Lotus 2000
Eigenbau eines Computers (1980er Jahre)

Eberhard von Faber
August 2018

Obwohl PCs und andere Computer in den 1980er Jahren auch in der DDR langsam an Universitäten und in Forschungseinrichtungen Verbreitung fanden, war es doch für den Privatmann nicht oder kaum möglich, einen solchen zu kaufen. Deshalb habe ich 1985 begonnen, mir einen Computer auf Basis der Z80-Architektur selbst zu bauen. Bis auf das 5V-Netzteil kamen keine vorgefertigten Baugruppen zur Anwendung. Alles ist selbst entworfen und aus Einzelteilen gebaut, die in der DDR beschaffbar waren. Die Herausforderung bestand darin, den Computer samt Betriebssystem ohne externe Hilfe aufzubauen. Eine der wenigen externen Hilfestellungen[1] bestand darin, dass ich die ersten 1 KByte-Programmcode auf einen E2PROM mit Hilfe eines fremden Entwicklungsrechners gebrannt habe. Zusätzlich konnte ich ein paar einfache Platinen bekommen bzw. abkupfern. Alle anderen Einsteckkarten sind mit Universalplatinen gefertigt. Auch die Software wurde überwiegend von Hand assembliert, da ja keinerlei sonstige Infrastruktur zur Verfügung stand. Doch ohne den gelegentliche Rückgriff auf ein Entwicklungssystem hätten manche Funktionalitäten wohl nicht realisiert werden können. Eines der Hauptproblem bestand übrigens in der Beschaffung der Steckverbinder. In der DDR war eben fast alles knapp.

1 Übersicht Endausbau

Abb. 1 zeigt den Computer auf dem Stand von etwa 1990, danach hat sich die Hardware nicht mehr verändert. Hinten rechts steht der Computer mit neun Karten im doppelten Europakartenformat und den drei Netzteilen für die vier benötigten Spannungen (-12V, -5V, +5V, +12V). Das am höchsten integrierte Bauteil ist der Z80-SIO-Baustein. Der Systembus hat 2x 29 Kontakte, und es gibt einen zusätzlichen Koppelbus für Sonderaufgaben.

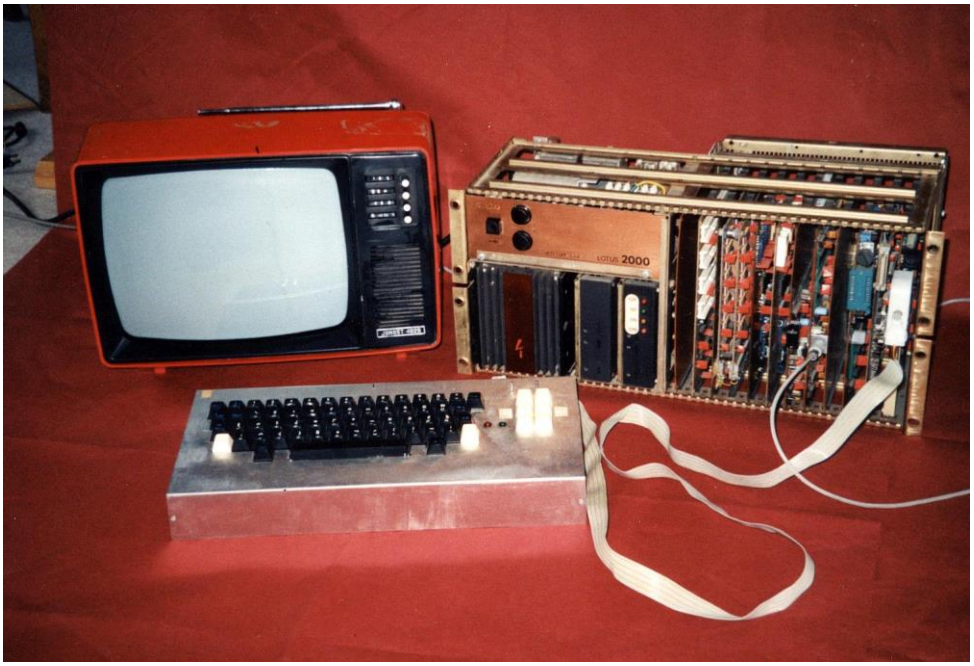


Abb. 1: Der fertige Computer mit „Display“ und Tastatur (Stand 1989)

[1] Ich werde weiter unten jeweils darauf eingehen, in wieweit Arbeiten an einem vorhandenen Entwicklungssystem nötig waren und in Anspruch genommen wurden.

Tab. 1 zeigt einige Eigenschaften. Verbaut wurden ca. 300 Integrierte Schaltkreise (ohne Netzteile). Die Zeit für Entwicklung und Bau von Hardware und Software betrug knapp fünf Jahre.

Tab. 1: Technische Daten Lotus 2000

Prozessor; Taktfrequenz	U880 (Z80-Version); 2,5 MHz (4 MHz nicht getestet)
Bussystem	16 Bit Adressen, 8 Bit Daten
Speicher (max.)	64 KByte; durch integriertes Bank-System erweiterbar
Besonderheiten	proprietärer Eigenbau; etwa 300 integrierte Schaltkreise, 20 Steckverbinder und Mengen anderer elektronische Bauelemente handgelötet; Computerelektronik völlig eigenständig entwickelt bzw. in Anlehnung an K1520 von Robotron realisiert; Software singulär (ca. 11 KByte)
Realisierung	Ende 1985 bis ca. 1990
Platinen; ICs	9 Platinen im Doppeleuropaformat; 271 ICs
Debugger-Elektronik mit Hexadezimaltastatur	Stopp und Schrittbetrieb mit Anzeige der Businhalte mittels Hexadezimalanzeigen sowie Anzeige der Steuersignale bzw. Zustände; 28 ICs
Stromversorgung	+12V (2,5A), +5V (12A), -5V (1A), -12V (0,5A); mit Einschaltreihenfolge und Überspannungsschutz; +5V per Schaltnetzteil
Bildschirm	24 oder 32 Zeilen á 64 Zeichen (max. 2048 Zeichen pro Bild); Zeichensatz: 256 alphanumerische und 256 Pseudografik-Zeichen; Funktion: prozessorunabhängig; Bildschirm: teilbar
Peripherie	▪ Tastatur mit 66 Halltasten (über U855-PIO); ▪ Tonbandgerät (Auslagern und Einlesen über U856-SIO); ▪ Drucker (Centronics über U855-PIO); ▪ EPROM-Programmiergerät (2708 bis 27128 über U855-PIO); ▪ weitere SIO- und PIO-Schnittstellen als Reserve (alle über Interrupt-Daisy-Chain betrieben)

In Abb. 2 erkennt man den Aufbau etwas besser. Die neun Karten des Endausbaus sind:

- CPU-Hauptplatine (U880D mit 2,5 MHz, Taktgenerator, diverse Bustreiber 8212 bzw. 8216),
- Speicherkarte Mem1 (2 KByte RAM und 6 KByte EPROM),
- Speicherkarte Mem2 (wie Mem1: 2 KByte RAM und 6 KByte EPROM),
- Speicherkarte Mem3 (16 KByte EPROM aus 16 x 2708-Schaltkreisen),
- Speicherkarte Mem4 (16 KByte dynamischer RAM aus KP565PY1),
- Peripherie 1 (Tastatur, Tonbandgerät): 2x Z80-PIO zur Ansteuerung der Tastatur und weiterer Geräte, 1x Z80-CTC-Baustein, 1x Z80-SIO für die Speicherung von Daten auf einem Tonbandgerät,
- Peripherie 2 (Drucker): Z80-PIO zur Ansteuerung eines Druckers, Z80-CTC-Baustein, Z80-SIO,
- Peripherie 3 (EPROM-Programmierung): 1x Z80-PIO für die Programmierung von EPROMs,
- CRT-Controller zur Ansteuerung eines Fernsehers als Bildschirm.

Die Entwicklungshilfen

- Hexadezimaltastatur und Bustreibersystem,
- Schrittsteuerung und Busanzeige sowie
- Selbstbau-Messinstrumente wie TTL-Prüfer

betrachten wir später.

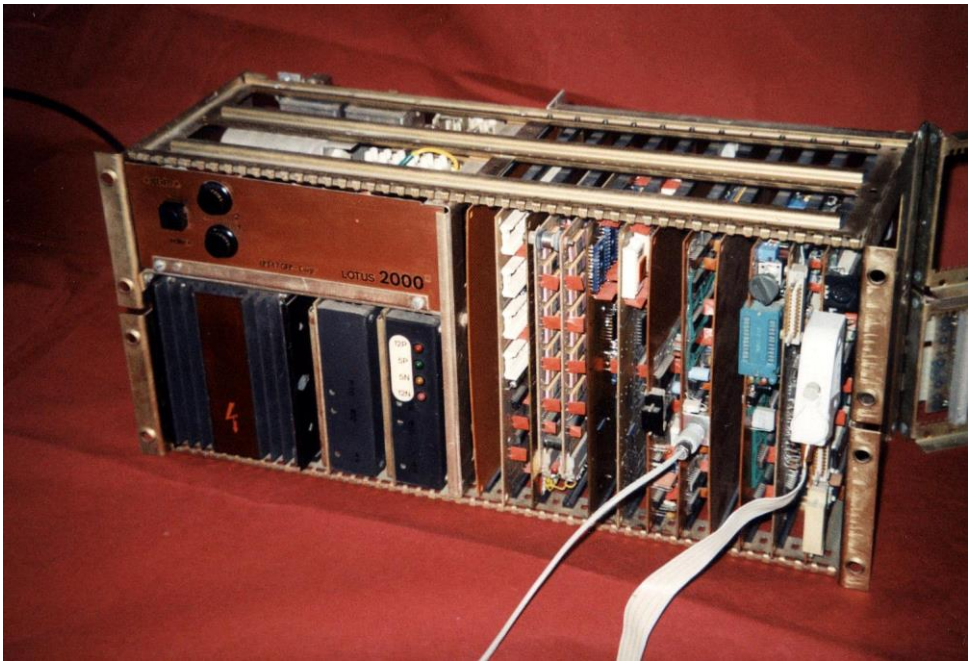


Abb. 2: Der fertige Computer (Stand 1990)

Abb. 3 zeigt die Rückseite des Computers. Hier ist ein zweiter Rahmen ausschwenkbar, der weitere Platinen aufnehmen kann. Insbesondere im Debugger-Modus mit der „Schrittsteuerung und Busanzeige“ ist das Umklappen sehr praktisch, da die Anzeigen dann nach vorne zeigen.

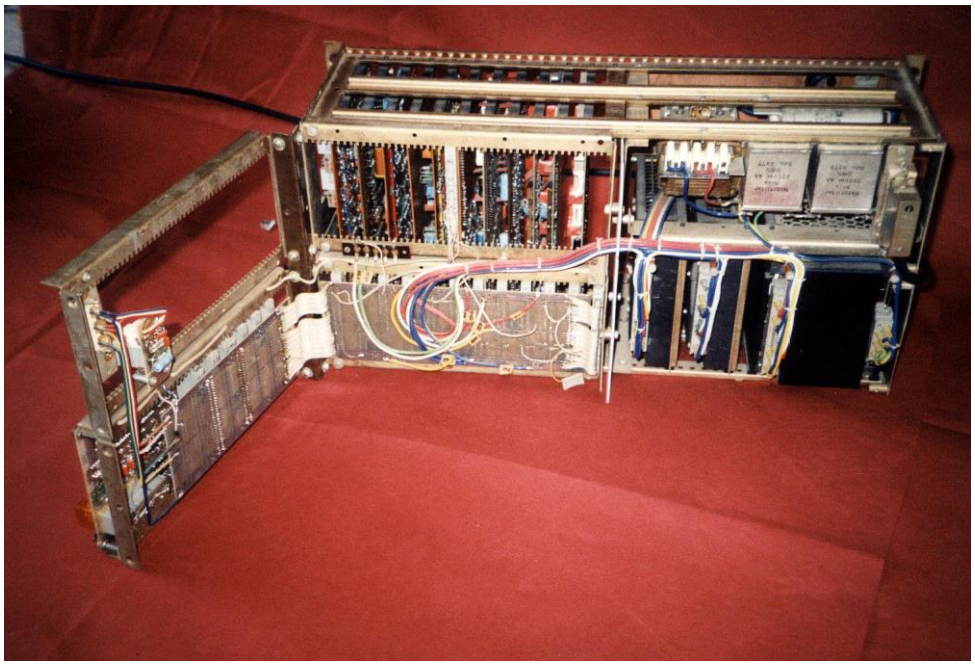


Abb. 3: Der fertige Computer (Rückseite, Stand 1990)

Dieser Beitrag ist wie folgt aufgebaut:

- In Kapitel 2 werden die CPU-Hauptplatine und weitere Baugruppen vorgestellt, die beim schrittweisen Aufbau des Computers eine wichtige Rolle gespielt haben.
- In Kapitel 3 geht es um die Ausgabe auf dem Fernseher, die Tastatureingabe, die Datenablage auf Magnetbändern und das Drucken.
- In Kapitel 4 findet man eine Übersicht über die unterschiedlichen Speicherlösungen.
- Kapitel 5 gibt weitere Einblicke in die Programmierung des „Betriebssystems“.
- Einen Ausblick bietet Kapitel 6. Ganz am Ende sind ein paar Informationen über den Erbauer zu finden.

2 Entwicklung

2.1 CPU-Hauptplatine

Die CPU ist das Herz eines Computers, so dass diese Platine zuerst fertiggestellt wurde. Abb. 4 zeigt die CPU-Hauptplatine. Wie bei den meisten Platinen wurde eine Universalplatine verwendet. Die Verdrahtung erfolgt per Hand mit Hilfe der dünnen grün und silbern isolierten Drähte. Der Baustein in der Mitte ist eine Z80-CPU in der DDR-Ausführung als U880D. Links liegt der 2,5 MHz Taktgenerator und oben links erkennt man die Reset-Logik. Die anderen Schaltkreise sind überwiegend Bustreiber für Adressen (8212) sowie Daten und Steuersignale (3216). Es werden besondere Steuersignale für die Speicherverwaltung erzeugt und es gibt einen 8-Bit-Ausgang (Latch) für Diagnosezwecke. Der Systembus ist auf allen Abbildungen links unten zu finden. Rechts unten erkennt man bei der CPU-Hauptplatine (Abb. 4) den zusätzlichen Koppelbus, mit dem besondere Steuersignale an andere Platinen verteilt werden.

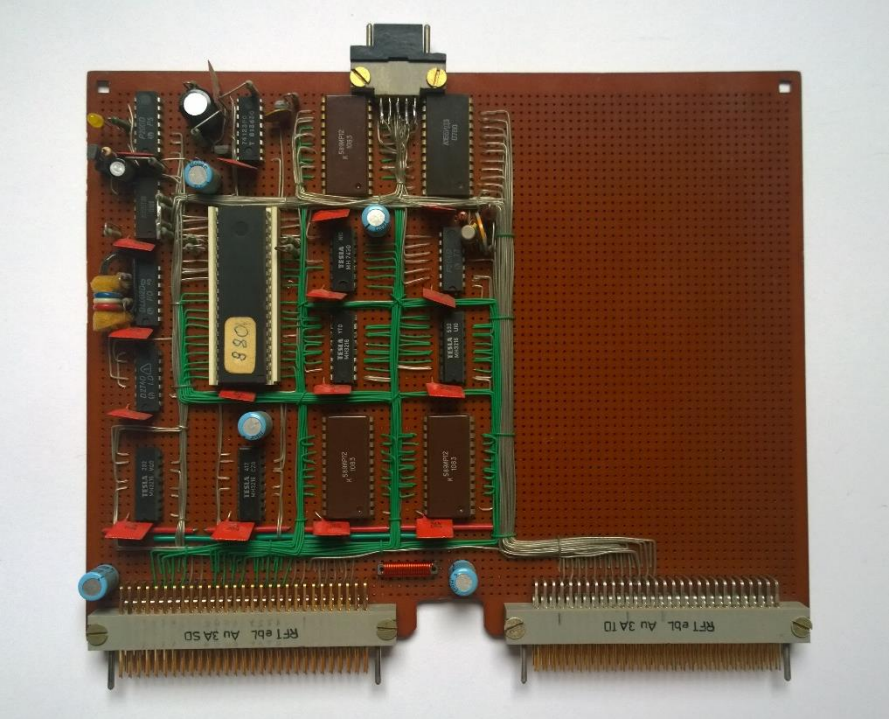


Abb. 4: CPU-Hauptplatine

2.2 Tastatur mit 20 Tasten und etwas Elektronik für erste Hardwaretests

Die CPU-Hauptplatine wurde ganz zu Anfang fertiggestellt. Doch wie kann man eine solche Logik testen, ohne Gefahr zu laufen, weitere ggf. schon fertiggestellte Platinen zu zerstören? Abb. 5 zeigt die Lösung. Zu sehen sind die Hexadezimaltastatur mit 20 Tastern, das Bustreibersystem (Platine mit vier Schaltkreisen) und die LED-Datenanzeige (8 LED verbunden mit der CPU-Hauptplatine).

Zunächst war an eine richtige alphanumerische Tastatur nicht zu denken. Daher habe ich eine Hexadezimaltastatur mit vier Sondertasten gebaut. Das Bild weist auf ein weiteres großes Problem der Materialbeschaffung in der DDR hin. Aus Mangel an Flachbandkabeln wurde eines aus zweiadrigen Leitungen geflochten. Die kleine Platine rechts in Abb. 5 ist ein „Bustreibersystem“. Zusammen mit der Hexadezimaltastatur kann man den Adressbus und den Datenbus schreiben sowie z.B. Reset oder einen nicht-maskierbaren Interrupt auslösen. Eine Ausgabe der CPU an den bereits erwähnten 8-bit-Latch erscheint an den 8 LED der Datenanzeige, die links oberhalb der Tastatur zu sehen ist. Hexadezimaltastatur und Bustreibersystem fungieren zugleich als Taktgeber für die CPU. Mit diesem System können viele Test an der CPU-Platine durchgeführt werden.

Abb. 5: Hexadezimaltastatur, Bustreibersystem und LED-Datenanzeige

2.3 Schrittsteuerung und Busanzeige fürs Debugging

Ein weiteres, später entwickelt und gebautes System zum Debuggen (Fehlersuche) ist die „Schrittsteuerung und Busanzeige“, die in Abb. 5 zu sehen ist. Dargestellt werden der Inhalt des Adressbusses (vier Siebensegmentanzeigen, hexadezimal) und des Datenbusses (zwei Siebensegmentanzeigen, hexadezimal). Zusätzlich die Art des Zugriffs (eine Siebensegmentanzeige, Buchstaben) und acht Steuerbussignale (einzelne LED). Die Taster bzw. Schalter erlauben es, die CPU anzuhalten (Wait), einen Schritt auszuführen (Step) bzw. die normale Programmausführung zu starten (Run). Mit Hilfe der Schalter kann die Programmausführung in einen stark verlangsamten Modus erfolgen, und es können ein Reset und ein nicht-maskierbarer Interrupt ausgelöst werden. Die Platine wird auf den hinteren, ausschwenkbaren Teil des Busses gesteckt. Nach dem Ausschwenken sind die Anzeigen an der Vorderseite des Lotus 2000 sichtbar.

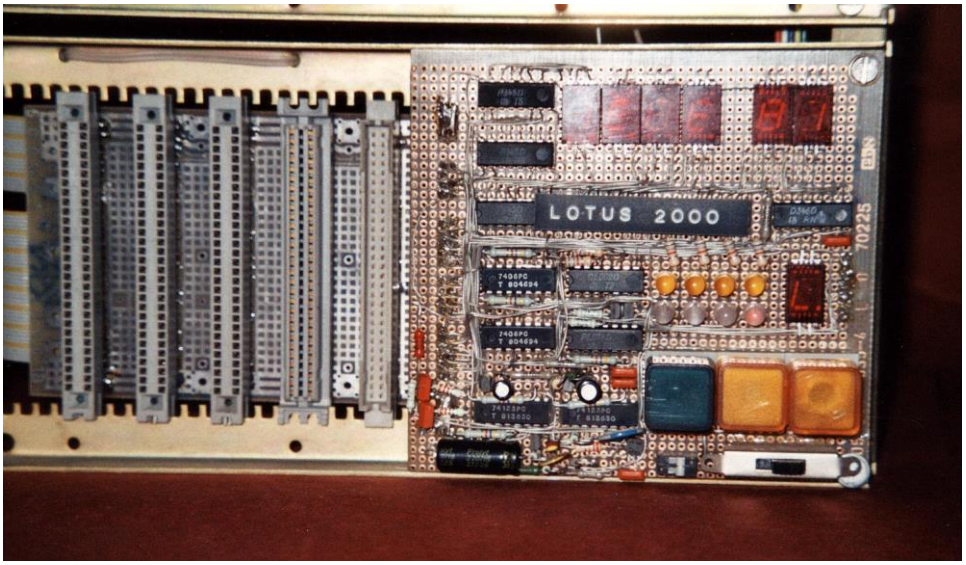


Abb. 6: Schrittsteuerung und Busanzeige

2.4 EPROMs selbst programmieren

Die Software (Betriebssystem) für Lotus 2000 wächst langsam. Es handelt sich schließlich um eine Eigenentwicklung auf einer proprietären Plattform.[2] Nachdem CPU-Hauptplatine, mindestens eine Speicherkarte und etwas Software stabil liefen, konnte es an die Entwicklung und Realisierung der Elektronik und der Software zum Programmieren von EPROMs gehen. EPROMs selbst programmieren zu können, war die Voraussetzung für eine eigenständige, von fremder Hilfe unabhängiger Entwicklung der Software.

Die Elektronik für die EPROM-Programmierung ist auf einer eigenen Platine untergebracht, die in Abb. 7 zu sehen ist. Die Elektronik ist relativ aufwändig, auch wenn neben der Z80-PIO (in der DDR-Ausführung als U855) das meiste Treiber sind und die eigentliche Herausforderung in der Erzeugung sauberer Flanken und Takte für die Programmierung ist. Das große Reed-Relais hinten links schaltet die Spannungen (-5V, +5V, +12V und +30V). Obwohl ich praktisch nie über 1 KByte-EPROMs hinausgekommen bin und nur einmal einen mit doppelten Kapazität eingesetzt habe, unterstützt die Elektronik alle Typen vom 2708 mit 1 KByte bis zum 27128 mit 16 KByte. Zur Umschaltung dient der Drehschalter links unten, der schwer zu besorgen war. Es mussten drei kleinere Platinen auf der Hauptplatine aufgestellt werden, eine für den Schalter, einen für die EPROM-Schwenkfassung und einen für die Satus-LEDs. Die Programmierung erfolgt vollständig interrupt-gesteuert.

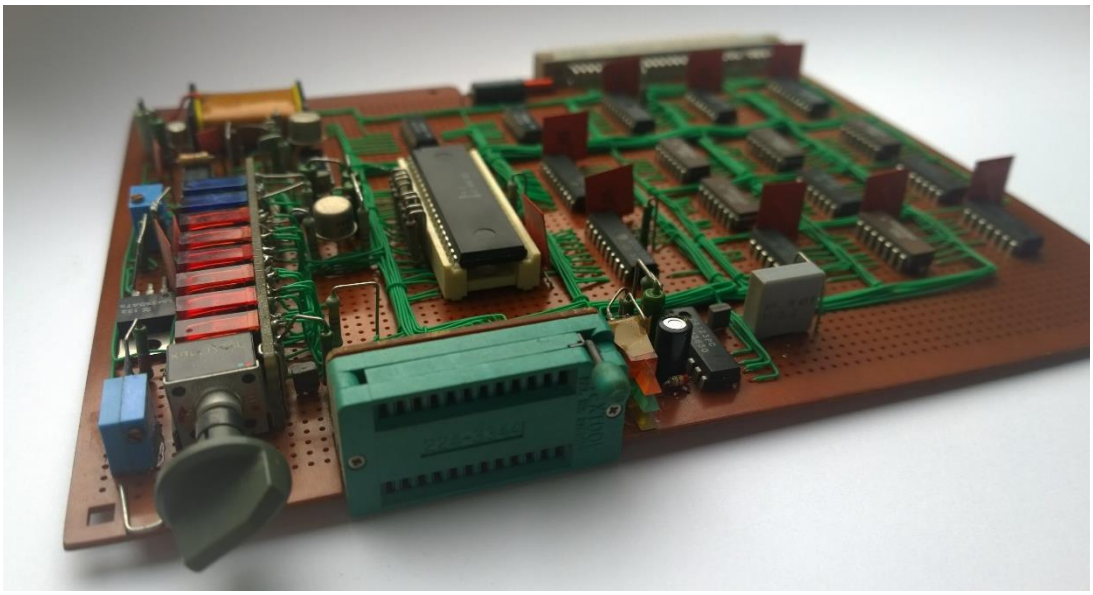


Abb. 7: Platine zum Programmieren von E2PROMs

Nachdem Elektronik und Programmiersoftware endlich fehlerfrei funktionierten, musste Software nicht bei jedem Neuversuch aufwendig eingetippt werden, und ich war endlich vollständig unabhängig von anderen Entwicklungsrechnern.

3 Sehen, Archivieren, Drucken

3.1 Sehen: Textausgabe auf dem Fernseher

In den Anfangszeiten mit Computern in Privathand[3] wurden Fernseher zur Anzeige (als „Displays“) genutzt, und es wurden auch nur alphanumerische Zeichen (mit Graphikersatzeichen) dargestellt. Mitte der 1980er Jahre war für mich ohnehin an nichts anderes als einen Fernseher zu denken. Daher musste ein Fernsehbild her. Den CRT-Controller zur Bildschirmausgabe zeigt Abb. 8. Dies ist die komplizierteste und komplexeste Elektronik des Lotus 2000. Der aufgezeichnete Stromlaufplan misst 75 cm x 42 cm. Eine Aufteilung auf mehrere Blätter war diesmal nicht möglich, und ich habe Millimeterpapier benutzt, um die Details darstellen zu können.

---

[2] Auch wenn sich die Gesamtarchitektur stark an die des K1520 von Robotron anlehnt, sind doch beträchtliche Teile der Hardware proprietär.

[3] Die ersten Heimcomputer, ZX80 und ZX81, waren auch Z80-basiert. Die Darstellung erfolgt auf einem Fernseher.

Die Schaltung umfasst 43 Schaltkreise (ICs). Für die Logik wurden, wenn möglich, die DDR-Low-Power-Varianten (z.B. DL004) verwendet, aber an kritischen Stellen musste Importware her.

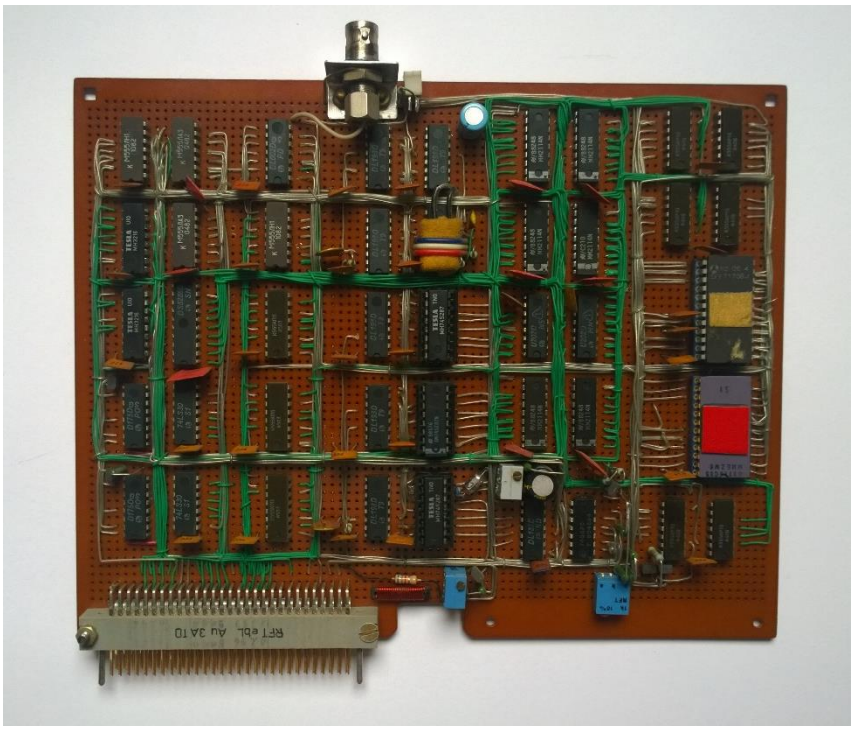


Abb. 8: CRT-Controller zur Bildschirmausgabe

Die Datenbustreiber sind bidirektional ausgelegt, denn der CRT-Controller verfügt über einen eigenen Speicher (RAM), den die CPU lesen und schreiben kann. Da die CPU auf der Hauptplatine jedoch in der Regel nicht darauf zugreift, wird der Adressraum ganz ausgeblendet und von Speicher auf einer anderen Platine belegt. Erst wenn die CPU Bildschirmprograme ausführt, erscheint der Speicher des CRT-Controllers im Adressraum der CPU. Ansonsten liest der CRT-Controller seinen Bildschirmspeicher selbst. Dazu gibt es eigene Zähler, die die Adressen erzeugen. Die dort ausgelesenen alphanumerischen Zeichen adressieren wiederum einen der EPROMs, der das Zeichen in Bildpunkte umsetzt. Da das Bild gemäß der damaligen Fernsehnorm Bildpunkt für Bildpunkt entlang einer Zeile an den Fernseher gesendet wird, wird jedes Zeichen in acht Zeilen nacheinander aufgerufen und in Bildpunkte dekodiert. Horizontal sind die Zeichen auf dem Bildschirm sechs Bildpunkte breit. Ist der letzte Punkt des letzten Zeichens am rechten Rand des Bildschirms übertragen, erzeugt die Elektronik die sogenannte horizontale Austastlücke (ein spezieller Impuls), und die Darstellung beginnt wieder am linken Rand. So geht es Zeile für Zeile. Wird die rechte untere Ecke erreicht, erzeugt die Elektronik die längere vertikale Austastlücke (einen speziellen Impuls) und die Darstellung beginnt wieder links oben. Auf diese Weise wird immer der aktuelle Inhalt dargestellt, und der Fernseher (Display) braucht sich nichts zu merken. Das Nachleuchten der Bildpunkte reicht für den Aufbau eines kompletten Bildes.

Der CRT-Controller ist umschaltbar. Es können 24 oder 32 Zeilen (mit jeweils 64 Zeichen) dargestellt werden. Dazu haben die Zeichen eine Höhe von 10 oder 8 Bildpunkten. Die Programmierung der EPROMs mit der Punktdarstellung der Zeichen war eine langweilige Tätigkeit. Es werden zwei Modi (Zeichen und Graphik) unterstützt, weshalb es zwei EPROMs für die Dekodierung von Zeichen in Bildpunkte gibt. Um neben der hellen eine graue Darstellung mit reduzierter Helligkeit zu unterstützen, muss für jedes Zeichen zusätzlich gespeichert werden, ob die hellen Punkte abgedunkelt oder die dunklen aufgehellt werden sollen. Zusätzlich wird ein blinkender Cursormodus unterstützt.[4] Insgesamt werden pro Zeichen also 12 Bit gespeichert, wobei

[4] Bei reiner Textausgabe sitzt der Cursor immer auf dem Zeichen. Um die Stelle von anderen unterscheiden zu können, kann das Zeichen unterstrichen werden und blinken. Nur bei der heute üblichen grafischen Darstellung kann der Cursor zwischen den Zeichen blinken und die Eingabestelle markieren.

8 Bit das Zeichen und die 4 Bit die Darstellung kodieren.[5] Da die CPU nur 8 Bit schreiben kann und zwei Ausgaben á 8 Bit als zu zeitraubend angesehen wurden, werden die 4 Bit vorher in ein Register (FDh) geschrieben. Werden nun laufend Zeichen in den 8 Bit-Zeichenspeicher geschrieben, können zeitgleich bis zu 4 Bit in den Darstellungsspeicher übernommen werden. Es ist aber auch möglich, nur den Darstellungsspeicher zu schreiben, etwa um vorhandenen Text auszugrauen. Da der Cursor im Betrieb ständig versetzt wird, kann dieser durch Schreiben eines Bits im Darstellungsspeichers verschoben werden, ohne dass sich etwas am Inhalt des Bildschirms bzw. des Zeichenspeichers ändert. Welche Schreiboperation im Darstellungsspeicher gerade gewünscht ist, kann durch Schreiben eines zweiten Registers (FCh) bestimmt werden.

Die Elektronik ist voller Zähler. Aus 9 MHz werden in vielen Schritten mehrere Frequenzen bis hinunter zur Bildwiederholfrequenz von 50 Hz gewonnen. Zum Schluss müssen die Bildpunktinformation der Zeichen, die Informationen für die Darstellung der Zeichen und die Austast- und Synchronsignale logisch verbunden werden, um das Ausgangssignal mit den richtigen Spannungen erzeugen zu können. Insgesamt erzeugt die Schaltung fünf verschiedene Spannungen (zwei für die Austast- und Synchronsignale, drei für die Helligkeit des Bildpunktes). Die Anforderungen an die Laufzeiten und die Signalqualität sind hier so hoch, dass schnelle programmierbare TTL-Speicher (74S287) zum Einsatz kamen, die an einem fremden Entwicklungssystem programmiert werden mussten. Diese Import-Schaltkreise waren rar und sicher nicht beim ersten Mal vollständig korrekt programmiert, weshalb sie, wie in Abb. 8 zu sehen ist, in schmucken Steckfassungen stecken. Ohne diese Schaltkreise hätte der Controller wohl auch nicht auf eine Platine gepasst.

Schaltkreise wurden meines Wissens systematisch importiert, während die DDR die Entwicklung eigener Varianten vorantrieb. Dadurch konnten die Anwender der Schaltkreise schon Schaltungen entwerfen und testen. Später, wenn die Produkte in Serie gingen, wurden die Chips aus DDR-Produktion eingesetzt. Es ist also nicht verwunderlich, dass auch Chips von Texas Instruments, National Semiconductor, Fairchild, Intel, Motorola usw. im Umlauf waren. Abb. 9 zeigt Beispiele ein und desselben Chips von unterschiedlichen Herstellern. Viele Chips kamen nicht nur aus der UdSSR (heute Russland), sondern auch von Tesla aus der ČSSR (heute Tschechien).

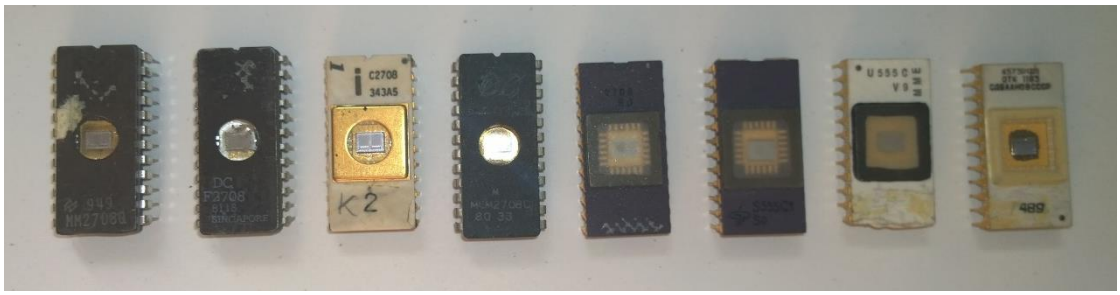


Abb. 9: Schaltkreis 2708 (v.l.n.r. National Semiconductor, Fairchild, Intel, Motorola, 3x DDR, UdSSR)

Die Software (Routinen im Betriebssystem) waren nicht minder kompliziert als die Elektronik. Auf die Umschaltung zwischen 24 und 32 Zeilen á 64 Zeichen wurde schon eingegangen. Der Bildschirm kann in zwei bzw. drei logische Einheiten geteilt werden, wobei zwei davon wie zwei eigenständige Bildschirme bedient werden können. Die Software erlaubt Schreiben, Lesen, Positionieren des Cursors in etwa einem Dutzend Varianten, Scrollen, Einsetzen und Verschieben von Text, Teilen des Bildschirms in zwei programmierbare Teile, Scrollen in einer der beiden, Einfügen und Löschen von Zeichen und vieles andere mehr. Insgesamt waren 1986 etwas mehr als 50 Programme implementiert, die etwa 1,5 KByte belegten.

3.2 Schreiben: Texteingabe wie am richtigen Computer

Ein richtiger Computer braucht eine richtige Tastatur. Durch großes Glück konnte ich eine ausrangierte, für den Schrott bestimmte Tastatur ergattern bzw. das, was von ihr übrig war. Viele der Tastschalter waren defekt, die Elektronik fehlte ganz. Nun ja, nicht ganz, denn die Tasten funktionierten mit hochmodernen Hall-Elementen. Leider glitten die Tastenköpfe trotzdem nicht ganz reibungsfrei, wenn sie betätigt wurden.

[5] Die vier Modi: Zeichen oder Graphik, Dunkel aufhellen ja/nein, Hell abdunkeln ja/nein, Cursormodus mit Blinken ja/nein.

Irgendwie habe ich genug davon an den rechten Platz bekommen können. Die fertige Tastatur hat 66 Tasten bzw. Hallelemente. Ich habe ein Blech beschafft, mit der Laubsäge bearbeitet und dann abgekantet. Das Ergebnis sieht man links in Abb. 10. Alle Tasten sind mindestens dreifach belegt. Aber es gibt Modi und Sondertasten. Ich konnte mich ja austoben und alle möglichen Funktionen integrieren, die mir irgendwie sinnvoll und hilfreich erschienen. Da ich, wie erwähnt, die Tasten neu einsetzen musste, war es ohnehin nötig viele der Tastenköpfe neu zu beschriften. Das sieht man ganz gut rechts in Abb. 10.

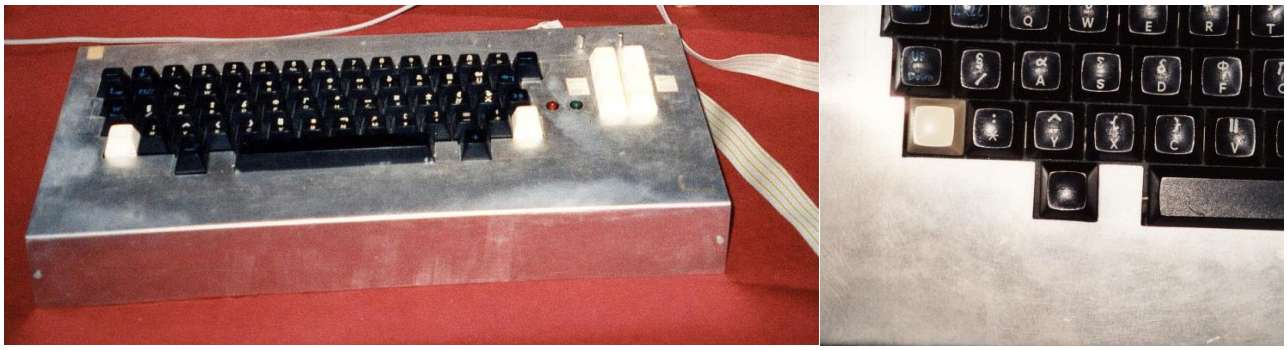


Abb. 10: Tastatur – alphanumerisch und mehr

Im Gehäuse der Tastatur steckt einiges an Elektronik. In heutigen PCs übernehmen Treiberprogramme des Betriebssystems fast alles. Bei Lotus 2000 liefert die Tastatur die fertigen Codes einschließlich Groß- und Kleinschreibung und sonstigen Umschaltungen (wie der Steuerungs- bzw. Ctrl-Taste). Da die Wiederholung beim Festhalten einer Taste ebenfalls durch die Tastaturelektronik erledigt wird, kann man mit einem Regler die Wiederholungsfrequenz einstellen. Und natürlich kann die Tastatur auch Signaltöne abgeben, da sie mit einem Lautsprecher ausgestattet ist. Die Umsetzung zwischen Tastenposition in einem Gitter und dem Tastencode übernimmt ein eingebauter EPROM (2708). Um Elektronik und Software im Computerinneren einfach zu halten, wird die Tastatur, anders als heute, über acht Datenleitungen (Parallelport, bidirektional) angeschlossen. Das war möglich, denn endlich hatte ich Flachbandkabel!

Wie die entsprechende Einsteckkarte im Computer aussieht, ist in Abb. 11 zu sehen.

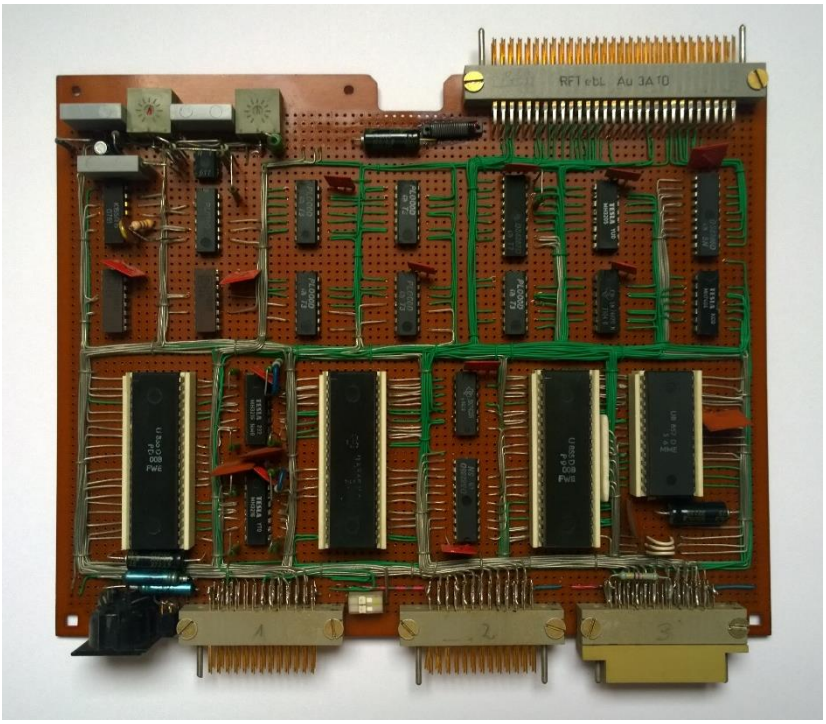


Abb. 11: Peripherie 1 für Tastatur und Tonbandgerät

Die Elektronik für die Tastatur macht nur einen kleinen Teil aus. Die Einsteckkarte ist mit zwei Z80-PIO-Chips (U855) zur Ansteuerung der Tastatur und weiterer Geräte, einem Z80-CTC-Baustein (U857) und einer Z80-SIO (U856) für die Speicherung von Daten auf einem Tonbandgerät bestückt. Alle physischen Geräte wurden von

der Software mit ihren Hardwareadressen angesprochen. Allerdings führte ich später „logische“ Geräte ein, um z.B. die üblicherweise durch eine Tastatur eingegebenen Zeichen auch von einem ganz anderen Gerät einlesen zu können. Solche Emulationen habe ich aber nie wirklich benutzt.

3.3 Archivieren: der ungewöhnliche Lagerplatz

Heutzutage speichert man neu erstellte Programme und Daten auf der Festplatte, einem USB-Stick oder auf einem Server und schaltet den PC aus. Davor verwendete man Disketten oder CDs/DVDs dafür. Nach dem Einschalten gehen die Programme und Daten den umgekehrten Weg. Lotus 2000 hatte keine dieser Speichermöglichkeiten. Es ist sehr lästig, ein langes Programm immer wieder einzutippen. Es auf einem EPROM zu speichern, ist auch nicht sehr bequem. Deshalb musste ein anderer nicht-flüchtiger Speicher her. Daten zeichnete man auch schon damals seit langem auf Magnetbändern auf. Damalige Heimcomputer verfügten auch über diese Möglichkeit. Bei Lotus 2000 habe ich deshalb diese Funktion ebenfalls eingebaut.

Die Elektronik ist auf der eben betrachteten Platine für die Tastatur zu finden. Siehe Abb. 11 im letzten Abschnitt. Die schwarze Buchse links unten ist eine sogenannte Diodenbuchse, mit der man damals Tonbandgeräte, Schallplattenspieler, Radios und Verstärker miteinander verband. Das Herzstück ist der Z80-SIO-Baustein (U856), der Daten in einen seriellen Datenstrom umwandeln und verschiedene serielle Übertragungsprotokolle unterstützt. Letztere sind notwendig, damit die Gegenstelle den Datenstrom richtig interpretieren kann. Lotus 2000 setzt den sogenannten Manchester-Code ein, bei der die Information durch die Phasenlage kodiert wird: Der Takt ist gleichbleibend. Bei jedem Takt (fallenden Taktflanke) schaut der Empfänger, in welche Richtung das Datensignal wechselt und erkennt daraus die übertragene Information. Und aus den Wechseln kann der Empfänger auch noch den Takt erkennen. Neben einer SIO braucht man etwas digitale und etwas analoge Elektronik und einen Timer-Schaltkreis (U857), der unterschiedliche Taktraten (Frequenzen) ausspuckt, je nachdem, wie er programmiert wird.

Alle auf der Platine realisierten Funktionen (Tastatureingabe, Tonbandaufnahme, weitere Ein- und Ausgaben) sind vollständig interrupt-gesteuert realisiert. D.h., die CPU startet gemäß dem ausgeführten Programm eine Aktion. Dann kann sie warten oder ein anderes Programm ausführen. Sobald der Peripheriebaustein (U855, U856 oder U857) fertig ist, wird die CPU mit einem „Interrupt“ darauf aufmerksam gemacht. (Dies erfolgt hardwarebasiert. Die Programmausführung verzweigt zwangsweise in ein programmtechnisch definierbares Unterprogramm, das den ursprünglichen Service fortführt.)

Bei dem Datensatz auf dem Tonband kann man vielleicht von Datei sprechen. Im Innern des Lotus 2000 Computers eher nicht. Der Bediener muss alles selbst festlegen und unter Kontrolle halten. Das Einlesen eines Datensatzes vom Band ist aber eher einfach und funktioniert absolut zuverlässig. Die Hardware horcht an der Leitung zum Tonbandgerät. Erkennt sie die „Eingangsmelodie“, weiß sie, es geht gleich los („sync“). Dann wird gelesen und die Daten werden in einen vom Bediener vorab festgelegten Speicherbereich geschrieben. Schließlich erklingt die „Fertig-Melodie“ und das Programm hat seinen Zweck erfüllt. OK. Das Ganze ist blockgesteuert, Prüfsummen erkennen mögliche Lesefehler und es gibt zahlreiche andere Fehlerzustände usw. Ganz so einfach ist es also nicht. Aber ich notierte mir stolz: „200h fehlerfrei!“. 512 Byte Programmcode sind nicht viel. Die komplexe Elektronik im Inneren des U856-SIO-Chips erledigt eben schon einiges selbst.

3.4 Drucken: die unbekannte Funktion

Mit dem Drucken ist eine Vervielfältigung verbunden bzw. möglich. Damit sich die „Konterrevolution“ ihrer in der DDR nicht bemächtigt, um den „real existierenden Sozialismus“ zu stürzen, waren Kopierer und Drucker streng kontrolliert, Mangelware und in privater Hand so gut wie nicht zu finden. [6] Aber es gab Ausnahmen: Mit Hilfe der „richtigen Währung“ (D-Mark) konnte mir meine Verwandtschaft auf der anderen Seite des Eisernen Vorhangs einen Drucker aus DDR-Produktion (Robotron 6320) schenken. Gerade während der Entwicklung wichtiger Teile von Lotus 2000 war es doch sehr angenehm, einiges ausdrucken zu können. Aber

[6] Ende 1989 habe ich während der Wende dann wirklich Material politischen Inhalts gedruckt. – Die Einsteckkarte war aber natürlich lange vor 1889 fertig, denn sie wurde u.a. für den Anschluss der alphanumerischen Tastatur benötigt.

dazu musste Lotus 2000 den Drucker ansteuern können. Die Karte, die das bewerkstelligt, ist in Abb. 12 zu sehen.

Die Einsteckkarte ist mit einem Z80-PIO-Chips (U855), einem Z80-CTC-Baustein (U857) und einer Z80-SIO (U856) bestückt bzw. letztere ist nicht eingesteckt. Für die Druckeransteuerung (Centronics-Schnittstelle, mittlerer Steckverbinder) wird der U855 verwendet, der zwei 8-Bit-Ports hat. Die Realisierung war sehr einfach, weil eine 8120-Platine von Robotron verwendet werden konnte, die aber modifiziert wurde. Nicht sehr einfach war dagegen die Software, da sie die Druckerbefehle (EPSON) beherrschen bzw. umsetzen musste.

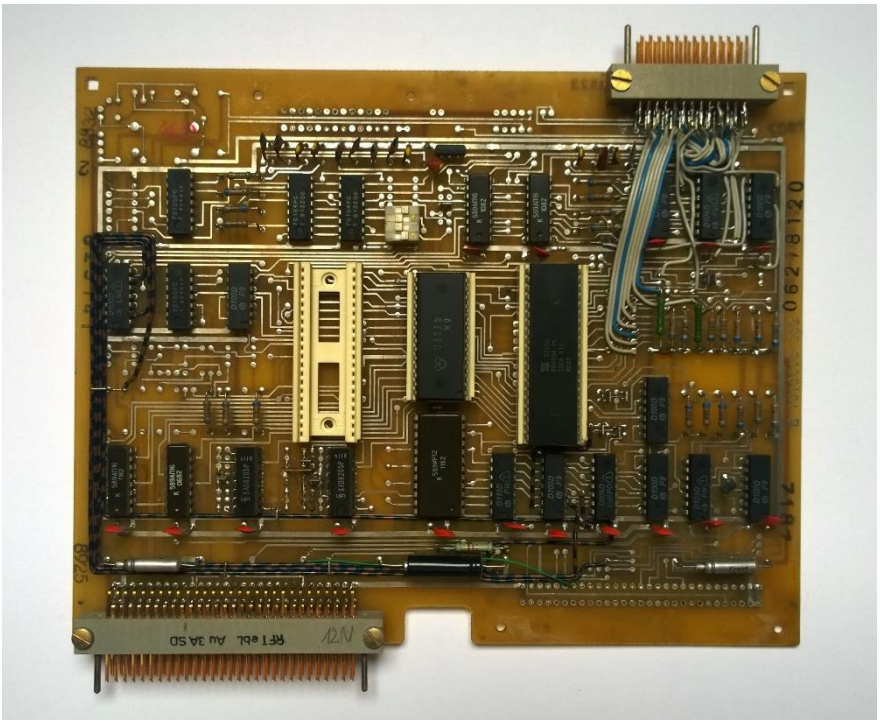


Abb. 12: Peripherie 2 (Drucker)

4 Speicherhunger: mehr als gestillt

4.1 Mem1 und Mem2: die Basis

Wenn die Stromversorgung nach dem Einschalten stabil ist, wird der CPU der Startschuss gegeben, jetzt mit der Programmausführung bei Adresse 0000h zu beginnen. Deshalb ist es gut, wenn dort ein Programm steht - und zwar das richtige. Ich hatte großes Glück, dass ich eine Original-Leiterkarte Robotron-7030 bekommen konnte, die ich mit 6 Kilobyte EPROM-Speicher (2708, nichtflüchtig) und 2 Kilobyte RAM-Speicher (flüchtig) bestücken konnte. Eine solche Speicherkarte braucht man aus den oben genannten Gründen sehr früh bzw. ganz zu Anfang. Ein Computer soll ja schließlich Programme ausführen und Daten verarbeiten. Abb. 13 zeigt die Einsteckkarte mit 8 Kilobyte!

Sechs Kilobyte mit Programmen zu füllen, die man mit einfachen Assembler-Befehlen (kein Hochsprache!) entwickelt und dann per Hand in Maschinencode (Zahlen) übersetzt, um sie dann einzutippen und auf einen EPROM zu brennen, ist nicht sehr bequem. Aber ich bin stolz auf das darin enthaltene „Betriebssystem“, das aus vielen, vielen Routinen besteht, die letztlich die Funktionen meines Lotus 2000 ausmachen.

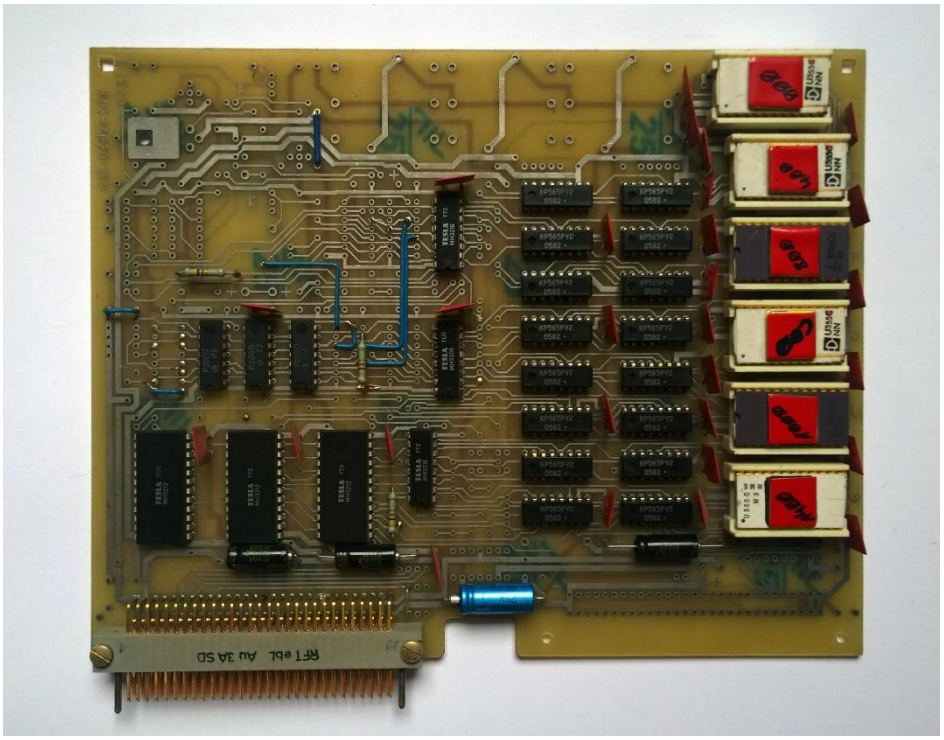


Abb. 13: Speicher, die ersten 8 Kilobyte

Eine dieser nützlichen Routinen belegt vielleicht 50 Speicherplätze. Dann passen in 6 Kilobyte etwa 120 davon. Mit Hilfe meines Druckers konnte ich das ja dann auch ansehen. Wie ein solcher Plot aussieht, zeigt Abb. 14. Ich bin fast sicher, dass da viele Unterprogramme aufgerufen werden. Sonst wäre der Code viel länger. Das erste Programm ist das Programm „List“, welches genau den Auszug eines Speichers erzeugt, der in dem Bild dargestellt ist. (Allerdings ist dieser hier auf den Drucker „umgelenkt“.)

```
42D0 18E9 A73E 20C9 EDED FDFD 442E 4C49 5354 *.0>  ØØ²²D.LIST*
42E0 0DCD D632 D8CD 4B10 3E14 A7C9 CDC4 04D8 *.-2+K.>.  *
42F0 7BE6 F05F CD79 15FE 0330 043A 0A18 4F0D *{µ≡_-y§▪♥0♦:..O.*.



 Adr. 0 1  2 3  4 5  6 7  8 9  A B  C D  E F  0123456789ABCDEF
4300 C5D5 D454 10CD A411 2121 3206 0DCD A409 *+ LT.-ñ.!!2♠.-ñ.*
4310 E1C1 D8CD 8333 0D20 FACD A411 3E0D C300 *ß++83. -ñ.>.+>*
4320 1C20 4164 722E 2030 2031 2020 3220 3320 *. Adr. 0 1  2 3 *
4330 2034 2035 2020 3620 3720 2038 2039 2020 * 4 5  6 7  8 9  *
4340 4120 4220 2043 2044 2020 4520 4620 2030 *A B  C D  E F  0*
4350 3132 3334 3536 3738 3941 4243 4445 4620 *123456789ABCDEF *
4360 200D FFFF FFFF EDED FDFD 442E 5245 4144 * .     ØØ²²D.READ*
4370 0DCD C404 D8EB CD83 33D8 3A0C 1CA7 3E14 *.-♦+s-83+:..O>.*
4380 C8AF C9A7 3E0D CD00 1CE5 7CCD C10F 7DCD *L» O>.->.σ!-⊥.}-*
4390 C10F 3E20 CD00 1C06 107E CDC1 0F23 CB40 *⊥.>  ->.♠.~-⊥.#+@*
43A0 3E20 C400 1C10 F2E1 3E2A CD00 1C06 107E *>  ->..≥ß>*->.♠.~*
43B0 23CD 4F07 CD00 1C10 F63E 2AC3 001C FFFF *#-O.->..÷>*+>.   *
```

Abb. 14: Speicherauszug mit den Betriebssystem-Befehlen „List“ und „Read“

Während ich bei „Mem1“ eine Original-Leiterkarte ergattern konnte, war es für die nächste Karte nicht so einfach. Doch Speicher wurde benötigt, denn mein Betriebssystem wuchs und wuchs. Immerhin konnte ich in diesem Fall Fotos der Leiterbahnen der 7030-Karte des K1520-Systems von Robotron bekommen, dessen

grundlegende Architektur meinem Lotus 2000 zugrunde lag. Der Besitz der Karte wie auch der der Filme besagte nicht, dass ich wusste, was da drauf war, welche Chips wohin mussten, wie das Ganze funktionierte und wie es für meine Belange angepasst werden musste. Das musste ich durch Reverse-Engineering (Knobeln) herausfinden. Abb. 15 zeigt Mem2 mit den nächsten 8 KByte Speicherplatz bzw. Adressraum. Man erkennt, dass nicht alles bestückt ist und an den blauen Leitungen, dass Korrekturen durchgeführt wurden. Trotzdem: ein Heimspiel. Aber die Software: Die kam nicht von „alleine“.

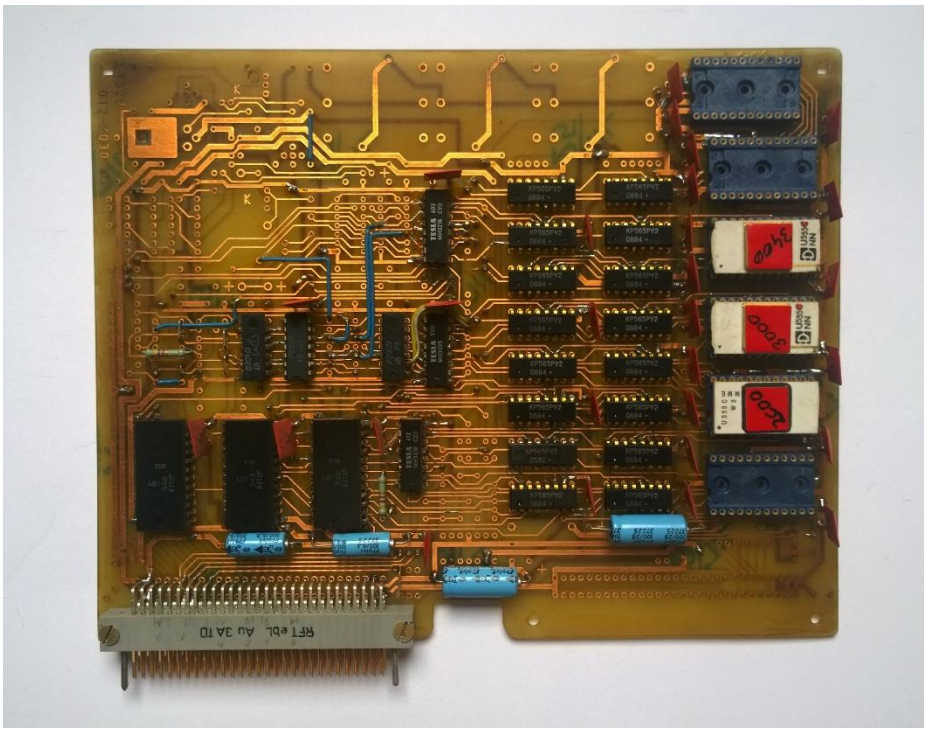


Abb. 15: Speicher, die nächsten/zweiten 8 Kilobyte

4.2 Mem3 mit 16K EPROM (irgendwann)

Lotus 2000 hat jetzt 12 Kilobyte: 4 Kilobyte flüchtigen, temporären Speicher (RAM) und 9 Kilobyte genutzten persistenten Speicher (EPROM). Das ist schon recht ordentlich für ein System dieser Klasse. Die CPU (Z80 bzw. U880) hat einen 16-Bit-Adressbus, kann also 64 Kilobyte adressieren. Ein Viertel davon, 16 Kilobyte, sind durch Mem1 und Mem2 bereits belegt, da die leeren Stellen auch zählen.

Jetzt kann mit EPROM (nicht-flüchtigem Speicher) weiter aufgerüstet werden. Irgendwo ist eine 7040-Platine übriggeblieben. Sie wird angepasst und bestückt ähnlich wie eben beschrieben. Abb. 16 zeigt die fertige Einsteckkarte. Der aktuelle „Füllstand“ beträgt 50% (32 KByte) des Adressraumes (64 KByte). Die Endadresse ist 7FFFh. Die rekonstruierten Stromlauf- und Bestückungspläne sind mit ihren Anpassungen Bestandteil der umfangreicher werdenden Dokumentation des Lotus 2000.

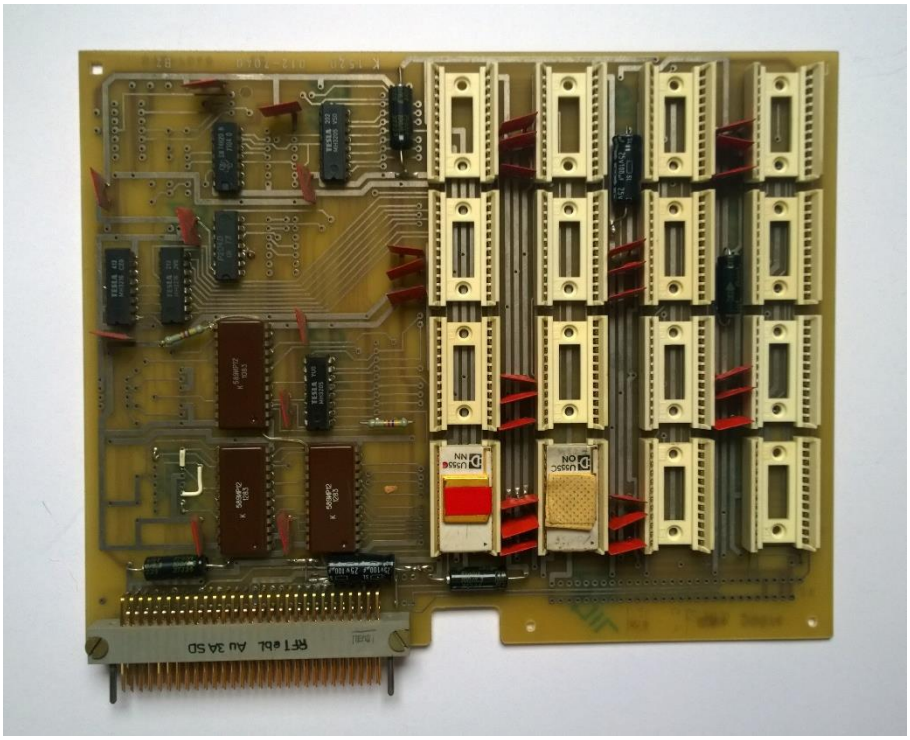


Abb. 16: Mem3 mit 16 Kilobyte EPROM, nur 2 Kilobyte sind hier bestückt

4.3 Mem 4: es wird dynamisch

Die Angst, der Speicher könnte ausgehen, war damals viel realer, und er hatte Gründe. Schön, dass ich Leiterkarten bekam, die mit russischen Speicherchips zu bestücken waren. Allerdings war sie für ein gänzlich anderes Steckersystem ausgelegt, und ich hatte keine Ahnung, wie diese „dynamischen Speicher“ funktionierten. Das oben erwähnte Reverse-Engineering (Knobeln) begann. Ein Zwischenergebnis zeigt Abb. 17.

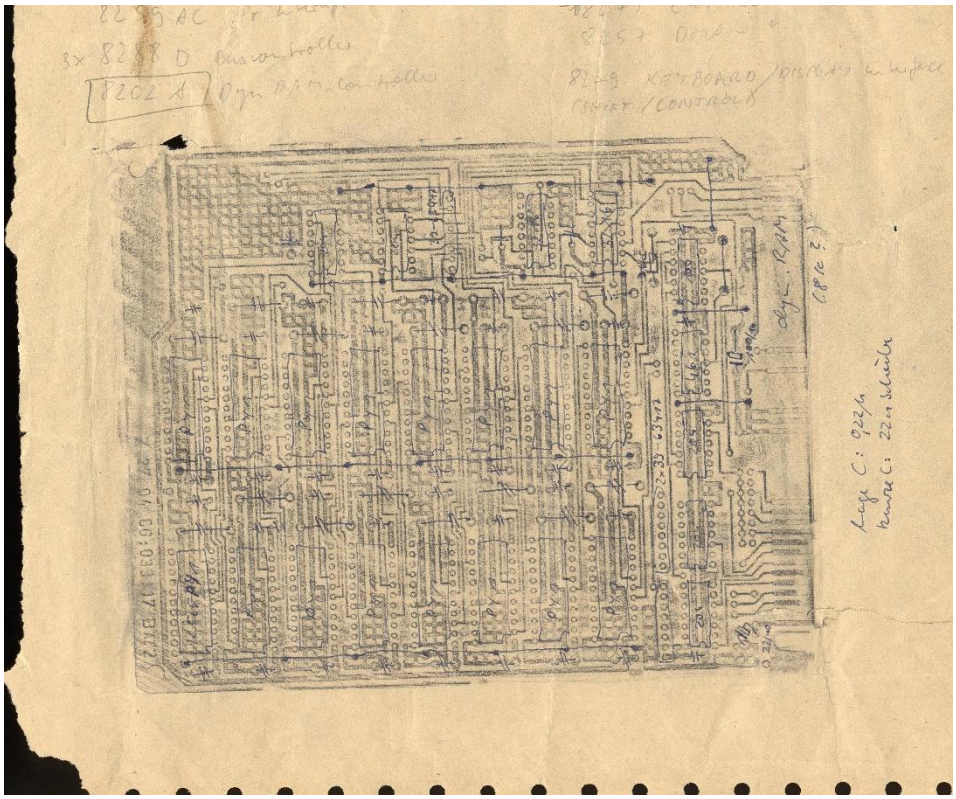


Abb. 17: Karte für… 8K dynamischen RAM?

Die Analyse war richtig und die Speicherchips ließen sich auch beschaffen. Allerdings musste das Bussystem auf das von Lotus 2000 angepasst werden. Heraus kam eine Einsteckkarte, die zwei weitere Platinen mit zwei Steckverbindern huckepack trägt und beherbergt. Die Draufsicht zeigt Abb. 18. Es liegen drei Leiterkarten übereinander. Eine vierte kleinere liegt daneben.

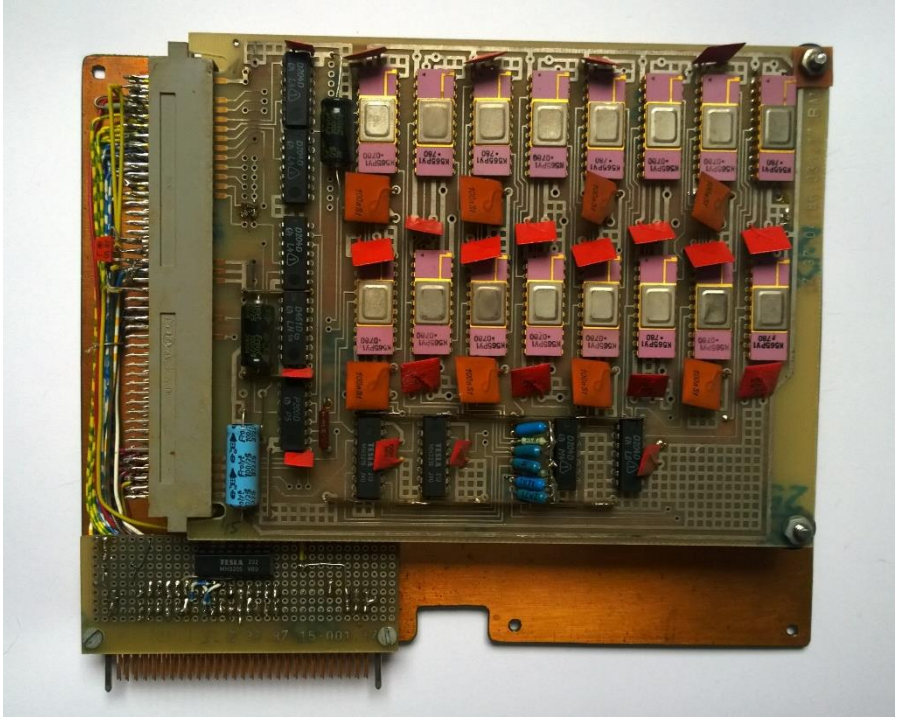


Abb. 18: Zweimal 8 Kilobyte RAM

Das Besondere und die eigentliche Herausforderung bestand darin, dass dieser Speicher ein „dynamischer“ Speicher war, der durch die CPU permanent aufgefrischt werden musste. Da verhielt sich der bisher verbaute RAM- und EPROM-Speicher viel gütlicher. Aber der Z80-Prozessor (U880-CPU) kann DRAMs (dynamische Speicher) auffrischen. Die Elektronik vom Lotus 2000 spielt auch mit. Das geht so: Die CPU liest regelmäßig Programminstruktionen (M1-Zyklus). Das Lesen dauert zwei Taktzyklen. Danach dekodiert die CPU den Befehl und führt 1-Byte-Instruktionen direkt aus. Zur gleichen Zeit adressiert sie die nächsten aufzufrischenden Speicherstellen (7 Bit-Adresse) und die DRAMs reagieren entsprechend. Aber das Timing und die Elektronik hat mir schon einiges Kopfzerbrechen beschert.

5 Programme

Bisher stand die Elektronik im Vordergrund. Schließlich stellt sie den sichtbaren Teil des Computers dar. Bei alledem darf man nicht vergessen, dass das Programmieren des „Betriebssystems“ mit den vielen, vielen Routinen eine sehr zeitraubende Angelegenheit war. Aber sie gehörte zu diesem Hobby natürlich dazu. Alle Routinen oder Programme waren hierarchisch aufgebaut. Es gab sehr einfache grundlegende, die wiederum von anderen, komplexeren verwendet wurden. Am Ende stehen die Befehle, die der Bediener per Textbefehl zusammen mit Parametern per Tastatur eingibt und auf dem Bildschirm sieht. Die verarbeitende Routine sucht die Befehle im Speicher des Computers und bringt sie zur Ausführung. Alle Eingaben werden vollständig geprüft und alle Fehlermöglichkeiten (auch bei der Ausführung) erkannt und angezeigt. Da die höherwertigen Befehle im Speicher gesucht werden, kann ihr Vorrat leicht ergänzt werden, und es besteht auch die Möglichkeit, alle verfügbaren Befehle anzuzeigen. Da die Befehle bzw. die dahinterstehenden Programme ihre Kenner (Startsignaturen) im Speicher haben, kann man schon fast von Dateien (Files) sprechen. Das gleiche konnte auch für Daten Anwendung finden, etwa um einen Text im Editor aufzurufen.

Abb. 19 zeigt ein Beispiel, wie die Software des Lotus 2000 entstand und dokumentiert wurde. Die Programme wurden im Assembler-Code entwickelt. D.h., es konnten nur die Befehle verwendet werden, die die CPU direkt ausführen konnte. Der Befehlssatz des Z80 war, an heutigen Maßstäben gemessen, äußerst bescheiden. Er hat 2x13 Byte Register (je nach Zählweise) zur Nutzung als interner Speicher und Operationen sowie für die Adres-

sierung. Speicher kann direkt und indirekt (über Register) adressiert werden. Es gibt eine Fülle von Ladebefehlen, grundlegenden Operationen, Block-Suche, Block-Transfers, Verschiebungen und Rotationen, Bitmanipulationen, Ausgaben und Befehlen zur Programmverzweigung. Ein toller Prozessor! Viele damalige Heimcomputer und selbst die moderneren graphischen Taschenrechner der Serie vom TI-81 bis TI-86 der Firma Texas Instruments nutzen diesen Prozessor ebenfalls.

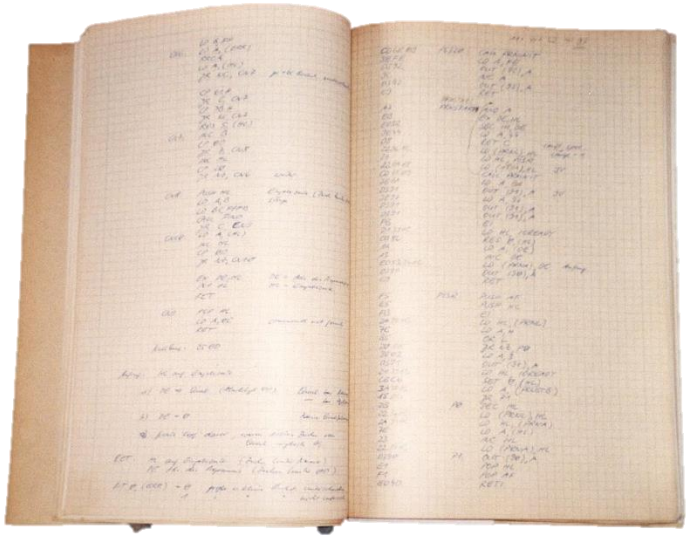


Abb. 19: Ein dickes Buch mit Programmen (Assembler)

Damit die CPU die Programmbefehle versteht, müssen sie von den eingängigen Mnemoniks (Abkürzungen) in Maschinencode (Zahlenwerte) umgewandelt werden. Die Übersetzung habe ich wie in Abb. 19 zu erkennen ist, meistens von Hand gemacht. Auf der rechten Seite stehen rechts die Menemonics und am linken Rand die Codes. Direkte Adressen wurden später eingesetzt. In professionellen Systemen erledigt diese Umwandlung ein Programm. Nur ab und zu hatte ich die Gelegenheit, ein paar Routinen damit zu erstellen. In jedem Fall bestand das Ergebnis aus Zahlen, die im Speicher abgelegt wurden. Ein Speicherauszug (Dump) ist in Abb. 20 zu sehen. Dieser wurde verwendet, um die endgültigen, absoluten Adressen ablesen und eintragen zu können.

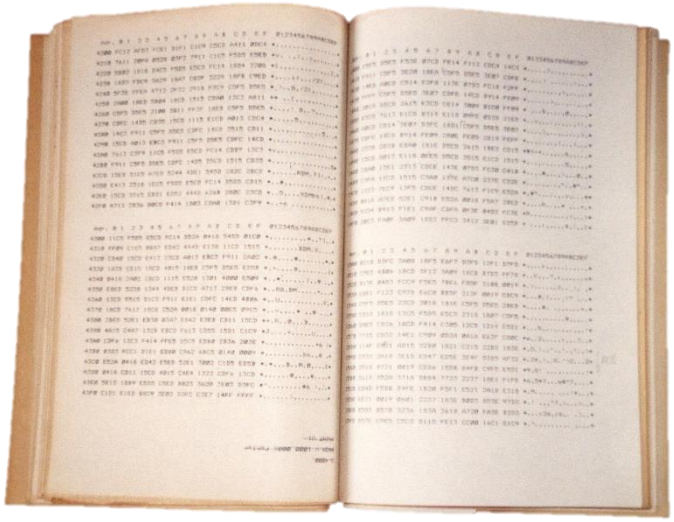


Abb. 20: Speicherauszug (Dump)

Ohne den Drucker wäre ich in dem Zahlensalat wohl hoffnungslos untergegangen. Es war ohnehin schon eine Herausforderung, eine eigenes „Betriebssystem“ zu entwickeln, ohne auf eine professionelle Software-Entwicklungsumgebung mit Editor (zum Aufschreiben des Programms) und Assembler (für die Übersetzung in Maschinencode) zurückgreifen zu können.

6 Weitere Pläne

Natürlich sollte Lotus 2000 noch viel größer und leistungsfähiger werden. „Einem Ingenieur ist nichts zu schwör.“ Pläne existieren für diverse Speichererweiterungen. Abb. 21 zeigt, dass ich dafür bereits Chips beschafft hatte. Sogar an einen Direct-Memory-Access (DMA) hatte ich gedacht, der Speicherzugriffe ohne Programm und CPU erlaubt und daher ungleich schneller ist.

Abb. 21: Vorräte an Speicher und Z80-Bausteinen

Auch an anderen Chips bestand kein Mangel (Abb. 22). Da in der DDR fast alles knapp war bzw. man nicht wissen konnte, ob das benötigte Material dann verfügbar war, wenn man es braucht, musste der Bastler Vorräte anlegen. Wahrscheinlich hat das die Knappheit verstärkt.

Auf eine Besonderheit muss an dieser hingewiesen werden. Damals gab es diverse Zeitschriften, die Schaltpläne und Daten von Schaltkreisen veröffentlichten oder komplette Bauanleitungen abdruckten. In der DDR waren einige davon „halb-professionell“, d.h., sie verteilten Informationen an die Industrie (und die Profis), konnten aber auch von Bastlern gelesen und genutzt werden. Die Elektronik war auch noch nicht so kompliziert wie heute, so dass beide Welten Veröffentlichungen noch gemeinsam nutzen konnten. Das hat auch die Entwicklung von Lotus 2000 unterstützt. Viele Informationen über das K1520-System von Robotron waren in der „public domain“, so dass sie für die Entwicklung genutzt werden konnten. Und natürlich gab es eine Fülle von Büchern. Viele habe ich noch heute.

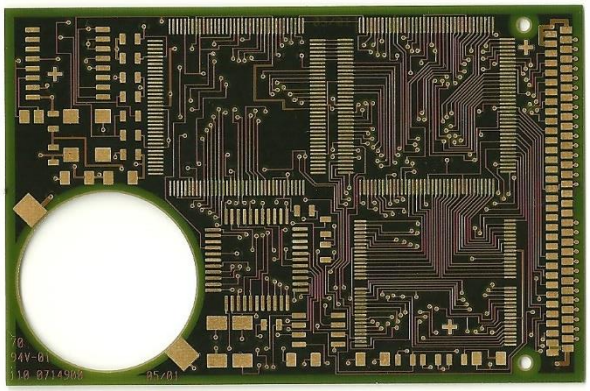
Abb. 22: Vorräte an einfachen Logikbausteinen

Eigentlich ist Lotus 2000 nicht fertig geworden, wobei ich allerdings niemals definiert hatte, wie der fertige Zustand auszusehen hätte. Die Entwicklung der Hardware und die Programmierung war ein schönes Hobby und zugleich ein ziemlich zeitraubendes. Der praktische Wert des Computers ist fraglich. Man konnte allenfalls

Texte schreiben. Mit Textverarbeitung im heutigen Sinne hatte das aber nichts zu tun. Es gab keine Spiele und an eine Internetverbindung war Mitte der 1980er Jahre nicht zu denken.

Lotus 2000 zu bauen war eine Herausforderung für mich als angehenden Ingenieur. Der Bau hatte aber einen schönen Nebeneffekt, der sich erst 1991 zeigte: Das Projekt hat meinen ersten Arbeitgeber überzeugt und mir den Weg in die Industrie geebnet. Dort konnte ich mich „austoben“ bei der Entwicklung von Hightech-Elektronik, ganz so wie ich mir es vorgestellt hatte:

Mein nächster Computer war nur so groß wie eine Scheckkarte. Und die CPU war nicht so leistungsfähig. Doch die Programme und ein paar Daten beanspruchten 256 KByte.
Er wurde Anfang 1993 auf der CeBIT der Weltöffentlichkeit vorgestellt.
Die 7 Chips und der Steckverbinder hatten insgesamt 488 Pins.
Zwei der Chips mit je 144 Pins waren selbst entwickelt (ASICs)
und passten nur für diesen Minicomputer…



Das System machte manche Notebooks
zu den sichersten PCs der Welt.

Über den Erbauer

Spätestens nach Erreichen des Teenager-Alters habe ich angefangen, elektronische Geräte zu „basteln“. Anfangs half mein Vater mit Bauplänen, oder ich nutzte Bausätze. Doch schon bald stand die Entwicklung eigener Geräte im Vordergrund. Schon in der 7. Klasse war mir klar, dass ich Elektrotechnik studieren und dann in der Forschung und Entwicklung (F&E) arbeiten wollte. Allerdings hat mich auch die Physik interessiert, so dass ich beides studierte und danach auf dem Gebiet der Theoretischen Physik promovierte. Die Entwicklung und der Bau von Lotus 2000 fällt in die Zeit meines Studiums Mitte der 1980er Jahre, doch auch während der Promotion habe ich noch daran gearbeitet. Als ich mein Hobby zum Beruf machte, war damit Schluss. Später habe ich auch größere IT-Systeme entwickelt, als Berater und Gutachter für IT-Sicherheit gearbeitet. Heute entwickle ich für einen großen IT-Konzern Methoden, Prozesse und Standards zur Absicherung einer großtechnischen, weit verteilten und komplexen IT-Produktion.

Prof. Dr. Eberhard von Faber