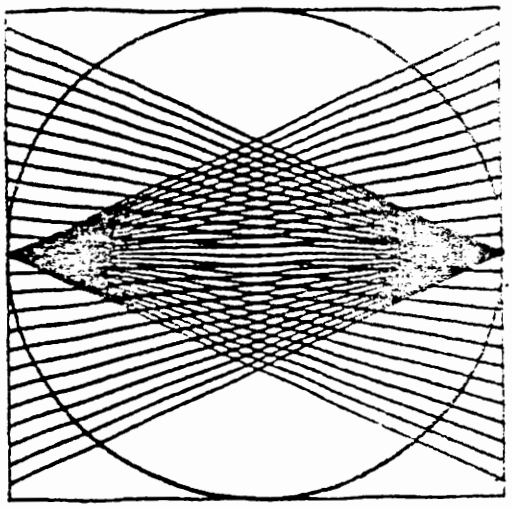
BIO - BUTLER IN AND OUT

* ORGAN FOR BUTLER BRUGERGRUPPE *



* BUTLER BRUGERGRUPPE *

NR. 1 *** 1986

INDHOLD

Redaktionelt:

Der er desværre atter gået for lang tid siden nr. 2.
Ærgerligt, for vi har fået en del forespørgsler, der gav udtryk for, at der er et behov for et fælles forum for udveksling af erfaringer.
Blandt årsagerne til forsinkelserne vil jeg fremhæve nogle som særligt væsentlige:

1) Vi mangler den multinationale koncern i baghånden. Redaktionen bliver (som bekendt?) hverken lønnet af IBM, Comodore, AT&T eller ITT, ja vi bliver faktisk slet ikke lønnet, så der er nogle småjobs, der også kræver lidt tid mellem skriveriet. Derfor har vi ikke så meget tid til at opfinde læserindlæg ...
godt gættet! det fører frem til årsag nr.

2) Vi mangler DIT bidrag til bladet. Vi har modtaget reaktioner både af typen "Det er alt for banalt!" og "Det gik helt hen over hovedet på mig, det var alt for teknisk!". DIT bidrag er med til at lægge linjen! om en sådan findes. Efter min opfattelse skal der både være plads til den snedige assemblerrutine til komplexe beregninger og den nyttige og måske banale COMALprocedure, der løser et simpelt problem. Det har tidligere vist sig, at de fleste, der skriver programmer, (i det mindste de ikke professionelle) er ved at træde tærne af sig selv af bar generthed, når man beder dem demonstrere deres programmer. "Det kan sikkert skrives meget bedre" - "det er sikkert gjort uelegant"- etc. Muligvis rigtigt, men det er da ikke særligt smart at vi alle sammen opfinder det varme vand igen, og det er en af konsekvenserne af denne skyhed. Desuden kan man lige så godt se i øjnene, at stort set ethvert program kan skrives bedre, men det er 100% uinteressant, hvis det ikke bliver udnyttet. Derfor, fingrene på tasterne og skriv dit bidrag. **NU!**

Send dit bidrag på diskette (den returneres sammen med en fra diskoteket efter eget valg), og eller på papir (nyt farvebånd om muligt, og helst beskrevet inden for formatet 24 * 16 cm, der skal være plads til et hoved (bladets, naturligvis) og margin.)

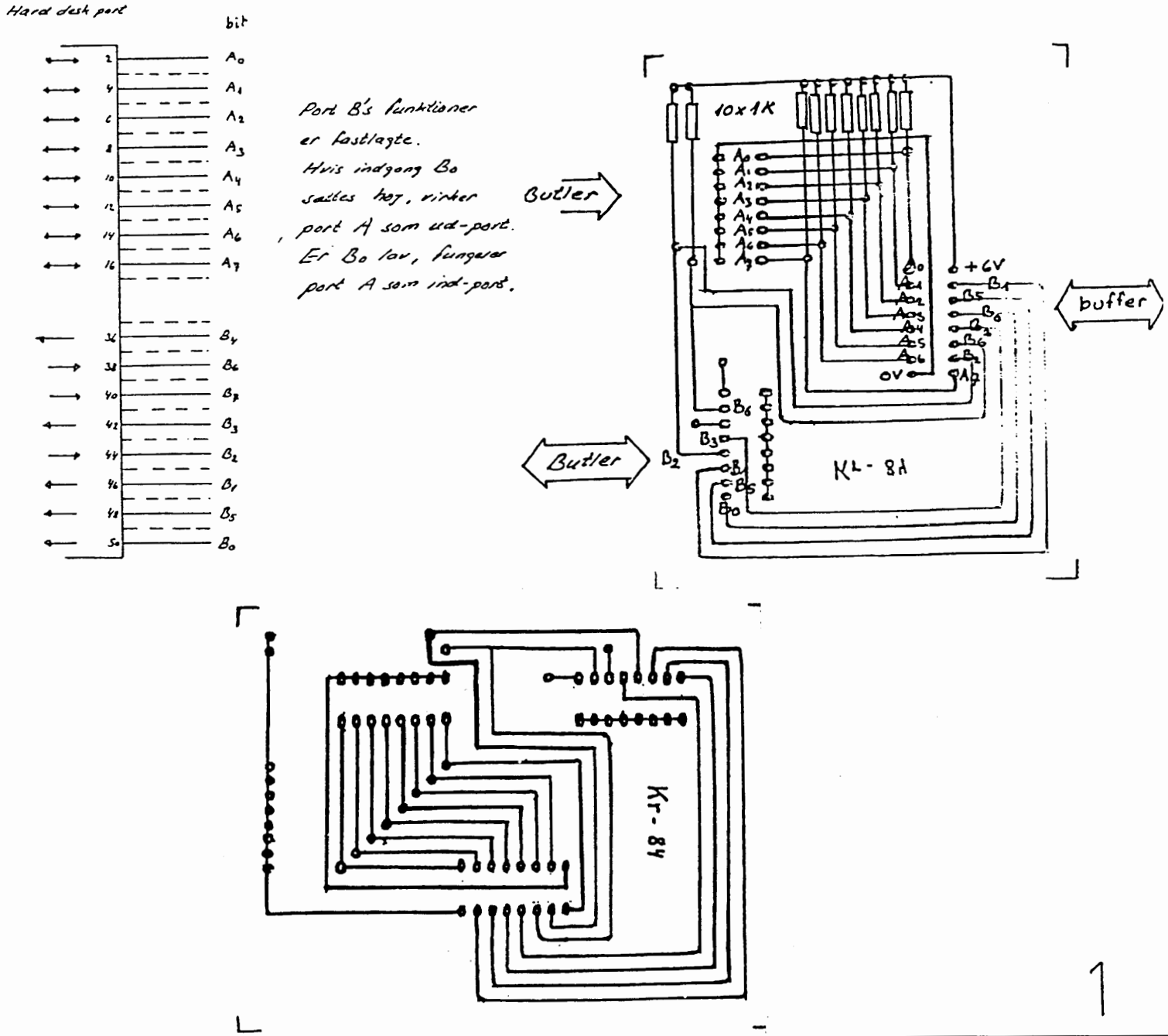
3) Den sidste årsag har vi om muligt haft endnu mindre indflydelse på. Det første års kontingent blev lovet betalt af Paul Bjørnums firma BOGIKA Data-Systemer Aps. Det løfte har bestyrelsen indtil for nylig taget for gode varer, selv om kontanterne var påfaldende lang tid om at manifestere sig. Dette gav anledning til yderligere forsinkelse, da bestyrelsen af praktiske årsager havde truffet beslutning om selv at stå for trykning og udsendelse af bladet. Trykkerier (og P&T) giver ikke gerne kredit, og vi var ude af stand til at overtale trykkeriet til at trykke penge.
Bestyrelsen fattes naturligvis også penge (den består af lutter lønmodtagere) og financierer derfor nødigt bladets drift af egen lomme. Vi har for dette nummers vedkommende (og måske også for det næstes, Jo! det er rigtigt nok, der kommer flere!) fundet en midlertidig ordning med BOGIKA DATASYSTEMER A/S.
I næste nummer kommer der flere detaljer om baggrunden for den ejendommelige økonomiske situation, samt tid og sted for generalforsamlingen.

P. b. v.
Carsten Linde
formand for BBG

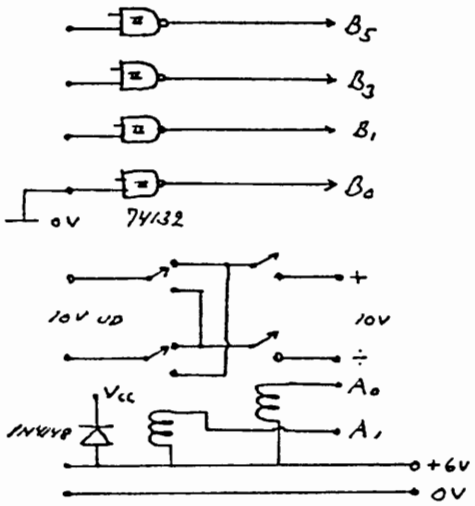
P.S. Hjælp din kollega, send et program, en programstump, en ide! Hvordan har DU brugt analogindgang, lyd osv.

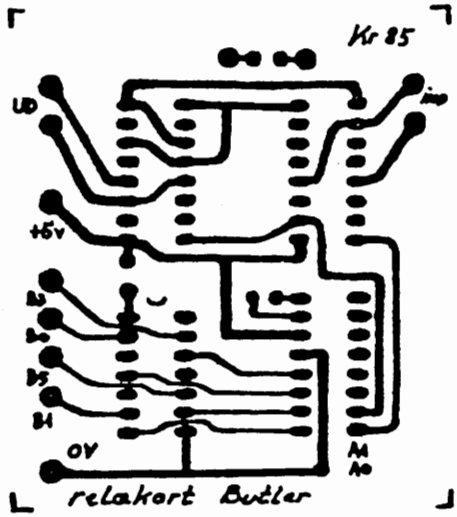
BUTLER-EKSPRESSEN

Den almindelige opfattelse af datamaskinens anvendelse (og dette gælder ikke mindst blandt skoleelever) er, at det er noget med en skærm, nogle taster og måske et enkelt "blib" nu og da.
Som lærer i faget har jeg søgt at råde bod på denne misforståelse ved at lade skolens Butler styre noget udenfor maskinen. Der er mange muligheder for egnede styrbare sager lige fra en kompliceret robot med stepmotorer til en simpel gearmotor, der hejser en "elevator" op og ned. En meget lang vindueskarm i vort datalokale afgjorde sagen, så valget faldt på et elektrisk tog med DC-motor.
Inden det kom så vidt påkrævedes et større detektivarbejde for at finde de rigtige portud- og indgange, og et lille panel med lamper og kontakter blev fremstillet for at eksperimentere med og demonstrere funktionen af Butlerens port.
De forskellige enheder og print er således opbygget hen ad vejen. En færdig konstruktion kunne nok laves lidt smartere og mere sammenhængende.
Her følger en beskrivelse af kredsløbene fra port til skinner.

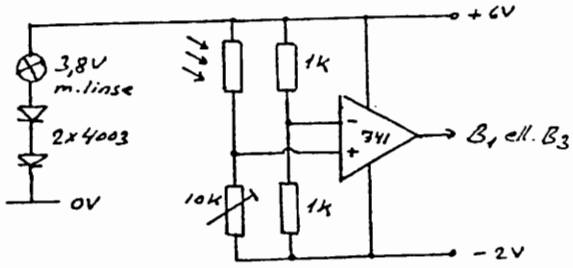


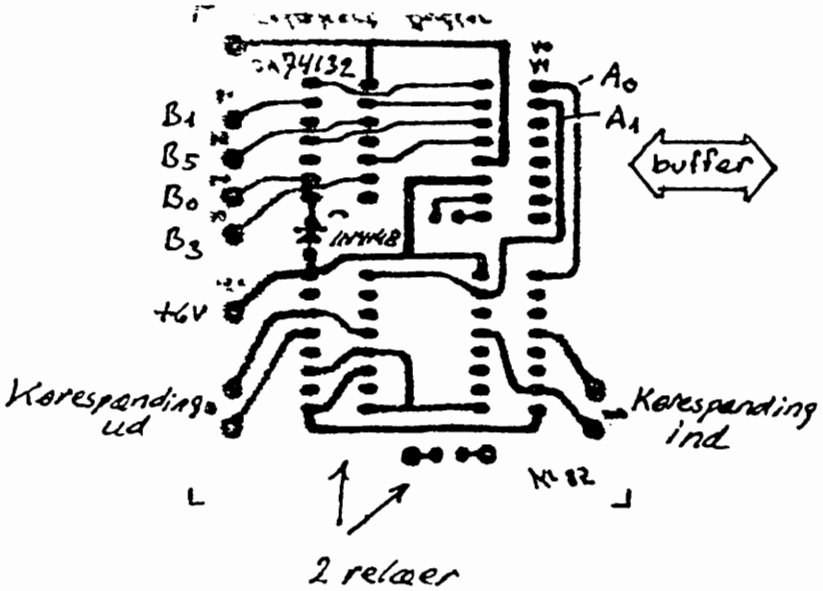
Bufferprintet er med fladkabel forbundet til et relækort med to relæer med dobbelte skiftekontakter. De passer i en DIL-16 sokkel, men kræver 6 volt for at trække. Derfor er der indskudt en diode mellem forsyningsspændingen og TTL-kredsen, der er monteret på printet.
Indgang B_0 skal holdes lav af hensyn til port A's funktion. De 3 andre indgange kan forbindes med følere til styring af toget. Her er kun anvendt B_1 og B_3.

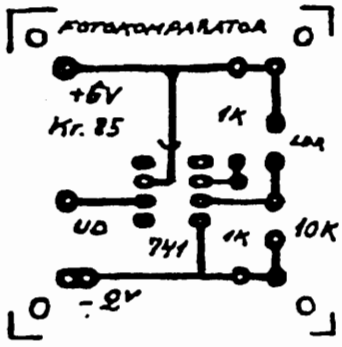




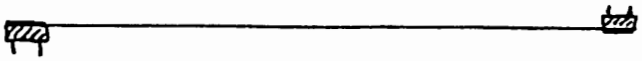
Følerne er anbragt ved banens endestationer, hvor toget ved sin ankomst bryder en lysstråle. Herved går operationsforstærkerens udgang lav, men det viser sig nødvendigt at forsyne 741 med -2 volt for at komme tilstrækkeligt langt ned til at få 74132 til at skifte.
Indgang B_1 lav giver bitmønsteret 0001 0011 på port B, og B_3 lav giver 0001 1001. Når toget ikke er på en endestation, vil bitmønsteret være 0001 0001. Bit 0 er jo høj af hensyn til port A, og bit 4 er høj, da den svæver. Omsat til decimal er de 3 tal: 19,25 og 17.
Hvad angår uddata på port A, vil 0000 0001 få toget til at køre.
0000 0011 vil vende retningen, og 0000 0000 afbryder kørespændingen.



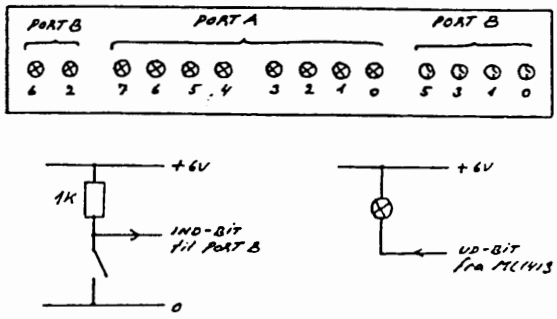




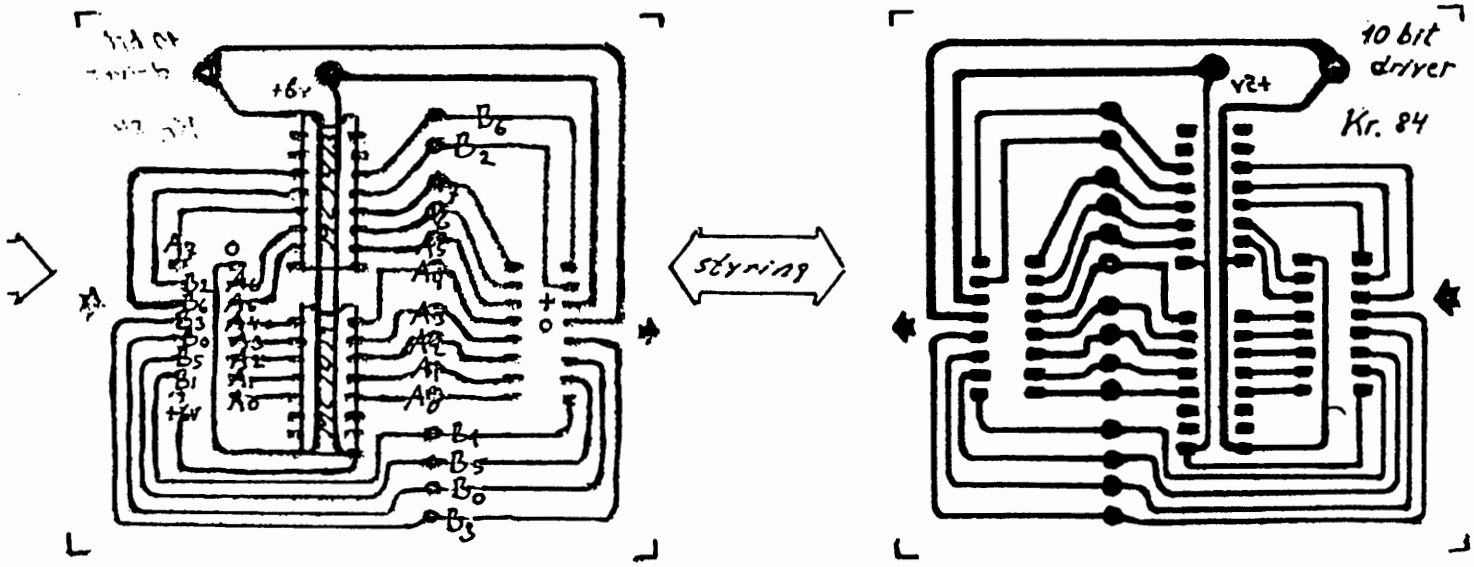
2 16-leder fladkabler forbinder porten til et print, hvor portudgangene er bundet op til + med 1 K modstande.
Fra printet føres et 16-lederkabel til et bufferprint med 2 stk. MC 1413 (ULN 2003 A). I de 16 ledere er indeholdt foruden + og 0: Port A, bit 0 til 7 og port B bit 2 og 6 (ud) samt bit 0, 1, 3 og 5 (ind).
PÅ FLADKABLET ER STIKKENE MONTERET HVER SIN VEJ !



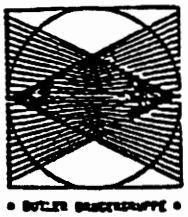
Foruden til en DIL-16 fatning er bufferprintets udgange og de 4 port B indgange ført ud i en række printspyd. Hertil er fast forbundet et panel med 10 glødelamper (6 V - 0,05 A) og 4 afbrydere.



Panelet er nyttigt, når man skal følge med i tilstandene på portene og desuden velegnet til demonstration af og træning i at få de rigtige bitmønstre ud.
Under styring af Butlerekspressen skal alle kontakter være afbrudt for ikke at blokere indgangene. Desuden skal B_0 altid være afbrudt, når man bruger port A som ud-port.
Det bør nok her nævnes, at MC1413 tåler en strøm på 500 mA pr. udgang, og at den er diodebeskyttet mod induktive spændinger fra relæer o.lign.



E.com



Brugergruppen er kommet i besiddelse af en stribe disketter fra C-Users Group. C133 indeholder en fuldskærms editor til progamskrivning. Jeg er helt på det rene med, at mange har adgang til ComPas, Polypascal eller Turbopascal med kombineret editor/compiler, eller fortolkede sprog som COMAL80 eller BASIC. Kan man i øvrigt ikke drømme om at skrive i andre sprog er der naturligvis ingen grund til at tænke på endnu en editor, så stop her.

Andre står måske i den situation, at lysten til assembler, C, Modula-2 osv. er til stede, compileren måske også, (ASM.com leveres med CP/M, Small C kan fås fra brugergruppen, for nu at nævne legale muligheder), men editoren mangler. ED.com fik man for resten med, da man købte CP/M, men den kan jo ærligt talt fritage enhver fra lysten til at se en datamat i et længere tidsrum. Write-It.com og Word-Star er anvendelige, men det tager oceaner af tid, når editoren skal indlæses, den fejlbehæftede fil hentes, rettes, nedskrives, editoren forlades, compileren hentes, programfilen hentes, compileres til første fejl, og så en gang til...

Et tekstbehandlingsprogram er ikke en programeditor, alene af den grund at det tager alt for lang tid at komme ud og ind af den.

Kravene til en programeditor er mangfoldige, og der er flere på markedet, der indfrier dem på fortrinlig vis, men de koster kroner.

e.com er ikke mindst af denne årsag interessant - den er GRATIS.

Ud over det har den flere gode egenskaber:

+ hurtig start trods størrelsen, e minfil.asm, og du er klar til at rette/skrive dvs. ingen indgangsmenu.

+ start med hop til bestemt linje, dvs. efter en compilerfejlmelding som "Missing ";" in line 323 " aktiveres editoren med e minfil.asm -323, og vupti er cursoren på den suspekte linje (ok, der går nogle få sekunder).

+ indeholder alle nødvendige instruktioner til:

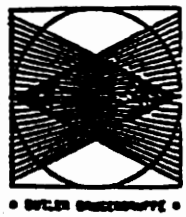
søgning, forlæns
baglæns
uden hensyn til store/ små bogstaver
antal gange

udskiftning, do

blokflyt

blokkopi

blokskriv/læs til disk



blokprint

blokslet

blokjustering

automatisk indrykning (som Compas)

linjescroll (en linje kan være indtil 255 karakterer)
af en linje eller af en side ad gangen.

+ store filer (max ca 4400 linjer, der automatisk skrives/læses på disken)

+ hop til linjenr xxxx

+ tekster kan flettes ind i hinanden.

+ filer kan slettes eller omdøbes i e.

+ diskettens resterende plads kan ses.

+ arbejder på tværs af usernumre/diskettedrev.

+ hvis man editerer på en ikke-initialiseret diskette (uden ^c) får man ganske vidst den rædselsvækkende BdosERR: Disk R/O men teksten er ikke tabt.

+ sætter kortvarigt Cursor på den sidste åbne parantes eller klamme Æ Å, ( ), æ å , som støtte for C-programmøren

+ består af 1 .com fil, dvs ingen overlays.

+ er skrevet i C. Kildeteksten medfølger, dvs. den kan modificeres af brugeren (hvis man har adgang til en BDS C compiler (v.1.50a) underforstået.) F. eks. har jeg indbygget automatisk skift af printerens karaktersæt til USASCII, 6 karakter venstre margin og 12 kar/inch.

+ kontroltasterne ligger omtrent som WStraditionen byder, dog med piltaster. Funktionstasterne kan stort set placeres efter egne lyster (omcompilering).

+ der medfølger en udmærket brugsanvisning (på engelsk). 17 sider.

Ulemper:

- den er skrevet i C, og 8/16 bitsmaskiner har det ikke særligt godt med hverken C eller andre højniveausprog. Derfor fylder den ad H.T. Da den anvender mellemfil på disken, betyder det dog ikke en alvorlig begrænsning for arbejdslageret, men på maskiner med begrænset diskettekapacitet er det da et handicap. Var den skrevet i assembler, ville dens 32k sikkert skrumpe ind til 8k,. . . men prisen!

- Butlerens langsomme skærm har gjort det nødvendigt at arbejde med "gnister" på skærmen.

- Den langsomme skærm giver ligeledes problemer, når man sætter en kraftig finger på pilOP og pilNED. Cursoren fræser videre, indtil den indbyggede buffer er tømt. Det problem er der mindst en løsning på.

- P.t. kan "DIR" kun vise de 64 første filer på en disk.

- e.com anvender skærmfunktionerne BRIGHT og DIM. Jeg har oversat det til INVERS og ikkeInvers. Dermed bliver CURSORlinjen inverteret. Det ser lidt spøjst ud, men er ikke så tosset efter et stykke tid. Funktionen kan kobles fra (under editeringen), men giver desværre alle linjer som inverteret, og det ser tåbeligt ud!). Denne "feature" vil jeg modificere ved først givne lejlighed.

- Blokfunktionerne er begrænset til hele linjer, dvs. skal et udtryk pilles ud af en linje for at blive kopieret ind i andre linjer, er det nødvendigt at "knække" linjerne, så udtrykket står på en linje for sig, og indkopiering kan ske mellem linjestumperne.

- Der kan ikke anvendes "wild-card" karakterer i søgemønsteret, eks * og ? som det kendes fra CP/M. (Til gengæld er søgehastigheden ret høj).

Alt i alt betragter jeg e.com som et smidigt og robust redskab til skrivning af programmer, når der ikke er mulighed for at anvende en integreret skærmeditor. Jeg har i en længere periode anvendt Compas editoren til skrivning af C programmer, men den kan nu få lov til at samle lidt støv eller "bugs" i diskoteket i denne sammenhæng. Det ville for resten ikke være et uoverskueligt problem at dressere e.com til at sætte de der fjollede linjenumre inden den skrev linjerne til en COMAL-80 fil, og høvle dem af igen, når den læste filen ind? Når COMAL-80 nu ikke har en skærmeditor, mener jeg.

Disketten indeholde kildeteksten og tilpasninger til forskellige terminaler. Der er, heldigt nok, også plads til e.com tilpasset BUTLER.

På disketten ligger desuden funktioner til BDS C, der gør det muligt at regne med meget, MEGET lange heltal, op til 99 cifre. Det er i det mindste meget sjældent, man får brug for større precision.

Li.

P.S.
Måske sidder der en og anden, der ikke har 100% fod på, hvad C er for en underlig fisk. I de næste numre af brugerbladet, som forhåbentlig udkommer med en noget højere frkvens - andet vil jo grænse til det umulige - vil jeg ridse nogle af de grundlæggende træk op på dette firsernes og måske halvfemsernes "transportable assembler".
Jeg vil på forhånd love at spare så meget på papiret, at der OGSÅ ER PLADS TIL DIT INDLÆG!

Artikler modtages med tak og en diskette fra diskoteket efter eget valg!

September 11, 1984

C User's Group
112 N. Main Street
Yates Center, KS 66783

Gentlemen and/or Ladies:

Enclosed is my membership form and a check for $10. I will be ordering some of your library disks later, starting off with Small-C and ROFF4. (Did you read the Microsystems article? Wow!)

A disk of files for submission to your library accompanies this letter, as you have probably noticed. This disk contains games and a (quasi-)universal terminal definition system. Before running any of the other programs, run TERMINAL to create the file TERMINAL.SYS. Other terminal configuration methods described in your newsletter don't allow for object code portability, whereas this one does and has been used with not only C but also BASIC, Pascal and assembler. The games were fun projects of mine and I've decided to put them all into the public domain. I only learned C recently, so two were converted from Basic, the other from assembler (Don't criticize my C too much! Give me a chance!). They are written in C/80 (Software Toolworks), which means they should be easily portable to any Unix-compatible C.

Here is a listing of the files on the disk:

DUMPSTAR.C, .SUB, .COM - Dump Star video game
TZ.C, .R, .COM, .DOC - Twilight Zone adventure game

NO FAIR READING TZ.R! (Maybe we ought to squeeze it?)

CMAZE.C, .SUB, .COM - Perspective maze game
(Careful - doesn't use TERMINAL.SYS; also does direct port I/O to an AY-3-9513 Cricket at ports B0,B1 hex)
TERMINAL.C, .COM, .DOC - Terminal configuration program
RDTERM.C, .COM - Check your terminal configuration
RANDOM.C - Random function for C/80
PORTIO.C - I/O port access for C/80
DOG.LTR - this letter

Someone can probably improve upon my random number generator. It is pretty crude. If anyone installs any terminals beyond what TERMINAL supports, I'd like to get a copy of the new values so that the program can be enhanced.

Operating system. Once the cross-assembler is complete I want to make a small-C for the 68000 which will then be used with the operating system as well as other projects. Currently the operating system is in Z80 assembler. When this is finished it will all become public domain. If anyone is working on similar projects, I'd sure like to get in touch with them.

Have fun and ... C you later!

Richard Rodman
6607 Mayfair Drive #302
Falls Church VA 22042
(703) 241-1681

Submission to BDS 'C' User Group January 1985

DESCRIPTION: Text utilities

Size	Name	Comments
		'e' screen editor, version 4.8
50K	E.doc	User's Guide, Tutorial introduction and implementation notes for the e screen editor. This is a fully-fledged screen editor designed for programmers (not a word processor).
		'e' source code (in C)
10K	E.h	Header file for 'e'
12K	E	Main function for 'e'
8K	E1	The rest of the code for 'e'
8K	E2	
6K	E3	
6K	E4	
6K	E5	
6K	E6	
4K	E7	
6K	E8	
4K	E9	
8K	ETERM	Terminal interface (this one is for the TeleVideo range but E.doc provides instructions for adapting it to other terminals. Terminals must have cursor addressing, clear to end of line, line insert, line delete, clear to end of page, to work with 'e')
8K	EHAZE	Terminal interface for Hazeltine range of terminals
8K	EKAY	Terminal interface for the KayPro 10
8K	EADDS	Terminal interface for the ADDS range of terminals
8K	EDEC52	Terminal interface for DEC VT52 terminal
2K	E.sub	'SUBMIT' file to compile and load 'e'
32K	E.com	Ready-compiled and loaded version of 'e', for Butler ~~TeleVideo~~ terminals
20K	L2.com	The L2 loader, here for convenience since 'e' must be loaded with this loader, and not with clink.
6K	NEWFOR48	Summary of upgrades from 'e' version 4.6
		Textcom file comparison utility
4K	TEXTCOM.doc	Instructions on using TEXTCOM
12K	TEXTCOM	The source code of TEXTCOM, a file comparison utility which is able to 'get back into sync' after finding a difference (insertion, deletion, or change) between two text files.
2K	TEXTCOM.sub	A 'SUBMIT' file to compile and load TEXTCOM
4K	INFO.doc	This file

(Note file sizes given above are round up to the nearest 2K bytes)

BDS C vers 1.50a is recommended for compiling these programs.

The above files have been submitted to the BDS C User Group by
G. Nigel Gilbert, MICROLOGY, 4 Deanery Road, Godalming, Surrey GU7 2PQ, England

Feedback on bugs, improvements you have added, and other comments are welcome.

A>

Files on This Disk:

Documentation:

-CATALOG.DOC	2k	This file.
-README.DOC	6k	Overall description of the disc.
AN.NRO	1k	General header file for documents.
CMDUTIL.NRO	13k	Kevin Kenny's subroutine library.
CORO.NRO	12k	Coroutine package for BDS 'C' -- function descriptions.
CORODOC.NRO	24k	Coroutine package -- detailed writeup.
CPROFILE.NRO	3k	Profiler for BDS 'C' programs.
CTOA.NRO	2k	CRL-to-assembly postprocessor for BDS 'C'.
DIFF.NRO	4k	Text file comparator.
INSTALL.DOC	3k	Installation instructions for programs.

Source code:

CASM2.C	24k	Kevin Kenny's modified CASM, needed for CTOA.
CMDUTIL.C	6k	Utility functions needed by several of the
CMDUTIL.H	1k	programs on this disc.
CORO.H	3k	\
CORO1.C	7k	Coroutine package for BDS 'C'.
CORO2.CSM	8k	/
CPROFILE.C	19k	Profiler for BDS 'C' programs.
CPROFIL2.CSM	12k	/
CTOA.C	20k	\
CTOA.H	5k	\
CTOA2.C	11k	CRL-to-assembly postprocessor for BDS 'C'.
CTOATBLS.CSM	9k	/
CTOATBLS.H	2k	/
DIFF.C	11k	Text file comparator
GENREL.C	6k	Service program to help build CPROFILE.
RETAB.C	5k	Example program for coroutine package.

A>

user 3
A>

Very Large Integer Package for BDSc

by

Hugh S. Myers

4/9/84

---

files:	size:	contents:
math.csm	22k	Very high precision math subroutine package as adapted and corrected from the March 1977 Dr. Dobb's Journal article by M.G. Dinneley by way of an M80 version by Thomas Hill (see the June and July 1983 issues of Lifelines). this version corrects errors common to both of its predicessors as well as making the transition to BDSc's csm format.
math.crl	8k	crl version of math.csm
vli.csm	14k	a 'front end' for math.crl. This contains all of the named subroutine calls for BDSc as well as all of the housekeeping necessary to use math.crl. At last count there were some 37 functions available to BDSc with 5 more for odds and ends. While I would not recommend any of my csm files as highly optimized, this one is at least easy to extend with new functions as needed.
vli.crl	4k	crl version of vli.csm
qpm.csm	4k	An example of extensions to vli.csm. This contains two functions that test either a string or an integer for primality. If the number is 1>n>65536, these functions will return TRUE or FALSE as the result of a seive test for "is n prime?" Otherwise TRUE, FALSE or MAYBE is returned. For an example of usage, see the program prime.c.
qpm.crl	2k	crl version of qpm.crl (what else??!!?)
calc.c	12k	A reverse polish programmable calculator based somewhat on the HP11c.
e.c	2k	Program that does nothing but compute 'e' to some desired number of digits beyond the decimal...(is that like beyond the pale?)
m.c	2k	An implementation of Fermat's little theorem as a practical test of primality for microcomputers. (I'm not all that sure of just how practical!)
ratc.c	4k	A set of rational number functions (add, subtract, multiply and divide) as an extension to the system.
prime.c	2k	Need large prime numbers? This is a program that will generate them, up to 99 digits as is, more with modification to the source. For instance did you know that 782329335341236322564646443567896556379245731112 59 is prime? This program is a great eater of time so use it accordingly. Note also its use of

		a function from qpm.crl; qprime(s).
sqr1.c	2k	A demonstration of how an s100 computer and BDSc can compute 1.0000001^134217728 with 50 digits of accuracy. See the April 1984 issue of Scientific American's Computer Recreations column.
v.c	6k	In source a series of examples on vli function calls, as compiled, a demonstration of how they work.
p.c	2k	Algorithm P...a probabilistic primality test. From "Seminumerical Algorithms: The Art of Computer Programming", Vol. 2, by D. E. Knuth.
read.me	nk	this, and you are.

---

notes and misc.

Both vli.csm and math.csm need to be made better hence the contribution to the usergroup. There are a lot of things that I don't understand about writing tight efficient csm files, so feedback would be useful. I am concerned for instance in that the test versions of math and vli don't have better overflow and underflow return methods. Also local storage sizes are too small, but it seems unwieldly to change them all to 600 bytes. A little study here should indicate the correct size for each. Also if a form of global storage were possible from csm a great deal of storage overhead and some subroutine overhead as well could be dealt with.

Hugh S. Myers
208-342-4936
922 Pierce Court
Boise, Idaho
83712

A>

The Twilight Zone
Adventure Game

Hardware

The Twilight Zone Adventure Game requires a CP/M-compatible operating system with a Z80 processor and 32K of TPA space. This implies a RAM memory size of at least 40 to 48K.

Adventure Games

In case you are unfamiliar with adventure games (where have you been?), here is a quick rundown. An adventure game is an alternate universe in which you can do various things in order to achieve a certain goal. You will start in a given location at the beginning of the game. You can examine your surroundings and then move on to another location. Some adventure games require you to gather treasure and gain the maximum number of points by the end of the game. Others require you to perform a particular action in order to win. In any case, the outcome of the game is determined by where you go and what you do.

The Twilight Zone Game

You accomplish things in this game by giving two word sentences which consist of a verb and a noun.

You can move around by typing the verb GO followed by the direction you wish to go. The main directions used in the game are NORTH, SOUTH, EAST, WEST, UP, DOWN, IN, and OUT. You can also use one letter abbreviations (N, S, E, W, U, D, I and O) without a verb. There are a few other directions you can go, but you must learn these from playing the game.

It is very important to carefully assess your surroundings as you travel around. You can do this by using the words LOOK and EXAMINE followed by the name of the object to be looked at.

It will take you awhile to solve this game. Since it would be very tedious to start from the beginning every time you want to play the game a SAVE command is provided. Use this command when you are finished with a session or when you are about to do something that you think may not necessarily work out. At any time you can use the LOAD command to retrieve the earlier position in the game.

There are several other commands which can be used which you will discover as you play the game.

The object of the Twilight Zone Adventure Game is one of the first things that you must discover. Certain clues will only be given once, so you must pay close attention to all that goes on around you. If you get "stuck", try talking to a friend who is familiar with the game and take a close look at all of the clues that you have collected.

This game has a very large universe. It is essential that you make maps as you go along. Look out for traps and mazes which will be particularly hard to find your way around in and out of.

With these suggestions we hope that you find your trip to be a pleasant one. Remember the land in which you are travelling and that anything can happen. You are moving into a land of both shadow and substance, of things and ideas. You've just crossed over into ...

The Twilight Zone.

A>

CUG-DDJ#1:

The files on this disk are the first year's collection of "C/Unix Programmer's Notebook" columns which were written by Anthony Skjellum. They appeared in the September, 1983 through September, 1984 DDJ issues. Importantly, this also marks the introduction of the first C Users Group DDJ collection volume.

The text files (.DDJ) were written with Wordstar. Anyone lacking this program will need to reset bit 7 of each byte in the files so that they can be viewed correctly on any system.

Address of Doctor Dobb's Journal:

M & T Publishing, Inc.
2464 Embarcadero
Palo Alto, CA 94303
(415) 424-0600

Author of this volume:

Anthony Skjellum
% Pyramid Systems, Inc.
1695 Shenandoah Rd.
San Marino, CA 91108

Copyright notice:

A>
CUG-DDJ#2: (2nd Doctor Dobb's CUG Volume)
prepared by: A. Skjellum.
date: 29-Jul-84

This volume contains the article and related programs for the October, 1984, C/Unix Programmer's Notebook. The programs included are of three categories:

* general purpose routines (gpr.c)
* simple runge-kutta order 4 integration (rk4.c)
* system runge-kutta order 4 integration (rks.c)

Also three test programs are included which drive the runge-kutta software.

The text file (OCT84COL.DDJ) was written with Wordstar. Anyone lacking this program will need to reset bit 7 of each byte in the file so that it can be viewed intelligably on any system.

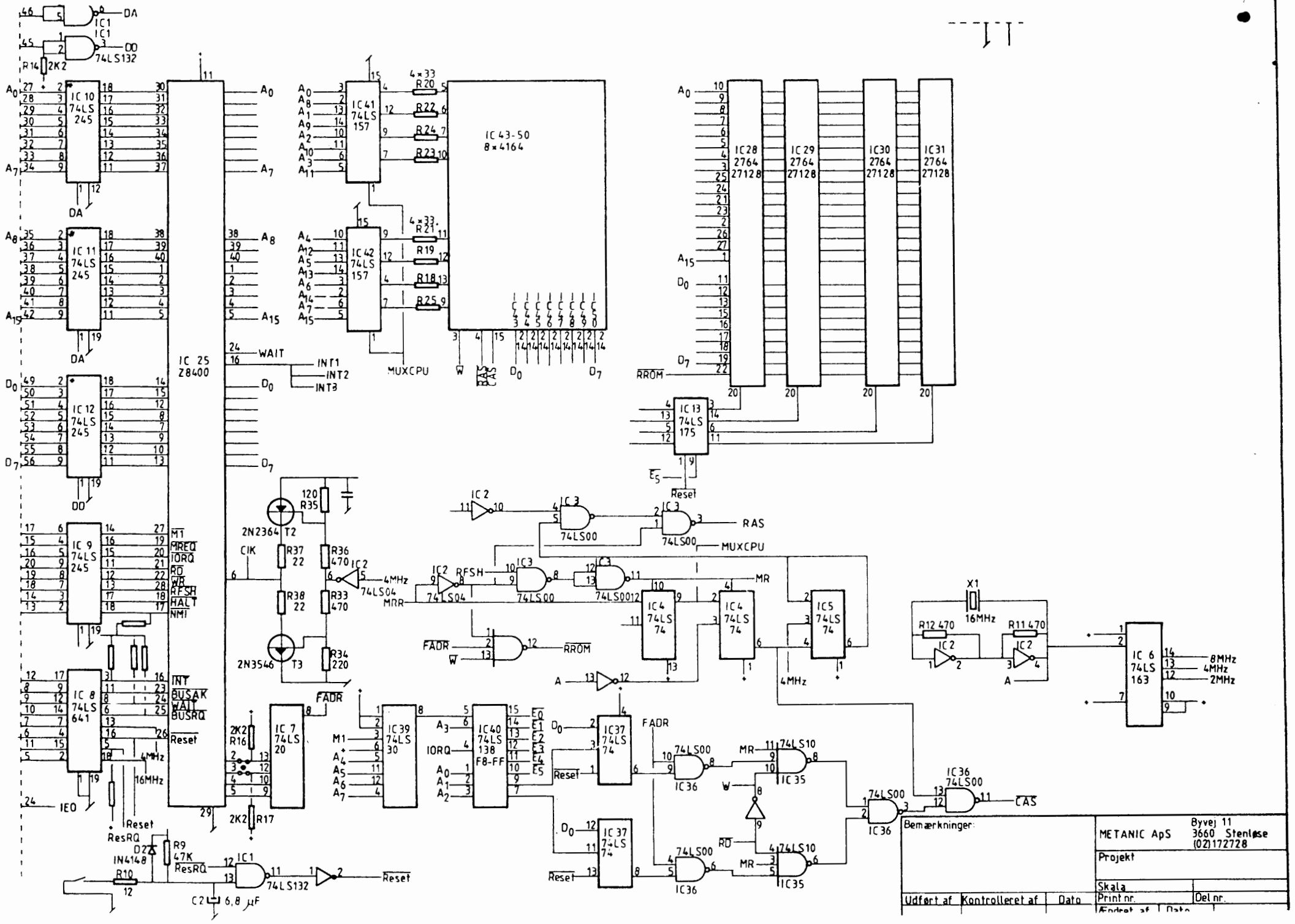
Address of Doctor Dobb's Journal:

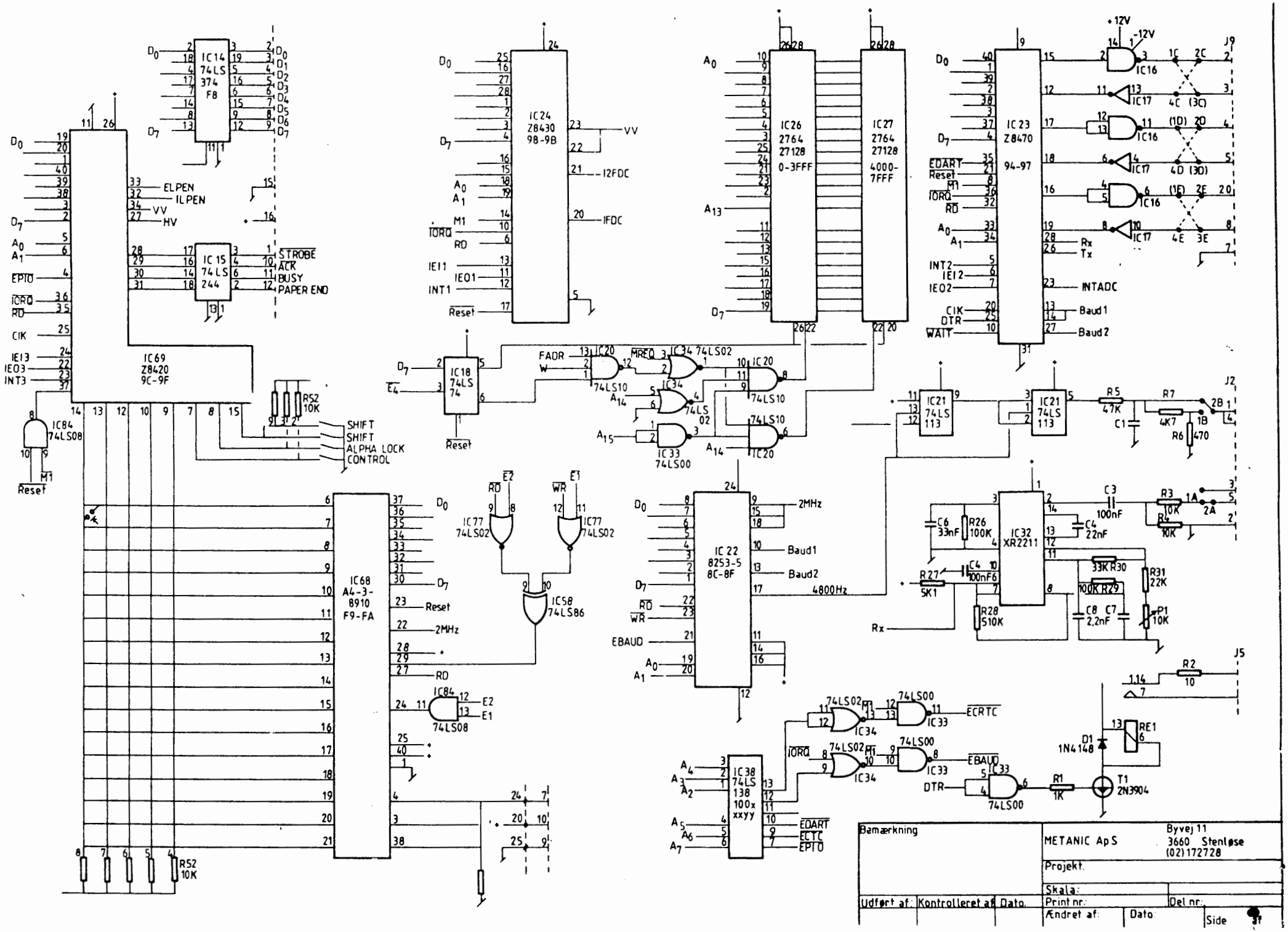
M & T Publishing, Inc.
2464 Embarcadero
Palo Alto, CA 94303
(415) 424-0600

Author of this volume:

Anthony Skjellum
% California Institute of Techology

```
A>user 1        C139 / C140 / C141
A>
A: ART2      DOC : ARTICLE  DOC : CRC       COM : CRCKLIST CRC
A: KAREL     H   : KAREL    SUB : KAREL1    C   : KAREL1   COM
A: KED       C   : KED      COM : KED       H   : KED2     C
A: MENU      DOC : README   DOC : README2   DOC : SESSIONS DOC
A: UGUIDE    DOC : VERSION  DOC
A>
A: --READ--  ME  : BACKUP   C   : BDSCAT    ALL : CLOCK    C
A: CLOCK     H   : CMODEM   C   : CRC       COM : CRCKLIST CRC
A: DATEDEMO  C   : DDTTOMAC C   : DDTTOMAC  DOC : FIXDIR   C
A: FIXDIR    DOC : KEY      C   : MARGIN    C   : MGAME    C
A: READ-ME   1ST : STRIP    C   : VERIFY    C   : WCT2     C
A: YAMBOOT   C   : ZCASM    DOC : ZCASM     SUB : ZCASM13  C
A>
A: CIRCLE    C   : CLOCK    C   : CLOCK     DOC : CMATH    C
A: CMATH     DOC : COEF     H   : CRC       COM : CRCKLIST CRC
A: EXPSPI    C   : FCNPLOT  C   : FGETSN    C   : FLOAT    H
A: FONT      DEF : GRAPH    DOC : HDWLIB    C   : LOGINFO  DTA
A: LOGON     C   : LOGON    COM : PMKLIB    C   : PMKLIB   DOC
A: PUNCT     C   : PUNCT    COM : READ      ME  : README   DOC
A: ROSE      C   : SPIRAL   C   : SWAPCOPY  COM : TRIGPLOT C
A>user 0
A>
```


IC 25
Z8400
IC 10
74LS
245
IC 11
74LS
245
IC 12
74LS
245
IC 9
74LS
245
IC 8
74LS
641
IC 7
74LS
20
IC 39
74LS
30
IC 40
74LS
138
F8-FF
IC 41
74LS
157
IC 42
74LS
157
IC 43-50
8×4164
IC 28
2764
27128
IC 29
2764
27128
IC 30
2764
27128
IC 31
2764
27128
IC 13
74LS
175
IC 37
74LS
74
IC 4
74LS
74
IC 5
74LS
74
IC 6
74LS
163
IC 35
IC 36
74LS00
74LS04
74LS10
74LS132
2N2364 T2
2N3546 T3
IN4148
X1
16MHz
4MHz
8MHz
2MHz
RAS
CAS
MUXCPU
RRÖM
FADR
MR
RESET
WAIT
INT1
INT2
INT3
IEO
ResRQ
BUSAK
BUSRQ
METANIC ApS
Byvej 11
3660 Stenløse
(02)172728
Projekt
Skala
Print nr.
Del nr.
Udført af
Kontrolleret af
Dato
Bemærkninger:



Bemærkning	METANIC ApS	Byvej 11 3660 Stenløse (02) 172728	
	Projekt:		
	Skala:		
Udført af: Kontrolleret af: Dato:	Print nr:	Del nr:	
	Ændret af: Dato:	Side	

De to almindeligste programmeringssprog til undervisningsbrug i Danmark er Basic og Comal. I de senere år har man fra udlandet hørt om andre programmeringssprog der påstås at være velegnede i en undervisningssammenhæng. Blandt disse høres ofte sprogene Prolog og Logo omtalt, og de har da også (så vidt jeg ved) været prøvet enkelte steder i Danmark. Denne note er en kort introduktion til disse to programmeringssprog.

Indhold

P R O L O G

L O G O

****

Artiklerne om Prolog og Logo, hvor første del bringes i dette nummer og sidste del i næste, er venligst stillet til rådighed af prof. H.B. Hansen, RUC.

****

PROLOG

Deklarativ programmering

Prolog er et programmeringssprog til såkaldt deklarativ programmering. Et Prologprogram til løsning af et eller andet problem beskriver nogle fakta om de objekter der indgår i problemet, og beskriver nogle relationer imellem disse, i det væsentlige uden at foreskrive hvordan problemet skal løses i detaljer. Der er tale om at beskrive et problem på en så detaljeret og utvetydig måde at det kan løses uden at foreskrive de enkelte skridt i løsningsalgoritmen, således som det er almindeligt i andre programmeringssprog (såkaldt imperativ programmering). Et Prologprogram er altså en meget nøjagtig specifikation af problemet, og selve løsningen af problemet sker så ved en i Prologsproget indbygget løsningsstrategi, som forøvrigt bygger meget stærkt på baksporingsmetoden, der er beskrevet i disse noters 3. del om lægning af et puslespil.

Et Prologprogram består af følgende tre sætningstyper:

1. Deklarering af fakta om objekter og deres indbyrdes relationer.
2. Deklarering af regler om objekter og deres indbyrdes relationer.
3. Spørgsmål om objekter og deres indbyrdes relationer.

Man kan opfatte udførelsen af et Prologprogram på den måde at alle deklarationerne af fakta og regler befinder sig i en database, og et spørgsmål giver da anledning til at denne database gennemsøges for at finde et svar på spørgsmålet.

Fakta

Et faktum kunne være: Peter ejer bogen. Dette skrives i Prolog således:

```
ejer(peter,bogen).
```

De to ord "peter" og "bogen" repræsenterer objekter, og "ejer" er den relation der gælder imellem dem. Et andet faktum kunne være:

```
elsker(peter,lone).
```

hvilket (f.eks.) udtrykker at Peter elsker Lone.

Rækkefølgen af objekterne i sådanne fakta er af betydning. Det er Peter der ejer bogen og ikke omvendt, og det fremgår ikke om Lone elsker Peter. Antallet af objekter i et faktum er ikke begrænset til to. Man

kunne tænke sig følgende faktum med kun ét objekt:

```
dreng(peter).
```

der udtrykker at Peter er en dreng; eller følgende faktum med tre objekter:

```
forældre(peter,marie,ole).
```

der f.eks. kan udtrykke det faktum at Peters mor hedder Marie og hans far hedder Ole.

I Prolog hedder objekterne argumenter, og relationsbetegnelserne hedder prædikater.

Regler

Lad os sige at Peter elsker alle mennesker. Så kunne vi naturligvis skrive alle fakta ned, således:

```
elsker(peter,lone).
elsker(peter,ole).
elsker(peter,marie).
.....
```

osv. for ethvert menneske i databasen (dvs. for alle de argumenter for hvilke der gælder prædikatet "menneske"). Ved hjælp af en regel kan dette udtrykkes kortere i Prolog:

```
elsker(peter,X) :- menneske(X).
```

Her er X en variabel. Variable kendes af Prolog ved at de begynder med et stort bogstav. Det er derfor jeg hidtil har skrevet "peter", "lone", etc. med lille begyndelsesbogstav. Tegnet :- læses "såfremt", og reglen udtrykker at Peter elsker et hvilket som helst objekt, såfremt dette objekt er et menneske.

Her er en mere kompliceret regel:

```
broder_til(X,Y) :- dreng(X),forældre(X,Mor,Far),forældre(Y,Mor,Far).
```

Denne regel udtrykker at et objekt X er broder til et andet objekt Y såfremt:

a) X er en dreng, og

b) X og Y har samme forældre.

Kommaerne imellem prædikaterne betyder altså "og".

Spørgsmål

Lad os antage at databasen indeholder følgende:

```
dreng(peter).
dreng(ole).
pige(lone).
pige(marie).
forældre(peter,marie,ole).
forældre(lone,marie,ole).
broder_til(X,Y) :-
   dreng(X),forældre(X,Mor,Far),forældre(Y,Mor,Far).
```

Vi ønsker nu at få at vide om Peter er bror til Lone. Derfor taster vi på Prologterminalen:

```
?-broder_til(peter,lone).
```

Dette er et spørgsmål i Prologsproget, og det får Prologsystemet til at udføre følgende proces:

1. Der søges i databasen efter et match på spørgsmålet. Herved forstås et faktum eller venstresiden af en regel, der har samme prædikat som spørgsmålet, og samme antal argumenter. Argumenterne sammenlignes parvis fra venstre mod højre. Hvis et par tilsvarende argumenter begge er konstanter, så skal de være ens for at få et match. Hvis derimod det ene argument er en variabel, så er der altid match.
   I vores tilfælde matcher
   ```
   broder_til(peter,lone)
   ```
   med
   ```
   broder_til(X,Y)
   ```
2. De variable i matchet bestemmes i overensstemmelse med konstanterne. I vores tilfælde betyder det at X får værdien peter og Y får værdien lone. Disse værdier indsættes i hele reglen. Den lyder nu:
   ```
   broder_til(peter,lone) :-
      dreng(peter),forældre(peter,Mor,Far),forældre(lone,Mor,Far).
   ```
3. Der sættes et mærke i databasen ved det faktum eller den regel der matchede. Et eventuelt senere forsøg på matching med samme prædikat vil starte lige efter dette mærke. I vores tilfælde mærkes broder_til reglen.
4. Hvis matchingen skete med et faktum returnerer Prolog med værdien "succes" til det sted hvor matchingen blev forsøgt. Hvis der derimod

L O G O

Lidt om Logo

Logo er et programmeringssprog der er specielt formgivet med henblik på undervisning af børn og unge. Der er tale om et sædvanligt imperativt sprog som f.eks. Basic og Comal, men Logo indeholder nogle faciliteter der gør dette sprog ret så kraftfuldt i forhold til Basic og Comal. Disse faciliteter er:

1. Logo indeholder sætninger der gør det let at arbejde med en grafisk skærm ved hjælp af den såkaldte skildpaddegrafik.
2. Logo indeholder datastrukturen en liste som en på forhånd kendt datastruktur i sproget. Notationen er lidt afvigende i forhold til den vi har set for Prolog, men det er den samme datastruktur det drejer sig om.
3. Logo tillader generel rekursivitet i algoritmerne.

Denne note om Logo koncentrerer sig om grafikken i sproget. Man kan sige at denne grafik repræsenterer et nyt syn på geometrien. Traditionelt har geometri oftest været opfattet som en "statisk" disciplin hvor man analyserer færdige figurer, men Logo anlægger et "dynamisk" syn. Det bliver selve processen at tegne en figur der bliver det interessante. På ethvert tidspunkt er figuren i en vis tilstand, og det er denne tilstand man kan ændre ved hælp af kommandoer i sproget.

Et andet karakteristisk træk ved Logo er at det bliver det naturligste at beskæftige sig med de "lokale" egenskaber ved de geometriske objekter, i stedet for de "globale" egenskaber som de f.eks. kommer til udtryk i den analytiske geometris ligninger. Den globale orientering af planen - f.eks. ved hjælp af et koordinatsystem - er en unaturlig betragtningsmåde i Logo, hvor alt er relativt til den øjeblikkelige tilstand.

Skildpaddegrafik

På en Logoskærm findes der en lille trekantet figur der hedder en skildpadde. Skildpadden kan bevæges rundt på skærmen ved hjælp af visse Logokommandoer. F.eks. vil kommandoen:

```
forward 2
```

bevirke at skildpadden bevæger sig 2 enheder fremad efter næsen, mens:

```
right 30
```

vil bevirke at skildpadden drejer sig 30 grader mod højre i forhold til sin egen orientering. Kommandoerne "back" og "left" får skildpadden til

at kravle baglæns og dreje til venstre.

Skildpadden kan trække en streg efter sig når den bevæger sig. Det vil ske efter afgivelse af kommandoen "pendown". Efter afgivelse af kommandoen "penup" kommer der derimod ingen streg når skildpadden bevæger sig.

Programmer der bevæger skildpadden og tegner figurer på skærmen kan udformes som procedurer. En procedure til tegning af en ligesidet trekant kan f.eks. se således ud:

```
to trekant
   pendown
   repeat 3
     forward 100
     right 120
   end
end
```

Ordet "to" er signalet til at der nu kommer en procedure. Bemærk at man måler vinkler som ydre vinkler i Logo. Kommandoen "right 60" i stedet for "right 120" i trekantproceduren ville føre til et resultat som vist på fig. 1. Som det ses kan man programmere løkker i et Logoprogram på næsten samme måde som i f.eks. Comal (repeat). Man kan angive antallet af gange der skal repeteres som vist i "trekant", men der findes også en "repeat - until" sætning i Logo.

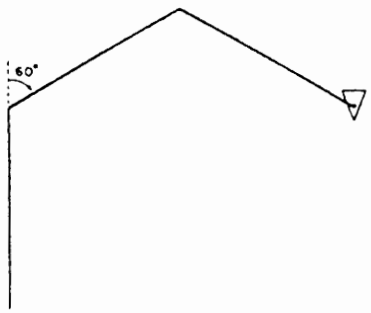


Fig. 1

Procedurer kan have parametre. De skrives i parentes efter procedurens navn. På nogle Logosystemer skal der ikke parentes om parametrene, men den slags pedantiske detaljer vil vi som sædvanlig se stort på. En procedure må gerne kalde sig selv rekursivt. Et eksempel:

```
to polygon(side,vinkel)
   forward side
   right vinkel
   polygon(side,vinkel)
end
```

Fig. 2 viser eksempler på polygoner der kan tegnes med denne procedure for forskellige værdier af "side" og "vinkel". Størrelsen af "side" er ikke så interessant, idet den blot bestemmer figurens udstrækning, men forskellige værdier af "vinkel" vil resultere i meget forskellige figurer, som det fremgår af fig. 2.

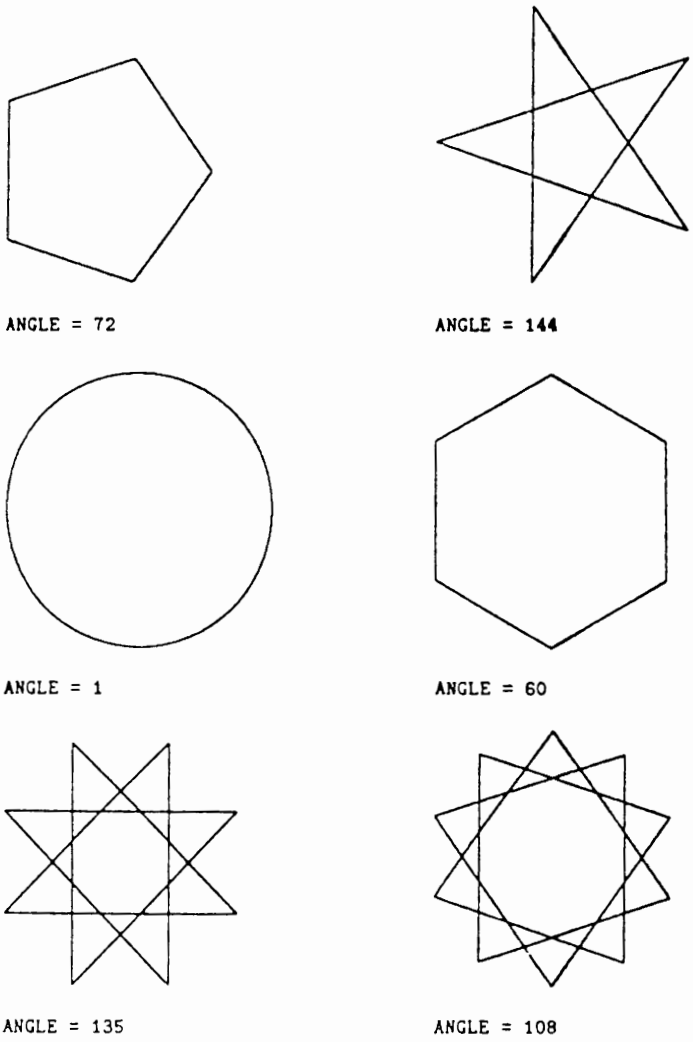


Fig. 2

Proceduren "polygon" vil aldrig stoppe, men skildpadden vil løbe rundt og rundt i sin egen bane når proceduren først er startet. Det er imidlertid ligegyldigt for vores formål. Cirklen på fig. 2 er f.eks. startet med følgende lille program:

```
pendown
polygon(1,1)
```

og den er altså egentlig en regulær 360-kant. Hvor "pæn" den bliver afhænger af den grafiske skærms opløselighed, dvs. hvor mange punkter den indeholder. Det er nemlig sådan at afstanden 1 som regel er defineret som "cirka" afstanden mellem nabopunkter på en rasterskærm (i virkeligheden er det et vanskeligt problem, idet hvert punkt jo har 8 nabopunkter, men de har lidt forskellig afstand fra punktet - vandret/lodret og diagonalt; det vil jeg dog ikke gå nærmere ind i her).

Polygon-teoremet

Jeg vil nu vise at der må gælde følgende teorem:

> Den rute der tegnes af proceduren "polygon" vil lukke sig når den totale vinkeldrejning når et helt multiplum af 360 grader.

Lad skildpadden starte i punktet P_o med retningen R_o. Hver gang en ny inkarnation af "polygon" bliver kaldt befinder skildpadden sig i et nyt punkt P_i, hvis afstand fra det forrige punkt P_{i-1} er lig med "side", og den har en ny retning R_i der er lig med R_{i-1}+ vinkel, idet vi (arbitrært) regner vinkler positivt mod højre. Når den samlede vinkeldrejning modulo 360 grader efter n rekursioner bliver 0, må skildpadden igen have retningen R_o, men nu befinder den sig i et punkt P_n. Vi skal vise at $P_o=P_n$.

Hvis $P_o \neq P_n$ kan vi foretage endnu n rekursioner, og så vil skildpadden være i et punkt P_{2n} der forholder sig til P_n ligesom P_n forholder sig til P_o. Specielt må de tre punkter ligge på samme rette linie L. Ved fortsat kørsel med programmet vil skildpadden altså bevæge sig mod det uendelige, idet den dog hele tiden holder sig i en endelig afstand fra linien L.

Vi går nu tilbage til starten af processen med startbetingelse (P_o,R_o). Efter præcis 1 rekursion vil tilstanden være (P_1,R_1), og hvis side$\neq$0 og vinkel$\neq$0 må $P_1 \neq P_o$ og $R_1 \neq R_o$. P_1 vil ikke nødvendigvis ligge på linien L. Efter yderligere n rekursioner vil skildpadden igen have retningen R_1, og den vil befinde sig i et punkt P_{n+1} der ligger på en ret linie L_1 gennem P_1, som ikke nødvendigvis falder sammen med L, men tværtimod i almindelighed danner en vinkel med L. Men nu er der modstrid, for skildpadden kan ikke være i nærheden af to skærende linier på samme tid, når den fjerner sig i det uendelige. Derfor må det være forkert at $P_o \neq P_n$.

Dette var et eksempel på et skildpaddebevis. Lad os tage et andet et med det samme. Et Logoprogram der tegner en trekant med siderne a, b og c, og vinkelspidserne A, B og C må være noget i retning af følgende:

```
to trekant
   forward a
   right 180-A
   forward b
   right 180-B
   forward c
   right 180-C
end
```

Men da skildpadden har foretaget en total drejning på 360 grader må man have:

$$(180-A) + (180-B) + (180-C) = 360$$

TIPS TIL BUTLEREN.

Tip 1 : Ved opstart starter cursoren et stykke inde på skærmen. Ønskes en smallere venstre margin kan følgende bruges :

```
10 OUT 138,2        // Valg af register
20 OUT 139,97       // Valg af venstre margin
```

Herved rykkes 4 karakterer til venstre. Ved opstart haves værdien 93, øges tallet rykkes til venstre, mindskes tallet rykkes mod højre
De viste instruktioner kan evt. lægges i en COMAL80I.NIT fil.

Tip 2 : Laver du selvdefinerede tegn v.hj.a. ESC G koden og gemmer disse som strengvariable, kan de ikke udskrives til højre på skærmen, da variablen skal dimensioneres til mindst 14 (jvf. BUTLER manualen s. 6.7).
Dette kan klares ved POKE addr, b , hvor addr er en adresse hvis værdi afhænger af comalversionen, mens b er det antal karakterer, der skal kunne skrives på en linie (0<b<255).
Husk at afslutte med POKE addr, 80 .

Værdierne for comal-80 ver. 2.03 er flg. :

COMAL-80	:	40169
COMAL80S	:	26758
COMAL80D	:	40520
CMAL80DS	:	27097

Eksempel jvf. BUTLER manualen s. 6.7 i COMAL-80 :

```
10 DIM tegn$ of 14
20 tegn$ = ""27"G"+chr$(255)+chr$(129)+chr$(165)+
           chr$(129)+chr$(153)+chr$(129)+chr$(165)+
           chr$(153)+chr$(129)+chr$(255)+chr$(0)+
           chr$(0)
30 CURSOR 75,10
40 PRINT tegn$
50 POKE 40169, 120
60 CURSOR 75,10
70 PRINT tegn$
80 POKE 40169, 80
90 END
```

Et lille program til på let måde at definere tegn svarende til både 40 og 80 karakterer pr. linie kan rekvireres hos undertegnede. Formatteret diskette og frankeret svarkuvert bedes vedlagt. Tegnene gemmes på disketten og kan derved bruges i andre programmer. Programmet er lavet både i COMAL-80 og PASCAL.

Tip 3: Venstre margin ved skærmdump er mulig på flg. måde på TAXAN- og EPSON-printere :

```
10 SELECT OUTPUT "lp:"
20 PRINT ""27"l";chr$(kar) // kar er den ønskede margin-
                              bredde i karakterer.
30 PRINT                   // (skrivehovedet placeres)
40 SELECT OUTPUT "ds:"
50 PRINT ""27"H";chr$(opl) // Dump med opløseligheden opl
60 END
```

Jeg har ikke haft mulighed for at undersøge muligheden på andre printere.

Tip 4: I pascal foreslås potensopløftning foretaget på flg. måde i diverse lærerbøger :

a^x = EXP(x * LN(a))

Denne måde er temmelig langsom, men hvis x er heltallig kan flg. funktion anvendes :

```
FUNCTION potens(a:real; x:integer) : real;
VAR
  p:real;
BEGIN
  p:=1;
  IF n<0 THEN
    BEGIN x:=1/x; n:=-n; END;
  REPEAT
    IF n MOD 2 = 1 THEN p:=p*x;
    x:=x*x; n:=n DIV 2
  UNTIL n=0;
  potens:=p
END;
```

John Hansen
Læborggård 9
6600 Vejen
Tlf. 05-368081