 1																	
38-40		GND																	

Auf den ersten Blick wirkt die Signalführung recht ungeordnet, auf den zweiten übrigens auch, aber damit werden wir leben (müssen). Eine nähere Betrachtung der einzelnen Leitungen offenbart schnell die möglichen Risiken

bei der Benutzung des Romports: Unkritisch ist der Masseanschluss (Pins 38 – 40). Hier liegen 0 Volt an, dieser Anschluss bildet damit das Bezugspotential für unsere externen Basteleien. Die für die Versorgung von Rom-Modulen vorgesehene 5 Volt-Spannung (Pins 1 und 2) kann für umfangreichere externe Schaltungen oft nicht den nötigen Strom liefern. Je nach Rechnertyp und bereits eingebauten Hardwarezusätzen liegt die maximal mögliche Stromentnahme so zwischen 0 und 1 Ampere. Wer seinen Mega 1 mit PAK, Fastram- und Grafikkarte ausgestattet hat, könnte wohl eher noch ein wenig Strom gebrauchen, übrig ist da mit Sicherheit nichts. In so einem Fall muss die externe Hardware eine eigene Stromversorgung erhalten.

Die Daten- und Adressleitungen, sowie die Signale /UDS und /LDS stammen direkt vom Prozessor. Ein Kurzschluss auf diesen Leitungen führt aber keineswegs zur sofortigen Zerstörung, der Rechner fährt dann eben nicht hoch. Der Zweck der Daten- und Adressleitungen dürfte klar sein, doch /UDS und /LDS bedürfen der Erklärung: Da der 68000er Prozessor einen 16 Bit breiten Datenbus hat, braucht er eigentlich nicht jedes Byte (= 8 Bit) einzeln anzusprechen, denn es werden ja immer 2 Byte (= 1 Word) auf einmal bearbeitet. Rein theoretisch reicht es also, wenn man nur jedes 2. Byte adressiert. Damit wäre dann die Adressleitung A0 überflüssig. Einige im und am Atari verwendete Komponenten verfügen jedoch nur über die Möglichkeit einer byteweisen Verarbeitung von Daten, dazu gehören zum Beispiel Floppy- und DMA-Controller. Um mit diesen Komponenten umgehen zu können, ist es notwendig, auch einzelne Bytes anzusprechen. Hier liegt nun der tiefere Sinn dieser beiden Signale. /UDS bedeutet "Upper Data Strobe" und wird aktiv, wenn auf eine gerade Adresse zugegriffen wird, /LDS bedeutet "Lower Data Strobe" und wird beim Zugriff auf ungerade Adressen aktiv. Der "/" (slash) vor dem Signalnamen sagt uns, dass diese Signale "Lo-aktiv" sind. Das bedeutet, dass sie dann aktiv sind, wenn ihre Spannung 0.8 Volt unterschreitet, also gegen Masse geht. Bleiben noch /ROM3 und /ROM4. Diese Signale werden im ST(E) vom Glue erzeugt, bei TT und Falcon von Spezialchips. Sie sind ebenfalls Lo-aktiv und dienen der Auswahl eines von zwei je 64KB umfassenden Adressräumen, die am Romport zur Verfügung stehen.

Der Adressraum des Romports

Viele haben sicherlich schon davon gehört, dass sich am Romport nur maximal 128KB verwalten lassen. Das mag für umfangreiche Programme sicherlich recht mager sein, für uns bedeutet das aber, dass wir über 130.000 Schaltmöglichkeiten zur Verfügung haben! Damit sollten wir in jedem Falle auskommen und werden auch entsprechend verschwenderisch damit umgehen.

Wie kommt man mit den zur Verfügung stehenden Leitungen aber auf gerade 128KB? Dazu betrachten wir die anliegenden Signale 'mal ein bisschen genauer: Wir haben 16 Datenleitungen, die bei der Adressierung keine Rolle spielen, weil eben nur Daten darüber laufen. Weiterhin haben wir 15 Adressleitungen, nämlich A1 bis A14. Schon interessanter, denn damit können wir 2 hoch 15, also 32.768 Adressen ansprechen. Diese 32.768 Adressen können entweder gerade sein, dann ist /UDS aktiv, oder ungerade, dann ist /LDS aktiv. Somit hätten wir schon 65.536 ansprechbare Adressen. Jetzt

kommen /ROM3 und /ROM4 ins Spiel und verdoppeln unsere Möglichkeiten noch einmal, macht 131.072 Adressen. Das sind auf's Bit genau 128KB. Der Adressraum des Romports hat, wie eine Wurst, auch einen Anfang und ein Ende, aber dazu mehr wenn's interessant wird.

Die Grundausstattung

Ein Lötkolben allein genügt nicht, soviel steht fest. Wie bei allem im Leben gibt es aber auch hier Sachen, die zwingend erforderlich sind, und andere, die die Arbeit erleichtern. Ich mache an dieser Stelle dabei keinen Unterschied, weil ich in frühen Tagen mit unzulänglichen Arbeitsmitteln meine liebe Not hatte und deshalb vorbelastet bin. Außerdem möchte ich niemandem die Möglichkeit vorenthalten, seine masochistische Ader auszuleben.

1. Der Lötkolben

Man hört und liest immer wieder von empfohlenen Lötkolben mit "geringer Leistung" oder "15 Watt" oder dergleichen. Das wollen wir gleich einmal als Unsinn zu den Akten legen. Ein Lötkolben kann gar nicht genug Leistung haben. Zur Erläuterung: Die Temperatur an der Lötspitze beträgt üblicherweise zwischen 350 und 370 Grad Celsius. Bei einer Lötung sinkt die Temperatur naturgemäß ab, weil die Lötspitze an das kalte Lötgut gehalten wird. Ein Lötkolben mit geringer Leistung wird die Lötspitzentemperatur nicht so schnell wieder auf das Soll bringen können, wie einer mit hoher Leistung. Der Effekt ist eine längere thermische Belastung des Lötgutes und ein weniger flüssiges Arbeiten. Lötkolben, die ohne Trafo direkt am Stromnetz betrieben werden, sind für Elektronikarbeiten nur bedingt zu gebrauchen. Zum einen gibt es keine vernünftige Temperaturregelung, weiterhin sind die Lötspitzen zu klobig und benetzen schlecht, die Leistung ist zu niedrig (in der Regel 30 Watt), das Kabel ist nicht flexibel genug und man hat weder Ablage noch Reinigungsschwamm. Da ist es schon besser in den sauren Apfel zu beißen und sich eine Lötstation zuzulegen. Die Marke ist dabei egal, aber so um die 100DM muß man als Ausgabe schon wenigstens einkalkulieren.

2. Lot und Flussmittel

Richtig gelesen, das Lötzinn heiß in Wirklichkeit Lot. Es gibt eine Vielzahl von Loten, die sich in ganz bestimmten Eigenschaften voneinander unterscheiden. Dabei spielen Kriterien wie Schmelztemperatur, Leitfähigkeit und natürlich der Preis eine große Rolle. In der Elektronik hat sich bei Lötungen auf Kupferleiterplatten eine ganz bestimmte Sorte durchgesetzt: L-Sn60PbCu2 (Spitzname: Elektroniklot). Das "L" steht dabei für "Lot", der Rest dieser eigenartigen Bezeichnung hat natürlich ebenfalls einen tieferen Sinn, das ist nämlich die Zusammensetzung unseres Lotes. Wer in der Chemiestunde aufgepasst hat, kommt schnell dahinter: Sn ist das chemische Zeichen für Zinn, Pb für Blei und Cu für Kupfer. Wir haben in unserem Lot also 60% Zinn (Sn60) und 2% Kupfer (Cu2), der Rest ist Blei. Warum ausgerechnet diese Zusammensetzung? Ganz einfach: In dieser Zusammensetzung ist unser Lot "eutektisch", das heißt, es geht bei Erwärmung über seinen Schmelzpunkt von 183 Grad hinaus vom festen direkt in den flüssigen Zustand über ohne erst zu einer Art "Brei" zu werden. Diese Eigenschaft hat beim Löten echte Vorteile. Der Kupferanteil sorgt dafür, daß sich

bei Lötungen die Kupferleitbahn nicht so schnell auflöst. Flussmittel gibt es mindestens so viele wie Lote. Aber auch hier haben sich in der Elektronik ganz bestimmte Typen durchgesetzt. Normalerweise findet sich das Flussmittel als Flussmittelseele im Lötdraht. Die Bezeichnung sollte aber auf der Spule angegeben sein und in etwa F-SW26 lauten. Auch hinter dieser Bezeichnung verbergen sich wichtige Informationen: Das "F" steht für Flussmittel (schau an), "SW" bedeutet "Schwermetall" und "Weichlöten", die angehängte Zahl gibt Aufschluss über die Art der Rückstände. Werte von 26 und größer sagen aus, dass die Rückstände nach dem Erkalten nicht mehr aggressiv sind und deshalb nicht entfernt werden müssen. Finger weg von Lötfett (z.B. F-SW21) oder säurehaltigen Flussmitteln (z. B. F-SW11)! Die richtige Kombination aus Lot und Flussmittel für unsere Zwecke ist Lötendraht mit Flußmittelseele, Durchmesser 1mm. Auf der Spule steht also: L-Sn60PbCu2 und F-SW26 (F-SW32 ist auch ok).

3. Entlötlitze, Entlötsaugpumpe

Beides wichtige Hilfsmittel zur Entfernung von Lot. Die häufig vertretene Ansicht Entlötlitze sei das einzig Wahre, kann ich nicht teilen. Nach meinen Erfahrungen ist beides zwingend erforderlich.

4. Seitenschneider

Genau genommen heißt das Ding, das wir brauchen "Feindrahtschneider" und ist zum Durchtrennen von Kupferdrähten bis 1mm Durchmesser gedacht. Die billigen Dinger reichen vollkommen und kosten so 10 DM.

5. Flachzange

Breite ca. 3 – 5mm, wenn man 'mal was umbiegen will.

6. Kram

Das sind wichtige Utensilien, die sich mit der Zeit ansammeln. Dazu gehören zum Beispiel Teppichmesser, Schraubendreher mit 1,5mm-Klinge, Säge, Abisolierzange, Zugang zu einem Schraubstock, Platinenhalter oder sowas, Malerpinsel (so 1"), Spiritus zur Entfernung von Flußmittel und 'ne Rolle Klopapier oder dergleichen.

7. Der Arbeitsplatz

Der Wohnzimmer- oder Küchentisch ist nicht das Gelbe vom Ei, aber es hat sicherlich nicht jeder die Möglichkeit sich eine Bastelecke oder sowas einzurichten. Hell soll's sein und gut belüftet, der Lötdämpfe wegen. Der Tisch sollte möglichst nicht wackeln und eine vernünftige Arbeitshöhe haben. Tapeziertische sind, trotz ihrer verführerisch großen Arbeitsfläche ungeeignet.

Bauelemente und so

Was wir für die erste Zeit brauchen, ist verdächtig wenig. Wahrscheinlich fällt mir der Rest nach Drucklegung ein, aber ein paar Kleinigkeiten lassen sich sicher auch noch besorgen wenn's ernst wird.

1. Romportstecker

Sicherlich der härteste Brocken, da als Bauelement nicht käuflich zu erwerben. Wer über die Möglichkeit verfügt Platinen zu ätzen, kann für sich (und vielleicht auch für

andere Interessierte) so ein Ding selbst herstellen: Man nehme eine beidseitig kupferkaschierte Leiterplatte und säge sie sich so zurecht, dass sie in den Romport passt, also auf 41,5mm Breite. Nach einer Probesteckung erkennt man auf der Kupferfläche Kratzspuren der Romportkontakte. Diese Kratzspuren decke man mit Anreibesymbolen oder Ätzresist ab und werfe das Ganze ins Ätzbad – fertig. Wer sich von diesen Mühen freikaufen möchte, besorge sich irgendeine Platine für den Romport (Scannerinterface, Epromkarte o.ä.) und säge den für uns interessanten Stecker möglichst großzügig ab.

2. Flachbandkabel

40polig soll es sein und ca. 10cm lang. Am besten nimmt man ein IDE-Kabel für den Anschluss von Festplatten und schneidet sich ein Stück ab, da ist dann auch gleich der richtige Stecker dran. Das Ende ohne Stecker wird später am Romportstecker angelötet, und zwar im Wechsel eine Ader oben, die nächste unten. Dabei ist die markierte Ader Pin 1 und gehört dementsprechend an Pin 1 unseres Romportsteckers.

3. Platinen

Lotpunkttraster soll es sein, die Verdrahtung ist einfacher als bei Platinen mit Lötstreifenraster. Eine Europakarte (160mm x 100mm) reicht für den Anfang.

4. Bauelemente

Da brauchen wir (zunächst) nicht viel:

2 x Pfostenleiste 2reihig, 2 x 32 Pins 4 x IC-Sockel 20pol 2 x IC 74LS374 2 x IC 74LS244 1 x Elko 100uF 4 x Kondensator 100nF (ca.) 8 x LED, 3mm, Farbe egal 8 x Widerstand 330R (ca.)

Dazu natürlich Strippe, möglichst dünn. Klingeldraht ist ungeeignet, viel zu dick und zu starr, dann besser einzelne Litzen aus dem Flachbandkabel verwenden.

Die Programmierung

Für heute war's das im Wesentlichen. Bleibt nur noch die Klärung der Programmierfrage. Programmierkenntnisse sind für diese Geschichte eigentlich nicht erforderlich, aber eine Programmiersprache sollte man schon besitzen. Die Beispiele hier werden in GFA-Basic sein, weil's so schön einfach ist, und sie sich problemlos auf jede andere Hochsprache übertragen lassen.

So, hier war der erste Teil zu Ende.

Ulrich Skulimma

Zweiter Teil

Tja, nun ist es endlich soweit. Nach Durchführung der erforderlichen Lötarbeiten sollen allen geneigten Mitbastlern einige rechner- gesteuerte Lichter aufgehen. Doch zuvor einige klärende Worte zu den Bauelementen, die bei dieser ersten Schaltung Verwendung finden. Fangen wir zur Abwechslung einmal hinten an und nehmen unsere "Anzeige" unter die Lupe.

Die LED

LED bedeutet zu deutsch "Licht Emittierende Diode". Es handelt sich also um eine Diode, die Licht aussenden kann.

Das tut sie beileibe nicht grundsätzlich, sondern nur dann, wenn der Strom in einer bestimmten Richtung durch sie hindurch- fließt. Schließt man eine LED falsch herum an einen Stromkreis an, bleibt nicht nur das gewünschte Leuchten aus, unter Umständen wird die LED dabei auch zerstört.

Deshalb ist es sehr erfreulich, dass man den LED's ihre Polarität auch ansehen kann. Dem aufmerksamen Betrachter fallen sofort zwei Anschlüsse auf, die den Diodenkörper verlassen. Sinnigerweise ist der eine Anschluss der Pluspol, der andere der Minuspol. Der versierte Elektroniker nennt den Pluspol Anode und den Minuspol Kathode. Anode und Kathode lassen sich durch die Länge der Anschlusspins voneinander unterscheiden. Die Anode, also der Pluspol, ist der längere von beiden.

Wer nun seine LED's irgendwo ausgelötet hat, wird mit diesem Unterscheidungsmerkmal wenig anfangen können, da beide Pins wahrscheinlich auf gleiche Länge gekürzt wurden. Hier hilft ein Blick auf den Diodenkörper weiter: Man erkennt im Kunststoffkörper der LED zwei Pole, von denen einer deutlich größer ist, als der andere. Der größere der beiden Pole ist die Kathode, also der Minuspol, der kleinere die Anode.

Die Anschlussrichtung steht somit fest, aber damit ist das Thema LED noch nicht erledigt. Wird eine LED von einem zu großen Strom durchflossen, so wird sie zwar besonders hell, aber nicht besonders lange leuchten. Aus diesem Grunde ist es notwendig den Strom zu begrenzen, der durch die LED fließt.

Handelsübliche LED's sind normalerweise für einen Strom von maximal 20 mA (= 0,02 A) ausgelegt. Um diesen Strom von 20 Milliampere nicht zu überschreiten, muss der LED ein Widerstand vorgeschaltet werden, der den Strom auf den zulässigen Wert begrenzt.

In unserem Fall wird das ansteuernde IC mit +5 Volt gespeist, kann demnach für die LED-Ansteuerung höchstens eine Spannung von 5 Volt bereitstellen. Je nach Farbe benötigt (oder "verbraucht") die LED selbst eine Spannung von ca. 1,5 Volt bis 2,2 Volt, damit sie überhaupt etwas tut (das nennt sich Spannungsabfall, muss aber nicht entsorgt werden). Somit bleiben uns bestenfalls 3,5 Volt, die für die LED-Leuchtkraft zu gebrauchen sind.

Nehmen wir jetzt das allseits beliebte Ohmsche Gesetz zur Hand, können wir den nötigen Vorwiderstand errechnen (siehe Bild rechts oben):

Der nächste geläufige Widerstand von 180R ist für den LED-Betrieb ausreichend hoch.

Der Grund für den empfohlenen Widerstand von 330R ist, dass

Das Ohmsche Gesetz

$$\text{Strom (in Ampere)} = \frac{\text{Spannung (in Volt)}}{\text{Widerstand (in Ohm)}} \quad I = \frac{U}{R}$$

in unserem Fall brauchen wir aber den Widerstand, also:

$$R = \frac{U}{I} \quad \longrightarrow \quad R = \frac{3,5 \text{ Volt}}{0,02 \text{ Ampere}} = 175R$$

unsere LED's mit einem geringeren Strom zwar etwas weniger stark leuchten, das treibende IC dafür aber auch nicht so stark belasten. Kurz gesagt: Der Stromverbrauch unserer Schaltung wird da- durch gesenkt.

Der Widerstand

Der Sinn und Zweck der eingesetzten Widerstände ist mit den vorangegangenen Ausführungen zur LED eigentlich hinreichend erklärt.

Zu Widerständen allgemein kann man ganze Romane verfassen, so vielseitig einsetzbar ist dieses unscheinbare Bauelement. Für uns aber soll für's erste genügen, dass man damit einen Strom begrenzen kann.

Die Kondensatoren

Die Einsatzmöglichkeiten sind gewaltig, wir benutzen sie aber erst einmal nur zur Glättung und Stabilisierung der Versorgungsspannung für unsere Schaltung.

Betrachten wir zunächst den Elko, genauer: Elektrolytkondensator. Wer genau hinguckt hat, hat vielleicht bemerkt, dass dieser Kondensatortyp, genau wie eine LED, einen Plus- und einen Minuspol hat. Bei der stehenden Ausführung ist der Anschluss für den Pluspol etwas länger, als der für den Minuspol. Der Minuspol ist auf dem Kondensator selbst zusätzlich noch irgendwie kenntlich gemacht, häufig durch einige Minuszeichen an der entsprechenden Seite. Die liegende Ausführung hat an der Plusseite eine Einschnürung. Auf den Minuspol weist in der Regel ein breiter Pfeil. Der Elko dient uns als Strom- speicher und sorgt dafür, dass wir durch das Schalten der LED's keine Spannungseinbrüche befürchten müssen.

Der kleine Kondensator mit 100nF ist anderer baulicher Natur, auf die Polarität muss dabei nicht geachtet werden. Seine Funktion ist das Abfangen (Puffern) von Spannungsspitzen, die durch die Schaltvorgänge in unserem IC entstehen können.

Das IC

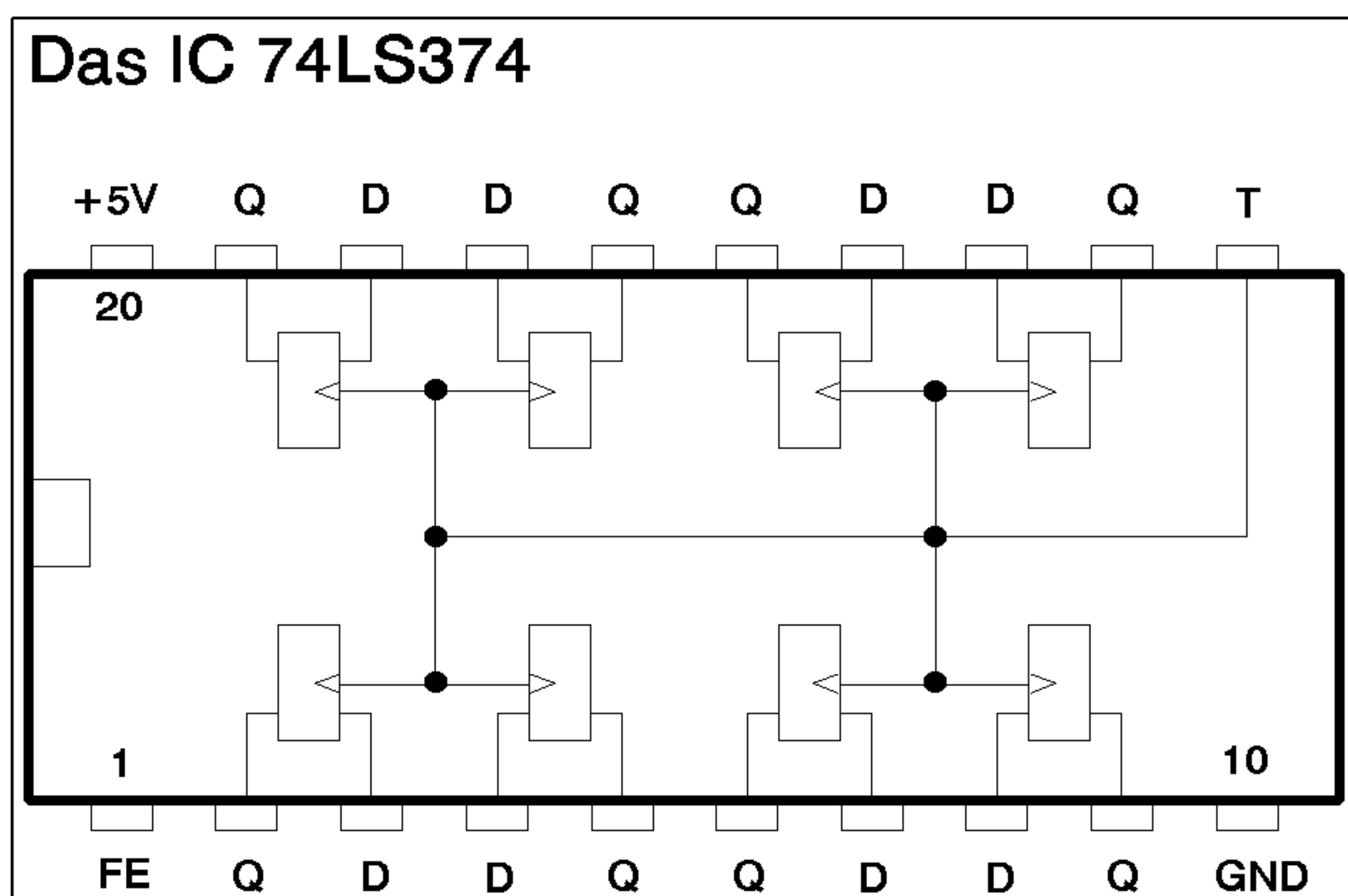
Richtig gelesen, zunächst brauchen wir nur eines, nämlich ein 74LS374.

In diesem IC sind eine ganze Reihe von Bauelementen verborgen, die uns aber nicht weiter interessieren. Wichtig für uns ist die Funktion.

Das 74LS374 enthält acht Flip-Flop's, was im Groben nichts anderes bedeutet, als dass sie, je nach äußeren Umständen, an ihren Ausgängen 'mal eine Hi-Pegel oder 'mal einen

Lo-Pegel haben. Der Atari soll uns nun dabei behilflich sein, solche "äußeren Umstände" (sprich: Bedingungen) herbeizuführen, die ein Flip-Flop dazu veranlassen, das zu tun, was wir wollen.

Dazu sehen wir uns eines von den acht Stück etwas genauer an:



Da gibt es einen Eingang, einen Ausgang und einen Takteingang, das war's schon. Der Pegel (Hi oder Lo), der an den Eingang gelegt wird, steht nur dann am Ausgang an, wenn die Taktleitung in der Zwischenzeit einen Sprung von Lo nach Hi durchführt. Anders ausgedrückt: Das Signal am Eingang wird mit der steigenden Flanke des Taktsignals am Ausgang sichtbar. Solange also die Taktleitung nicht von Lo nach Hi geht, ist es egal, welches Signal am Eingang liegt, am Ausgang steht das Signal zur Verfügung, das bei der letzten LO->Hi-Bewegung des Taktsignals am Eingang anstand. Die Flip-Flop's im 74LS374 sind also Zwischenspeicher, die genau die Daten an ihren Ausgängen bereithalten, die mit der letzten steigenden Flanke (= die Bewegung von Lo nach Hi) an ihre Eingänge gelegt waren.

Dem wachen Verstand eines jeden Bastelwütigen wird der mit FE bezeichnete Pin 1 des 74LS374 nicht entgangen sein.

Hierbei handelt es sich um den Eingang für ein sogenanntes "Freigabesignal". Wird dieser Eingang auf Hi gelegt, werden die an den Eingängen anliegenden Daten mit der steigenden Flanke des Taktsignals zwar in den Flip-Flop's gespeichert, aber die Ausgänge verhalten sich, als wären sie gar nicht da. Sie bleiben hochohmig und die gespeicherten Daten können nicht ausgelesen werden.

Diesen Zustand nennt man "Tri-State", weil er, abgesehen von Lo und Hi, der dritte mögliche Zustand dieses IC's ist.

Erst wenn der Freigabeeingang auf Lo gelegt wird, werden die in den Flip-Flop's gespeicherten Daten an den Ausgängen sichtbar.

Wozu gibt es nun diesen dritten Zustand, den Tri-State? Die Erklärung ist schnell geliefert:

Wenn ein IC seine Daten permanent auf den Leitungen liegen lässt, kann ein anderes IC diese Leitungen nicht benutzen, sie sind ja nicht frei, sondern besetzt.

Über den Freigabeeingang wird das IC dazu bewegt, die Leitungen freizugeben, sich selbst also in einen Zustand zu versetzen, der andere IC's bei der Benutzung dieser Leitungen nicht stört.

In unserem Fall spielen andere IC's noch keine Rolle, deshalb können wir den Freigabeeingang des 74LS374 erst 'mal mit Masse verbinden.

Eine weitere Frage wirft sicherlich der Takteingang auf: Woher nehmen wir den Takt? Kein Problem. Zur gesteuerten

Datenübernahme stellt uns der Atari am Romport entsprechende Leitungen zur Verfügung: /ROM3 und /ROM4. Genau genommen dienen diese Leitungen dazu bei einem Lesezugriff auf den Romport der angeschlossenen Hardware mitzuteilen, dass in dem Augenblick, wenn /ROM3 oder /ROM4 Lo sind, die anstehenden Daten eingelesen werden.

Wir verwenden als Taktleitung für unser IC das Signal /ROM3. Dieses Signal wird dann aktiv, wenn wir aus dem höheren der beiden 64KB-Blöcke des Romport etwas lesen wollen. Das Schreiben in diesen Bereich ist nicht möglich, dem hat Atari hardwaremäßig einen Riegel vorgeschoben.

Aus diesem Grunde können für die Ansteuerung einer externen Hardware am Romport auch nicht die Datenleitungen benutzt werden. Wir missbrauchen deshalb die Adressleitungen zur Datenübertragung, und zwar folgendermaßen:

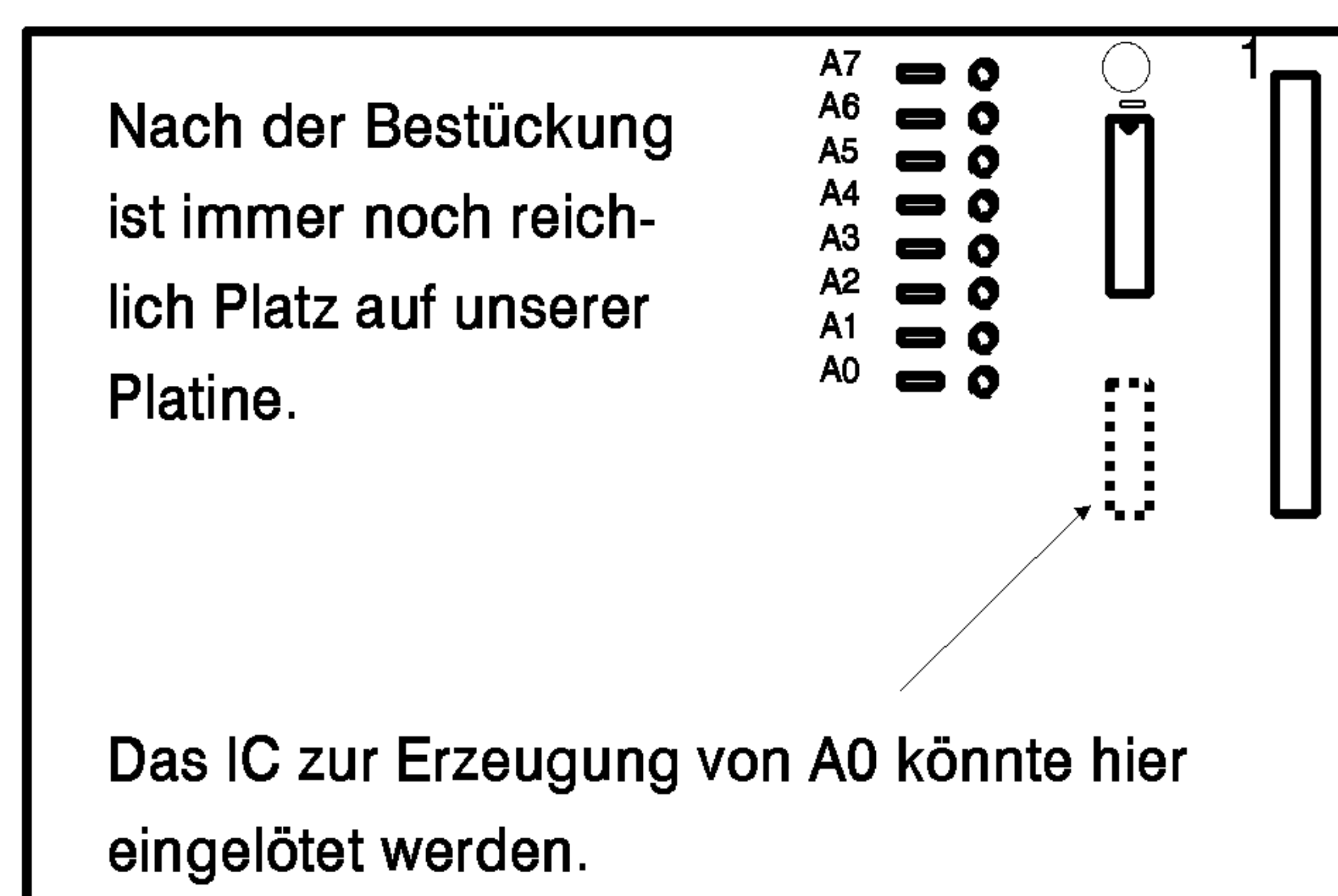
An die Eingänge unserer Flip-Flop's werden nicht die Daten-, sondern die Adressleitungen angelötet und /ROM3 dient uns als Taktleitung.

Wenn wir jetzt einen Lesezugriff auf den richtigen Speicherbereich durchführen, haben die Adressleitungen die Lo-Hi-Kombination, die unsere angelegte Adresse als Zahlenwert im dualen Zahlensystem hat. /ROM3 wird Lo, um für uns uninteressante Daten einzulesen. Wenn /ROM3 dann wieder Hi, also inaktiv wird, wird mit der steigenden Flanke unsere angelegte Adresse in den Flip-Flop's gespeichert und steht bis zum nächsten Lesezugriff zur Verfügung.

Jetzt wird's ernst

Wer keinen Romportstecker auftreiben konnte, hat vielleicht, wie ich auch, die Adern des Flachbandkabels direkt an den Romport angelötet. Wenn der Rechner noch läuft, scheint die Sache gut gegangen zu sein und der schwierigste Teil der Lötarbeit ist erledigt.

In jedem Fall sollte, auf die eine oder andere Art, der Romport über ein 10cm bis 15cm langes Flachbandkabel jetzt als 40polige Buchsenleiste zur Verfügung stehen.



Mit gebührendem Elan geht's an die Bestückung der Euro-Platine. Normalerweise werden die niedrigsten Bauelemente zuerst eingesetzt, wir machen's aber anders, dann müssen wir nicht so genau planen (soll heißen: Gar nicht) und beginnen mit der Pfostenleiste.

Die wird, entsprechend der Adernzahl des Flachbandkabels auf 40 Pins gekürzt und an der rechten oberen Ecke der Platine eingelötet. Es folgen im Abstand von ca. 1,5 cm ein 20poliger IC-Sockel mit den zwei Kondensatoren und

untereinander die acht LED's. Die Anode (=plus, langer Pin) gehört in Richtung IC.

Zwischen den LED's sollte ein Leerraster verbleiben, damit die Leuchtkörper nicht aneinanderstoßen und das Löten einfacher wird. Es macht sich gut, wenn man die LED's zunächst nur mit einem Pin anlötet, dann kann man sie noch vernünftig ausrichten (das Auge sieht mit!).

Jetzt noch einen Widerstand links neben jede LED gesetzt und schon geht's an die Verdrahtung.

Hier muss eigentlich immer nur Punkt A mit Punkt B verbunden werden, soweit kann eigentlich nichts schief gehen, wenn man seine A- und B-Punkte kennt. Hangeln wir uns diesmal von leicht in Richtung schwer, die Kabellänge nicht zu knapp bemessen.

1. Zunächst werden alle Widerstände auf der LED-abgewandten Seite miteinander und mit Pin 39 oder 40 der Pfostenleiste (Masse) verbunden.
2. Das andere Ende jedes einzelnen Widerstandes wird mit der Kathode (Minus) der nebenliegenden LED verlötet.
3. Die Kondensatoren werden zwischen Pin 1 und Pin 20 des IC-Sockels gelötet, beim Elko zeigt Plus zu Pin 20.
4. Pin 1 und Pin 10 des IC-Sockels werden miteinander und mit Pin 39 oder 40 der Pfostenleiste verbunden.
5. Pin 20 des IC-Sockels verbinden wir mit Pin 1 oder 2 der Pfostenleiste.

6. Die LED's werden nun im Geiste durchnummeriert, und zwar entsprechend der verwendeten Adressleitungen. in diesem Fall also von 0 bis 7.

Bei der Verdrahtung der Adressleitungen geht man am besten Leitung für Leitung vor, um Verwechslungen auszuschließen. Von der Anode der LED Nr.7 eine Verbindung zu einem Ausgang (Q) des 74LS374 herstellen. Der zu dem Ausgang gehörende Eingang wird mit A7 (Pin 23 der Pfostenleiste) verbunden.

Entsprechend verfährt man mit den anderen LED's und Adressleitungen.

Die verzweifelte Suche nach A0 kann getrost abgebrochen werden, dieses Signal gibt es nicht, dazu später noch eine Art Denksportaufgabe.

7. Jetzt noch /ROM3 (Pfostenleiste Pin 31) mit dem Takteingang (Pin 11) des 74LS374 verbinden – fertig.

Die Programmierung

Wie schon in der letzten Ausgabe angedeutet, liegt der Romport in einem bestimmten Adressbereich des Atari.

Dieser Bereich lässt sich recht nett in hexadezimalen Zahlen beschreiben:

Der Bereich von FA0000 bis FFFFFFFF wird von dem Signal /ROM4 bedient, /ROM3 ist für den Bereich von FB0000 bis FBFFFF zuständig. Das heißt: Sobald man auf einen dieser Bereiche lesend (!) zugreift, wird eines der /ROM-Signale aktiv.

Was es dort zu lesen gibt, ist für uns unwichtig, denn wir wollen ja nur den Zustand unserer Adressleitungen in dem IC speichern!

Da wir als Taktleitung /ROM3 verwenden, hat nur der Lesezugriff auf den Bereich FB0000 bis FBFFFF einen Effekt. Genau betrachtet ist unser Adressbereich sogar noch kleiner, denn wir verwenden nur die Adressleitungen A0 (A1) bis A7. Damit können wir höchstens 256 Zustände erzeugen.

Hexadezimal ausgedrückt sind es FF Zustände. Dementsprechend reicht unser nutzbarer Bereich nur bis FB00FF.

In GFA-Basic (und auch woanders) gibt es einen einfachen Befehl, um von einer beliebigen Adresse Daten einzulesen. Dieser Befehl heißt "PEEK", gefolgt von der gewünschten Adresse in Klammern.

Natürlich muß man auch angeben, was mit den eingelesenen Daten passieren soll. Man kann sie z.B. auf dem Bildschirm ausgeben und gleichzeitig versuchen die zweite LED zum Leuchten zu bringen. Dann lautet der Befehl:

"PRINT PEEK(&HFB0000+2)".

Auf dem Monitor erscheint als eingelesener Wert eine 255, um die wir uns keine Gedanken machen, und die zweite LED leuchtet auf. Das in dem Basicbefehl enthaltene "&H" ist ein Hinweis darauf, dass der direkt folgende Ausdruck eine hexadezimale Zahl ist.

Die auf das Pluszeichen folgende 2 ist in diesem Fall eine Dezimalzahl. Ersetzt man sie durch eine 4, sollte die dritte LED leuchten.

Mit ein paar Zahlenspielen lassen sich die LED's nun in beliebiger Weise zum Leuchten bringen. Dabei kann zu unserem Basiswert von FB0000 auch eine Variable addiert werden. Die Programmbeispiele verdeutlichen die Möglichkeiten (hoffe ich).

Das A0-Problem

Jetzt' ha'm wir den Salat, keine A0-Leitung! Nur nicht verzagen, die können wir uns selbst bauen. Wenn sie da wäre, würde A0 beim Zugriff auf eine gerade Adresse inaktiv (Lo), bei einer ungeraden Adresse Hi. Eine Betrachtung von /UDS oder /LDS lohnt in jedem Fall, um A0 zu generieren.

Noch ein Tipp: Das Signal muß etwas verzögert werden (z.B. 74LS00), damit es als A0 in dem 74LS374 gespeichert werden kann. Mehr wird nicht verraten, denn für die denkende Mehrheit der Bewohner dieses Planeten gibt es möglicherweise etwas zu holen. Funktionierende Lösungsvorschläge für dieses elektronische Rätsel haben die Chance auf einen netten Gewinn!

Ende Teil 2 und zu gewinnen gibt's auch nix mehr.

Ulrich Skulimma

Wir basteln uns ein Interface – Teil 3

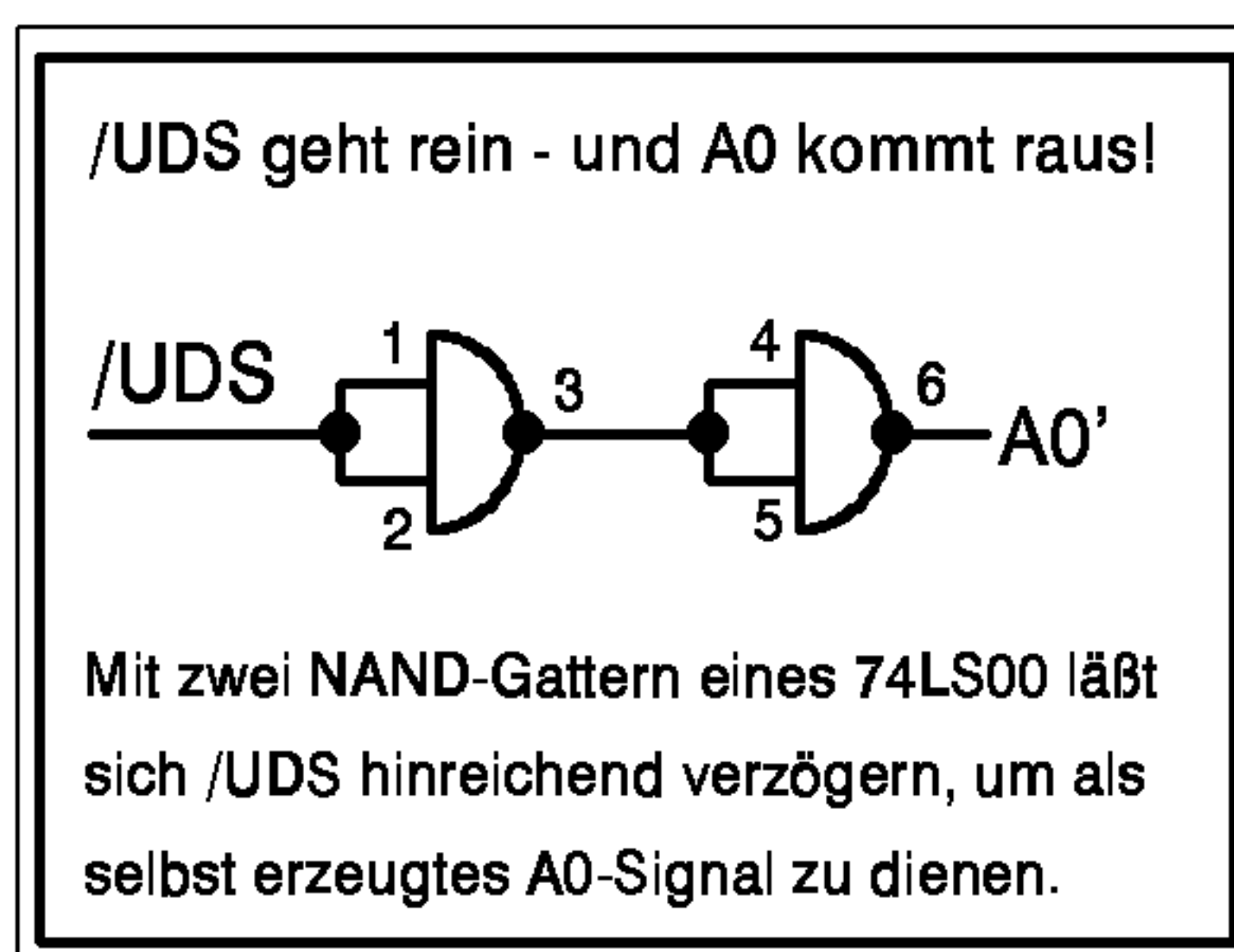
Heute wollen wir uns wieder mit der wichtigsten Sache dieses Sonnensystems befassen – unserem Interface. Das Einlesen von Daten über den Romport steht in dieser Ausgabe auf dem Programm. Doch zuvor noch die Auflösung des A0-Rätsels.

A0 selbst generiert

Tipps zur Entwicklung eines Lösungsvorschlages gab es in der letzten Ausgabe eigentlich genug. Ich hoffe, dass sich all jene, denen des Rätsels Lösung bisher vorenthalten blieb, mit Flüchen und Bannsprüchen an meine Adresse zurückhalten konnten. Ein Lauflicht mit acht LED's macht eben mehr her, als eines mit nur sieben. Außerdem ist mit fehlendem A0-Signal unser unteres Byte nicht komplett.

Die tragende Rolle bei der Erzeugung von A0 kommt dem Signal /UDS zu. Beim Zugriff auf eine gerade Adresse wird dieses Signal aktiviert, geht in diesem Fall von Hi nach Lo (/UDS ist Lo-aktiv!). Im Prinzip hätten wir damit schon unsere A0-Leitung, doch leider wird /UDS gemeinsam mit den /ROM-Signalen deaktiviert, so dass unser Flip-Flop im IC 74LS374 immer nur den inaktiven Zustand von /UDS speichert. A0 wäre dann also immer Hi (vielleicht nicht der schlechteste Zustand ;-)). Genau betrachtet stehen wir hier vor einem Timing-Problem, was nichts anderes heißt, als dass uns der zeitliche Ablauf der Signale nicht in den Kram passt. Glücklicherweise soll /UDS nicht früher sondern später am Flip-Flop anliegen, als von der CPU geliefert. Wir müssen durch geeignete Maßnahmen also dafür sorgen, dass /UDS etwas verzögert wird. Ein Knoten in der Leitbahn verspricht mäßigen Erfolg, wir entscheiden uns deshalb für die bewährte Methode die Gatterlaufzeiten eines IC's als Signalverzögerung zu missbrauchen.

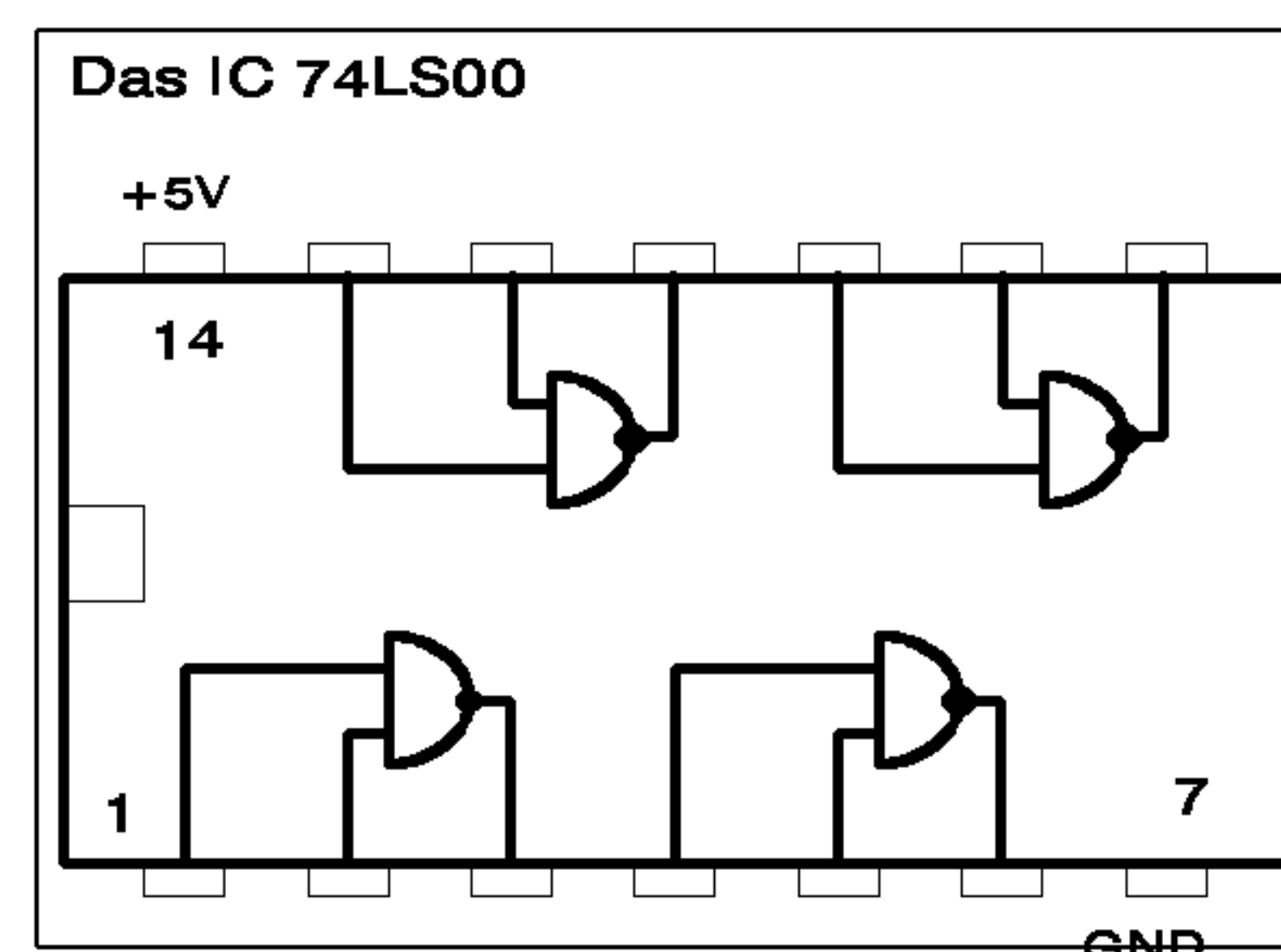
Dazu kann man fast jedes beliebige IC hernehmen, ich habe mich für ein 74LS00 entschieden, weil's gerade auf dem Tisch herumlag. Das 74LS00 invertiert das Eingangssignal, deshalb lassen wir es durch zwei Gatter laufen. Am Ausgang des zweiten Gatters steht dann das verzögerte /UDS an, das haargenau dem gewünschten A0-Signal entspricht, und mit dem unser Flip-Flop gefüttert werden kann.



rade Adresse Lo, durch Invertierung also Hi, und würde somit die Forderungen an das A0-Signal ebenfalls erfüllen. Diese Annahme ist aber nur solange richtig, wie mit Bytes, also 8 Bit gearbeitet wird. Der 8-Bit-Zugriff ist beim Atari auf jede Adresse des Romports erlaubt, der 16-Bit-Zugriff jedoch nur auf gerade Adressen.

Beim Zugriff auf 16 Bit werden sowohl /UDS, als auch /LDS aktiv. Ein invertiertes /LDS würde nun ein aktives A0-Signal liefern, abrrrr das stimmt nicht ..., /UDS hingegen funktioniert richtig.

Der messerscharfe Verstand einiger Bastler wird nun folgern, dass bei Invertierung des Signals eigentlich auch /LDS zur Erzeugung von A0 herangezogen werden könnte. Denn /LDS wird bei Zugriff auf eine unge-



Die andere Richtung: So, wie man Daten über den Romport aus dem Rechner herausschauft wissen wir nun.

Jetzt geht es darum, Daten auf dem gleichen Wege in den Atari einzuschleusen.

Dazu benötigt man zunächst einmal Daten, die man einlesen kann. Eine zusätzliche Bastelarbeit, oder gar ein zweiter Rechner ist glücklicherweise nicht erforderlich, denn alles, was für die Produktion externer Daten benötigt wird, ist mit der LED-Ansteuerungsgeschichte bereits erschlagen.

In dem am Romport hängenden 74LS374 kann ein Byte gespeichert werden (A0 haben wir ja jetzt), das bis zum nächsten Lesezugriff in den Adressbereich von FB0000 bis FBFFFF auch dort verfügbar bleibt. Dieses Byte läßt sich natürlich auch wieder in den Rechner einlesen. Dazu ist ein weiteres IC erforderlich, das das anstehende Byte zum richtigen Zeitpunkt auf den Datenbus des Atari legt, ohne die anderen Prozesse im Rechner zu stören.

Das IC

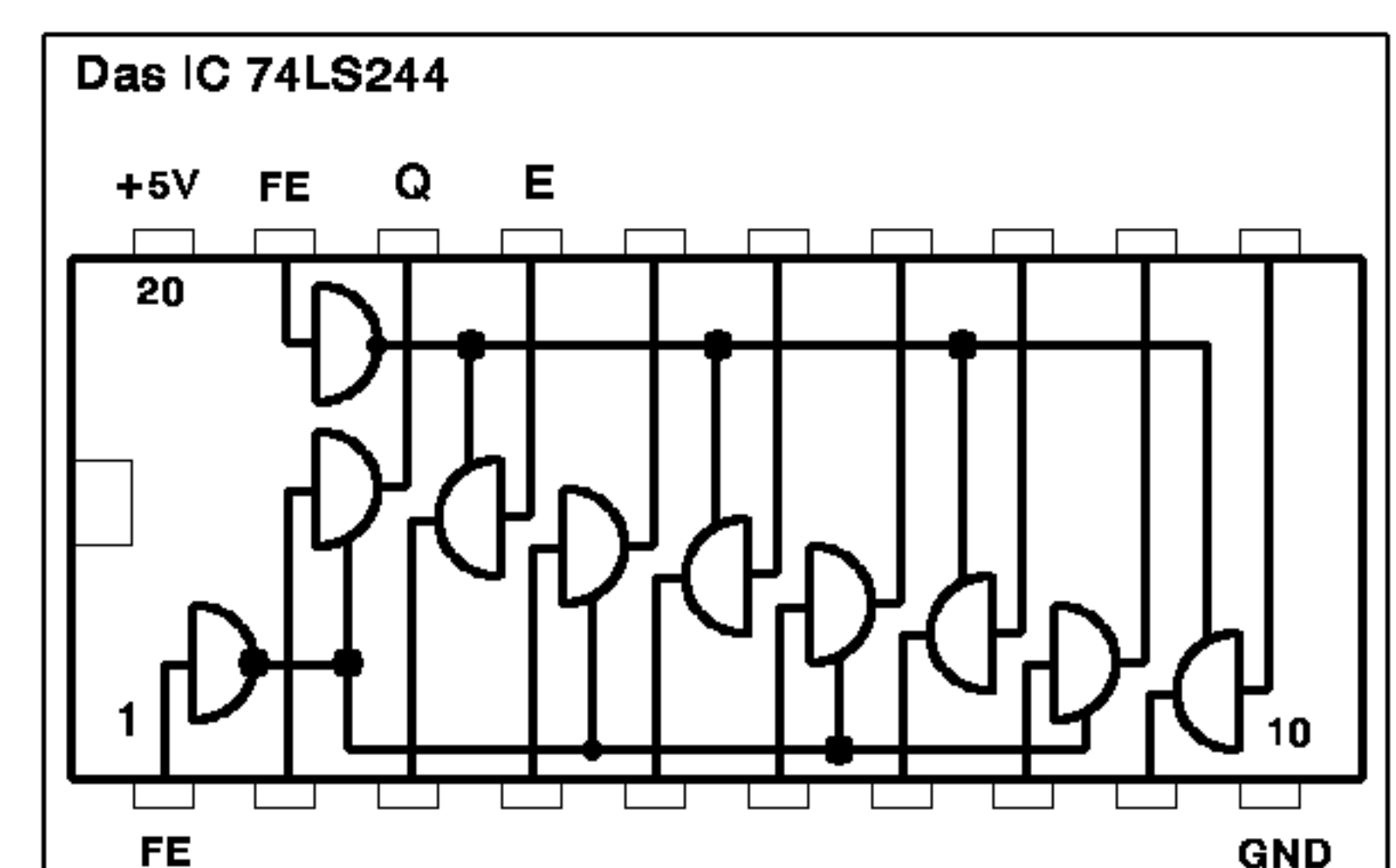
Zum Einschleusen von Daten in den Rechner bietet sich ein 74LS244 an. Dieses IC ist ein 8-Bit-Bustreiber, dessen Ausgänge auch in einen hochohmigen Zustand (tri-state) versetzt werden können. Das IC verfügt über 20 Pins, 8 Eingänge (E), 8 Ausgänge (Q), einen Anschluß für 5Volt (Pin 20), einen für Masse (Pin 10) und zwei mit FE bezeichnete Anschlüsse (Pins 1 und 19).

Zwischen 5Volt und Masse gehört (wie bei jedem IC!) ein Kondensator von 100nF, um mögliche Spannungsspitzen, die durch die Schaltvorgänge im IC entstehen können, abzufangen. Das 74LS244 tut eigentlich nichts anderes, als die Daten, die an den Eingängen anliegen, an den Ausgängen zur Verfügung zu stellen. Insofern könnte man das IC eigentlich als überflüssig betrachten, denn es stellt ja lediglich etwas zur Verfügung, was ohnehin schon da ist.

Fast richtig – aber wie die Bezeichnung "Bustreiber" schon andeutet, werden die an den Eingängen anliegenden Signale aufbereitet, so dass sie mehr Bauelemente ansteuern (treiben) können. Außerdem lassen sich die Ausgänge in den tri-state, also den hochohmigen Zustand, versetzen. Dadurch haben die an den Eingängen anliegenden Signale keinen Zugang zum Datenbus und können somit auch keinen "Datensalat" verursachen. Über

FE läßt sich dieser hochohmige Zustand der Ausgänge herbeiführen: Liegen die FE-Eingänge des IC's auf Hi-Pegel, so verhält sich das 74LS244 so, als ob es gar nicht vorhanden wäre.

Legt man die FE-Eingänge allerdings auf Lo-Pegel, so werden die an den Eingängen anliegenden Signale an den Ausgängen des IC's sichtbar und man kann sie verarbeiten.



Rein in die Schaltung

Nachdem das 74LS00 oder sowas zur Generierung von A0 unterhalb des 74LS374 eingelötet und verdrahtet ist (100nF –Kondensator nicht vergessen), bietet sich als Wohnort für das 74LS244 der Platz unterhalb des 74LS00 an (die drei IC’s fluchten). Die Verdrahtung sollte keine Probleme bereiten.

Erst ’mal die Spannungsversorgung herstellen, dann die Eingänge des 74LS244 mit den Ausgängen des 74LS374 verbinden. Netterweise liegen die Ausgänge des 74LS374 direkt an den Anoden der LED’s, so dass man hier hübsch der Reihe nach vorgehen kann. Es ist wichtig die Übersicht nicht zu verlieren, denn zu jedem Eingang gehört auch ein Ausgang, der mit der richtigen Datenleitung des Romports verbunden werden muss. A0 vom 74LS374 soll über das 74LS244 auf der Datenleitung D0 des Romports landen, A1 entsprechend auf D1, usw., dann gibts auch keine faltige Stirn beim Probelauf. Zu guter letzt werden noch die beiden FE-Eingänge (Pin 1 und 19) mit /ROM4 verbunden, das war’s dann auch schon.

Wie funktioniert?

Wie ein Datenbyte in die Flip-Flops geladen wird wissen wir: Es wird eine Adresse im Bereich von FB0000 bis FB00FF per PEEK-Befehl angesprochen. Es findet also ein Lesezugriff statt. Die gewählte Adresse wird durch die Speicherung in den Flip-Flops zu einem Datum (ist die Einzahl von "Daten") und bringt die entsprechenden LED’s zum Leuchten. Dadurch, dass die Ausgänge der Flip-Flops des 74LS374 mit den Eingängen des 74LS244 verbunden sind, steht dieses Datum jetzt auch am Datenbus des Atari zur Verfügung. Das 74LS244 lässt dieses Datum nur nicht durch, weil die FE-Eingänge auf Hi-Potential liegen.

Die FE-Eingänge des 74LS244 werden durch /ROM4 gesteuert. Dieses Signal wird aktiv, also Lo, wenn aus dem Speicherbereich von FA0000 bis FAFFFF gelesen wird. Der GFA-Befehl "PRINT PEEK(&HFB0000+8)" speichert den Zahlenwert "8" im 74LS374 und lässt die vierte LED leuchten. Als Wert auf dem Bildschirm erhält man "255", denn es wird zwar /ROM3 aktiv, um die angelegte Adresse als Datum zu speichern, nicht aber /ROM4, um den gespeicherten Wert einzulesen. Folgt dann der GFA-Befehl "PRINT PEEK(&FA0001)", so sollte auf dem Bildschirm der in den Flip-Flops gespeicherte Wert von "8" erscheinen. Serviert der Rechner einen anderen Wert, dann sind irgendwelche Leitungen vertauscht.

Warum FA0001 und nicht FA0000, wird sich manch’ einer fragen, denn bei Abfrage der Adresse FA0000 erhält man den Zahlenwert 255 auf dem Bildschirm. Die Antwort verbirgt sich im Motorola-Datenformat. Das obere Byte eines Datenwortes (1word = 2byte = 16bit), also D8 bis D15, liegt immer an einer geraden Adresse. Das untere Byte, also D0 bis D7, immer an einer ungeraden. Das wirkt auf den ersten Blick verkehrt herum, funktioniert aber trotzdem. Die Abfrage von D0 bis D15 bringt für obiges Beispiel den Wert 65288 auf den Schirm. Dieser setzt sich folgendermaßen zusammen:

Die Datenleitungen D8 bis D15 werden von unserer Hardware nicht bedient und bleiben im "Ruhezustand". Der ist für Datenleitungen Hi, deshalb hat das obere Byte den hexadezimalen Wert FF (dezimal 255). Das untere Byte hat den in den Flip-Flops gespeicherten hexadezimalen Wert 08 (dezimal 8). Der hexadezimale Wert des Wortes (= 2byte) ist somit FF08 (dezimal 65288). Doch mit den Zahlenspielereien befassen wir uns später.

Die Wahl der /ROM-Signale für das Speichern bzw. Auslesen von Daten ist nicht zufällig. Der Atari überprüft nach dem Einschalten das Vorhandensein eines Rom-Modules. Dabei wird FA0000 abgefragt und bei bestimmten vorhandenen Datenworten auf die Präsenz eines Rom-Modules geschlossen. Diese automatische Abfrage würde nun schon ein Datum in unsere Flip-Flops schreiben, wenn /ROM4 als Taktleitung benutzt würde. Durch das Ausweichen auf den von /ROM3 bedienten Adressbereich wird diese Möglichkeit ausgeschlossen. Das Signal /ROM3 auch zum Einlesen der Daten zu benutzen, würde bedeuten, dass man, wenn das Datum in den Flip-Flops des 74LS374 nicht verändert

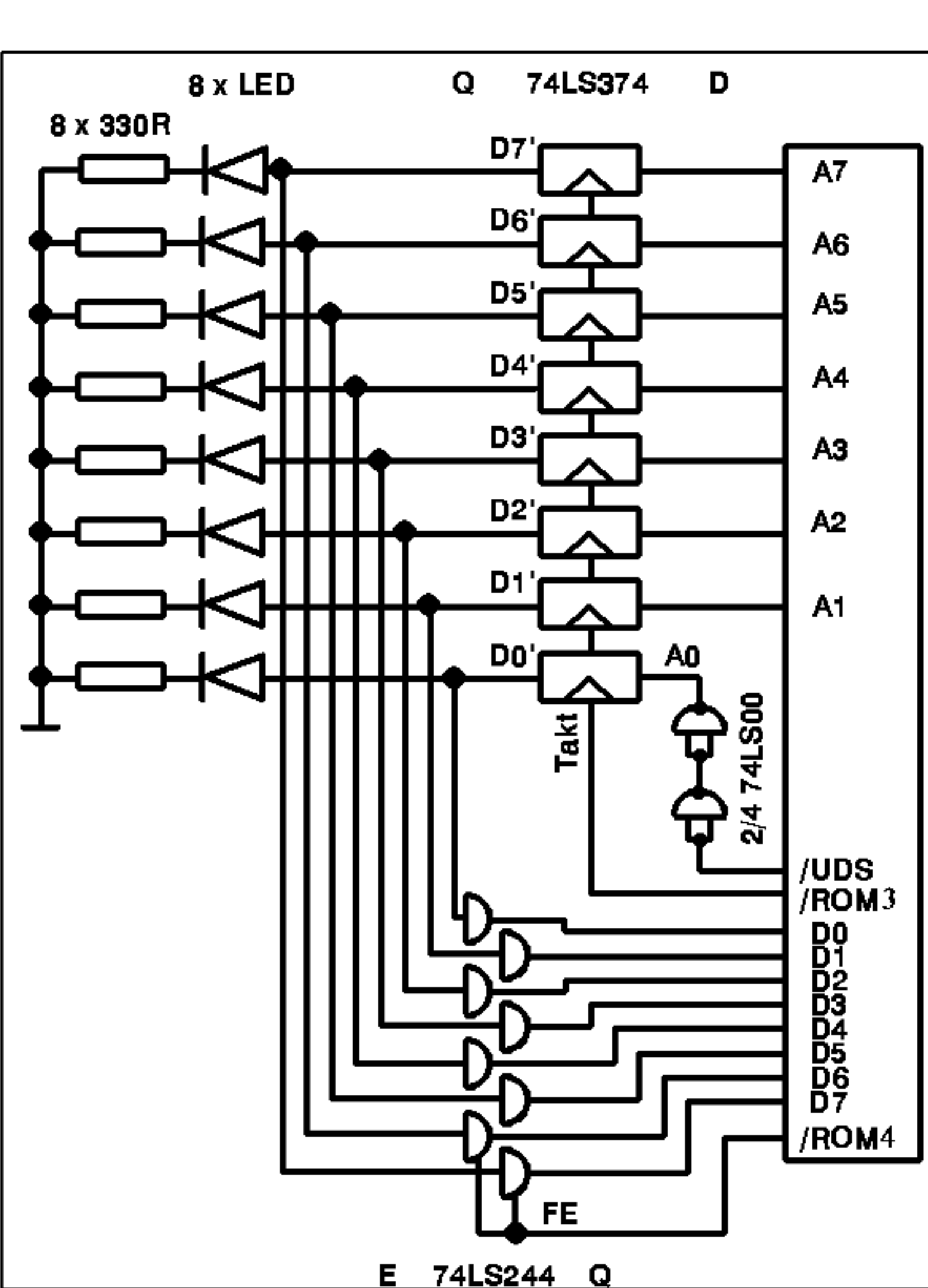
```
' Mit diesem Programm werden Daten in den
' Flip-Flops hochgezählt, permanent einge-
' lesen und mit einem Vorgabewert verglichen.
' Bei Übereinstimmung stoppt das Programm.
'
'
INPUT "Zahl bis 255: ",z           !Vorgabewert
FOR i=0 TO 255 a=PEEK(&HFB0000+i) !Daten hochzählen
  b=PEEK(&HFA0001)                !Daten in b lesen
  PRINT b                         !gelesene zeigen
  IF b=z                          !wenn b=Vorgabe
    PRINT "Zahlenwert erreicht, Taste" c=INP(2) !warten auf Taste
    EDIT                          !zum Interpreter
  ENDIF
  PAUSE 1
NEXT i
```

werden soll, immer mit der Adresse abfragen müsste, die als Datum in den Flip-Flops liegt. Auch nicht’ gut.

Die Schaltung im Ganzen

Auf den ersten Blick mag das alles nicht sehr beeindruckend wirken, doch die Grundvoraussetzungen für Steuerungen mit dem Computer sind mit dieser Schaltung schon gegeben.

Genau betrachtet wurde bisher lediglich ein 8bit Datenbus realisiert. Über das 74LS374 schaffen wir Daten raus, über das 74LS244 läuft’s in die andere Richtung. Die LED’s dienen der optischen Kontrolle und helfen uns in diesem Stadium Schaltungsfehler aufzuspüren.



Hier noch einmal alles im Überblick, die Kondensatoren habe ich dabei weggelassen.

Wie geht’s weiter? In der nächsten Ausgabe wird es etwas theoretisch. Da gibt es einiges über das Dual- und Hexadezimalsystem zu lesen, außerdem sollen aufgetauchte Fragen beantwortet und Anregungen behandelt werden. Einige Erläuterungen zur Programmierung runden das trockene Thema ab. Damit es nicht gar zu langweilig wird, realisieren wir noch einen einfachen Digital/Analog-Wandler. Dazu ist noch ein Widerstand nötig, der Wert sollte so zwischen 10R und 50R liegen.

Ende Teil 3. Weiter geht’s mit ’nem neuen Dokument.

Ulrich Skulimma