PD

Program Descriptions I

for HOFACKER Software
for your ATARI Computer

IMPORTANT:
Do not use the delivered disks for your own files. It will destroy itself!

First Edition
First Printing
December 1982 in the Federal Republic of Germany

Publisher:
Ing. W. HOFACKER GmbH, Tegernseerstr. 18, D-8150 Holzkirchen, W.-Germany

US-Distributor
ELCOMP PUBLISHING,INC., 53 Redrock Lane, Pomona, CA 91766

PD

Program Descriptions I

for HOFACKER Software for your ATARI Computer

PREFACE

This book contains the descriptions of our software products for the ATARI 800 and ATARI 400.
You can find the descriptions you need by looking through the contents of this booklet.

Los Angeles
December 1982

TABLE OF CONTENTS

Printer Interface

Order-No. 7211 $19.95

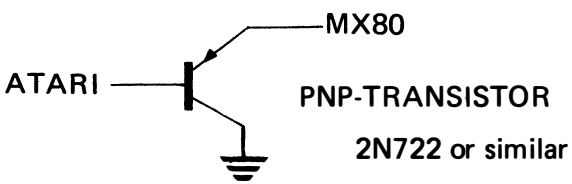
Screen to Printer Interface for the ATARI 400/800
Many ATARI users would like to connect a parallel interface to the computer. For many people buying an interface is too expensive. On the other hand, they may not have the experience to build one by their own. Also a lot of software is needed.

The following instructions make it easy, to hook up an EPSON, Centronics or an Okidata printer to the ATARI.
Only seven of the eight bits of the data link are used for a printout. The eight bit creates a strobe impulse. Also the trigger input of port 4 is used for the BUSY-request of the printer.
There is a formfeed every 66 print lines. So it is necessary to adjust the paper before starting the printing. You may need to make several trials to find the best position of the paper. After each system reset the line counter is set to zero, so you have to privide your own formfeed for a correct paper position.

You can control the length of a line by a POKE 1770, xxx. After doing so, press system reset and enter LPRINT.

The program SCREENPRINT is called by BASIC thru an USR (1670) and by the assembler with a GOTO $0687.

You may install pnp transistors between the game output and the printer, as it is shown in this small figure.



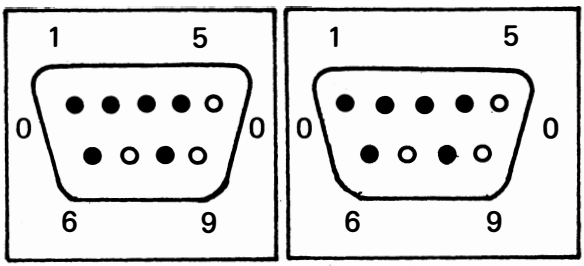
The next figure shows the connection of the ATARI game outlets and the connector for the MX-80 printer. This is a so-called Centronics interface and the program can be used with each printer and this interface.

EPSON MX80 – ATARI 400/800
Interconnection-Scheme

MX80-Connector	ATARI-Connectors	
	Port3	Port 4
Pin#	Pin#	Pin#
1 (19) $\overline{\text{STROBE}}$		4
2 (20) DATA 1	1	
3 (21) DATA 2	2	
4 (22) DATA 3	3	
5 (23) DATA 4	4	
6 (24) DATA 5		1
7 (25) DATA 6		2
8 (26) DATA 7		3
9 (27) DATA 8		8
11 (29) BUSY		6
(GND)	8	8

(19)—(29) = Ground (GND)

Plugs seen from the rear view.
Front view of the computer outlets.



PORT 3 PORT 4

The next figure shows the program.

```
*********************************
*  UNIVERSAL PRINT FOR ATARI    *
*                               *
*  400/800 VERSION ELCOMP       *
*                               *
*                               *
*                               *
*                               *
*                               *
*********************************

               BASIS     EPZ $58
               PT        EPZ $FE
               PST       EQU $600

                         ORG PST

0600: 00                 DFB 0
0601: 02                 DFB 2
0602: 0006               DFW PST
0604: 6E06               DFW INIT
0606: A93C               LDA #$3C
0608: 8D02D3             STA $D302
060B: A9EB               LDA #PND
060D: 8DE702             STA $02E7
0610: A906               LDA #PND/256
0612: 8DE802             STA $02E8
0615: A96E               LDA #INIT
0617: 850A               STA $0A
0619: A906               LDA #INIT/256
061B: 850B               STA $0B
061D: 18                 CLC
061E: 60                 RTS

061F: 2B0642
0622: 063F06
0625: 42063F
0628: 063F06 HANDLTAB DFW DUMMY,
       WRITE-1,RTS1-1,WRITE-1,RTS1-1,
       RTS1-1
062B: 01      DUMMY    DFB 1
```

```
062C: A930    OPEN      LDA #$30
062E: 8D03D3            STA $D303
0631: A9FF              LDA #$FF
0633: 8D01D3            STA $D301
0636: A934              LDA #$34
0638: 8D03D3            STA $D303
063B: A980              LDA #$80
063D: 8D01D3            STA $D301
0640: A001    RTS1      LDY #1
0642: 60                RTS
0643: C99B    WRITE     CMP #$9B
0645: D01D              BNE PRINT
0647: ADEA06  CARR      LDA LINLEN
064A: 8DE906            STA LCOUNT
064D: CEE806            DEC COUNT
0650: 100D              BPL NOFF
0652: A90C              LDA #12
0654: 206406            JSR PRINT
0657: EEE906            INC LCOUNT
065A: A941              LDA #65
065C: 8DE806            STA COUNT
065F: EEE906  NOFF      INC LCOUNT
0662: A90D              LDA #13
0664: 20D106  PRINT     JSR OUTCHAR
0667: CEE906            DEC LCOUNT
066A: F0DB              BEQ CARR
066C: D0D2              BNE RTS1
066E: A91F    INIT      LDA #HANDLTAB
0670: 8D1B03            STA $031B
0673: A906              LDA #HANDLTAB/256
0675: 8D1C03            STA $031C
0678: A941              LDA #65
067A: 8DE806            STA COUNT
067D: ADEA06            LDA LINLEN
0680: 8DE906            STA LCOUNT
0683: 4C2C06            JMP OPEN

0686: 68      BASIC     PLA
0687: A558    NORMAL    LDA BASIS
0689: 85FE              STA PT
068B: A559              LDA BASIS+1
068D: 85FF              STA PT+1
068F: A917              LDA #23
```

```
0691: 8DE606             STA ROW
0694: A927    ROWLOOP    LDA #39
0696: 8DE706             STA COLUMN
0699: A200               LDX #0
069B: A1FE    LOOP       LDA (PT,X)
069D: 297F               AND #$7F
069F: C960               CMP #$60
06A1: B002               BCS LOOP1
06A3: 6920               ADC #$20
06A5: 20D106  LOOP1      JSR OUTCHAR
06A8: E6FE               INC PT
06AA: D002               BNE *+4
06AC: E6FF               INC PT+1
06AE: CEE706             DEC COLUMN
06B1: 10E8               BPL LOOP
06B3: A90D               LDA #13
06B5: 20D106             JSR OUTCHAR
06B8: CEE606             DEC ROW
06BB: 10D7               BPL ROWLOOP
06BD: 60                 RTS

06BE: 48414E
06C1: 532057
06C4: 41474E
06C7: 455220
06CA: 32372E
06CD: 372E38
06D0: 31 AUTHOR    ASC "HANS WAGNER

06D1: AC13D0 OUTCHAR  LDY $D013
06D4: D0FB            BNE OUTCHAR
06D6: A080            LDY #$80
06D8: 0980            ORA #$80
06DA: 8D01D3          STA $D301
06DD: 297F            AND #$7F
06DF: 8D01D3          STA $D301
06E2: 8C01D3          STY $D301
06E5: 60              RTS

06E6: 17      ROW     DFB 23
06E7: 27      COLUMN  DFB 39
06E8: 41      COUNT   DFB 65
06E9: FF      LCOUNT  DFB 255
```

```
06EA: FF      LINLEN    DFB 255
              PND       EQU *

BASIS         $58
PT            $FE
PST           $0600
HANDLTAB      $061F
DUMMY         $062B
OPEN          $062C
RTS1          $0640
WRITE         $0643
CARR          $0647
NOFF          $065F
PRINT         $0664
INIT          $066E
BASIC         $0686       UNUSED
NORMAL        $0687       UNUSED
ROWLOOP       $0694
LOOP          $069B
LOOP1         $06A5
AUTHOR        $06BE       UNUSED
OUTCHAR       $06D1
ROW           $06E6
COLUMN        $06E7
COUNT         $06E8
LCOUNT        $06E9
LINLEN        $06EA
PND           $06EB
```

Program description:

Address	
0600 – 061E	end of the booting part
0610 – 062B	HANTAB for the ATARI OS
062C – 0642	opens the ports for output
0643 – 066D	printer driver
066E – 0685	Initialize. Now LPRINT and PRINT "P" uses the printer driver
0686 – 06BD	label BASIC starting address for a call by BASIC Label NORMAL starting address for a call by assembler.

06BE – 06D0	Copyright notice
06DL – 06E5	Subroutine, puls one ASCII character from the accumulator to the printer
06E6 – 06EA	values for the various counters ROW sets the number of horizontal lines to 23. COLUMN sets the number of characters of one line to 39. COUNT sets the number of lines between two formfeeds to 65 LCOUNT, LINLEN contains the actual parameters for the number of characters and lines.

Boot-Routine

```
PST       EQU $0600
PND       EQU $0700
FLEN      EQU PND-PST+127/128*128
                      ORG $6000

6000: A210    BOOTB     LDX #$10
6002: A903              LDA #3
6004: 9D4203            STA $0342,X
6007: A908              LDA #8
6009: 9D4A03            STA $034A,X
600C: A980              LDA #$80
600E: 9D4B03            STA $034B,X
6011: A94A              LDA #CFILE
6013: 9D4403            STA $0344,X
6016: A960              LDA #CFILE/256
6018: 9D4503            STA $0345,X
601B: 2056E4            JSR $E456
601E: 3029              BMI CERR
6020: A90B              LDA #$0B
6022: 9D4203            STA $0342,X
6025: A900              LDA #PST
6027: 9D4403            STA $0344,X
602A: A906              LDA #PST/256
602C: 9D4503            STA $0345,X
602F: A900              LDA #FLEN
```

```
6031: 9D4803              STA $0348,X
6034: A901                LDA #FLEN/256
6036: 9D4903              STA $0349,X
6039: 2056E4              JSR $E456
603C: 300B                BMI CERR
603E: A90C                LDA #$0C
6040: 9D4203              STA $0342,X
6043: 2056E4              JSR $E456
6046: 3001                BMI CERR
6048: 00                  BRK
6049: 00        CERR      BRK
604A: 433A      CFILE     ASC "C:"
604C: 9B                  DFB $9B

PST             $0600
PND             $0700
FLEN            $0100
BOOTB           $6000     UNUSED
CERR            $6049
CFILE           $604A
```

The program on cassette comes as a bootable driver and you can use it either with the ATARI BASIC ROM or the Editor/Assembler cartridge from ATARI.
You can print via your interface with LIST "P:
and use the other PRINT command as described in your manuals from ATARI®.
How to load the cassette:

Turn off the computer
Press the start key
Turn on the computer
Release the start key
Press PLAY on the recorder and
Press RETURN

Order-No. 7291 $19.95

PRINTING VIA THE RS232 INTERFACE

The file RS232.SYS on your disk allows you to use the ELCOMP RS232-interface. The screen will flicker during operation, but don' t pay any attention to that.

The interface only works with 300 Baud. For the connections see figure below.

After the file was loaded you can use the DOS-command B to get to a cartridge, if there' s one installed.

After RESET the file has to be loaded again.
The following BASIC program:

```
10 OPEN #1,8,0,"R:"
20 FOR X=1 TO 10
30 PRINT #1,"ELCOMP-RS232",X
40 NEXT X
50 CLOSE #1
```

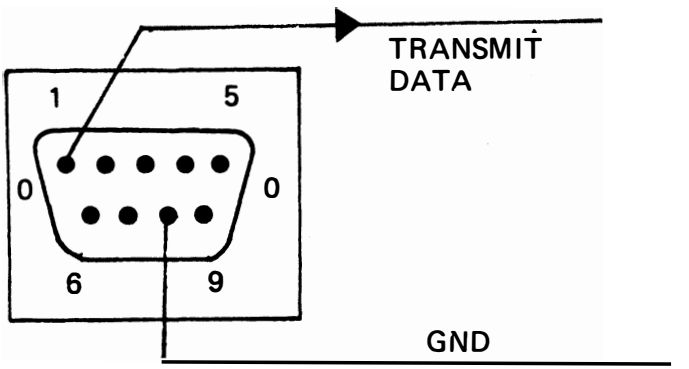
will generate the following printout:

```
ELCOMP-RS232    1
ELCOMP-RS232    2
ELCOMP-RS232    3
ELCOMP-RS232    4
ELCOMP-RS232    5
ELCOMP-RS232    6
ELCOMP-RS232    7
ELCOMP-RS232    8
ELCOMP-RS232    9
ELCOMP-RS232    10
```

The program on cassette comes as a bootable driver and you can use it either with the ATARI BASIC ROM or the Editor/Assembler cartridge from ATARI. You can print via your interface with LIST"R: and use the other PRINT command as described in your manuals from ATARI®.

How to load the cassette:

- **Turn off the computer**
- **Press the start key**
- **Turn on the computer**
- **Release the start key**
- **Press PLAY on the recorder and**
- **Press RETURN**



GAME PORT 3

IMPORTANT:
Do not use the delivered disks for your own files. It will destroy itself!

ATMASD

Order-No. 7099	**disk version**	**$ 89.00**
Order-No. 7999	**cartridge vers.**	**$129.00**

ATMASD

Disk-Version

Instructions — Working with the MACRO-ASSEMBLER

1. Remove all cartridges
2. Insert the disk into drive 1 of your ATARI 800/48K RAM
3. Turn on the console, the system will start to boot. After booting, the DOS menue will appear.
4. Type L for binary LOAD and load the file named ATMASD.OBJ
   After loading the ATMAS Machinelanguage-monitor starts automatically.
5. Typing K for a Coldstart brings you directly into the Editor of the Assembler.

How to get back to DOS
a) from the monitor ATMASD–1: by typing Q
b) from the Editor: by typing M in the command line followed by ESCAPE/ESCAPE
c) System Reset

The RUN Assembler Command from the DOS menue can be used only when the editor/assembler objectcode is loaded. Run Assembler initiates the monitor. Typing E with the object-code in memory then causes a warmstart.

The ATMASD Printer options

For the convenience of the user:,
The editor/assembler and the machinelanguage-monitor are equipped with printer routines for three different devices:

1. Using a printer in the machinelanguage-monitor
   a) YES (Y) after PRINT? gives you three options.
      (1) Output to printer via serial port of the ATARI RS232 Interface
      (2) Output via the parallel port of the ATARI Interface
      (3) Output via the expansion from Elcomp Publishing, Inc.

2. Using a printer in the Editor
   Enter the command line by typing ESCAPE and type L ESCAPE/ESCAPE. This command lists the source code to the screen only. With the CTRL 1 you can stop the listing and restart it again.
   L followed by one of the following numbers brings the source code to your printer via:
      (0) RS232 Port 1 of the ATARI Interface
      (1) Parallel port of the ATARI Interface
      (2) Port-Interface from Elcomp

3. Using a printer in the assembler
   An assembly listing on the printer can be generated by placing the pseudo opcode OUT at the beginning of the source listing.
   Example:
   OUT [Option] P u
   Options: LN (Listing and symboltable)
   LNM not expanded Macros
   u ist the Device as mentioned above.

   0 = RS232
   1 = Parallel
   2 = ELCOMP

This description applies also to
ATAS-1 32K / 48K
Order No. 7098
Order No. 7998
ATAS-1 is identical to ATMASD, but ATAS-1 has no macro capability.

Editor/Assembler for ATARI 800
32k or 48k

Version 1.1. 17.12.81 by Hans-Christoph Wagner

User Manual

Editor and Assembler are delivered as a combined machine language program on cassette or diskette. If desired, they can also be run as stand-alone programs. For example, the Editor could also be effectively used for higher programming languages like PASCAL etc.

The Editor and Assembler as delivered can be run directly on the 48k or 32k - Version. ATMAS runs on 48k Disk only. Please specity your system when you order.

The Editor and Assembler programs are stored beginning at location 0700 (Cassette Version).

1. EDITOR

The Editor partitions the screen display area into three parts:

– 'Status Line' = fist line on top
– 'Text Window' = mid-area of the screen
– 'Command line' = dotted line on bottom

1.1. Status Line

On the Status Line the actual Editor status is displayed in the status line on top of the screen.

P : nnnnn	display the number of bytes from beginning of text to cursor position.
T : nnnnn	shows the number of bytes still free in the textbuffer.
C : nnnnn	displays the number of bytes still free in the C-register.

The last two characters display either C-register status or an error message.

1.2. Text Window

The Text Window occupies the mid-area of the screen and comprises 21 lines. When executing an editing operation the result is displayed immediately in the Text Window. The cursor can be moved backwards and forwards as well as linewise up and down (scrolling-up and scrolling-down).

1.3. Command Line

The commands for the Command Mode operations of the editor have to be written into this line (see 1.4. 2).

1.4. Modes of operation

The Editor can be operated in two different modes: Direct mode and Command mode.

1.4.1. Direct Mode

In the Direct Mode, text can be written into the textbuffer, i. e. when thext is typed in, it is displayed in the Text Window on screen and simultanously stored in the textbuffer. Every character is inserted at the actual cursor positon on screen.

When pressing a key longer than one sec., the key-function will be repeated about eight times/second. This repeat function works as long as the cursor remains blinking i. e. only within the Text Window an screen.

For cursor moving, text manipulation etc., a number of direct commands are available. They are all initiated by pressing the CTRL-Key plus the command assigned 'Character key' and directly executed.

There are the following direct commands:

CTRL – A	Cursor one position forward
– C	Causes switching between two modes when consecutively opplied: lowest possible cursor position within Text Window either on bottom line or Sifth line from bottom
– D	Cursor to text end
– E	Cursor to text beginning
– F	Closing C-register
– G	Reexecute command on Command Line
– H	Delete one character back
– I	'TAB': cursor to next Tab position
– J	Insert content of C-Register at actual cursor position
– K	Erase C-Register
– L	'Formfeed': inserts CTRL-'L'
– P	Jump into ATMONA–1 Monitor. When you want to jump back to the Editor you have to key in E. You then return to the same Editor status as before. The cassette version automaticly starts in the ATMONA-Monitor. Hit the K-key for coldstart of the Editor / Assembler.

The cassette version automatically starts in the ATMONA-Monitor. Hit the K-key for coldstart of the Editor/Assembler.

– Q Cursor one position backward

– R Open C-Register

– S Cursor to beginning of next line

– T Control display; all control characters are displayed 'reverse' (switched on/off by applying the command consecutively)

– U Delete one character forward

– V Full Line Mode. The first 76 characters of a line are shown; a maximum of 10 lines are displayed. The mode can be switched on/off by applying the command consecutively.

– W Cursor to begin on same line or preceding line.

– X Deletes current line from beginning till cursor position/or whole previous line (when cursor is placed at first position of the line).

– Y Jump into Assembler

– Z Insert CTRL-'Z' (Assembler stop sign)

ESCAPE–KEY Opens Command Line when entering the Command Mode

– ↑ Deletes all text from text-begin till the actual cursor position.

1.4.2. Command Mode

With 'ESCAPE' one enters the Command Mode by opening the Command Line. ESCAPE is displayed as a $ sign. Commands to be executed are written into the Command Line. For the numerous commands in this mode, different single characters (not shifted!) are assigned.

The following 'Command Mode' options are available:

@ n : Set Tabulator to value n

B : Cursor one position back

D : Delete one character back

E : Erase C-Register

F : Cursor one position forward

G : Insert C-Register at actual cursor position

H ⟨ byte ⟩ : Insert Hex byte at actual cursor position .

I ⟨ string ⟩ : Insert ASCII string at actual cursor position

J : Jump to beginning of Command Line (i. e. reexecute whole Command Line)

K : Erase textbuffer (!!!)

L : Lists text on screen .

L1 : Lists text on Screen and printer via Microtronics Interface.

If you want to print via the joystickinterface 3 + 4, please connect your Centronics printer according to the following figure.

EPSON		ATARI	
		PORT3	PORT4
PIN#		PIN#	PIN#
1 (19)	STROBE		4
2 (20)	DATA 1	1	
3 (21)	DATA 2	2	
4 (22)	DATA 3	3	
5 (23)	DATA 4	4	
6 (24)	DATA 5		1
7 (25)	DATA 6		2
8 (26)	DATA 7		3
9 (27)	DATA 8		8
11 (29)	BUSY		6
(GND)		8	8

R : Read text file from cassette or diskette

ATMAS needs
R ⟨ Device ⟩ [NAME] ESC (Example: R D1 : FILE) Reads Source File from Disk 1 with the name "FILE". Extensions .SRC will be added automatically. If you want to read from cassette R C : ESCAPE
Reads next source file from cassette.
The text will be inserted at the actual cursor position. (Chaining, appending).

S ⟨string⟩ : Search for 'string' (ASCII-string) in text between actual cursor position and text end.

T : Delete all characters between actual cursor position and begin of next line.

U : Jump to user program starting on location $A800 hex ($A800 for disk and $8000 for cartridge).

W : Write text from text beginning till actual cursor position on cassette or diskette refer to Read Command

Syntax legend:

n : numbers 1 to 9 are allowed; default value of program as delivered is preset to 9

byte : Hex byte between 00 and FF

string : ASCII string ended by 'ESCAPE', same as with name

: Space; obligatory as delimiter when a name is used; otherwise optional

⟨⟩ : [ ] = optional

In the Command Mode there are additionally two Direct Mode commands allowed:

CTRL-X : Erase Command Line and jump back to Direct Mode

CTRL-H : Delete last character on the Command Line

Different commands can be chained in the Command Line. They can be separated by 'ESCAPE' as delimiter is obligatory; in all other cases it can be used optionally. If an integer number n between 2 and 255 is set before a command, it will be executed n times; excepted are the commands R, W and Z. When finally 'ESCAPE' is keyed in twice the commands on the Command Line are immediately executed. The Editor prompts the execution by displaying a # sign at the latest position on the Command Line and switches automatically over to Direct Mode.

For example, assume that the following commands have been written to the Command Line: ($=display form of 'ESC')

$2SINT$3DITESTJ($) ($) = replaced by prompt # after execution

Starting from the actual cursor position, a search for 'INT' ist started twice. When 'INT' is found the second time three characters in sequence will be deleted and 'TEST' inserted back. This whole operation will be repeated in the text which follows over and over till the search for 'INT' is exhausted, which is doubly indicated by the error message S? (see below) at the end of the Status Line.

In short, one can see in this example that every second 'INT' string in the text between the actual cursor position and the text end will be replaced by 'TEST'.

1.5 Status and Error Messages

The Status and Error Messages are displayed at the last two positions on the Status Line on top.

Status Messages:

OK : normal status; C-Register closed

CR : C-Register open

Error-Messages:

RW : illegal Read or Write command: e.g. illegal device # / no character preceding, actual cursor position if write, ERROR caused by a nonexisting device or file or error during write or read

CO : Command line overflow

E? : illegal or no command on Command Line

H? : wrong or no Hex-argument

I? : textbuffer overflow (T:0000)

L? : wrong I/O – device for listing /aborted listing / device not present

S? : string search exhausted/string not present in text?

T? : illegal Tabulator value

C? : C-Register overflow

#? : illegal argument for repetitive execution of a command.

All error messages are skipped by the next following Direct Mode command.

1.6 C-Register

The socalled C-Register is in fact an additional textbuffer as a mean for text copying to any place in the text stored in the main textbuffer. To copy text into the C-Register one has to open it by the command 'CTRL-R'. The C-Register when opened is then connected parallel to the main textbuffer and editing commands will be executed

equally in both text areas. Text is copied to the C-Register by the editing commands which are moving the cursor backwards through the text (e.g. CTRL-Q;, CTRL-W; CTRL-E, CTRL-H, CTRL-X). At the end of the text copying one has to close the C-Register by 'CTRL-F'. The text within the G-Register is now protected during normal editing operations but can be inserted or copied as many times as wanted by giving the available appropriate editing commands.

The C-Register is automatically closed after execution of Command Line or by a jump into the Assembler.

2. Assembler

The Assembler can be started directly from the Editor by pressing CTRL-Y. The Assembler translates (in three passes) the source-code text stored in the Editor text-buffer starting from textbegin till a CTRL 'Z' (Assembler stop sign) is encountered. If no CTRL'Z' is set in the text, assembling is executed till the end of the text. When the Assembler enters the second pass an action indication is displayed on screen at the last position of the line on top. As output option you can get from the third pass on a list of the assembled text, a label list or both together edited on screen. These lists can also be output to a printer.

If during assembling no errors have been encountered, the Assembler stops after completion. By pressing any key you can return to the Editor.
If an error is found during assembling, the Assembler stops immediately and a self-explanatory error message is displayed on screen.By pressing any key you return to the Editor with the cursor positioned directly after the erroneous text part. These features contribute considerably to a quicker correction.

ATAS ERRORS
There exist the following error messages:
'DIVISION BY ZERO'
'SAME LABEL TWICE'
'TOO MANY LABELS' : LABEL-List overflow
'WRONG OPCODE'
'NO HEX' : Illegal HEX-Argument
'ORG - ERROR'
'LABEL NOT DEFINED'
'BAD BRANCH'
'OPCODE UNKNOWN;CHECK ADDRESSING'
'STRING TOO LONG' : more than 250 char.
'NO ASCII' : no or illegal ASCII-Argument as CTRL 'Z' (=CTRL-Z)or CR (CTRL-M or RETURN)

48K ATMAS has the following error messages:
"LINE TOO LONG" (If line more than 127 ch.)
"TOO MANY LABELS"
"DIVISION BY ZERO"
"NO ASCII"
"UNDEFINED EXPRESSION"
"NUMBER ERROR"
"SYNTAX ERROR"
"NAME UNKNOWN"
" ORG ERROR"
"SAME LABEL TWICE"
"WRONG DELIMITER"
" MACRO ERROR"
" IMPOSSIBLE BRANCH"
"ADDRESSING ERROR"

"OPCODE DIFFERENT" Opcode is recognized different between pass 2 and pass 3. For instance a lable was recognized as an absolute label in the second pass and then in 3 pass it turned out as zero page instruction.
Define the lable more carefully!

Another example: Two ORG-Commands were in conflict.

WARNING
During the list of the assembled code a warning can appear:

"WARNING OPERAND OVERFLOW". This happens, if you want to put a two byte expression into a one byte location. This must not be an error, because it can happen on purpose. If no warnings occur at the end the assembler prints "NO WARNINGS".

2.1 Formats and Syntax

The Assembler interprets all existing Opcodes of the 65XX-Assembler language set as well as a few additional Pseudo-Opcodes which are in fact control commands for the Assembler

2.1.1 65XX-OPcodes:

ADC; AND; ASL; BCC; BCS; BEQ; BIT; BMI; BNE; BPL; BRK;
BVC; CLC; CLD; CLI; CLV; CMP; CPX; CPY; DEC; DEX; DEY;
EOR; INC; INX; INY; JMP; JSR; LDA; LDX; LDY; LSR; NOP;
ORA; PHA; PHP; PLA; PLP; ROL; ROR; RTI; RTS; SBC; SEC;
SED; SEI; STA; STX; STY; TAX; TAY; TSX; TXA; TXS; TYA

2.2.2 Pseudo-OPcodes:

EQU	:	EQUal
EPZ	:	Equal Page Zero
ORG	:	ORGanize, fix start address
DFB	:	DEFine Byte, insertion of a byte
DFW	:	DeFine Word, insertion of a word (= 2 bytes, lower and higher byte)
ASC	:	ASCII-String, insertion of ASCII-string
OUT	:	OUTput

The new ATMAS 48K has the following Pseudo-opcodes:
MACRO, MEND

and the following 6502 Opcodes: GOTO = JMP (both opcodes are existing)
JSR can be omitted.

2.2.3 Format of Addressing-Modes:

Impl. Accu : [Label]␣OPC [␣;⟨Com⟩] ⟨RETURN⟩
Immediate : [Label]␣OPC␣# ⟨Expr⟩ [␣⟨Com⟩] ⟨ Return ⟩
Abs, Zp, Rel : [Label]␣OPC␣⟨ Expr⟩ [␣⟨ Com ⟩] ⟨ Return ⟩
AbsX;ZpX : [Label]␣OPC␣⟨ Expr⟩, X [␣⟨ Com ⟩] ⟨ Return ⟩
AbsY;ZpY : [Label]␣OPC␣⟨ Expr ⟩, Y[␣⟨Com ⟩] ⟨ Return ⟩
IndX. : [Label]␣OPC␣(⟨ Expr.⟩, X) [␣⟨Com⟩] ⟨ Return ⟩
IndY. : [Label]␣OPC␣(⟨ Expr⟩), Y [␣⟨ Com ⟩] ⟨ Return ⟩
Ind. : [Label]␣OPC␣(⟨ Expr ⟩) [␣⟨ Com ⟩] ⟨ Return ⟩

2.2.4 Format of Pseudo-OPcodes:

EQU	:	⟨Label⟩␣EQU␣⟨ Expr ⟩ [␣⟨ Com ⟩] ⟨ Return ⟩
EPZ	:	⟨Label⟩␣EPZ␣⟨Expr⟩ [␣⟨ Com ⟩] ⟨Return ⟩
ORG	:	[Label]␣ORG␣⟨PL⟩ [, ⟨PP⟩] [␣⟨Com⟩] ⟨ Return ⟩
DFB	:	[Label]␣DFB␣⟨ Expr ⟩ [, ⟨ Expr⟩] " [␣⟨Com ⟩] ⟨ Return ⟩
DFW	:	[Label]␣DFW␣⟨Expr⟩ [, ⟨ Expr⟩] " [␣⟨ Com ⟩] ⟨Return⟩

ATMAS
ASC : [LABEL]␣ASC␣⟨ DELIM ⟩ [STRING] ⟨DELIM ⟩ [, ⟨ DELIM ⟩
[STRING] ⟨ DELIM ⟩] "[␣⟨ COMM ⟩] ⟨ RETURN⟩

OUT : [LABEL]␣ OUT␣[L] [N] [M] [P]␣⟨ COMM ⟩] ⟨ RETURN ⟩
[m] = MACROS not expanded

ATAS ASC: [LABEL]␣ASC␣⟨ DELIM ⟩ [STRING] ⟨ DELIM ⟩ [␣ COMM] ⟨ RETURN ⟩
OUT : [LABEL]␣ OUT␣ [L] [N] [P] [␣ COMM] ⟨ RETURN ⟩

2.2.5 Syntax legend

Expressions within ⟨ ⟩ are mandatory. Expressions within [ ] are optional. A " after] indicates that the whole expression within [ ] can be repeated as many times as desired. A ␣ indicates that at least one Space or one CTRL-I must be inserted. OPC indicates a legal OPcode.

Textlines beginning with * or CTRL-L are skipped by the Assembler as comment. An empty line,i. e. a line containing only ⟨ RETURN ⟩ is also skipped.

Label: Label consisting of at least one and up to a maximum of eight letters or digits where the first character must always be a letter. All eight

characters of a label are significant.

String: String of ASCII-characters; their values will be inserted here according to their position within the string. A string may contain a maximum of 250 characters and must be opened and closed by the same delimiter ⟨DELIM⟩.

DELIM: All non-alphanumeric characters
The use of \ as delimiter increases automatically the value of the last string character by $80 or 128.

COM: Comment may be placed here.

REturn: Carriage return (CR); effectuated by RETURN-key or CTRL-M.

Expr: for ATAS – Expr stands for an expression of which its arithmetically calculated value will be used as argument for the OPcode. Expr can be a decimal number, a Hexnumber (as $45, $ F62A etc.), a label, an ASCII-character (as 'A, 'B, '# etc.), or in combination with the arithmetic operators + – * / an Expr containing sums, products or quotients of Expr's. The arithmetic value range is –65535 to +65535. ASCII-characters imply immediate addressing and will be used as seven bit value' argument. E. g., LDA 'X is identical to LDA # $58 or LDA # 88. An asterisk * as term in an Expr will be interpreted as the value of the actual program pointer, which makes relative indexing very easy.

For ATMAS use brackets for priority

PL, PP: PL = logical start address of the assembled program
PP = physical start address, which means that the assembled program has been only dumped to a memory range with PP as start location.
This feature enables the user to assemble a program for a memory range which, for instance, is at the time of assembling occupied by another program and then dump it on a memory area starting at PP.
Later he can transfer the program by a simple block transfer to its final destination.
L : L = Assembler output listing
N : N = Label listing
P : Printer option

Short description of the "Normal Demo" program

First the labels are defined. EXW EPZ $ F0.1 means that memory location F0 and F1 are defined as the hexbuffer. The number 1 (F0.1) is a comment and no definition! It is not recognized as a pseudo op during assembly.

After that we define the operating system routines, we use in the program. To show you that we can use EPZ or EQU for definitions we wrote SCROUT EQU $F6A4.
(EQU = EPZ)

CLS CRN
EOL
CR
} defining ATASCI or ASCII values

ORG sets the beginning of the program to A800 hex. With the command U you can start the program for texting from the editor. Physical and logical addresses are the same.

In location A80C we can use the word MESSAGE instead of JSR MESSAGE. The following ASC Pseudo-OPcode followed by a string between delimiters puts this string into memory. (A80F–A817). The assembler adds 128 to the last character of the string, because the delimiter (bad slash) performs this function. Message prints the string to the screen. After performing this subroutine the program will be continued at A818 hex. Because of stack manipulation the locations A80F to A817 are skipped by the CPU. This combination is convenient for a powerful MACRO, if you work with programs containing a lot of text.

To perform the same operation with a MACRO, we define the MACRO with the name PRINT (see MACRO DEMO) with formale parameter STRING.

```
PRINT   MACRO STRING
        MESSAGE          Definition
        ASC STRING       of the
        MEND             MACRO
```

At address A83D hex (see MACRO DEMO), for instance, we can find a macrocall wit the actual parameter \ HEXADECIMAL: $ \ . The assembler replaces now in the macro definition the formal parameter STRING by the actual parameter \ HEXADECIMAL: $\ . Thus every formale parameter now can be replaced by the actual parameter by the MACRO. A MACRO can have more than one formal para-

meter, which are separated by a komma. During a macrocall the formal parameter has been replaced by the actual parameter in the same order. A macro can have local labels. The look alike normal global labels, however, must be followed by a @. During every macro call this @ will be replaced by a different four digit decimal number. This gives you the option for nested MACROS as long as the hardware stack.

The following program shall demonstrate the use of a macro assembler. The savings are not extremely high, because most of the macros are only used once in the text.
For more information and tutorials about macros refer to literature on the subject.

```
**************************************
*                                    *
*         MACRO DEMO                 *
*                                    *
*         FOR ATMASD 1               *
*                                    *
*         ATARI 800 48K              *
*                                    *
**************************************


*         MACRO-DEFINITIONS :

INPUT     MACRO STOPCHAR     MACRO WITH PARAM.
GETLOOP@  GETCHAR
          CMP #STOPCHAR
          PHP
          ROR LASTDIG        IF STOPCHARTHEN SET LASTDIG
          PLP
          BEQ OK@            PRINT STOPCHAR
          CMP '0
          BCC GETLOOP@       IF ASCII <0 THEN WAIT
          CMP '9+1
          BCS GETLOOP@       IF ASCII >9 THEN WAIT
OK@       SCROUT             PRINT ON SCREEN
          MEND               MACROEND


OUTPUT    MACRO EXPRESSION
          LDA EXPRESSION+1   LOAD HIGHBYTE
          BYTEOUT            PRINT IT
          LDA EXPRESSION     LOAD LOWBYTE
          BYTEOUT            PRINT IT
          MEND

PRINT     MACRO STRING
          MESSAGE            JUMP SUBROUTINE
          ASC STRING         DEF.STRING
          MEND

INIT      MACRO LOCATION,FLAG,LINE
          LDA #0
          STA LOCATION       CLEAR LOCATION
          STA LOCATION+1
          STA FLAG           CLEAR FLAG
          MESSAGE
          DFB CLSCRN+128     CLEAR SCREEN
          PRINT LINE
          MEND

INC2      MACRO PT
          INC PT             INCREMENT LOWBYTE
```

```
                        BNE *+4             NO OVERFLOW
                        INC PT+1            OTHERWISE ALSO HIGHBYTE
                        MEND


                *       DEFINE LABELS

                EXW     EPZ $F0.1           HEXEXPRESSIONBUFFER
                AUX     EPZ $F2.3           AUX.POINTER FOR PRINTROUTINE
                LASTDIG EPZ $F4             FLAG TO SHOW END OF INPUT

                SCROUT  EQU $F6A4           SCREENOUT
                GETCHAR EQU $F6DD           WAIT FOR INPUT ASCII(KEY)=
                                                              >ACCU
                CLSCRN  EQU $7D
                EOL     EQU $9B
                CR      EQU $0D


                        ORG $A800

                *       MAINLOOP

A800: A90085+START      INIT EXW,LASTDIG,\DECIMAL :\ INITIALISE PAR.,
A803: F085F1+                                        PRINT STRING
A806: 85F420+
A809: 72A8FD+
A80C: 2072A8+
A80F: 444543+
A812: 494D41+
A815: 4C20BA+
A818: 20DDF6+INLOOP     INPUT EOL           KEYINPUT
A81B: C99B08+
A81E: 66F428+
A821: F008C9+
A824: 3090F1+
A827: C93AB0+
A82A: ED20A4+
A82D: F6    +
A82E: 24F4              BIT LASTDIG
A830: 3006              BMI FINISH          IF RETURN THEN FINISH
A832: 205FA8            DECHEX              NEXT DECIMAL POSITION TO EXW
A835: 4C18A8            JMP INLOOP          ALWAYS TAKEN
A838: 2072A8 FINISH     MESSAGE
A83B: 0D8D              DFB CR,CR+128
A83D: 2072A8+           PRINT \HEXADECIMAL : $\
A840: 484558+
A843: 414445+
A846: 43494D+
A849: 414C20+
A84C: 3A20A4+
A84F: A5F120+           OUTPUT EXW
A852: 9AA8A5+               '
A855: F0209A+
A858: A8    +
A859: 20DDF6            GETCHAR             WAIT FOR ANY KEY
A85C: 4C00A8            GOTO START          STARTS AGAIN

                *       SUBROUTINES:

A85F: 290F    DECHEX    AND #%00001111      MAKE ASCII => BIN.
A861: A211              LDX #17             16 BIT HEXADECIMAL
```

```
A863: D005           BNE DEC3  A.T. ALWAYS TAKEN
A865: 9002    DEC2   BCC *+4       IF BIT=0 ONLY ROTATE
A867: 6909           ADC #9        OTHERWISE ADD 9+CARRY=10
A869: 4A             LSR ;         ROTATE ACCU THROUGH CARRY IN EXW
A86A: 66F1    DEC3   ROR EXW+1
A86C: 66F0           ROR EXW       BIT => CARRY
A86E: CA             DEX           ONLY X-1 BITS
A86F: D0F4           BNE DEC2      IF X<>0 THEN ROTATE
A871: 60             RTS

A872: 68      MESSAGE PLA          (STACK)=>AUX
A873: 85F2           STA AUX
A875: 68             PLA
A876: 85F3           STA AUX+1
A878: A200           LDX #0
A87A: E6F2D0+MESLOOP INC2 AUX
A87D: 02E6F3+
A880: A1F2           LDA (AUX,X)   NEXT PRINT-CHARACTER
A882: 297F           AND #%01111111 BIT 7:=0
A884: C90D           CMP #CR
A886: D002           BNE MESS2     IF CHAR=CR THEN CHAR=EOL :ATASCII
A888: A99B           LDA #EOL                                  RETURN
A88A: 20A4F6 MESS2   SCROUT        PRINT CHAR
A88D: A200           LDX #0
A88F: A1F2           LDA (AUX,X)
A891: 10E7           BPL MESLOOP   IF ASCII(CHAR)>=128 THEN MESLOOP
ASSED                                                    :TEXT P
A893: A5F3           LDA AUX+1     AUX => (STACK)
A895: 48             PHA
A896: A5F2           LDA AUX
A898: 48             PHA
A899: 60             RTS           JUMP TO NEXT OPCODE

A89A: 48      BYTEOUT PHA          ACCU => (STACK)
A89B: 4A             LSR ;         ACCU := HIGHNIPPLE (ACCU)
A89C: 4A             LSR
A89D: 4A             LSR
A89E: 4A             LSR
A89F: 20A5A8         HEXOUT        PRINT HEXADECIMAL
A8A2: 68             PLA           (STACK)=> ACCU
A8A3: 290F           AND #%00001111 ACCU := LOWNIPPLE (ACCU)
A8A5: C90A    HEXOUT CMP #9+1
A8A7: B004           BCS ALFA      IF NIPPLE >9 THEN ALFA :10-15 ARE
TERS                                                     CHARAC
A8A9: 0930           ORA '0        ACCU := ASCII(ACCU) MAKE ASCII
A8AB: D003           BNE HEXOUT2 A.T.                      NUMBERS
A8AD: 18      ALFA   CLC
A8AE: 6937           ADC 'A-10     ACCU := ASCII(ACCU) MAKE ASCII
 A-F                                                     CHARACTERS
A8B0: 4CA4F6 HEXOUT2 GOTO SCROUT   RTS VIA SCROUTROUTINE

PHYSICAL ENDADDRESS:$A8B3

*** NO WARNINGS

INPUT           MACRO          OUTPUT          MACRO
PRINT           MACRO          INIT            MACRO
INC2            MACRO          EXW             $F0
AUX             $F2            LASTDIG         $F4
SCROUT          $F6A4          GETCHAR         $F6DD
CLSCRN          $7D            EOL             $9B
CR              $0D            START           $A800
```

```
INLOOP          $A818           GETLOOP0003     $A818
OK0003          $A82B           FINISH          $A838
DECHEX          $A85F           DEC2            $A865
DEC3            $A86A           MESSAGE         $A872
MESLOOP         $A87A           MESS2           $A88A
BYTEOUT         $A89A           HEXOUT          $A8A5
ALFA            $A8AD           HEXOUT2         $A8B0

                ****************************************
                *                                      *
                *         NORMAL DEMO                  *
                *                                      *
                *         FOR ATMASD 1                 *
                *                                      *
                *         ATARI 800 48K                *
                *                                      *
                ****************************************

                *         DEFINE LABELS

                EXW       EPZ $F0.1          HEXEXPRESSIONBUFFER
                AUX       EPZ $F2.3          AUX.POINTER FOR PRINTROUTINE
                LASTDIG   EPZ $F4            FLAG TO SHOW END OF INPUT

                SCROUT    EQU $F6A4          SCREENOUT
                GETCHAR   EQU $F6DD          WAIT FOR INPUT ASCII(KEY)=
                                                          >ACCU
                CLSCRN    EQU $7D
                EOL       EQU $9B
                CR        EQU $0D


                          ORG $A800

                *         MAINLOOP

A800: A900      START     LDA #0
A802: 85F0                STA EXW            CLEAR EXW
A804: 85F1                STA EXW+1
A806: 85F4                STA LASTDIG        CLEAR FLAG
A808: 2072A8              MESSAGE
A80B: FD                  DFB CLSCRN+128     CLEAR SCREEN
A80C: 2072A8              MESSAGE
A80F: 444543              ASC \DECIMAL :\
A812: 494D41
A815: 4C20BA
A818: 20DDF6 GETLOOP      GETCHAR
A81B: C99B                CMP #EOL
A81D: 08                  PHP
A81E: 66F4                ROR LASTDIG        IF EOL THEN SET LASTDIG
A820: 28                  PLP
A821: F008                BEQ OK             PRINT EOL
A823: C930                CMP '0
A825: 90F1                BCC GETLOOP        IF ASCII <0 THEN WAIT
A827: C93A                CMP '9+1
A829: B0ED                BCS GETLOOP        IF ASCII >9 THEN WAIT
A82B: 20A4F6 OK           SCROUT             PRINT ON SCREEN
A82E: 24F4                BIT LASTDIG
A830: 3006                BMI FINISH         IF RETURN THEN FINISH
A832: 205FA8              DECHEX             NEXT DECIMAL POSITION TO EXW
A835: 4C18A8              JMP GETLOOP        ALWAYS TAKEN
A838: 2072A8 FINISH       MESSAGE
```

```
A83B: 0D8D              DFB CR,CR+128
A83D: 2072A8            MESSAGE
A840: 484558            ASC \HEXADECIMAL : $\
A843: 414445
A846: 43494D
A849: 414C20
A84C: 3A20A4
A84F: A5F1            LDA EXW+1       LOAD HIGHBYTE
A851: 209AA8          BYTEOUT         PRINT IT
A854: A5F0            LDA EXW         LOAD LOWBYTE
A856: 209AA8          BYTEOUT         PRINT IT
A859: 20DDF6          GETCHAR         WAIT FOR ANY KEY
A85C: 4C00A8          GOTO START      STARTS AGAIN

                *     SUBROUTINES:

A85F: 290F    DECHEX  AND #%00001111 MAKE ASCII => BIN.
A861: A211            LDX #17        16 BIT HEXADECIMAL
A863: D005            BNE DEC3  A.T. ALWAYS TAKEN
A865: 9002    DEC2    BCC *+4        IF BIT=0 ONLY ROTATE
A867: 6909            ADC #9         OTHERWISE ADD 9+CARRY=10
A869: 4A              LSR ;          ROTATE ACCU THROUGH CARRY IN EXW
A86A: 66F1    DEC3    ROR EXW+1
A86C: 66F0            ROR EXW        BIT => CARRY
A86E: CA              DEX            ONLY X-1 BITS
A86F: D0F4            BNE DEC2       IF X<>0 THEN ROTATE
A871: 60              RTS

A872: 68      MESSAGE PLA            (STACK)=>AUX
A873: 85F2            STA AUX
A875: 68              PLA
A876: 85F3            STA AUX+1
A878: A200            LDX #0
A87A: E6F2    MESLOOP INC AUX        INCREMENT LOWBYTE
A87C: D002            BNE *+4        NO OVERFLOW
A87E: E6F3            INC AUX+1      OTHERWISE ALSO HIGHBYTE
A880: A1F2            LDA (AUX,X)    NEXT PRINT-CHARACTER
A882: 297F            AND #%01111111 BIT 7:=0
A884: C90D            CMP #CR
A886: D002            BNE MESS2      IF CHAR=CR THEN CHAR=EOL :ATASCII
A888: A99B            LDA #EOL                                 RETURN
A88A: 20A4F6  MESS2   SCROUT         PRINT CHAR
A88D: A200            LDX #0
A88F: A1F2            LDA (AUX,X)
A891: 10E7            BPL MESLOOP    IF ASCII(CHAR)>=128 THEN MESLOOP
ASSED                                                       :TEXT P
A893: A5F3            LDA AUX+1      AUX => (STACK)
A895: 48              PHA
A896: A5F2            LDA AUX
A898: 48              PHA
A899: 60              RTS            JUMP TO NEXT OPCODE

A89A: 48      BYTEOUT PHA            ACCU => (STACK)
A89B: 4A              LSR ;          ACCU := HIGHNIPPLE (ACCU)
A89C: 4A              LSR
A89D: 4A              LSR
A89E: 4A              LSR
A89F: 20A5A8          HEXOUT         PRINT HEXADECIMAL
A8A2: 68              PLA            (STACK)=> ACCU
A8A3: 290F            AND #%00001111 ACCU := LOWNIPPLE (ACCU)
A8A5: C90A    HEXOUT  CMP #9+1
```

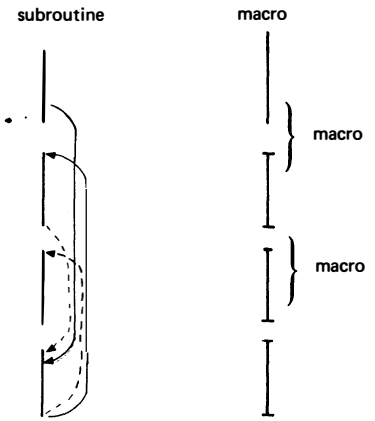
```
A8A7: B004            BCS ALFA          IF NIPPLE >9 THEN ALFA :10-15
TERS                                                      ARE CHARAC
A8A9: 0930            ORA '0            ACCU := ASCII(ACCU) MAKE ASCII
A8AB: D003            BNE HEXOUT2 A.T.                        NUMBERS
A8AD: 18      ALFA    CLC
A8AE: 6937            ADC 'A-10         ACCU := ASCII(ACCU) MAKE ASCII
 A-F                                                      CHARACTERS
A8B0: 4CA4F6 HEXOUT2  GOTO SCROUT       RTS VIA SCROUTROUTINE

PHYSICAL ENDADDRESS: $A8B3

*** NO WARNINGS

EXW             $F0             AUX             $F2
LASTDIG         $F4             SCROUT          $F6A4
GETCHAR         $F6DD           CLSCRN          $7D
EOL             $9B             CR              $0D
START           $A800           GETLOOP         $A818
OK              $A82B           FINISH          $A838
DECHEX          $A85F           DEC2            $A865
DEC3            $A86A           MESSAGE         $A872
MESLOOP         $A87A           MESS2           $A88A
BYTEOUT         $A89A           HEXOUT          $A8A5
ALFA            $A8AD           HEXOUT2         $A8B0
```

Introduction to the use of MACROs

Introduction to the use of MACROs

The purpose of macros as well as of subroutines is to replace often needed routines by a simple call.

Although macros and subroutines are similar there are differences which you have know in order to be able to use them the best way. The next figure shows the difference:



Subroutines as well as macros are written only once by the programmer but the assembler writes subroutines only once while inserting the whole macro every time they are called. By that technique the stackoperations necessary with the use of subroutines are sowed.

In general we can say:
macros need more memory but they are faster subroutines are easier to use.

At the beginning of the main program where the macro is defined, we use formal parameters. These formal parameters are used instead of the actual parameters handled over to the macro every time we call ist. Parameters used only within the macro are named local parameters.

Example:
macro-definition:

```
INC2  MACRO PT
      INC PT
      BNE *+4
      INC PT+1
      MEND
```

Name of the macro ist INC2, name of the formal parameter ist PT.
Call of macro. INC2 AUX
Name of the actual parameter is AUX.

Another example shows a blocktransfer-routine. This routine moves a block of bytes between STARTP and ENDP to a location beginning at INTOP.

```
MOV     LDA   STARTP
        CMP   END
        BNE
        LDA   STARTP+1
        CMP   ENDP+1
        BEQ   ENDMOV
        LDX   #Ø
        LDA   (STARTP,X)
        STA   (INTOP,X)
        INC   STARTP
        BNE   *+4          INC2 STARTP
        INC   STARTP+1
        INC   INTOP
        BNE   *+4          INC2 INTOP
        INC   INTOP+1
        JMP   MOV
ENDMOV  RTS
```

If you have defined the following macro at the beginning:

```
INC2  MACRO PT
      INC PT
      BNE *+4
      INC PT+1
      MEND
```

you can replace the parts of the program marked by brachets by calling macro INC2 STARTP and INC2 INTOP.

A sample session with the ATAS-1

Order-No. 7098	**32k RAM**	**$49.95**
Order-No. 7998	**48k RAM**	**$49.95**

For loading see the instructions given for locating ATMONA-1 (ATMONA-1 + ATARI MONITOR Version 1). The ATMONA-1 is part of the assembler ATAS-1.

Important: This assembler is written for a 32K system, either ATARI 400 or ATARI 800. It will only run in this environment. If you have less memory, it will not work, but if you have more memory it also will not work. Therefore, if you have a 48K system, you have to remove the upper 16K of memory.
After loading you will see the message ATAS-1.

To start the assembler, type K for coldstart. This will erase the text buffer and should only be typed in at the beginning of a new session.
After doing this, you see two lines on the screen. One on the top:

```
P:00000        T:09984        C: 00512        OK
```

and a line of dots on the bottom.

The cursor is at the beginning of the text window. If you type in some text, you will see an increasing number at P: and a decreasing number at T:.
The number at P: shows you how many bytes you have used for your text; the number at T: shows you how many bytes of the

textbuffer are free.

Now, let's erase the text you have written by typing

ESC K ESC ESC

After typing the first ESC, you will see a $ sign at the beginning of the command line. This means you have opened the commandline for entering commands. The letter K indicates you will erase the textbuffer. The next ESC closes the command and it is executed with the next ESC. The command line now looks like following:

K#

The number sign indicates that you executed the command.

As an example we will type in the following program:

```
EOUTCH   EQU $F6A4
PUTLIN   EQU $F385

         ORG $AC00          ( ATMASI 48K )

*        ORG $A800          ( ATMAS  48K )
*        ORG $8D00          ( ATAS   48K )
*        ORG $6000          ( ATAS   32K )

         LDA #$7D
         JSR EOUTCH
         LDX #MES:L         ( ATMAS & ATMASI )
         LDY #MES:H         ( ATMAS & ATMASI )

*        LDX #MES           ( ATAS 32K & 48K )
*        LDY #MES/256       ( ATAS 32K & 48K )

         JSR PUTLIN
         BRK

MES      ASC "DIESE MELDUNG WIRD"
         ASC "AUF DEM BILDSCHIRM AUSGEGEBEN"
         DFB $9B
```

This program uses two monitor routines of the ATARI monitor. The routine EOUTCH puts one character on the screen; the routine PUTLIN outputs text until a $9B is encountered. The maximum length of text is 128 bytes.

The instruction ORG $A800 means that our program starts at memory location $A800.
In this program, we load the accumulator with $7D and put this command on the screen with JSR EOUTCH. This is the internal command for the ATARI to erase the screen and set the cursor in the upper left position of the screen. Then we use the subroutine PUTLIN to print text on the screen. The starting address of this text is transferred to the subroutine PUTLIN via the X and the Y registers. The low byte of the address must be in the X-register and the high byte in the Y-register.

After the JSR PUTLIN instruction, the program is terminated by a BRK instruction.

A new ORG $4050 tells the assembler to store the following text beginning at this location.

The pseudo instruction ASC is used for storing the text. The end of the text is indicated by the pseudo instruction DFB $9B.
(DFB equals DeFine Byte). ATARI uses $9B for ending a text instead of $0D as it is normally used as an ASCII-character.

The ATAS-1 uses the normal interpretation of text just like other assemblers. A label starts in the first column of a text line: an opcode is preceeded by at least one space character.

To get a neat looking output in our assemblertext, we use the TAB function of the editor. Every label starts in the first column, every op-code starts in the 9th column.

We set the TAB-function by typing

ESC @9ESC ESC

Our command line looks like:

$@9$#

Now we start writing the text:

```
EOUTCH   CTRLI    EQU $F6A4
PUTLIN   CTRLI    EQU $F385
CTRL I            ORG $AC00
CTRLI             LDA $ 7D      and so on.
```

CTRL I means, pressing both the CTRL and the I key at one time.
If you make a typing error, you can erase the preceeding character by CTRLN.

At the end of the text, type CTRL Z. This indicates to the assembler that the assembling stops here. ATAS I doesn't use an END-instruction.

Order-No. 7022	cassette version	$19.95
Order-No. 7023	disk version	$24.95
Order-No. 7024	cartridge version	$59.00

ATMONA-1

ATMONA-1 — Machine Language Monitor for the ATARI

ATMONA-1 was developed by Ing. W. Hofacker GmbH for the ATARI 400 and 800 computers. It is a machine language monitor supplied on a bootable cassette.

For loading the cassette, use the following instructions:

1. Turn off the computer, remove all cartridges and turn off the disk (if any).
2. Insert the cassette in the program recorder and press PLAY.
3. **While holding down the START key,** turn on the computer.
4. When the computer beeps, release the START key and press RETURN

   The program is now loaded into the computer.

After loading, the computer will show a copyright notice and the title "ATMONA-1".

Now you can enter one of the following instructions:

D	Disassemble
M	Memory dump, with or without ASCII characters
C	Change the content of a memory location
F	Fill a memory block with a specified byte
B	Block transfer
L	Load machine code from tape
S	Save machine code on tape
G	Goto a specified address (Start a program)
X	Breaks the executed instruction and goes back to input level. Same as SYSTEM RESET

Commands:

You must type in only what's underlinded.

DISASSEMBLER

START? 1000	Disassemble starts at 1000 HEX.
PRINT? N	No print (see note).
	Now disassembling begins, printing 16 lines

	on the screen. Hit any key (except the X key) for the next 16 lines.
X	Terminates the disassembling.
M	Starts memory dump.
DUMP	
FROM: 1000	Dump begins at 1000 HEX
TO: 1100	Dump ends at 1100 HEX
ASCII? Y	Hex bytes are also printed as ASCII characters.
N	Only hex bytes are printed.
PRINT? N	No printer is used (see note).
	The dump starts, displaying 16 lines on screen. Hit any key except the X key for the next 16 lines. Dump ends at the ending address, or
X	Terminates the dump.

Note: Printing only with printer option (See appendix).

CHANGE	
ADDRESS: 1000	
1000 00 = ⟩ FF	The content of memory address 1000 HEX is changed from 00 HEX to FF HEX.
1001 00 = ⟩	The content of memory address 1001 is
RETURN	not changed.
1002 00 = ⟩ X	Terminates the change.
FILL	
FROM: 1000	Fills the memory block starting at 1000 HEX until 1100 HEX with the hex byte AA.
TO: 1100	
WITH: AA	
BLOCKTRANSFER	
FROM: 1000	The contents of the memory block (starting at 1000 HEX, ending at 1100 HEX) are
TO: 1100	transfered to 2000 HEX until 2100 HEX.
INTO: 2000	

LOAD — Loads a machine language program into memory. Insert cassette, press PLAY and RETURN.

SAVE — Saves machine code on tape. Starting address of the code is 1000 HEX, ending address is 1100 HEX. Press RECORD and PLAY on the tape recorder, then RETURN.

FROM: 1000

TO: 1100

GOTO 1000 — Starts a machine language program at memory location 1000 HEX. This address must be the starting address of the program.

Appendix: Using a printer in the machinelanguage-monitor

a) YES (Y) after PRINT? gives you three options.
   (1) Output to printer via serial port of the ATARI RS232 Interface
   (2) Output via the parallel port of the ATARI Interface
   (3) Output via the expansion from Elcomp Publishing, Inc.

Order-No. 7049	cassette version	$49.95
Order-No. 7050	disk version	$54.00

ATMONA-2

ATMONA-2

The ATMONA-2 consists of two separate programs:
the ATMONA-1 which comes up when the cassette is booted and the SUPERTRACER which starts at memory location 0F00.
For loading the cassette, see the loading instructions given for ATMONA-1.
After booting the cassette you can use all the functions of ATMONA-1.
To start the SUPERTRACER, type GOTO 0F00.
(For disk version refer to page 35!)
You will see the first command line of the SUPERTRACER:

SUPERTRACER (T) (G) (X) (C) (P)

T Start tracing thru a program. The starting address is momentary content of the program counter PC. Every single step is displayed on the screen, showing you the contents of the program counter PC, the accumulator AC, the X- and the Y-register XR, YR, the stackpointer SP, the flags in binary representation, and the mnemonic code of the next instruction. To execute this instruction, type SPACE or any other key except those keys shown in the command line.

G The same as T, but now the steps are executed automatically until a stop condition is executed. If the program doesn't find any, hit SYSTEM RESET.

X Terminates the SUPERTRACER and switches back to ATMONA-1.

C Enters the CHANGE command level.

P Sets the PRINTER option (see note ATMONA-1).

The CHANGE command level:

This level is reached by typing C in the first command level. You will see the change command line:

CHANGE (A) (O) (R) (X)

A Changes address. When the program is started by the G command, it will halt and display the contents of the registers every time this address is encountered.

O The program will now stop every time this given operand is encountered.
With both of these instructions you can set the address-stop or the operand stop. Both can be set individually.

R Changes the contents of the registers. These are not stopping conditions. The program executes the next instruction with these predefined values. When you type R you enter the third command level (see below).

X Terminates the CHANGE command level. The contents of all registers is shown on the screen.

REGISTER CHANGE level:

The commandline is:

CHANGE PC, XR, YR, AC, SP, FLAGS

PC Sets the program counter to a predefined value. The next T or G instruction will start the program at this address.

XR, YR, AC, SP The contents of these registers can be changed.

FLAGS

SR: 80 80 HEX is equal to 1000 0000 binary. Therefore the N bit of the status register is set to 1.

RETURN

Terminates the register change level.

The ATMONA-2 can be used in three different ways:

1. Searching for bugs in machine language programs.
2. Stepping thru unknown programs.
3. For educational purposes. Learning 6502 machine code.

The following is an example of item # 3. For those who want to learn 6502 code, it shows the execution of some instructions.
We will use the following program mini-COUNT. The program counts the X and the Y registers as a 16-bit number until this number is equal to a 16-bit number stored in the memory locations $ 1FFE and $ 1FFF. (Hex numbers will be indicated by a preceding $sign).

Program COUNT:

```
                          MEML = $1FFE
                          MEMH = $1FFF
$2000 A9 00               LDA # 00      Load accu with 00
                                        immediately
$2002 AA                  TAX           Transfer accu to X reg.
$2003 A8                  TAY           Transfer accu to Y reg.
$2004 CC FF 1F  M0        CPY MEMH      Yreg = MEMH ?
$2007 D0 05               BNE M1        If not equal, goto M1
$2009 EC FE 1F            CPX MEML      Xreg = MEML ?
$200C F0 07               BEQ FIN       If equal then end
$200E E8          M1      INX           Increment Xreg.
$200F D0 F3               BNE M0        IF Xreg not equal 0, goto
                                        M0
$2011 C8                  INY           Increment Yreg
$2012 18                  CLC
$2013 90 EF               BCC M0        Jump to M0
$2015 4C 47 07  FIN       JMP 0747      Jump to ATMONA
```

Using the CHANGE option of ATMONA 1, we enter the program at starting address $2000:

```
CHANGE 2000

2000 00 =) A9
     .
     .                    until
2017 00 =) 07
```

We leave the CHANGE option by typing X and look at the program by Disassembling. The program on the screen must be the same as the listing above, except there are no symbolic names.

Before we trace this program, we change the contents of the two memory locations $1FFE to 01 and $1FFF to 00.

Now we enter the SUPERTRACER by GOTO 0F00 and CHANGE the REGISTER PC to 2000, AC to FF, XR to FF and YR to FF.

We leave this program level by hitting the RETURN key and the CHANGE option by typing X.

The following two lines must now appear on the screen:

```
  PC   AC   XR   YR   SP   NV.BDIZC  MNM  OPERAND
2000   FF   FF   FF   FF   00000000  LDA  # 00
```

and also the ST command line:

SUPERTRACER (T) (G) (X) (C) (P)

You will note that all the registers have the value we entered. The programcounter is $ 2000 and the next instruction which will be executed is LDA # 00 which means to load the accumulator with 00 immediately.

This instruction is executed when we type T for TRACE. The program stops at the next instruction, and the screen shows two new lines:

```
  PC   AC   XR   YR   SP   NV.BDIZE  MNM  OPERAND
2000   00   FF   FF   FF   00110010  TAX
```

The content of the accumulator has changed to 00 (as it should) and the zero flag in the status register is set to 1 because a zero has been read into the accumulator.

If we type T once more we can see the result of the instruction TAX:

```
  PC   AC   XR   YR   SP   NV.BDIZE  MNM  OPERAND
2003   00   00   FF   FF   00100010  TAY
```

In this manner we can trace our whole program and we see that each instruction accomplishes.

For another short demonstration we will use the ADDRESS STOP and the GO function. First we step back to the ATMONA-1 by typing X and CHANGE $ 1FFE to 05. After leaving the change option, we restart the SUPERTRACER by GOTO 0F00. We CHANGE ADDRESS to 200E and REGISTER PC to 2000. With RETURN and X we step back to the ST command level.
After starting the program with G, it stops at address $ 2007 because there is reference to our stopping address 200E in the operand field (BNE 200E). A new GO forces the program to continue until address 200E is reached.
Type G again if you wish to step thru the program until the compare instruction in memory location $2009 is fullfilled.
The program will then start tracing the ATMONA 1.

The ATMONA-2 Disk Version comes on a bootable disk. After loading, you enter the ATMONA-1, which is included. Then you type Q to enter DOS.
In DOS you load the Superstepper object code, using the L-command.

The filename is: SUPRSTEP.OBJ.

Then you go back to the ATMONA-1 with B-RUN ATMONA.

To get to the Superstepper type GOTO 2A00.

How to work with ATMONA-2

How to work with ATMONA-2

With ATMONA-2 you can step through a machine language program in 6502 code. You can stop at a previously entered address, opcode or operand. You can T(race in single steps or simulate the program with a G(oto.

Lets do an example:
First load the ATMONA-2.
Now you can look at the code of ATMONA-1 by D(isassembling D 0747, P=No. You will see the disassembled listing on the screen. At address 0758 you should see a JSR 0763. For debugging purposes we will halt the program at that address.
By typing X we return to ATMONA-1 (included in ATMONA-2).
We start ATMONA-2 by GOTO 0F00. You will see the first command line:

SUPERTRACER (T) (G) (X) (P)

Let' s type C for change.
Now you will see the second command line:

CHANGE (A) (O) (C) (R) (X)

To define a stop address we type A.
It will respond with:

ADDRESS:

and here you enter 075B. The second command line will return and you can define other stopping conditions. To start our test-program we change the contents of the program counter by typing R for register and we will see the following prompt line:

CHANGE PC, XR, YR, AC, SP, FLAGS

Here you can change the contents of the 6502 CPU registers.
In our example we change the program counter by typing P, and after the prompt PC: we enter 0747 which is the starting address of the testprogram. Now we type RETURN for exit of this command level and X for reaching the highest command level.

ATMONA-2 will display your starting address and the starting conditions.
Type G(oto and the supertracer will simulate the program until it reaches a stopping condition. You will see a slow execution of ATMONA-1 because 0747 is its starting address. After a few seconds it stops at address 075B, showing us the contents of the registers and the mnemonic code of the next instruction to be executed.
You are in the first command level and you can T(race (single step) or define new stopping conditions.

Instead of stopping at an address you also can stop at a defined opcode or a defined operand.
The supertracer ATMONA-2 stops at non executable opcodes, RTI or BRK instruction.
While running the supertracer may be stopped by pressing the RESET button.

LEARN FORTH

LEARN FORTH Order-No. 7053 $19.95

LEARN FORTH has been developed by ELCOMP for the ATARI 400/800. It allows you to type in and run the sample programs published recently in books about FORTH (Brodie, FORTH Learning By Using, etc.)
LEARN FORTH is available on cassette or on disk and will work even with a cassette based ATARI 400.
LEARN FORTH by ELCOMP is a subset of fig-FORTH. It contains all definitions of fig-FORTH, but it does not have an editor or screen windows (SCREENS). It allows you to save your definitions. Command VLIST shows the definitions already implemented.

Loading the disk-version

The disk-version boots itself and comes up with a message on the screen.

Loading the cassette-version

LEARN FORTH for ATARI comes on two cassettes. Load the first one like a bootable cassette (turn the computer on while holding down the yellow START key, load the program after the beep). Your ATARI will come up with a message on the screen. Now enter L and load the second cassette with the object code on it. After that you can do a coldstart by pressing 'K'. You then are in LEARN FORTH.
If you have extended LEARN FORTH by your own definitions you save the object code on a new cassette using command SAVE.

Error messages :

STAPEL LEER	the stack is empty
STAPEL VOLL	the stack is full
<NAME> ?	<NAME> is unknown
FALSCH	error during compilation

Warnings :

<NAME> SCHON VORHANDEN <NAME> defined before

<NAME> GESCHUETZT <NAME> is in protected area of the dictionary

If an error FALSCH occurred during compilation you may erase the name entered from the dictionary by

```
SMUDGE FORGET <NAME>
```

A hint :

The length of the program is about 5.5 kbyte. All definitions are tested, but it is possible that there are hidden errors included. In case you detect an error please let us know about it and where it occurred. You help us and others by doing so.

NOTE:
Typing in MON in the Learn FORTH brings you back into the LOADER.
(LOADER = 1. Cassette)
Then you can go back to Learn FORTH by pressing W for warmstart or K for coldstart.

FORTH HANDY REFERENCE

Stack inputs and outputs are shown; top of stack on right
This card follows usage of the Forth Interest Group (S.F. Bay Area); usage aligned with the *Forth 78* International Standard.

For more info: Forth Interest Group
P.O. Box 1105
San Carlos, CA 94070.

Operand key:

n, n1, . . .	16-bit signed numbers
d, d1, . . .	32-bit signed numbers
u	16-bit unsigned number
addr	address
b	8-bit byte
c	7-bit ascii character value
f	boolean flag

STACK MANIPULATION

DUP	(n → n n)	Duplicate top of stack.
DROP	(n →)	Throw away top of stack.
SWAP	(n1 n2 → n2 n1)	Reverse top two stack items
OVER	(n1 n2 → n1 n2 n1)	Make copy of second item on top.
ROT	(n1 n2 n3 → n2 n3 n1)	Rotate third item to top.
-DUP	(n → n ?)	Duplicate only if non-zero.
>R	(n →)	Move top item to "return stack" for temporary storage (use caution).
R>	(→ n)	Retrieve item from return stack
R	(→ n)	Copy top of return stack onto stack.

NUMBER BASES

DECIMAL	(→)	Set decimal base.
HEX	(→)	Set hexadecimal base.
BASE	(→ addr)	System variable containing number base

ARITHMETIC AND LOGICAL

+	(n1 n2 → sum)	Add.
D+	(d1 d2 → sum)	Add double-precision numbers.
-	(n1 n2 → diff)	Subtract (n1-n2).
*	(n1 n2 → prod)	Multiply.
/	(n1 n2 → quot)	Divide (n1/n2).
MOD	(n1 n2 → rem)	Modulo (*i.e.* remainder from division).
/MOD	(n1 n2 → rem quot)	Divide, giving remainder and quotient.
*/MOD	(n1 n2 n3 → rem quot)	Multiply, then divide (n1*n2/n3), with double-precision intermediate.
*/	(n1 n2 n3 → quot)	Like */MOD, but give quotient only.
MAX	(n1 n2 → max)	Maximum.
MIN	(n1 n2 → min)	Minimum.
ABS	(n → absolute)	Absolute value.
DABS	(d → absolute)	Absolute value of double-precision number.
MINUS	(n → -n)	Change sign.
DMINUS	(d → -d)	Change sign of double-precision number.
AND	(n1 n2 → and)	Logical AND (bitwise).
OR	(n1 n2 → or)	Logical OR (bitwise).
XOR	(n1 n2 → xor)	Logical exclusive OR (bitwise).

COMPARISON

<	(n1 n2 → f)	True if n1 less than n2.
>	(n1 n2 → f)	True if n1 greater than n2.
=	(n1 n2 → f)	True if top two numbers are equal.
0<	(n → f)	True if top number negative.
0=	(n → f)	True if top number zero (*i.e.*, reverses truth value).

MEMORY

@	(addr → n)	Replace word address by contents.
!	(n addr →)	Store second word at address on top.
C@	(addr → b)	Fetch one byte only.
C!	(b addr →)	Store one byte only.
?	(addr →)	Print contents of address.
+!	(n addr →)	Add second number on stack to contents of address on top.
CMOVE	(from to u →)	Move u bytes in memory.
FILL	(addr u b →)	Fill u bytes in memory with b, beginning at address.
ERASE	(addr u →)	Fill u bytes in memory with zeroes, beginning at address.
BLANKS	(addr u →)	Fill u bytes in memory with blanks, beginning at address.

CONTROL STRUCTURES

DO . . . LOOP	do: (end+1 start →)	Set up loop, given index range.
I	(→ index)	Place current index value on stack.
LEAVE	(→)	Terminate loop at next LOOP or +LOOP.
DO . . . +LOOP	do: (end+1 start →) +loop: (n →)	Like DO . . . LOOP, but adds stack value (instead of always '1') to index.
IF . . . (true) . . . ENDIF	if: (f →)	If top of stack true (non-zero), execute. [Note: *Forth 78* uses IF . . . THEN.]
IF . . . (true) . . . ELSE . . . (false) . . . ENDIF	if: (f →)	Same, but if false, execute ELSE clause. [Note: *Forth 78* uses IF . . . ELSE . . . THEN.]
BEGIN . . . UNTIL	until: (f →)	Loop back to BEGIN until true at UNTIL. [Note: *Forth 78* uses BEGIN . . . END.]
BEGIN . . . WHILE . . . REPEAT	while: (f →)	Loop while true at WHILE; REPEAT loops unconditionally to BEGIN. [Note: *Forth 78* uses BEGIN . . . IF . . . AGAIN.]

TERMINAL INPUT-OUTPUT

	(n →)	Print number
R	(n fieldwidth →)	Print number, right-justified in field.
D	(d →)	Print double-precision number.
D.R	(d fieldwidth →)	Print double-precision number, right-justified in field.
CR	(→)	Do a carriage return.
SPACE	(→)	Type one space.
SPACES	(n →)	Type n spaces
	(→)	Print message (terminated by ").
DUMP	(addr u →)	Dump u words starting at address.
TYPE	(addr u →)	Type string of u characters starting at address
COUNT	(addr → addr+1 u)	Change length-byte string to TYPE form.
?TERMINAL	(→ f)	True if terminal break request present
KEY	(→ c)	Read key, put ascii value on stack.
EMIT	(c →)	Type ascii value from stack.
EXPECT	(addr n →)	Read n characters (or until carriage return) from input to address
WORD	(c →)	Read one word from input stream, using given character (usually blank) as delimiter

INPUT-OUTPUT FORMATTING

NUMBER	(addr → d)	Convert string at address to double-precision number.
<#	(→)	Start output string.
#	(d → d)	Convert next digit of double-precision number and add character to output string.
#S	(d → 0 0)	Convert all significant digits of double-precision number to output string.
SIGN	(n d → d)	Insert sign of n into output string.
#>	(d → addr u)	Terminate output string (ready for TYPE)
HOLD	(c →)	Insert ascii character into output string.

DISK HANDLING

LIST	(screen →)	List a disk screen
LOAD	(screen →)	Load disk screen (compile or execute).
BLOCK	(block → addr)	Read disk block to memory address.
B/BUF	(→ n)	System constant giving disk block size in bytes.
BLK	(→ addr)	System variable containing current block number.
SCR	(→ addr)	System variable containing current screen number.
UPDATE	(→)	Mark last buffer accessed as updated.
FLUSH	(→)	Write all updated buffers to disk.
EMPTY-BUFFERS	(→)	Erase all buffers.

DEFINING WORDS

xxx	(→)	Begin colon definition of xxx.
	(→)	End colon definition.
VARIABLE xxx	(n →) xxx: (→ addr)	Create a variable named xxx with initial value n; returns address when executed.
CONSTANT xxx	(n →) xxx: (→ n)	Create a constant named xxx with value n; returns value when executed.
CODE xxx	(→)	Begin definition of assembly-language primitive operation named xxx.
CODE	(→)	Used to create a new defining word, with execution-time "code routine" for this data type in assembly.
<BUILDS . . DOES>	does: (→ addr)	Used to create a new defining word, with execution-time routine for this data type in higher-level Forth

VOCABULARIES

CONTEXT	(→ addr)	Returns address of pointer to context vocabulary (searched first).
CURRENT	(→ addr)	Returns address of pointer to current vocabulary (where new definitions are put)
FORTH	(→)	Main Forth vocabulary (execution of FORTH sets CONTEXT vocabulary).
EDITOR	(→)	Editor vocabulary; sets CONTEXT.
ASSEMBLER	(→)	Assembler vocabulary; sets CONTEXT
DEFINITIONS	(→)	Sets CURRENT vocabulary to CONTEXT.
VOCABULARY xxx	(→)	Create new vocabulary named xxx.
VLIST	(→)	Print names of all words in CONTEXT vocabulary.

MISCELLANEOUS AND SYSTEM

(	(→)	Begin comment, terminated by right paren on same line; space after (
FORGET xxx	(→)	Forget all definitions back to and including xxx.
ABORT	(→)	Error termination of operation
xxx	(→ addr)	Find the address of xxx in the dictionary; if used in definition, compile address
HERE	(→ addr)	Returns address of next unused byte in the dictionary.
PAD	(→ addr)	Returns address of scratch area (usually 68 bytes beyond HERE).
IN	(→ addr)	System variable containing offset into input buffer, used, *e.g.*, by WORD.
SP@	(→ addr)	Returns address of top stack item
ALLOT	(n →)	Leave a gap of n bytes in the dictionary.
	(n →)	Compile a number into the dictionary

FORTH

Order-No. 7055 disk $39.95

POWER-FORTH is a development of FIG-FORTH 1.1. POWER-FORTH consists of two parts. The first part is written in machine-language and will be loaded as you turn the computer on. The second part, which is on the backside of your disk, is written in FORTH and can be loaded only in FORTH.

This second part again consists of different parts (packages). For example the UTILITY package which should be loaded first, because it is required by the other packages. To do so use the FORTH-command '6 LOAD'. After the package has been loaded the computer will respond with 'OK'.

Besides the UTILITY-package there are others for screen-editing, for graphics and I/O, and for special player-missile graphics.

The commands for the different packages are described seperately later.

Besides the different packages POWER-FORTH includes a short arcade-game for demonstration of the PM-graphics.

Lots of fun with these powerful programs wishes :

ELCOMP PUBLISHING INC.
53 Redrock Lane
POMONA CA 91766

IMPORTANT:
Do not use the delivered disks for your own files. It will destroy itself!

Instead of HOME in FORTH-79 POWER-FORTH uses the definition LEER.

```
( CREATE FOR ADAPTING FORTH 79 OR POLY-FORTH TO FIG-
FORTH )

: CREATE 0 VARIABLE -2 ALLOT ;
```

Description of the POWER-FORTH kernel

Same like FIG-FORTH 1. 1 but the following differences :

?TERMINAL	asks the yellow keys. If no key is pressed then 0 is put to the stack, otherwise : START=1, OPTION=2, SELECT=4.
C/L	has the value 32 (characters/line)
LEER	erases the screen. : LEER 7D EMIT ;
HALLO	prints the copyright statement
MON	is illegal (jump $2B47)
.SCREEN	prints the contents of the screen via ports 3&4
SEC/DR	contains number of sectors/drive (720)
DUNIT	variable for drive #
DSTAT	(VAR) I/O STATUS after use of disk
IOSEC	(VAR) SECTOR#
IOBUF	(VAR) SECTOR-BUFFER ADDRESS
DRIVE	instead of DRO/DRI (n —>) n is drive # then you can work on another disk "direct", using command OFSET (drive 1 simulation)

POWER-FORTH always uses

IOCB#3 and #4
IOCB#3 input from keyboard
IOCB#4 output to screen

Note :
Not each block K*720 can be accessed (ERROR 139) this is sector zero of the disk.(k=[0,1...]

UTILITY PACKAGE

DOS	(—>)	jump to DOS (frontside of disk)
CASE	(—>)	similar to ON..GOTO in BASIC. For example : CASE A K L ; defines A and if A is called K or L is executed, depending on whether a 0 or a 1 is in the stack. This is true for n different words.
DUMP	(adr n —>)	prints n bytes starting at adr in hexadecimal and ASCII.
; :	(—>)	DECOMPILER. The word following ; : will be dispersed if it is a double function ;:(no blank)
'S	(—> SP)	puts actual value of stack-pointer to stack
.S	(—>)	prints stack without changing it
LOAD-ED	(—>)	loads editor-package
LOAD-IO	(—>)	loads I/O package
LOAD-PM	(—>)	loads player-missile package

DISKCOPY (—>) copies by sector with one drive This way it is possible to copy a disk with text (for 32k-version change in colordefinitions: "READ" and "WRITE" on SCR#11 B800 into 7C00 hex.

DR1->DR2(—>) copies drive1 to drive2 by sector.

Editor package

The editor described here is for creating text-screens with POWER-FORTH.

The editor is stored on the backside of the POWER-FORTH disk and can be loaded while in FORTH by

6 LOAD (loading the utility package)

and then

LOAD-ED

In a similar way you can load

LOAD-IO and
LOAD-PM

After the editor has been loaded it can be activated with command EDITOR. If a screen should be entered you first have to enter 45 LIST.

If the text area is already occupied then you have to search for another (empty) one. If it is empty only the line numbers appear (without text).

```
SCR # 45
  1
  2
  3
  4
  5
  6
  7
  8
  9
 10
 11
 12
 13
 14
 15
```

Now the editor expects the input of a command.

0 M (blank between 0 and M)

will place the cursor one line down. Now you can enter line 0.

You can enter a maximum of 32 characters. After 32nd character it will be cut off. For more than 32 characters you have to continue in the next line (in our case 1 M).

To check what has been entered so far use command L.

To finish the input enter FLUSH . This writes the contents of the buffer to disk.

Command EDITOR opens the vocabulary of the editor.

L (—>) erases the screen and shows the actual text-window (SCREEN) assigned to the variable SCR.

E (n—>) erases lines of text-window

S (n—>) inserts a blank line at line n

D (n—>) deletes line n

M (n—>) deletes line n and waits for input

CLEAR (n—>) deletes text-window n

COPY (n1n2—>) copies text-window n1 to n2

-> (—>) shows next text-array

<- (—>) shows preceeding text-array

I/O-Package

These commands bring addresses to the stack.(Now you can use them like variables)

ICCOM (—>adr) address of CIO-command-byte

ICSTA (—>adr) address of CIO-status-byte

ICBAD (—>adr) address of CIO-buffer

ICBLE (—>adr) address of CIO-buffer length

ICAX1 (—>adr) 1st CIO-auxiliary-byte

ICAX2 (—>adr) 2nd CIO-auxiliary-byte

TO# (n—>) changes channel # in CIO-addresses

To open the single devices

K:
S:
E:
P:
C:
} (—>adr) proper address to stack

Check of status-byte

?ICERR (—>) if there is an IO-error it is printed

Opening of a file

OPEN (FILE AUX1 AUX2 n—>)
opens IO-channel n. For example : E: 8 0 4 OPEN is the same as the BASIC-command OPEN#4,8,0,"E:"

CLOSE (n—>) closes IO-channel #n

GET (n—>d) gets byte d from IO-channel n

PUT (dn—>) sends byte d via channel n

PR-ON (—>) output goes via printer (P:)

PR-OFF (—>) undo command PR-ON

SOUND (n1 n2 n3 n4 —>)
turns on channel n4 with the frequency n3, the distortion n2, and the volume n1

RESSND (—>) turns off all sound channels

GR. (n—>) opens the screen on channel 6 with graphics-mode n (mixed mode)

GR.16 (n—>) like GR. but full graphics

SETCOLOR (n1 n2 n3 -->)
register n3 is loaded with color n2 and brightness n1

COLOR (n—>) register n is defined for commands PLOT and DRAWTO

PLOT (x y -->) a dot is printed at location x, y

DRAWTO (x y —>) a line is drawn from the actual position to position x, y

POSITION (x y -->) cursor is placed at x, y

.#6G"... prints string via channel 6 (suitable for MODE1 & MODE2)

STICK (n—>d) gets from stick n the value d

STRIG (n—>d) gets from trig n the value d

Player-Missile-Graphics with POWER-FORTH

POWER-FORTH comes with a package, which allows you to take advantage of the graphics- and sound-capabilities of your ATARI-800 also in FORTH.

Note:aexp=number on the stack
pnum=player number

The following commands are available :

HSTICK
VSTICK
aexpHSTICK
aexpVSTICK

The last two commands are for asking of the joysticks.

n VSTICK reads joystick #n and delivers the following to the stack :

-1 if the joystick is pushed forward
+1 if the joystick is pushed backward
0 if the joystick is in standard position

n HSTICK reads joystick #n and delivers :

+1 if the joystick is pushed to the right
-1 if the joystick is pushed to the left
0 if the joystick is in standard position

aexp PMG

This command is for initializing of the player-missile-graphics.
aexp can be 0, 1, or 2.

0 = turn off PMG
1 = turn on PMG for single line resolution
2 = turn on PMG for double line resolution

The two kinds of resolution will be called PMG-mode from now on. One bit defines, whether one or two lines on the screen are occupied. One line is the height of one pixel in graphics mode 8. In graphics mode 7 the height of one pixel is two lines on the screen, similar to 2 PMG (note the blank between 2 and PMG !)

Double line resolution only requires half of the memory necessary for single line resolution : 128 byte per player instead of 256.

PMCLR
format : pmnum PMCLR
example : 4 PMCLR

This command deletes the area in memory for that particular player. PMCLR considers the mode and only deletes the matching area.
Note, that 4 PMCLR and 7 PMCLR will clear all missiles, not just the named ones.

PMCOLOR
format : aexp aexp pnum PMCOLOR
example : 8 13 2 PMCOLOR

PMCOLOR is used in POWER-FORTH for a player-missile-set the same way SETCOLOR is used . There is no command corresponding with the general COLOR-command in PMG. This command is not necessary because each player has its own color. In the example above player 2 and missile 6 are set at a medium luminence (8) and a green color (hue 13).

Note that PMG has no default colors after RESET or after power up !

PMWIDTH
format : aexp pnum PMWIDTH
example : 1 2 PMWIDTH

Similar to how you can select single or double line resolution using command PMG you can define the width of players and missiles using command PMWIDTH. The only difference is that PMG defines the

resolution for all players and missiles while PMWIDTH allows you to define each player and missile separately. aexp defines the width and can be 0, 1, or 2.

PMMOVE
format : aexp aexp pnum PMMOVE
example : 1 120 0 PMMOVE
0 80 1 PMMOVE

Once a player or a missile has been defined it can be moved around on the screen with command PMMOVE. POWER-FORTH allows you to move around each player and each missile independently.

The second parameter in the command defines the position referring to the left border. That number can range from 0 to 255. The lowest and the highest positions are outside the visible area. PMWIDTH doesn't affect the position, it only enlargens the player to the right.

The first parameter in PMMOVE defines a relative, vertical movement. within 128 or 256 bytes of memory.

Vertical movements are achieved by moving bytes within memory. A relocation to higher memory locations corresponds with a down-movement on the screen, a relocation to lower memory locations corresponds with an up-movement on the screen. POWER-FORTH allows movements from -255 (255 pixels up) and +255 (255 pixels down).

Note :
The +/- definition referes to the values delivered by a joystick.

Example : 0 VSTICK 100 0 PMMOVE

This command moves player 0 up or down on the screen, depending on the joystick-position.

MISSILE
Format : aexp aexp pnum MISSILE
Example : 3 48 4 MISSILE

This command makes it easy to shoot with a missile. The second aexp defines the vertical position of the missile. (0 means top of screen). The first aexp defines the height of the missile.

Example : 3 64 4 MISSILE
Places a missile with a height of 3 or 6 lines (depending on the PMG-mode) at a position 64 pixels away from the top of the screen.

Caution :
MISSILE does not turn on bits automatically. The defined bits are worked up with the actual missile memory using an exclusiv-or function. That way you can delete existing missiles by creating other ones.

```
Example : 4 40 5 MISSILE
          8 40 5 MISSILE
```

The first command creates a 4 pixel high missile at position 40. The second command deletes the first missile and places a 4 pixel high missile at position 44.

PMADR
Format : aexp PMADR
Example : 0 PMADR

This function can be used in every arithmetic expression. It is used to find out the address of a player or a missile. It is helpful, if the programmer wants to bring data into the area of the player or if he wants to read data there.

Caution :
m PMADR, where m is a number between 4 and 7 delivers the same address for all missiles. they are in the same memory area.

BUMP
Format : aexp pmnum BUMP
Example : 0 1 BUMP IF ... ELSE ...
pnum=[0..3]or[4..7] aexp=[0..3]or[8..11]
player missile player playfield

BUMP may be used in all arithmetic expressions. This command allows to read the collision registers. It delivers a 1, if a collision has been detected, otherwise it delivers 0. This always referres to a a pair of objects. The first parameter (aexp) can be either a number of a player or a number of a playfield (8 - 11).

Possible BUMPs :

PLAYER -> PLAYER	(0-3 to 0-3)
PLAYER -> MISSILE	(0-3 to 4-7)
PLAYFIELD -> PLAYER	(8-11 to 0-3)
PLAYFIELD -> MISSILE	(8-11 to 4-7)

0=PLAYER1
1=PLAYER2
2=PLAYER3
3=PLAYER4
4=MISSILE1
5=MISSILE2
6=MISSILE3
7=MISSILE4
8=PLAYFIELD1
9=PLAYFIELD2
10=PLAYFIELD3
11=PLAYFIELD4

Caution :

p p BUMP where p is a value from 0 to 3 (both the same) always delivers 0. Example: 0 0 BUMP

It is recommended to set the collision registers to zero after a certain period of time. You can use for example : 0 0 BUMP, which doesn't bring a value to the TOS, but it deletes the collision registers.

SHAPE [byte],'aexp SHAPE NAME

This command allows to enter something or to read a stack in the FORTH dictionary.

For example :

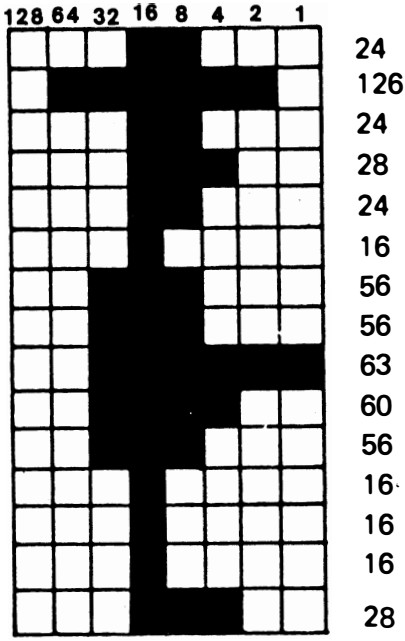
The figure can be used as a player.

```
28 16 16 16 56 60 63 56 56 16
24 28 24 126 24 15 SHAPE COWBOY
```

It is important that the bytes are in the stack in reverse sequence. Also important is the last number in the stack (TOS). This number defines the size of the shape.(length) For instance:COWBOY now became a new command.

COWBOY

Format : addr COWBOY
Example : 0 PMADR 128 + COWBOY
will place the cowboy starting at the 128th vertical position off player 0. The player will be displayed on the screen.



Demonstration for player/missile

This demonstration program is well suited to learn how to program your own games with player missile graphics.
There are different screens. We start the description with screen 55.
To start the program we first have to enter the following commands :

```
LOAD-ED
LOAD-IO
LOAD-PM
55 LOAD
```

After that the program can be started by entering GAME.

```
SCR # 55
  0 ( DEMO GAME HCW )
  1 0 VARIABLE POINTS 0 VARIABLE XM
  2 0 VARIABLE YM 0 VARIABLE YV
  3 0 VARIABLE XV 0 VARIABLE X
  4 0 VARIABLE YM1
  5 126 126 126 3 SHAPE RACKET
  6 HERE VARIABLE RND
  7 : RANDOM RND @ 31421 * 6972 +
  8           DUP RND ! ;
  9 : RNDNR RANDOM U* SWAP DROP ;
 10
 11 : PFINI 5 GR. 0 0 2 SETCOLOR
 12          1 COLOR 0 0 PLOT
 13          79 0 DRAWTO 79 39 DRAWTO
 14          0 39 DRAWTO 0 0 DRAWTO ;
 15  -->
 OK
```

First we define the variables needed.

POINTS is for the score
XM and YM are for the coordinates of the missile
XV and YV are for the movement-vector of the missile
YM1 is for the next to last position of the missile
X is for the X-coordinate of the player. The shape of the player is defined in line 5. In our example we use a random number generator (lines 6-9).
In lines 11 through 14 the command for drawing the playfield is defined (very similar to BASIC commands).

```
SCR # 56
  0 ( DEMO GAME HCW )
  1 : PLINI 2 PMG 15 5 0 PMCOLOR
  2         0 PMCLR 1 0 PMWIDTH
  3         0 PMADR 85 + RACKET
  4         120 X ! 0 120 0 PMMOVE ;
  5
  6 : INIT PFINI PLINI 0 POINTS ! ;
  7
  8 : INITM 4 PMCLR 1 19 4 MISSILE ;
  9
 10 : MVECT 2 RNDNR 1+ YV !
 11         2 RNDNR 2 + MINUS XV !
 12         19 DUP YM ! YM1 !
 13         120 XM ! ;
 14
 15 --->
 OK
```

PLINI will activate and initialize the player/missile graphics. "2 PMG" turns on the player/missile graphics, which results in a pink color for the player and the missile. PMWIDTH results in twice the width. Line 3 places the previously defined shape at the 85th vertical position and finally the player will be placed in the middle of the screen.

INIT combines these two commands and in addition deletes the variable POINTS.
INITM deletes the missile area and generates the missile belonging to the player at the 19th vertical position.
MVECT calculates the start vectors of the missile using the random number generator. It also defines the start coordinates of the missile.

```
SCR # 57
  0 ( DEMO GAME HCW )
  1 : SETP 0 HSTICK 2 * DUP 0<
  2          IF X @ 47 > *
  3          ELSE X @ 193 < *
  4          THEN X +!
  5          0 X @ 0 PMMOVE ;
  6 : MXY XV @ XM +! YV @ YM +! ;
  7 : MM YM @ YM1 ! MXY
  8       XM @ 50 < XM @ 205 > OR
  9       IF 15 20 10 0 SOUND RESSND
 10          XV @ MINUS XV ! MXY MXY
 11       ENDIF
 12       YM @ 19 < YM @ 92 > OR
 13       IF 15 20 10 0 SOUND RESSND
 14          YV @ MINUS YV ! MXY MXY
 15       ENDIF ;                  -->
 OK
```

SETP moves the player horizontal within the borders. Very handy for that is command HSTICK which converts the joystick position into a number relative to the actual position (-1 0 +1).
MXY adds the vectors YV and XV to the actual coordinates of the missile.
MM checks whether the missile is within the borders of the playfield. If it is not it lets it rebounce with the same angle that it hit the border. During the bounce a noise is turned on for a short time. (lines 9 and 13).

SETM moves the missile relative to the last position (YM1).
SCORE prints the score for the player.
WAIT waits for the red fire button on the joystick to be pressed.

```
SCR # 58
  0 ( DEMO GAME HCW )
  1 : SETM YM @ YM1 @ - XM @ 4
  2                       PMMOVE ;
  3 : .SCORE  ." SCORE " POINTS ? ;
  4 : WAIT BEGIN 0 STRIG 0= UNTIL ;
  5 : GAME INIT BEGIN INITM MVECT
  6                   0 0 BUMP
  7                   BEGIN SETP
  8                         SETM
  9                         MM
 10                         0 4 BUMP
 11                   UNTIL
 12                   1 POINTS +!
 13                   LEER .SCORE
 14                   WAIT ?TERMINAL
 15             UNTIL ;       ;S
 OK
```

GAME combines all commands described so far. If you enter GAME now the game will start. After a score a a game may be interrupted by pressing the red fire button and the yellow START key at the same time.

Demo for HSTICK and VSTICK

This program can be loaded with '59 LOAD' and started with 'DEMO'.

Screen 59 contains a short sample program for the use of VSTICK and HSTICK.

```
SCR # 59
  0 ( DEMO HSTICK VSTICK HCW )
  1 4 DIM X
  2 : MOV >R R HSTICK 2 * R X +!
  3       R VSTICK 2 * R X @ R>
  4                      PMMOVE ;
  5
  6 : INI 120 0 X ! 150 1 X !
  7       2 PMG 10 5 0 PMCOLOR
  8       10 13 1 PMCOLOR 1 PMCLR
  9       0 PMCLR 0 PMADR 60 +
 10       COWBOY 1 PMADR 60 +
 11       COWBOY 0 120 0 PMMOVE
 12       0 150 1 PMMOVE ;
 13 : DEMO INI BEGIN 0 MOV 1 MOV
 14              ?TERMINAL UNTIL ;
 15
 OK
```

In line 1 an array of size 4 is defined.
MOV moves a player depending on the joystick position.
HSTICK delivers a value different from 0 if STICK indicates a horizontal movement (-1 for left, +1 for right).
VSTICK is the same as HSTICK, but for vertical movements (-1 for up, +1 for down).

DICTIONARY

```
VLIST
BUMP    BMP    HITCLR    MBUMP    PBUMP
MFBUMP    MPBUMP    PPBUMP    PFBUMP
MASK    MSK3    MISSILE    VSTICK
HSTICK    HORI    VERT    MONE    ONE
NULL    COWBOY    SHAPE    PMCOLOR
PMMOVE    MMOVE    MVMOVE    MUPDOWN
MDOWN    MUP    PMOVE    PVMOVE    PUPDOWN
PUP    PDOWN    PMWIDTH    PWIDTH
MWIDTH    MSK2    MSK1    4^    DIM
PMCLR    PMADR    PMG    PMGO    PMG2
PMG1    MSIZ    PT    PMBAS    GRACTL
DMACTL    PMBASE    PMOFFSET    INIT2
INIT1    (MUP)    (MDWN)    (PDOWN)
(PUP)    NEXT    STRIG    STRIG4    STRIG3
STRIG2    STRIG1    STICK    STICK4
STICK3    STICK2    STICK1    .#6"
(.#6")    TYPE#6    POSITION    PLOT
2DUP    DRAWTO    COLOR    COLR    SETCOLOR
GR.16    GR.    GRN    RESSND    SOUND
PR-OFF    PR-ON    PUT    GET    CLOSE
OPEN    ?ICERR    C:    P:    E:    S:
K:    FILE    TO#    ICAX2    ICAX1
ICBLE    ICBAD    ICSTA    ICCOM    +IONO
IOCB    CIO    (CIO)    EDITOR    DR1->DR2
DISKCOPY    WRITE    ?SECW    READ
SECW    SEC    LOAD-PM    LOAD-FLOAT
LOAD-IO    LOAD-ED    .S    DEEP    'S
;:    ?;    ?;S    ?;CODE    PFA->ID.
?COLONDEF    DOCOL    GETPFA    NOT
U.R    DUMP    .ADR    .ASC    .HEX
CASE    DOS    INDEX    LIST    R/W
-DISC    -BCD    -->    LOAD    .LINE
(LINE)    BLOCK    BUFFER    DRIVE
EMPTY-BUFFERS    FLUSH    UPDATE
?LOADING    +BUF    PREV    USE    IOBUF
IOSEC    DSTAT    DUNIT    SEC/DR
```

```
B/SCR   B/BUF   LIMIT   FIRST   .SCREEN
MON   HALLO   LEER   VLIST   ?
U.   .   .R   D.   D.R   #S   #
SIGN   #>   <#   SPACES   WHILE
ELSE   IF   REPEAT   AGAIN   END
UNTIL   +LOOP   LOOP   DO   THEN
ENDIF   BEGIN   BACK   FORGET   '
MESSAGE   M/MOD   */   */MOD   MOD
/   /MOD   *   M/   M*   MAX   MIN
DABS   ABS   D+-   +-   S->D   COLD
ABORT   QUIT   (   DEFINITIONS
FORTH   VOCABULARY   IMMEDIATE
INTERPRET   ?STACK   DLITERAL   LITERAL
[COMPILE]   CREATE   ID.   ERROR
(ABORT)   -FIND   NUMBER   (NUMBER)
WORD   PAD   HOLD   BLANKS   ERASE
FILL   QUERY   EXPECT   ."
(.")   -TRAILING   TYPE   COUNT
DOES>   <BUILDS   ;CODE   (;CODE)
DECIMAL   HEX   SMUDGE   ]   [
COMPILE   ?CSP   ?PAIRS   ?EXEC
?COMP   ?ERROR   !CSP   PFA   NFA
CFA   LFA   LATEST   TRAVERSE   -DUP
SPACE   ROT   >   <   U<.   =   -
C,   ,   ALLOT   HERE   2+   1+
HLD   R#   CSP   DPL   BASE   STATE
CURRENT   CONTEXT   OFFSET   SCR
OUT   IN   BLK   VOC-LINK   DP
FENCE   WARNING   TIB   WIDTH   S0
+ORIGIN   C/L   BL   3   2   1
0   USER   VARIABLE   CONSTANT
;   :   C!   !   C@   @   TOGGLE
+!   DUP   SWAP   DROP   OVER   MINUS
D+   +   0<   0=   R   R>   >R
LEAVE   ;S   RP!   SP!   SP@   XOR
OR   AND   CR   U/   U*   CMOVE
?TERMINAL   KEY   EMIT   ENCLOSE
(FIND)   DIGIT   I   (DO)   (+LOOP)
(LOOP)   OBRANCH   BRANCH   EXECUTE
CLIT   LIT   OK
```

In addition to POWER-FORTH there is a floating point package available.

Loading of the floating point package :

1. go to DOS
2. load FLOAT.OBJ using command L
3. go back to FORTH
4. turn diskette (use backside)
5. enter LOAD-FLOAT while in FORTH

Floating Point Package

F	(—>f)	after this command you can enter a floating point number
.F	(f—>)	prints the floating point number which is in the stack
F+	(f1f2—>fs)	adds two FP-numbers
F-	(f1f2—>fu)	subtracts two FP-numbers
F*	(f1f2—>fm)	multiplies two FP-numbers
F/	(f1f2—>fo)	divides two FP-numbers
LN	(f—>fin)	calculates the LN of a number
LOG	(f—>flog)	calculates the LOG of a number
EXP	(f—>fexp)	calculates the e-power of a number
FIX	(f—>n)	changes a FP-number into a integer number
FLOAT	(n—>f)	changes an integer number into a FP-number
FDUP	(f—>ff)	doubles a FP-number

FSWAP (f1f2-->f2f1) exchanges two FP-numbers

FVAR (f-->) defines a FP-variable

FCON (f-->) defines a FP-constant

F! (fadr-->) stores a FP-number at adr

F@ (adr-->f) gets a FP-number from adr

F1/ (f-->fr) calculates the reciprocal value

^ (f1f2-->f1↑f2) calculates f1 power f2

SQRT (f--> f) calculates the square root

PI (-->π) the number π is put to the stack

FLIT comp. (f-->) like LITERAL but for FP-numbers
transit (-->f)

COS
SIN
TAN } (f-->fr) calculates the goniometric functions (0 - 90°)

F= (f1f2-->flag) like = with integers

F> (f1f2-->flag) like > with integers

F< (f1f2-->flag) like < with integers

Order-No. 7223 $29.95

HOW TO USE THE ASTROLOGY-PROGRAM FOR THE ATARI 800

The Astrology Program from ELCOMP Publishing allows you to calculate a complete horoscope. The program prints out the following:

1. Planetary positions
2. Houses
3. Ascendent
4. Aspects

To run the program you must know:

1. Your birthdate
2. The location (longitude and latitude) of your birth
3. The birth time HH,MMSS
4. The time zone (how many houses away from Greenwich)
   East of Greenwich is positiv, west of Greenwich is negative.
   Note: Do not forget the daylight savingstime, war time a. s. o.

How to start the program

a) Insert Disk into disk drive I
b) Enter DOS
c) LOAD binary file ASTROZ.BIN
d) GOTO BASIC-type B
e) RUN "D:SC4"
f) Type in : 1 and 24
g) Wait one minute
h) Type : N
i) RUN "D:ASTRO.BAS"
j) Type in your information
k) You can print out the screen via:

SHIFT-S
SHIFT-D
SHIFT-X
SHIFT-C

if you have a MX-82 FT from EPSON or a MX-100.

EPROM Board (Cartridge), Order-No. 7043 $29.95 /// EPROM Board KIT, Order-No. 7224 $14.95

EPROM Cartridge for the ATARI 800/400

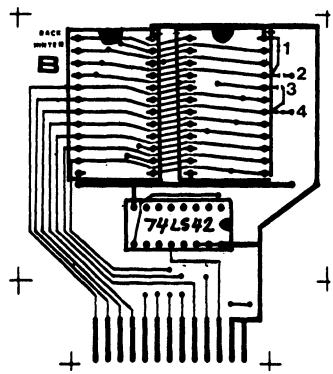
With this versatile ROM-Module you can use
2716
2732
and 2532 type EPROMs

To set the board for the specific EPROM, just solder their jumpers according to the list shown below. Without any soldering you can use the module for the 2532 right away.

If you use only one EPROM, insert it into the right socket. If you use two EPROMs, put the one with the higher address into the right socket.

The modul must be plugged into the left slot of your ATARI-computer with the parts directed to the back of the computer.

EPROM	2716	2732	2516	2532
1	V	O	V	V
2	O	V	O	O
3	V	V	V	O
4	O	O	O	V



V = means connected (jumper)
O = means OPEN

ATARI is a trademark of ATARI Inc.

Order-No. 7210	cassette version	$29.95
Order-No. 7216	disk version	$34.95
Order-No. 7217	cartridge version	$69.00

ATEXT

PREFACE

ATEXT comes on cartridge, disk, or cassette.

The disk version contains a file called AUTORUN.SYS and other system-files described in APPENDIX IV.

All versions of ATEXT are bootable. After the program was loaded, the copyright statement will come up. Now press K and the text window will come up.

To boot the cassette version hold down the START-key when turning on the computer and after the beep press PLAY on the recorder and RETURN on the computer.

The cartridge version plugs into the left slot and is ready after you turn on the computer.

IMPORTANT:
Do not use the delivered disks for your own files. It will destroy itself!

THE EDITOR

The EDITOR has two different modes :
Control-commands and commands in the command-line at the bottom of the screen.

The following control-commands are available :

CTRL A	cursor to next format-command
CTRL C	delete up to next format-command
CTRL D	close copy-register
CTRL E	open copy-register
CTRL G	repeat command-line
CTRL H	delete last character
CTRL I	cursor to next TAB-position
CTRL J	insert contents of copy-register at current cursor position
CTRL K	delete copy-register
CTRL N	delete last word
CTRL O	cursor to end of preceeding word
CTRL P	cursor to beginning of next word
CTRL S	cursor to end of text
CTRL T	display control-characters (on/off)
CTRL U	delete next character
CTRL V	delete to last format-command
CTRL W	cursor to beginning of text
CTRL X	delete last line
CTRL Z	delete next line
CTRL -	cursor to beginning of preceeding line
CTRL =	cursor to beginning of next line
CTRL +	cursor one character backwards
CTRL *	cursor one cahracter forwards
ESC	open/close comand-line

All CTRL-characters not mentioned above will be inserted at the current cursor-position.

The following commands are available in the command-line :

@	set TAB-position
B	one character backwards
C	change string
D	delete preceeding character
E	delete string
F	one character forwards
G	insert contents of copy-register
H	insert hexbyte
I	insert string
J	repeat command-line
K	delete textbuffer !!
L	call formatter
M	jump to DOS
R	read file via CIO
S	search for string
T	delete next line
W	write file via CIO

You reach the command-line by using the ESC-key. The first dot in the command-line will be replaced by the dollar-sign. Each character typed in now will appear in the comandline. The command-line is terminated by pressing the ESC-key twice. Then the comand-line wil be executed and the number-character (#) will be displayed.

There can be more than one command on comand-line if they are seperated by single ESC-signs. Some commands can be executed several times by entering a number from 2 to 255 in front of the command. You also can repeat the whole line using the J-command which brings you back to the beginning of the line. Of course this is possible only if the command was executed. It is not possible if an error occured, because then the program jumps back to the control-mode.

In the upper lefthand corner of the screen there is a counter, displaying the cursor position in the current line. If you write a line longer than 255 characters, then the cursor will disappear. To warn you, there will be a beep starting at the 250th character.

Beginning at the 20th character the text of the current line will be scrolled.

The second counter shows the number of characters in front of the cursor. The third counter shows the remaining free memmory locations. The fourth counter shows the remaining free locations in the copy register.

In the upper righthand corner the status of the copy-register (OK = closed, CR = opened) or error-messages of the editor are displayed.

THE COPY-REGISTER

You can read in a part of your text backwards into the copy-register by using the commands CTRL +, CTRL -, CTRL O, CTRL Q or CTRL W. It also is possible to delete text and read it into the copy-register at the same time by using the commands CTRL H, CTRL X, CTRL V, or CTRL N.

If you are finished with reading in, you can close the copy-register with CTRL D.

Now the text in the copy-register can be inserted as often as needed with CTRL J.

The command CTRL K deletes the contents of the copy-register.

More detailed description of the command-line commands :

CHANGE (C)

Format: C<string1>ESC[/<stringn>ESC]'<string2>ESC
Example: $CAtext 1.0$/Atext$/text$TEXT$#..
This commandline changes Atext 1.0 into ATEXT 1.0
Strings preceeded by / are searched locally, this means that nested search is possible.

ERASE (E)

Format: E<string>ESC
Example: $EAtext$#...
Next 'Atext' will be erased

HEX (H)

Format: H<byte>ESC
Example: $H1D$#..
Inserts the hex-value 1D which is an end-of-file marker for a disk-/cassette-file.

INSERT (I)

Format: I<string>ESC
Example: $IHello$#...
Inserts the word 'Hello' at the actual cursor-position

SET TAB (@)

Format: @<n>ESC
where n=[1..9]

LIST (L)

Format: L[<2>|<file>]ESC ESC
Sends text starting at actual cursor-position via the formatter to a device.

Devices:

LE:	screen (unsplitted)
LP:	printer
LR:	RS-232
LDn:name	diskdrive #n (n=[1..4])

After you have saved the file on disk it is possible to send the file to a printer using the COPY-command of DOS.
Example in DOS: D1:TEST.TXT,P:

Splitted screen:

A single L Sends only the first 38 characters to

the screen, a L followed by a 2 sends all characters from the 38th position to the screen-device.

Example: L#.....
The first 38 characters of a line will be displayed on screen.

$L2$#...
The characters from the 38th position will be shown.

READ (R)

Format: R<file>ESC ESC
Inserts file at actual cursor-position.
Example: RDn:name reads from diskdrive (n=[1..4])
RC: reads from cassette

WRITE (W)

Format: W<file>ESC ESC
Writes a file from actual cursor-position to hex 1D or to end of text
Example: WDn:name writes to diskdrive (n=[1..4])
WC: writes to cassette

THE FORMATTER

A format-command line looks like the following:

CTRL-L[comm]'<CR>

Each command-line, even empty ones, puts a space between two paragraphs.
You can put as many commands in one line as you want. It is not necessary to seperate the commands.

There are the following format-commands:

A<T|F>	automatic line feed after each paragraph default is false.
C<T|F>	center next lines. You only can center if left and right margins were defined earlier. This command has priority over the right margin justification. Default is false.
D[n]	insert n line-feeds
E[n]	indent n blanks at each new paragraph. Default is zero. This only works after the definition of the right margin.
F<T|F>	right margin justification. Default is false
H	Stop formatting and wait for pressing 'OPTION' or 'SELECT' to continue. 'START' is abort formatting. Use this command for exchange of the font at a spinwriter e.g.
I<file>	include file. For the file there are the same commands like with the read/write command.

L[n] left margin is n blanks. Default is zero. The left margin has to be bigger or equal the right.

N<T[n]|F>pagenumbering starting at page n. The first pagenumber is not printed. The following numbers are printed centered at the top of the page.

O sets left and right margin as well as the automatic line feed to the default values.

P form feed

R[n] right margin. If n=0 then it is no longer printed in the zigzag-mode.

S[n] n lines per page. Default is 56

W[char] write char directly to the printer. This command allows you to send control commands to the printer. For example (hex 0F) turns the EPSON MX80 to the narrow font.

#[n] define new form-length. Default is 72 For US-standard set form-length in the beginning of the text to 66.

@[n] define new form-width. Default is 80

APPENDIX I

ERROR MESSAGES OF THE FORMATTER:

CIO ERROR DURING PRINT
An error occurred in the CIO-routine during print of text or the BREAK-key has been pressed.

CIO ERROR DURING INCLUDE
Something went wrong during the reading of the file or the BREAK-key has been pressed.

LINE TOO LONG
The formatter reached a line longer than 256 characters.

ILLEGAL COMMAND
An unknown command is in the format-command line.

ZERO IS NOT ALLOWED
A zero is at not allowed location.

T(RUE OR F(ALSE EXPECTED
The user forgot a T or a F.

CIO ERROR DURING OPENING FILE
During the opening of an include-file an error occurred. Maybe the file does not exist at all.

FORMLENGTH LOWER THEN PAGELENGTH
The formlength never can be smaller than the pagelength

INSPACES GREATER THEN (RIGHTM-LEFTM)
You tried to indent more than the difference between left and right margin.

LEFTM GREATER THEN RIGTHM
The left margin is bigger than the right.

(RIGHTM-LEFTM) TOO SMALL
During justification no blank was found in a line. Try to seperate a long word.

LIST ABORT WITH 'START'
You stopped the printout by pressing the yellow START-key.

'START' DURING HALT
You stopped the printout by pressing the yellow START-key during the H-command.

After the error-message you can press any key and the cursor stays at the location of the error.

APPENDIX II

THE USE OF TABS WITH ATEXT

To print schedules or charts you need a function to indent several lines the same number of blanks. To do that conveniently there is a TAB-function.

The TAB-function only works in the unformatted print-mode.

The TAB-function sets the cursor to the next predefined loaction. The default value for the distance between the single TAB-positions is 9 blanks. This can be changed with the fuction TAB SET (@) in the command-line.

APPENDIX III

EXPLANATION TO THE SPECIAL CHARACTERS

ESC this character stands for the one time use of the ESC-key.

[..] the characters or numbers between these brackets can be left out. Then 1 is assumed for the value, except with the commands R, L, and E of the formatter.

[..]' means that the contents of the brackets can be repeated as often as needed

<..> the contents of these brackets can not be left out

| this symbol means OR

F,T these two characters stand for true or false. This means you can turn a function on or off.

APPENDIX IV

PRINTING VIA PORTS OR VIA ELCOMP-RS232-INTERFACE

On the disk with the wordprocessor there are two additional files which can be loaded while in DOS.

If you load the file 'CENTRNX.SYS' you can print on a printer with CENTRONIX interface. The signals go via gameports 3 and 4 and the ELCOMP interface (for details see construction article). (See page 1)

If you load the file 'RS232.SYS' you can use the ELCOMP-RS232 interface via port 3 (for details see construction article). (See page 9)

After you loaded one of the two files you get back to the editor by using command 'B' of DOS.

Note that you have to load the files again after you pressed the RESET-key.

The cassete-version of ATEXT does not have the features described above.

APPENDIX V

ERROR-MESSAGES OF THE EDITOR
The error-messages of the editor of ATEXT are only two characters long and are displayed in the upper righthand corner of the screen.

CO	commandline overflow; more than 40 characters in the commandline
E?	invalid character in commandline
#?	the number in front of a command is larger than 255
CH	a string is missing at the command C
I?	no more space in textbuffer
C?	copy-register full
H?	wrong hex-number at command H
T?	TAB-value larger than 9 or smaller than 1
S?	string not found

L? something went wrong during execution of the L-command. (wrong file or stop during printout)

RW read-/write-error. (file does not exist or error during read-/write-operation)

APPENDIX VI

SEMI-AUTOMATIC SEPERATION

ATEXT has a feature to tell the formatter where a word can be seperated. To do so you have to insert a CTRL B between the syllables. Then, if it becomes necessary, the formatter seperates the word there.

Please note that the counter in the upper lefthand corner of the screen counts one character to much, because of the additional CTRL B.

inserted in front of it. This can be seen, if you enter CTRL-T.

To print a letter several times the include-command has to be used. The letter has to be on disk and should contain a formfeed command at the end. Clear the textbuffer (command K in the command line) and enter :

CTRL-L ID:LETTER (RETURN)

Open the copy register with CTRL-E, move the cursor up one line, and close the copy register with CTRL-D. Now the copy register contains the above command. This command can be inserted now as often as needed, using command CTRL-J (insert contents of copy register at actual cursor position).
Place the cursor at the beginning of the text with CTRL-L.

(ESC)LE(ESC)(ESC)

will now read the file from disk and print it on the screen as often as wished.

(ESC)LP(ESC)(ESC)

will print it on the printer.

ELCOMP PUBLISHING INC.
53 REDROCK LANE
POMONA, CA 91766
(714) 623-8314

YOUR ORDER NO.	SLSMN.	TERMS	DATE SHIPPED

Dear customer :

This is to inform you that ELCOMP has published a new book, called "Games for the ATARI". The book describes step by step, how to program your own games. It tells you how to create players and missiles and how to move them on the screen. How to create sound, to generate special effects, etc. There are many ready to type in and run programs listed in the book. One program is in machine-language. It is listed in assembler language and as hex-dump and is completely commented. The name of that program is GUNFIGHT.
Price for the book is $7.95.

Sincerely yours,

Linda Schwarz, office manager

BUSINESS ACCOUNT : BANK OF AMERICA BANK NUMBER 16-8 / 1220

This letter is written with ATEXT. For information refer to page 84.

A SAMPLE SESSION WITH ATEXT-1

The wordprocessor ATEXT-1 has so many commands and features, that it may be a bit confusing for those of you with little experience with computers. But once you know how to use all the different commands you won't want to miss them anymore.

Here is what you have to enter to get a printout like the attached letter (this letter is not a standard letter, it is meant only to show different techniques with the wordprocessor) :

CTRL-L O (RETURN)	set default values
CTRL-L D4 (RETURN)	four line feeds
CTRL-L R66L10 (RETURN)	right margin at 66, left margin at 10
CTRL-L CT (RETURN)	centering mode lines will be printed centered, if possible, otherwise they are printed normal

ELCOMP PUBLISHING INC. (RETURN)

CTRL-L (RETURN)	a CTRL-L is needed to start new lines a a RETURN doesn't start a new line !

53 REDROCK LANE (RETURN)

CTRL-L (RETURN)

POMONA, CA 91766 (RETURN)

CTRL-L (RETURN)

(714) 623-8314 (RETURN)

CTRL-L D5 (RETURN)	five line feeds
CTRL-L CF (RETURN)	no longer centering
CTRL-L O (RETURN)	no margins

YOUR ORDER NO. (12 blanks) SLSMN. (11 blanks)
TERMS (12 blanks) DATE SHIPPED (RETURN)

CTRL-L D5 (RETURN)	five line feeds
CTRL-L L10R66 (RETURN)	new margins
Text	enter the text here. If one line isn't enough, just hit RETURN and continue on the next line. Everything will be printed one by one later. Only CTRL-L causes the start of a new line. Lines on the screen can be up to 255 characters long, but it is recommended to go to the next line after about 100 - 150 characters (see counter in the upper lefthand corner). This makes editing easier. With long words you should make suggestions for division. Insert CTRL-B between the syllables. For example: in ter est ing.
CTRL-L D4 (RETURN)	four line feeds

CTRL-L L30 (RETURN) new left margin

Sincerely yours, (RETURN)

CTRL-L D3 (RETURN) three line feeds

Linda Schwarz, office manager (RETURN)

CTRL-L 0 (RETURN) no margins

CTRL-L D12 (RETURN) twelve line feeds

BUSINESS ACCOUNT : BANK OF AMERICA (15 blanks)
BANK NO. 16-8 / 1220 (RETURN)

It is not necessary to start a new line for each formatter command. For example a line could look as follows :

CTRL-L D5CF0 (five line feeds, centering mode off, default margins)

Now we want to save that letter. To do that we first have to place the cursor at the beginning of the text, which is done with CTRL-W. Then we press ESC, to get to the command line at the bottom of the screen. A "$" will be displayed there. We want to save the letter on disk under the name "LETTER". To do that we enter : WD:LETTER(ESC)(ESC)
Now the text will be saved. After a successful save a "#" will be displayed in the command line. If you don't have a disk drive, you can save the text on cassette with : WC(ESC)(ESC)

Next we want to print the letter on the screen. To do that we first have to leave the command line by pressing ESC two times. Then we place the cursor at the beginning of the text with CTRL-W. We go to the command line again by pressing ESC. There we want to call the formatter (L) and send the output to the screen (device E). Thus we have to enter : LE(ESC)(ESC)

To send the output to another device just enter P (for a parallel printer) or R (for a seriell printer) instead of the E.

To load the file from disk later we enter :

(ESC)RD:LETTER(ESC)(ESC)

To change a word in the text we place the cursor at the beginning of the file (CTRL-W), go to the command line (ESC), and search for the word to be changed (command S). If we want to change the word "called" into "named", we proceed as follows :
We go to the command line (ESC) and there we enter :

Scalled(ESC)(ESC)

This will place the cursor behind the first word "called" after the actual cursor position. A CTRL-G would move the cursor to the next word "called" if there would be one. Now we can delete the word "called" with command CTRL-N (delete last word) and insert the new word.
Another way to do that is to use command C (change). Go to the command line (ESC) and enter :

Ccalled(ESC)named(ESC)(ESC)

The command in the command line always can be repeated with CTRL-G. Then the actual command is executed again, effective from the actual cursor position on.

If you want to use the same form of a letter for a different text, then you can remove the old text by moving the cursor to the beginning of the text and deleting all lines you want removed with command CTRL-Z. Then you can create space for the new text by pressing RETURN several times. If you enter the new text later, then terminate the lines with CTRL-=, rather than with RETURN, because RETURN is already there and the new text will be

Order-No. 7042	**completely assembled**	**$179.00**
Order-No. 7292	**KIT**	**$ 49.95**

EPROM-BURNER FOR THE ATARI 800/400®

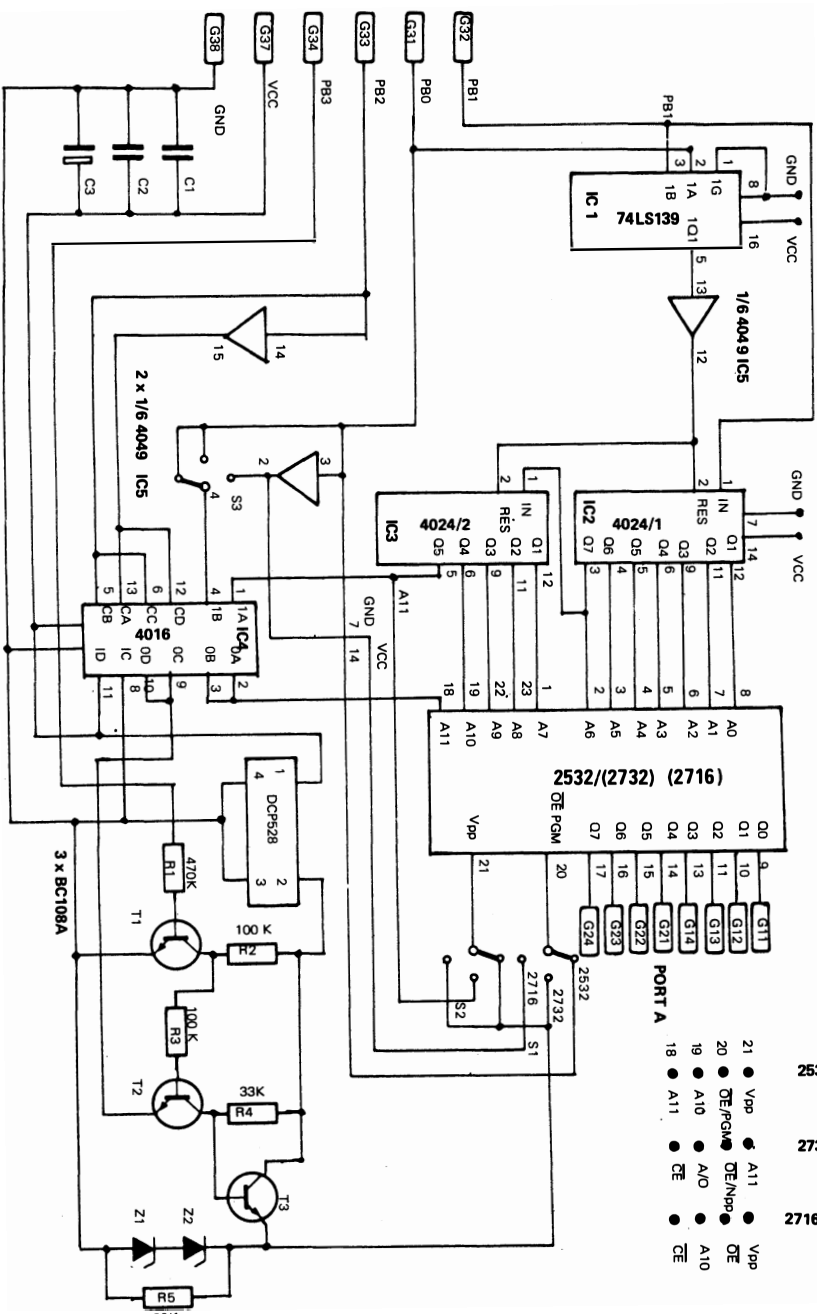
EPROMBURNER FOR THE ATARI 800/400.

With this epromburner you can burn your own EPROMS.It is possible to burn four different types. The four types are the 2532(4k), the 2732(4k),the 2516(2k) and the 2716(2k). The burner uses the game ports 1 ,2 and 3.

1) THE HARDWARE.

The circuit of the epromburner is shown in FIG. 1.The data for the burner is exchanged via game port 1 and 2. The control signals are provided by game port 3.The addresses are decoded by two 7 bit counters 4024. The physical addresses for the EPROMS are always in the range of 0000 to 07FF for 2k and 0000 to 0FFF for 4k. This counter is reset by a signal, decoded from PB0 and PB1 via the 74LS139. PB2 is used to decide if a 2532, or a 2716 has to be burned.
Not all signals for the different types of EPROMS are switched by software.A three pole, double throw switch is used to switch between the different types.The software tells you when you have to set the switch into the correct position. For burning, you need a burning voltage of 25 Volts.This voltage is converted from the 5 Volts of the game port to 28 Volt by the DCDC converter DCP 528. This voltage is limited to 25 Volts by two Zener diodes in serie (ZN 24 and ZN 1). Three universal NPN transistors are used to switch between low level voltages and the high level of the burning voltage.

Fig. 1


G38
G37
G34
G33
G31
G32
PB3
PB2
PB0
PB1
GND
VCC
C3
C2
C1
IC 1
74LS139
1G
1A
1B
1Q1
1/6 4049 IC5
2 x 1/6 4049 IC5
S3
IC3
4024/2
IC2
4024/1
RES
Q1
Q2
Q3
Q4
Q5
Q6
Q7
A11
4016
IC4
2532/(2732) (2716)
A0
A1
A2
A3
A4
A5
A6
A7
A8
A9
A10
OE PGM
Vpp
DCP528
470K
R1
T1
100 K
R2
100 K
R3
T2
33K
R4
T3
Z1
Z2
R5
33K
3 x BC108A
S1
S2
2716
2732
2532
PORT A
G11
G12
G13
G14
G21
G22
G23
G24
2532
2732
2716 / 2516
Vpp
A11
OE/Vpp
A10
A9
A8
A/0
CE

2) ASSEMBLING THE BURNER.

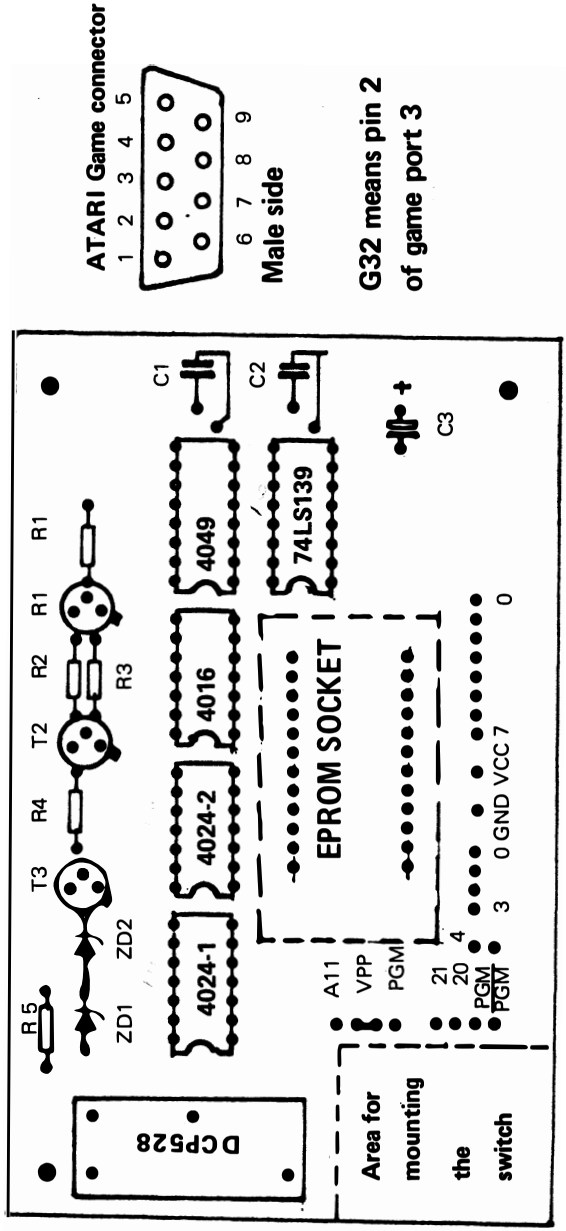


Fig. 2: Parts Layout

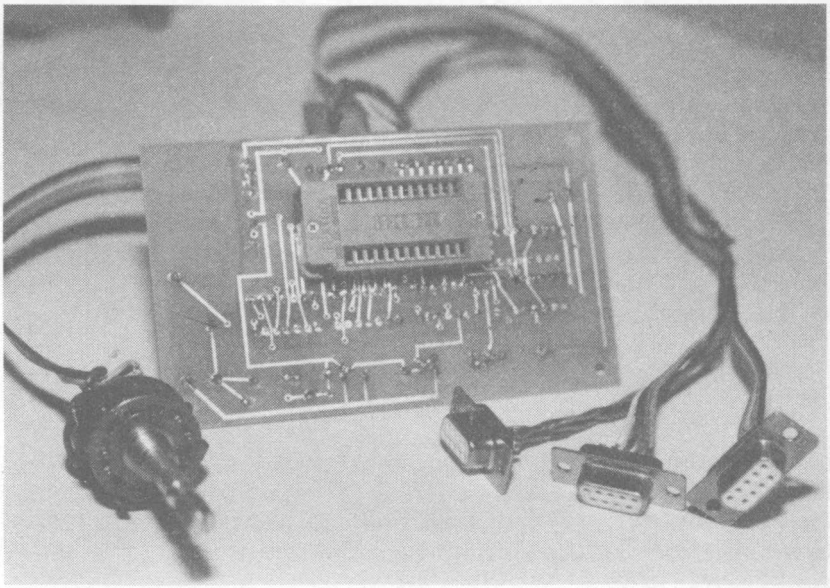


Figure 3

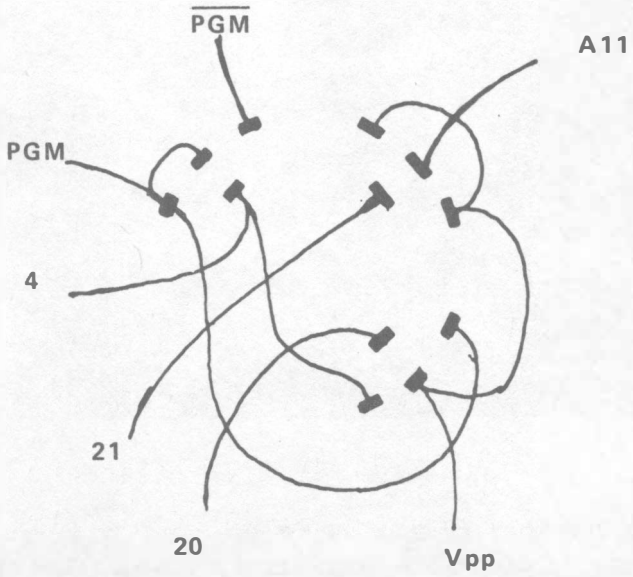


Fig. 4: Rear side of the 3P2T switch

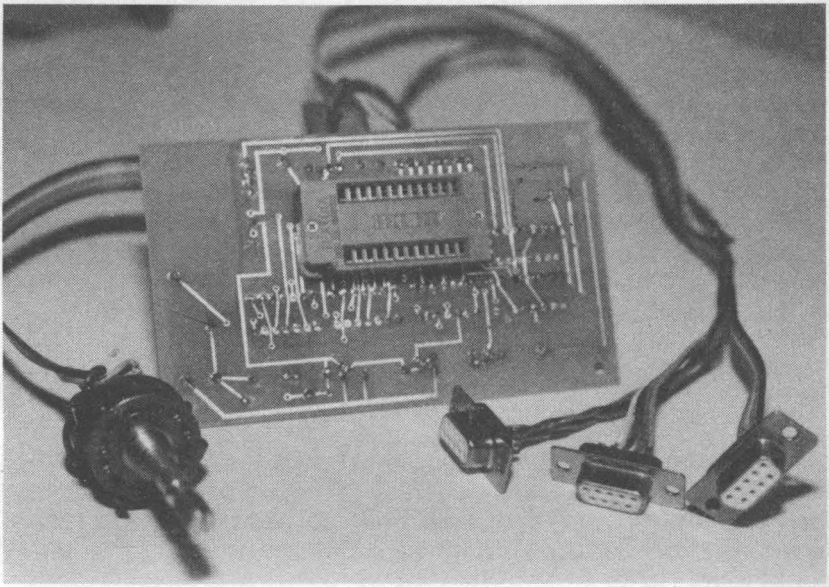


Figure 3

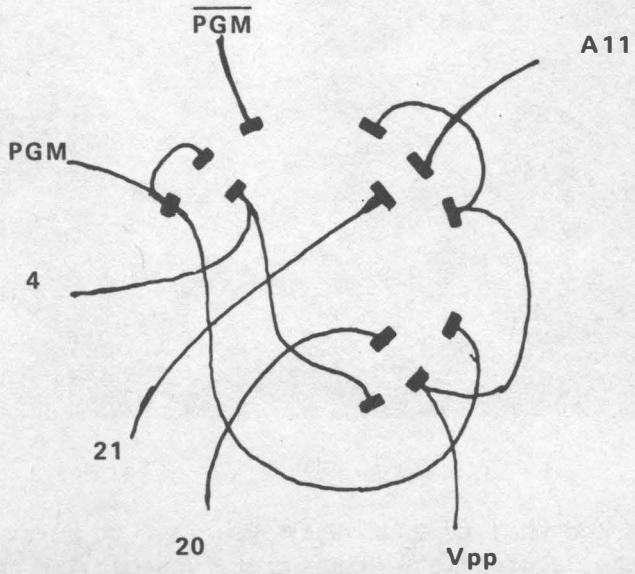
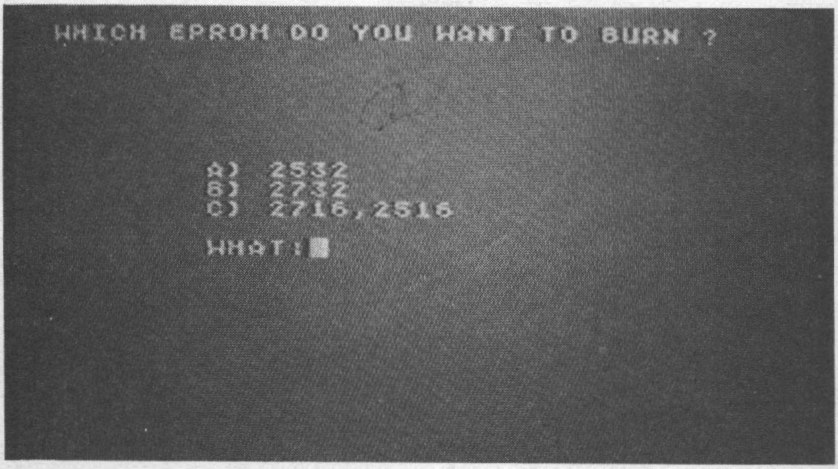


Fig. 4: Rear side of the 3P2T switch

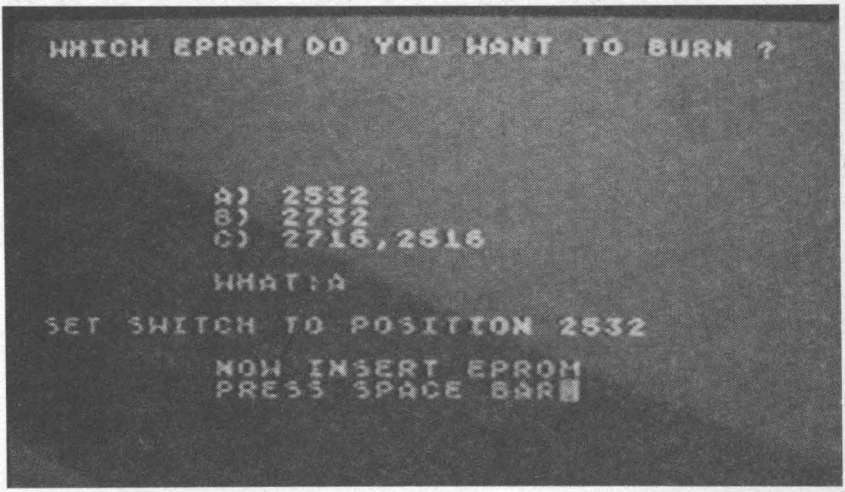
FIG.2 shows the parts layout.It is recomended to use sockets for the integrated circuits. Attention !.The component side for the integrated circuits is the side showing the text EPROMBURNER, but the socket for the EPROM is mounted opposite to this component side. (see FIG. 3) The picture of the burner is shown in FIG. 3. After assembling the board, the connections to the ATARI are made. Use three female plugs and a flatband cable. Last the three pole double throw switch is assembled. The wiring of the switch and the connection to the board is shown in FIG.4.

3) THE SOFTWARE

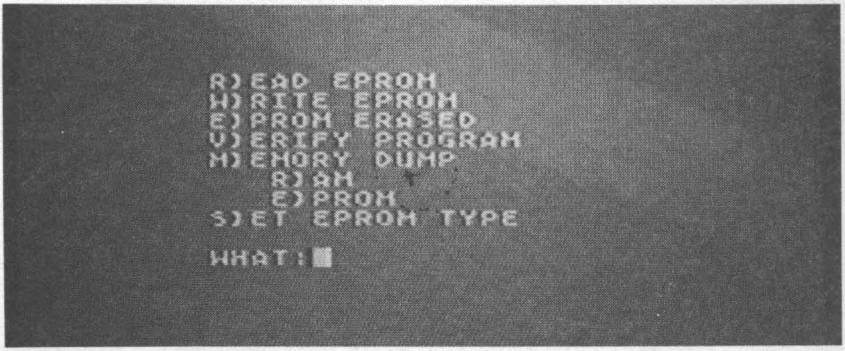
The software for the burner is completely written in machine code. It comes on a bootable diskette. To load the program, insert the disk and REMOVE ALL CARTIDGES. Turn on the disk drive and the ATARI. After a short moment, you will see the first menue:



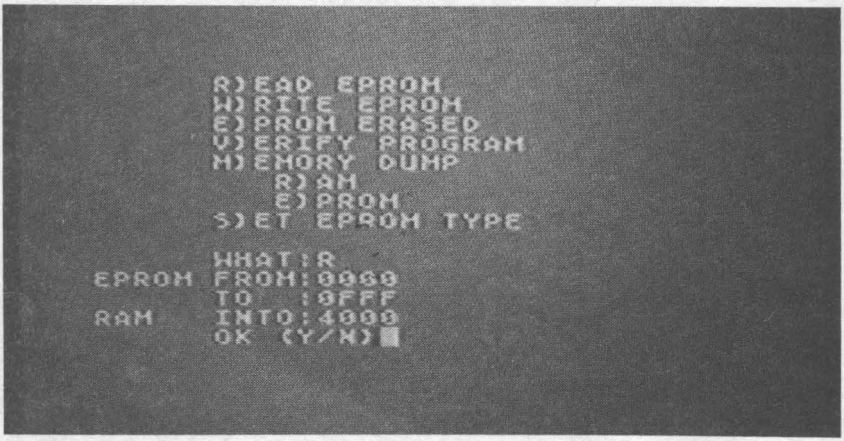
You are asked what type of EPROM you want to burn. After typing the appropriate character, you get the message to set the switch to the correct position and insert the EPROM. This is shown in the following example:



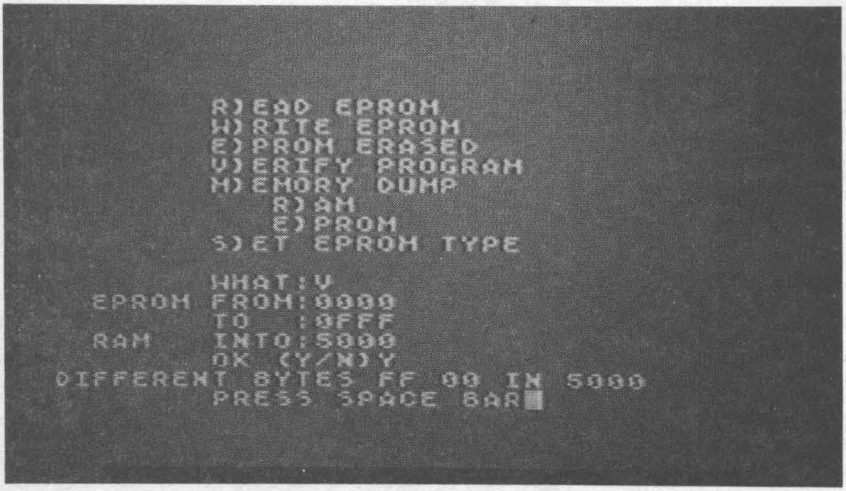
Then, pressing the space bar, you see the main menue:



First we want to R)EAD an EPROM. Type R and then the addresses FROM and TO. The physical addresses of the EPROM are always in range between 0000 and 0FFF. You can read the whole EPROM or only a part of it. Next you have to type the address INTO which the content of the EPROM is read. All adresses which are not used by the system or the burner software (A800 to AFFF) are accessible. By typing Y after the question OK (Y/N), the program is loaded. There is a very important key, the X key. This key cancels the input and leads back into the main menue. An example of reading an EPROM is shown in the next figure:



To verify that the content of the RAM is idetical to the content of the EPROM, type V. After specifing the adresses for the EPROM and the RAM and typing Y, the contents are compared. If there are any differences, you get an error message, such as the following:



You may then make a memory dump. Type M for M)EMORY, either R for R)AM or E for E)PROM, and the address range. There is a slight difference in memory dumps. With the memory dump of RAM, the bytes are printed, if it is possible, as ASCII characters.

Burning an EPROM begins by testing as to whether or not the EPROM is erased in the address range you want to burn. Type E and the address range. You will get the message EPROM ERASED when the assigned address range has been erased, or the message EPROM NOT ERASED IN CELL NNN.
For writing the EPROM, type W, the address range in RAM, and the starting address in EPROM. After hitting Y, you have to wait two minutes for burning 2k and four minutes for burning 4k.Don't get angry, the program will stop. After burning one cell the program does an automatic verify. If there is a difference you recieve the error message EPROM NOT PROGRAMMED IN CELL NNN and the burning stops. Otherwise if all goes well the message EPROM PROGRAMMED is printed.
For changing the type of EPROM you want to burn, type S. The first menue is shown and you can begin a new burning procedure.

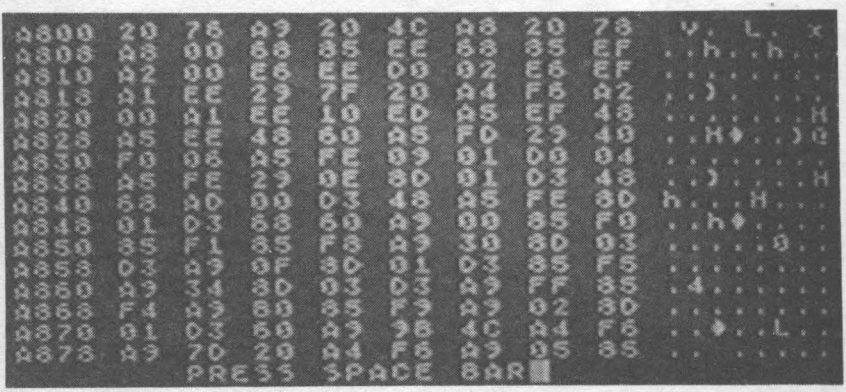
4) PARTS LIST.

IC1	74LS139	
IC2,IC3	4024	
IC4	4016	
IC5	4049	
T1,T2,T3	UNIVERSAL NPN TRANSISTOR	
	30V,0.3W (2N 3390 & 2N3399)	
R1	470 K	RESISTOR
R2,R3	100 K	RESISTOR
R4,R5	33 K	RESISTOR
Z1	1 V	ZENER DIODE
Z2	24 V	ZENER DIODE
M1	DCP528	DCDC CONVERTER
		ELPAC POWER SYSTEMS
C1,C2	100 NF	CAPACITOR

C3	10 MF	TANTAL CAPACITOR
S1	3P2T	SWITCH
1	24 PIN	TEXTOOL SOCKET
3	14 PIN	IC SOCKET
2	16 PIN	IC SOCKET
3	FEMALE PLUGS, ATARI GAME CONNECTORS	

5) STEP BY STEP ASSEMBLING.

1. Insert and solder sockets.
* Component side shows the text EPROMBURNER.
2. Insert and solder resistors.
3. Insert and solder Zener diodes.
* The anodeS are closest to the to the
* transistors.
4. Insert and solder transistors.
5. Insert and solder capacitors.
* The + pole of the tantal is marked.
6. Mount the DCDC converter module.
7. Turn the board to the soldering side.
8. Insert from this side the TEXTOOL socket.
* The knob should be in the
* upper right corner. Solder the socket.
9. Make the connections on the switch. (FIG.4)
* Connect switch and board via
* a 7 lead flatband cable.
10. Connect the plugs to the board. (FIG.5)
11. Insert the integrated circuits.(FIG.2)
12. Turn off the ATARI. Insert the plugs.
* Insert the diskette and turn on the ATARI.



The burner software also allows you a memory dump (RAM or EPROM).

ATEXT-1 WRITERS

REFERENCE CARD

cH means CTRL H; press both the CTRL and the character key.

ERASING

ONE CHARACTER	<--cH	cU-->
ONE WORD	<--cN	
ONE LINE	<--cX	cZ-->
UNTIL FORMAT COMMAND	<--cV	cC-->

MOVING THE CURSOR

ONE CHARACTER	<--c+	c*-->
ONE WORD	<--cO	cP-->
ONE LINE	<--c-	c=-->
BEGIN OF TEXT	<--cW	
END OF TEXT		cS-->
FORWARD NEXT TAB		cI-->
FORWARD NEXT FORMAT COMMAND		cA-->

THE COPY BUFFER

OPENING	cE
CLOSING	cD
INSERTING	cJ
ERASING	cK

Entering text into copy buffer by cursor movement or erasing.

THE COMMAND LINE

OPENING	ESC
CLOSING	ESC
EXECUTING	ESC
REPEATING	cG

ESC means the ESC key. All commands are executed from the current cursor position towards the end of the text.

S EARCH ESC SAB ESC ESC
searches for string AB.

C HANGE

ESC CAB ESC BA ESC ESC
changes string AB to BA.

E RASE

ESC EAB ESC ESC
erases the next string AS.

I NSERT

ESC IHELLO ESC ESC
inserts the string HELLO at the current cursor position.

THE COMMAND LINE cntd.

@n n=1..9 SETTAB

ESC @5 ESC ESC
sets TAB to 5.

H EX Inserts control characters.

ESC H2F ESC ESC
inserts the byte 2F into the text. This command is used to send control characters to the printer.

L IST Sends text to the specified device.

ESC L ESC ESC	first 38th character to screen
ESC L 2ESC ESC	from 38th character to EOL to screen
ESC LE: ESC ESC	To screen
ESC LP: ESC ESC	To parallel printer
ESC LR: ESC ESC	To ELCOMP RS232 interface

ESC LD:NAME ESC ESC
To disc as file NAME. No format commands are saved on disk.

R EAD Reads text from the specified device.

ESC RD:NAME ESC ESC
Reads file NAME from disk
ESC RC: ESC ESC
Reads next file from cassette

W RITE Writes to specified device

ESC WD:NAME ESC ESC
Writes file NAME to disk

ESC WC: ESC ESC
Writes text to the cassette

Attention! All writing starts at the current cursor position.
A file is read in at the current cursor position.

K ILL

ESC K ESC ESC
erases the text buffer

D ELETE

ESC D ESC D ESC ESC
deletes two characters backward

T

ESC T ESC ESC
erases the next line

F

ESC F ESC ESC
moves the cursor one character foreward

J

ESC J ESC ESC
moves the cursor to the next format command

G

ESC G ESC ESC
inserts the copy buffer at cusor position

M

ESC M ESC ESC
exit to DOS

Restart from DOS:
K Coldstart
E Warmstart

THE FORMAT COMMANDS

Every format command begins in a new line, starts with cL and ends with the RETURN key.

Ln sets left margin to n
Rn sets right margin to n
Dn insert n linefeeds

Example:
cLL10R66D2 sets left margin to 10, right margin to 66 and inserts 2 linefeeds.

En insert n blanks at each new paragraph.
Sn sets lines per page.
#n sets form length
@n sets form width

The following commands must be set to T RUE or F ASLSE.

A automatic line feed after each paragraph.
C center next lines
F right margin justification
N n page numbering starts with n

OTHER COMMANDS:

Wc	write control characters to printer
INAME	include file NAME
O	sets the default values
P	formfeed
H	stop formatting (printing) and wait for pressing "OPTION" or "SELECT" to continue.

Order-No. 7214 cassette version $19.95
Order-No. 7215 disk version $24.95

INVENTORY CONTROL

This inventory control program is for ATARI 400/800. The number of items this program can handle depends on the memory size of your computer.

This program comes up with the menu on which you select the various options by pressing one of the number keys. There is no need to press the RETURN key.

1 & 6: Read or Store Data

You can read or store data on disk or cassette. When this routine ends, the machine displays the menu again.

2: Define the Items

You get a list printed on the screen on which you can put your information. Each entry has the following information:

Inventory number	(IN-NO)
Manufacturer	(MANUF)
Recorder level	(RECL)
Present level	(PL)
Code number	(COD)
Description	(DESCRIPT)

Into each entry, you can record alphanumeric characters except the X key. It is not allowed, because it is the exit function. The input of the recorder level and the present level must be numeric.

3: Entry Editing

You can make changes using the cursor functions. A SHIFT DELETE, deletes the article.

4: Inventory Maintenance

In this mode you can keep track of the merchandise sold and received. The changes are displayed as soon as they are made.

5: Report

You can get a printout of all items in a various description, like recorder level, and item number etc.

USING THE KEYBOARD

You can use the cursor functions of the ATARI as normal. CTRL← and CTRL→ moves the cursor to the right or to the left without changing the text. During option 3, the cursor can also be moved up and down.

TAB key sets the cursor to the next row. RETURN sets the cursor in the last free field of the entry (waiting position). X-key leaves the current option.

Hitting the return key while in waiting position leads to the next entry. SHIFT DELETE erases the line. During option 3, this line must not be rewritten.

The ESC key moves the next block of information into the screen. Special hint: Erroneous input is as far as possible ignored. But don't use the CAPS LOWR and the inverse key.

MAILING LIST FOR THE ATARI - 800

Order-No. 7212	cassette version	$19.95
Order-No. 7213	disk version	$24.95

Program Description:
The following mailing list program can handle as many addresses as you want. Fifty addresses each are put together to one block and then stored on disk. The addresses can be printed on the screen of your ATARI or printer as lists or as labels. (The labels are formatted two in a row) beside one another. To run this mailing list, the user needs 16K byte of usable RAM. Please note that there is also space for the DOS available.

How to Start the Program:
Insert the disk into drive one and type in, RUN"D:ADRESS" Now the program automaticly comes up with a menu on the screen. Now you can make your selection of the appropriate function.

Input:
For your input of data, you will get a form you can easily fill out. Name, street, state, zip, city, phone number, and also a short note can be filled in. This program allows only a certain amount of characters for each single entry. (Name:25 char. street:25 char., state:3 char., zip:8 char., city:20 char., phone:15 char., note:25 char.). If you want to put in more then the characters allowed by the program, the cursor is locked. To make changes and corrections, you can use edit functions already built in BASIC. If you type RETURN, the current line will be stored in the computer. Lower case characters are possible. But when the menu appears, the computer switches to the upper case mode. With "*" you can exit this mode again... NOTE: You cannot use the character * for input in your address text.
If you type space instead of a name, you can not access to that later. Therefore you also cannot delete it later. "Be carefull with the space bar"!

Changing addresses:
The program asks for the name you want to change. You type this name in and hit RETURN. If the name is in the computer, it will appear on the screen. If not there will be an error message. If the name is in the computer, you can make changes like, under function 1, INPUT in the menu described before. With RETURN you can go to the next line for input. You have to go to each line with the cursor, even to the note line to complete the changing process. If you don't do so, the change will not be accepted.

Delete Address:
After you select this part of the menu, you will be asked for the name you want to delete. Now you type in that name and "Y" (YES) for delete. All other input will keep the name in the mailing list.

Output Data:
Data output can be done in different ways. An address list, single addresses, addresses from a certain city, or a certain parameter can be outputed to the screen or to the printer. The print out could be a listing (address list, or on lables, two across). Though the lable output prints two lables across, sometimes the program asks for another name. If there are nor other or second name, you have to type in a dummy address.

How to Open a File:
Before doing anything with this program, you have to open a file. The computer will remind you in case you forget it. Eight characters are allowed for file names. (Capital letters only - no graphics). One file allows you to store 50 addresses. The menu shows you the name of the file, which is accessed or handled at that time. The menu also tells you how many addresses can still be stored.

How to Access a File:
This command allows you to work with a file already filled with names and so on.

Store Data in a File:
To store data before switching off the computer or going to another file, you have to save the address information on disk. Please check if there is any room left on the diskette before you attempt to save a file.

NOTE: This program is protected against any wrong input. If you hit the asterik "*" you leave the current mode and return to the menu.

The BREAK-KEY is disabled.

Elcomp Publishing, Inc.
53 Redrock Lane
Pomona, CA 91766
(714) 623-8314

MAILING LIST

The program can control as many addresses as you want. Fifty addresses at a time are put together to a data-file and stored. The menu at the beginning gives you the following functions:

1.) Enter data:
You get a list with name, street, state, zip code, city, telephone, note, and can enter the matching data. The length of the input is limited. If you try to enter too much, the cursor stops. For corrections, you can use the normal control functions. After you hit RETURN the line is stored. If you enter "*" you will get back to the menu.

2.) Change data:
The program will ask you for the name to be changed. If the name exists, the address will be displayed and you can continue like under number one. If you hit RETURN, you go to the next line. You have to enter all lines!

3.) Delete addresses:
The program will ask you for the name to be deleted. If you enter "Y" the address will be deleted.

4.) Output data:
You have the following options here: Total list, single names, all addresses of one town, search for special note. You can direct the output to the screen or to the printer. On the printer you can get a normal printout or labels (two addresses in a line).

5.) Create a file:
Before you enter datas you have to name a file. In one file there is space for fifty addresses.

6.) Read data:
This command reads data out of existing files.

7.) Store data:
This command is used to save datas.

This program requires the BASIC cartridge in the left slot.

Order-No. 7201	cassette version	$29.95
Order-No. 7200	disk version	$39.95

Invoice Writer for ATARI-400/800®

Invoice Writer for ATARI-400/800

The ATARI-400/800 are powerful consumer computers for games and education. But the very small business man also can use them for cutting costs.

The invoice program from ELCOMP is very simple but does a great job in writing your invoices. You need the following hardware:

1. ATARI 400 or 800 with at least 16k of RAM
2. ATARI BASIC in ROM
3. interface module ATARI 850
4. printer with serial (RS 232) interface

This program writes invoices for the very small business. The invoice is written onto a special form that is supplied in limited quantities with the program. An address label is included in this form.

Program description:

Lines 222, 224, 226 and lines 1000, 1002, 1004 contain the address of your company. You can replace these three strings by your company' s address.

Lines 360, 370, 380 calculate the discount. S1 is the number of products sold to that customer. If S1 is 1....5, then discount is 25%, if S1 is G....10, then discount is 33 %, if S1 is 11 or more, then discount is 40 %. You can change these numbers easily e.g., if you want 33% discount for 6....20 products and 50% for more than 20 products change line 370 to:

IF S1 ⟩ 5) AND (S1 ⟨ 21) THEN D=0.33

and line 380 to:

IF S1 ⟩ 20 THEN D=0.5

Lines 405, 410, 420, 430, 440 calculate the costs for shipping:

If S1 is 1....14, shipping costs (V) is $2.—, if S1 is 15...29, then V=$1.25, if S1 is 30....39, then V=$1.50, if S1 is 50....99, then V= $2.—, if S1 is 100 or more, then V= $2.50. This can also be changed, if needed.

At the end of the program (lines 1500...) you can enter your product file. Every record of this file is written in a DATA-statement and contains:

1. order number
2. description (up to 26 characters)
3. price

e.g. 1600 DATA 117,50 BLANK CASSETTES C10,19.80

Important: The last statement has to contain 0,0,0 (end of file marker).

Program handling:
Start the program with RUN. The first thing you have to enter is the TAX RATE, next is the DATA (you have to enter it only once at the beginning), then FIRST INVOICE NUMBER (will be incremented at every invoice), then ACCOUNT NUMBER, then DISCOUNT, yes or no, if you enter 1 for yes, discount is calculated depending on the number of products sold, if you enter 0 for no, discount is 0%.

The next you have to enter is SHIPPING, here you have a choice of automatic calculation of shipping costs, or you can enter a certain amount by hand.

Next thing to enter is CUSTOMERS ORDER NUMBER and his address, next you can select the way of payment and the way of transportation. Then SHIP TO SAME ADDRESS will appear, you can enter Y or another address. After that the head of the invoice will be printed.

The next thing you have to enter is the first item number. If the program reads the end of the file, you will get ITEM NUMBER NOT FOUND and you can enter another number.

When you want to finish the invoice, enter 0 and the program will finish that invoice.
Now you can enter the next account number and so on.

GUNFIGHT

Order No. 7207 $ 19.95

GUNFIGHT is for two players. Each player uses one joystick. Plug the joystick in port 1 and 2. Port 1 refers to the left player, port 2 to the right player.
GUNFIGHT starts with the Copyright-statement. Press one of the two trigger buttons and the playfield appears. Now you can move the cowboys.

The Game
You shoot by pressing the trigger. Each player has 10 cowboys and each cowboy 50 bullets. By holding the button pressed the gun keeps firing until the cowboy has no more bullets. There are 2 ways to be killed: Your opponent can shoot you or you can drown in the sea.
After each round the game waits for you to press the trigger. The game-status appears. It shows you the actual number of cowboys and bullets. When you press the trigger the game goes on with the next round. The winner of the game will be shown after you press the trigger button. Pressing the trigger the game starts again with the Copyright-statement.

The Sea
Don' t touch this blue area because you will be drowned.

If you run out of bullets:
If you don' t have any bullets you can only flee and hope that the enemy will touch the sea.

You can hide your cowboy behind any one of the three rocks. But when your opponent also hides behind a rock, he can shoot you as long as his gun is behind the rock.

The program starts at hex 1000. To key the program in, you need a machine language monitor or the ATARI® Editor/Assembler cartridge.
The starting address of the program GUNFIGHT is 1016 hex.
Start with GOTO 1016. Skip the location between 175F and 1F98.

1F98 = Start of display list
2000 = End of display list and beginning of the background. Type in everything like it is. (Use the fill-command of a machine language monitor).
2FFF End of background

If you bought the program on cassette,you can load the program as follows:
1.Switch the computer off
2.Switch the TV an cassette recorder on
3. Press the START button
4. Hold the START button down
5. Switch the computer on
6. Release the START button
7. Press PLAY on the recorder after the beep
8. Press RETURN

KNAUS-OGINO Birthcontrol

KNAUS OGINO

For thousands of years now man thinks about birth-control. The present mean, the 'pill' was developed in the 1950s. All kinds of ways of birth-control have been developed, from droppings of a crocodile as a pessar (1850 B. C.) to abstinence (1st book of Mose, Chapter 38, verse 9).

A much better method is the one developed in 1930 by two independent doctors. H. Knaus (Austria) and D. Ogino (Japan) acquired this technique on the basis of the female cycle. A summary can be found in the Family Encyclic by pope Pius XI. (1931). There it says : The first fertile day can be calculated by subtracting 19 days from the shortest cycle known. The last fertile day can be calculated by subtracting 10 days from the number of days of the longest cycle known by that particular woman.

Since this sounds like an estimate this technique didn't disseminate very well. Nowadays we know that a fertilization is possible only a short time after the expulsion of the ovum. With a healthy woman this time is about six hours. We also know, since Knaus and Ogino, that the expulsion of the ovum happens pretty exactly 14 days before the next menstruation. Since the life-span of the sperm is about two days the time, where a fertilization can happen is elongated by that period.

If we use the results of the two scientists together with statistics and error-mathematics we are able to make more accurate statements.

The only moment known is the start of the menstruation (date and time of day). We subtract 14 days from that moment and get the expulsion of the ovum. Plus one and minus three days therefrom tells the time of abstinence. If we add 266 days to the time of the expulsion of the ovum we get the date of birth (in case of a fertilization).

The program which comes with this article does all the calculations and also saves all previous dates. Every month you load the old dates and add the new one. The reliability becomes better and better with every new date.

Finally a note :
All informations are without any guarantee !

Order-No. 7222 $29.95

PRODUCTS FOR ATARI* 400/800 FROM E L C O M P

BOOKS:

ATARI BASIC - Learning by using
An excellent book for the beginner. Many short programs and learning exercises. All important features of the ATARI computers are described (screen drawings, special sounds, keys, paddles, joysticks, specialized screen routines, graphics, sound applications, peeks, pokes, and special stuff). Also suggestions are made that challenge you to change and write program routines.

Order # 164 $7.95

GAMES for the ATARI Computer
This book describes advanced programming techniques like player-missile-graphics and use of the hardware-registers. Contains many ready to run programs in BASIC and one called GUNFIGHT in machine-language.

Order # 162 $7.95

SOFTWARE IN BASIC:

Invoice Writing for Small Business
This program makes writing invoices easy. Store your products in DATA statements with order-number, description, and price. The program later retrives the description and price matching to the entered order-number. The shipping cost and the discount may be calculated automatically depending on the quantity ordered or entered manually. The description to the program tells you how to change the program and adapt it to your own needs. Comes with a couple of invoice forms to write your first invoices onto it.

Order # 7201 cassette version $29.95
Order # 7200 disk version $39.95

Mailing List
This menu driven program allows the small business man to keep track of vendors and customers. You can search for a name or address of a certain town or for an address with a certain note. 50 addresses are put into one file.

Order # 7212 cassette version $19.95
Order # 7213 disk version $24.95

Inventory Control
This program is menu driven. It gives you the following options: read/store data, define items, entry editing, inventory maintenance (incoming-outgoing), reports. The products are stored with inventory number, manufacturer, recorder level, present level, code number, description.

Order # 7214 cassette version $19.95
Order # 7215 disk version $24.95

Programs from Book # 164
The programs from book no. 164 on cassette. (Book included)

Order # 7100 $29.00

Game Package
Games on cassette. (Bomber, tennis, smart, cannon fodder, etc).

Order # 7216 $9.95

SOFTWARE IN MACHINE LANGUAGE:

ATMONA-1
This is a machine language monitor that provides you with the most important commands for programming in machine-language. Disassemble, dump (hex and ASCII), change memory location, block transfer, fill memory block, save and load machine-language programs, start programs. Printer option via three different interfaces.

Order # 7022 cassette version $19.95
Order # 7023 disk version $24.95
Order # 7024 cartridge version $59.00

ATMONA-2
This is a tracer (debugger) that lets you explore the ATARI RAM/ROM area. You can stop at previously selected address, opcode, or operand. Also very valuable in understanding the microprocessor. At each stop, all registers of the CPU may be changed. Includes ATMONA-1.

Order # 7049 cassette version $49.95
Order # 7050 disk version $54.00

ATMAS
Macro-Assembler for ATARI-800/48k. One of the most powerful editor assemblers on the market. Versatile editor with scrolling. Up to 17k of source-code. Very fast, translates 5k source-code in about 5 seconds. Source code can be saved on disk or cassette. (Includes ATMONA-1)

Order # 7099 disk version $89.00
Order # 7999 cartridge version $129.00

ATAS
Same as ATMAS but without macro-capability. Cassette-based.

Order # 7098 32k RAM $49.95
Order # 7998 48k RAM $49.95

ATEXT-1
This wordprocessor is an excellent buy for your money. It features screen oriented editing, scrolling, string search (even nested), left and right margin justification. Over 30 commands. Text can be saved on disk or cassette.

Order # 7210 cassette version $29.95
Order # 7216 disk version $34.95
Order # 7217 cartridge version $69.00

GUNFIGHT
This game (8k machine-language) needs two joysticks. Animation and sound. Two cowboys fight against each other other. Comes on a bootable cassette.

Order # 7207 $19.95

HARDWARE:

PRINTER INTERFACE
This construction article comes with printed circuit board and software. You can use the EPSON printer without the ATARI printer interface. (Works with gameports 3 and 4).

Order # 7211 $19.95

EPROM BURNER (2516 or 2532 EPROMs)
Works with gameports. No additional power supply needed. Comes completely assembled and with software.

Order # 7042 $249.00

EPROM BOARD (CARTRIDGE)
Holds two 4k EPROMS (2532).

Order # 7043 $29.95

ELCOMP Publishing Inc., 53 Redrock Lane
Pomona CA 91766, Phone: (714) 623-8314

Payment: check, money order, VISA, MASTERCHARGE, Eurocheck.
Orders from outside USA: add 15% shipping. CA residents add 6.5% tax
* ATARI is a registered trademark of Atari Inc.

NOTES

NOTES

NOTES

NOTES

HOFACKER

Books + Software for ATARI VIC-20 OSI SINCLAIR TIMEX

Payment: check, money order, VISA, MASTER CHARGE, Eurocheck
Orders from outside USA: add 15% shipping. CA residents add 6.5% tax
*ATARI is a registered trademark of ATARI Inc.
*VIC 20 is a registered trademark of Commodore

ELCOMP PUBLISHING, INC
53 Redrock Lane
Pomona, CA 91766
Phone: (714) 623 8314

PRODUCTS FOR ATARI* 400/800 FROM ELCOMP

BOOKS:

ATARI BASIC – Learning by Using
An excellent book for the beginner. Many short programs and learning exercises. All important features of the ATARI computers are described (screen drawings, special sounds, keys, paddles, joysticks, specialized screen routines,graphics, sound applications, peeks, pokes, and special stuff). Also suggestions are made that challenge you to change and write program routines.
Order #164 **$7.95**

Games for the ATARIComputer
This book describes advanced programming techniques like player-missile-graphics and use of the hardware-registers. Contains many ready to run programs in BASIC and one called GUNFIGHT in machine language.
Order #162 **$7.95**

Programming in 6502 Machine Language on your PET+CBM
2 complete Editor/Assemblers (Source code 3 hexdump + description plus a powerful machine language monitor (Hexdump)).
Order #166 **$19.95**

How to program your ATARI in 6502 machine language
Introduction to machine language for the BASIC programmer
Order #169 **$9.95**

SOFTWARE IN BASIC FOR ATARI

Invoice Writing for Small Business
This program makes writing invoices easy. Store your products in DATA statements with order-number, description, and price. The program later retrives the description and price matching to the entered order-number. The shipping cost and the discount may be calculated automatically depending on the quantity ordered or entered manually. The description to the program tells you how to change the program and adapt it to your own needs. Comes with a couple of invoice forms to write your first invoices on to it.
Order #7201 **cassette version** **$29.95**
Order #7200 **disk version** **$39.95**

Mailing List
This menu driven program allows the small business man to keep track of vendors and customers. You can search for a name or address of a certain town or for an address with a certain note. 50 addresses are put into one file.
Order #7212 **cassette version** **$19.95**
Order #7213 **disk version** **$24.95**

Inventory Control
This program is menu driven. It gives you the following options: read/store data, define items, entry editing, inventory maintenance (incoming-outgoing), reports. The products are stored with inventory number, manufacturer, reorder level, present level, code number, description.
Order #7214 **cassette version** **$19.95**
Order #7215 **disk version** **$24.95**

Programs from Book # 164
The programs from book no. 164 on cassette. (Book included)
Order #7100 **$29.00**

GamePackage
Games on cassette. (Bomber, tennis, smart, cannon fodder, etc.)
Order #7216 **$9.95**

Microcomputer Hardware Handbook (845 pages)
Descriptions, pinouts and specifications of the most popular microprocessors and support chips.
A MUST for the hardwarebuff.
Order-No. 29
$14.95

Care and Feeding of the Commodore PET
Eight chapters exploring PET hardware. Includes repair and interfacing information. Programming tricks and schematics.
Order #150 **$9.95**

SOFTWARE IN MACHINE LANGUAGE for ATARI

ATMONA-1
This is a machine language monitor that provides you with the most important commands for programming in machine-language. Disassemble, dump (hex and ASCII), change memory location, block transfer, fill memory block, save and load machine-language programs, start programs. Printer option via three different interfaces.
Order #7022 **cassette version** **$19.95**
Order #7023 **disk version** **$24.95**
Order #7024 **cartridge version** **$59,00**

ATMONA-2
This is a tracer (debugger) that lets you explore the ATARI RAM/ROM area. You can stop at previously selected address, opcode, or operand. Also very valuable in understanding the microprocessor. At each stop, all registers of the CPU may be changed. Includes ATMONA-1.
Order #7049 **cassette version** **$49.95**
Order #7050 **disk version** **$54.00**

ATMAS
Macro-Assembler for ATARI-800/48k. One of the most powerful editor assemblers on the market. Versatile editor with scrolling. Up to 17k of source-Code. Very fast, translates 5k source-code in about 5 seconds. Source code can be saved on disk or cassette. (Includes ATMONA-1)
Order #7099 **disk version** **$89.00**
Order #7999 **cartridge version** **$129.00**

ATAS
Same as ATMAS but without macro-capability. Cassette-based.
Order #7098 **32k RAM** **$49.95**
Order #7998 **48k RAM** **$49.95**

ATEXT-1
This wordprocessor is an excellent buy for your money. It features screen oriented editing, scrolling, string search (even nested), left and right margin justification. Over 30 commands. Text can be saved on disk or cassette.
Order #7210 **cassette version** **$29.95**
Order #7216 **disk version** **$34.95**
Order #7217 **cartridge version** **$69.00**

GUNFIGHT
This game (8k machine-language) needs two joysticks. Animation and sound. Two cowboys fight against each other. Comes on a bootable cassette.
Order #7207 **$19.95**

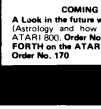
FORTH for the ATARI

FORTH from Elcomp Publishing, Inc. is an extended Fig-Forth-version, Editor and I/O package included. Utility package includes decompiler, sector copy,Hex-dump (ASCII), ATARI Filehandling, total graphic and sound, joystick program and player missile. Extremely powerful!
Order #7055 **disk** **$39.95**

Floating point package with trigonometric functions (0 – 90°).
Order #7230 **disk** **$29.95**

Learn-FORTH from Elcomp Publishing, Inc.
A subset of Fig-Forth for the beginner. On disk (32k RAM) or on cassette (16k RAM).
Order #7053 **$19.95**

Expansion boards for the APPLE II



The Custom Apple + Other Mysteries
A complete guide to customizing the Apple Software und Hardware
Order-No. 680 **$24.95**

We also stock the boards which are used in the book "The Custom Apple ..." (barebords)
6522 I/O Board No. 605 **$39.00**
EPROM Burner No. 607 **$49.00**
8K EPROM/RAM Board No. 609 **$29.00**
Prototyping board for the Apple II No. 604 **$29.00**
Slot repeater board for the Apple II No. 606 **$49.00**
Order two boards and get the book free !

COMING SOON ! ORDER NOW !

A Look in the future with your ATARI
(Astrology and how to do your own horoscope on the ATARI 800. **Order No. 171** **$9.95**
FORTH on the ATARI – Learning by Using
Order No. 170 **$7.95**

Hardware – ADD–ONS for ATARI

PRINTER INTERFACE
This construction article comes with printed circuit board and software. You can use the EPSON printer without the ATARI printer interface. (Works with gameports 3 and 4).
Order #7211 **$19.95**

RS-232 Interface for your ATARI 400/800
Software with connector and construction article.
Order #7291 **$19.95**

EPROM BURNER for ATARI 400/800
Works with gameports. No additional power supply needed. Comes compl. assembled with software (2716, 2732, 2532).
Order #7042 **$179.00**

EPROM BURNER for ATARI 400/800 KIT
Printed circuit board incl. Software and extensive construction article.
Order #7292 **$49.00**

EPROM BOARD (CARTRIDGE)
Holds two 4k EPROMs (2532). EPROMs not included.
Order #7043 **$29.95**

EPROM BOARD KIT
Same as above but bare board only with description.
Order #7224 **$14.95**

ATARI, VIC 20, Sinclair, Timex and OSI

New – for your ATARI 400/800
Astrology and Biorythm for ATARI (cass. or disk).
Order #7223 $29.95
Birth control with the ATARI (Knaus Ogino)
Order #7222 cass. or disk $29.95

Books + Software for VIC-20 (requires3k RAM Exp.)
#4870 Wordprocessor for VIC-20, 8k RAM $19.95
#4883 Mailing List for VIC-20, 16k RAM $14.95
#141 Tricks for VICs - The VICstoryProgr. $9.95
#4880 TIC TAC VIC $9.95
#4881 GAMEPACK I (3 Games) $14.95
#4885 Dual Joystick Instruction $9.95

INPUT/OUTPUT Programming with your VIC
Order #4886 $9.95
#4896 Miniassembler for VIC-20 $19.95
#4881 Tennis, Squash, Break $9.95
#4894 Runfill for VIC $9.95

Universal Experimenter Board for the VIC-20
(Save money with this great board). This board plugs right into the expansion slot of the VIC-20. The board contains a large prototyping area for your own circuit design and expansion. The construction article shows you how to build your own 3k RAM expander and ROM-board.
Order #4844 **$18.95**

Software for SINCLAIR ZX-81 and TIMEX 1000
#2399 Machine Language Monitor $9.95
#2398 Mailing List $19.95

Programming in BASIC and machine language with the ZX-81 (82) or TIMEX 1000.
Order #140 (book) $9.95

Books for OSI
#157 The First Book of Ohio $7.95
#158 The Second Book of Ohio $7.95
#159 The Third Book of Ohio $7.95
#160 The Fourth Book of Ohio $7.95
#161 The Fifth Book of Ohio $7.95
151 8K Microsoft BASIC Ref. Man. $9.95
152 Expansion Handbook for 6502 and 6802 $9.95
153 Microcomputer Appl. Notes $9.95

Complex Sound Generation
New revised applications manual for the Texas Instruments SN 76477 Complex Sound Generator.
Order # 154 **$6.95**

Small Business Programs Order #156
Complete listings for the business user. Inventory, Invoice Writing, Mailing List and much more. Introduction to Business Applications. **$14.90**


HOFACKER
HOFACKER
HOFACKER
HOFACKER
HOFACKER
HOFACKER
HOFACKER
HOFACKER


DESCRIPTION BOOK
SAM D. ROBERTS