Bildverarbeitungssystem Robotron A 6472

Wir waren die Schmiede, die den Kessel für den Cugnotschen Dampfwagen auf einem Rennfeuer geschmiedet haben. Aus alter FuE. Rolf Böhm, 26.09.2022.

Wie alles begann

Bei meinem Studium, 1979-1984 an der TU Dresden, gab es zwar durchaus schon etwas „EDV“ aber das war noch im Lochstreifenzeitalter, in welchem 5 Meter Lochband eine große Menge Daten waren. 1983 gab es eine Exkursion, die mein ganzes Leben verändern sollte: Nach Berlin in das Zentralinstitut für Kybernetik und Informationsprozesse (ZKI) bei der Akademie der Wissenschaften der DDR. Dort hat uns Dr. Wilhelmi einen phantastischen Computer vorgestellt: Das Bildverarbeitungssystem KTS1M oder A6472. Unzählige Leiterplatten waren durch breite Flachbandkabel miteinander verbunden, was gewaltige Datenströme erahnen ließ. Wir standen vor einem „Displayprozessor“ mit einer uns als nicht anders als *wahnsinnig* erscheinenden Verarbeitungsgeschwindigkeit von 12½ Megabyte pro Sekunde. Maßeinheiten mit „Mega“ hat es damals in der EDV noch überhaupt nicht gegeben. Die Kabel mündeten in einem *Farbmonitor,* der Bilder anzeigte – *Computerbilder*. Heute selbstverständlich, aber damals gab es ja überhaupt erst ein paar Jahre Farbfernseher. Das erste Mal in meinem Leben hörte ich etwas von Pixeln. ***Digitale Bildverarbeitung.*** Was für ein Wort. 1983 in der DDR. Damit habe ich mich unheilbar infiziert.

Als dann Robotron 1984 in Berlin[1] mit der Produktion dieser Systeme begann, suchten die einen Kartografen als Softwareentwickler. Keine Frage, da musste ich anfangen, Mohrenstraße 62. Dass ich gar nicht programmieren konnte, war kein Problem, denn wer konnte damals schon programmieren? Das hat man nebenbei gelernt, mit ein paar Lehrgängen, hilfreichen Kollegen und ordentlich Rechenzeit.

Das Bildverarbeitungssystem Robotron A6472

Die KTS1M hieß im Robotron-Jargon ***Bildverarbeitungssystem Robotron A6472.*** Was ist das für ein Computer gewesen? Zunächst ein Steuerrechner K1630. Der K1630 war ein gewöhnlicher Rechner mit damals beachtlichen 16-Bit-Verarbeitungsbreite, ansonsten gar nicht viel anders als ein heutiger PC, etwas größer. Hauptspeicher, Prozessor, Terminal, Festplatte, Drucker. Dazu kamen noch ein bis zwei 1/2-Zoll-Magnetbandeinschübe.

Lochstreifen war bereits ein Auslaufmodell. Ein einziges Mal habe ich ein kleines Bild 256×256 auf Lochstreifen ausgestanzt, das war mit 200 Metern Länge ein ganzer Lochbandwickel. Magnetband ging aber wunderbar. Auf ein Magnetband passten ungefähr 10 Megabyte. Ich habe diese urigen und vor allem super zuverlässigen 1/2-Zoll-Magnetbänder über alles geliebt.

[1] Das Kombinat Robotron wird fast immer mit dem Hauptstandort Dresden identifiziert, doch auch in Berlin gab es einen größeren Ableger, den Robotron-Vertrieb-Berlin (RVB) mit 5.000 Mitarbeitern.

Der K1630 wurde nun mit Bildverarbeitungs-Spezialperipherie ausgestattet zum Bildverarbeitungssystem A6472. Kernstück des Bildverarbeitungssystems war der ***Displayprozessor*** oder DIP, den wir bereits am ZKI als Kabel- und Platinensalat gesehen hatten und der nun bei Robotron in Serie ging, Robotron-Baugruppenchiffre DIP K2067. Dazu kam ein ***Bildspeicher*** BSP K3567, oft noch eine zweite weitgehend bauartgleiche Einheit, der sog. „Gemeinsame Speicher" GSP K3667. Eine ***Grafiksteuerung*** GST K7067 speicherte ein weiters Bild für ein „Grafikoverlay". Jede dieser Einheiten hat einen oder mehrere Schrankeinschübe ausgefüllt. Das System hatte eine frühe Maus, die ***Rollkugeleinheit*** RKE K7767, auch Trackball genannt. Schließlich war der ein ***Farbmonitor*** K7226 das sichtbare Attribut eines Bildverarbeitungssystems.

Weiterhin gab es einen riesengroßen hochpräzisen Scanner, das Film-Eingabe-Ausgabe-Gerät FEAG 200 des VEB Carl Zeiss Jena. Viele Systeme hatten auch eine Fernsehkamera zur Bildaufnahme.

Es gab 3 Varianten. Die ***A6472***[2] war unser „Brotpferd" und ungefähr schrankwandgroß. Ein kleineres System (sozusagen unser „Fohlen") war die lediglich schreibtischgroße ***A6471*** ohne Displayprozessor. Hier wurde lediglich ein einzelnes Bild in der Grafiksteuerung gespeichert und auf Farbmonitor angezeigt wurde. Auch nicht schlecht, aber die gesamte Bildverarbeitung musste hier in dem langsamen Hintergrundrechner K1630 erfolgen. Und dann gab es noch unser „Paradepferd" ***A6473,*** quasi eine Quadriga, denn in der A6473 waren vier 72er Systeme parallelgeschaltet. Die A6473 mit ihren 4 Displayprozessoren war insbesondere für sowjetische Raumfahrtzentren vorgesehen, aber auch die Geophysiker in Leipzig hatten eine A6473.

Digitale Bildverarbeitung war in den frühen 1980er Jahren etwa völlig Exotisches. Gewöhnliche Rechner schafften einige 100.000 Operationen pro Sekunde. Der Hauptspeicher des K1630 war 256 Kilobyte, also 1/4 Megabyte groß. Demgegenüber waren digitale Bilder wahrhaft gigantisch. So reizvoll es war, Bilder digital zu bearbeiten – ein einziges „512er" Bild war ja bereits 1/4 Megabyte groß. Damit wäre der ganze Hauptspeicher des K1630 bereits vollständig belegt gewesen. Wohin also mit diesen Bilddaten?

Also hat man Bildspeicher als zusätzliche externe Geräteeinheiten hinzugefügt. Die Halbleiterchips des K3567/K3667 konnten zusammen die als gigantisch geltende Datenmenge von zwei Megabyte speichern, pro Bild 512×512 Pixel zu je einem Byte. Die vier Bilder der K3567 wurden im Betriebssystemjargon als Geräteeinheiten RM0: bis RM3: bezeichnet, „RM" wie ***refresh memory,*** Bildwiederholspeicher. Die zweite Einheit K3667 speicherte 4 weitere Bilder, RM4: bis RM7:.

Die Grafiksteuerung hieß Graphics Device GD0:, sie speicherte ein weiteres Bild. Die Maus war der Trackball TB0:. Der Displayprozessor wurde im System als Display Device DD0: geführt, was man aber kaum bemerkte, denn der wurde, wie wir noch sehen werden, mit Spezialwerkzeugen programmiert.

---

[2] Computer hatten bei Robotron unterschiedliche Geschlechter. Es hieß *der* R300 und *der* K1630, aber *die* BESM-6, *die* IBM360, *die* 1040 (EC1040) und *die* 1055. Auch sagten wir stets *die* SM-4.

Der Farbmonitor war aus Softwaresicht kein externes Gerät; er war lediglich über ein analoges Monitorkabel an den DIP angeschlossen und damit für die Software so wenig existent, wie die Neonröhren, die die Rechnerräume beleuchteten.

Der Displayprozessor K2067

Die Bilder mussten ja nicht nur gespeichert, sondern auch verarbeitet werden. Wenn man jedes Pixel mit ein paar Befehlen bearbeitet, kommen schnell ein paar Millionen Operationen zusammen. Das dauert. Komplexere Algorithmen brauchten auf dem Steuerrechner K1630 schon einmal eine Stunde. Nun lieferte die Fernsehkamera aber 25 Bilder pro Sekunde, das war die berühmte Videonorm. Ein Bild war 512×512 = ***262.144 Byte*** groß, Byterate 6,25 Megabyte pro Sekunde. Pro Pixel standen gerade einmal 80 Nanosekunden[3] Verarbeitungszeit zur Verfügung. Anschließend war sofort das nächste Pixel zu bearbeiten.

Darum wurde also der Displayprozessor, kurz DIP, benötigt. Der DIP war ein ***Parallel-Pipeline-Prozessor,*** was zunächst heißt, dass er völlig anders funktionierte, als ein gewöhnlicher Computer. Ein gewöhnlicher Rechner arbeitet „sequentiell“, d. h. er arbeitet einzelne „Maschinenbefehle“ nacheinander ab. Er verarbeitet etwa drei Zahlen, z. B. 2+3=5. Der nächste Befehl kommt immer erst dann an der Reihe, wenn das Ergebnis 5 wieder in einer Speicherzelle abgespeichert worden ist. — „Parallel“ heißt nun, dass der DIP mit einem einzigen Befehl *alle* Pixel eines Bildes bearbeitet, 262.144 Byte mit einer einzigen Anweisung. — „Pipeline“ bedeutet, dass mehrere Operationen wie in einer Erdölleitung hintereinander aufgereiht ausgeführt werden. Dabei wird ein „Datenstrom“ bearbeitet, der die Pipeline quasi durchfließt. Und es können sogar mehrere „Erdölleitungen“ parallel arbeiten, was es noch einmal schneller macht. Zwischenergebnisse werden zwischenzeitlich nicht umständlich „über Bus“ abgespeichert, sondern bleiben immer „in der Pipeline drin“.

Das erfordert freilich eine völlig andere Art des Programmierens. Man schreibt nicht „sequentielle Anweisungen“ in einem Programm auf, in dem diese dann hintereinander abgearbeitet werden. Es funktioniert völlig anders. Zunächst ist ***Videolesen*** ein fundamentaler Begriff. Ein Bildspeicher ist zunächst ein „normaler Speicher“ der vom Steuerrechner K1630 gelesen/geschrieben wird. Mit dem Videolesen gibt es nun ein zweites, völlig anderes Zugriffsregime. Hierbei leitet der Bildspeicher seinen gesamten Inhalt, 512×512 Pixel, 262.144 Byte in der 1/25. Sekunde eines Fernsehbildes in den Display-prozessor hinein. Einzelne Speicherplätze spielen keine Rolle mehr. Zuvor sind im DIP Steuerdaten in 12 Maschinenregister geladen worden. Das waren die sagenumwobenen DIP-Steuerregister, die einen bestimmten „DIP-Zustand“ beschreiben. Weiterhin bearbeitet der DIP Bilddaten mit Hilfe von Lookup-Tabellen, sog. „Funktionsformern“. Das sind Tabellen mit exemplarisch 256 Speicherplätzen, die vor der Abarbeitung mit bestimmten Zahlenwerten geladen werden. Mit diesen können Bytewerte eines einlaufenden Quell-Datenstromes in

---

[3] 1 Pixel in 80 Nanosekunden – das sind die Wilhelmischen 12½ Megabyte/Sekunde. Wegen der Bild- und Zeilenaustastlücken werden praktisch nur 6¼ Megabyte pro Sekunde erreicht. Indem 3 Farbkanalprozessoren zu je etwa 32 Einzelbaugruppen parallel- und hintereinandergeschaltet waren, wurde eine Verarbeitungs-geschwindigkeit von 600 Millionen Operationen pro Sekunde erreicht. Damit war das System ungefähr 1000× schneller, als gewöhnliche Rechner dieser Zeit.

beliebig andere Ziel-Zahlenwerte umgeformt werden. Nachdem die Bilddaten den DIP passiert haben, werden sie wieder in einem Bildspeicher abgespeichert. Dieser Vorgang wird ***Videoschreiben*** genannt.

Der Datenstrom wird immer auch auf dem Farbmonitor angezeigt. So erfolgt die Datenverarbeitung nicht länger „stumm und blind“ in den Tiefen des Rechners, sondern man sieht sofort, was gerechnet wurde. Bildverarbeitung im wahrsten Sinne des Wortes.

Die Baugruppen des DIP

Was passiert da nun genau? Der DIP ist eine Maschine mit einer sehr schönen Architektur. Es gibt (etwas vereinfacht) die folgenden Baugruppen[4]:

1. ***Bildspeicher-Videolesen.*** Die Bildspeicher geben die Daten im Videotakt ab. Dabei kann eine Bildversetzung X, Y, sowie eine Vergrößerung 1 … 16 eingestellt werden. Auch lassen sich statt des Vollbildes nur einzelne Quadranten oder Halbbilder aufschalten[5].

2. Ein ***Bildmultiplexer*** verteilt die 8 Bildspeicher auf 4 DIP-Eingangsdatenströme 1 bis 4.

3. ***Verzögerungsregister*** können Eingangsdatenstrom 1 um 1 Pixel versetzt in Strom 2 umleiten und um 2 Pixel versetzt in Strom 3. Strom 3 kann um 1 Pixel versetzt in Strom 4 umgeleitet werden. Das wird für Pixelnachbarschaftsoperationen und Filter eine große Hilfe sein.

4. Es folgen 4 ***Eingangsfunktionsformer*** (EFF). Dies sind 4 Lookup-Tabellen, die den einlaufenden Bytewerten 0 … 255 Zielwerte 0 … 1023 zuweisen. Damit genauigkeitserhöhende Erweiterung der Bittiefe von 8 auf 10.

5. Nun folgen in einer ***ersten ALU-Stufe*** 2 arithmetisch-logische Einheiten (Arithmetic Logic Units). Eine ALU 1 verknüpft Datenstrom 1 mit 2, eine ALU 2 Datenstrom 3 mit 4. Die Verknüpfung erfolgt typischerweise mit Plus, Minus, AND oder OR. Es sind aber auch exotische Operatoren wie NAND oder XOR möglich[6]. Bitbreite nun 11 Bit.

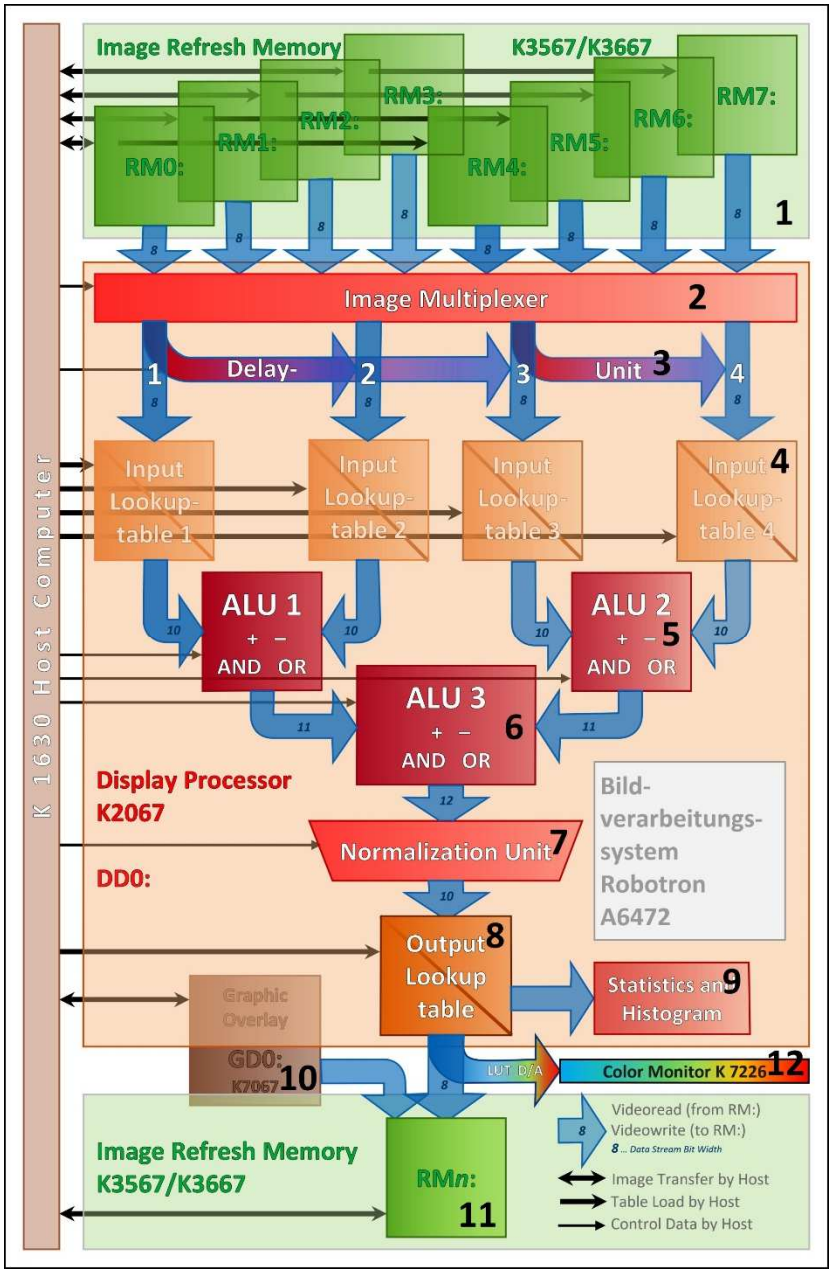
6. In einer ***zweiten ALU-Stufe*** verknüpft ALU 3 Datenstrom (1 und 2) mit (3 und 4). Bitbreite nun 12 Bit.

---

[4] Genau genommen gibt es das Ganze sogar dreimal, nämlich je einmal für jeden der Farbkanäle Rot, Grün und Blau, wovon wir hier aber nur einen Kanal darstellen. Insgesamt gibt es im DIP also 12 Eingangsdatenströme, 12 Eingangsfunktionsformer, 9 ALU, 3 Normierungseinheiten, 3 Ausgangsfunktionsformer, 3 Korrekturfunktionformer.

[5] So kann man statt mit 8 Bildern der Größe 512×512 auch mit bis zu 32 kleineren Bildern arbeiten. Es gibt 9 mögliche Quadranteneinstellungen 0 = Vollbild 512×512; 1 … 4 = je ein Bildquadrant 256×256;
5 … 8 = je die rechte, linke, obere, untere Bildhälfte 256×512 bzw. 512×256.

[6] Multiplizieren und Dividieren konnte die DIP-Hardware nicht. Darüber, wie man das am elegantesten per Software löst, entbrannte jahrelang ein Wettbewerb zwischen den Entwicklern von Akademie und Robotron. In der Akademie saßen die Theoretiker, wir waren die Praktiker. Das ist aber ein anderes Kapitel.



7. Eine ***Normierungseinheit*** (NE) gestattet die Addition (oder Subtraktion) einer Konstanten. Weiterhin kann eine Bitverschiebung nach links oder rechts erfolgen, d. h. Multiplikation mal (Division durch) 2, 4, 8, 16 usw. Damit Bitbreitenreduktion auf 10.

8. Die letzte Einheit der DIP-Pipeline ist wieder eine Lookup-Tabelle. Der ***Ausgangsfunktionsformer*** (AFF) wandelt einlaufende Werten 0 … 1023 in Bytewerte 0 … 255. Damit Bitbreitenreduktion von 10 auf 8.

Damit ist die eigentliche Rechnung beendet. Der Datenstrom fließt wie folgt weiter:

9. Die Daten werden in ***Statistikeinheiten*** abgezweigt, welche Minimum, Maximum, Summe und ein Histogramm[7] berechnen. Die Ergebnisse lassen sich über sog. Statistik- und Histogrammregister vom Steuerrechner K1630 auslesen.

10. Ein in der ***Grafiksteuerung*** gespeichertes Bild kann per „Grafikoverlay“ eine zusätzliche Vordergrundgrafik samt Cursor einblenden, was allerdings nur eine Nebenfunktion darstellt.

11. ***Bildspeicher-Videoschreiben.*** Am Ende kann der finale Hauptdatenstrom wieder in einen Bildspeicher eingeschrieben werden. Der Videoschreib-Bildspeicher kann nicht zugleich videogelesen werden.

12. Ein Digital-Analogwandler (D/A) erzeugt ein analoges Fernsehsignal für die Bildanzeige per ***Farbmonitor.*** Zuvor durchlaufen die Daten mit den Korrekturfunktionsformern noch

[7] Das Histogamm ist gesondert anzufordern; dafür werden zwei DIP-Durchläufe benötigt.

weitere Lookup-Tabellen-Reihe zur Monitoranpassung dient. Auf die Daten selbst hat dies allerdings keinen Einfluss, denn Videoschreiben ist ja bereits zuvor abgezweigt worden.

Weil es die gesamte Prozessorarchitektur dreifach gibt, je einmal für die Farbkanäle Rot, Grün und Blau (s. Fußnote oben), kann der DIP auch Echtfarbbilder mit 24 Bit Farbtiefe bearbeiten.

Vier Arten der Bildverarbeitungs-Prozessierung

Wie arbeitet man nun mit so einer Maschine? Es gibt 4 grundlegende Regime.

Am Anfang steht das triviale ***1:1-Durchschalten*** eines Bildes „durch den Displayprozessor". Ein in einem Bildspeicher befindliches Bild wird lediglich angezeigt. Dies ist ein durchaus häufiger Fall. Bildverarbeitung im eigentlichen Sinn erfolgt dabei noch nicht.

Die nächste Stufe besteht darin, einen ***singulären DIP-Zustand*** einmalig einzustellen und diesen permanent beizubehalten. Damit kann der DIP z. B. ein Kamerabild auswerten. Die Bearbeitung erfolgt fortlaufend 25× pro Sekunde. So können Algorithmen umgesetzt werden, die in einem einzigen Pipelinedurchlauf berechenbar sind. Das Ergebnis wird angezeigt und kann, z. B. über eine serielle Schnittstelle, eine Prozessteuerung ansprechen. Blitzschnell, die Algorithmen dürfen jedoch nicht sehr umfangreich sein. Beispiel: Artefakterkennung in Produktionsprozessen.

Schließlich lassen sich komplexere Algorithmen in ***sequentieller DIP-Prozessierung*** umsetzen. Auch hier wird zunächst wieder ein gewisser DIP-Zustand programmiert. Die Bilddaten durchlaufen den DIP, jetzt aber nur ein einziges Mal. Das Ergebnisbild wird in einem Bildspeicher abgespeichert (videogeschrieben). Anschließend wird ein neuer DIP-Zustand eingestellt und nun ist das abgespeicherte Bild das Quellbild des neuen Zyklus'. Das wird beliebig fortgesetzt. Dieses Verfahren ähnelt der sequentiellen Art der Befehlsabarbeitung in gewöhnlichen Computern, nur dass pro Sekunde nicht 100.000 Zahlen verarbeitet werden, sondern 25 Bilder à 250.000 Byte[8]. Die Bildspeicherbelegung will vom Programmierer sorgsam durchdacht sein, denn mit den 8 Bildspeichern gibt es ja nur 8 „Speicherregister". Wenn diese nicht ausreichen, müssen Bilder zeitaufwändig in Dateien ausgelagert werden[9]. In wenigen Sekunden kann der DIP einige hundert „Befehle" mit vielen hundert Millionen Operationen abarbeiten.

Das vierte und letzte Regime ist die Bildverarbeitung ohne Displayprozessor, nur im K1630, sog. ***Host-Prozessierung***. Statt schnellem videolesen und videoschreiben langsamer Speicherzugriff über Rechner-IO. Die Algorithmen werden unabhängig vom DIP „in gewöhnlicher Programmierung" realisiert. Bedeutsam ist dies auch, um Bilder zu scannen, zu drucken und in Dateien zu speichern. Die Host-Prozessierung ist das „Tor zur Welt", denn natürlich müssen die Bildspeicher mit Bildern gefüllt und die Ergebnisse wieder gesichert werden. Der Hauptnachteil der Host-Prozessierung ist ihre geringe Geschwindigkeit. Eine

---

[8] Praktischer Wert 12,5 Bildzyklen pro Sekunde, weil das Umprogrammieren auch etwas Zeit beansprucht.
[9] Dies ähnelt ein wenig der Registernutzung beim Assembler programmieren: Wenn die Register nicht ausreichen, muss man Daten über BUS im Hauptspeicher ablegen.

Bilddatei speichern, dauert einige Sekunden, oft rechnet er aber auch Minuten, manchmal Stunden.

Wie der DIP programmiert wurde

Es leuchtet ein, dass das Programmieren so eines Prozessors eine gewaltige Aufgabe ist. Ein DIP-Zustand wird programmiert, indem zwölf 16-Bit-Maschinenregister mit – maschinentypisch – ziemlich kryptischen Steuerdaten geladen wurden. Dies sind die legendenumwobenen DIP-Steuerregister[10].

Da galt es also zunächst, ein Programmiersystem für das Ungetüm zu entwickeln. Eine übliche Programmiersprache wie Assembler, Fortran (damals weit verbreitet).Pascal oder C scheidet aus, denn wie soll man damit Parallelprozesse beschreiben?

Die Steuersprache DCTR

Die erste, 1982 in der Dissertation von Dr. Wolfgang Wilhelmi entwickelte DIP-Steuersprache war ***DCTR — DIP Control and Transfer Routine.*** DCTR war sehr leistungsstark, u. a. wurde dabei ein 64-Bit-Code benutzt. DCTR war aber auch eine sehr kryptische Angelegenheit. Programmiert wurde, indem man mächtige „DCTR-Makros“ in den Assemblerquelltext eingefügt hat, wodurch sich das Assemblieren stark verlangsamt hat. Außerdem musste man immer das gleichnamige Unterprogramm DCTR mitlinken, was den Softwareprozessor realisierte, „auf dem DCTR lief“. Das bedeutete Speicherverbrauch. Programmaufruf aus höheren Programmiersprachen ging überhaupt nicht.

---

[10] Die 12 DIP-Steuerregister. Wir geben in eckigen Klammern [15-0] die Bitbelegung an; hierbei bitte beachten, dass es 3 Farbkanäle R, G, B gibt (s. Fußnote oben):
· Die Register 1 bis 3 sind die 3 Bildauswahlregister R G und B. Diese ordnen die Bildspeichernummern 0 … 7 den 4 Quelldatenströmen im Multiplexer zu. Datenstrom 0 wird mit den Bits [12-14], 1 mit [8-10], 2 mit [4-6] und 3 mit [0-2] programmiert. Die dazwischenliegenden Bits [11], [7] und [3] schalten die Verzögerungen 1, 2 und 3 ein.
· Register 4 ist das Tabellenauswahlregister, die genaue Funktion ist mir leider nicht mehr bekannt, möglicherweise werden damit beim Tabellenladen die einzelnen Tabellen angewählt.
· Register 5 ist das Tabellenstatusregister. Die 12 Bit [3-0], [7-4], [11-8] schalten die Eingangsfunktionsformer (R, G bzw. G) ein. Die Bits [14, 13, 12] schalten die Ausgangsfunktionsformer R G B ein. Bit [15] aktiviert die Grafiküberlagerung. Bei nicht aktivierter Tabelle läuft der Datenstrom 1:1 durch.
· Nr. 6 bis 8 sind 3 ALU-Register R G B, diese sind besonders kryptisch, pro ALU gibt es eine Hexadezimalziffer Basiscode plus ein „Überbit“ [Bits 12-14]. Bit-Zuordnung je ALU 2 [8-11 und 14], ALU 1 [4-7 und 13], ALU 3 [0-3 und 12]. Um ALU 1 z. B. auf Addition zu stellen setze man die Bits 00220, für Subtraktion die Bits 20140 (Wert je oktal).
· Die Register 9 bis 11 sind die Normierungsregister, die je die Normierungseinheit R G B steuern. Je ein Summand von –1023 … 1024 wird auf den Bits [0-10] übergeben. Die Bits [11-15] stellen mit einem Wert –16 … 15 Links- bzw. Rechtsverschiebungen ein, je nach Wert Division ÷ 65536 bis Multiplikation × 32768. (Da rechts/links Nullbits eingeschoben werden, sind Beträge >10 wenig sinnvoll, das Ergebnis ist dann immer Null.)
· Register 12 ist das Histogrammregister, damit (wohl) Histogramm-Anforderung. So wurde das Histogramm R exemplarisch durch das Setzen von Bit [5] aktiviert.

Die Steuersprache DSR

Weil DCTR sehr schwer zu programmieren war, hat der geniale ZKI-Programmierer Andreas Graf 1985 ein viel einfacheres System geschaffen, die ***DIP Steuer Routine DSR.***

DSR, das waren zunächst 5 winzige Unterprogramme, die die Hardware angesprochen haben und in einer Unterprogrammbibliothek DIPOTS.OLB bereitgestellt wurden[11]. Dazu passend hat Andreas Graf eine Programmiersprache entwickelt, ebenfalls DSR genannt. „In DSR" konnten die DIP-Steuerregister in einer Art Assemblersprache mit ca. 10 je zweibuchstabigen Befehlen beschrieben werden, denen jeweils Parameter folgen konnten. 10 Befehle, das war recht überschaubar. „TS" heißt etwa Tabellenstatusregister, „NE" Normierungseinheit[12]. Schließlich hat Graf einen Cross-Compiler geschrieben (der sinnigerweise auch wieder den Namen „DSR" trug) – logisch, denn wenn man eine Programmiersprache entwickelt, muss man natürlich auch den dazugehörigen Compiler bereitstellen. Dieser Cross-Compiler hat nun das DSR-Quellprogramm in einen Assembler-Unterprogramm übersetzt, das sog. DSR-Anwenderprogramm. Dieses wurde dann zur Laufzeit (also von einem Hauptprogrammrahmen aus) aufgerufen. Es hat dann den kryptischen bitweisen Zusammenbau der DIP-Steuerregister organisiert und mit den DIPOTS-Unterprogrammen einen DIP-Zustand eingestellt.

Klingt kompliziert, ich will auch nicht behaupten, das Programmieren so eines hoch-komplexen Prozessors wäre ein Kinderspiel. Letztendlich war es aber mit etwas Übung recht überschaubar:

· Die 10 DSR-Befehle auswendig lernen
· Pro DIP-Zustand ein DSR-Quellprogramm schreiben
· Die DSR-Quellprogramme mit dem Compiler DSR in die Anwenderprogramme compilieren
· Einen Hauptprogrammrahmen schreiben, der die DSR-Anwenderprogramme aufruft
· Assemblieren
· Linken
· Starten

So haben wir bei Robotron dann jahrelang den DIP programmiert.

---

[11] Das Unterprogramm XGREGI lädt die 12 DIP-Steuerregister. Die Funktionsformer werden mit dem Unterprogramm XGLFF geladen. XGBSRD programmiert das Videolesen der Bildspeichern, XGBSWR deren Videoschreiben. Wenn alles eingestellt ist, synchronisiert XGSYNC den DIP, d. h. es wird ein Durchlauf gestartet und dessen korrekte Beendigung abgewartet. XGLFF musste man direkt rufen. Den Aufruf der anderen DIPOTS-Programme hat das DSR-Anwenderprogramm besorgt. Mit dem „XG" im Programmnamen hat sich Andreas Graf für ein paar Jahre „verewigt".

[12] Leider gibt es keine DSR-Dokumentation mehr. Die 10 Grafschen DSR-Befehle waren m. W.:

BA			Bildauswahl
TS			Tabellenstatus
AR	**AG**	**AB**	ALU (R/G/B)
NE			Normierungseinheit
HR	**HG**	**HB**	Histogramm (R/G/B)
RM			Refresh Memory, d. h. Programmieren der Bildspeicher.

Es folgen je gewisse Parameter, z. B. programmiert die Befehlszeile „**RM 4,0,FR,1,,7,8**" Bildspeicher 4, Quadrant 0 (=Vollbild). „**FR**" (from) bedeutet Videolese-Aufruf, d. h. XGBSRD, „**TO**" wäre Videoschreiben, d. h. XGBSWR. Die Vergrößerung ist 1. Es folgt nach einem leeren Parameter die Bildverschiebung X=7 und Y=8.

Außer DSR konnte man in dem DSR-Quellprogramm auch anderen Assemblercode hineinschreiben. Der wurde von dem Cross-Compiler DSR einfach 1:1 in das DSR-Anwenderprogramm kopiert[13]. So konnte man sein Hauptprogramm auch gleich mit in dem DSR-Quellprogramm mit abcoden. Genial war, dass das DSR-Entwicklerprogramm auch von Fortran, Pascal oder C aus aufgerufen werden konnte. So war der DIP nun auch in Fortran, Pascal und C programmierbar. Das war insbesondere für Entwickler bedeutsam, die Assembler nicht brachten. Und als besonderen Clou gab es noch einen geniale DSR-Parameter-Übergabemechanismus als Zugabe[14].

Der Interpreter DIP

Man musste aber immer noch dieses DIP-Anwenderprogramm schreiben und in einem Hauptprogrammrahmen unterbringen. Das war manchmal nervig, insbesondere wenn es schnell gehen musste.

Bei uns bei Robotron hatten wir nun ebenfalls clevere Entwickler. Christian Gruner hat einen Parser für die Sprache DSR gebaut und damit einen Interpreter geschrieben. DSR wurde nun nicht mehr compiliert, sondern *auf dem DIP sofort ausgeführt.* Separater Programmcode war nun nicht mehr erforderlich. Und indem Christian Gruner das Programm sinnigerweise auch noch „DIP“ genannt hat, war der DIP nun nicht länger ausschließlich eine in Schränken herumstehende hochkomplexe Hardware. „DIP“ war nun auch ein Konsolenprogramm, mit dem man die Hardware DIP im Dialog „in DSR“ direkt ansteuern konnte. So haben wir das vermeintliche Ungetüm Displayprozessor dann völlig handzahm gekriegt.

Wobei, Interpreter – das galt damals zunächst als fürchterlicher *faux pas.* Interpreter hatten seinerzeit den Ruf, gegenüber Compilern *fürchterlich langsam* zu sein. Nur blutige Oberschüler hackten in den 1980ern auf Commodores auf „Basic-Interpretern“ herum. Das Geniale war aber nun, dass der Interpreter genau in der Zeit auf dem K1630 die Interpretation rechnete, in der der DIP die Bildverarbeitung gemacht hat. Das war gleich noch einmal Parallelität der anderen Sorte: Parallelität von Interpreterzeit und Rechenzeit. Die vermeintliche Laufzeitverschwendung infolge Interpretation spielte überhaupt keine Rolle, denn solange der DIP ackerte, hatte der Host ja nichts zu tun. Da konnte er auch den Code interpretieren.

Wir brauchten überhaupt nicht mehr zu programmieren. Wir haben einfach die DSR-Befehle in ein Batchfile reingeschrieben (Batchfiles hießen damals „Indirektkommandodatei“, Dateityp „.cmd“) und das geniale Grunersche DIP hat die Abarbeitung besorgt[15]. Manchen

---

[13] Das sah so ähnlich aus, wie heutzutage Javascript in HTML.

[14] Per „P-Syntax“ können DSR-Entwicklerprogramme, Variable über Unterprogrammparameter übernehmen. Wenn man statt obigem Beispiel „`RM 4,0,FR,1,,7,8`“ nun „`RM 4,0,FR,1,,P1,P2`“ notiert hat, sind die beiden Verschiebungen X bzw. Y sind nicht länger unveränderlich 7 bzw. 8. Sie können vielmehr vom rufenden Hauptprogramm beliebig vorgegeben werden, und zwar erfolgt die Übergabe an das DSR-Unterprogramm über die Unterprogrammparameter Nr. 1 und Nr. 2.

[15] Und mir fallen immer mehr Raffinessen wieder ein. Verlangsamt wurde eine derartige Batch-Abarbeitung dadurch kolossal, dass das System das Programm DIP bei jeder Batch-Zeile jedes Mal erst umständlich von der Festplatte geladen hat. Bei irgendeiner Konsolenabarbeitung stört es nicht, wenn der Rechner da mal bissl hakelt, aber den Displayprozessor mit seinen 25 Operationen pro Sekunde hätte das völlig aus dem Zeitplan geschmissen. Es gab aber das legendäres Betriebssystemkommando „FIX“, steht heute noch in der RSX-

Algorithmus, an dem man zuvor tagelang herumdebuggt hätte, konnte man nun auf einem Messestand in 10 Minuten zum Laufen bringen. Wenn es dann lief, hat man den DSR-Quellcode aus dem Batchfile in ein richtiges DSR-Programm kopiert und „ordentlichen Code“ draus gemacht.

Die Entwicklung geht noch weiter

Das war aber noch nicht alles. Während das Wilhemische DIP-Steuersprache DCTR eine Art Maschinensprache für den DIP war, konnte man das Grafsche DSR als eine Art „Assembler für den Displayprozessor“ auffassen. Und das Grunersche DIP war ein Interpreter für die Assemblersprache. Das Ganze hatte aber einen großen Nachteil: Es war vollkommen maschinenabhängig. Ähnlich wie ein Assembler-Befehlssatz nur eine einzige Prozessor-architektur programmiert, lief „DSR immer nur auf dem Displayprozessor“. Denn da steckten ja stets die 12 Steuerregister in deinem Assemblercode.

Ab 1987 hat unsere Hardwareabteilung ein neues System entwickelt, das die A6472 ablösen sollte, das Bildanalysegerät BAG[16], mit EC1834 alias IBM PC als Steuerrechner. Doch dann hätten wir unsere gesamte DIP-Software neu implementieren müssen. Andreas Graf und Christian Gruner waren Entwickler von „maschinenorientierter Software“. Ich war in der Arbeitsgruppe „problemorientierte Software“. Also war es mein Job, mir über so etwas Gedanken zu machen. Es war nun ein phantastischer Gedanke, hier eine problemorientierte, „höhere“ Programmiersprache zu entwickeln, also so etwas, wie ein Fortran oder C für die parallele Bildverarbeitung. Wenn man mit so etwas den DIP programmiert, wäre der Umstieg auf ein anderes System kein Problem. Dann hätte man für die neue Architektur „nur noch“ einen neuen Compiler schreiben müssen, und der hätte dann den alten Code auf der neuen Architektur zum Laufen gekriegt.

Das war natürlich eine phantastische Herausforderung. Doch wie soll so etwas gehen, eine höhere Programmiersprache entwickeln, die „Parallelprozessor“ kann. Letztendlich ist dabei meine Programmiersprache VIP entstanden, woraus dann eine ganze Dissertation geworden ist. Aber das ist ein anderes Kapitel.

Die Algorithmen

Programmieren war aber nur die halbe Miete. Der DIP stellte noch eine andere, viel gewaltigere Aufgabe, welche lautete: „Sämtliche Algorithmen völlig neu denken.“ Einfach ein Pixel aus einem Bild „herausgreifen“ und „bearbeiten“ – das sollte man per Displayprozessor besser nicht tun. Denn das hätte ja jedes Mal 1/25. Sekunde gedauert, unendlich lange.

Dokumentation: „Ein Programm im Hauptspeicher fixieren“. Hat nie jemand gemacht, denn mit einem gefixten Programm war ja der Hauptspeicher dicht, da blieb ganz schnell das System „hängen“. Hier war es aber ideal, vorausgesetzt, man hat sich vorher die Hauptspeicherbelegung genau zurechtgebastelt. Dann ging es mit einem „gefixten“ DIP blitzschnell. Und am Ende von dem Batchfile bitte das „UNFIX DIP“ nicht vergessen, damit der Hauptspeicher wieder „Luft bekommen“ hat.

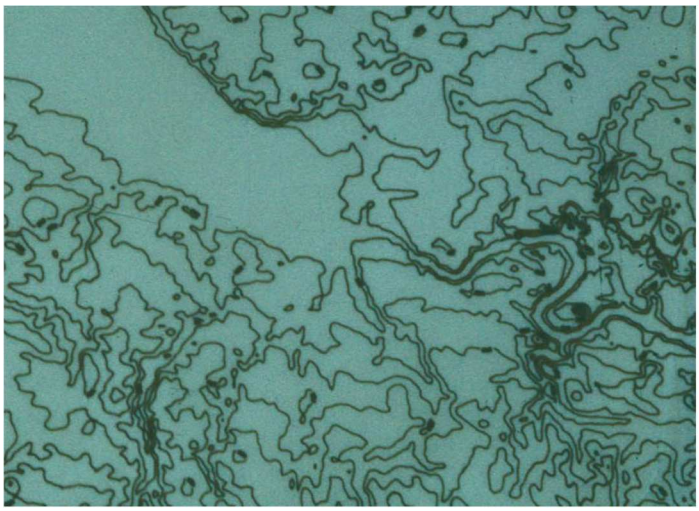
[16] 1989 ist die Entwicklung von unserem BAG abgebrochen worden, weil es dafür keine Produktionskapazität gab. Dass war ein harter Schlag für uns Entwickler. War eben doch nicht so toll mit der DDR-Wirtschaft. Uwe Pellgrü, den Hardwareentwickler (der bei mir im Nebenzimmer gesessen hat), hat das so mitgenommen, dass er gleich einmal einen Ausreiseantrag in den Westen gestellt hat.

Seine Stärke spielt der Prozessor nur dann aus, wenn alle Pixel eines Bildes derselben Vorschrift unterzogen werden. Unsere Grundaufgabe lautet also: „sequentielle“ Algorithmen „parallelisieren“. Da denkt man natürlich: Was ist denn das und wie soll das denn gehen? Es ist dies damals ein Riesenforschungsgebiet der mathematischen Grundlagenforschung gewesen, zu dem damals viel patentiert und veröffentlicht worden ist.

Es galt das Denken völlig umzustellen. Man durfte also nicht länger überlegen: „Teste, ob Hangpunkt Nordhang Borsberg Koordinate X, Y genau Z Meter hoch liegt, auf dass dann dort dort eine Höhenlinie entlang geplottet wird. Sondern: „Bearbeite ein Bild so, dass aus einem Höhenmodell mit einem Mal die komplette Höhenliniendarstellung entsteht“.

Wenn man ein kartografisches Höhenlinienoriginal zeichnet, ist das ein Arbeit von vielen Tagewerken. Und so etwas soll auf einmal in Sekundenbruchteilen gehen? Also – ein bisschen ein verrückter Gedanke war das schon. Kann so etwas funktionieren[17]?

Mittlerweile ist es für uns ganz selbstverständlich, dass Computerkarten aller Art in Sekundenbruchteilen entstehen, aber damals waren die Algorithmen noch völlig unbekannt und überhaupt erst einmal zu entwickeln. Da musste überhaupt erst einmal probiert werden, was überhaupt in der Technik steckte. Kein Mensch wusste damals, was ein digitales Geländemodell war. Das musste überhaupt erst einmal alles erfunden werden:



Höhelinien Elbtalzone. Rechts im Bild die markante Elbschleife bei Königstein. (Bild: Siegfried Kuhn, Kleinreinsdorf)

Wir waren die Schmiede, die den Kessel für den Cugnotschen Dampfwagen auf einem Rennfeuer geschmiedet haben, 50 Jahre vor der Errichtung der Göltzschtalbrücke.

[17] Die Theorie der „Parallelisierung sequentieller Algorithmen“ füllt Dissertationen. Knappe Illustration: Höhenlinien erhält man, indem man ein Rasterhöhenmodell mit einer Stufenfunktion „abtreppt“ und dann mit einem Hochpassfilter die Konturen extrahiert. Die Stufenfunktion besorgen die Funktionsformer und für den Hochpass nutzt man Verzögerungsregister und ALU. Die Mathematiker und Physiker in unserer Arbeitsgruppe haben mir als Kartografen gezeigt, wie man so etwas macht.

Epilog

1990 ist dann der ganze Ostblock untergegangen und unser großes Kombinat Robotron war doch nur ein unendlich kleiner Teil davon.

Von 1984 bis 1990 sind bei Robotron etwa 200 Bildverarbeitungssysteme produziert worden. Es war ein schönes System und übrigens das einzige digitale Bildverarbeitungssystem, das im Ostblock in Serie produziert worden ist. So eine Anlage war schon eine recht ordentliche Technik und die 6 Jahre, die ich bei Robotron als Entwickler tätig sein durfte, waren eine schöne Arbeit. Wir waren eine kleine FuE-Abteilung „EE3 – Entwicklung Software Bildverarbeitung“ mit 30 Mitarbeiterinnen und Mitarbeitern. Unten im Sockelgeschoss Mohrenstraße Ecke Glinkastraße waren unsere Rechnerräume. Heute ist dort das Bundesgesundheitsministerium. Auch Inbetriebnahmen und Messestände von Chabarowsk bis Budapest gehörten zu unserer abwechslungsreichen Arbeit.

Digitale Bildverarbeitung war eine ganz junge Technologie, es erschlossen sich unzählige Anwendungsbereiche. Vieles davon ist heute selbstverständlich, aber damals war ein digitales Bild etwas völlig Exotisches. Wir hatten Fernerkundungssysteme an der TU Dresden, am Zentralinstitut für Physik der Erde Potsdam, beim Forst in Eberswalde in Betrieb genommen. In der Satellitenbodenstation Neustrelitz wurden mit unseren Anlagen die Bilder des DDR-Wetterdienstes erzeugt und im Institut für Züchtungsforschung in Quedlinburg Mikroskopbilder von Chromosomen. Medizinische Systeme standen an der Berliner Charité[18] und im Zentralinstitut für Molekularbiologie in Berlin-Buch. Industrielle Anwendungen gab es im Schwermaschinenkombinat Ernst Thälmann in Magdeburg, am AdW-Institut für Mechanik in Karl-Marx-Stadt und im VEB Papierfabrik Greiz. Dr. Kuhn hat bei der Geophysik Leipzig mit unseren Systemen Höhenmodelle hergestellt, ich denke, das waren mit die ersten größeren Rasterhöhenmodelle die es in der Welt überhaupt gegeben hat. Berühmt waren auch die Arbeiten von Dr. Ohser (später Prof.) an der Bergakademie Freiberg. Die Post hat mit Schrifterkennung angefangen und außerdem gab es viele militärische Systeme. Unsere Anlagen gingen nach Polen, Ungarn, in die ČSSR, nach Rumänien, Bulgarien, Finnland, etliche Systeme auch nach Indien, China und Vietnam.

Der allerwichtigste Abnehmer unserer Bildverarbeitungssysteme war aber die Sowjetunion. Etwa die Hälfte aller Anlagen überhaupt sind zwischen Moskau und Chabarowsk in Betrieb genommen worden. Die große A6473 mit ihren 4 Displayprozessoren war der typische Rechner der sowjetischen Raumfahrtzentren. Die Sowjetunion war also dasjenige Land, das unsere Arbeit zum überwiegenden Teil bezahlt hat. Mit einem kleinen Programm von Benedikt Eckelt und mir – dem DIP-Handler DPH (2021 noch in Spuren in Online-Dokumentation im Netz auffindbar) haben die Russen bei der Vega-Mission 1986 den Kern des Halleyschen Kometen entdeckt. Unsere Techniker waren damals mit dabei gewesen. Der hat es dann sogar auf die Titelseite der DDR-Zeitschrift NBI geschafft.

---

[18] An der Charité hat Klaus Voss (später als Lehrstuhlinhaber der Chef von Dr. Ortmann an der Friedrich-Schiller-Universität in Jena) das System AMBA (automatische Mikroskopbildanalyse) entwickelt, incl. der genialen Programmiersprache LAMBA, die in einem sehr schnellen Maschinencode lief, was natürlich in der Bildverarbeitung wichtig ist. Bissl sowas wie Turbo-Pascal, nur ein paar Jahre früher und eben an Bildverarbeitung angepasst. Ich habe es geliebt, mit diesem blitzschnellen Just-In-Time-Compilersystem Programme zu entwickeln. Prof. Voss hat mir mal den Trick verraten, wie man Vorwärtsreferenzen in einem Ein-Pass-Compiler auflöst. Allein AMBA ist ein Riesenthema für sich.

1990 war dann ganz plötzlich Schluss und nicht nur unsere Bildverarbeitungsabteilung, das gesamte Kombinat Robotron ist — schlagartig verdampft.

68.000 Computer-Fachleute hatten plötzlich nichts mehr zu tun, wenn man sich das aus heutiger Sicht mal vorstellt. Die ganze Industrieforschung war plötzlich verschwunden. Unsere Abteilung hat zwar überlebt, die Kollegen haben 1991 die Imtronic GmbH gegründet, der ich noch eine Weile freundschaftlich verbunden war (sog. Management-Buy-Out). Da aber unsicher war, ob das etwas wird, bin ich wieder in meinen alten Lehrberuf eingestiegen. Ich war ja gelernter Kartograf und habe wieder mit dem Zeichnen von Landkarten angefangen. Als es dann bei Robotron immer wackeliger wurde, habe ich gekündigt. Der 15.11.1990 war mein letzter Arbeitstag gewesen. Und 14 Tage später hat auch mein Chef Glen Meller hingehauen. Es war dies schon auch eine Zeit der Melancholie.

Als unser Betrieb dann 1990 untergegangen ist, bin ich noch einmal nach Neustrelitz zu Dr. Grundmann gefahren. Die hatten ein FEAG (Film-Ein-/Ausgabegerät), auf dem ich mir noch ein paar Karten ausbelichten konnte. Am ZIAC (Zentralinstitut für anorganische Chemie der AdW) in Berlin-Adlershof war bei Frau Dr. Müller eine A6472 noch recht lange in Betrieb. An der durfte ich 1991 noch ein wenig rechnen, obwohl das nun nur noch von akademischem Wert war. So war ich möglicherweise der letzte DIP-Programmierer, den es in der Welt gegeben hat. Ich habe mir dann noch ein paar Bilder auf Magnetband abgezogen. Und dann war endgültig Schluss.

— — —

Heute zeugen die Aufsätze von Siegfried Junge und Gerhard Merkel – das waren damals unsere Chefs und Direktoren – von dieser wundersamen untergegangenen Welt. Ein herzliches Dankeschön geht an die Macher der Seite www.robotrontechnik.de. Dem Rechenwerk Halle gebührt der Dank, ganz viel von der Betriebssystemdokumentation, die wir hatten, bewahrt zu haben. Wenn man das wieder liest, ist es so, als wäre es gestern gewesen. Rechner booten, Platten mounten und dann wieder ein paar von den schönen QIOW$C abcoden.

Ein paar von den Magnetbändern habe ich auch noch rumliegen. Die schmeiße ich nicht weg, denn an denen hängt nach all den Jahren immer noch mein Herz.

— — — — —